**GIT**

**The version control system**

[**https://www.tutorialspoint.com/git/git\_basic\_concepts.htm**](https://www.tutorialspoint.com/git/git_basic_concepts.htm)

**About Version Control:**

What is “version control”, and why should we care? Version Control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Version Control System (VCS) is a software that helps software developers to work together and maintain a complete history of their work.

Below are the functions of VCS:

* Allows developers to work simultaneously.
* Does not allow overwriting each other’s changes.
* Maintains a history of every version.

Following are the types of VCS:

* Centralized version control system (CVCS).
* Distributed/Decentralized version control system (DVCS).

**Distributed Version Control Systems:**

Centralized version control system uses a central server to store all files and enables team collaboration. But the major drawback of CVCS is its single point of failure, i.e., failure of the central server. Unfortunately, if the central server goes down for an hour, then during that hour, no one can collaborate at all. And even in a worst case, if the disk of the central server gets corrupted and proper backup has not been taken, then you will lose the entire history of the project. Here, DVCS comes into picture.

DVCS clients not only check out the latest snapshot of the directory but they also fully mirror the repository. If the server is down, then the repository from any client can be copied back to the server to restore it. Every checkout is a full backup of the repository. Git doesn’t rely on central server and that is why you can perform many operations when you are offline. You can commit changes, create branches, view logs, and perform other operations when you are offline. You require network connection only to publish your changes and take the latest changes.

**Advantages of Git:**

**Free and open source**

Git is released under GPL’s open source license. It is available freely over the internet. You can use Git to manage property projects without paying a single penny. As it is an open source, you can download its source code and also perform changes according to the requirement.

**Fast and small:**

As most of the operations are performed locally, it gives a huge benefit in term of speed. Git does not rely on the central server; that is why, there is no need to interact with the remote server for every operation. The core part of Git is written in C, which avoids runtime overheads associated with other high level languages.

**DVCS Terminologies:**

**Local Repository:**

Every VCS tool provides a private workplace as a working copy. Developers make changes in their private workplace and after commit, these changes become a part of the repository. Git takes it one step further by providing them a private copy of the whole repository. Users can perform many operations with this repository such as add file, remove file, rename file, move file, commit changes, and many more.

**Working Directory and Staging Area or Index:**

The working directory is the place where files are checked out. In other CVCS, developers generally make modifications and commit their changes directly to the repository. But Git uses a different strategy. Git doesn’t track each and every modified file. Whenever you do commit an operation, Git looks for the files present in the staging area. Only those files present in the staging are considered for commit and not at all the modified files.

**Commits:**

Commit holds the current state of the repository. A commit is also named by SHA1 hash. You can consider a commit object as a node of the linked list.

**Branches:**

Branches are used to create another line of development. Bu default, Git has a master branch, which is same as trunk in subversion. Usually, a branch is created to work on a new feature is completed, it is merged back with the master branch and we delete the branch. Every branch is referenced by HEAD, which points to the latest commit in the branch. Whenever you make a commit, HEAD is updated with the latest commit.

**Tags:**

Tags assign a meaningful name with specific version in the repository. Tags are very similar to branches, but the difference is that tags are immutable. It means, tag is a branch, which nobody intends to modify. Once a tag is created for a particular commit, even if you create a new commit, it will not be updated. Usually, developers create the tags for product releases.

**Clone:**

Clone operation created the instance of the repository. Clone operation not only checks out the working copy, but it also mirrors the complete repository. Users can perform many operations with this local repository. The only time networking gets involved is when repository instances are being synchronized.

**Pull:**

Pull operations copies the changes from a remote repository instance to a local one. The pull operation is used for synchronization between 2 repository instances. This is same as the update operation in Subversion.

**Push:**

Push operation copies changes from a local repository instance to a remote one. This is used to store the changes permanently into the Git repository. This is same as the commit operation in Subversion.

**HEAD:**

HEAD is a pointer, which always points to the latest commit in the branch. Whenever you make a commit, HEAD is updated with the latest commit. The heads of the branches are stored in .git/refs/heads/ directory.

**How to install GIT?**

Install GIT for MAC:

<https://www.atlassian.com/git/tutorials/install-git>

ow Howdsfdshjfjkj