What is Git?

Git is an application that keeps track of changes (text changes) that are made and move back and forth to utilize a particular version. It is referred as Version control System and Source code management. Git is similar to undo and redo mechanism.

Other version control systems

* Source Code Control System (SCCS) - Unix only

In this mechanism changes are not saved as whole document. It just saves the snapshots of changes. To retrieve any particular version of document, Snapshot is applied to original document.

Original Doc

Version 2 Doc

Version 1 Doc

Snapshot 1 Snapshot 2

Latest

Document

* Revision Control System (RCS) – Cross platform

It is quite opposite to SCCS, rather than original document snapshots are applied on latest version of document to obtain previous versions of document.

Original Doc

Latest Doc(v2)

Version1 Doc

Snapshot 2 Snapshot 1

Both SCCS and RCS are single file version control systems and one person can access a file at an instance.

* Concurrent Versions System (CVS)

Documents are stored in a remote server and more than one user can access the file concurrently.

* Apache Subversion(SVN)

Also keeps track of non-text such as images. Not only keeps track of multiple files, but also the whole directory by taking the snapshots of the directories. Above systems are not directory oriented systems.

* BitKeeper SCM

Closed source, proprietary, distributed version control.

* Git was developed in April 2005 by Linus Torvalds

Distributed version control, open source and free software, cross-platform.

**Distributed version control:**

Different users maintain their own repositories, instead of working from a central repository. No single master repository but many working copies. No need to communicate with any central server which makes the system faster.

**Git Architecture:**

Git uses three-tree architecture. Repository, Staging index, working directory are the three entities of three-tree architecture.

**Repository**

2. git commit file.txt ↑

**Staging index**

1. git add file.txt ↑

**Working**

Git refers snapshots of changes using hash values (SHA-1). Git generates a 40-characters hexadecimal string checksum for each change set. Snapshots are interlinked using linked list data structure. New commit holds the hash value of previous commit.

**Initializing git**

To start tracking changes of a project we need to initialize git using a command **git init** Git should be initialized in the root directory of the project directory.

$ git init

Initialized empty Git repository in F:/git\_practise/FirstProject/.git/

Git files are stored in the .git directory in the git initialized directory.

.git directory consists of following sub-directories

/HEAD

/branches

/config

/description

/hooks

/info

/objects

/refs

**Committing Changes**

After performing modifications in the source file we need to save changes to the local repository. Before saving the changes to repository, they need to be saved in the staging index. To add files to the staging index we use git add command.

$ git add file.txt

We can add single file to staging index by specifying file name in the git add command and to add all the files to staging index we use .(dot)

$ git add .

Once we are done with adding changes to staging index we then perform saving changes to repository using git commit command.

$ git commit –m “initial commit”

-m option is mandatory while committing a change. –m refers to commit message which is a primary mean of communication to understand about the project.

It is possible to add and commit in a single step, only during modifications not during new file inclusion.

$ git commit –am “add and commit”

**Branching:**

Branch is an independent line of development. By default branch is master. A new branch can be created using command

$ git branch <name>

To list all the branches present in the project we use git branch command.

To switch between the branches we use git checkout command.

$ git checkout -b <name> is used to create and checkout a new branch. We need to commit our changes or stash them in order to switch to other branch.

Changes between the branches can be seen using the following command

$ git diff master..newBranch

$ git branch --merged is used to view all the branches which have commits of the current branch.

Renaming a branch $ git branch –m <oldName> <newName>

Deleting a branch$ git branch –d <branchName> we can’t delete a branch which we are currently on.