# Repository

Folder which stores code inorder to maintain versions/changes made to data

# Types of SCM-

1.Centralised System

2. Distributed System

# Centralized System

1. Centralized location holding the repository
2. Multiple user will access same directory
3. Easy to understand and setup
4. No local copy on User’s system
5. User always need to be online, in-order to make any changes to the code
6. More chances of conflict
7. Slow as need to be online and access central repository every time.
8. E.g. Tortoise SVN

# Distributed System

1. Remote location of repository
2. Local copies of the same code are available on user’s system
3. Once user finished his work, can commit /save changes to local copy and then push to remote repository
4. Less conflicts
5. Faster as accessing local copy of repository
6. Even though server of remote repository is down, code will be available in local copy, Hence doesn’t affect the work
7. E.g. GIT, Mercurial

# Stage Of GIT Workflow

1. Working directory/Workspace –

where code is created in local machine, Only one branch will be active at a time

1. Staging Area-

Once changes are finalised to make final commit, files are put up on staging area. Files can later be reverted from staging to workspace area- (Git add)

1. Local Repository-

Code is committed to local repository with snapshot, commit id is generated as40 alphanumeric code (Git Commit)

1. Remote Repository- push the code to remote repositories, which is committed to local (Git push)

Commits are incremental snapshots -Only changed part is saved for every commit as snapshot (doesn’t save every file for every commit)

# Commit

1. Every change is saved with 40 alphanumeric Commit Id
2. It uses SHA & Checksum concepts to maintain multiple changes
3. Tags are assigned as descriptive message to each commit, to identify commits easily
4. When commit is made, new incremental snapshot is created

# PUSH

Copies changes from local repositories to remote repository permanently.

# PULL

1.Copies changes from remote repositories to local repository, however local changes remain intact.

2.Pull is used to synchronise local and remote repositories

# Branch

1.Default branch main/master

2.Each task is separately worked upon, by creating a separate git branch.

3.Once change is finalised, separated branch can be merged into main branch

4.separating task by creating new branch every time, assured reliable code into main/master branch.

New changes can be developed and tested separately, without affecting main branch

5.Multiple users can work simultaneously on different branches and task, later all can be merged together

# GIT

1.Open Source

2.Fast-as working on distributed concepts

3.Light weight-

4. Secure-uses common cryptography or hash factor (SHA1)to identify commits, or identify defects in database

Git init

Git status

Git log

Git log --oneline

Git add ./git add file name

Git commit –m “commit message”

Git resett commitId

Git stash

Git stash pop

Git stash clear

Git remote add origin url

Git remote -v

Git push origin branchname

Git merge sourcebrach

Git add upstream

(origin-one personal created, upstream-from where forked)

Head is pointer to indicate all commit will be added to branch where head is pointing.