**Notes**

**Version Control Systems**

Records changes to files/projects over time so that you can later

* revert files to a particular previous state
* revert project to a particular previous state
* Compare changes over time
* who modified what over time
* what/who caused an issue
* Easy recovery

**Local VC**

Time-stamped version folders of projects

**Centralized VC**

Single server with all versioned files. Clients check out files from this central server.

Suffers from SPOF (single point of failure)

Also needs a backup which cannot be on the server.

Ex: Apache Subversion

**Distributed VC**

Clients fully mirror the repository.

Each clone is a full backup of all data (provided the clones are up-to-date)

**Requirement of a good DVC**

* Fast
* Simple design
* 1000s of parallel branches (allow non-linear devnt.)
* fully distributed
* ability to handle large projects

**Git**

Git doesnot store the changes made over time. Rather, it stores data as snapshots of the files.

Git thinks about its data as a stream of snapshots.

It is much easier to revert to a previous state since all Git needs to do is use that snapshot.

All operations are local since Git clones the entire repository **including the history**

No need to fetch/ request from central server.

Check-sum of data using SHA-1

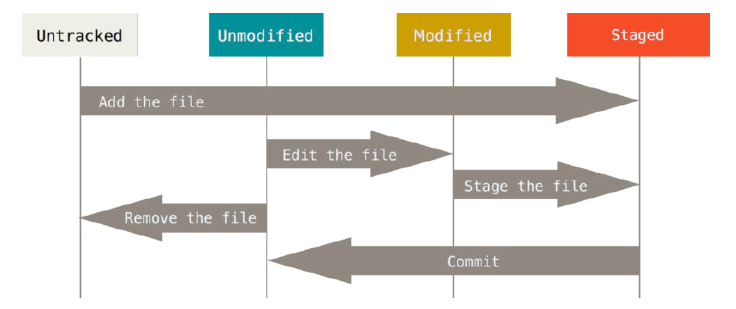
**3** **states** – Committed, Modified, Staged

3 sections of a Git project – Working directory, Staging Area, .git directory (repository)

.git repository is copied when you clone the repo. This has the metadata and the compressed object database

the files of project are pulled out of the compressed database in the .git directory

**Lifecycle of the statuses of files**



Untracked files are those files that are not in any of the previous snapshots (i.e. new files from **git’s perspective)**