**Git Theory || Install || Setup**

## **Version Control System**

* 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.
* It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more
* **Centralized Version Control Systems [CVS, Subversion, and Perforce]**
* **Distributed Version Control Systems**

## **Centralized Version Control Systems [CVS, Subversion, and Perforce]**



These systems (such as CVS, Subversion, and Perforce) have a single server that contains all the versioned files, and a number of clients that check out files from that central place.

Centralized collaboration, so everyone knows to a certain degree what everyone else on the project is doing.

Administrators have fine-grained control over who can do what, and it’s far easier to administer a CVCS than it is to deal with local databases on every client.

**Downside**

* Single point of failure that the centralized server represents. If that server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they’re working on.
* If the hard disk the central database is on becomes corrupted, and proper backups haven’t been kept, you lose absolutely everything — the entire history of the project except whatever single snapshots people happen to have on their local machines.

## **Distributed Version Control Systems [Git, Mercurial,Bazaar or Darcs]**



* This is where Distributed Version Control Systems (DVCSs) step in. In a DVCS (such as Git, Mercurial,Bazaar or Darcs), clients don’t just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history.
* So Every clone is really a full backup of all the data.
* This allows you to set up several types of workflows that aren’t possible in centralized systems, such as hierarchical models.

## **What is Git?**

**Snapshots, Not Differences**

* The major difference between Git and any other VCS (Subversion and friends included) is the way Git thinks about its data. Conceptually, most other systems store information as a list of file-based changes.
* With Git, every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot.
* To be efficient, if files have not changed, Git doesn’t store the file again,
* just a link to the previous identical file it has already stored.

**Nearly Every Operation Is Local**

* With git each user has entire history of the project stored locally. Hence making access to file history is extremely fast and allows full functionality even when n/w disconnected. It also means every user has a backup.
* Here most of the operations are performed locally in git, that makes git mush faster compare to version control systems.

**Git Has Integrity**

* Everything in Git is checksummed before it is stored and is then referred to by that checksum.
* This means it’s impossible to change the contents of any file or directory without Git knowing about it.
* Git uses SHA-1 hash, it’s a 40-character string composed of hexadecimal characters (0–9 and a–f) and calculated based on the contents of a file or directory structure in Git.

## **The Three States**

* **Modified** means that you have changed the file but have not committed it to your database yet.
* **Staged** means that you have marked a modified file in its current version to go into your next commit snapshot.
* **Committed** means that the data is safely stored in your local database.

## **The basic Git workflow goes something like this:**

* You modify files in your working tree.
* You selectively stage just those changes you want to be part of your next commit, which adds *only* those changes to the staging area.
* You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

## **Git Installation**

* $ sudo dnf install git-all
* $ git --version

## **First-Time Git Setup**

You should have to do these things only once on any given computer; they’ll stick

around between upgrades.

#git config --system /etc/gitconfig [System specific]

#git config --global ~/.gitconfig [user home directory specification]

#git config --local .gitconfig [current repo]

**Your Identity**

$ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com

**Your Editor**

$ git config --global core.editor vim/nano/notepad++

**Checking Your Settings**

$ git config --list

$ git config user.name