## What is SCM?

Software configuration management (SCM or S/W CM) is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management. SCM practices include revision control and the establishment of baselines. If something goes wrong, SCM can determine what was changed and who changed it. If a configuration is working well, SCM can determine how to replicate it across many hosts.

## What is Version **Control**? (Revision Control or Source Control)

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.

## Advantages of Version Control Systems?

It allows you to revert 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.

Collaboration, Backup

## Types of Version Control Tools

Local Version Control Systems - RCS

Centralized Version Control Systems - CVS, Subversion, and Perforce

Distributed Version Control Systems - Git, Mercurial, Bazaar or Darcs

## Centralized Version Control Systems

CVCS have a single server that contains all the versioned files, and several clients that check out files from that central place.

## Distributed Version Control Systems

In a DVCS the clients don’t just check out the latest snapshot of the files: they fully mirror the repository. Thus, if any server dies, and these systems were collaborating via it, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.

## Advantages of Git

Speed

Simple design

Strong support for non-linear development (thousands of parallel branches)

Fully distributed

Able to handle large projects like the Linux kernel efficiently (speed and data size)

## Snapshots, Not Differences

The major difference between Git and any other VCS is the way Git thinks about its data. Conceptually, most other systems store information as a list of file-based changes. These systems think of the information they keep as a set of files and the changes made to each file over time.

Git doesn’t think of or store its data this way. Instead, Git thinks of its data more like a set of snapshots of a miniature filesystem. Every time you commit, or save the state of your project in Git, it 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. Git thinks about its data more like a stream of snapshots.

## Nearly Every Operation Is Local

Most operations in Git only need local files and resources to operate – generally no information is needed from another computer on your network.

to browse the history of the project, Git doesn’t need to go out to the server to get the history and display it for you – it simply reads it directly from your local database.

You can commit offline & can upload the work once git is online.

## Git Has Integrity

Everything in Git is check-summed with SHA-1 hash 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. This functionality is built into Git at the lowest levels and is integral to its philosophy. You can’t lose information in transit or get file corruption without Git being able to detect it.

## Git Generally Only Adds Data

When you do actions in Git, nearly all of them only add data to the Git database. It is hard to get the system to do anything that is not undoable or to make it erase data in any way.

## The Three States

Committed - Data that is safely stored in your local database

Modified - Changed files but have not committed it to database yet

Staged - Marked modified files in its current version to go into your next commit snapshot

**The Git directory** is where Git stores the metadata and object database for your project. This is the most important part of Git, and it is what is copied when you clone a repository from another computer.

**The working directory** is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.

**The staging area** is a file, generally contained in your Git directory, that stores information about what will go into your next commit.

The basic Git workflow goes something like this:

1. You modify files in your working directory.

2. You stage the files, adding snapshots of them to your staging area.

3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

## Initializing a Repository in an Existing Directory

$ git init

This creates a new subdirectory named .git that contains all your necessary repository files – a Git repository skeleton.

## Cloning an Existing Repository

$ git clone [url]

To get a copy of an existing Git repository. Git receives a full copy of nearly all data that the server has.

$ git clone <https://github.com/libgit2/libgit2>

This creates a directory named “libgit2”, initializes a .git directory inside it, pulls down all the data for that repository, and checks out a working copy of the latest version.

$ git clone <https://github.com/libgit2/libgit2> mylibgit

This command does the same thing as the previous one, but the target directory is called mylibgit.

## Recording Changes to the Repository

## Tracked files - files that were in the last snapshot they can be unmodified, modified, or staged

Untracked files - any files in your working directory that were not in your last snapshot and are not in your staging area

## Checking the Status of Your Files

The main tool you use to determine which files are in which state

$ git status

Short Status

$ git status -s

## Tracking New Files / Staging Modified Files

To begin tracking a new file

$ git add filename