Notes covers about :

What is Version control systems ? Difference between VCS,Central VCS,Disributed CVS.

What is GIT ? How is GIT different from any other CVCS,VCS,DCVS?

SHA-1 HASh(commit value in git log)

Git and git project 3 stages

VERSION 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.

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.

Local Version control:

Easier for errors

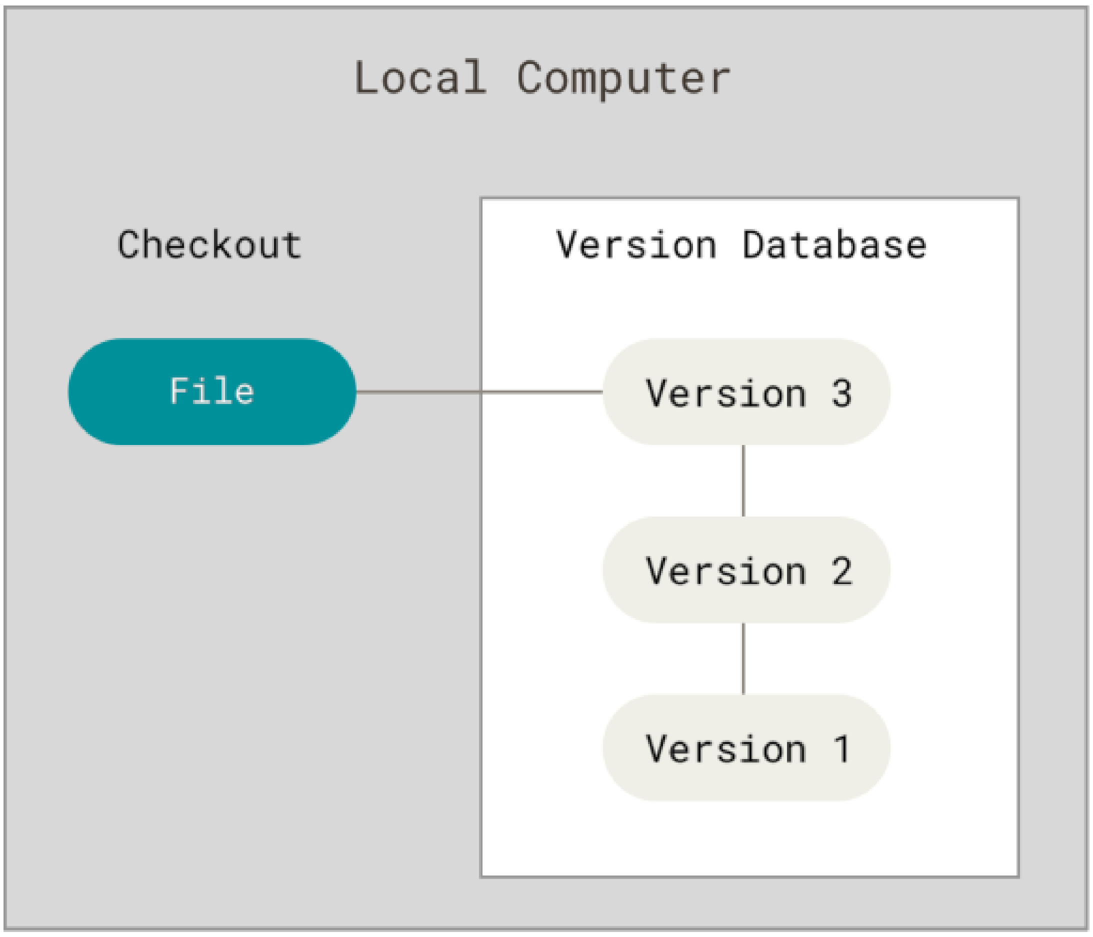

Figure 1. Local version control diagram

Centralised version control systems:

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.

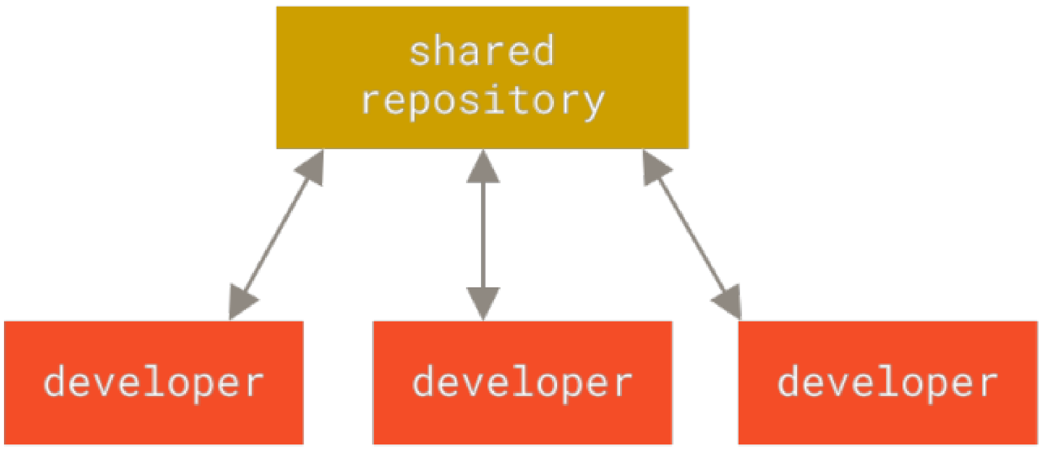

Figure 2. Centralized version control diagram

Cons:
single point of failure that the centralized server represents.
If the 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 Control Version systems:

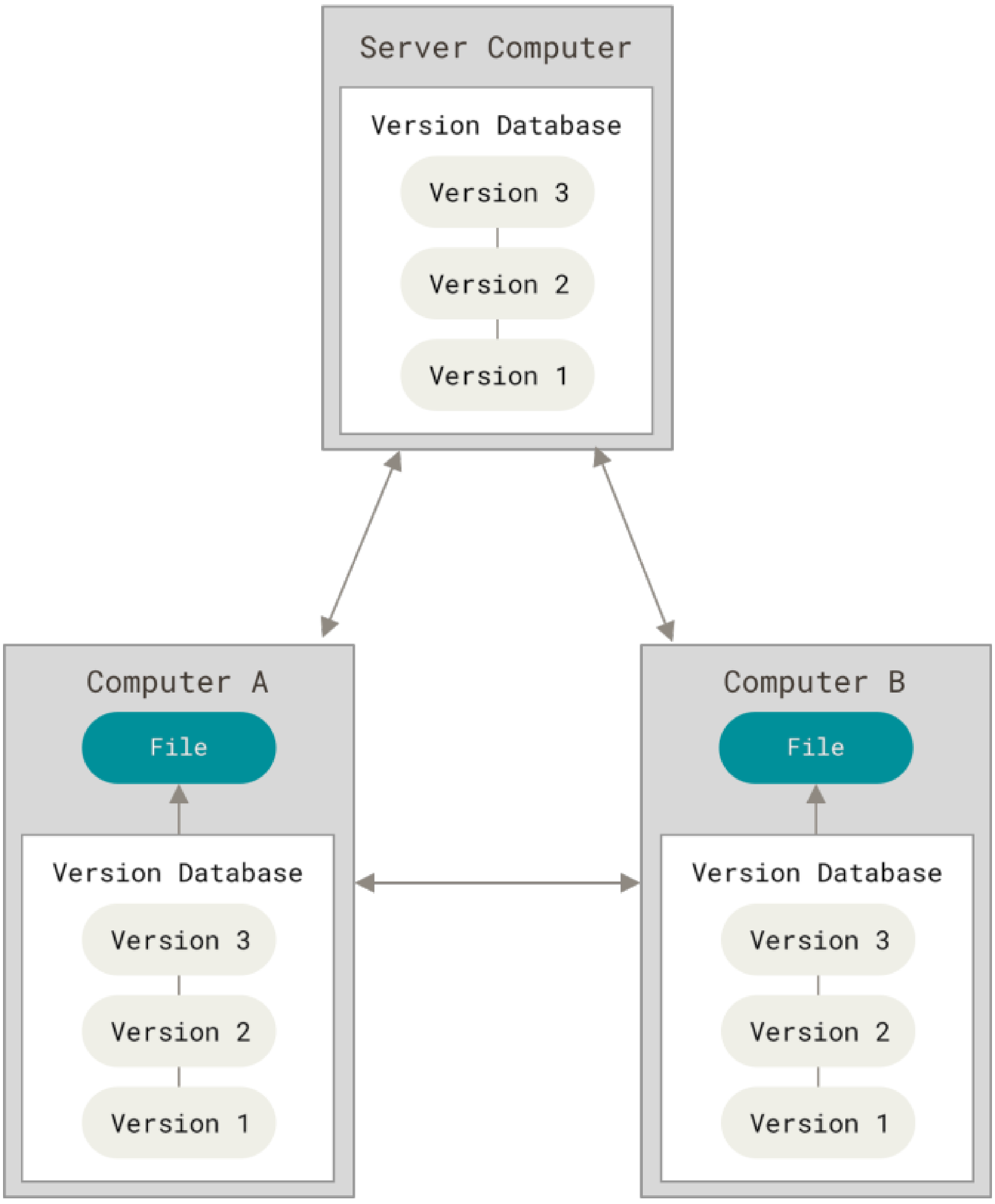

Figure 3. Distributed version control diagram

if any server dies, and these systems were collaborating via that server, 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.

set up several types of workflows that aren't possible in centralized systems, such as hierarchical models.

FEATURES OF GIT :

- simple design
- Speed
- Support for non-linear development(parallel branches)
- Fully distributed
- Handle like linux kernel efficiently

WHAT IS GIT ?

- Git considers snapshots not differences.
- other systems think of the information they store as a set of files and the changes made to each file over time (this is commonly described as delta-based version control)

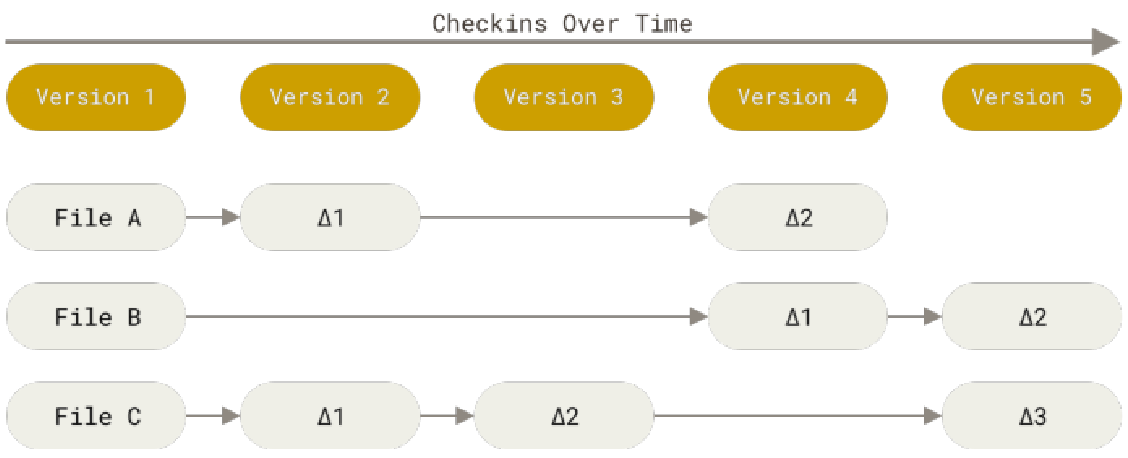

Figure 4. Storing data as changes to a base version of each file

With Git, every time you commit, or save the state of your project, it 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.

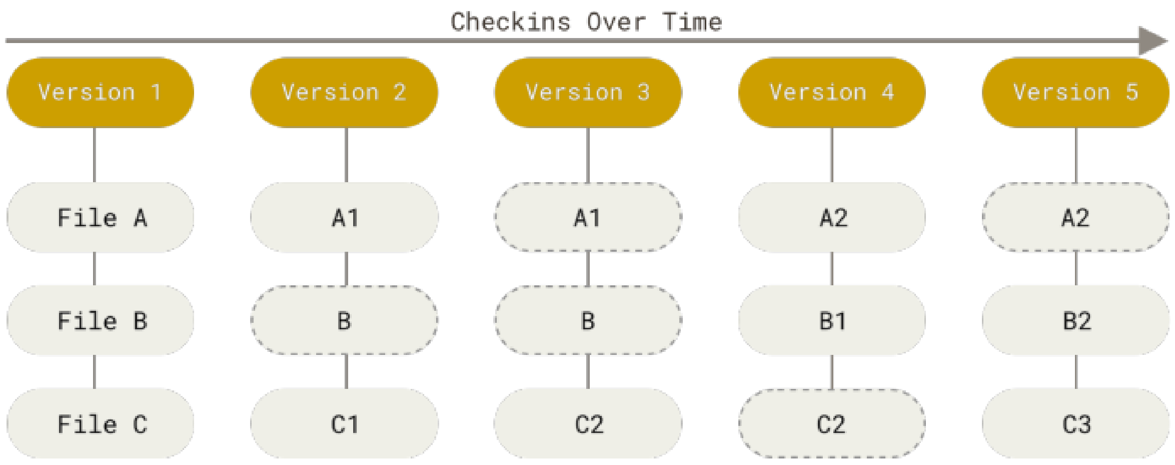

Figure 5. Storing data as snapshots of the project over time

SHA-1 Hash /Git integrity:

- Everything in Git is checksummed before it is stored and is then referred to by that checksum.
- The mechanism that Git uses for this checksumming is called a **SHA-1 hash.**
- a 40-character string composed of hexadecimal characters **(0–9 and a–f)**
- calculated based on the contents of a file or directory structure in Git. A SHA-1 hash looks something like this: 24b9da6552252987aa493b52f8696cd6d3b00373
- Git stores everything in its database not by file name but by the hash value of its contents.

THREE STATES/ARCHITECTURE OF GIT :

Git has 3 main states that your files can reside in modified, staged, and committed:

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

The three main sections of a Git project: the working tree, the staging area, and the Git directory

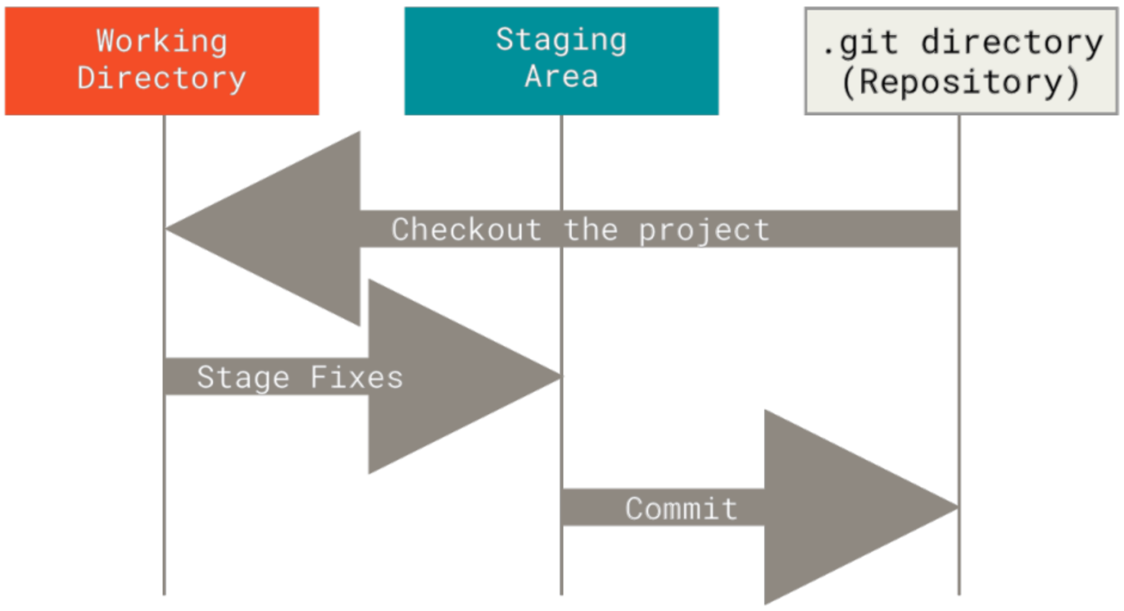

Figure 6. Working tree, staging area, and Git directory

- The working tree 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.
  - Modify files in working tree
- The staging area is a file, generally contained in your Git directory, that stores information about what will go into your next commit.
  - selectively stage just those changes you want to be part of your next commit, which adds only those changes to the staging area
- The Git directory is where Git stores the metadata and object database for your project.
  - You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

If a particular version of a file is in the Git directory, it's considered **committed**. If it has been modified and was added to the staging area, it is **staged**. And if it was changed since it was checked out but has not been staged, it is **modified.**

First Time Setting Up git

Config list

- You can view all of your settings and where they are coming from using:
  git config --list --show-origin

Identifier

- Set your user name and email address. As every Git commit uses this information, and it’s immutably baked into the commits you start creating:
  git config --global user.name "John Doe"
  git config --global user.email johndoe@example.co

Editor

Default Branch name

Git will create a branch named master when you create a new repository with git init.

From Git version 2.28 onwards, you can set a different name for the initial branch.

To set main as the default branch name do: **git config --global init.defaultBranch main**