VERSION CONTROLLING & WORKFLOWS WITH GIT - I

Content

[1. Version controlling 2](#_Toc159682840)

[Methods of version controlling 2](#_Toc159682841)

[Keeping multiple copies of files with different names in a cloud storage 2](#_Toc159682842)

[Automated version controlling systems 3](#_Toc159682843)

[2. History on Version Controlling Systems (VCS) 3](#_Toc159682844)

[Source Code Control System (SCCS) 3](#_Toc159682845)

[Centralized Version Control Systems 3](#_Toc159682846)

[Distributed Version Control Systems 4](#_Toc159682847)

[3. GIT (The De-Facto Standard Today in VCS) 4](#_Toc159682848)

[GIT Basic Concepts 4](#_Toc159682849)

[GIT Concepts 5](#_Toc159682850)

# Version controlling

* a system that records changes to a file or set of files over time so that you can recall specific versions later.

## Methods of version controlling

* Keeping multiple copies of files with different names
* Google Drive / OneDrive
* Undo / Redo Buffer

## Keeping multiple copies of files with different names in a cloud storage

Pros

* All the versions can be safely stored.
* Can be accessed from anywhere.

Cons

* If a file is uploaded with the same name as an existing file, the existing file will be replaced.

The system can,

* Revert
* Compare
* Push
* Pull
* Merge

Can add a log file to view all the changes of the versions which were done by the group members, when working as a group, in the same project.

But the management of the log file is not easy.

## Automated version controlling systems

* GIT
* SVN
* Mercurial

# History on Version Controlling Systems (VCS)

## Source Code Control System (SCCS)

* The first proper VCS.
* Was in the first-generation VCS which are now known as Local Version Control Systems.
* Intended to track changes for individual files.
* Checked out files could only be edited locally by one user at a time.
* Collaboration among large teams was problematic.

## Centralized Version Control Systems

* The second generation of VCS.
* Allowed users to refer files stored on a centralized repository through a network.
* Allowed multiple users to use the files concurrently.
* Follows a client-server approach.
* Require network access and central authority to maintain the centralized repository.

## Distributed Version Control Systems

* The third generation of VCS.
* No requirement of a central authority.
* The contributors may develop features which may or may not be relevant to the main open-source project.
* Has peer-to-peer approach (P2P).
* Widely used today (Git).

# GIT (The De-Facto Standard Today in VCS)

* Created by Linus Torvalds in 2005.
* Originally designed to do version control on Linux kernel.
* Support non-linear development (thousands of parallel branches).
* Fully distributed.

## GIT Basic Concepts

* No need of a centralized repository.
* Each user maintains a local repository of their own.
* When staring, the local repository can be obtained,
  + From another user.
  + From cloning it from the central location.
  + Initialize a brand-new repository from scratch.
* This repository will contain what others have done originally, what the user has done themselves and even the work of others as selected by the user.
* They can commit and update their local repository without any interference from any others.

## GIT Concepts

Has 02 types of repositories.

1. Local repository

Owned by each user.

1. Remote repository

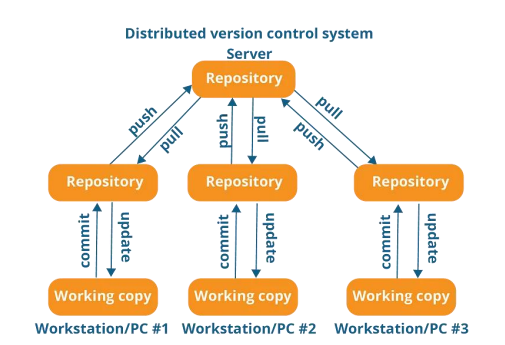
Centralized location accessible to all users.

The users will push their new updates to this to be shared with others.

The users will pull new updates from the others from this.

Can be used to identify the right version when there are multiple versions.

GIT is not GitHub.



* Git works on Commits.
* A Commit is very simply a snapshot of all changes in the local repository (more correctly, the working directory).
* Each Commit has a unique Commit ID.
* A Commit ID is a unique SHA-1 hash that is created whenever a new commit is recorded.
* The most recent commit is called the Head (more correctly, it refers to the currently checked-out branch's latest commit).

