Git and Github

Git Version Control System

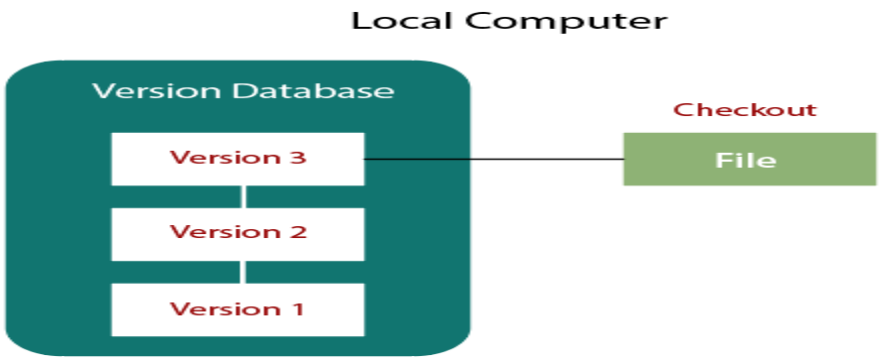
- A version control system is a software that tracks changes to a file or set of files over time so that you can recall specific versions later. It also allows you to work together with other programmers.
- The version control system is a collection of software tools that help a team to manage changes in a source code. It uses a special kind of database to keep track of every modification to the code.
- Developers can compare earlier versions of the code with an older version to fix the mistakes.

Some key benefits of having a version control system are as follows.

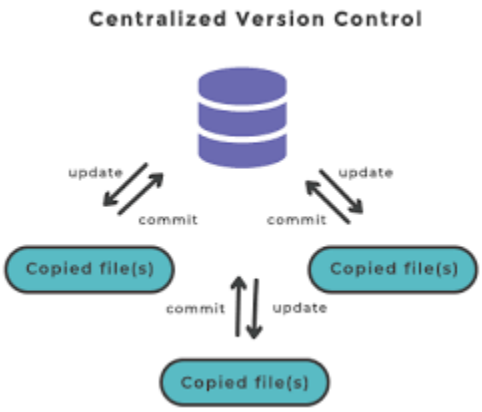
- Complete change history of the file
- Simultaneously working
- Branching and merging
- Traceability

Types of Version Control System

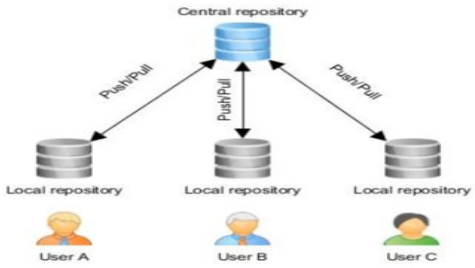
- Localized version Control System



- Centralized version control systems



- Distributed version control systems



What does Git do?

- Manage projects with Repositories
- Clone a project to work on a local copy
- Control and track changes with Staging and Committing
- Branch and Merge to allow for work on different parts and versions of a project
- Pull the latest version of the project to a local copy
- Push local updates to the main project

Working with Git

- Initialize Git on a folder, making it a Repository
- Git now creates a hidden folder to keep track of changes in that folder
- When a file is changed, added or deleted, it is considered modified
- You select the modified files you want to Stage
- The Staged files are Committed, which prompts Git to store a permanent snapshot of the files
- Git allows you to see the full history of every commit.
- You can revert back to any previous commit.
- Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commit!

Git commands

Key/Command	Description
git config --global user.name [name]	Set author name to be used for all commits
git config --global user.email [email]	Set author email to be used for all commits
git config color.ui true	Enables helpful colorization of command line output

Core Commands

Key/Command	Description
git init [directory]	Creates new local repository
git clone [repo]	Creates local copy of remote repository
git add [directory]	Stages specific [directory]
git add [file]	Stages specific [file]
git add -A	Stages all changed files
git add .	Stages new and changed files, NOT deleted files
git add -u	Stages changed and deleted files, NOT new files
git commit -m "[message]"	Commit everything that is staged
git status	Shows status of changes as untracked, modified or staged

Synchronization of Changes

Key/Command	Description
git fetch	Downloads all history from the remote branches
git merge	Merges remote branch into current local branch
git pull	Downloads all history from the remote branch and merges into the current local branch
git push	Pushes all the commits from the current local branch to its remote equivalent

Tip: git pull is the combination of git fetch and git merge

Undo Changes

Key/Command	Description
git checkout -- [file]	Replace file with contents from HEAD
git revert [commit]	Create new commit that undoes changes made in [commit], then apply it to the current branch
git reset [file]	Remove [file] from staging area
git reset --hard HEAD	Removes all local changes in working directory

git reset --hard [commit]	Reset your HEAD pointer to previous commit and discard all changes since then

Branches

Key/Command	Description
git branch [branch]	Create a new branch
git checkout [branch]	Switch to that branch
git checkout [branch] -b	Create and checkout new branch
git merge [branch]	Merge [branch] into current branch
git branch -d [branch]	Deletes the [branch]
git push origin [branch]	Push [branch] to remote
git branch	Lists local branches
git branch -r	Lists remote branches
git branch -a	Lists local and remote branches

History

Key/Command	Description
git log	Lists version history for the current branch
git log --author=[name]	Lists version history for the current branch from certain author
git log --oneline	Lists compressed version history for the current branch
git show [commit]	Outputs metadata and content changes of the specific commit
git blame [file]	Shows who changed what and when in file

Git Setup

git init [directory]	create a Git repository from an existing directory
git clone [repo / URL]	clone / download a repository onto local machine
git clone [URL] [folder]	clone a repository from a remote location into a specified folder [folder] on your local machine

Git Branches

git branch	list all branches in the repository
git branch -a	list all remote branches
git branch [branch]	create a new branch under the specified name
git checkout [branch]	switch to another branch (either an existing one or by creating a new one under the specified name]
git branch -d [branch]	delete a local branch
git branch -m [new_branch_name]	rename the branch you are currently working in
git merge [branch]	merge the specified branch with the current branch

Undoing Changes

git revert [file/directory]	undo all changes in the specified file/directory by creating a new commit and applying it to the current branch
git reset [file]	unstage the specified file without overwriting changes
git reset [commit]	undo all changes that happened after the specified commit
git clean -n	see which files should be removed from the current directory
git clean -f	remove the unnecessary files in the directory

Git Configuring

git config --global user. name "[your_name]"	set an author name that will be attached to all commits by the current user
git config --global user. email "[email_address]"	set an email address that will be attached to all commits by the current user
git config --global color.ui auto	set Git's automatic command line coloring
git config --global alias. [alias_name] [git_command]	create a shortcut (alias) for a Git command
git config --system core.editor [text_editor]	set a default text editor for all the users on the machine
git config --global --edit	open Git's global configuration file

Rewriting History

git commit --amend	replace the last commit with a combination of the staged changes and the last commit combined
git rebase [base]	rebase the current branch with the specified base (it can be a branch name, tag, reference to a HEAD, or a commit ID)
git reflog	list changes made to the HEAD of the local repository

Making Changes

git add [file/directory]	stage changes for the next commit
git add .	stage everything in the directory for an initial commit
git commit -m " [descriptive_message]"	commit the previously staged snapshot in the version history with a descriptive message included in the command

Managing Files

git status	show the state of the current directory (along with staged, unstaged, and untracked files)
git log	list the complete commit history of the current branch
git log --all	list all commits from all branches
git log [branch1]..[branch2]	show which commits are on the first branch, but not on the second one
git diff	see the difference between the working directory and the index (which changes have not been commited yet)
get diff --cached	see the difference between the last commit and the index
get diff HEAD	see the difference between the last commit and the working directory
git show [object]	show the content and metadata of an object (blob, tree, tag, or commit)

Remote Repositories

git remote add [name] [URL]	create a new connection to a remote repository and give it a name to serve as a shortcut to the URL
git fetch [remote_repo] [branch]	fetch a branch from a remote repository
git pull [remote_repo]	fetch the specified repository and merge it with the local copy
git push [remote_repo] [branch]	push a branch to a remote repository with all its commits and objects