**Git**



* **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. Eg: Books launched in various editions.
* **Different Generations of Version Control:**
  + First Generation: Local version control system done on local computer: SCCS and RCS.
  + Second Generation: Centralized version control system: CVS, SVN, Perforce, TFS (by Microsoft).
  + Third Generation: Decentralized version control system: BitKeeper, GNU Bazaar, Mercurial, Git, TFS (revised).
* **Why VCS?**
  + Who made the change?
  + What has changed?
  + Where is the change applied? (which version and branch)
  + When was the change made?
  + Why was the change made?
* **What is GIT?**
  + Git is a VCS for tracking changes in source code in software development.
  + Created by Linus Torvalds in 2005
  + Goals: speed, data integrity, fully distributed, support for distributed, non-linear workflows(branches) and able to handle large projects efficiently.
* **Staging Area in Git**
  + The working directory in our system is local environment, the repository is remote environment.
  + The staging environment is where the staged files (files added through git add) will be stored before pushing them to the repository.
  + It is last step before code is pushed into prod environment where code is being tested.
* **Git Coomands:**
  + **Configuration:**
    - git config --global user.name "[name]"
      * Sets the name you want attached to your commit transactions
    - git config --global user.email "[email address]"
      * Sets the email you want attached to your commit transactions
    - git config --global color.ui auto
      * Enables helpful colorization of command line output
  + **Working with Repositories**
    - git init
      * Turn an existing directory into a git repository
    - git clone [url]
      * Clone (download) a repository that already exists on GitHub, including all of the files, branches, and commits
    - git fork
      * Make a copy of repository in the local system (will copy repo but not connected to the repo)
  + **Making Changes**
    - git log
      * Lists version history for the current branch
    - git log --follow [file]
      * Lists version history for a file, including renames
    - git diff [first-branch]...[second-branch]
      * Shows content differences between two branches
    - git show [commit]
      * Outputs metadata and content changes of the specified commit
    - git add [file]
      * Snapshots the file in preparation for versioning
    - git restore --staged [file]
      * Unstage files from staging area
    - git status
      * Shows the current state of your repository, including tracked and untracked files, modified files, and branch information.
    - git commit -m "[descriptive message]"
      * Records file snapshots permanently in version history
    - git commit -a or git commit --all
      * Commits all modified and deleted files in the repository without explicitly using git add to stage the changes.
    - git show <commit>
      * Shows the details of the specified commit, including its changes.
  + **Redo Commits**
    - git reset [commit]
      * Undoes all commits after [commit], preserving changes locally
    - git reset --hard [commit]
      * Discards all history and changes back to the specified commit
  + **Synchronize Changes**
    - git fetch
      * Downloads all history from the remote tracking branches
    - git merge
      * Combines remote tracking branch into current local branch
    - git push
      * Uploads all local branch commits to GitHub
    - git pull
      * Updates your current local working branch with all new commits from the corresponding remote branch on GitHub. git pull is a combination of git fetch and git merge
    - git pull --rebase
      * Fetches changes from the remote repository and rebases the current branch onto the updated branch.
    - git rebase -I HEAD~N
      * Squash last N commits into a single commit
  + **Branches**
    - git branch [branch-name]
      * Creates a new branch
    - git checkout [branch-name]
      * Switches to the specified branch and updates the working directory
    - git merge [branch]
      * Combines the specified branch’s history into the current branch. This is usually done in pull requests, but is an important Git operation.
    - git branch -d [branch-name]
      * Deletes the specified branch
    - git stash
      * Save the uncommitted changes over the latest commit (used before pulling from the remote)
      * Stashes the changes in the working directory, allowing you to switch to a different branch or commit without committing the changes.
    - git stash pop
      * Pop the most recent stash from the stash stack
    - git stash list
      * Lists all stashes in the repository.
* **Git Submodules [**[**Reference**](https://git-scm.com/book/en/v2/Git-Tools-Submodules)**]**
  + Submodules allow you to keep a Git repository as a subdirectory of another Git repository.
  + git submodule add <url>
    - Add submodule to project using submodule url
  + git submodule init
    - to initialize your local configuration file
  + git submodule update
    - to fetch all the data from that project and check out the appropriate commit listed in main project
  + git clone --recurse-submodules <mainProjectURL>
    - clone all submodules of a project using main project url
  + git submodule update --init –recursive
    - Update all submodules recursively
  + git pull --recurse-submodules
    - Pull changes across all submodules recursively
  + git submodule sync –recursive
    - Update the submodules recursively if git pull / update fails
* Questions:
  + Tag:
    - Git has the ability to tag specific points in a repository’s history as being important.
    - git tag -l(optional -l)
      * Listing existing tags
    - git tag -a v1.4 -m "my version 1.4"
      * Create a tag with version v1.4 and message
    - git show v1.4
      * Show the details of the tag
    - git tag -a v1.2 9fceb02(commit id)
      * Tagging a commit using commit id
    - git push origin <tagname>
      * Sharing a tag to the remote repository
    - git push origin --tags
      * To push more than one tag
    - git tag -d <tagname>
      * Delete a tag in the local repository
    - git push origin --delete <tagname>
      * Delete a tag from the remote repository
    - git checkout <tagname>
      * To switch to a tag
    - git checkout -b <newBranch> <tagName>
      * To checkout a tag on a new branch
  + Head and footer(tail) in conflicts