ENSE 375 Version Control Report
Git Commands and GitHub Operations

Yash Patel

As discussed in the previous section, Git is a version control system that enables a user to record changes to a project over time. Git allows users to take snapshots of the project over time, restore previous versions of the project (or a single file) based on these snapshots, and simultaneously collaborate on multiple versions of a file. Git logs every snapshot (or *commit*) created, along with the information of who created the commit and when. In addition, Git repositories contain three areas: the *working tree* exists on the file system, which can display every change to each file; the *index*, or *staging area*, enables users to save changes made to a file and prepare for future commits, and the *Git history* provides a log of all previous commits to a repository.

Below is a list of numerous useful commands for Git version control:

- **mkdir** – This command creates a new directory, which acts as a sub-directory of the current working directory.
- **vi <filename>** – Creates a new file using vi.
- **git init** – This command creates a .git folder in a particular directory.
- **git config – global user.name "abcdef"** – This command provides a global username for a file. Replacing the user.name with user.email gives a global email for the file.
- **git config – local user.name "qwerty"** – This command provides a local username for a file. Replacing the user.name with user.email gives a local email for the file
- **git config – list** – This command displays all the set field configuration values provided by the user.
- **git status** – This command returns the status of the working tree and the staging area.
- **git add** – This command turns an untracked file into a tracked one and moves it into the staging area.
- **git commit** – This command creates a commit of all files that have been added to the staging area.
- **git log** – This command provides a user with a commit graph, which contains all the commits created, the hexadecimal character hash for all the commits, the author name and email and other information provided by the user, and a message if displayed along with the commits.

- **git diff** – This command enables the user to see the differences between a tracked file in the working area and the same file in the staging area
- **git add .** – The dot after the git add function adds all the new and modified files to the staging area at once
- **git rm <filename>** – This command removes a file and logs its removal.
- **git checkout -- <filename>** – This command replaces the current version of a file in the working tree with a previous version that was stored in the staging area.
- **git diff -- staged** – If a user wants to see the differences between a file in the staging area and the last commit of the file created, this is the command for it.
- **git merge – abort** – Abort a merge process.
- **git stash** – Stash command saves the new changes made by the user in a stash for the user to apply later on.
- **git stash list** – This command displays all the created stashes.
- **git stash list -p** – The -p after the stash list command observes the changes and edits occurred with each stash point.
- **git stash apply** – This command reapplies the most recent stash
- **git stash apply stash@{1}** – The stash@{1} is the label given to a specific stash. Adding the label after the stash apply command reapplies the specific stash as per the label (this stash might not be recent).
- **git stash save “message”** – This command provides a message along with the created stash so that it is easier to understand which stash is responsible for which particular change or edit.
- **git clone <location>** – This command is used for retrieving a specific project from a given remote location of a repo
- **git remote** – This command displays all of the user’s remotes.
- **git remote -v** – This command displays all of the remotes along with their full location addresses.
- **git pull** – This command combines the fetch and the merge commands into a single pull command
- **git push** – This command pushes all components of the most recent commit from the local repository to the remote repository.
- **git remote add** – This command adds a remote Git repository.
- **git remote add <alias> <location>** – This command adds a remote Git repository with an alias pointing to a location.
- **git remote remove <remotename>** – This command removes a remote Git repository.
- **git checkout -b “branchname”** – This command creates and switches to a new branch.
- **git branch** – This command displays only the locally tracked branches.

- **git branch -r** – This command displays only the remotely tracked branches.
- **git branch -a** – This command displays all tracked branches, whether locally tracked or remotely tracked.