**Sahil Gujarathi**

**Employee Code:- 10707**

**Git and Github Document**

**Git and GitHub**

**Git and GitHub**

**Git is a version control system that tracks file changes, while GitHub provides a platform for hosting Git repositories and collaboration.**

**Basic Commands of Git Bash :**

**• ls : list of all files**

**• pwd : Exact location or path**

**• cd : Current directory**

**• .touch : To make file**

**• mkdir : To make folder**

**• rm : Delete file**

**• rm -rf : Forcefully delete directory**

**• ls -a : list of all files.**

**Branching in Git**

**Branching is a core concept in Git that allows us to create separate lines of development within a repository. Each branch can evolve independently from other branches. This is useful for working on features, bug fixes, or experiments in isolation from the main codebase.**

**Dealing with Conflicts in Git (Merging)**

**Conflicts occur when changes from different branches cannot be automatically reconciled by Git. To resolve conflicts, you need to manually edit the conflicting files and then commit the resolved changes.**

**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. Git is a distributed version control system, which means every developer has a full history of the project locally.**

**Git Workflow**

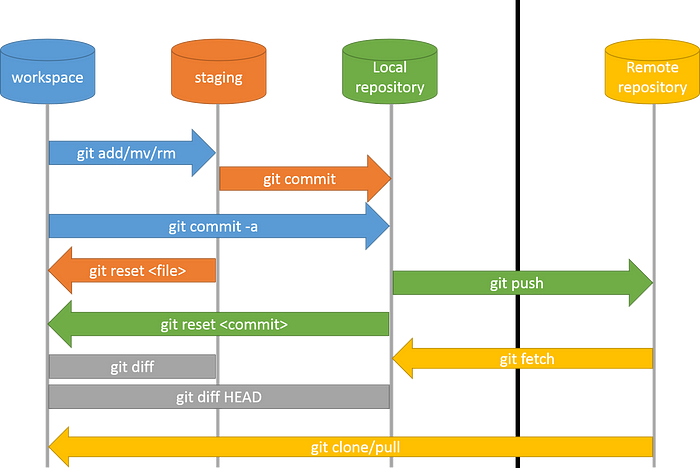
**The Git workflow involves multiple stages:**

**- Working Directory: Where you modify files.**

**- Staging Area: Where you prepare files to be committed.**

**- Local Repository: Where you commit changes.**

**- Remote Repository: Where you push changes to share with others.**

****

**Working Directory, Staging Area, Local Repository, and Remote Repository**

**- Working Directory: Your local workspace.**

**- Staging Area: Files are added here before committing.**

**- Local Repository: Stores committed changes locally.**

**- Remote Repository: Stores changes on a server for collaboration.**

**Git Commands**

**- `git config`: Configures Git settings.**

**- `git add`: Adds changes to the staging area.**

**- `git commit`: Commits changes to the local repository.**

**- `git push`: Pushes changes to the remote repository.**

**- `git pull`: Fetches and merges changes from the remote repository.**

**- `git clone`: Clones a repository.**

**- `git branch`: Manages branches.**

**- `git merge`: Merges branches.**

**Git Stash :**

* Git stash : Purpose of stash is to save the file or changes before committing it .Whichever files are present in the staging area, it will move that files to stash before committing it.
* Git stash POP : Whenever we want files for commit from stash we should use this command.
* Git stash clear : By doing this, all files from stash area is been deleted.
* Git status : It will show the status of all files which are untracked r tracked files. Changes ready to be committed: these are files that have been committed and are ready to be pushed to the central repository.
* Git log : to see all your commits
* Git commit -a -m "message for commit" : -a: commit all files and for files that have been staged earlier need not to be git add once more
* Git push origin master : pushes your files to github master branch

**Steps to Add Git in a Local Folder**

**1. Navigate to the folder: `cd /path/to/your/folder`**

**2. Initialize Git: `git init`**

**3. Add files: `git add .`**

**4. Commit changes: `git commit -m 'Initial commit'**

**Replacing Git URL when SSH is Set Up**

1. **Edit the remote URL: `git remote set-url origin**

**git@github.com:username/repository.git**

**Uncommit Changes**

**- `git reset --soft HEAD~1`: Keeps changes staged.**

**- `git reset --hard HEAD~1`: Discards changes completely.**

**Adding a Remote Origin and Pushing**

1. **Add remote: `git remote add origin**

**git@github.com:username/repository.git`**

**2. Push changes: `git push -u origin master`**

**Git Diff :**

**• Git diff : Difference between to Changes of new code and older code .**

**• Git diff – stage : List the changes between staging area and our last commit.**

**• Git diff HEAD <file\_name> : To view changes in specific file.**

**• Git checkout HEAD : It will detached head**

**• Git reset : git reset is used when we want to unstage a file and bring our changes back to the working directory. git reset can also be used to remove commits from the local repository.**

**Reverting Back**

**- Git revert : git revert is used to remove the commits from the remote repository. Since now our changes are in the working directory, let’s add those changes to the staging area and commit them.**

**- `git revert <commit\_hash>`: Reverts the specified commit.**

**Commit History**

**- `git log`: Shows commit history.**

**- `git log --oneline`: Shows a concise commit history.**

**Flow of Commands to Reflect Changes on GitHub**

**Steps like adding changes, committing them, and pushing to a specific branch on GitHub to update the repository.**

**1. `git add .`**

**2. `git commit -m 'Your message'`**

**3. `git push origin branch\_name`**

**Cloning a Git Repo**

**git clone to create a local copy of a repository from a remote location.**

**- `git clone git@github.com:username/repository.git`**

**Collaborating**

**- Use branches and pull requests to collaborate with others.**

**Git Fetch and Git Merge**

**- `git fetch`: Fetches updates from the remote repository.**

**- `git merge`: Merges fetched changes.**

**Git Fork :**

**Forking a repository means creating a copy of the repo. When you fork a repo, you create your own copy of the repo on your GitHub account. When several developers want to work on a project but need to make changes that are inappropriate for the original repository, forking is frequently used in open-source software development.**

**.gitignore**

**- A file specifying which files and directories to ignore in the repository.**

**Conclusion**

**The documentation provides a thorough guide to Git and GitHub, covering essential commands and advanced techniques. It helps users manage version control, collaborate effectively, and optimize project workflows. From starting repositories to branching and merging, it offers clear instructions for users of all levels, emphasizing best practices for efficient development.**