# Git & GitHub 101

## Basic CLI Commands

1. To list all files or folder in a folder
   `ls`
2. Make a new folder
   `mkdir folder_name`
3. Go inside a folder
   `cd folder_name`
4. To delete a whole non-empty directory/folder
   `rm directory_name -rf`
5. Write a file in Git Bash Vim
   `vim file_name`
   1. use insert key to enable the writing mode in any file
   2. then after finishing edits, press the left-right arrow key to disable the writing mode and then write `:x` to exit out
6. Copy + Paste in CLI
   1. Use the insert key to paste in CLI or highlight the statement then right click and copy that statemen and then right-click on CLI shows the options.

## Basic Git Commands

1. To make a new file
   `touch names.txt`
2. To check if git is installed in your PC
   `git`
3. To initialize an empty Git repository in your folder
   `git init`
4. To view the changes or the untracked files in the project that's not been saved yet
   `git status`
5. Staging the files
   `git add file_name` or `git add .` (to stage everything in the current folder)
6. Committing the files

## Working with Existing Projects on GitHub

Use Git Bash for Windows.

You can't directly change the contents of a repo unless you have access to it. To solve this, you create a copy (fork) of this project in your own account. In our own copy, we can do anything we want with it. After forking, we:

1. Cloning the forked project to local machine
   `git clone forked_repo_url`
2. The public repo that we forked out local copy from is known as the upstream url. We can save it as
   `git remote add upstream insert_upstream_url`
3. Creating a new branch
   `git branch branch_name`
4. Then shift the head to the above branch using the checkout command
5. Then stage. Then commit.
6. Then push. We can't push to upstream (no access). Can push to our forked repo though (origin)
   `git push origin your_branch_name`
7. Always make different branches for different pull requests if you're working on different features. 1 branch = 1 pull request (never commit on main (2))
8. To remove a commit
   1. we can remove a commit with the reset command Now it's unstaged.
   2. then add. to stage the remaining files
   3. then we can use the stash command to stash it elsewhere
   4. then, we'll have to force push this branch since the online repo contains a commit which the local repo does not

   `git push origin your_branch_name -f`
9. To make forked project even (updated) with the main project

```
git commit -m"your_message_here"
```

7. To unstage or remove a file from the staging level
   ```
   git restore --staged file_name.txt
   ```
8. To view the entire history of the project
   ```
   git log
   ```
9. Removing a commit from the history of a project
   ```
   git reset insert_commit_hash_id_to_which_you_want_to_go_back_to_here
   ```
   (all the commits or changes before this will go back to the unstaged area now)
10. After you stage a few files but then you want to have a clean codebase or reuse those files later, we can stash those changes to go back to the commit before they were staged
    ```
    git stash
    ```
11. Bringing back those changes or pop them from the stash
    ```
    git stash pop
    ```
12. To clear the changes or files in your stash
    ```
    git stash clear
    ```

## How Git works

1. Connecting your Remote Repository to Local Repository
   ```
   git remote add origin insert_https_project_link_here
   ```
2. Pushing local changes to remote repository
   ```
   git push origin master
   ```
   (we're pushing to the url origin, and the branch master)
3. To view all your remote urls
   ```
   git remote -v
   ```
4. Never commit on the main branch since it's the one used by the people, to prevent any mishaps
5. Shifting the head to a branch (head is the pointer which points to where all you do your changes)
   ```
   git checkout branch_name
   ```
6. Merging your branch to main of project
   ```
   git merge branch_name
   ```
1. Shift the head to your main branch
   ```
   git checkout main
   ```
2. Fetching all the commits/changes from the main project (upstream)
   ```
   git fetch --all --prune
   ```
   (here prune gets deleted commits too)
3. Reset the main branch of origin (forked) to main branch of upstream (main project)
   ```
   git reset --hard upstream/main
   ```
4. Check and verify your changes
   ```
   git log
   ```
   *click q for exit from log*
5. Then push all these local changes to your online forked repo
   ```
   git push origin main
   ```

Method 2

1. To fetch all **at once**
   ```
   git pull upstream main
   ```
2. Then push to the origin url or your forked project
   ```
   git push origin main
   ```

Method 3

1. Update using the Fetch Upsteam button on forked repo

10. Squashing all your multiple commits into one commit
    ```
    git rebase -i insert_hash_code_of_commit_above_which_all_your_required_c
    ```
    If there's 4 commits. Keep 1 as the pick and then `s` or squash the other 3 into that one
11. Merge conflicts and how to resolve them
    1. They happen when multiple users edit the same code line and then push it. Git won't know which one to merge and then there'd be a conflict
    2. This has to be resolved manually by repo maintainer

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/94dcc4e7-259a-4c09-bf05-d1ce5eb2d9e3/atlassian-git-cheatsheet.pdf