**BASIC COMMAND LINE AND GIT COMMANDS**

**1. cd (Change Directory)**

* **What it does**: Changes the current working directory.
* **Example**: **cd** folder\_name

**2. ls (List)**

* **What it does**: Lists all files and directories in the current folder.
* **Example**: **ls**

**3. mkdir (Make Directory)**

* **What it does**: Creates a new directory.
* **Example**: **mkdir** new\_folder\_name

**4. git init**

* **What it does**: Initializes a new Git repository in the current directory.
* **Example**: **git init**

**5. ls -a**

* **What it does**: Lists all files, including hidden files (those starting with a dot .).
* **Example**: **ls -a**

**6. touch**

* **What it does**: Creates a new empty file.
* **Example**: **touch** filename.extension

**7. git add**

* **What it does**: Adds files to the staging area.
* **Examples**:

**git add** filename.extension # Adds a specific file

**git add** --all # Adds all files

**8. git status**

* **What it does**: Shows the status of the working directory and staging area (e.g., which files are staged, unstaged, or untracked).
* **Example**: **git status**

**9. git config --global user.email**

* **What it does**: Sets the global email address for Git commits.
* **Example**: **git config --global** user.email [your.email@gmail.com](mailto:your.email@gmail.com)

**10. git commit -m "Your commit message"**

* **What it does**: Commits the staged changes with a message.
* **Why it’s useful**: Each commit is like a "save point" in your project. You can go back to it if you find a bug or want to make changes.
* **Example**: **git commit -m** "Your commit message"

**11. vi (Visual Editor)**

* **What it does**: Opens a file in the Vi text editor for editing.
* **Example**: **vi** filename.extension
* **Tip: To save and exit, press Esc, then type :wq and hit Enter.**

**12. cat (Concatenate)**

* **What it does: Displays the contents of a file.**
* **Example: cat** filename.extension

**13. git restore --staged**

* **What it does**: Unstages a file (removes it from the staging area).
* **Example**: **git restore --staged** filename.extension

**14. rm -rf (Remove)**

* **What it does**: Deletes a file or directory forcefully.
* **Example**: **rm -rf** filename.extension
* **Warning:** This command is irreversible, so use it with caution.

**15. git reset [--soft | --mixed | --hard] <commit-hash>**

* **What it does**: A powerful command to undo changes in your repository.
  + **--soft:** Moves the HEAD pointer but keeps changes in the staging area and working directory.
  + **--mixed (default):** Moves the HEAD pointer and resets the staging area but keeps changes in the working directory.
  + **--hard:** Moves the HEAD pointer, resets the staging area, and discards all changes in the working directory.
* **Example**: **git reset** **--mixed a1b2c3d4**

**16. git log**

* **What it does**: Shows the commit history of your branch, including commit messages, authors, and timestamps.
* **Example**: **git log**

**17. git reflog**

* **What it does**: Shows a log of all actions (e.g., commits, checkouts, resets) in your repository. Useful for recovering lost commits.
* **Example: git reflog**

**18. git [command] -help**

* **What it does:** Shows all options for a specific Git command.
* **Example: git commit -help**

**19. git help --all**

* **What it does: Lists all possible Git commands.**
* **Example: git help –all**

**20. git branch**

* **What it does**: Lets you create, list, rename, or delete branches.
* **Examples**:

git branch # Lists all branches in git

git branch new-branch # Creates a new branch

git branch -a # List all the branches include in the repository (github)

git branch -r # List all the branches in the repository only (github)

**21. git checkout / git switch**

* **What it does**: Switches to a different branch or restores files from a specific commit or branch.
* **Examples**:

**git checkout branch-name** # Switches to the specified branch

**git checkout -b new-branch** # Creates and switches to a new branch

**22. git merge**

* **What it does**: Combines changes from one branch into another.
* **Example**:

git checkout main

**git merge feature-branch**

**23. git branch -d**

* **What it does**: Deletes a branch that you no longer need.
* **Example**: **git branch -d old-branch**
* **Use -D to force delete the branch**

**24. git remote add origin [URL]**

* **What it does**: Connects your local repository to a remote repository (e.g., GitHub).
* **Example: git remote add origin** <https://github.com/user/repo.git>

After remoting

* **git push --set-upstream origin master/main** - It pushes the commits from your local master branch to the remote repository (origin). It links your local master branch to the remote master branch (origin/master). This tracking relationship allows Git to know where to push and pull changes in the future.
* After setting the upstream, you can simply use git push or git pull without specifying the remote and branch name every time.

**25. git fetch**

* **What it does**: Downloads changes from the remote repository but doesn’t apply them.
* **Example: git fetch** origin

26**. git merge**

* Combines changes from one branch (e.g., a remote branch) into your current branch.
* It integrates changes from another branch into your work.
* EXAMPLE: **git merge origin/master**

**27. git pull**

* **What it does**: Fetches changes from the remote repository and merges them into your current branch.
* **Example**: **git pull origin main**

**28. git log origin/master**

* **What it does**: Shows the commit history of the master branch on the remote repository.
* **Example**: **git log** **origin/master**

29. **git diff origin/master**

* **What it does**: Compares your current branch with the master branch on the remote repository.
* **Example**: **git diff origin/master**

30. **git stash**

* **What it does**: Temporarily saves uncommitted changes so you can switch branches or tasks without losing work.
* **Examples**:

git stash # Saves changes

git stash list # Lists all stashes

git stash apply # Applies the most recent stash

git stash drop stash@{1} # Deletes a specific stash

git stash clear # Deletes a list of stash

git stash pop # Applies and deletes the most recent stash

31. **git push origin**

* **What it does:** Pushes your local branch changes to the remote repository (origin).
* **Key point:** Requires upstream tracking to be set first (use -u on first push).

32. **git branch -m old\_name new\_name**

* **What it does:** Renames a branch locally.
* **Important:** Does NOT rename the branch on remote - you'll need to push the new name and delete the old one.
* Examples:

git branch -m feature-old feature-new # Renames branch

git push origin :feature-old feature-new # Deletes old remote branch, pushes new one

git push -u origin feature-new # Sets upstream for new branch

33. **nano filename.extension**

* **What it does**: Opens a file in the Nano text editor (terminal-based).
* Common actions:
  + **Save: Ctrl+O then Enter**
  + **Exit: Ctrl+X**
* Example:

nano README.md

34. **git clone /path/location/**

* **What it does**: Creates a local copy of a repository from a specified path (local or remote).
* **Key use**: Clones repositories from GitHub/GitLab or local network paths.
* **Examples**:

git clone https://github.com/user/repo.git # From URL

git clone /home/user/project/ # From local path

35. **git remote**

* **What it does**: Lists all configured remote repositories.
* **Key point**: Shows shorthand names (like origin) for your remotes.
* **Example output**:

origin

upstream

36. **git remote -v**

* **What it does**: Shows all remotes with their URLs (verbose mode).
* **Key use**: Verifies fetch/push URLs for your remotes.
* **Example output**:

origin https://github.com/user/repo.git (fetch)

origin https://github.com/user/repo.git (push)

37.

38.

39.

40.

**Additional Notes**

* **git restore**: Use git restore --staged to onstage files, and git restore (without --staged) to discard changes in the working directory.
* **vi**: To save and exit, press Esc, then type: wq and hit Enter.
* **rm -rf**: Be very careful with this command, as it permanently deletes files and directories.
* **git reset vs git revert**: git reset moves the HEAD pointer and can discard commits, while git revert creates a new commit that undoes changes.
* After **git remote add origin [URL]. git push --set-upstream origin main.** Creates tracking relationship so future pushes can use just git push

**Branch Deletion Reminder**

* **Safe delete** (only if merged): git branch -d old-branch
* **Force delete** (unmerged changes): git branch -D old-branch

**Fetch vs Pull**

* **git fetch:** Downloads changes but doesn't merge
* **git pull:** Does fetch + merge in one step







 