1. **Git Commands:**

* git version : The command git version is used to check the version of git.

**git --version**

* git config: Configures Git settings. Commonly used to set up user information

**git config --global user.name "Your Name"**

**git config --global user.email "youremail@example.com"**

* **git config --list**: Displays all the Git configurations for the current user.

1. **Repository Management**

* **git init**: Initializes a new Git repository in the current directory

**git init**

* **git clone**: Creates a copy of an existing Git repository from a remote source (e.g., GitHub) to your local machine

**git clone** [**https://github.com/username/repository.git**](https://github.com/username/repository.git)

* **git remote –v :** To see the remote repository that is connected to your local repository

**git remote –v**

* **git remote add :** To add a new remote repository to your local repository

**git remote add origin** [**https://github.com/username/repository.git**](https://github.com/username/repository.git)

* **Pushing Changes to Remote:** To push your local commits to a remote repository:

git push <remote\_name> <branch\_name>

**git push origin main**

* **Pulling Updates from Remote:** To pull the latest changes from the remote repository and merge them into your local branch

git pull <remote\_name> <branch\_name>

**git pull origin main**

* **git remote remove:** To remove a remote repository from your local configuration:

git remote remove <remote\_name>

**git remote remove origin**

* **Renaming a Remote:** to rename an existing remote:

git remote rename <old\_name> <new\_name>

**git remote rename origin upstream**

* **Fetching Updates from Remote:** To fetch updates from the remote repository but not merge them into your local branch

**git fetch <remote\_name>**

* **Changing Remote URL :** To change the URL of a remote (e.g., after changing the remote repository address):

git remote set-url <remote\_name> <new\_url>

**git remote set-url origin** [**https://github.com/username/new-repository.git**](https://github.com/username/new-repository.git)

* **Viewing the Remote Repository’s Information:** to see detailed information about a remote repository

git remote show <remote\_name>

**git remote show origin**

**3. Staging and Committing**

* **git status**: Shows the status of changes in your working directory and staging area. It tells you which files are untracked, modified, or ready to be committed.

**git status**

* **git add**: Adds changes in the working directory to the staging area.

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

**git add . # Adds all changes in the directory**

* **git commit**: Commits the staged changes to the repository with a descriptive message. The -m option allows you to include a commit message.

**git commit -m "Commit message describing changes"**

**4. Branching and Merging**

* **git branch**: Lists all branches or creates a new branch

**git branch # Lists all branches or**

**git branch -a # Lists all branches**

**git branch branch-name # Creates a new branch**

**git branch -d <branch\_name> # Deletes a branch**

*Use -D to force-delete if it hasn't been merged.*

* **Listing All Remote Branches :**  To you want to list all branches on the remote repository

**git branch –r**

* **git checkout**: Switches to a different branch

**git checkout branch-name # Switches to an existing branch**

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

* **Pruning Deleted Remotes :** If a branch was deleted on the remote but still shows up locally.

**git remote prune origin**

When someone **deletes a branch on the remote**, your local Git doesn’t automatically remove the corresponding remote-tracking branch (like origin/old-feature).  
git remote prune origin removes these outdated references.

* **Fetching a Specific Remote Branch:** To fetch a specific branch from a remote.

git fetch <remote\_name> <branch\_name>

**git fetch origin feature-branch**

* **Setting the Upstream Branch for Pushing :** When pushing for the first time and want to set the remote branch you are pushing to:

git push --set-upstream <remote\_name> <branch\_name>

**git push --set-upstream origin feature-branch**

* **git merge**: Merges the specified branch into the current branch. This command integrates the changes from the feature branch into the main branch.

**git checkout main # Switch to the main branch**

**git merge branch-name # Merge branch-name into main**

* **Rebasing a Local Branch onto a Remote Branch:** If you want to rebase your local branch onto a remote branch (this can be useful to keep your history linear)

git fetch <remote\_name>

git rebase <remote\_name>/<branch\_name>

Example:

**git fetch origin**

**git rebase origin/main**

* 1. **Undoing Changes**
* git reset :Removes the specified file from the staging area but leaves the working directory unchanged. git reset --hard can also reset the working directory and staging area to the last commit.

**git reset <file>:**

* git revert**:** Creates a new commit that undoes the changes from a specified commit, leaving the history intact**.**

**git revert <commit>:**

* 1. **Viewing History**
* git log**:** Shows a history of commits in the repository, including commit hashes, messages, and timestamps. Use git log --oneline for a more concise view.

**git log:**

* git diff**:** Displays differences between various commits, the working directory, and the staging area. git diff without arguments shows changes not yet staged.

**git diff:**

* git show**:** Shows the details of a specific commit, including the changes made and the commit message.

**git show <commit>**

* 1. To undo the changes made to file before staging it.
* **git restore filename**
  1. To correct committed with the wrong message
* **git commit --amend -m "Corrected commit message**"

9. Recovering deleted branches

* **git reflog**
* **git checkout -b feature-ui <commit\_hash>**

10. To download the latest changes from the remote without merging

* **git fetch origin**

**11.** To remove accidentally committed sensitive file from Git history.

* **git filter-branch --force --index-filter \**
* **"git rm --cached --ignore-unmatch secrets.txt" \**
* **--prune-empty --tag-name-filter cat -- --all**

**12.** To merge changes from another branch

First switch to branch where changes are to applied and then merge another-branch

* **git checkout previous-branch**
* **git merge another-branch**

**13. To resolve a merge conflict manually**

You tried to merge two branches and Git reported a conflict in x.js. Then for conflict resolution do:

1. **Open x.js and resolve/remove the conflict markers (<<<<<<<, =======, >>>>>>>)**
2. After resolving:

* **git add app.js**
* **git commit # If Git didn’t auto-create a merge commit**

**14. The .gitignore**

The .gitignore file is used to tell Git which files or directories to ignore in your project. It's a useful way to avoid committing unnecessary files like log files, build outputs, and IDE configurations.

* 1. **Create a .gitignore File**

In the root directory of your Git project, create a file called .gitignore

touch .gitignore

* 1. **Add Rules to .gitignore**

Inside the .gitignore file, you add patterns for the files and directories you want Git to ignore. Each pattern should be written on a new line.

Here are some common examples:

* Ignore all .log files: **\*.log**
* Ignore a specific file: **secret\_file.txt**
* Ignore all files in a temp/ directory: **temp/**
* Ignore all files except important\_file.txt inside a folder:

**folder/\***

**!folder/important\_file.txt**

* Ignore files with a specific extension: **\*.bak**

**15. To see who changed a particular line in a file**

* **git blame script.py**

**16. Git stash**

# You modified files A and B, but not committed them.

**git stash** # saves changes and reverts to clean state

**git switch another-branch**

**# do something else...**

**git switch main**

**git stash apply** # brings back your changes

**17. To check branch is merge**

* **git branch –merged**

If branch not merged then displays its name

**18. To Delete multiple local branches at once**

* **git branch –d branc1 branch2 branch3**