**Git & GitHub**

**1. Installing Git & Doing the First Setup**

Before writing any commands, you need Git on your computer. To check if it’s already installed, open your terminal and type:

git --version

If it shows something like git version 2.40.0, you're good to go. If it says "command not found", head to [git-scm.com](https://git-scm.com/)and install Git.

Next, Git needs to know **who you are**. This is important because every change you make will have your name and email attached, so others know who contributed.

Run these commands (replace with your name and email):

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

git config --global user.email "you@example.com"

Want to double-check? Run:

git config --list

You should see something like:

user.name=Your Name

user.email=you@example.com

That’s it—Git now knows you.

**2. Creating Your First Repository (Local)**

A **repository** (or repo) is like a special folder where Git will track changes.

**Steps:**

1. Make a folder for your project:

mkdir demo

1. Move inside it:

cd demo

1. Turn it into a Git repo:

git init

That last command creates a hidden folder called .git, which is where Git tracks everything.

Want to see the current status?

git status

It will tell you if there are files to be saved, if something is untracked, etc.

**3. Adding & Committing Files**

Let’s create our first file:

touch a.txt

Add some text to it (open it in an editor or use echo "hello" > a.txt).

**Stage the file:**

git add a.txt

This means: “Hey Git, I want to save this file.”

**Commit (save) it:**

git commit -m "first commit"

Now that change is recorded permanently in history with your message “first commit”.

**4. Viewing History & Changes**

Want to see what happened?

git log

This will show each commit, who made it, when, and the message.

Want a shorter view?

git log --oneline

What if you want to know exactly what changed in your files (line by line)?

git diff

If you staged changes but haven’t committed yet:

git diff --staged

**5. Branching & Merging (Parallel Universes)**

Branches let you work on a new idea without messing up the main project.

**Create a new branch:**

git branch feature1

Switch to it:

git checkout feature1

# OR (new way)

git switch feature1

Now, any changes you make are in feature1.

**Merge back to main:**

Switch to main:

git checkout main

Merge:

git merge feature1

Boom! The feature1 changes are now in main.

**6. Delete a Branch**

If you're done with it:

git branch -d feature1

This safely deletes it (only if it’s merged).

**7. Connecting to GitHub**

**Steps on GitHub:**

1. Go to [GitHub](https://github.com/).
2. Click **New repository**.
3. Give it a name and click **Create repository**.

**Connect your local repo:**

git remote add origin https://github.com/yourusername/demo.git

git branch -M main

git push -u origin main

This links your local repo to GitHub and uploads it.

**8. Cloning a Repository**

If you want someone else’s project (or your own repo on another machine):

git clone https://github.com/yourusername/demo.git

Now you have a copy with full history.

**9. Working with Others (Fork & Upstream)**

If it’s **someone else’s repo**:

* Click **Fork** on GitHub → creates a copy in your account.

Clone your fork:

git clone https://github.com/yourusername/originalproject.git

Add original repo as “upstream” (so you can keep your copy updated):

git remote add upstream https://github.com/originaluser/originalproject.git

Update your fork:

git fetch upstream

git merge upstream/main

**10. Contribution Workflow (Open Source)**

1. Fork repo
2. Clone locally
3. Create branch:

git checkout -b myfeature

1. Make changes:

git add .

git commit -m "Added feature"

1. Push:

git push origin myfeature

1. Open a Pull Request on GitHub.

**11. Fixing Mistakes**

Unstage a file:

git reset HEAD filename

Undo last commit but keep changes:

git reset --soft HEAD~1

Completely remove last commit + changes:

git reset --hard HEAD~1

Safer undo:

git revert HEAD

**12. Temporary Saves (Stash)**

Save unfinished changes:

git stash

See stashes:

git stash list

Bring back:

git stash pop

Clear all:

git stash clear

**13. Tags & Releases**

Mark version:

git tag v1.0

Annotated tag:

git tag -a v1.0 -m "version 1"

Push tags:

git push origin --tags

**14. GitHub Profile README**

Create a repo with your **username** as repo name.  
Add README.md → Whatever you write here will show on your profile page.

**15. Markdown Quick Notes**

* # Heading → Big title
* \*\*bold\*\*, \*italic\* → styling
* Lists: - item1
* Links: [text](url)
* Images: ![alt](img.png)

**16. Advanced Tools**

* git cherry-pick <commit> → take a specific commit
* git rebase main → move commits on top for cleaner history
* git reflog → track HEAD movements (recover lost commits)
* git shortlog → commits grouped by author
* git bisect → find bug-introducing commit