**1) What is Git vs. GitHub?**

* **Git** is a version control tool on your computer that tracks changes to files (code, docs, data).
* **GitHub** is a website that hosts Git repositories so teams can **share, review, and collaborate** (issues, pull requests, CI/CD with Actions, wikis, etc.).

Use case analogy: Git is the notebook you write in; GitHub is the shared bookshelf where everyone can read and contribute.

**2) The three states: Working Directory → Staging → Local Repo**

* **Working Directory**: your actual files.
* **Staging Area** (index): a “shopping cart” of changes you plan to commit.
* **Local Repository**: your saved snapshots (commits) and history.

Core loop: edit → git add → git commit → repeat

**3) Branching & Merging (safe parallel work)**

* **Branches** let you experiment without breaking main.
* Typical flow:
  1. git checkout -b feature/login-ui
  2. make commits
  3. merge back: git checkout main → git merge feature/login-ui

**4) Pull Requests (PRs): collaborative code review**

* **Fork (optional)** a repo on GitHub if you don’t have write access.
* **Clone** to your machine, create a branch, commit changes, **push** to your fork, then **open a PR** into the original repo.
* Reviewers comment, request changes, and **merge** when approved.

**.**

**5) Fork vs. Clone (they’re different!)**

* **Fork**: makes **your own copy on GitHub** under your account.
* **Clone**: makes a **local copy on your computer**.

**6) Git commands**

1. Install Git, create a GitHub account.
2. Configure identity:  
   git config --global user.name "Your Name"  
   git config --global user.email "you@example.com"
3. On GitHub, create a repo (e.g., hello-git) → **Clone** it.
4. Add a file, then:  
   git add .

git commit -m "Initial commit"

git push -u origin main

1. Create a branch, edit README, commit, push, open a **PR**, request review, and **merge**.

**7) “Why GitHub?” .**

* **Reproducibility**: every result ties to a commit hash.
* **Traceability**: Issues + PRs document decisions and code review.
* **Automation**: GitHub Actions for CI/CD (tests, lint, deploy).
* **Team workflows**: branching strategies (feature branches, GitFlow), protected branches, required reviews.
* **Portfolio**: public repos and contribution graph for placements.

**8) Common mistakes & how to teach around them**

* “I committed to the wrong branch” → git switch -c fix-branch then PR into the right target.
* “I can’t push” → forgot -u origin <branch> on first push or don’t have permission (use a fork).
* “Merge conflicts scare me” → teach conflicts as **two edits to the same lines**; resolve in editor, git add, then git commit (or git merge --continue).
* “My repo is huge” → teach .gitignore early (e.g., ignore node\_modules/, datasets, build artifacts).









