ScopeTreasure Courses

GIT & GITHUB COURSES · LESSON G.4 · CODE SCOPE SERIES

GitHub Deep Dive

Your code's online home: creating a repo, the first push, a README that explains the project, and clone vs fork — plus a first look at Issues.

Introduction

Your repo lives on your laptop — which means it dies with your laptop. **GitHub** is where repositories live online: backed up, visible to employers, and open for teamwork. This lesson does the full first journey: create a repo on github.com, connect it, **push** your history up, write a **README** that explains the project to a stranger, and understand the two ways code travels back down — **clone** (your own copy to work on) and **fork** (your own copy of someone else's project, the doorway to open source in G.10). *(Roman Urdu: GitHub aap ke code ka online ghar hai — backup bhi, portfolio bhi, aur team ka kaam bhi wahin se hota hai.)*

Key points

- **A GitHub repo is created empty, then filled by a push:** New repository → name → Create. Do NOT tick "Add a README" when you plan to push an existing local repo — the two histories would collide.
- **`origin` is just a nickname:** `git remote add origin <url>` saves the repo's address in your local config. `git remote -v` shows the fetch/push addresses any time.
  (Roman Urdu: origin asal address ka short naam hai — asal pata git remote -v se hamesha dekh sakte hain.)
- **README.md is the project's front door:** it renders on the repo's home page. A good one answers three questions fast: what is this, how do I run it, what does it look like (a screenshot helps more than adjectives).
- **Clone vs fork: clone** = download a repo to your computer (you get a working copy; you can push only if you have permission). **Fork** = make your own copy of the repo *on GitHub*, under your account, which you CAN push to — then propose changes back with a pull request.
- **Issues are the repo's to-do list and discussion board:** bugs, ideas and tasks live there, numbered, labelled, and linkable to the pull requests that fix them.

Code examples

Example 1 — Create the repo and push for the first time

```
$ git remote add origin https://github.com/ayesha-khan/notes-app.git
$ git remote -v
origin    https://github.com/ayesha-khan/notes-app.git (fetch)
origin    https://github.com/ayesha-khan/notes-app.git (push)
$ git push -u origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 812 bytes | 812.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ayesha-khan/notes-app.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

What it does: on github.com you clicked New repository, named it `notes-app`, and created it empty. These commands connect your local repo to that address and push `main` up; `-u` sets up tracking, so from tomorrow your pushes and pulls are just `git push` and `git pull`.

What you will see: the repo page transforms from an empty setup screen to your actual project — file list, your commit messages, and a green **Code** button. The commit count on the page matches `git log` on your machine exactly, because it IS the same history.

Example 2 — A README that explains the project

```
# Notes App

A tiny app to save study notes in the browser — built while learning Git & GitHub.

**Run it:** Open `index.html` in any browser. No install, no server.

**What is inside:** `index.html` (the page) and a `notes/` folder with the sample notes used
in the demos.

Made with the Code Scope Git & GitHub series.
```

What it does: this file sits at the repo's root as `README.md`. GitHub renders Markdown files named README right on the repo home page — headings, lists and all — so a visitor understands the project in ten seconds without opening a single file.

What you will see: after `git add README.md`, commit and `git push`, the repo page shows the big title, the run instructions and the contents line, formatted like a small web page. Employers and teachers read this first;

a clear README is the cheapest portfolio upgrade there is.

Example 3 — Clone vs fork, side by side

```
$ git clone https://github.com/ayesha-khan/notes-app.git
Cloning into 'notes-app'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
Receiving objects: 100% (6/6), 812 bytes | 812.00 KiB/s, done.
$ cd notes-app && git log --oneline
7c8d9e0 (HEAD -> main, origin/main) Pehla commit: README jora
b2c3d4e Login page ka form jora
```

What it does: `git clone` downloads the whole repo — files AND full history — and automatically sets up `origin` pointing back at the source; this is how you start working on any repo on a new machine. A **fork** is different: it happens on the website (the Fork button) and creates `github.com/your-username/notes-app` — a GitHub-side copy you own and can push to; you then clone YOUR fork.

What you will see: the cloned folder is a complete repo — log, branches, everything — not a zip of files. The test that separates them in an interview: "clone copies a repo to your computer; fork copies it to your GitHub account."

Quick recap

- Create the GitHub repo empty → `git remote add origin url` → `git push -u origin main` once → plain `git push` / `git pull` forever after.
- README.md is the front door: what it is, how to run it, what is inside — written for a stranger with ten seconds.
- Clone = repo → your computer (push only with permission). Fork = repo → your GitHub account (push freely, propose changes back).

Common mistakes

1. **Ticking "Add a README" on GitHub, then pushing a local repo** — Git refuses: "failed to push some refs... fetch first", because the two histories are unrelated. The fix for a brand-new project: create the GitHub repo completely empty; if it already has commits, clone THAT repo and work inside the clone instead.
2. **Pushing to the wrong account or repo** — the remote was copied from a tutorial. The fix: `git remote -v` before the first push, every time; a wrong address is repaired with `git remote set-url origin <correct-url>`, no re-clone needed.
3. **An empty or missing README on a public repo** — visitors see a bare file list and leave; the project looks abandoned even if the code is good. The fix: README in the same commit as the first push, and update it whenever the run steps change.

Practice

1. Push one of your existing practice repos to GitHub following Example 1 exactly: empty repo on the site, `remote add`, `remote -v` check, `push -u`. Then open the repo page and count the commits — compare with `git log --oneline`.
2. Write (or upgrade) a README for that repo using Example 2 as the skeleton: title, one-line explanation, run steps, what is inside. Push it and view the rendered page — read it as a stranger: does it answer the three questions in ten seconds?

Practice quiz

1. You plan to push an existing local repo to a new GitHub repo. Should you tick "Add a README" when creating it?

- Yes, every repo needs a README at once
- No, create it empty; the two histories would collide and Git would refuse the push
- It makes no difference
- Yes, but only for private repos

Answer: No, create it empty; the two histories would collide and Git would refuse the push

2. What is origin, really?

- The GitHub website itself
- Just a nickname for the repo's address, saved by git remote add origin <url>; git remote -v shows the fetch and push addresses any time
- Your GitHub username
- A branch name

Answer: Just a nickname for the repo's address, saved by git remote add origin <url>; git remote -v shows the fetch and push addresses any time

3. What is the difference between clone and fork?

- They are the same
- Clone downloads a repo to your computer (you can push only with permission); fork makes your own copy on GitHub under your account, which you CAN push to, then propose changes back with a pull request
- Fork downloads, clone copies on GitHub
- Clone is for public repos, fork for private ones

Answer: Clone downloads a repo to your computer (you can push only with permission); fork makes your own copy on GitHub under your account, which you CAN push to, then propose changes back with a pull request

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/github-deep-dive · More free tools and notes at https://scopetreasure.com