🧭 Git & GitHub Roadmap (2026)

Beginner → Professional | Job-Ready | Open-Source & Enterprise Focused

🎯 Outcome: By the end, you can
- *Work confidently in teams*
- *Contribute to open source*
- *Pass Git interviews*
- *Use GitHub like a professional engineer*
- *Leverage AI (Copilot) responsibly*

🗓️ Suggested Timeline (16 Weeks)

Phase	Level	Duration
Phase 1	Beginner	Weeks 1–4
Phase 2	Intermediate	Weeks 5–8
Phase 3	Advanced	Weeks 9–12
Phase 4	Professional	Weeks 13–16

🟢 PHASE 1 — BEGINNER (Weeks 1–4)

Goal: Stop fearing Git. Start thinking in commits.

1️⃣ Core Concepts

- What Git **is vs is not**
- Distributed version control
- Repository, commit, branch
- Working Tree → Index (Staging) → Repository
- Git vs GitHub (critical interview question)

Garvit

2️⃣ Essential Commands & Workflows

```
git init
git clone <url>
git status
git add .
git commit -m "message"
git log --oneline
git branch
git checkout -b feature-x
git merge feature-x
git push
git pull
```

3️⃣ Real-World Use Cases

- Tracking assignments/projects
- Safe experimentation using branches
- Collaborating via GitHub repos
- Version history instead of ZIP files

4️⃣ Common Mistakes ❌ & Best Practices ✅

Mistake	Best Practice
Committing everything	Stage intentionally
Vague messages	Use meaningful commit messages
Working on main	Create feature branches
Ignoring .gitignore	Always configure it

5️⃣ Mini Practice Tasks 🧪

- Create a repo for **Notes App**
- Make 5 atomic commits
- Create a feature branch and merge it
- Push to GitHub and share repo link

🟡 PHASE 2 — INTERMEDIATE (Weeks 5–8)

Goal: Work like a team engineer, not a solo coder.

1️⃣ Core Concepts

- Branching models
- Merge conflicts (why they happen)
- Remote tracking branches
- Fork vs Clone
- Pull Requests (PRs)

2️⃣ Branching Strategies

Strategy	Used By	When to Use
Git Flow	Legacy enterprises	Release-heavy projects
GitHub Flow	Startups, OSS	Continuous delivery
Trunk-Based	FAANG, DevOps teams	High velocity CI/CD

3️⃣ Merge vs Rebase (CRITICAL)

Merge	Rebase
Preserves history	Cleans history
Safe for shared branches	Avoid on public branches
Preferred in teams	Preferred before PR

Rule of Thumb: ***Rebase locally, Merge remotely***

Gervit

4️⃣ Undoing Mistakes (Must-Know)

```
git restore file.txt
git reset --soft HEAD~1
git reset --hard HEAD~1
git revert <commit>
git reflog
```

5️⃣ GitHub Essentials

- Repositories & README.md
- Issues & labels
- Pull Requests
- Fork → Change → PR flow
- Code Reviews

6️⃣ Mini Practice Tasks 🧪

- Fork an open-source repo
- Fix a typo or small bug
- Open a PR with description
- Resolve a merge conflict locally

Garvit

🔵 PHASE 3 — ADVANCED (Weeks 9–12)

Goal: Understand Git deeply. Debug fearlessly.

1️⃣ Git Internals (Interview Gold 🏆)

Concept	Meaning
HEAD	Pointer to current commit
Index	Staging area
Blob	File content
Tree	Directory structure
Commit	Snapshot + metadata
Refs	Branch pointers

🔍 *git cat-file, git show, git ls-tree*

2️⃣ Advanced Git Commands

```
git cherry-pick
git stash / git stash pop
git rebase -i
git bisect
git worktree
```

3️⃣ GitHub Actions (CI/CD)

Beginner → Advanced

- Lint on PR
- Run tests automatically
- Build & deploy
- Matrix builds
- Secrets management
- Reusable workflows

CI is mandatory, not optional.

4️⃣ GitHub Security (Non-Negotiable in 2026)

- Dependabot
- CodeQL scanning
- Secrets scanning
- Branch protection rules

5️⃣ Release Management

- Semantic Versioning (SemVer)
- Tags & Releases
- Release notes
- Hotfix workflows

6️⃣ Mini Practice Tasks 🧪

- Add CI pipeline with GitHub Actions
- Enable Dependabot
- Create v1.0.0 release
- Use git bisect to find a bug

🔴 PHASE 4 — PROFESSIONAL (Weeks 13–16)

Goal: Operate like a Staff/DevOps engineer.

1️⃣ Git for Teams & Enterprise

- Monorepos vs Polyrepos
- Protected branches
- CODEOWNERS
- Commit signing (GPG)
- Audit logs

Garvit

2️⃣ Pull Request Best Practices (Promotion-Critical)

✅ Small PRs
✅ Clear description
✅ Screenshots / logs
✅ Linked issues
✅ Reviewer-friendly commits

❌ Huge PRs
❌ “Fix bug” messages
❌ No tests

3️⃣ Open-Source Contribution Workflow

1. Find issue (good first issue)
2. Discuss before coding
3. Fork & branch
4. Follow contribution guide
5. Clean commits
6. PR with context

4️⃣ GitHub Copilot & AI (2026 Perspective 🤖)

- Use Copilot for:
  - Commit messages
  - PR summaries
  - Test generation
- NEVER:
  - Blindly accept AI code
  - Commit secrets
  - Skip reviews

Garvit

5️⃣ Mini Practice Tasks 🧪

- Contribute to a real OSS project
- Set PR template
- Enforce branch rules
- Review someone else's PR

🏢 How Git Is Used in FAANG & Startups

Company Type	Git Style
FAANG	Trunk-Based + CI enforced
Startups	GitHub Flow
OSS	Fork + PR model
Enterprises	Protected Git Flow

🚫 GitHub Mistakes That Block Promotions

- Force-pushing shared branches
- No tests in PRs
- Breaking main branch
- Poor commit hygiene
- Ignoring code reviews

🎤 Git Interview Questions

- Explain Git internals
- Merge vs Rebase scenario
- Recover deleted commit
- Resolve conflict strategy
- CI/CD with GitHub Actions
- Secure secrets in pipelines
- How Copilot fits into workflow

Garvit

👷 Portfolio Projects (Must-Have)

Project	What It Proves
Open-Source PRs	Collaboration
CI-enabled App	DevOps readiness
Versioned Releases	Professional workflow
Monorepo Project	Scale thinking

Final Mentor Advice

Git is not a tool — it's a career skill.
Mastering Git separates coders from engineers.

👉 **Follow this blog**

👉 **Visit TechSkillForge**

👉 **Follow @pythonquizhub on Instagram**

👉 **Follow GitHub | LinkedIn**

Garvit