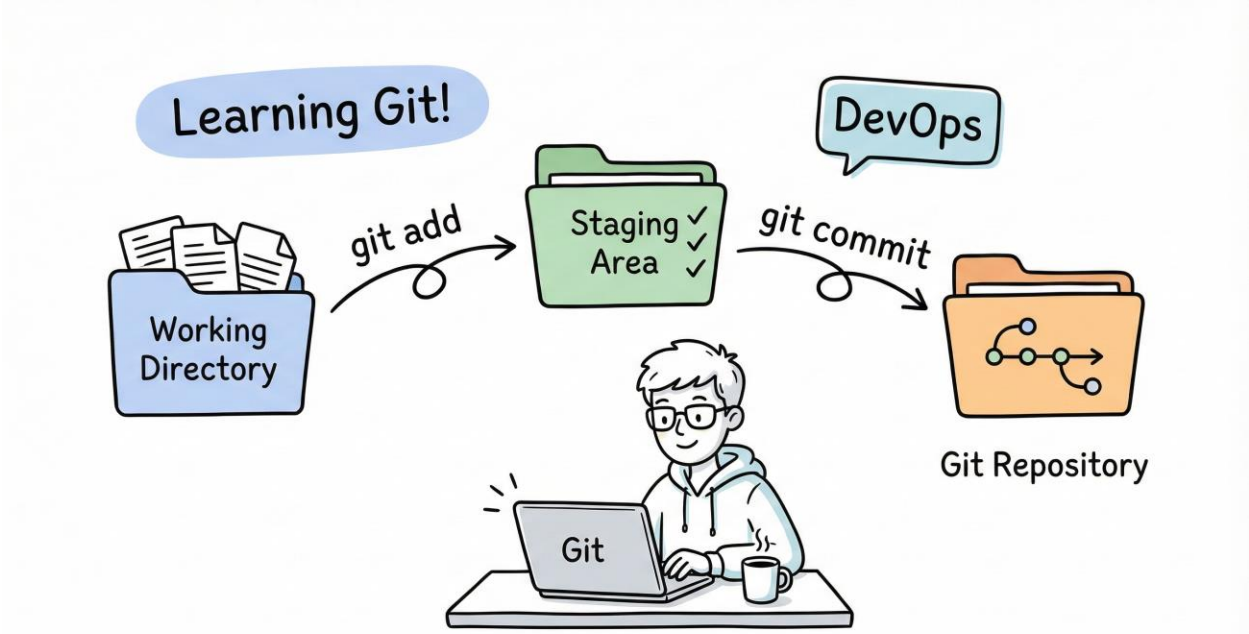
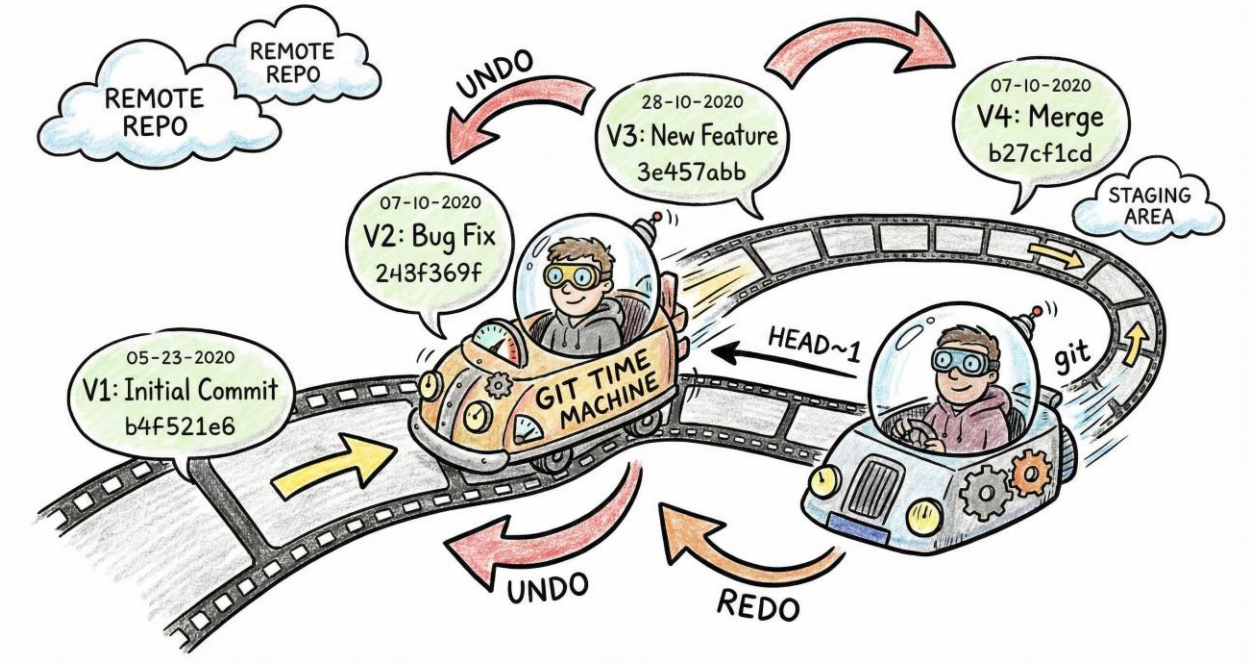
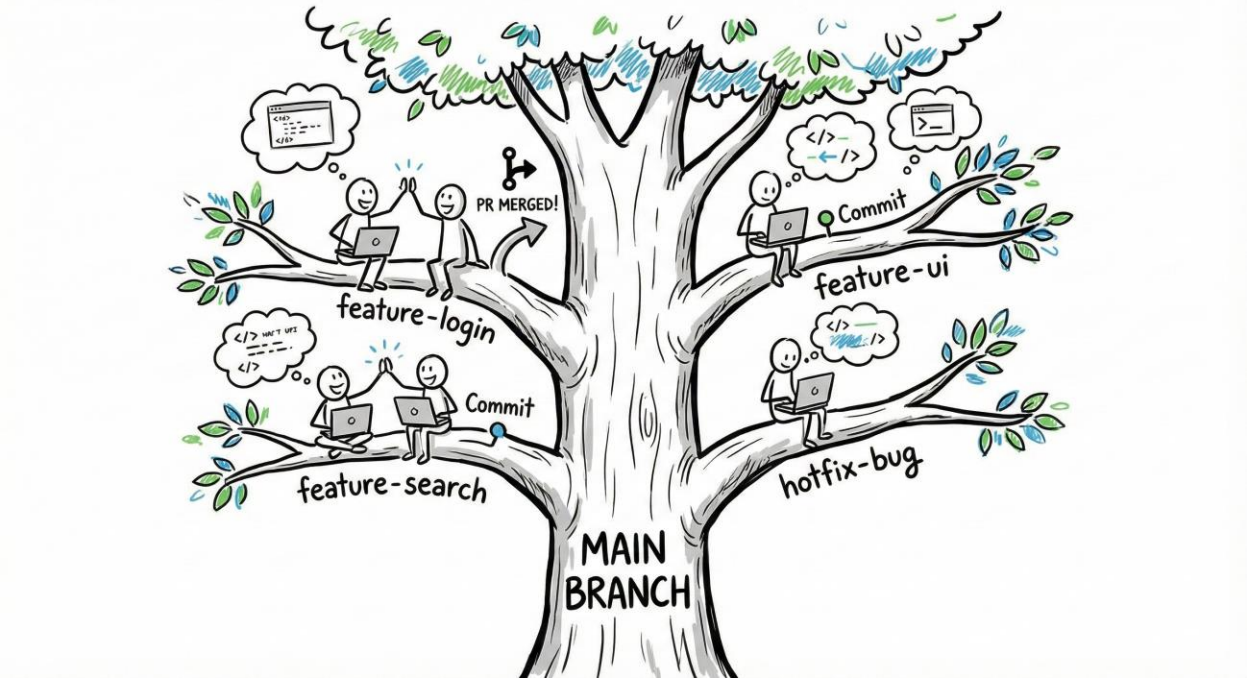
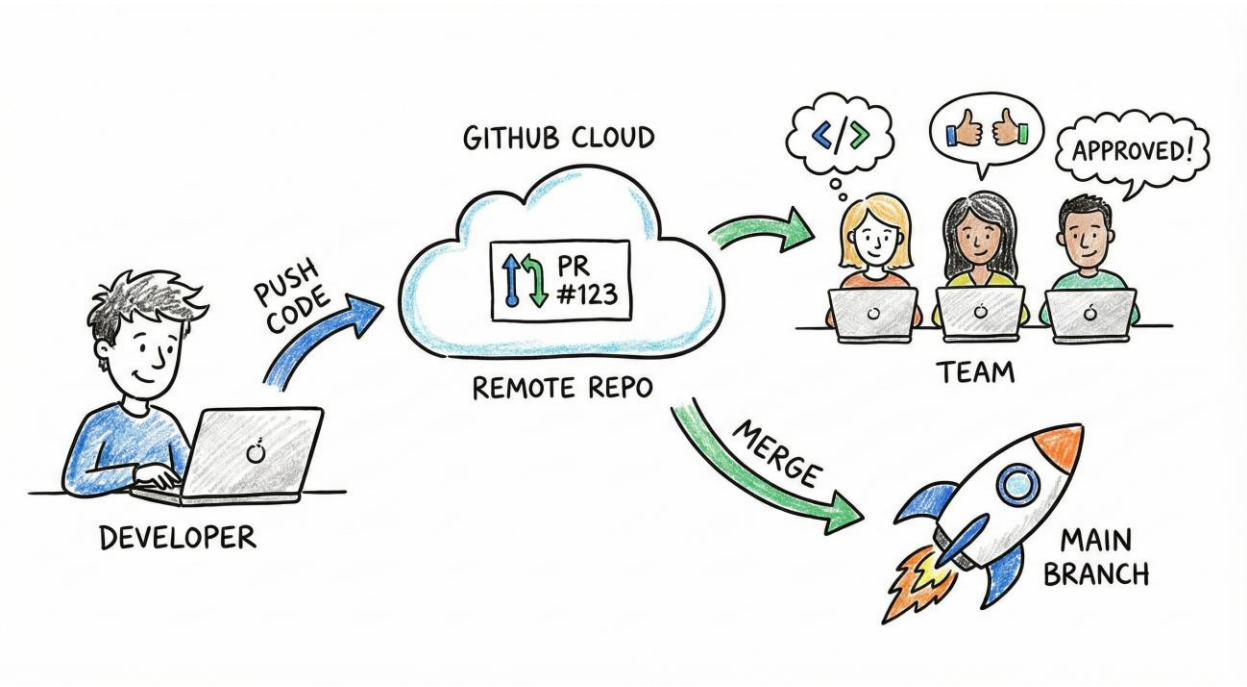
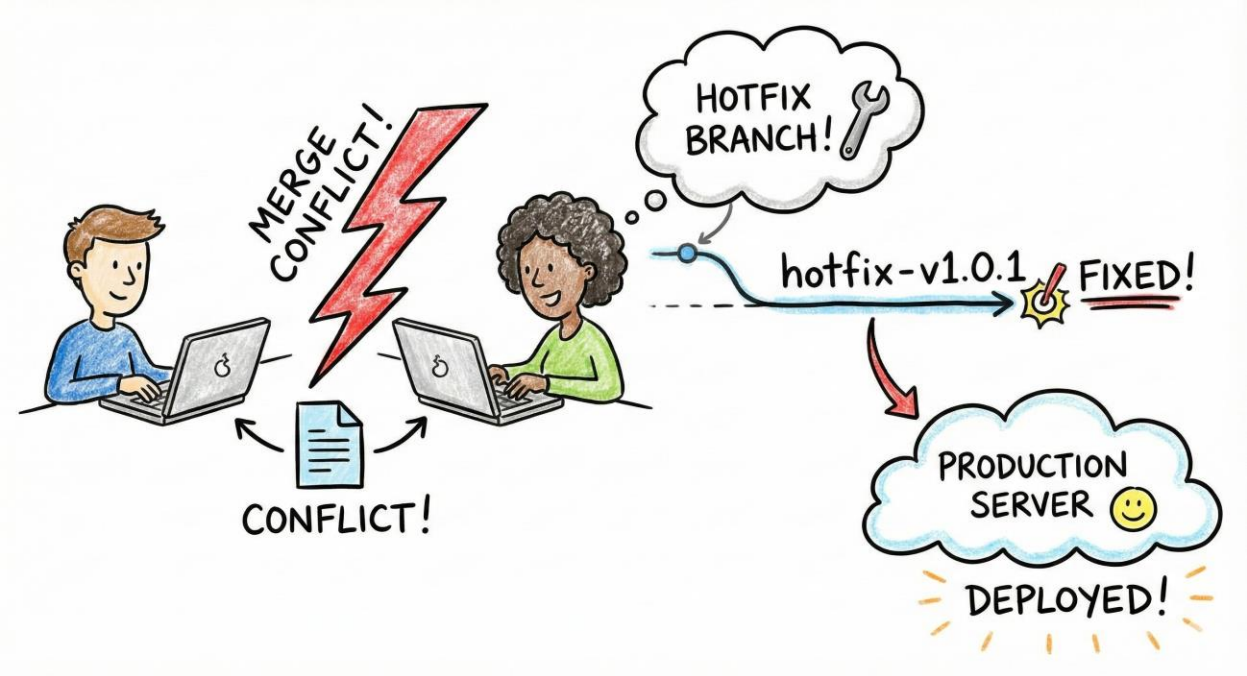

Learning Git!
DevOps
Working Directory
git add
Staging Area
git commit
Git Repository
Git


REMOTE REPO
REMOTE REPO
UNDO
28-10-2020
V3: New Feature
3e457abb
07-10-2020
V4: Merge
b27cf1cd
STAGING AREA
07-10-2020
V2: Bug Fix
243f369f
05-23-2020
V1: Initial Commit
b4f521e6
GIT TIME MACHINE
HEAD~1
git
UNDO
REDO


PR MERGED!
feature-login
Commit
feature-ui
Commit
feature-search
hotfix-bug
MAIN
BRANCH


GITHUB CLOUD
APPROVED!
PR
#123
PUSH
CODE
REMOTE REPO
TEAM
MERGE
DEVELOPER
MAIN
BRANCH



📅 DAY 1 — How Git Actually Works

What you must understand

Most people think Git is a tool.

In reality, Git is a **database of snapshots**.

Every time you commit, Git stores:

- A snapshot of files
- Who made the change
- When
- What changed
- Parent commit

Core Concepts

Concept	Meaning
Working Directory	Your project folder
Staging Area	Files ready for commit
Local Repo	Git database on your laptop
Remote Repo	GitHub / Azure Repos
Commit	A snapshot in time
Hash	Unique ID of a commit

Commands to Master

git init
git status
git add
git commit
git log
git show
git diff

What a DevOps Engineer should think

“What changed, who changed it, and can I go back?”

📅 DAY 2 — Undo, Fix & Rewrite History

In real companies, mistakes happen every day.

You must know:

- How to undo
- How to recover
- How to clean history

Important Commands

Command	Why it exists
git reset --soft	Undo commit but keep changes
git reset --hard	Delete commit completely
git revert	Create reverse commit (safe)
git reflog	Recover lost commits
git rebase -i	Clean history

Industry Rule

- reset is for your laptop
- revert is for production

📅 DAY 3 — Branching Like Real Teams

Companies never work on main directly.

They use:

- Feature branches
- Dev branch
- Release branches

Branch Concepts

Branch	Purpose
main	Production
dev	Integration
feature	New work
hotfix	Emergency fix

Commands

git branch
git checkout
git switch
git merge
git log --graph

Why branches exist

To allow 100 developers to work without breaking each other.

📅 DAY 4 — GitHub & Team Collaboration

GitHub is not just storage.

It is:

- Code review system
- Approval gate
- Audit trail

What you must know

- Repositories
- Forks
- Pull Requests
- Reviews
- Merges

Commands

git remote
git push
git pull
git fetch
git clone
git revert

DevOps Reality

No code goes to production without Pull Request.

📅 DAY 5 — Conflicts, Releases & Production Flow

This is where most freshers fail.

Conflicts happen when:

Two people change same lines.

You must:

- Understand conflict markers
- Decide which code to keep
- Test after merge

Release Management

Concept	Why
Tags	Mark stable versions
Hotfix	Emergency production fix
Rollback	Go back safely

Commands

git pull
git merge
git tag
git checkout -b
git stash

🎓 If you master these 5 days

You are no longer “Git user”.

You are a **version control engineer**.

You can:

- Work in any DevOps team
- Handle real production issues
- Survive audits & failures