📘 Git Bible for Developers

A complete, practical guide with commands, workflows, use-cases, and solutions

---

1. What is Git (In One Line)

Git is a **distributed version control system** that tracks code changes, enables collaboration, and allows you to safely experiment, rollback, and release software.

---

2. Core Git Concepts (Must Know)

Concept	Meaning
Repository (repo)	Project tracked by Git
Commit	Snapshot of changes
Branch	Independent line of development
HEAD	Current pointer to branch/commit
Remote	Online repo (GitHub/GitLab)
Staging Area	Where changes wait before commit
Merge	Combine branches
Rebase	Reapply commits on another base

---

3. Initial Setup (One Time)

```
git config --global user.name "Your Name"
git config --global user.email "you@email.com"
git config --global init.defaultBranch main
git config --global core.editor nano
```

Check config:

```
git config --list
```

---

4. Starting a Repository

4.1 New Project

```
git init
git status
```

4.2 Existing Project (Clone)

```
git clone https://github.com/user/repo.git
cd repo
```

---

5. Daily Workflow (Golden Routine)

```
git status
git add .
git commit -m "feat: meaningful message"
git pull --rebase
git push
```

---

6. File Operations

Track a File

```
git add file.dart
```

Untrack a File (keep locally)

```
git rm --cached file.env
```

Ignore Files

`.gitignore`

```
.env
build/
*.log
```

---

7. Commit Rules (VERY IMPORTANT)

Commit Message Convention

```
type(scope): short description
```

Types: - `feat` → new feature - `fix` → bug fix - `refactor` → code improvement - `docs` → documentation - `chore` → tooling / config - `test` → tests

Example:

```
git commit -m "feat(auth): add login validation"
```

---

8. Branching Strategy (Professional)

Main Branches

- `main` → production ready
- `develop` → integration branch

Supporting Branches

- `feature/login-ui`
- `bugfix/cart-crash`
- `hotfix/payment-fix`

---

9. Branch Commands

Create & Switch

```
git checkout -b feature/login
```

Switch Branch

```
git checkout develop
```

List Branches

```
git branch -a
```

Delete Branch

```
git branch -d feature/login
git push origin --delete feature/login
```

---

10. Merging (Safe Way)

```
git checkout develop
git pull
git merge feature/login
```

If conflict:

```
# Fix manually
git add .
git commit
```

---

11. Rebase (Clean History)

```
git checkout feature/login
git rebase develop
```

Abort rebase:

```
git rebase --abort
```

---

12. Undo & Fix Mistakes (MOST USED)

Undo Last Commit (Keep Code)

```
git reset --soft HEAD~1
```

Undo Last Commit (Delete Code)

```
git reset --hard HEAD~1
```

Undo File Change

```
git checkout -- file.dart
```

Fix Commit Message

```
git commit --amend
```

---

13. Stash (Save Work Temporarily)

```
git stash
git stash list
git stash apply
git stash drop
```

---

14. Remote Handling

Add Remote

```
git remote add origin https://github.com/user/repo.git
```

Check Remote

```
git remote -v
```

Change Remote URL

```
git remote set-url origin NEW_URL
```

---

15. Pull vs Fetch

Command	Meaning
git fetch	Download changes only
git pull	Fetch + merge
git pull --rebase	Fetch + rebase (recommended)

---

16. Tags (Releases)

```
git tag v1.0.0
git push origin v1.0.0
```

Annotated tag:

```
git tag -a v1.1.0 -m "Stable release"
```

---

17. Logs & History

```
git log --oneline --graph --all
git show COMMIT_ID
git blame file.dart
```

---

18. Common Real-World Scenarios & Solutions

🔩Pushed to Wrong Branch

```
git checkout correct-branch
git cherry-pick COMMIT_ID
```

🔩Merge Conflict Panic

```
git status
# Fix conflicts manually
git add .
git commit
```

🔩Accidentally Deleted Code

```
git reflog
git checkout COMMIT_ID
```

🔩Local Branch Behind Remote

```
git pull --rebase
```

---

19. Team Workflow (Recommended)

1. Create feature branch
2. Commit frequently
3. Push branch
4. Open Pull Request
5. Code review
6. Merge to develop
7. Release to main

---

20. Git for Flutter / Mobile Devs

```
build/
.dart_tool/
android/.gradle/
ios/Pods/
```

Never commit: - API keys - `.env` - Build folders

---

21. Golden Rules (Tattoo These)

- ❌Never commit broken code

- ❌Never work directly on `main`
- ✅Commit small, logical changes
- ✅Pull before push
- ✅Write meaningful messages

---

22. Power User Aliases (Optional)

```
git config --global alias.st status
git config --global alias.cm commit
git config --global alias.br branch
git config --global alias.co checkout
```

---

23. Final Advice

> Git is not about commands — it is about **confidence**. If you can recover from mistakes, you truly know Git.

🚫*This document is designed to be your lifelong Git reference.*