Git Basics Practice Lab

Git Basics — Step-by-step Lab (Windows CMD Edition)

Prerequisites

- Install Git for Windows → https://git-scm.com/download/win.
- Open **Git CMD** (or Command Prompt) and check:

```
git --version
```

Quick setup: Create a starter project

```
mkdir git-basics-lab
cd git-basics-lab
git init
echo # Git Basics Lab > README.md
(
echo /bin/
echo /.env
echo *.log
) > .gitignore
git add README.md .gitignore
git commit -m "chore: initial repo with README and .gitignore"
```

Check status & log:

```
git status
git log --oneline --decorate --graph
```

Exercises

1. Add a feature (branching + commit)

```
git checkout -b feature/add-greeting
echo console.log("Hello from feature"); > greet.js
git add greet.js
git commit -m "feat: add greet.js with console message"

git checkout main
git merge --no-ff feature/add-greeting
```

Verify:

```
type greet.js
git log --oneline --graph --decorate
```

---

2. Simulate and resolve a merge conflict

1. Create a file in two branches that conflict:

```
echo line: original > conflict.txt
git add conflict.txt
git commit -m "chore: add conflict file base"

git checkout -b feature/A
echo line: A change > conflict.txt
git commit -am "feat(A): change conflict line to A"

git checkout main
git checkout -b feature/B
echo line: B change > conflict.txt
git commit -am "feat(B): change conflict line to B"
```

2. Merge branches:

```
git checkout main
git merge feature/A
git merge feature/B
```

👉 Conflict will appear in `conflict.txt`. Open it in **Notepad**:

```
notepad conflict.txt
```

Manually edit to desired resolution (remove <<<<<<<, =======, >>>>>>> markers).

3. Stage and commit:

```
git add conflict.txt
git commit -m "fix: resolve merge conflict between A and B"
```

---

3. Undo mistakes safely

```
:: Amend last commit
echo extra note >> README.md
git add README.md
git commit --amend -m "chore: initial repo with README and .gitignore
(amended)"

:: Undo last commit but keep changes
git reset --soft HEAD~1

:: Discard local changes
git reset --hard HEAD

:: Safely undo a pushed commit
git revert <commit-hash>
```

---

4. Remote basics (push/pull/clone)

1. Create an empty repo on GitHub (e.g., `git-basics-lab`).
2. Connect and push:

```
git remote add origin https://github.com/YOURUSER/git-basics-lab.git
git push -u origin main
```

3. Clone into a new folder:

```
cd ..
git clone https://github.com/YOURUSER/git-basics-lab.git clone-copy
cd clone-copy
```

4. Create a branch, commit and push:

```
git checkout -b fix/typo
notepad README.md   :: make some edits and save
git commit -am "fix: typos in README"
git push -u origin fix/typo
```

---

5. Stash (work in progress)

```
notepad greet.js   :: make changes but don't commit
git stash push -m "WIP: experiment"
git stash list

git checkout main
git stash pop
```

---

6. Tagging releases

```
git tag -a v1.0 -m "Release v1.0"
git show v1.0
git push origin v1.0
```

---

7. Cherry-pick & interactive rebase

```
git checkout main
git cherry-pick <commit-hash>

git rebase -i HEAD~3
```

👉 In the editor, mark commits with `s` (squash) or `f` (fixup).

---

8. Inspecting history, blame and diffs

```
git log --oneline --graph --decorate --all
git log --stat
git show <commit-hash>
git diff
git diff --staged
git blame README.md
```

Challenge Scenarios

1. **Feature + PR flow:** Create `feature/login`, edit files, push, open Pull Request on GitHub, update branch after review.
2. **Accidental commit to main:** Use `git reset` (if not pushed) or `git revert` (if pushed) to undo.
3. **Concurrent conflict:** Two branches edit the same file → resolve conflict and commit.
4. **Clean history:** Make 4 commits on a feature branch, squash into 1 with `git rebase -i`, then merge to `main`.

Quick CMD Git Cheat Sheet

```
git init
git clone <url>
git status
git add <file>
git commit -m "msg"
git log --oneline --graph
git branch
git checkout -b <branch>
git merge <branch>
git fetch
git pull
git push
git stash / git stash pop
git revert <sha>
git reset --hard <sha>
git tag -a v1.0 -m "msg"
git diff / git blame
```