**Atchaya V - 6418561**

# Week 8

# Git Commit History

## 1. Introduction to Git Commit History

Git commit history is the chronological record of all changes made to a repository.

Each commit contains:

- A unique SHA-1 hash (e.g., 3f3a9c4a2...)

- Author details (name, email)

- Timestamp

- Commit message

- Changes snapshot (diff)

Understanding commit history is essential for:

- Tracking changes

- Collaborating in teams

- Rolling back to previous states

- Debugging issues

## 2. Viewing Commit History

2.1 Basic History:

git log

Shows commits in reverse chronological order.

2.2 Custom Log Formats:

git log --oneline

git log --pretty=format:"%h - %an, %ar : %s"

%h → Short hash

%an → Author name

%ar → Relative date

%s → Commit message

## 3. Filtering Commit History

3.1 Show Specific File History:

git log -- <file>

3.2 Search Commits by Keyword:

git log --grep="authentication"

3.3 Date Range Filter:

git log --since="2025-01-01" --until="2025-08-01"

3.4 Author Filter:

git log --author="John Doe"

## 4. Commit History Visualization

4.1 Branch Graph:

git log --oneline --graph --all --decorate

4.2 GUI Tools:

- gitk

- VS Code Source Control

- GitKraken

- Sourcetree

## 5. Working with Commit History

5.1 Show Changes in a Commit:

git show <commit-hash>

5.2 Compare Commits:

git diff <commit1> <commit2>

5.3 Checkout a Previous Commit:

git checkout <commit-hash>

## 6. Modifying Commit History

⚠ Use these commands with caution — rewriting history affects collaborators.

6.1 Amend Last Commit:

git commit --amend

6.2 Reorder/Combine Commits (Interactive Rebase):

git rebase -i HEAD~3

6.3 Remove a Commit:

git rebase -i HEAD~N

## 7. Reverting and Resetting

7.1 Revert a Commit:

git revert <commit-hash>

7.2 Reset to a Previous Commit:

git reset --hard <commit-hash>

git reset --soft <commit-hash>

## 8. Practical Examples

Example 1: Find Who Changed a Line:

git blame <file>

Example 2: Show Commit Stats:

git log --stat

Example 3: History of a Function:

git log -L :functionName:file

## 9. Best Practices for Commit History

1. Write meaningful commit messages.

2. Keep commits small and focused.

3. Use feature branches.

4. Review history before pushing:

git log origin/main..HEAD

## 10. Advanced History Commands

Show first commit:

git log --reverse

Show merge commits only:

git log --merges

Show commits affecting a directory:

git log -- path/to/dir

## 11. Using Git Bisect for Debugging

The `git bisect` command helps in identifying the commit that introduced a bug by performing a binary search through the commit history.

Basic usage:

1. Start bisect:

git bisect start

2. Mark the current commit as bad:

git bisect bad

3. Mark a known good commit:

git bisect good <commit-hash>

4. Git will check out a commit halfway between good and bad. Test and mark each as good or bad until the faulty commit is found.

5. End bisect:

git bisect reset

## 12. Cherry-Picking Commits

Cherry-picking allows you to apply specific commits from one branch into another without merging the entire branch.

Command:

git cherry-pick <commit-hash>

Options:

- `-x` : Adds a note about the original commit.

- `--no-commit` : Applies changes without committing them immediately.

Example:

git cherry-pick a1b2c3d

## 13. Stashing with Commit Context

When working on a feature and you need to switch branches without committing, `git stash` temporarily saves your changes.

Common commands:

- Save changes:

git stash

- Save with message:

git stash save "Work in progress on login feature"

- View stashes:

git stash list

- Apply latest stash:

git stash apply

- Apply and remove latest stash:

git stash pop

## 14. Annotated Tags in History

Tags are references to specific points in Git history, often used for marking release versions.

Creating an annotated tag:

git tag -a v1.0 -m "Release version 1.0"

Pushing tags to remote:

git push origin v1.0

Listing all tags:

git tag

## 15. Exporting Commit History

Export commit history for reports or documentation:

Command:

git log --pretty=format:"%h | %an | %ad | %s" --date=short > commits.txt

You can also export to HTML or PDF using third-party tools like `gitstats` or `gource` for visual history representation.