History of Linux and Git by Linus Torvalds
&
Merge vs Rebase in Git

History of Linux

Origins of Linux (1991):

- Created by Linus Torvalds as a free OS kernel for x86 architecture.
- Announced in 1991 via the comp.os.minix newsgroup.

Famous post: "I’m doing a (free) operating system…"

- Licensed under the GNU GPL in 1992.
- Combined with GNU components to form GNU/Linux.
- Adopted for servers and enterprise environments by the mid-1990s.
- Powers most web servers, cloud infrastructure, and mobile devices (Android).

History of Git

Origins of Git (2005):

- Created by Linus Torvalds in 2005 after disputes over BitKeeper.
- Designed for speed, efficiency, and scalability.
- Fast performance and distributed nature.
- Branching and merging capabilities for large projects.
- GitHub launched in 2008, making Git more accessible.
- Widely used for version control in software development.

Git Merge

- Joins two or more development histories together
- Creates a merge commit that combines the history of the branches
- Preserves the complete history of both branches

How Merge Works

- Run *git merge feature-branch* on the main branch
- Git creates a new merge commit with two parent commits
- The merge commit preserves the complete history of both branches

Advantages of Merge

- Preserves History: original commit history from both branches is preserved
- Easy to Understand: straightforward way to bring changes together

Disadvantages of Merge

- Messier History: repeated merges can lead to a complex, branched history
- Additional Merge Commit: creates an extra commit even for simple changes

Git Rebase

- Re-applies changes onto a different base commit
- Rewrites the commit history
- Results in a linear history

How Rebase Works

- Run *git rebase main* on the feature branch.
- Git moves the feature branch commits on top of the main branch commits.
- The commit history is linear, without the extra merge commit.

Advantages of Rebase

- Cleaner History: results in a linear history
- No Merge Commit: no extra commit is created

Disadvantages of Rebase

- History Rewriting: rewrites the commit history
- Dangerous in Public Branches: can lead to confusion and conflicts

Key Differences

Feature	Merge	Rebase
Commit History	Preserves branch history with merge commit.	Rewrites history to appear linear.
Merge Commit	Creates a new merge commit.	No merge commit.
Use Case	For preserving the complete history in a project with collaborators.	For a cleaner, linear history.
Risk	Safe, but creates cluttered history.	Changes commit history, risk of conflicts in shared branches.

When to Use Merge vs. Rebase

- Use Merge when:
  - Preserving complete history of all branches.
  - Working in a team with multiple contributors.
  - Want an explicit record of branch integration.
- Use Rebase when:
  - Want a clean, linear commit history.
  - Working on a private branch or feature branch.
  - Updating feature branch with latest changes from main branch.