Version control & Git

Nguyễn Thanh Quân - ntquan@fit.hcmus.edu.vn

Agenda

1. Introduction to Version Control Systems
2. Introduction to Git
3. Introduction to GitHub
4. Git Merge vs Rebase
5. GitHub Actions

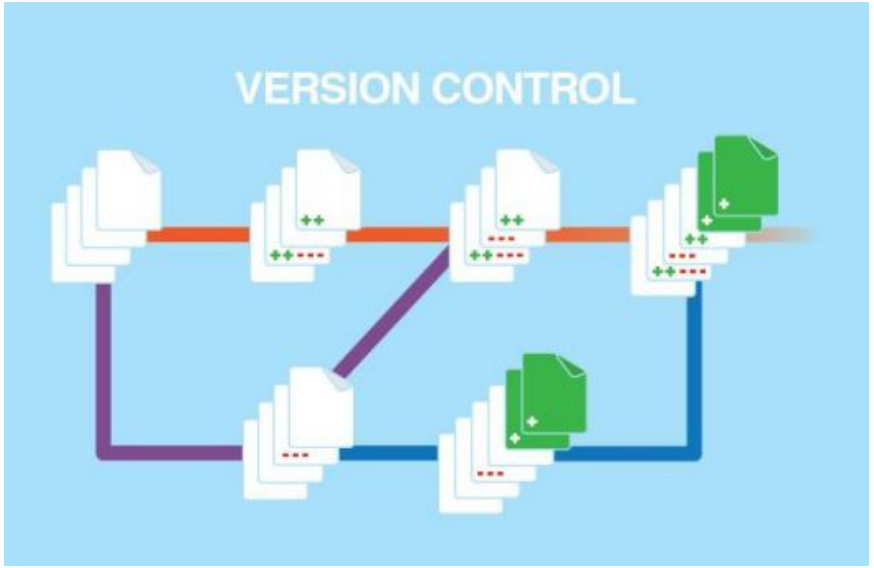
1. Introduction to Version Control Systems

A Version Control System (VCS) is a tool that helps track changes in source code or any files over time. This allows multiple people to work on a project without worrying about conflicts and makes it easy to revert to a previous version when needed.

1. Introduction to Version Control Systems

Why is VCS needed?

- **Tracking changes:** VCS stores all old and new versions of files, helping to monitor the development process of a project.
- **Code recovery**: Allows users to revert to previous versions if errors occur in the current version.
- **Effective collaboration**: Team members can work on the same project without worrying about code conflicts or data loss.
- **Branch management**: VCS allows the creation of multiple branches to develop different features in parallel and merge them when needed.



1. Introduction to Version Control Systems

Types of Version Control Systems (VCS):

- Local Version Control Systems
  - Stores versions of files on a local machine.
  - Simple but not suitable for team collaboration.
  - Example: RCS (Revision Control System).
- Centralized Version Control Systems (CVCS)
  - Uses a central server to store all versions.
  - Developers pull and push changes from/to the central repository.
  - Single point of failure: if the server crashes, data may be lost.
  - Examples: SVN (Apache Subversion), Perforce.
- Distributed Version Control Systems (DVCS)
  - Every developer has a complete copy of the repository.
  - Enables offline work and reduces dependency on a central server.
  - More robust against data loss.
  - Examples: Git, Mercurial.

2. Introduction to Git

What is Git?

- Git is a free and widely used distributed version control system, developed by Linus Torvalds in 2005. It is designed for easy source code management and supports team collaboration in software development.
- Git allows tracking the history of project changes, managing branches, and merging efficiently.

2. Introduction to Git

Basic Concepts in Git:

- **Repository (repo)**: A storage location that contains all source code files and the change history of a project.
    - **Local Repository**
    - **Remote Repository**
- **Commit**: Saves a version of the source code after making changes.
- **Branch**: Allows developing new features or fixing bugs without affecting the main codebase.
- **Merge**: Combines changes from one branch into another.
- **Clone**: Copies a repository from the server to a local computer.

2. Introduction to Git

Local Repository

- A Local Repository is a version of the repository stored on a user's personal computer.
- Each person working with Git has a clone of the repository on their local machine, allowing them to work offline and commit changes before syncing with a remote repository.
- Common Commands in Local Repository:
  - git init - Initialize a new Git repository.
  - git clone <repository_url> - Clone an existing repository to your local machine.
  - git status - Check the status of files in the working directory.
  - git add <file> - Stage a specific file for commit.
  - git add . - Stage all modified files for commit.
  - git commit -m "Message" - Commit changes with a message.
  - git log - View commit history.
  - git diff - Show differences between modified files and the last commit.
  - git checkout <branch> - Switch to another branch.
  - git branch - List all branches in the repository.

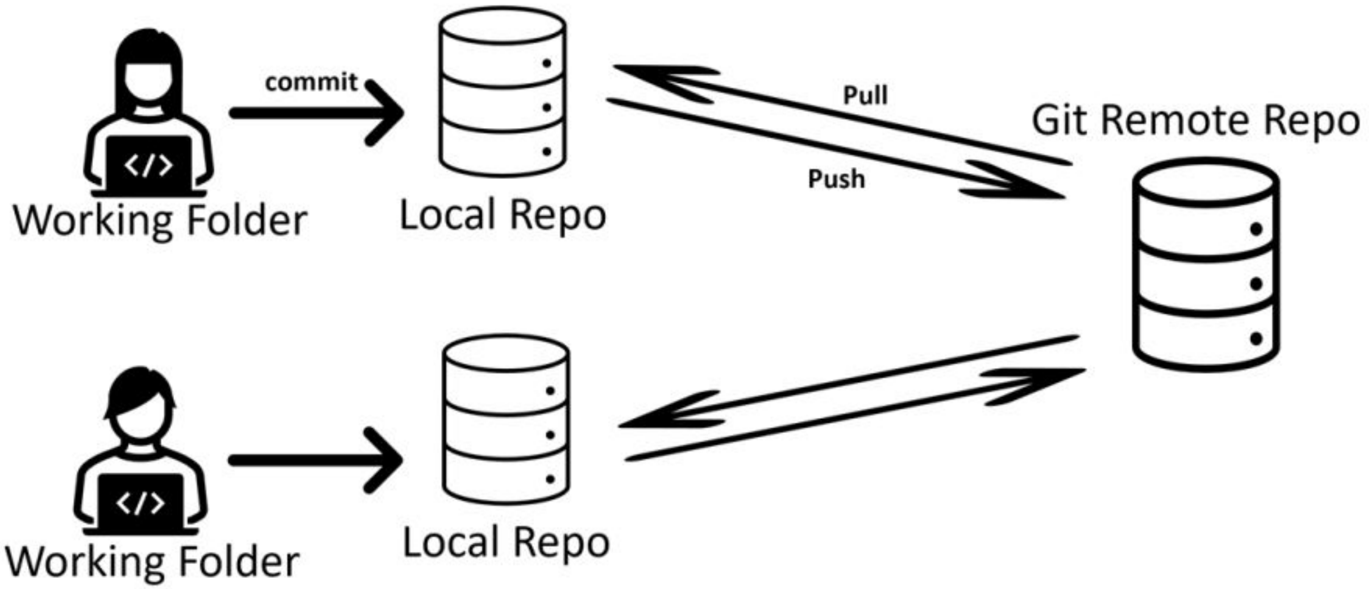
2. Introduction to Git

Remote Repository

- A Remote Repository is a version of the repository stored on a remote server (such as GitHub, GitLab, or Bitbucket).
- It enables multiple users to collaborate on the same project, share, and manage source code remotely.
- Common Commands for Working with a Remote Repository:
    - git remote add <name> <repository_url> - Add a remote repository.
    - git remote -v - List remote repositories.
    - git pull <remote> <branch> - Fetch and merge changes from a remote branch.
    - git push <remote> <branch> - Push local changes to a remote repository.
    - git fetch <remote> - Fetch updates from the remote repository without merging.
    - git merge <branch> - Merge fetched changes into the current branch.
    - git push -u <remote> <branch> - Push changes and set the upstream branch.
    - git clone <repository_url> - Clone a remote repository to your local machine.

2. Introduction to Git

Remote Repository vs Local Repository



3. Introduction to GitHub

What is GitHub?

- GitHub is a web-based platform for hosting and managing source code using the Git version control system. It allows developers to store code, track changes, and collaborate with others on the same project.
- GitHub provides an intuitive interface and powerful tools to support source code management and teamwork, helping teams work more efficiently.

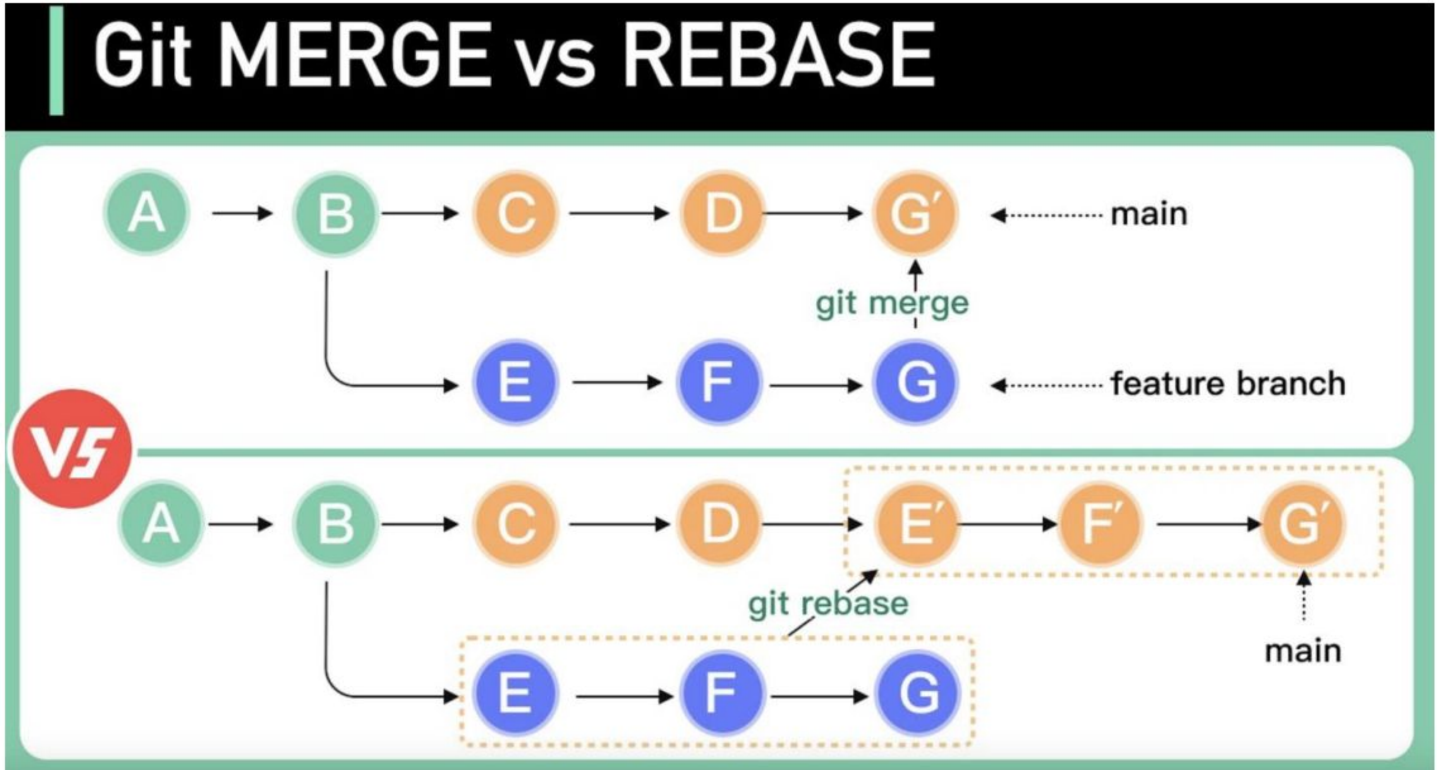
3. Introduction to GitHub

Each repository on GitHub can be either public (accessible to everyone) or private (restricted access), depending on the project's purpose and requirements.

Key Features of a GitHub Repository:

- **Code**: Stores the project's source code.
- **Commits**: Tracks the history of changes in the repository.
- **Branches**: Manages parallel development versions of the code.
- **Pull Requests**: A tool for submitting merge requests from a feature branch to the main branch.
- **Issues**: Used for tracking bugs, feature requests, or project tasks.

4. Git Merge vs Rebase



5. GitHub Actions

Video: https://www.youtube.com/watch?v=1oJQRlz1v94

Thank you