INTRODUCTION TO GIT &VERSION CONTROL

Seminar Presentation by Shrishail R Biradar , 2BA24MC042, Department of MCA, BEC BGK

What is Version Control?

- Version Control is a system that records the history of changes made to files over time. It allows developers to:
- - Track who modified what and when.
- - Revert to previous versions in case of mistakes or data loss.
- - Work collaboratively without overwriting each other's work.
- - Keep a full log of project development from beginning to end.

- Version Control Systems (VCS) are essential in modern software development, especially for managing source code and documentation.

Why Version Control is Important

- Without VCS, development becomes risky and disorganized. Here's why it's crucial:
- - Team Collaboration: Multiple developers can work on the same codebase simultaneously without conflict.
- - Release Management: Helps in tagging versions (e.g., v1.0, v2.0) for stable releases and tracking bugs.
- - Safe Experimentation: Developers can create branches to test new features without breaking the main project.
- - Data Protection: VCS acts like a backup system; even accidental deletions can be recovered.

Types of Version Control Systems

- 1. Local VCS (LVCS):
- - Stores changes in a local database.
- - Simple and fast, but limited to one machine.
- - Examples: RCS (Revision Control System).

- 2. Centralized VCS (CVCS):
- - A single server stores all versioned files.
- - Developers pull and push changes from/to this central server.
- - Risk: If the server crashes, everything is lost.
- - Examples: CVS, Subversion (SVN).

- 3. Distributed VCS (DVCS):
- - Every developer has a full copy of the repository.
- - Changes can be committed offline and synced later.
- - Reliable and ideal for open-source projects.
- - Examples: Git, Mercurial, Bazaar.

Centralized vs. Distributed VCS

- Feature | Centralized VCS | Distributed VCS
- ---|---|---
- Repository Location | Single central server | Full repo on every machine
- Offline Work | Not possible | Fully possible
- Speed | Slower | Faster
- Collaboration | Limited | Easier and safer
- Examples | CVS, SVN | Git, Mercurial

- DVCS is more robust, especially for remote and large teams.

Introduction to Git

- Git is the most widely used Distributed Version Control System (DVCS). Key facts:
- - Developed by Linus Torvalds (creator of Linux) in 2005.
- - Created to manage the Linux kernel codebase.
- - Designed for performance, flexibility, and data integrity.
- - Free, open-source, and platform-independent.
- - Used by small teams to large corporations like Google and Microsoft.

Key Features of Git

- - Distributed Architecture: Every developer has a complete history of the project.
- - Fast Performance: Especially for large projects.
- - Data Integrity: All commits are checksummed with SHA-1, ensuring safe storage.
- - Branching & Merging: Allows multiple lines of development.
- - Staging Area: Gives control over what goes into a commit.
- - Collaboration: Through services like GitHub and GitLab.

Installing Git

- 1. Visit: https://git-scm.com
- 2. Choose your OS (Windows/Linux/macOS).
- 3. Follow the installation wizard or use terminal commands.
- 4. Basic Configuration:
- git config --global user.name "Your Name"
- git config --global user.email "your@email.com"
- 5. Optionally install GUI tools (e.g., GitHub Desktop, SourceTree).

Basic Git Workflow

- A typical Git workflow looks like:
- 1. Edit Files: Make changes in your working directory.
- 2. Stage Changes: Add specific files using git add.
- 3. Commit Changes: Save the snapshot using git commit.
- 4. Push Changes: Upload local commits to a remote repo.
- 5. Pull Updates: Fetch new changes from remote repo.

- Each step adds structure, ensuring collaboration and change tracking.

Common Git Commands

- - git init: Start a new repository.
- - git clone <URL>: Copy a repo from remote.
- - git add <file>: Stage changes.
- - git commit -m "Message": Save changes.
- - git status: See current repo status.
- - git push: Upload changes to remote.
- - git pull: Download changes from remote.
- - git log: View commit history with messages and timestamps.

Branching and Merging

- Branching allows multiple versions of a project to exist simultaneously.
- - git branch new-feature: Create new branch.
- - git checkout new-feature: Switch to branch.
- - git merge main: Merge another branch into current.

- Benefits:
- - Develop features independently.
- - Fix bugs in isolation.
- - Merge only after testing.

Remote Repositories

- Remote repositories are hosted versions of your Git repo on the internet.
- Popular platforms:
- - GitHub: Most popular, supports social coding and collaboration tools.
- - GitLab: Offers built-in CI/CD.
- - Bitbucket: Supports private repos and Jira integration.

- Benefits:
- - Share your code globally.
- - Enable team collaboration.
- - Track issues, code reviews, and pull requests.

Benefits of Using Git

- - Improved Collaboration: Everyone works on their own branch.
- - Complete History: Every change is recorded with time and author.
- - Rollback Options: Revert to older versions easily.
- - Secure and Reliable: No data loss due to SHA-1 integrity checks.
- - Community Support: Massive online support, documentation, and tutorials.

Best Practices for Using Git

- - Write clear and descriptive commit messages.
- - Commit small changes often, not all at once.
- - Use branches for all features, fixes, and experiments.
- - Regularly pull changes to avoid merge conflicts.
- - Add .gitignore to skip tracking unnecessary files (e.g., logs, temp).
- - Always push your work to a remote for backup and collaboration.

Summary

- - Version Control is essential for tracking, collaboration, and rollback.
- - Git is the most widely adopted DVCS today.
- - It enhances productivity, safety, and organization in software development.
- - Mastering Git is a must for all modern developers, students, and professionals.

References

- - Git Official Documentation: https://git-scm.com/doc
- - Pro Git Book by Scott Chacon and Ben Straub: https://git-scm.com/book/en/v2
- - GitHub Guides: https://guides.github.com
- - freeCodeCamp Git Tutorials: https://www.freecodecamp.org/news/tag/git/
- - Codecademy Git Course: https://www.codecademy.com/learn/learn-git