Experiment No. 1

Title : Exploring Git Commands through Collaborative Coding.

Objective:

The objective of this experiment is to familiarise participants with essential Git concepts and commands, enabling them to effectively use Git for version control and collaboration.

Introduction:

Git is a distributed version control system (VCS) that helps developers track changes in their codebase, collaborate with others, and manage different versions of their projects efficiently. It was created by Linus Torvalds in 2005 to address the shortcomings of existing version control systems. Unlike traditional centralised VCS, where all changes are stored on a central server, Git follows a distributed model. Each developer has a complete copy of the repository on their local machine, including the entire history of the project. This decentralisation offers numerous advantages, such as offline work, faster operations, and enhanced collaboration.

Git is a widely used version control system that allows developers to collaborate on projects, track changes, and manage codebase history efficiently. This experiment aims to provide a hands-on introduction to Git and explore various fundamental Git commands. Participants will learn how to set up a Git repository, commit changes, manage branches, and collaborate with others using Git.

Key Concepts:

- Repository: A Git repository is a collection of files, folders, and their historical versions. It contains all the information about the project's history, branches, and commits.
- Commit: A commit is a snapshot of the changes made to the files in the repository at a specific point in time. It includes a unique identifier (SHA-1 hash), a message describing the changes, and a reference to its parent commit(s).
- Branch: A branch is a separate line of development within a repository. It allows developers to work on new features or bug fixes without affecting the main codebase. Branches can be merged back into the main branch when the changes are ready.
- Merge: Merging is the process of combining changes from one branch into another. It integrates the changes made in a feature branch into the main branch or any other target branch.
- Pull Request: In Git hosting platforms like GitHub, a pull request is a feature that allows developers to propose changes from one branch to another. It provides a platform for code review and collaboration before merging.

- Remote Repository: A remote repository is a copy of the Git repository stored on a server, enabling collaboration among multiple developers. It can be hosted on platforms like GitHub, GitLab, or Bitbucket.

Basic Git Commands:

- git init: Initialises a new Git repository in the current directory.
- git clone: Creates a copy of a remote repository on your local machine.
- git add: Stages changes for commit, preparing them to be included in the next commit.
- git commit: Creates a new commit with the staged changes and a descriptive message.
- git status: Shows the current status of the working directory, including tracked and untracked files.
- git log: Displays a chronological list of commits in the repository, showing their commit messages, authors, and timestamps.
- git branch: Lists, creates, or deletes branches within the repository.
- git checkout: Switches between branches, commits, or tags. It's used to navigate through the repository's history.
- git merge: Combines changes from different branches, integrating them into the current branch.
- git pull: Fetches changes from a remote repository and merges them into the current branch.
- git push: Sends local commits to a remote repository, updating it with the latest changes.

Materials:

- Computer with Git installed (https://git-scm.com/downloads)
- Command-line interface (Terminal, Command Prompt, or Git Bash)

Experiment Steps:

Step 1: Setting Up Git Repository

- Open the command-line interface on your computer.
- Navigate to the directory where you want to create your Git repository.
- Run the following commands:

git init

This initialises a new Git repository in the current directory.

Step 2: Creating and Committing Changes

- Create a new text file named "example.txt" using any text editor.
- Add some content to the "example.txt" file.
- In the command-line interface, run the following commands:

git status

- This command shows the status of your working directory, highlighting untracked files.

git add example.txt

- This stages the changes of the "example.txt" file for commit.

git commit -m "Add content to example.txt"

- This commits the staged changes with a descriptive message.

Step 3: Exploring History

- Modify the content of "example.txt."
- Run the following commands:

git status

- Notice the modified file is shown as "modified."

git diff

- This displays the differences between the working directory and the last commit.

git log

- This displays a chronological history of commits.

Step 4: Branching and Merging

Create a new branch named "feature" and switch to it:

git branch feature

git checkout feature

or shorthand:

git checkout -b feature

- Make changes to the "example.txt" file in the "feature" branch.
- Commit the changes in the "feature" branch.
- Switch back to the "master" branch:

git checkout master

- Merge the changes from the "feature" branch into the "master" branch:

git merge feature

Step 5: Collaborating with Remote Repositories

- Create an account on a Git hosting service like GitHub (https://github.com/).
- Create a new repository on GitHub.
- Link your local repository to the remote repository:

git remote add origin <repository_url>

- Push your local commits to the remote repository:

git push origin master

Conclusion:

Through this experiment, participants gained a foundational understanding of Git's essential commands and concepts. They learned how to set up a Git repository, manage changes, explore commit history, create and merge branches, and collaborate with remote repositories. This knowledge equips them with the skills needed to effectively use Git for version control and collaborative software development.