**FSD Laboratory 01**

**Name: Aditya N. Patil**

**Panel: 1**

**Batch: A2**

**Roll No: PA-30**

Aim: Version control with Git.

Objectives:

1. To introduce the concepts and software behind version control, using the example of Git.
2. To understand the use of 'version control' in the context of a coding project.
3. To learn Git version control with Clone, commit to, and push, pull from a git repository.

Theory:

1. What is Git? What is Version Control?

GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code. Git is a specific open-source version control system created by Linus Torvalds in 2005. Specifically, Git is a distributed version control system, which means that the entire codebase and history is available on every developer’s computer.

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

Sources- [What Is GitHub? A Beginner's Introduction to GitHub (kinsta.com)](https://kinsta.com/knowledgebase/what-is-github/)

[What is version control | Atlassian Git Tutorial](https://www.atlassian.com/git/tutorials/what-is-version-control)

1. How to use Git for version controlling?
2. Install Git: First, make sure Git is installed on your computer. You can download it from the official website (https://git-scm.com/) and follow the installation instructions for your specific operating system.
3. Configure Git: After installation, you should configure Git with your name and email address. This information is used to identify your commits.
4. Initialize a Git Repository: Create a new directory or navigate to an existing project folder and initialize a Git repository.
5. Add Files: Start tracking files by adding them to the staging area. You can add specific files or use wildcards to add multiple files.
6. Commit Changes: Once you've added your changes to the staging area, commit them with a meaningful message.
7. View Commit History: You can view the commit history to see a list of all commits in the repository.
8. Create Branches: Branches allow you to work on new features or bug fixes independently
9. Switch Branches: Switch between branches to work on different aspects of your project.
10. Merge Branches: When you're finished with a feature, merge it back into the main branch
11. Resolve Conflicts: If there are conflicting changes between branches, Git will prompt you to resolve them manually. Edit the conflicted files, then commit the resolved changes.
12. Push to Remote Repository: If you're working on a team or want to back up your code remotely, you'll need to push your changes to a remote repository
13. Pull from Remote Repository: To update your local repository with changes from the remote repository, use the git pull command.
14. Clone a Repository: If you want to start working on an existing Git repository, you can clone it to your local machine.

FAQ:

1. What is branching in Git?

Ans. Branching in Git is a feature that allows you to create separate lines of development within a Git repository. Each branch represents a distinct path of code changes, making it possible to work on multiple features or bug fixes simultaneously while keeping changes isolated until they are ready to be merged back into the main codebase. Branches help organize and manage collaborative software development efficiently.

1. How to create and merge branches in Git? Write the commands used.

Ans. To create and merge branches in Git, you can use the following commands:

1. Create a New Branch

To create a new branch and switch to it:

git checkout -b new-branch-name

To create a new branch without switching to it:

git branch new-branch-name

2. Switch to an Existing Branch:

- To switch to an existing branch:

git checkout existing-branch-name

3. Merge Changes from One Branch into Another

- First, ensure you are on the branch where you want to merge changes (e.g., the branch you want to update with changes from another branch).

- Then, merge changes from another branch:

git merge source-branch-name

4. Delete a Branch:

- To delete a branch after it's merged and no longer needed:

git branch -d branch-to-delete

Output: Screenshots of the output to be attached.

**Problem Statement:**

Created a public git repository for your team and submit the repo URL as a solution to this assignment, Learn Git concept of Local and Remote Repository, Push, Pull, Merge and Branch.

[adityapatil2340/assignment1 (github.com)](https://github.com/adityapatil2340/assignment1)

