**FSD Laboratory 01**

**Name:Harshil Patel**

**Batch:A2**

**Roll.no:26**

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?

Ans. Git is a distributed version control system (DVCS) used for tracking changes in source code during software development.

Git is particularly popular for its speed, flexibility, and ability to handle both small and large projects efficiently.

Version control is the practice of tracking and managing changes to files and code over time. It helps developers work collaboratively, keep track of changes, and easily revert to previous versions if needed.

1. How to use Git for version controlling?

Ans. Using Git for version control involves several key steps and commands. Here's a basic overview of how to get started with Git:

1.Install Git:

If you haven't already, you'll need to install Git 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.

2.Configure Git:

After installation, you should configure Git with your name and email address, which will be associated with your commits. Open a terminal or command prompt and run the following commands, replacing "Your Name" and "youremail@example.com" with your actual name and email address:

git config --global user.name "Your Name"

git config --global user.email [youremail@example.com](mailto:youremail@example.com)

3.Create a Git Repository:

You can initialize a new Git repository in an existing directory or create a new directory for your project. To initialize a new repository in an existing directory, navigate to the directory in your terminal and run: git init

4.Add and Commit Files:

Start adding files to your project directory or make changes to existing files. Once you've made changes, you need to stage them for commit using the git add command. For example:

git add filename.txt # Stage a specific file

git add . # Stage all changes in the current directory

After staging your changes, commit them to the repository with a descriptive commit message:

git commit -m "Initial commit" # Commit changes with a message

5.View the Version History:

You can view the commit history using the git log command:

git log

6.Create Branches:

Branches allow you to work on different features or bug fixes in isolation. You can create a new branch using the git branch command and switch to it using git checkout. For example:

git branch new-feature # Create a new branch

git checkout new-feature # Switch to the new branch

7.Merge Branches:

Once you've finished working on a branch, you can merge it back into the main branch (usually "master" or "main") using the git merge command:

git checkout main # Switch to the main branch

git merge new-feature # Merge the new-feature branch into main

8.Remote Repositories (Optional):

Git allows you to work with remote repositories hosted on services like GitHub, GitLab, or Bitbucket. You can clone remote repositories, push your changes to them, and collaborate with others. To clone a remote repository, use the git clone command:

git clone <https://github.com/yourusername/your-repo.git>

9.Push and Pull (Remote):

To share your local changes with a remote repository, use git push. To update your local repository with changes from a remote repository, use git pull:

git push origin main # Push changes to the remote repository

git pull origin main # Pull changes from the remote repository

FAQ:

1. What is branching in Git?

Branching in Git is a fundamental concept that allows developers to work on different lines of development within the same repository. Each branch represents an independent line of development and typically corresponds to a specific feature, bug fix, or task. Branches in Git provide isolation, enabling developers to work on their tasks without affecting the main codebase until their work is ready to be merged.

Here are some key points about branching in Git:

1.Main Branch (Master or Main): In most Git repositories, there is a default main branch, often named "master" or "main." This branch typically represents the stable and production-ready version of the project.

2.Creating a Branch: Developers can create new branches at any point in the project's history. To create a new branch, you use the git branch command, followed by the name of the new branch. For example: git branch new-feature

3.Switching Between Branches: You can switch between branches using the git checkout command. For example:

git checkout new-feature

This command not only switches to the specified branch but also updates your working directory to reflect the contents of that branch.

4.Working in a Branch: Once you're on a branch, you can make changes to your code as needed. These changes are isolated to the branch you're working in, allowing you to work on new features or bug fixes independently of other ongoing development work.

5.Committing Changes: As you make changes to your code, you commit those changes using git commit. These commits are specific to the branch you're on.

6.Merging Branches: When you've completed your work on a branch and want to incorporate those changes into another branch (often the main branch), you use the git merge command. For example:

git checkout main

git merge new-feature

This merges the changes from the "new-feature" branch into the "main" branch.

7.Deleting Branches: After a branch has served its purpose and its changes have been merged, you can delete the branch using the git branch -d command:

git branch -d new-feature

8.Remote Branches: In addition to local branches, Git also allows you to work with remote branches hosted on remote repositories like GitHub or GitLab. You can push your local branches to remote repositories, create branches on remote repositories, and collaborate with others by sharing branches.

Branching is a powerful feature in Git that enables effective collaboration, feature development, and bug fixing. It promotes a structured and organized approach to managing code changes within a project. When used correctly, branching can help teams work more efficiently and reduce the risk of conflicts and errors in their codebase.

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

Output: Screenshots of the output to be attached.

Here are the steps to create and merge branches:

Create a New Branch:

Switch to the New Branch:

Making Changes and Committing:

Make Changes:

While on the new branch, make the necessary changes to your code or work on the new feature.

Commit Changes:

After making changes, stage and commit them using the git add and git commit commands:

Merging a Branch:

Switch to the Target Branch:

To merge your changes from the feature branch into the target branch (often the main branch), switch to the target branch using git checkout. For example, to merge into the "main" branch:

Merge the Feature Branch:

Use the git merge command to merge the feature branch into the target branch:

If there are no conflicts between the branches, Git will perform the merge automatically. If conflicts arise (i.e., changes in both branches overlap in the same part of a file), you'll need to resolve them manually.

Resolve Conflicts (if necessary):

When conflicts occur, Git will indicate the conflicting files. Open these files in your text editor, resolve the conflicts, and save the changes. Then add and commit the resolved files:

Delete the Feature Branch (Optional):

After merging and ensuring everything is working as expected, you can delete the feature branch using the git branch -d command:

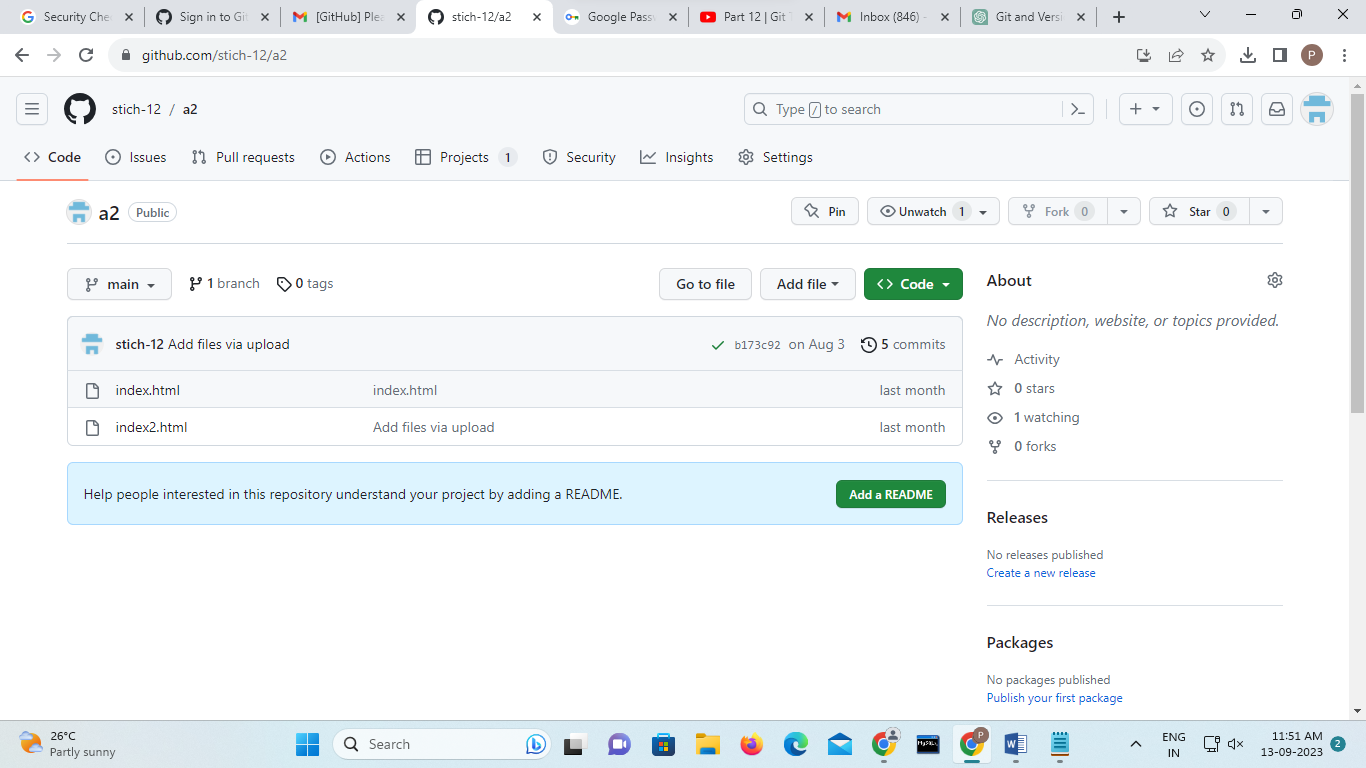
Pushing Changes (for Remote Collaboration):

Push Changes to Remote (Optional):

If you're working with a remote repository (e.g., GitHub), and you want to share your changes with others, use the git push command to push the merged changes to the remote repository:

**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.



git@github.com:stich-12/a2.git