GIT Repo URL: https://github.com/prateekp1304/FSD

Roll No: 08
Name: Prateek Patel
PRN: 1032210532

FSD Laboratory 01

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?

Git is a distributed version control system (DVCS) used for tracking changes in source code during software development. It was created by Linus Torvalds in 2005 and has since become the de facto standard for version control in the software development industry. Git is open-source and can be used for projects of all sizes.

Version control is a system that tracks changes to files and directories over time. It allows multiple people to collaborate on a project, keep a history of changes, and easily switch between different versions of the codebase version control is the broader concept that encompasses tools like Git. It is a crucial aspect of modern software development

2. How to use Git for version controlling?

Using Git for version control involves several key steps and commands. Here's a high-level overview of how to use Git for version controlling your software projects:

1. **Install Git**: If you haven't already, you need to install Git on your computer. You can download the installer for your operating system from the official Git website.
2. **Configure Git**: After installation, you should configure Git with your name and email address. This information will be associated with your commits.
3. **Create a Git Repository**: You can either create a new Git repository or clone an existing one from a remote repository.
4. **Stage Files**: Before committing changes, you need to stage the files you want to include in the next commit using the **git add** command. For example, to stage all files in the current directory.
5. **Commit Changes**: Once you've staged your changes, you can commit them to the Git repository with a descriptive message.

6. **Create Branches**: To work on different features or bug fixes independently, you can create branches using the **git branch** command and switch between them with **git checkout**.
7. **Merge Branches**: After completing work on a branch, you can merge it back into the main development branch (e.g., **main** or **master**) using the **git merge** command.
8. **Push to Remote**: To share your changes with others or back up your code, you can push your local repository to a remote repository.
9. **Pull Changes**: When working in a team, it's essential to stay up to date with changes made by others. You can pull the latest changes from the remote repository.
10. **Resolve Conflicts**: If there are conflicting changes between your branch and the remote branch, Git will notify you. You'll need to resolve these conflicts manually by editing the affected files and then committing the changes.
11. **Review History**: You can review the commit history using **git log** to see a list of commits, authors, and commit messages. You can also use graphical tools or Git platform interfaces for a more visual history.

FAQ:

1. What is branching in Git?

Branching in Git is a fundamental and powerful feature that allows developers to work on different lines of development simultaneously within the same Git repository. It essentially creates separate "branches" of your project's codebase, each of which can represent a new feature, bug fix, or experiment. Branches in Git provide the following benefits:

- **Isolation**: Each branch is an isolated workspace where you can make changes to the code without affecting the main codebase or other branches. This isolation is critical for parallel development and experimentation.
- **Parallel Development**: Multiple developers can work on different branches simultaneously, addressing different tasks or features. This parallelism helps teams collaborate more efficiently.
- **Feature Development**: Branches are commonly used to develop new features. You can create a new branch for each feature or enhancement, keeping the main codebase stable and allowing features to be developed independently.
- **Bug Fixes**: Branches are also useful for isolating bug fixes. You can create a branch to fix a specific issue, test the fix, and then merge it back into the main codebase when it's verified.
- **Code Experimentation**: Developers can create branches to experiment with new ideas or approaches without affecting the main project. If an experiment is successful, it can be merged; otherwise, the branch can be discarded.

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

To create and merge branches in Git, you'll use a combination of Git commands. Here are the steps with the corresponding commands:

Creating a Branch:

- Create a new branch. Replace **feature-branch** with the name you want for your branch. - git branch feature-branch
- Switch to the newly created branch. This can be done in one step using the '**-b**' flag. - git checkout -b feature-branch

Making Changes and Committing:

- Make your changes to the code in this branch.
- Stage and commit your changes. git add git commit -m "Your commit message here"

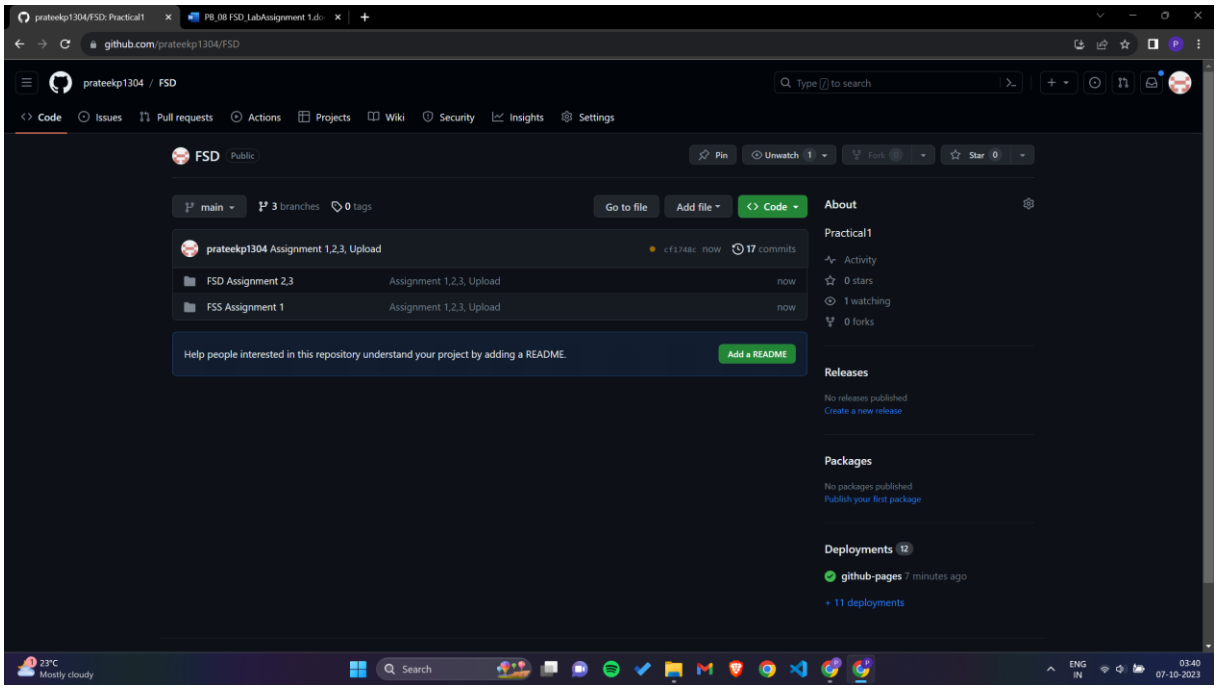
Merging the Branch:

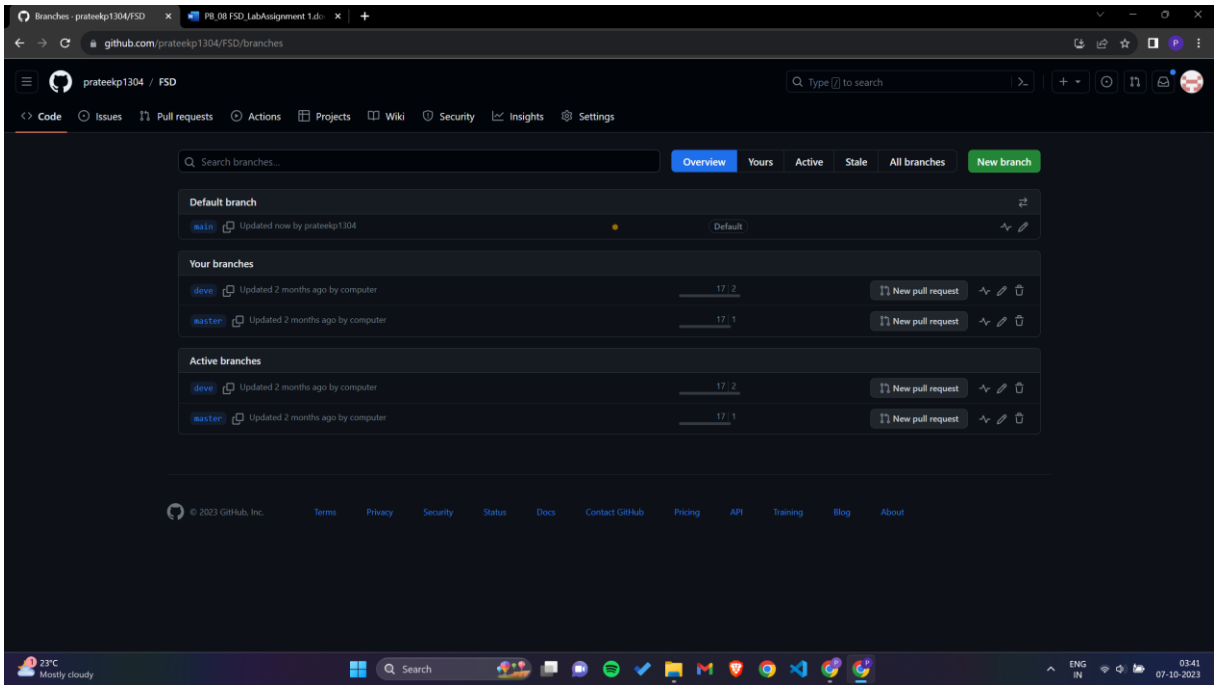
- Switch back to the main branch (or the branch you want to merge into). - git checkout main
- Merge the feature branch into the main branch. - git merge feature-branch

Output: Screenshots of the output to be attached.

Problem Statement: Create 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

Output Screenshots:


prateekp1304/FSD: Practical1
PB_08 FSD_LabAssignment 1.do
github.com/prateekp1304/FSD
prateekp1304 / FSD
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
FSD
Public
Pin
Unwatch 1
Star 0
main
3 branches
0 tags
Go to file
Add file
Code
About
Practical1
Activity
0 stars
1 watching
0 forks
prateekp1304 Assignment 1,2,3, Upload
17 commits
FSD Assignment 2,3
Assignment 1,2,3, Upload
now
FSS Assignment 1
Assignment 1,2,3, Upload
now
Help people interested in this repository understand your project by adding a README.
Add a README
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Deployments 12
github-pages 7 minutes ago
+ 11 deployments
23°C
Mostly cloudy
Search
ENG
IN
03:40
07-10-2023


Branches · prateekp1304/FSD
PB_08 FSD_LabAssignment 1.do
github.com/prateekp1304/FSD/branches
prateekp1304 / FSD
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Search branches...
Overview
Yours
Active
Stale
All branches
New branch
Default branch
main
Updated now by prateekp1304
Default
Your branches
deve
Updated 2 months ago by computer
17 2
New pull request
master
Updated 2 months ago by computer
17 1
New pull request
Active branches
deve
Updated 2 months ago by computer
17 2
New pull request
master
Updated 2 months ago by computer
17 1
New pull request
© 2023 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About
23°C
Mostly cloudy
Search
ENG
IN
03:41
07-10-2023