School of Computer Engineering & Technology
Class: Third Year B.Tech CSE (Semester V)
Course: Full Stack Development

FSD Laboratory 01

Name: Janhavi Pawar
Roll no:01
Panel :H
PRN:1032231531

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 designed to track changes in source code during software development. It allows multiple developers to collaborate on the same project, manage changes, and keep a history of all modifications. Here are some key features of Git:

- **Distributed Nature**: Every developer has a complete copy of the repository, including its history. This allows for offline work and ensures that the repository is not dependent on a central server.
- **Branching and Merging**: Git allows developers to create branches for new features or bug fixes, and then merge these branches back into the main codebase. This supports parallel development and experimentation.
- **Commit History**: Git keeps a detailed history of changes, allowing developers to track who made what changes and when. This history is useful for debugging and understanding the evolution of a project.
- **Efficiency**: Git is designed to handle large projects efficiently. It uses techniques like compression and delta storage to optimize performance.

Version control (also known as source control) is a system that manages changes to source code or other documents over time. It provides a way to track and manage different versions of files and collaborate with other team members. Key aspects of version control include:

- **Tracking Changes**: Version control systems record changes to files over time, allowing you to see what has been changed, who made the change, and why.
- **Reverting Changes**: You can revert to a previous version of a file or the entire project if a recent change introduces errors or issues.
- **Collaboration**: Multiple developers can work on the same project simultaneously. Version control systems handle merging changes from different contributors and resolving conflicts.
- **Branching and Merging**: Developers can create branches to work on new features or fixes without affecting the main codebase. These changes can be merged back into the main branch when they are complete and tested

2. How to use Git for version controlling?
Ans:-

1. Initialize a Repository

Start a new Git repository in your project directory:

```
git init
```

2. Track Files

Add files to the staging area:

```
git add <file>
```

Add all files:

```
git add .
```

3. Commit Changes

Save your staged changes to the repository:

```
git commit -m "Commit message"
```

4. Create and Switch Branches

Create and switch to a new branch:

```
git checkout -b <branch-name>
```

5. Merge Branches

Merge changes from one branch into another:

```
git checkout <target-branch>
git merge <source-branch>
```

6. View Changes and History

View the commit history:

```
git log
```

View differences between changes:

```
git diff
```

7. Work with Remote Repositories

Add a remote repository:

```
git remote add origin <remote-url>
```

Push changes to the remote repository:

```
git push -u origin <branch-name>
```

Pull changes from the remote repository:

```
git pull origin <branch-name>
```

8. Handle Conflicts and Undo Changes

Resolve merge conflicts manually, then add and commit the resolved files.

Discard local changes:

```
git restore <file>
```

Undo the last commit:

```
git reset --soft HEAD~1
```

FAQ:

1.What is branching in Git?
Ans:-

Git is a distributed version control system designed to track changes in source code during software development. It allows multiple developers to collaborate on the same project, manage changes, and keep a history of all modifications. Here are some key features of Git:

- **Distributed Nature**: Every developer has a complete copy of the repository, including its history. This allows for offline work and ensures that the repository is not dependent on a central server.
- **Branching and Merging**: Git allows developers to create branches for new features or bug fixes, and then merge these branches back into the main codebase. This supports parallel development and experimentation.
- **Commit History**: Git keeps a detailed history of changes, allowing developers to track who made what changes and when. This history is useful for debugging and understanding the evolution of a project.
- **Efficiency**: Git is designed to handle large projects efficiently. It uses techniques like compression and delta storage to optimize performance.

Version control (also known as source control) is a system that manages changes to source code or other documents over time. It provides a way to track and manage different versions of files and collaborate with other team members. Key aspects of version control include:

- **Tracking Changes**: Version control systems record changes to files over time, allowing you to see what has been changed, who made the change, and why.
- **Reverting Changes**: You can revert to a previous version of a file or the entire project if a recent change introduces errors or issues.
- **Collaboration**: Multiple developers can work on the same project simultaneously. Version control systems handle merging changes from different contributors and resolving conflicts.
- **Branching and Merging**: Developers can create branches to work on new features or fixes without affecting the main codebase. These changes can be merged back into the main branch when they are complete and tested

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

Creating a Branch

1. **Create a New Branch**

   To create a new branch, use the following command:

   ```
   git branch branch-name
   ```

   Replace branch-name with your desired branch name.

2. **Switch to the New Branch**

   After creating a branch, you need to switch to it to start working on it:

   ```
   git checkout branch-name
   ```

   You can combine the branch creation and checkout into a single command:

   ```
   git checkout -b branch-name
   ```

Merging Branches

1. **Switch to the Branch You Want to Merge Into**

   First, switch to the branch that will receive the changes (usually the main branch or another feature branch):

   ```
   git checkout main
   ```

   Replace main with the branch you want to merge into.

2. **Merge the Other Branch**

   Merge the branch with the changes into the current branch:

   ```
   git merge branch-name
   ```

   Replace branch-name with the branch you want to merge.

Example Workflow

Here’s a complete example of creating a branch, making changes, and merging it back:

1. **Create and Switch to a New Branch**

   ```
   git checkout -b feature-branch
   ```

2. **Make Changes and Commit**

   Edit your files and then stage and commit your changes:

   ```
   git add .
   git commit -m "Added new feature"
   ```

3. **Switch Back to the Main Branch**

   ```
   git checkout main
   ```

4. **Merge the Feature Branch into Main**

   ```
   git merge feature-branch
   ```

5. **Delete the Feature Branch (Optional)**

   If you no longer need the feature branch, you can delete it:

   ```
   git branch -d feature-branch
   ```

   Use -D instead of -d if you want to force delete the branch, even if it hasn’t been merged:

   ```
   git branch -D feature-branch
   ```

Summary of Commands

1. **Create a Branch:**

   ```
   git branch branch-name
   ```

2. **Switch to a Branch:**

```
git checkout branch-name
```

Or create and switch in one step:

```
git checkout -b branch-name
```

3. **Merge Branches:**

```
git checkout target-branch
git merge branch-name
```

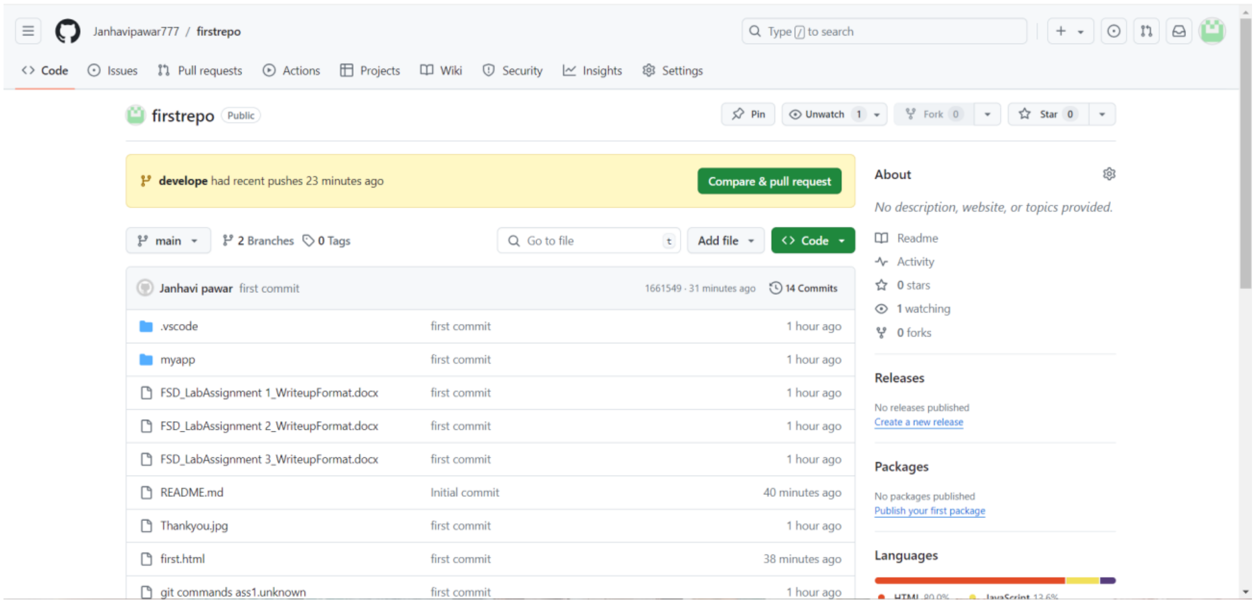
4. **Delete a Branch:**

```
git branch -d branch-name
```

Or force delete:

```
git branch -D branch-name
```

Output: Screenshots of the output to be attached.



Janhavipawar777 / firstrepo

firstrepo Public

develope had recent pushes 23 minutes ago

main · 2 Branches · 0 Tags

Janhavi pawar first commit 1661549 · 31 minutes ago 14 Commits

.vscode	first commit	1 hour ago
myapp	first commit	1 hour ago
FSD_LabAssignment 1_WriteupFormat.docx	first commit	1 hour ago
FSD_LabAssignment 2_WriteupFormat.docx	first commit	1 hour ago
FSD_LabAssignment 3_WriteupFormat.docx	first commit	1 hour ago
README.md	Initial commit	40 minutes ago
Thankyou.jpg	first commit	1 hour ago
first.html	first commit	38 minutes ago
git commands ass1.unknown	first commit	1 hour ago

About

No description, website, or topics provided.

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

HTML 80.0% JavaScript 13.6%

$ git init

Reinitialized existing Git repository in C:/Users/dell/Desktop/sem 5/FSD/.git/

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develop)

```
$ git add .

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develop)
$ git commit -m "first commit"
[develop 5807bc5] first commit
 1 file changed, 13 insertions(+)
 create mode 100644 first.html

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develop)
$ git branch -M main

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git remote add origin https://github.com/Janhavipawar777/firstrepo.git
error: remote origin already exists.

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git remote -v
origin  https://github.com/Janhavipawar777/janhavirepo1.git (fetch)
origin  https://github.com/Janhavipawar777/janhavirepo1.git (push)

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git remote set-url origin https://github.com/Janhavipawar777/firstrepo.git

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git push -u origin main
To https://github.com/Janhavipawar777/firstrepo.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/Janhavipawar777/firstrepo.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git fetch origin
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), 1.97 KiB | 4.00 KiB/s, done.
From https://github.com/Janhavipawar777/firstrepo
 + 868e38a...30a810f main       -> origin/main  (forced update)

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git push -u origin main
To https://github.com/Janhavipawar777/firstrepo.git
 ! [rejected]        main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/Janhavipawar777/firstrepo.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. If you want to integrate the remote changes,
hint: use 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git fetch origin

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git pull origin main
From https://github.com/Janhavipawar777/firstrepo
 * branch            main       -> FETCH_HEAD
fatal: refusing to merge unrelated histories

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git pull origin main --allow-unrelated-histories
From https://github.com/Janhavipawar777/firstrepo
 * branch            main       -> FETCH_HEAD
Auto-merging first.html
```

```
CONFLICT (add/add): Merge conflict in first.html
Automatic merge failed; fix conflicts and then commit the result.

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main|MERGING)
$ git.publish/30
bash: git.publish/30: No such file or directory

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main|MERGING)
$ git pull origin main --allow-unrelated-histories
error: You have not concluded your merge (MERGE_HEAD exists).
hint: Please, commit your changes before merging.
fatal: Exiting because of unfinished merge.

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main|MERGING)
$ git commit -m "first commit"
[main 1661549] first commit

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git push -u origin main
Enumerating objects: 54, done.
Counting objects: 100% (54/54), done.
Delta compression using up to 4 threads
Compressing objects: 100% (47/47), done.
Writing objects: 100% (52/52), 3.92 MiB | 2.76 MiB/s, done.
Total 52 (delta 13), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (13/13), completed with 1 local object.
To https://github.com/Janhavipawar777/firstrepo.git
   30a810f..1661549  main -> main
branch 'main' set up to track 'origin/main'.

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ get status
bash: get: command not found
```

```
dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    .vscode/launch.json

no changes added to commit (use "git add" and/or "git commit -a")

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git branch develope

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git check out develope
git: 'check' is not a git command. See 'git --help'.

The most similar command is
        checkout

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$
```

```
dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git checkout develope
D       .vscode/launch.json
Switched to branch 'develope'

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git push origin develope
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'develope' on GitHub by visiting:
remote:      https://github.com/Janhavipawar777/firstrepo/pull/new/develope
remote:
To https://github.com/Janhavipawar777/firstrepo.git
 * [new branch]      develope -> develope

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git status
On branch develope
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    .vscode/launch.json

no changes added to commit (use "git add" and/or "git commit -a")

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git pull https://github.com/Janhavipawar777/firstrepo.git
From https://github.com/Janhavipawar777/firstrepo
 * branch            HEAD       -> FETCH_HEAD
Already up to date.
```

```
dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git status
On branch develope
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    .vscode/launch.json

no changes added to commit (use "git add" and/or "git commit -a")

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ echo new.txt
new.txt

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git status
On branch develope
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    .vscode/launch.json

no changes added to commit (use "git add" and/or "git commit -a")

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git add new.txt
fatal: pathspec 'new.txt' did not match any files

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ ls
 first.html                        'git commands ass1.unknown'   index2.html
profile.jpg   service3.jpg
'FSD_LabAssignment   1_WriteupFormat.docx'          header-image.jpg
medium_weather_words_0e5facbfaf.png   README.md     thanks.jpg
```

```
'FSD_LabAssignment 2_WriteupFormat.docx'   index.html                    myapp/
service1.jpg   Thankyou.jpg
'FSD_LabAssignment 3_WriteupFormat.docx'   index2.css                    prac3.html
service2.jpg

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ touch new.txt

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ ls
 first.html                              'git commands ass1.unknown'    index2.html
prac3.html    service2.jpg
'FSD_LabAssignment     1_WriteupFormat.docx'                 header-image.jpg
medium_weather_words_0e5facbfaf.png   profile.jpg    service3.jpg
'FSD_LabAssignment 2_WriteupFormat.docx'   index.html                    myapp/
README.md     thanks.jpg
'FSD_LabAssignment 3_WriteupFormat.docx'   index2.css                    new.txt
service1.jpg   Thankyou.jpg

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git commit -m "file>"
On branch develope
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    .vscode/launch.json

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        new.txt

no changes added to commit (use "git add" and/or "git commit -a")

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git add .vscode/launch.json
git add new.txt
```

```
dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git add -A

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git commit -m "file>"
[develope d99ffb4] file>
 2 files changed, 15 deletions(-)
 delete mode 100644 .vscode/launch.json
 create mode 100644 new.txt

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git push origin develope
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 268 bytes | 268.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Janhavipawar777/firstrepo.git
   1661549..d99ffb4  develope -> develope

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (develope)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git merge develope
Updating 1661549..d99ffb4
Fast-forward
 .vscode/launch.json | 15 ---------------
 new.txt             |  0
 2 files changed, 15 deletions(-)
 delete mode 100644 .vscode/launch.json
 create mode 100644 new.txt

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git log--oneline--decorate
git: 'log--oneline--decorate' is not a git command. See 'git --help'.

dell@DESKTOP-87AJFOL MINGW64 ~/Desktop/sem 5/FSD (main)
$ git log --oneline --decorate
d99ffb4 (HEAD -> main, origin/develope, develope) file>
1661549 (origin/main) first commit
5807bc5 first commit
30a810f Add files via upload
d99ffb4 (HEAD -> main, origin/develope, develope) file>
1661549 (origin/main) first commit
5807bc5 first commit
30a810f Add files via upload
872a2ae Initial commit
f60e58e first commit
868e38a Add files via upload
```

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.