

FSD ASSIGNMENT -1

By-Yashvardhan Tekavade
PB-15 Batch-1
Panel-1 TY-CSF

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 an open-source distributed version control system. It is designed to handle minor to major projects with high speed and efficiency. It is developed to coordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace. Git is the foundation of many services like GitHub and GitLab, but we can use Git without using any other Git services. Git can be used privately and publicly. Git was created by Linus Torvalds in 2005 to develop the Linux Kernel. It is also used as an important distributed version-control tool for DevOps.

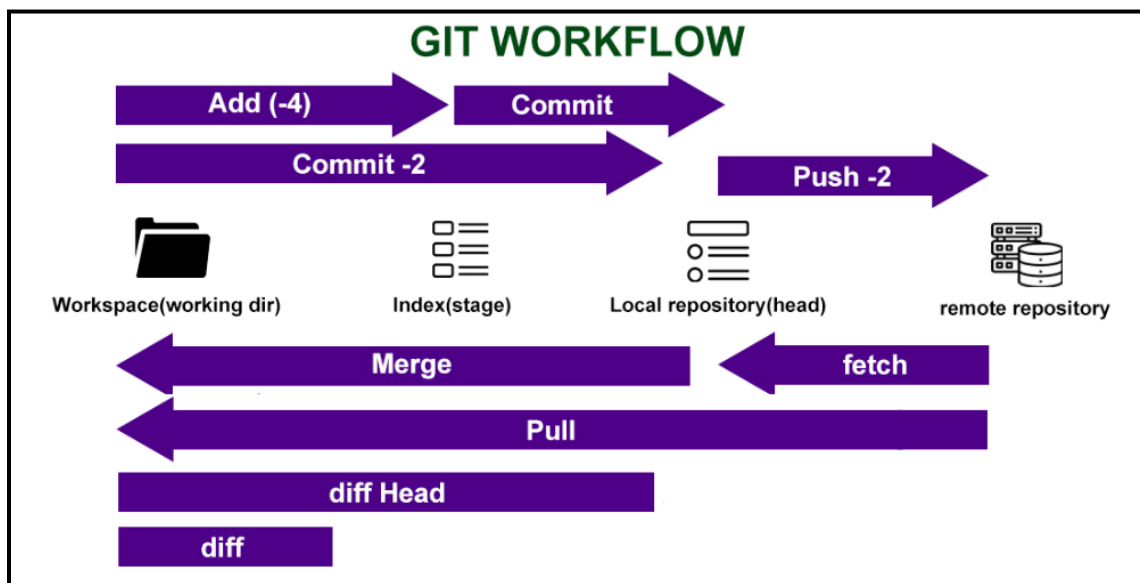
Git is easy to learn and has fast performance. It is superior to other SCM tools like Subversion, CVS, Perforce, and Clear Case.

Version control - also known as source control or revision control - is an important software development practice for tracking and managing changes made to code and other files. It is closely related to source code management. With version control, every change made to the code base is tracked. This allows software developers to see the entire history of who changed what at any given time — and roll back from the current version to an earlier version if they need to. It also creates a single source of truth. Version control (or source control or revision control) serves as a safety net to protect the source code from irreparable harm, giving the development team the freedom to experiment without fear of causing damage or creating code conflicts.

2. How to use Git for version control?

The GitHub workflow can be summarised by the “commit-pull-push” mantra.

1. Commit -Once you’ve saved your files, you need to commit them - this means the changes you have made to files in your repo will be saved as a version of the repo, and your changes are now ready to go up on GitHub (the online copy of the repository).
2. Pull- Now, before you send your changes to GitHub, you need to pull, i.e. make sure you are completely up to date with the latest version of the online version of the files - other people could have been working on them even if you haven’t. You should always pull before you start editing and before you push.
3. Push - Once you are up to date, you can push your changes - at this point in time your local copy and the online copy of the files will be the same.

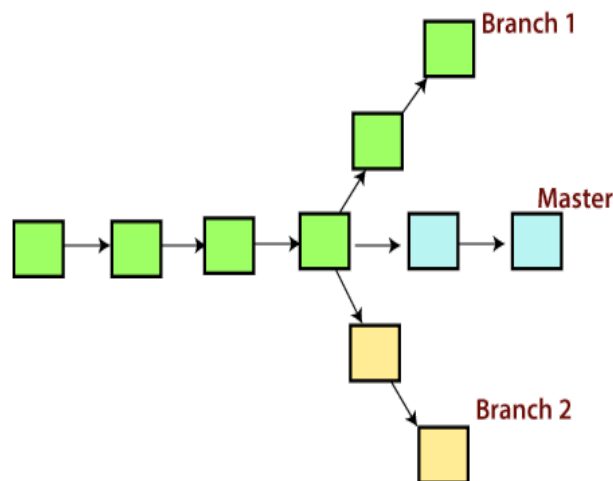


FAQs-

1. What is branching in Git?

A branch is a version of the repository that diverges from the main working project. It is a feature available in most modern version control systems. A Git project can have more than one branch. These branches are a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug, you spawn a new branch to summarize your changes. So, it is complex to merge the unstable code with the main code base and also facilitates you to clean up your future history before merging with the main branch.

We can perform various operations on Git branches. The git branch command allows you to **create**, **list**, **rename** and **delete** branches. Many operations on branches are applied by git checkout and git merge command. So, the git branch is tightly integrated with the **git checkout** and **git merge** commands.



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

- **Create Branch**-You can create a new branch with the help of the git branch command. This command will be used as:
Syntax: \$ git branch <branch name>
- **Delete Branch**-You can delete the specified branch. It is a safe operation. In this command, Git prevents you from deleting the branch if it has unmerged changes. Below is the command to do this.
Syntax: \$ git branch -d<branch name>
- **Merge Branch:** Git allows you to merge the other branch with the currently active branch. You can merge two branches with the help of **git merge** command. Below command is used to merge the branches:
Syntax: \$ git merge <branch name>

Commands:

```
Admin@yAsshh MINGW64 ~ (master)
```

```
$ git init
```

```
Reinitialized existing Git repository in C:/Users/Admin/.git/
```

```
Admin@yAsshh MINGW64 ~ (master)
```

```
$ git add .
```

```
Admin@yAsshh MINGW64 ~ (master)
```

```
$ git commit -m "first commit"
```

```
[master (root-commit) 135bdb1] first commit
```

```
1 file changed, 1 insertion(+)
```

```
create mode 100644 hi.txt
```

```
Admin@yAsshh MINGW64 ~ (master)
```

```
$ git branch -M main
```

```
Admin@yAsshh MINGW64 ~ (master)
```

```
$ git remote add origin <https://github.com/yashtekavade/fsd.git>
```

```
Admin@yAsshh MINGW64 ~ (master)
```

```
$ git push -u origin main
```

```
info: please complete authentication in your browser...
```

```
Enumerating objects: 3, done.
```

```
Counting objects: 100% (3/3), done.
```

```
Writing objects: 100% (3/3), 213 bytes | 213.00 KiB/s, done.
```

```
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
```

```
To https://github.com/yashtekavade/fsd.git
```

```
* [new branch]    main -> main
```

```
branch 'main' set up to track 'origin/main'.
```

```
Admin@yAsshh MINGW64 ~ (master)
```

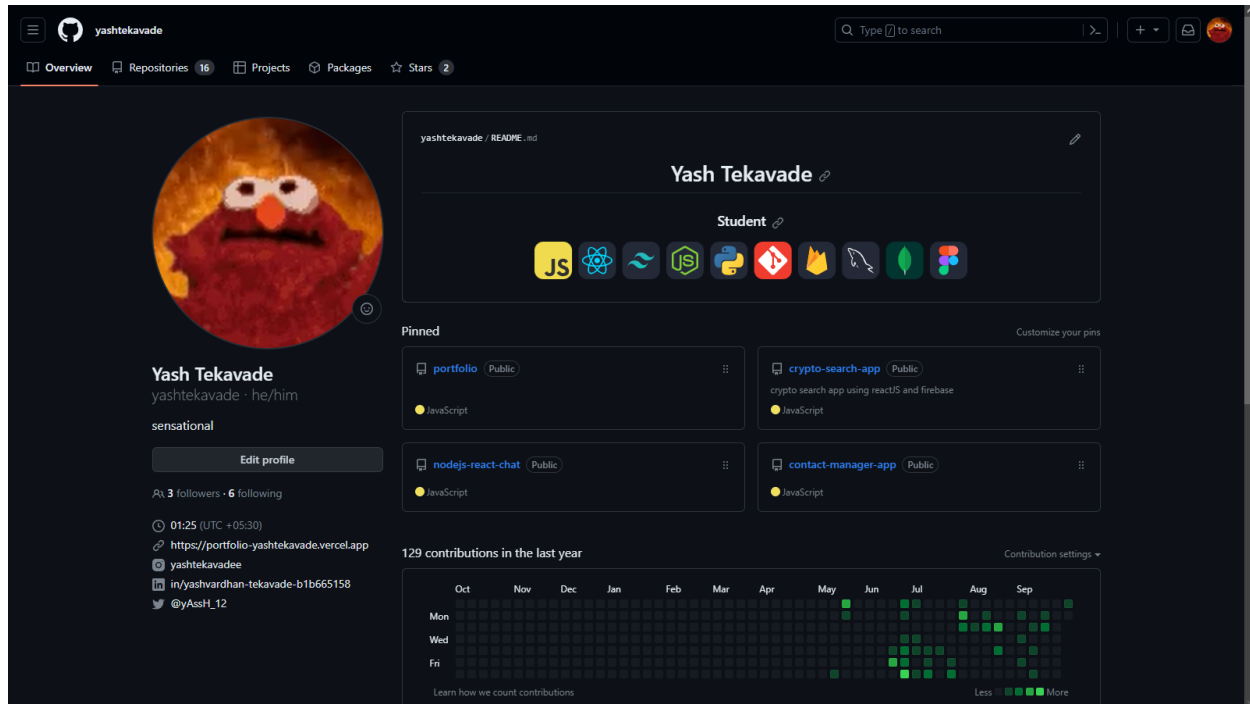
```
$ git status
```

```
On branch main
```

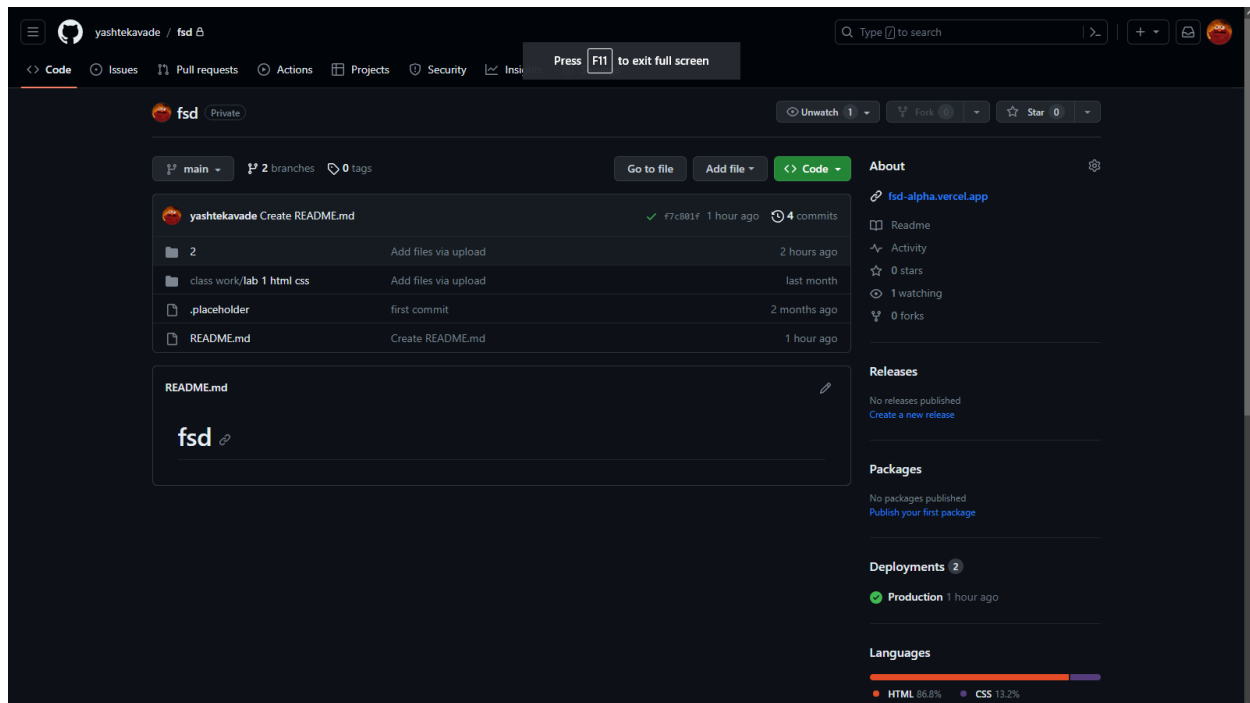
```
Your branch is up to date with 'origin/main'.
```

```
nothing to commit, working tree clean
```

Screenshots:



GitHub profile of yashtekavade. The profile shows a circular avatar with a red and orange background. The name is Yash Tekavade, with the handle yashtekavade · he/him. The bio is sensational. The profile includes a list of pinned repositories: portfolio, crypto-search-app, nodejs-react-chat, and contact-manager-app. The profile also shows 129 contributions in the last year, with a calendar view showing activity from October to September.



GitHub repository view of fsd. The repository is private and owned by yashtekavade. It shows a list of files and folders, including README.md, class work/lab 1.html.css, .placeholder, and README.md. The repository has 2 branches and 0 tags. The README.md file is open, showing the text fsd. The repository also has a sidebar with links to About, Releases, Packages, and Deployments.