Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **ZETA**

Department: **DCSE**

Submitted By:
VANSHIKA
2110992051
G27

Submitted To:
Dr..Anuj Jain

Aim: Setting up of Git Client

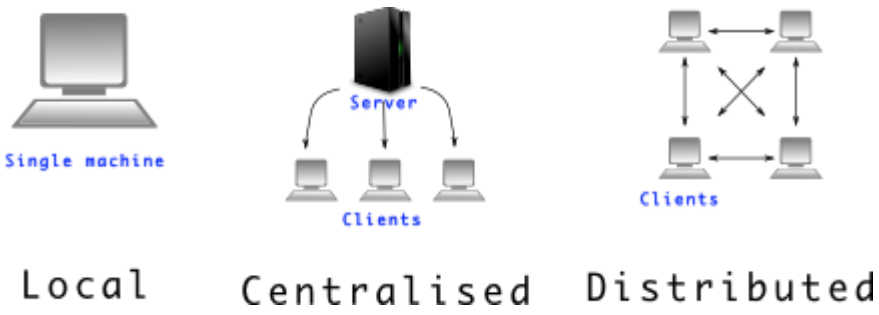
Theory:

What is Git?

Git is a software used for tracking changes in any set of files, usually used for coordinating work among members of a team.

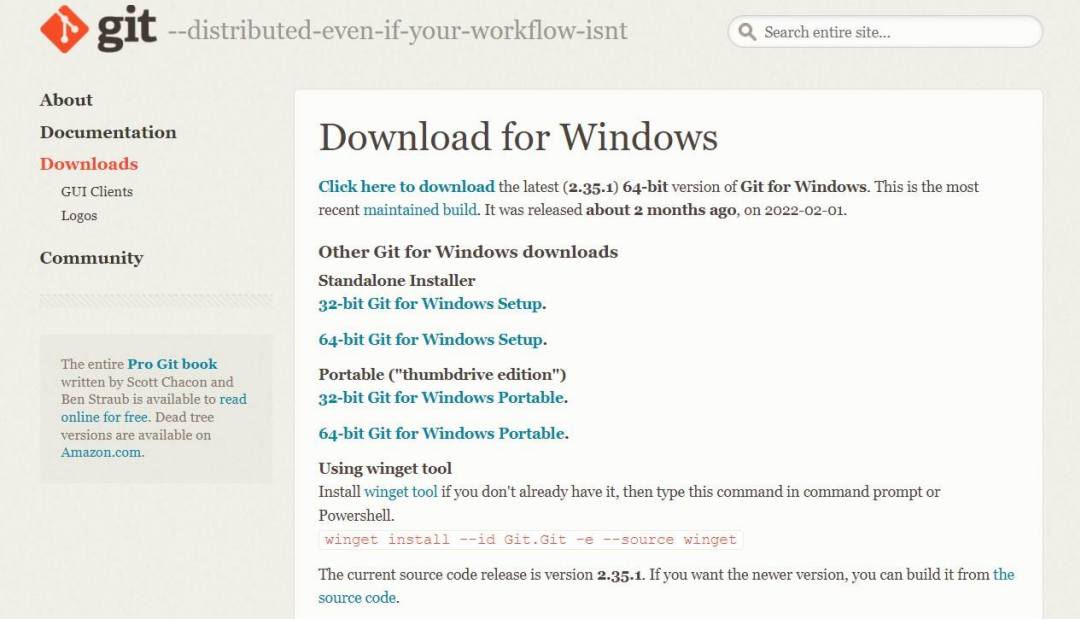
History of VCS:

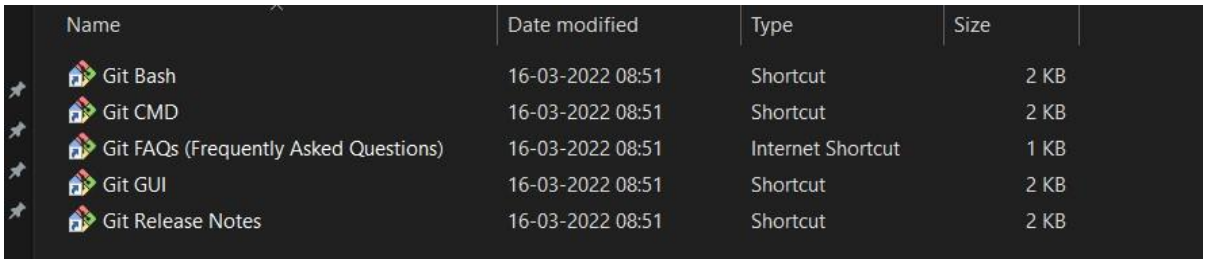
- **Local VCS:** No internet is needed because it uses a database to keep track of files.
- **Centralized VCS:** Centralized version control systems are based on the idea that there is a single "central" copy of your project somewhere (probably on a server), and programmers will "commit" their changes to this central copy. "Committing" a change simply means recording the change in the central system.
- **Distributed VCS:** A type of version control where the complete codebase including its full version history is mirrored on every developer's computer.



How to install GIT on Windows?

There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to https://gitscm.com/download/win and the download will start automatically. Note that this is a project called Git for Windows, which is separate from Git itself; for more information on it, go to https://gitforwindows.org.





Check version of git by using git –version command.

CHITKARA UNIVERSITY

Aim: Setting up GitHub Account

Theory:

What is GitHub?

GitHub is a code hosting platform for version control and collaboration. In other words, it manages repositories.

Advantages:

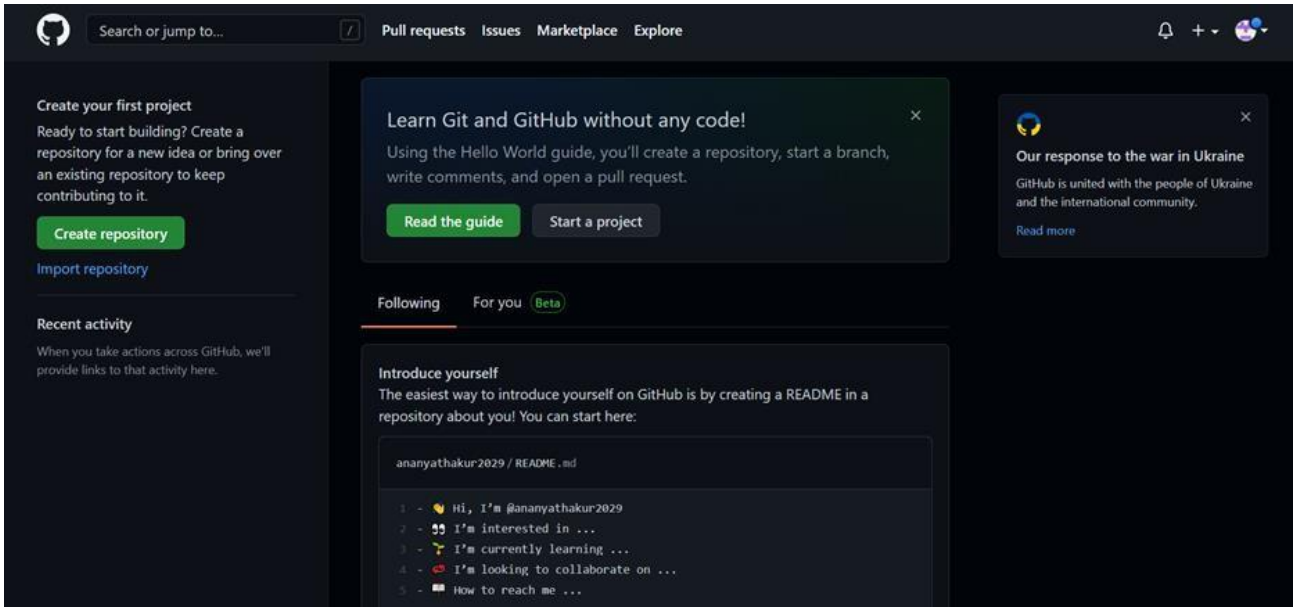
- It makes it easy to contribute to Open-Source projects.
- Track changes in your code across versions.

Procedure:

Search for GitHub in any search engine or https://github.com/signup

If you're a new user add your email and click on **Sign up for GitHub**. Otherwise click on **Sign In** at the top right corner

Signing into GitHub:

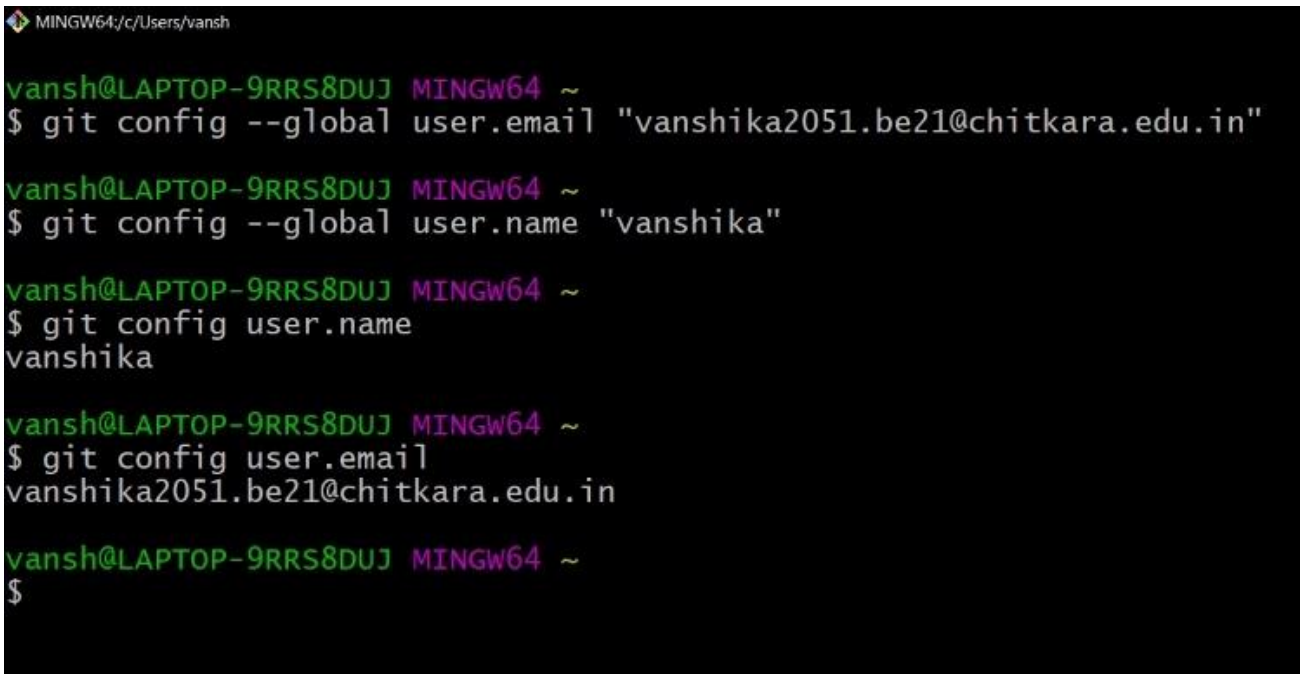


Linking GitHub account with Git Bash:

Username:
git config --global user.name “username in github”

Email:
git config --global user.email “your email in github”

Check Username & Email:
git config user.name git
config user.email



Aim: Program to Generate log

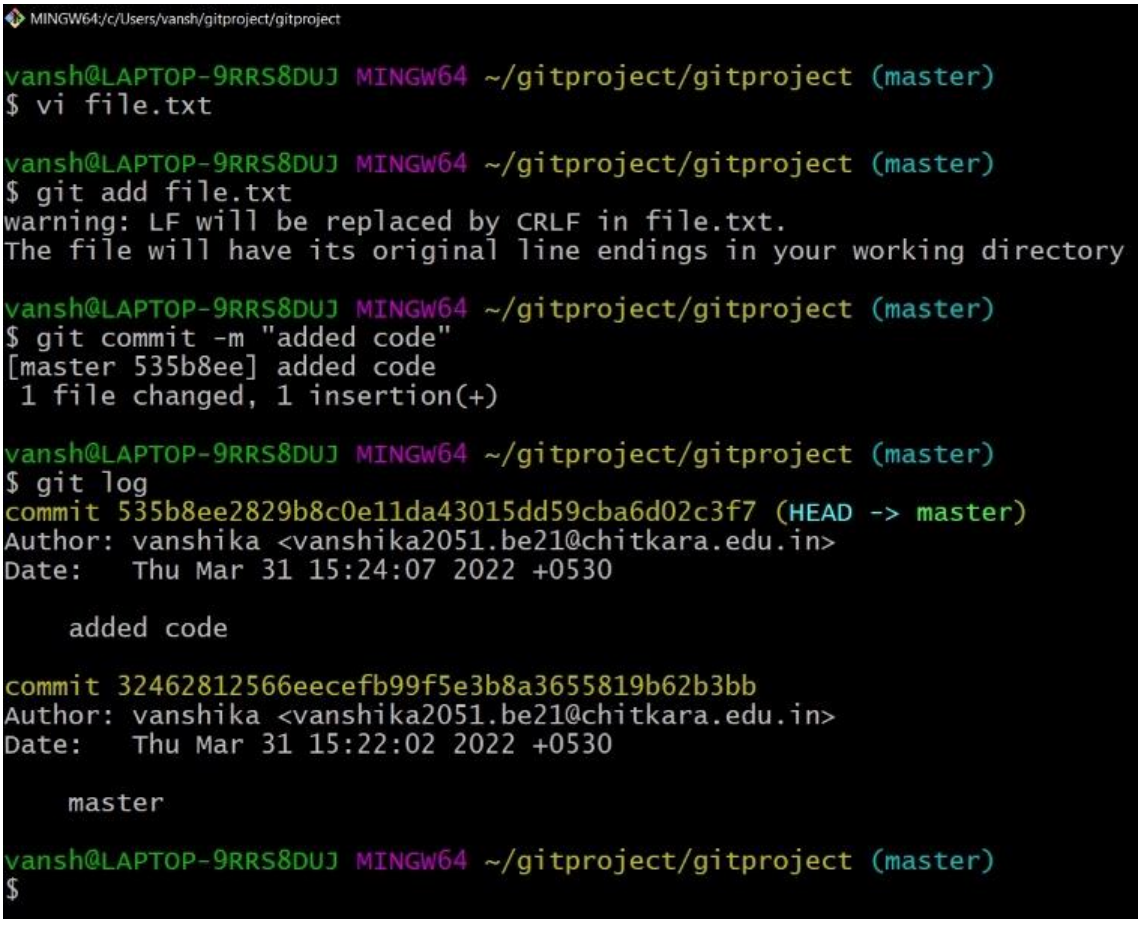
Theory:

Git Logs:

Logs are nothing but the history which we can see in Git by using the code Git log. It contains all the past commits, insertions and deletions which can be seen anytime.

Why do we need logs?

Logs help us to check the changes made in code or files and by whom. It also contains the details of insertions and deletions and also the time it was changed at.



- **Use command** git log **to access logs.**

Aim: Create and visualize branches

Theory:

How to create branches?

The main branch in git is called the master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the files which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches).

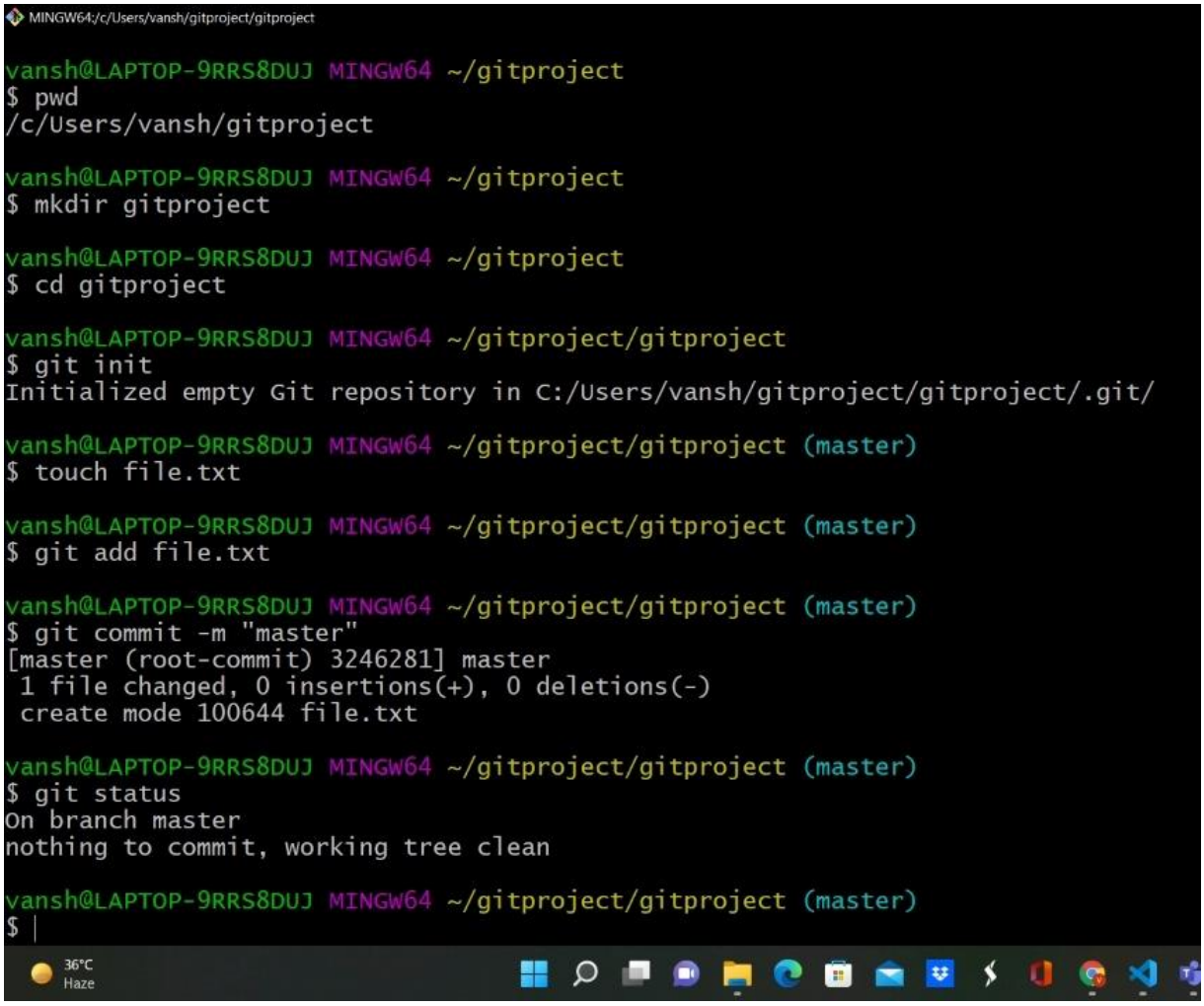
1. **For creating a new branch:** git branch "name of the branch"

2. **To check how many branches we have:** git branch

3. **To change the present working branch:** git checkout "name of the branch"

Visualizing branches:

To visualize, we have to create a new file in the new branch “activity1” instead of the master branch. After this we have to do three step architecture i.e working directory, staging area and git repository.



After this I have done the 3 Step architecture which is tracking the file, send it to stagging area and finally we can rollback to any previously saved version of this file.

After this we will change the branch from activity1 to master, but when we switch back to master branch the file we created i.e “hello” will not be there. Hence the new file will not be shown in the master branch. In this way we can create and change different branches. We can also merge the branches by using the git merge command.

In this way we can create and change different branches. We can also merge the branches by using git merge command.

Experiment No. 05

Aim: Git lifecycle description

Theory:

Stages in GIT Life Cycle:

Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

- Working directory
- Staging area
- Git directory

Working Directory:

Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory.

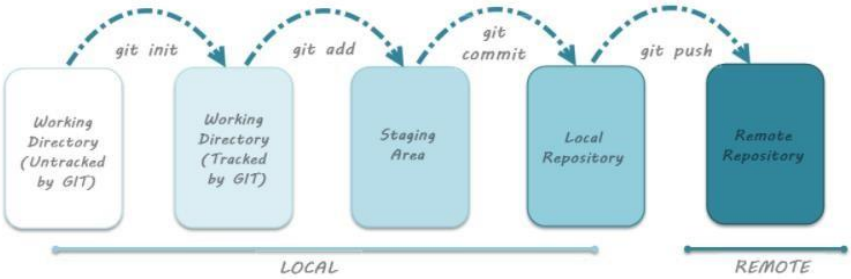
Staging Area:

Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.

Git Directory:

Now that the files to be committed are grouped and ready in the staging area, we can commit these files. So, we commit this group of files along with a commit message explaining what is the commit about. Apart from commit message, this step also records the author and time of the commit. Now, a snapshot of the files in the commit is recorded by Git. The information related to this commit is stored in the Git directory.

Remote Repository: means mirror or clone of the local Git repository in GitHub. And pushing means uploading the commits from local Git repository to remote repository hosted in GitHub.



Snapshots: