**File**

**of**

**Source Code Management**

**by**

**Name: Bhawana Yadav**

**Roll No: 2110990367**

**Group: 8**

**Cluster: Beta**

Index

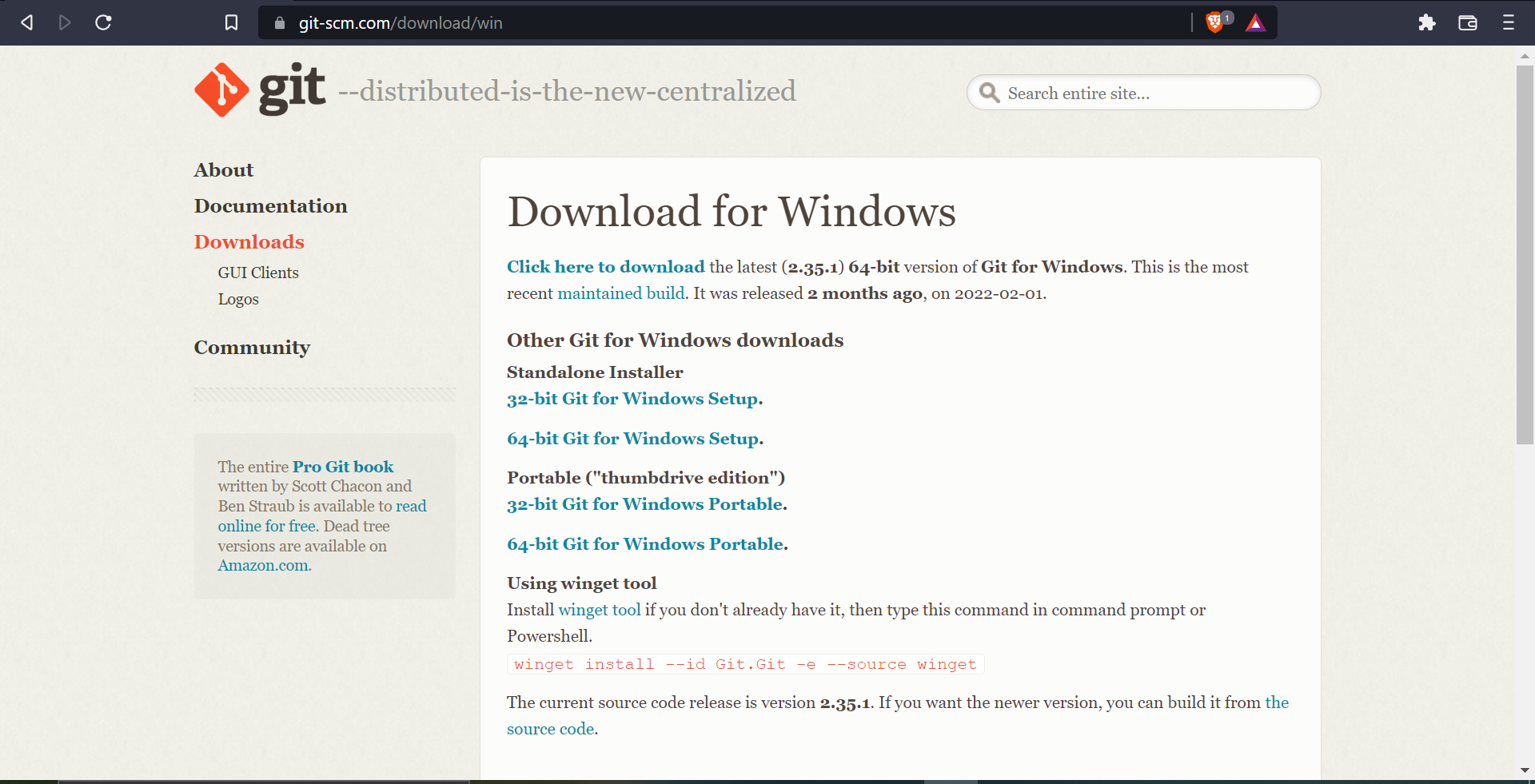
|  |  |  |
| --- | --- | --- |
| Experiment | Topic | Page No. |
| 1 | Setting up Git Client | 3-6 |
| 2 | setting up GitHub Account | 7-8 |
| 3 | How to use Git log | 9-12 |
| 4 | Create and visualize Branch | 13-14 |
| 5 | Life Cycle of the Git | 15-20 |

**Experiment 1**

**Aim**: Setting up Git **client.**

**Description**: Git installation: Git client can be installed on various devices(Windows,Mac or Linux) from <https://git-scm.com/download/win>

You can always go for the default settings. It is absolutely fine or as per your choices you can change the default settings while installation.



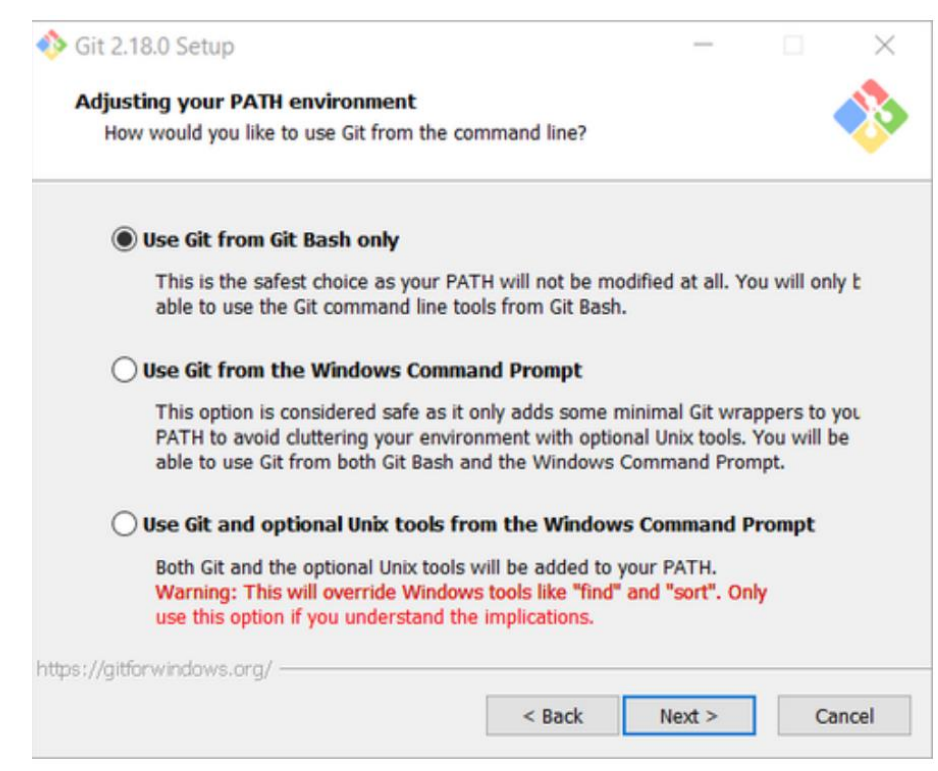
I will recommend going for default settings.

In the Adjusting your PATH screen, all three options are acceptable:

1. Use Git from Git Bash only: no integration, and no extra command in your command path.

2. Use Git from the windows Command Prompt: add flexibility – you can simply run git from a windows command prompt, and is often the setting for people in industry – but this does add some extra commands.

3. Use Git and optional Unix tools from the Windows Command Prompt: this is also a robust choice and useful if you like to use Unix like commands like grep.



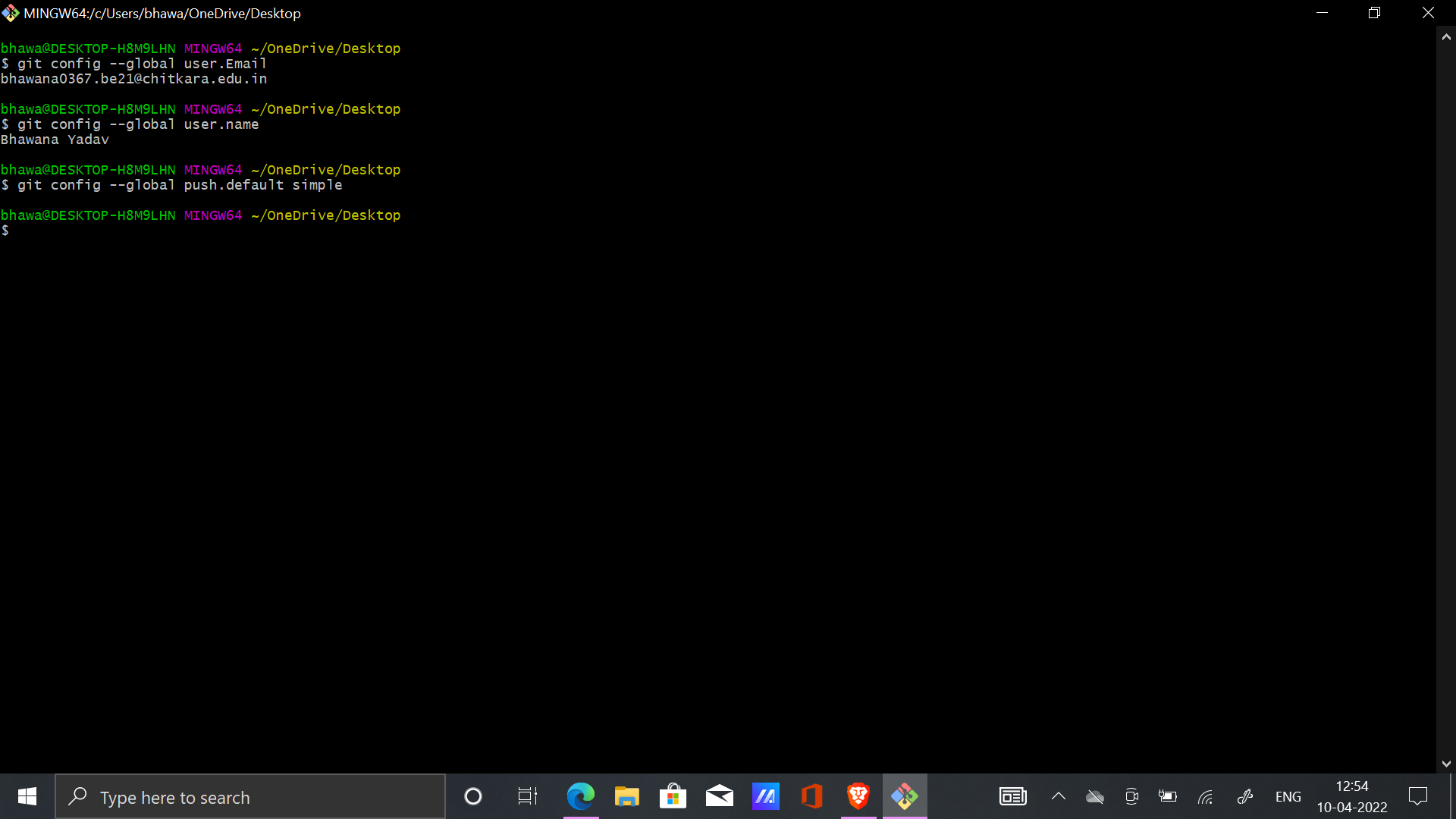
After the installation process is complete, there is some custom configuration you must do. Follow the instructions below

a. From within File Explorer, right-click on any folder. A context menu appears containing the commands " Git Bash here" and "GitGUI here". These commands permit you to launch either Git client. For now, select Git Bash here.

b. Enter the command git config --global user. Email "name@msoe.edu" This links your Git activity to your email address. Without this, your commits will often show up as "unknown login". Replace name with your own MSOE email name.

c. Enter the command git config --global user.name "Your Name" Git uses this to log your activity. Replace "Your Name" by your actual first and last name.

d. Enter the command git config --global push. Default simple This ensures that all pushes go back to the branch from which they were pulled. Otherwise pushes will go to the master branch, forcing a merge.

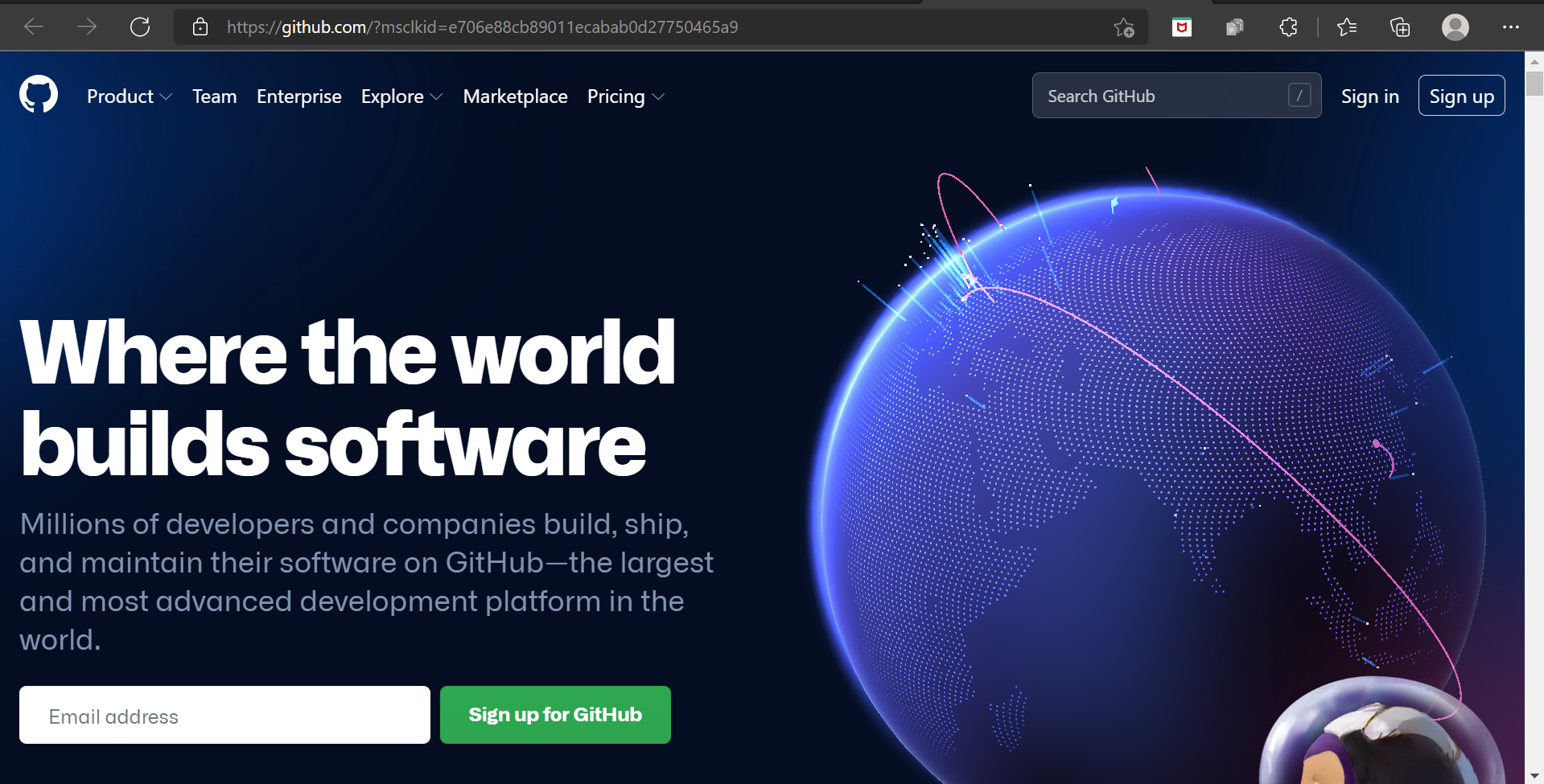


**Experiment 2**

**Aim**: Setting up **GitHub Account**

**Description:**

**Step 1**. Creating an account: To sign up for an account on GitHub.com, navigate to [GitHub: Where the world builds software · GitHub](https://github.com/?msclkid=e706e88cb89011ecabab0d27750465a9) and follow the prompts. To keep your GitHub account secure you should use a strong and unique password. For more information, see “Creating a strong password”.



**Step 2.** Choosing your GitHub product: You can choose GitHub Free or GitHub Pro to get access to different features for your personal account. You can upgrade at any time if you are unsure at first which product you want.

**Step 3.**Verifying your email address: To ensure you can use all the features in your GitHub plan, verify your email address after signing up for a new account.

**Step 4**. Different Options: there will be different options to select from you can select according to your need and set up your GitHub account successfully.

**Experiment 3**

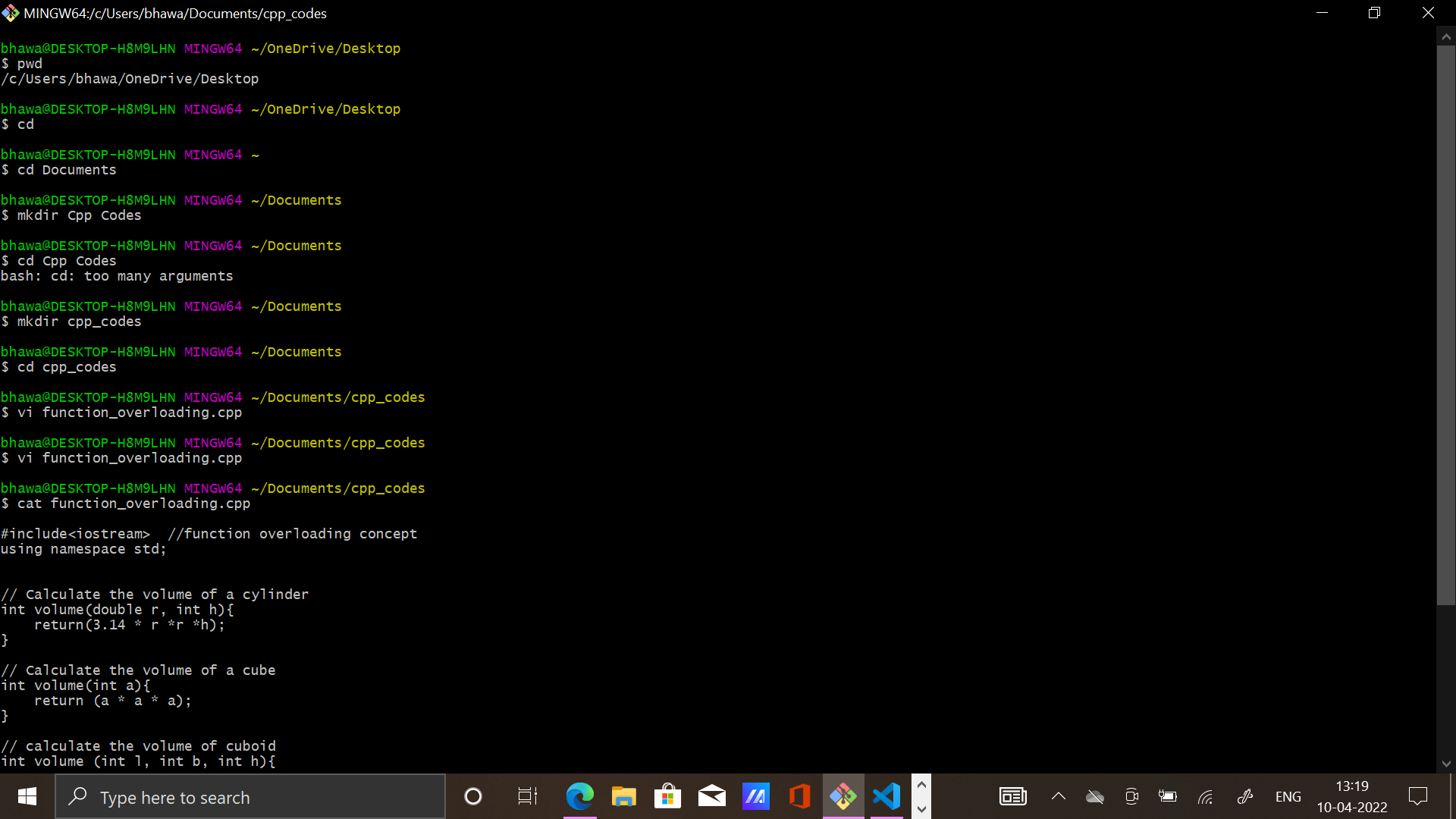
**Aim**: How to use Git log

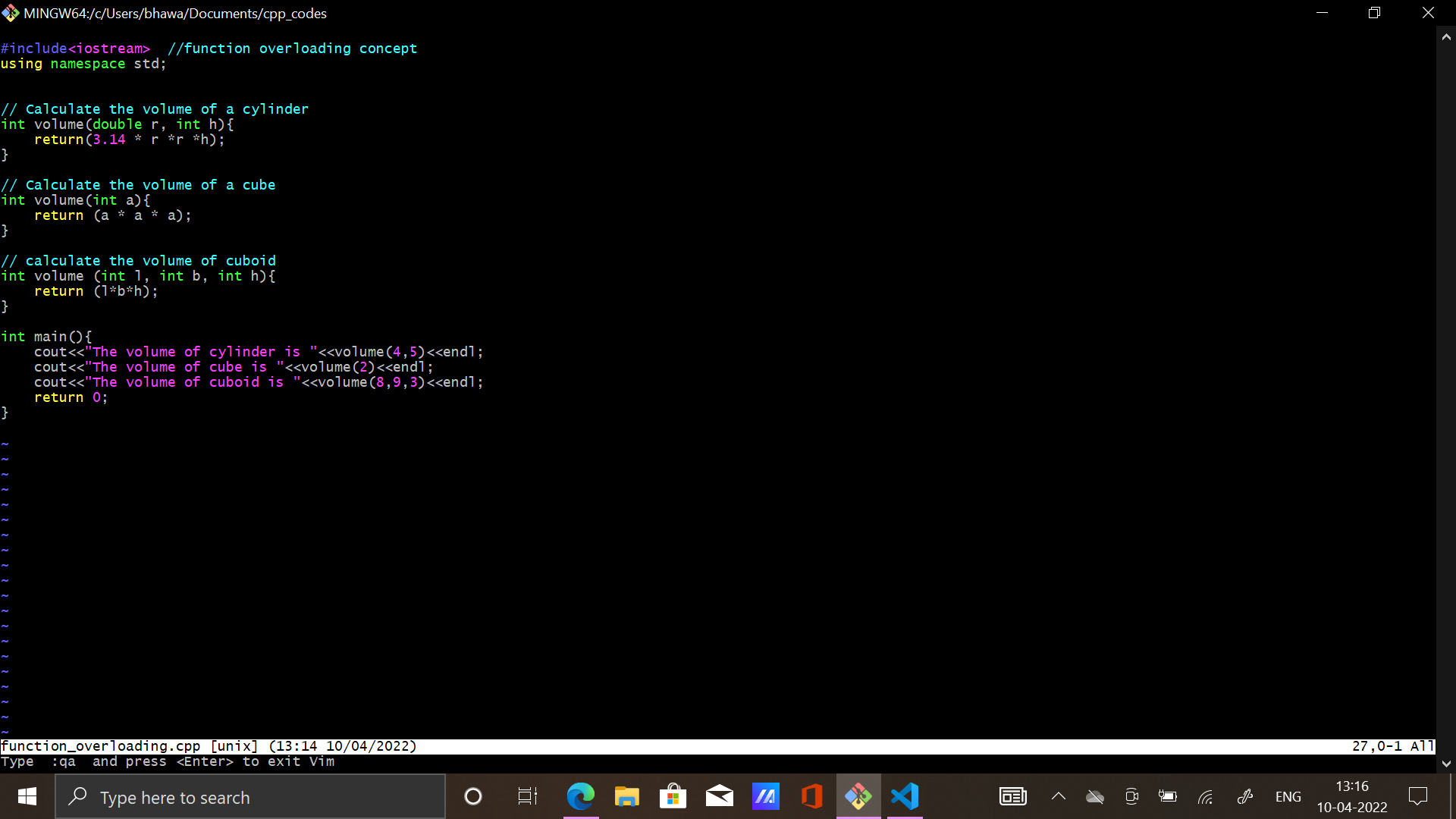
learning basic Git commands and how to make a repository.

**Description**: We Will take a look at some basic commands

1. ls: To list all the content of the current working directory.
2. Clear: To clear the page
3. pwd: Tells about the current working directory
4. cd: changes the directory
5. mkdir: to create new directory
6. vi: Editor that can be used during creating code files for eg:vi hello.cpp
7. :wq: write and quit used for saving the file
8. cat: to see the content inside a file

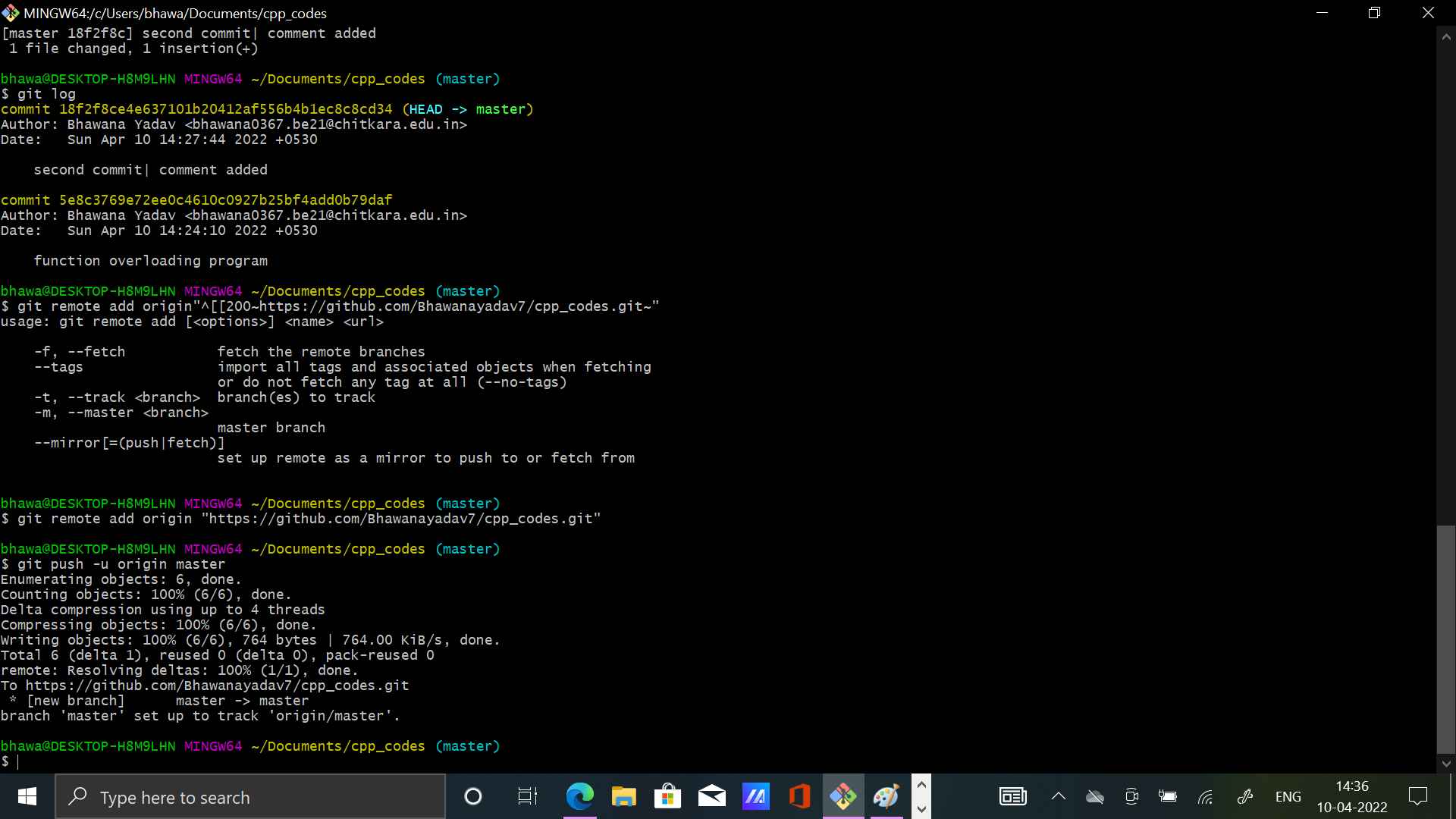
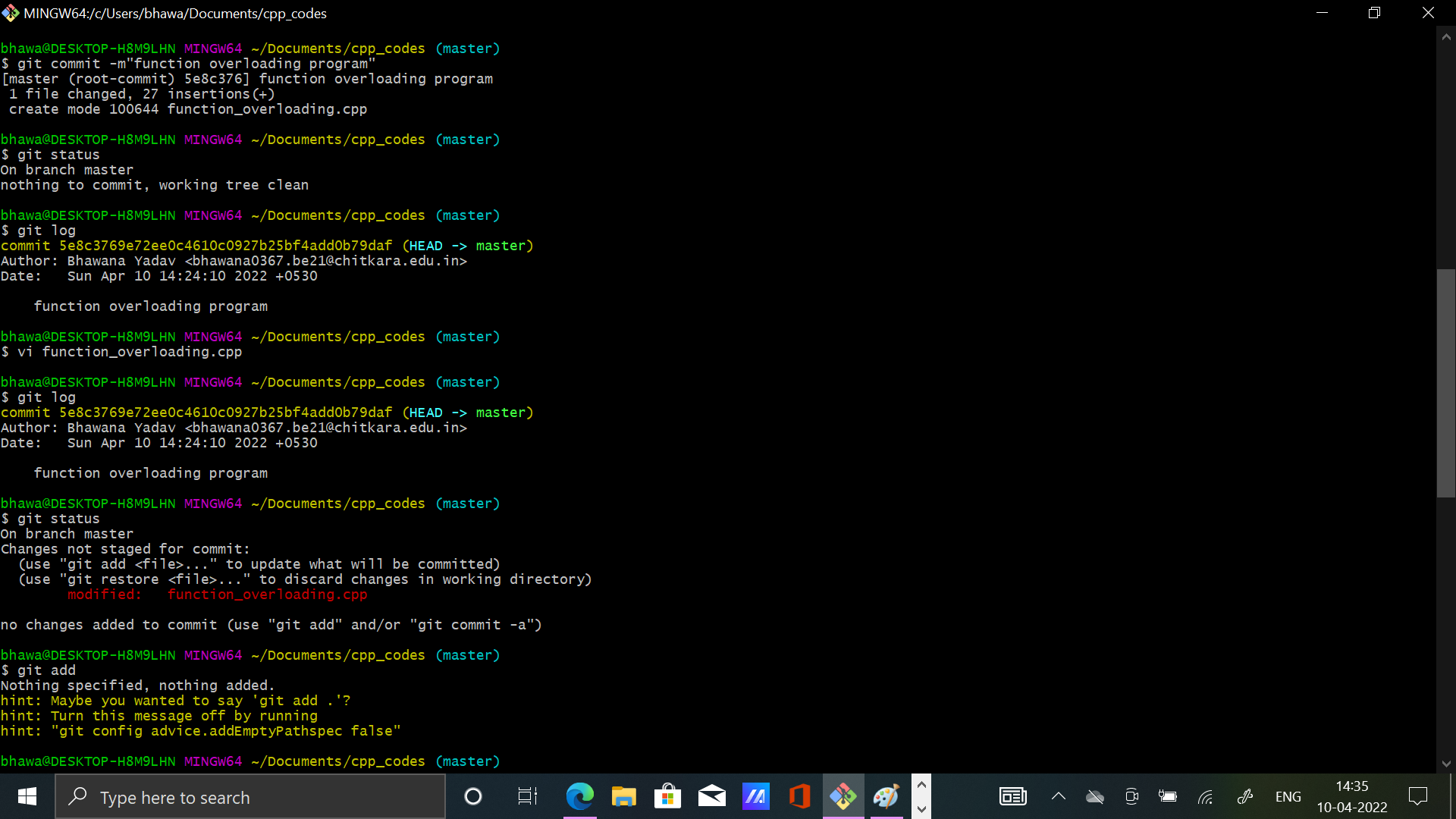
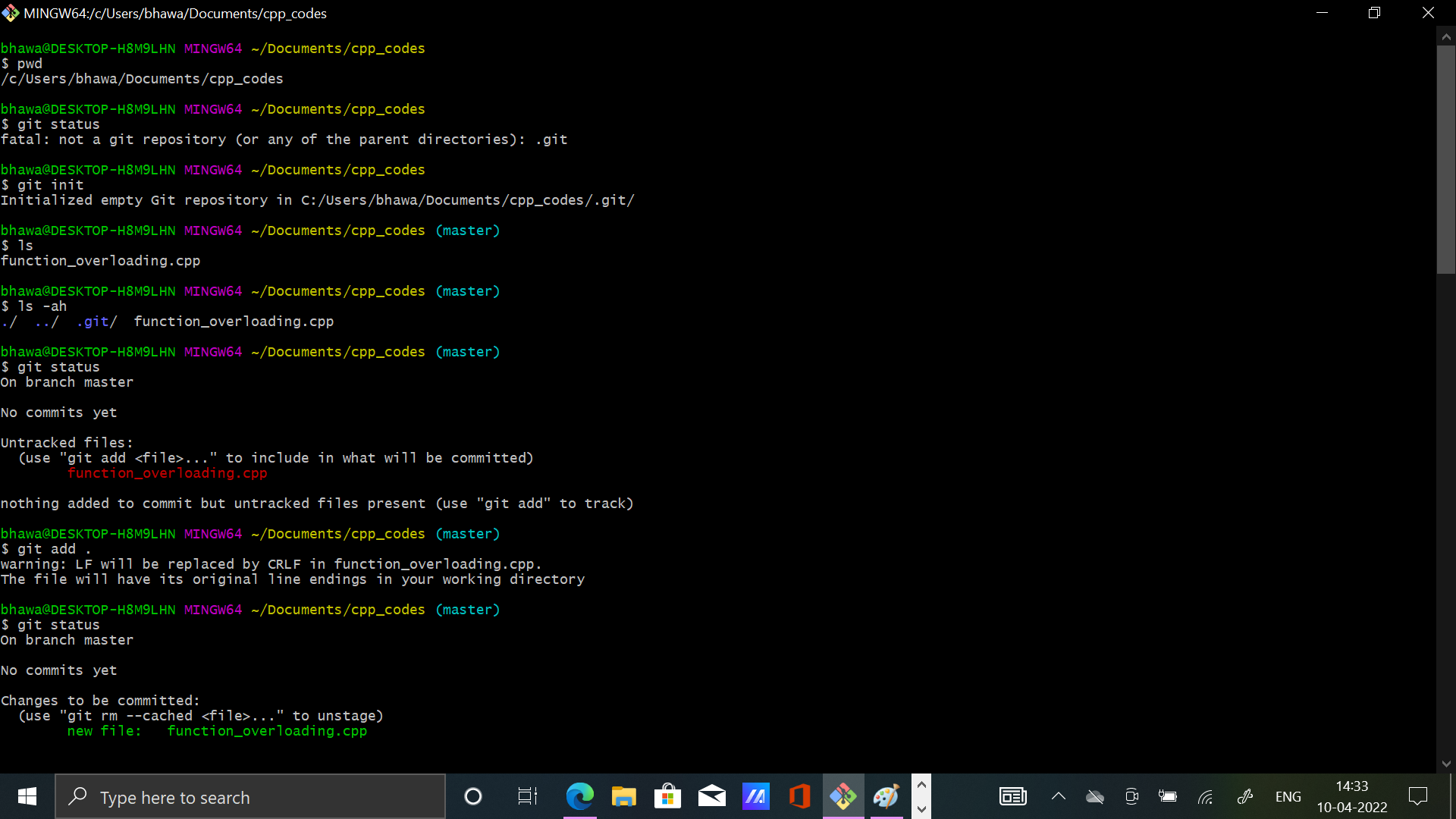
All these commands are shown in the screenshots below you can also take help from there

****



**Running Git commands and making a repository**

1. git status: Tells about the status of a repository.
2. git init: converts the folder into Repository.
3. git add: adds file.
4. git commit – m”message”: to commit inside the Repository.
5. git log: shows commit status.
6. git remote and origin “url link”: refers to remote repository and origin is the default remote name in git
7. git push -u origin master: to push the files into repository



These pictures can help you in setting up a repository and in committing and pushing files.

**Experiment 4**

**Aim**: Create and visualize branches in Git

**~** What are branches?

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master. As you start making commits, you’re given a master branch that points to the last commit you made. Every time you commit, the master branch pointer moves forward automatically.

**~** How to create branches?

The main branch in git is called 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 also can merge both the parent(master) and child (other branches).

1. To check how many branches we have : git branch

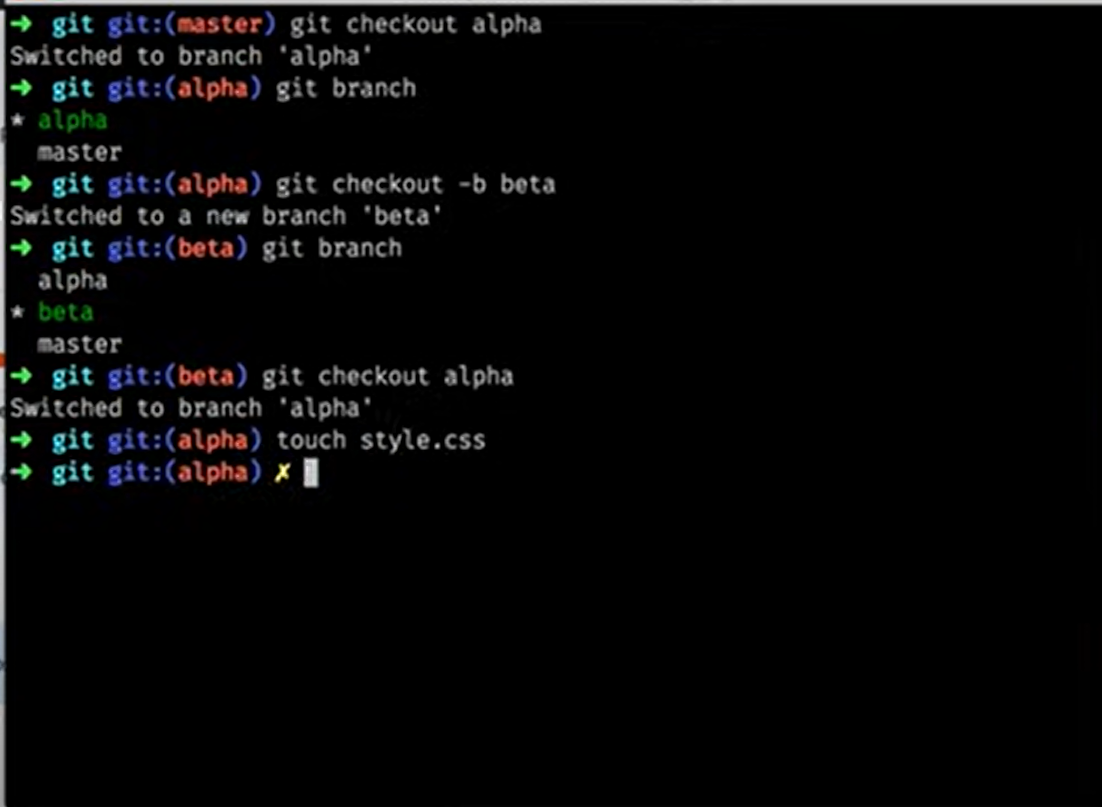
2. For creating a new branch: git branch “name of branch”

3. To change the present working branch: git checkout “name of the branch”

**~**What happens when you create a new branch?

Well, doing so creates a new pointer for you to move around. Let’s say you want to create a new branch called testing.

You do this with the git branch command: $ git branch testing



**Experiment 5**

Aim: Git lifecycle description

Description:

**Stages in GIT Life Cycle**

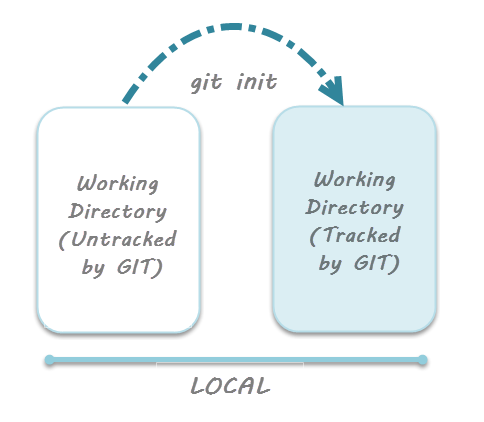
Files in a Git project have various stages like Creation, Modification, Refactoring and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, when a project is under Git version control system, they are present in three major Git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

* ***Working directory***
* ***Staging area***
* ***Git directory***

These stages are the essence of Git. You get great flexibility in tracking the files due to these stages that files can reside in under Git. Let's understand each of these states one by one.

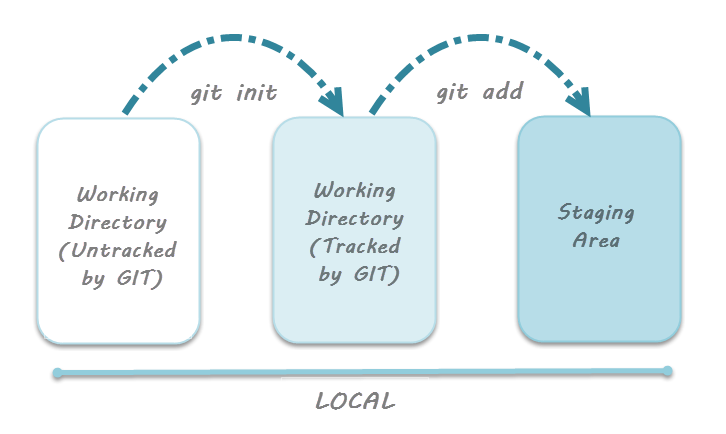
**Working Directory**

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.



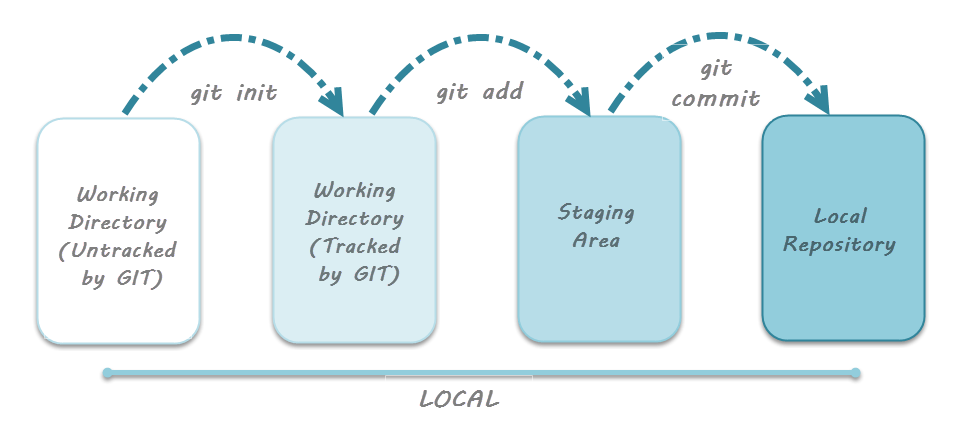
**Staging Area**

**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.



**Additional lifecycle stage with GitHub**

However, we work in teams and collaborate with multiple people on a given project. This makes it imperative to understand the additional stage related to GitHub. While dealing with GitHub, there’s a concept of ***Remote repository*** and a related process called ***Pushing*** the files.

