

|  |
| --- |
|  |

# INDEX-

1. Setting up of Git Client.
2. Creating a GitHub account.
3. Making Directory & Various Commands.
4. Generating logs.
5. Git Branching.
6. Git lifecycle.

### -WHAT IS GIT?

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning-fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

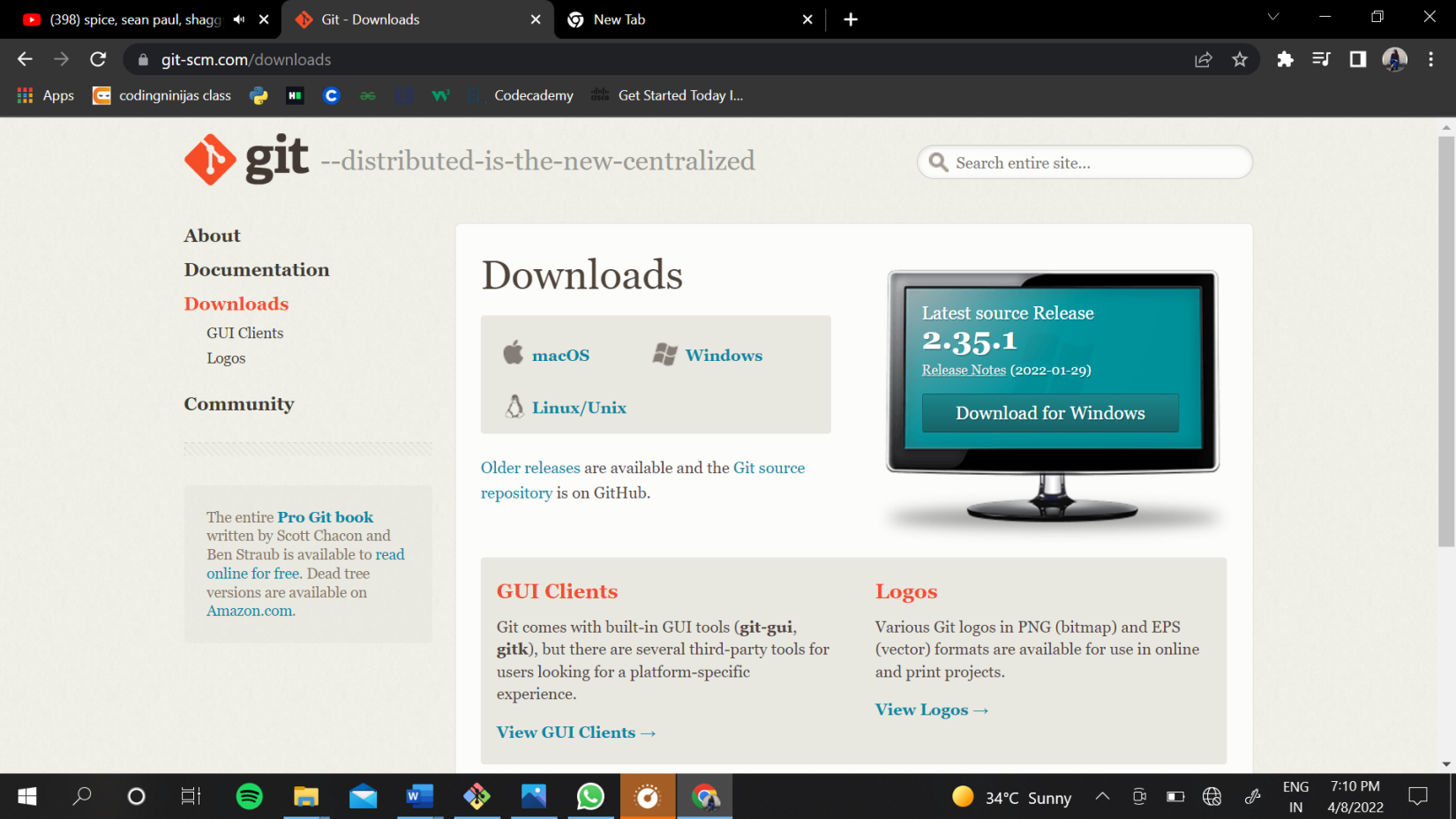
### -WHAT IS GITHUB?

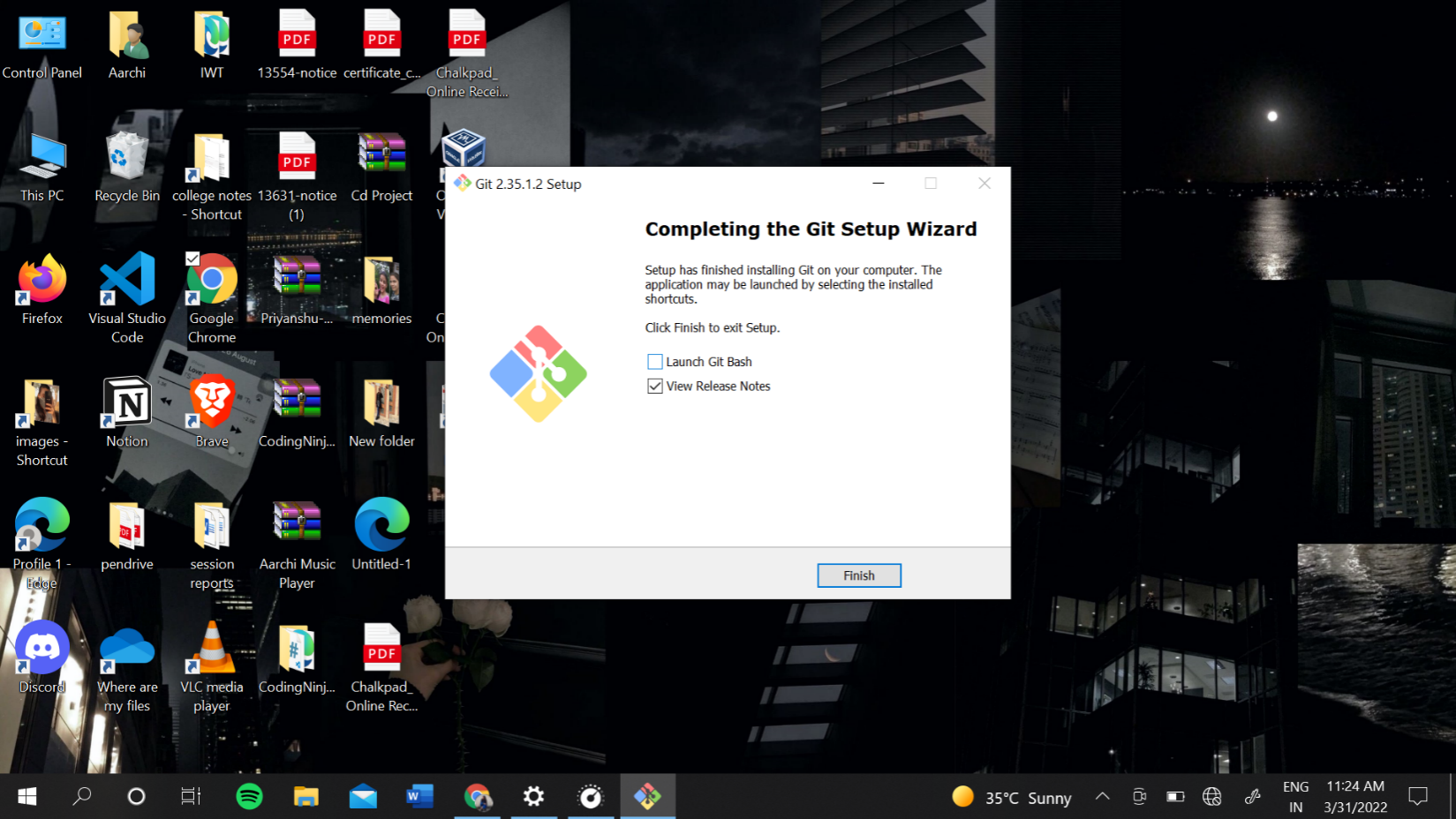
GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code.

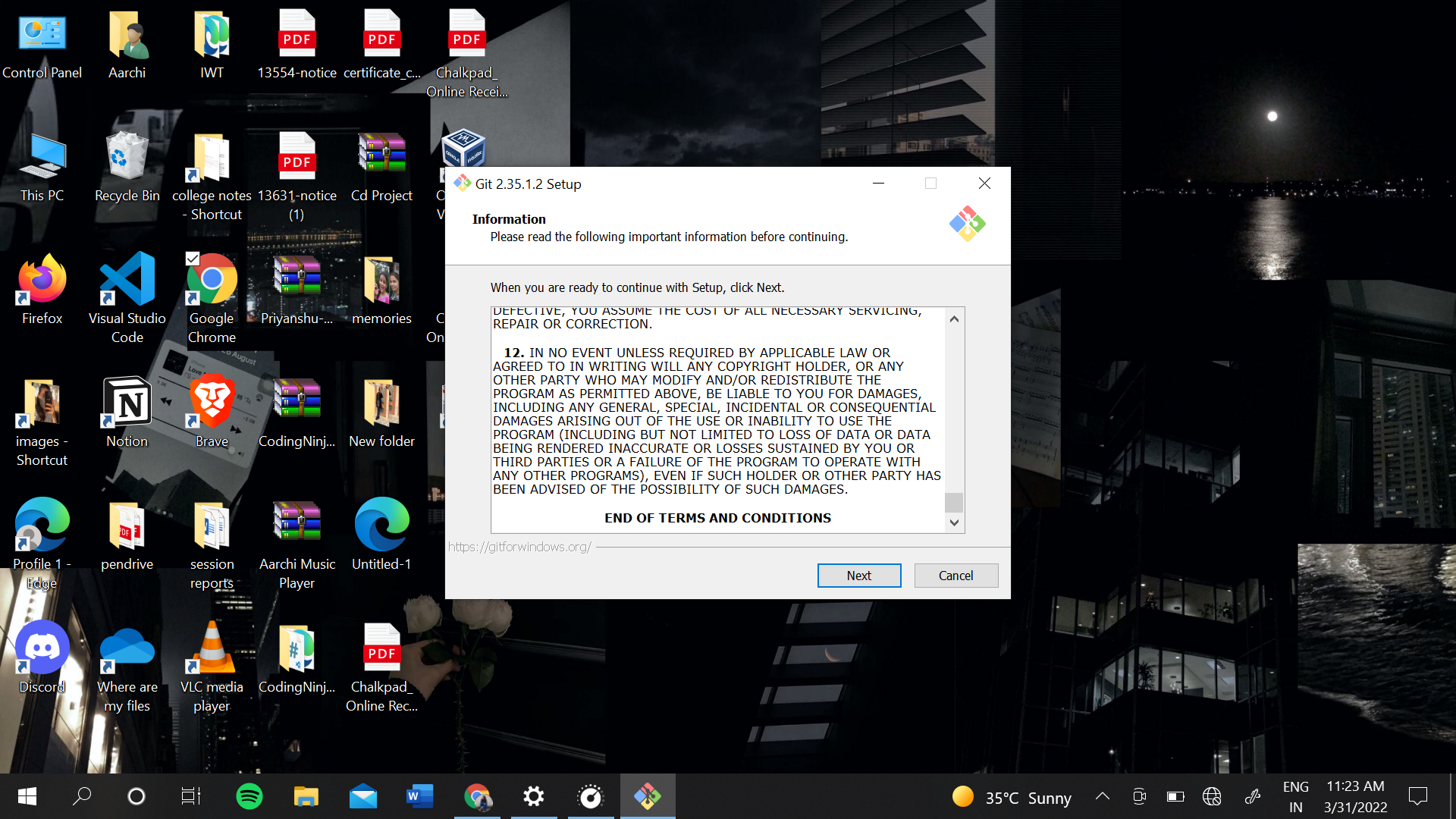
### ---------------------------------------------------------------------------------------

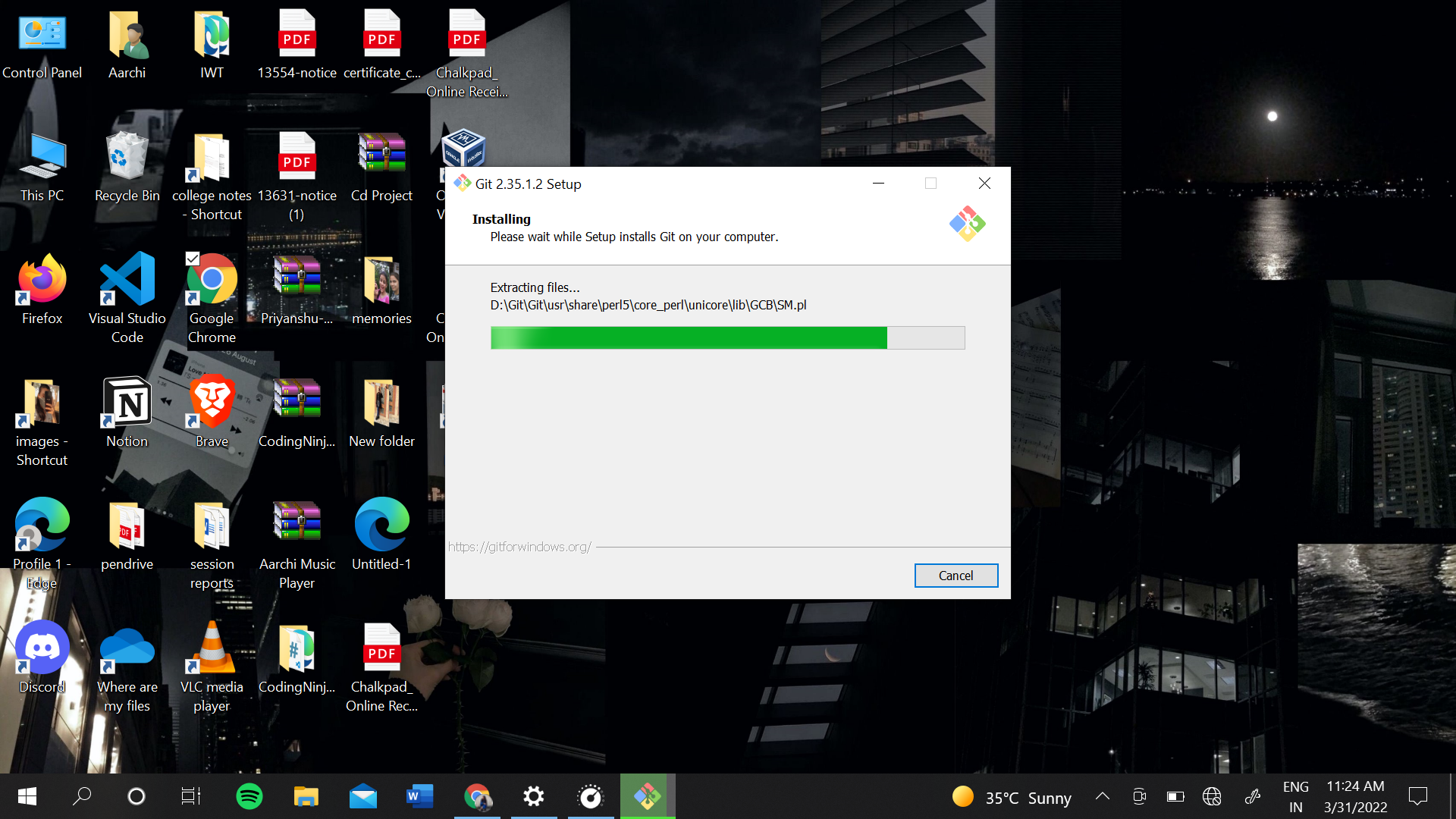
### -INSTALLING GIT

The official build is available for download on the Git website- <https://git-scm.com/download/win> and the download will start automatically. Follow the steps listed further to proceed with the installation.





Accept the terms & conditions



Installing will begin shortly

----------------------------------------------------------------------------------------

### -CREATING GITHUB ACCOUNT

Just go to the website <https://github.com/> and choose the ‘Sign Up” option. Enter the required details and you are done with creating account on GitHub.

----------------------------------------------------------------------------------------------------------------------

### -MAKING A DIRECTORY AND VARIOUS COMMANDS

mkdir=> It is used for creating a directory using Git Bash.

cd command=>Both Bash and Windows console host have a cd command. cd is an acronym for 'Change Directory'. cd is invoked with an appended directory name. Executing cd will change the terminal sessions current working directory to the passed directory argument.

pwd=>The Bash command pwd is used to print the 'present working directory'. This is the folder or path that the current Bash session resides in.

ls=>The Bash command ls is used to 'list' contents of the current working directory.

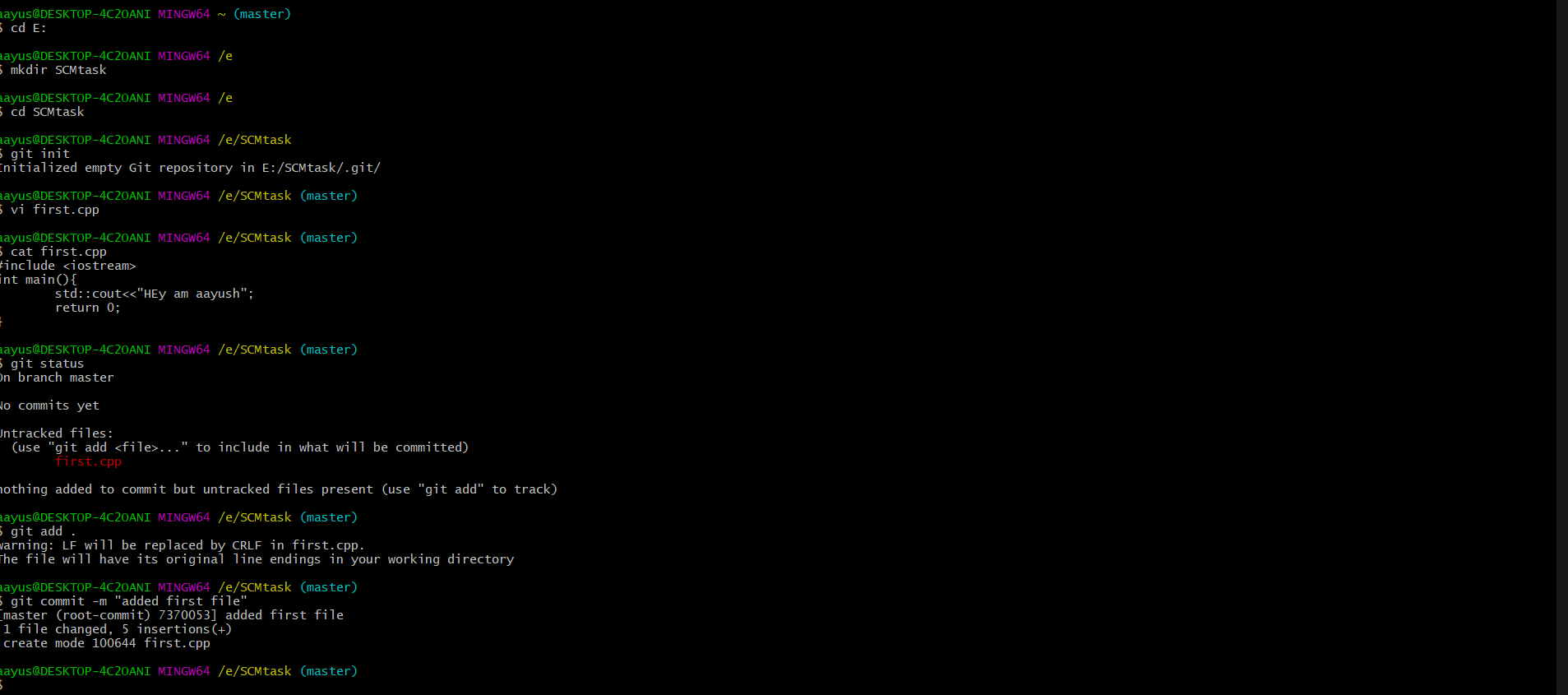
git status=>The main tool you use to determine which files are in which state is the git status command. the command tells you which branch you’re on and informs you that it has not diverged from the same branch on the server.

git add =>In order to begin tracking a new file, you use the command git add. git add is a multipurpose command — you use it to begin tracking new files, to stage files, and to do other things like marking merge-conflicted files as resolved. It may be helpful to think of it more as “add precisely this content to the next commit” rather than “add this file to the project”.

git commit=> The git commit command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as “safe” versions of a project—Git will never change them unless you explicitly ask it to. Prior to the execution of git commit, The git add command is used to promote or 'stage' changes to the project that will be stored in a commit. These two commands git commit and git add are two of the most frequently used.

git push=> When you have your project at a point that you want to share, you have to push it upstream. The command for this is simple: git push . If you want to push your master branch to your origin server (again, cloning generally sets up both of those names for you automatically), then you can run this to push any commits you’ve done back up to the server: $

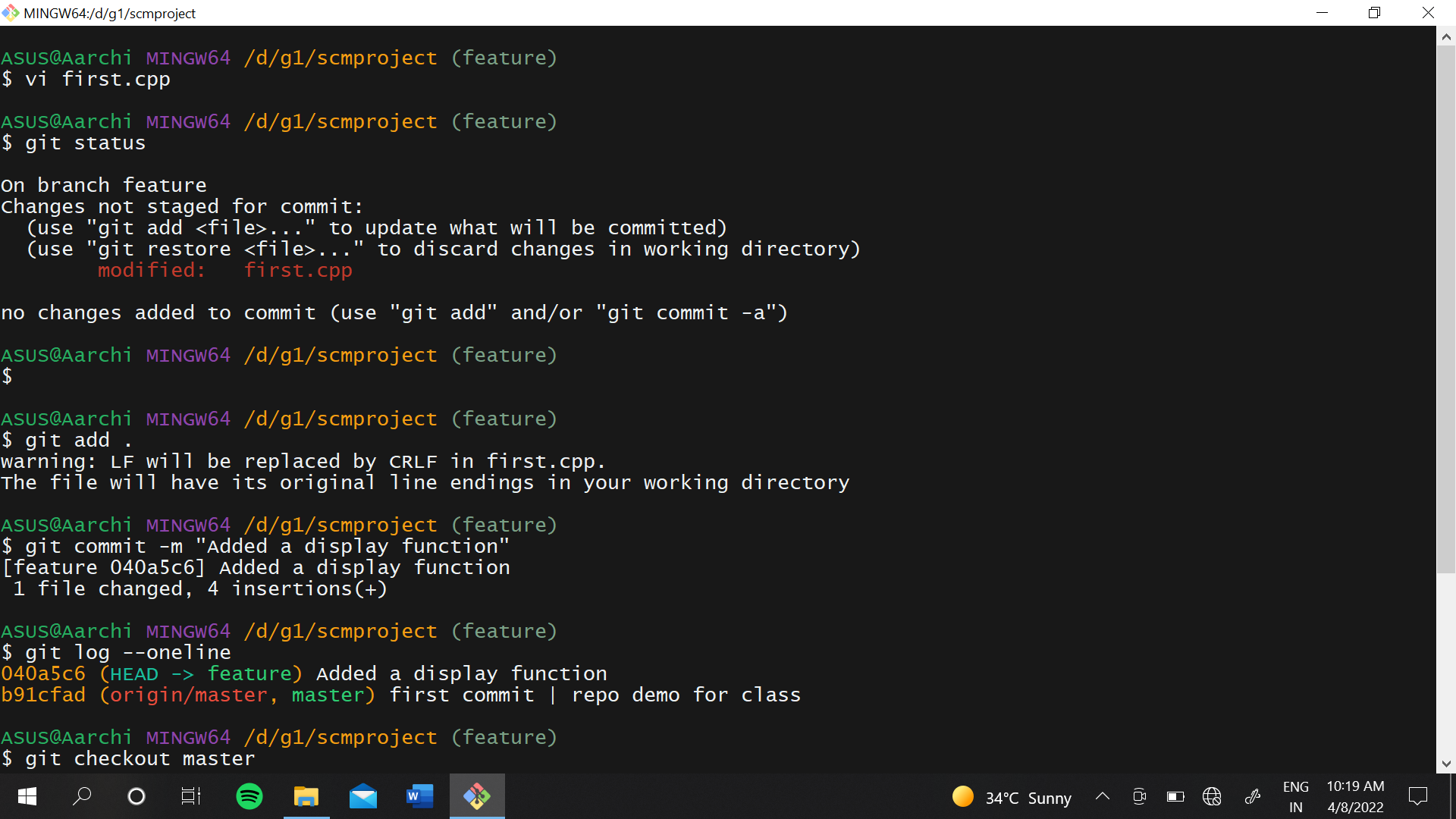
git push origin master.



------------------------------------------------------------------------------------------

### -GENERATING 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.



------------------------------------------------------------------------------------------

### -GIT BRANCHING

A branch in Git is an independent line of work(a pointer to a specific commit). It allows users to create a branch from the original code (master branch) and isolate their work. Branches allow you to work on different parts of a project without impacting the main branch.

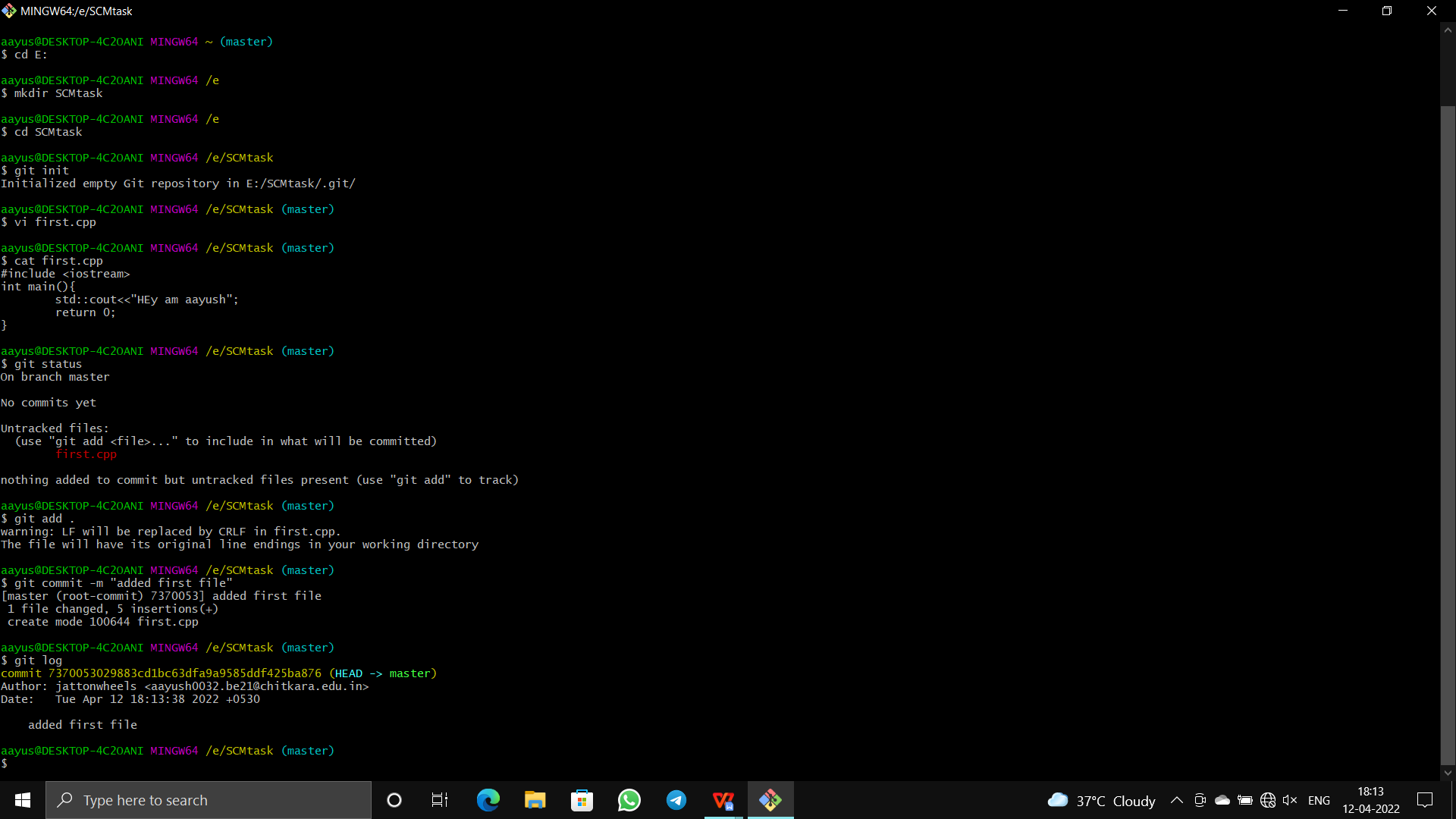
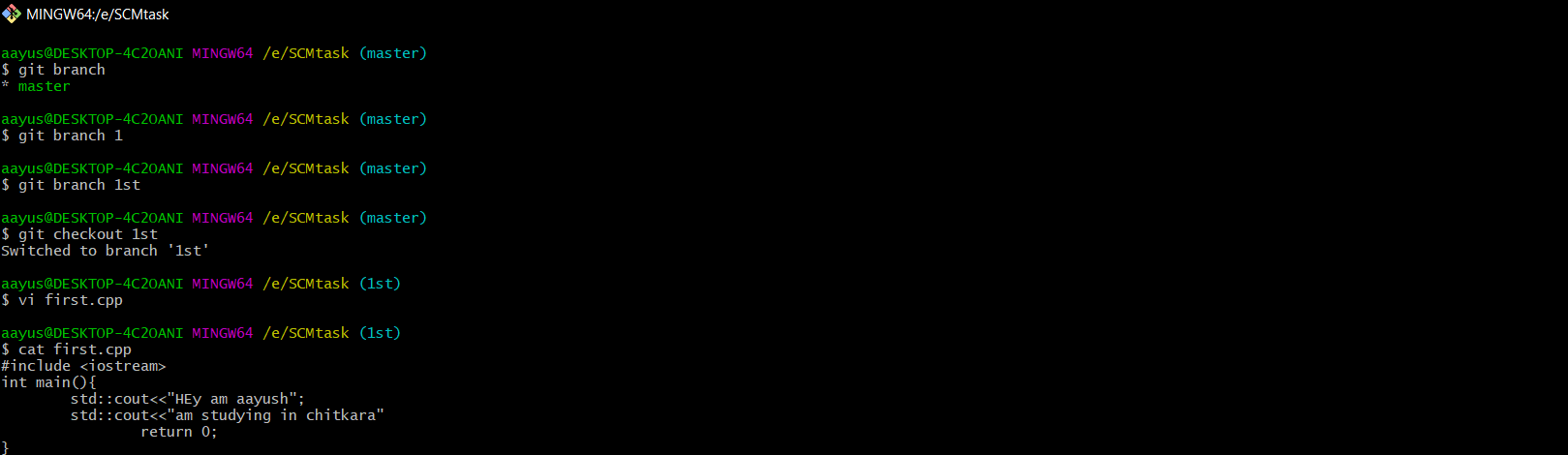
Firstly, add a new branch, let us suppose the branch name is activity1.

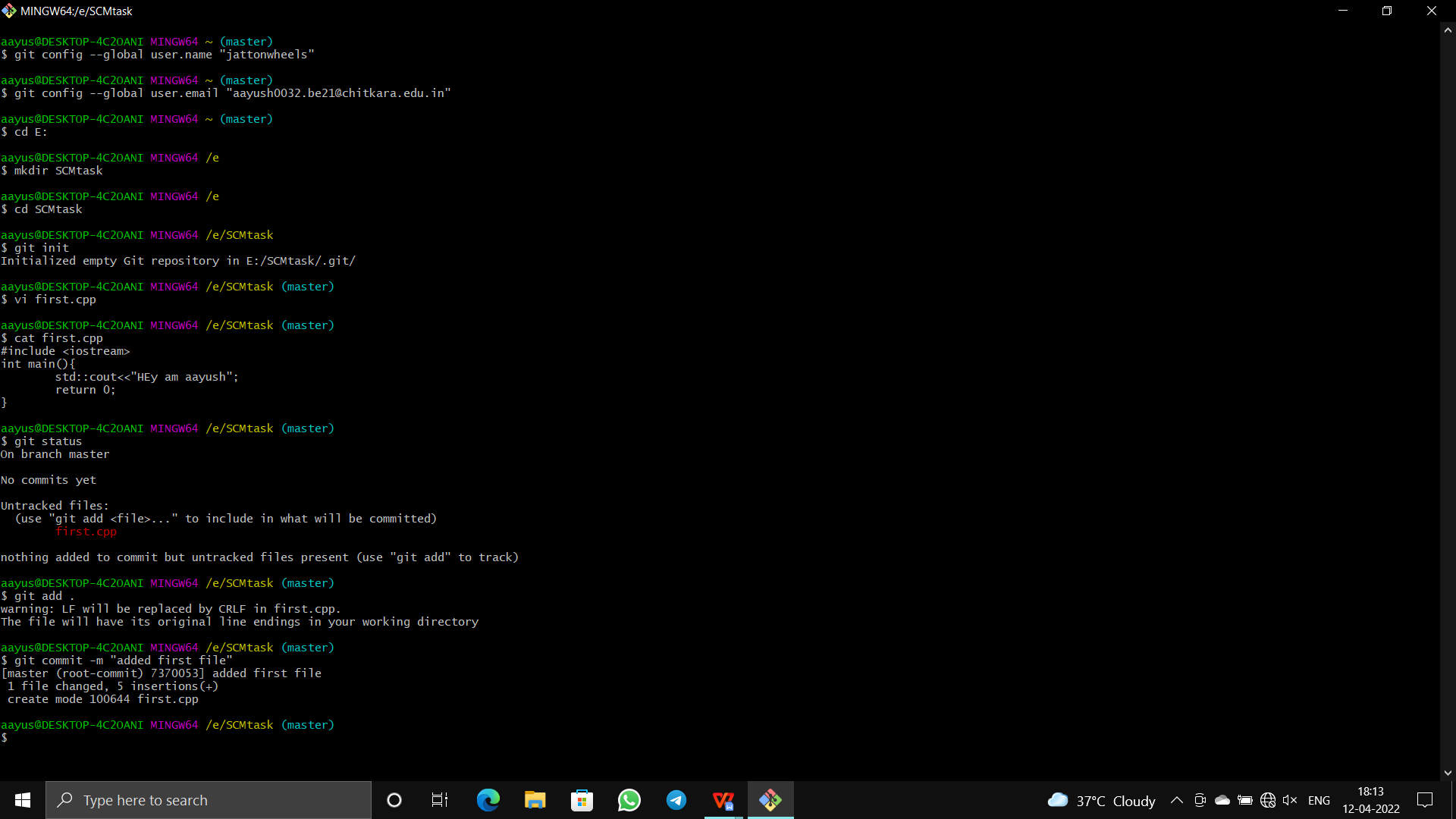
For this use command 🡪

**git branch name [**adding new branch**]**

**git branch [**use to see the branch’s names**]**

**git checkout *branch name* [**use to switch to the given branch]





### -GIT LIFECYCLE.

It is important for us to have an abstract idea of the different stages of Git before going into more detailed understanding of Git.

Files in a **Git** project have various stages like **Creation, Modification, Refactoring**, and **Deletion** and so on.

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.

