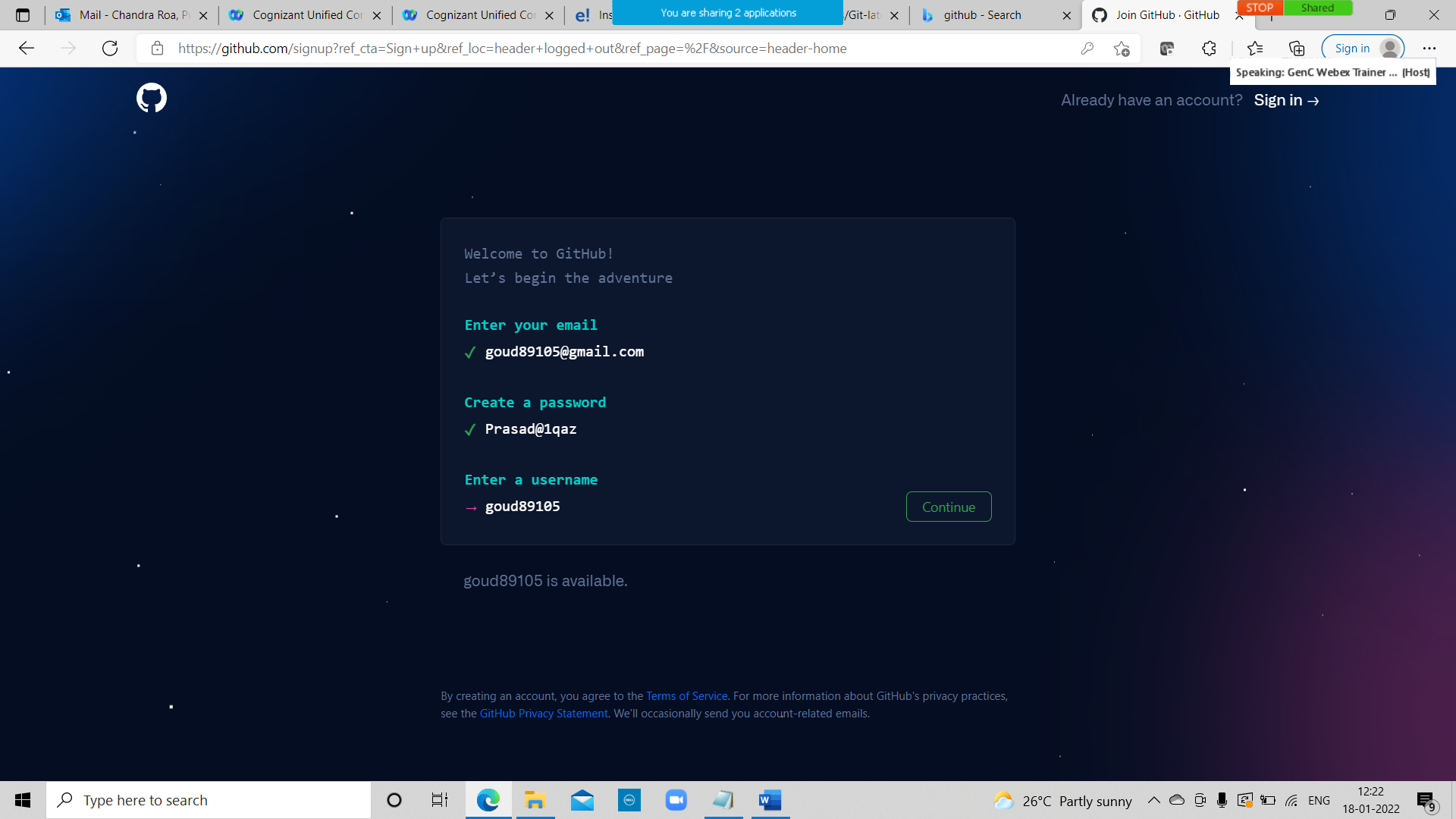
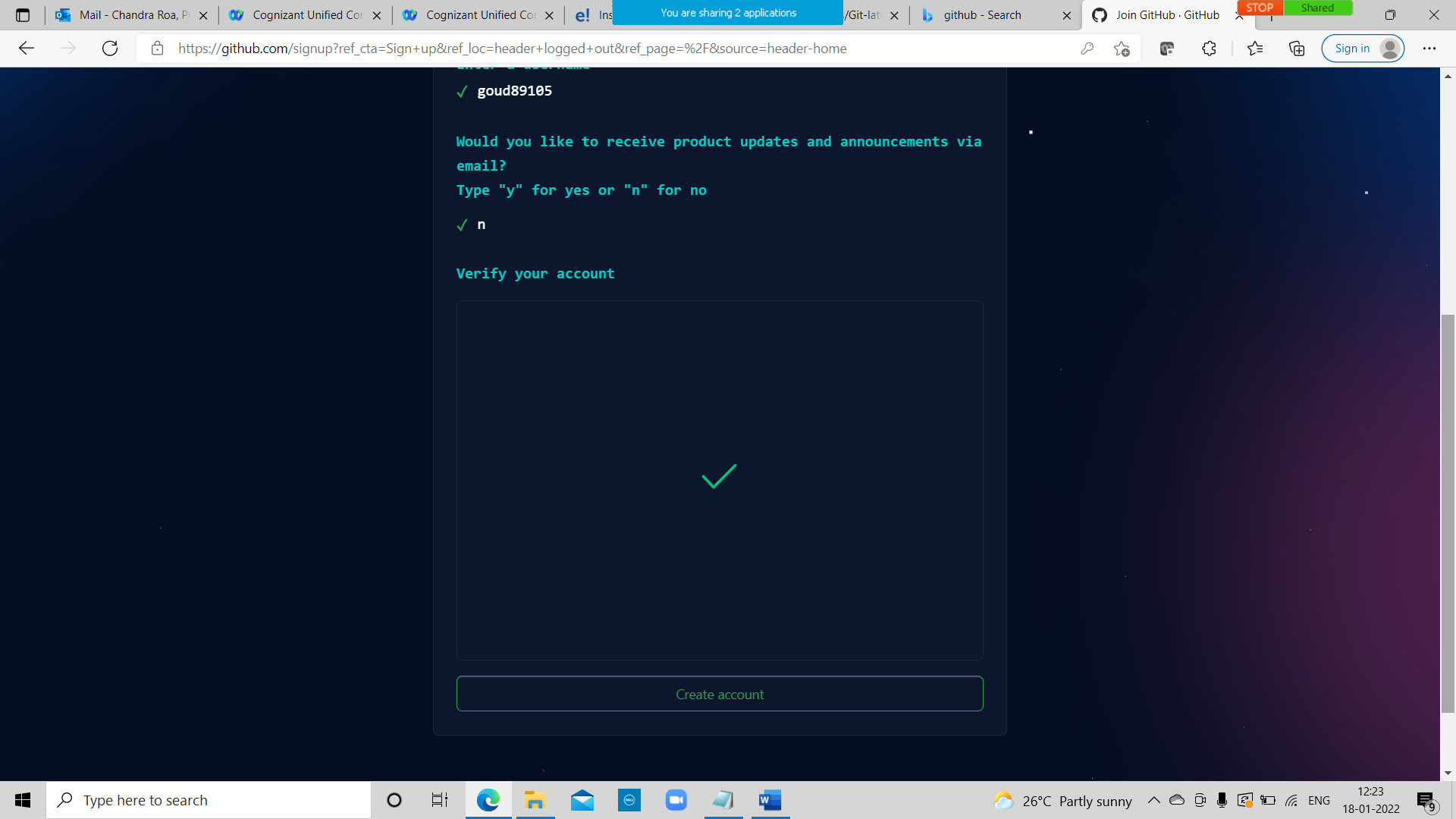
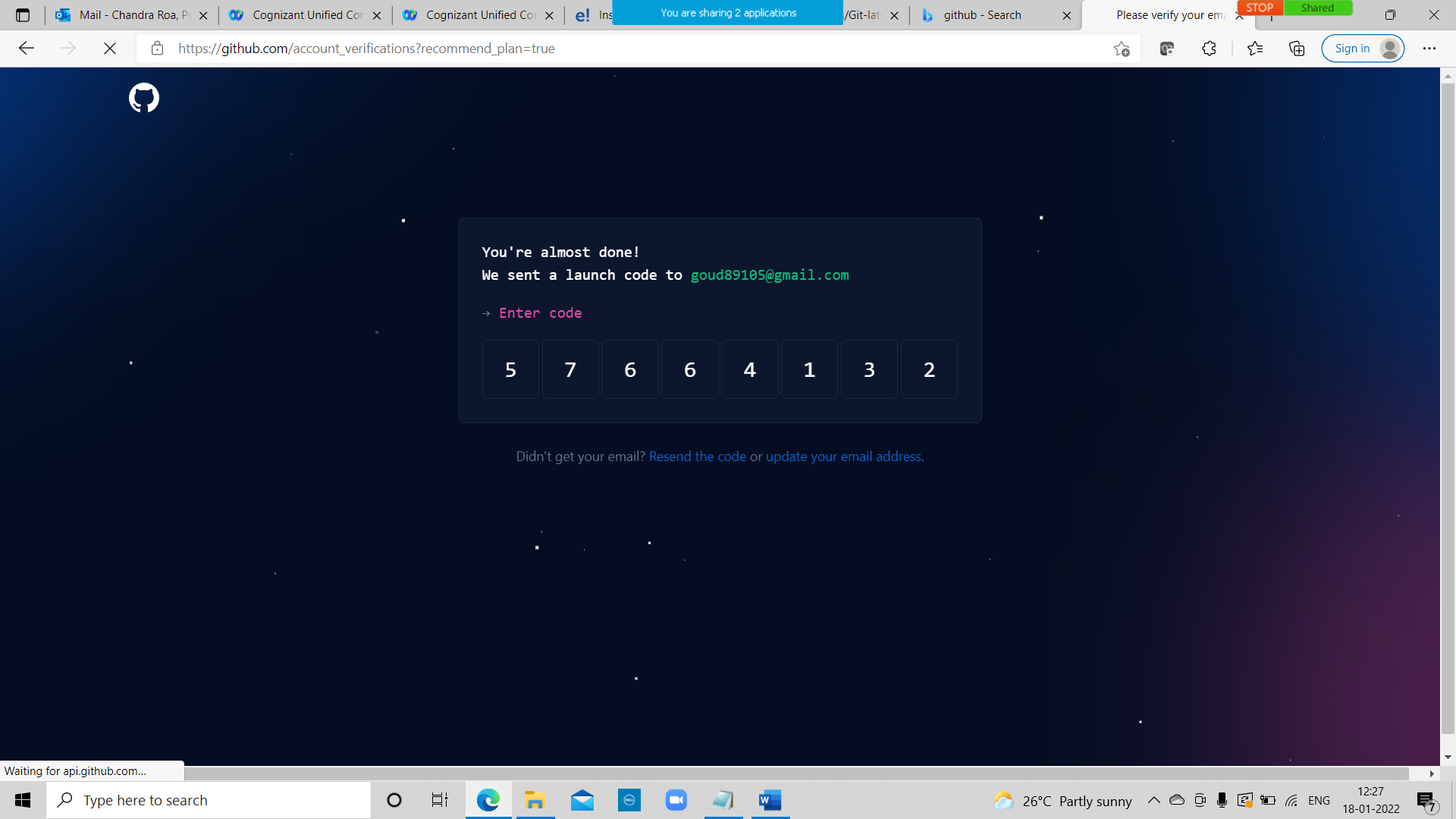
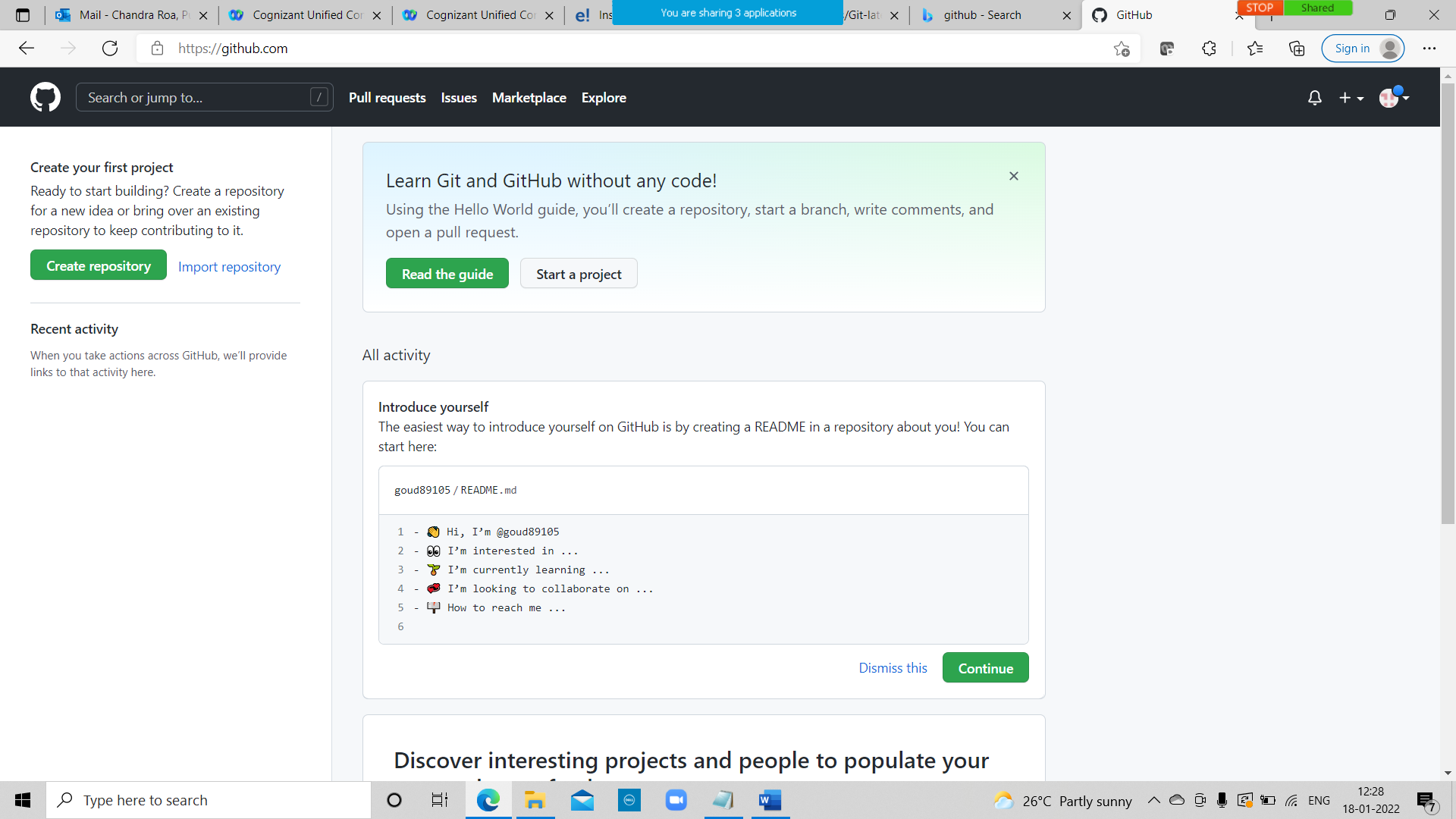
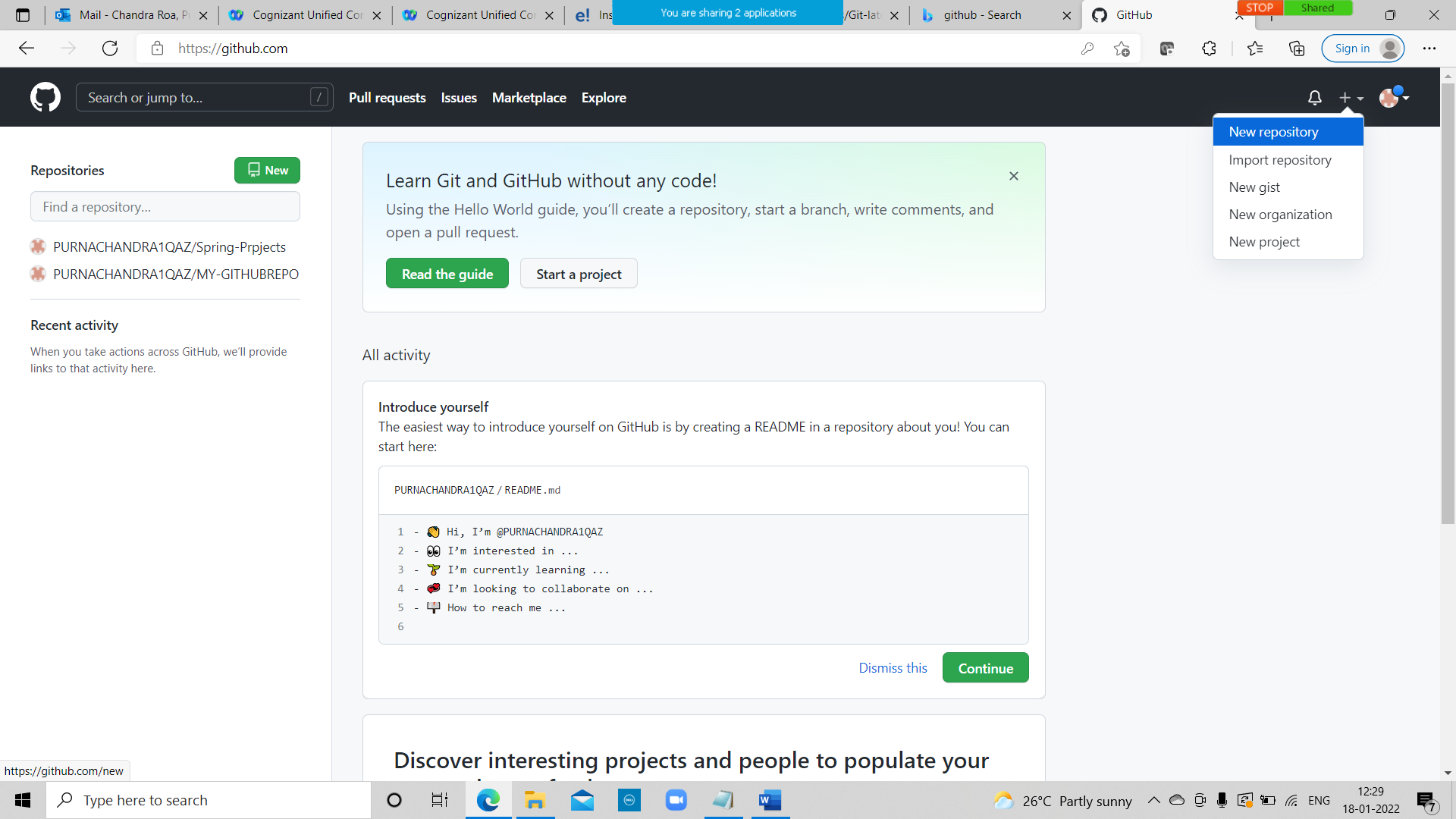
GitHub:

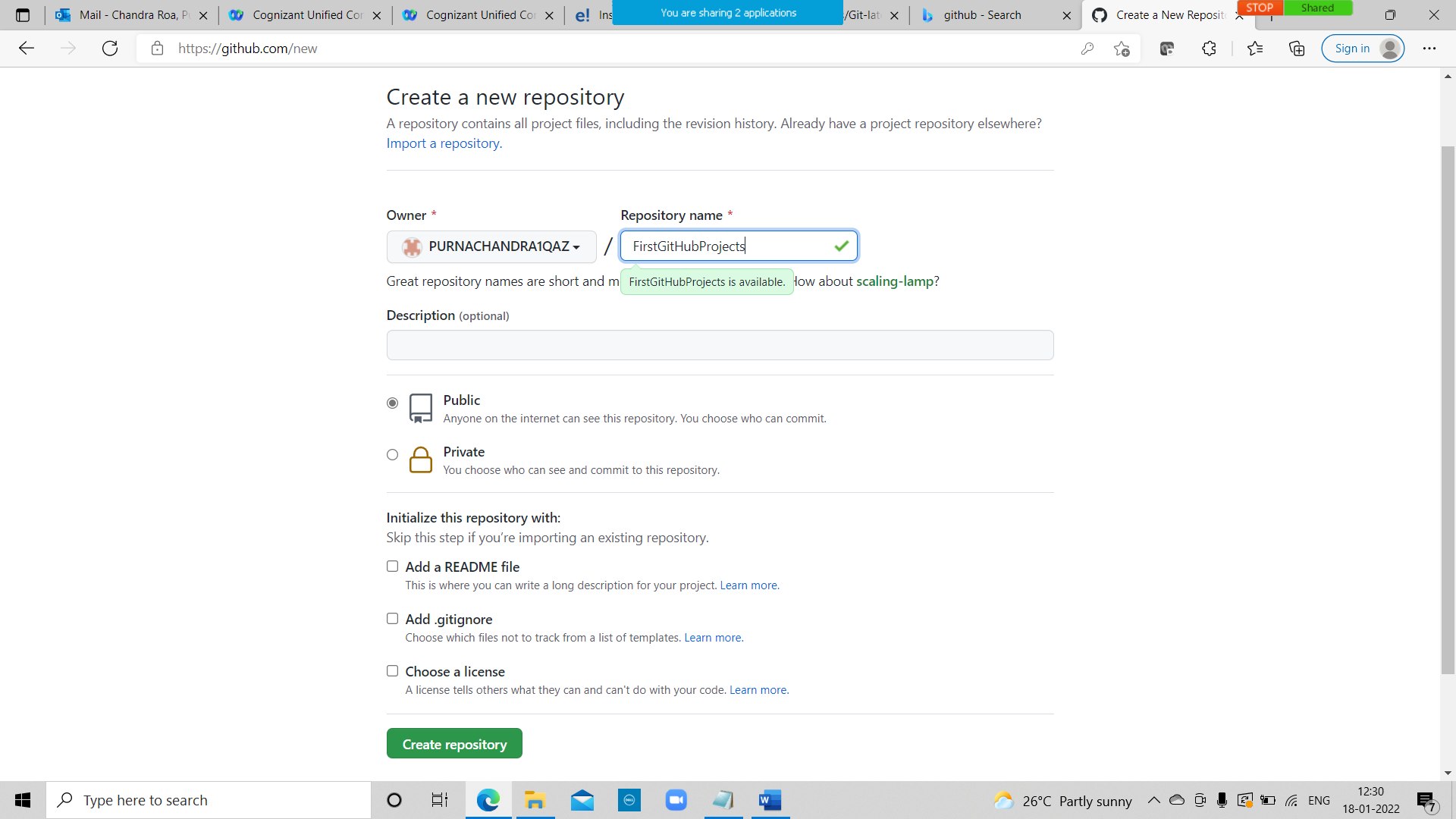




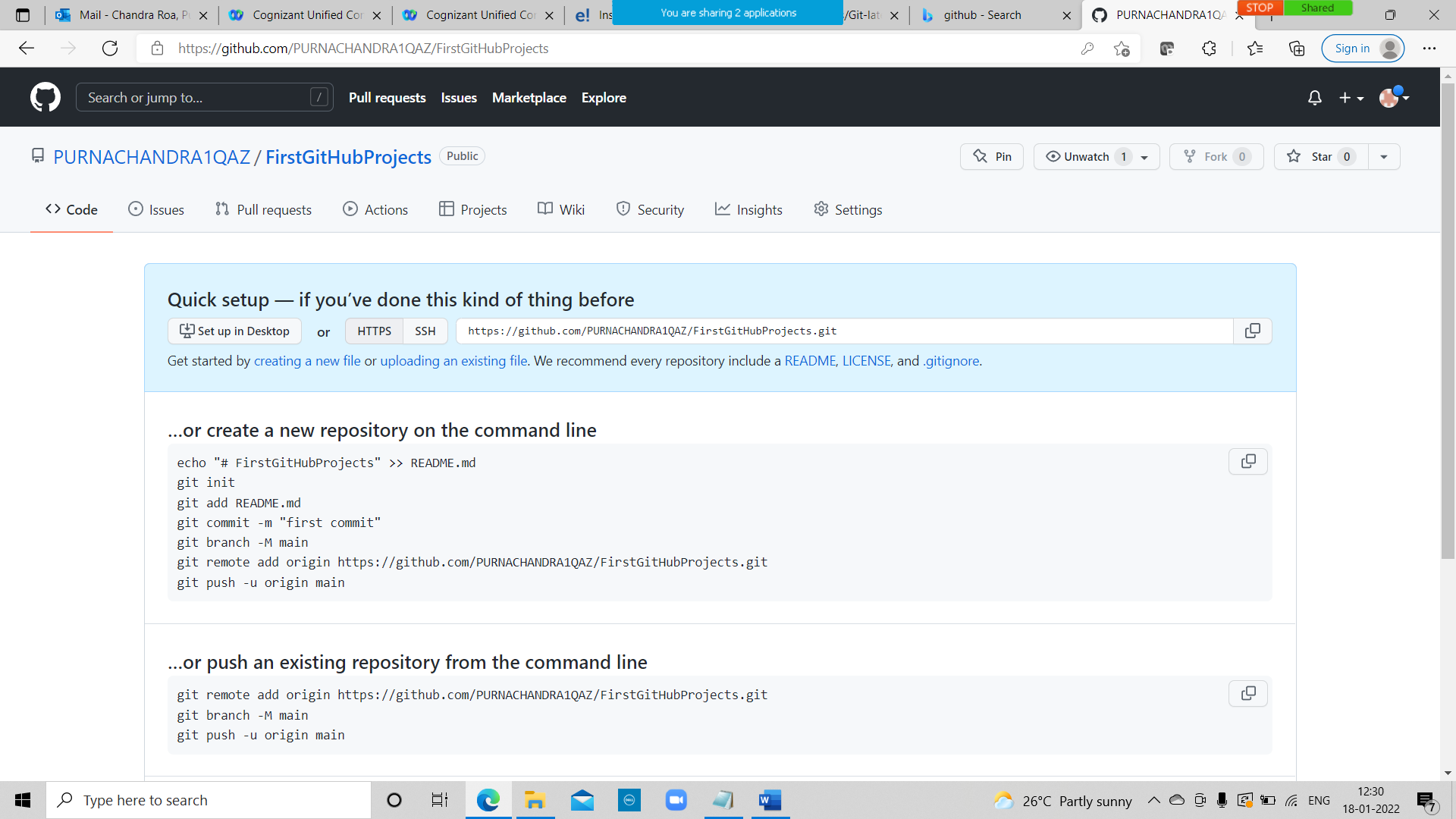




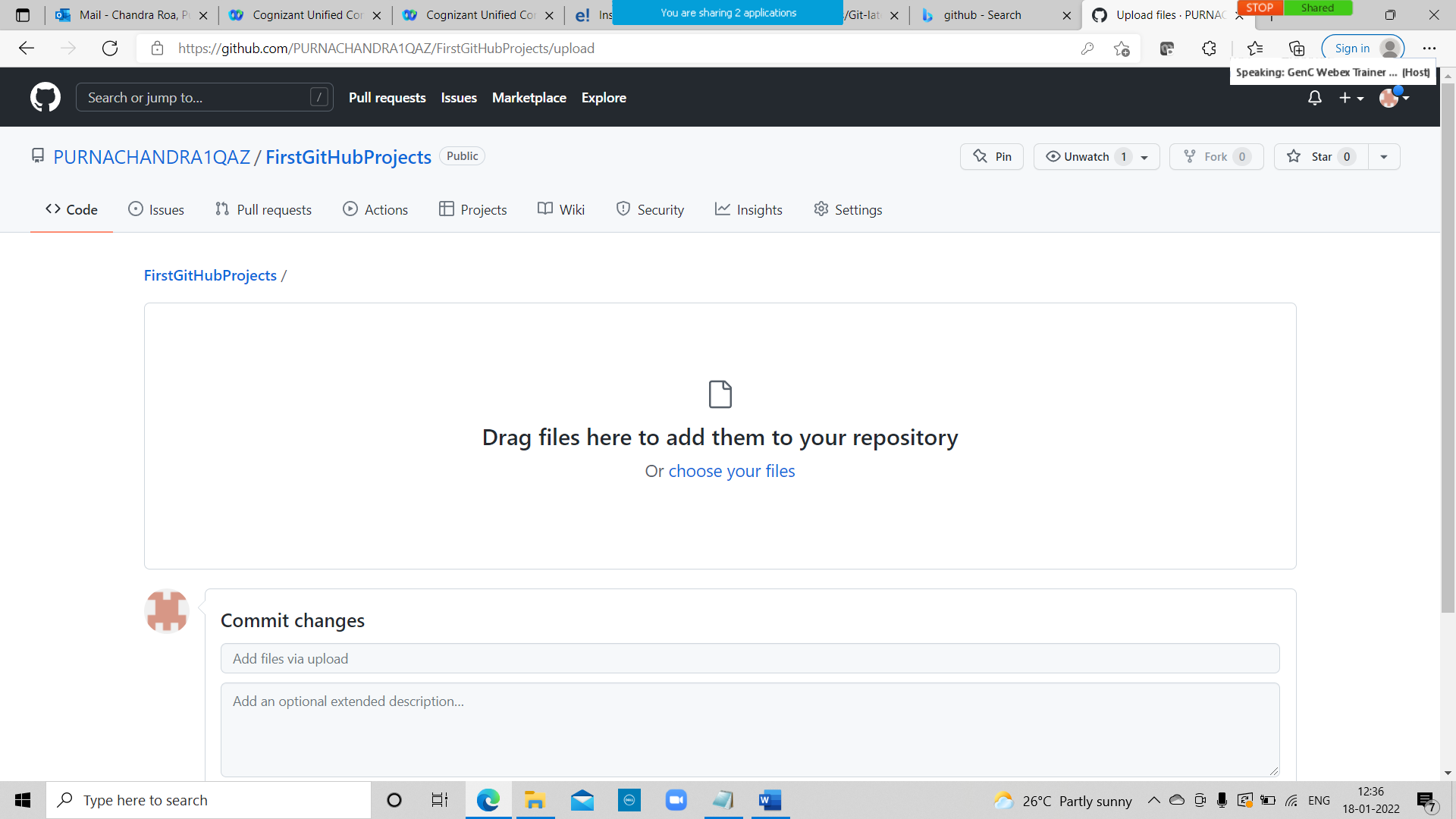


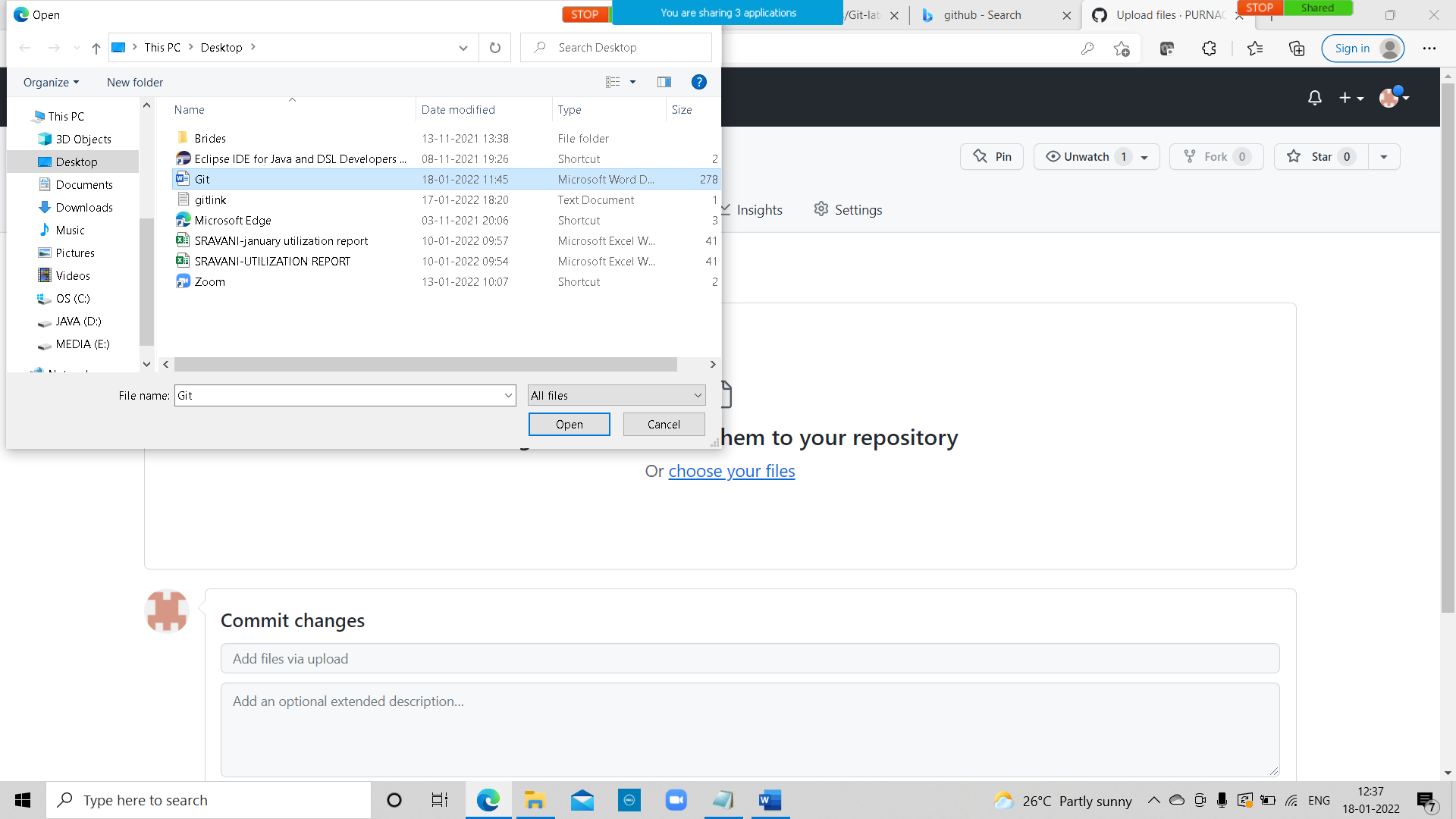


Click on create repository.

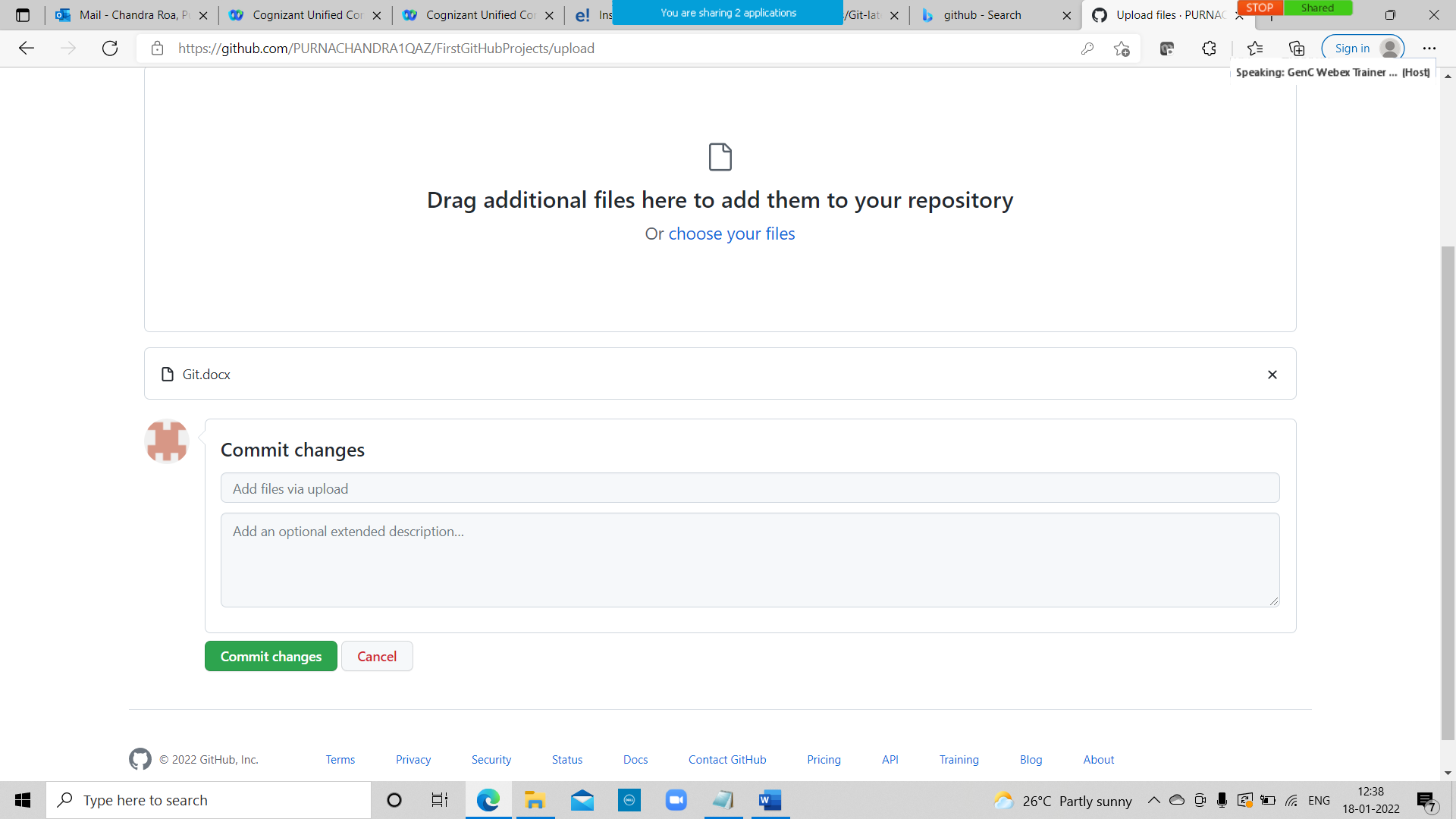


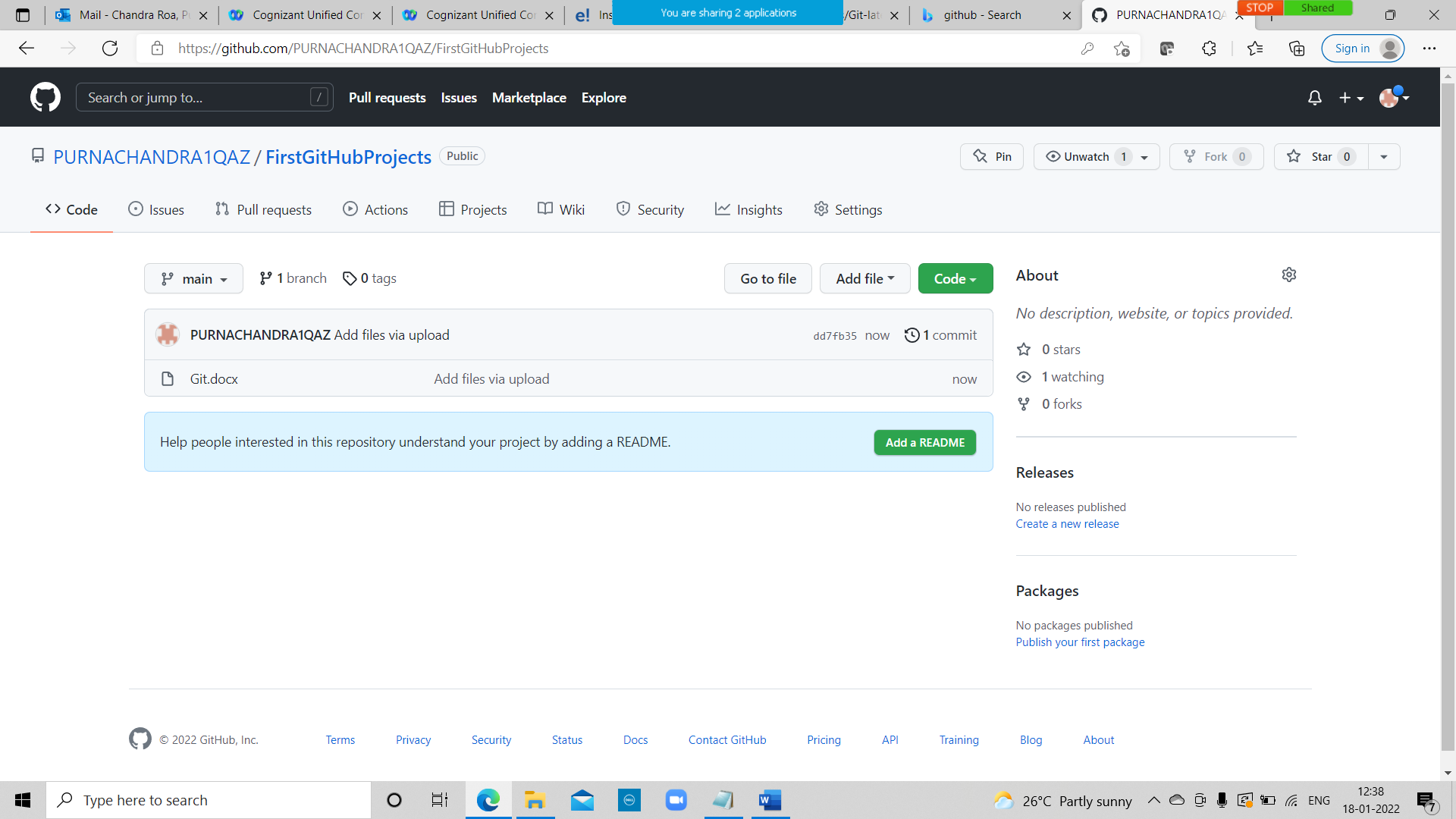
Click on upload an exisiting file





Click on commit changes





Git:

**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 co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

What is Git:

What is GitHub:

Difference between git and github:

A sample Work Flow:

Git: Version Control System

Track changes in files/folders

We can collaborate between Teams

|  |
| --- |
| Version Control  Version3  Version 2  Version1 |
| Version 3  Version2  Version1 |

Jithul (File)

Ankit (File)

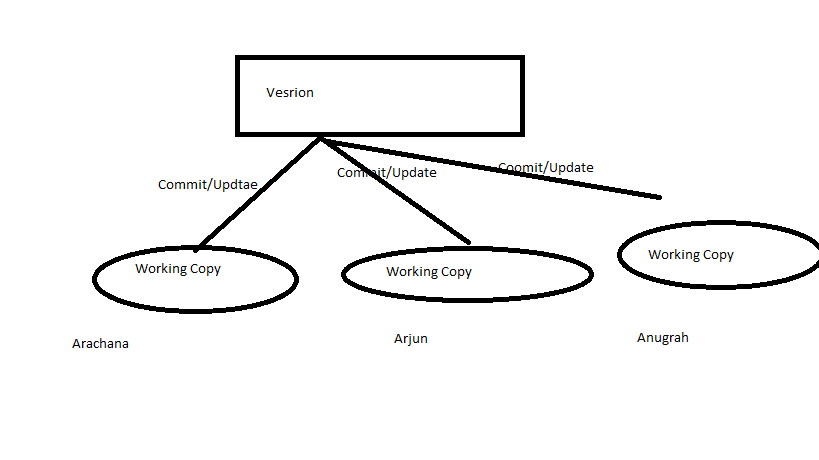
|  |
| --- |
| Checkout  File |

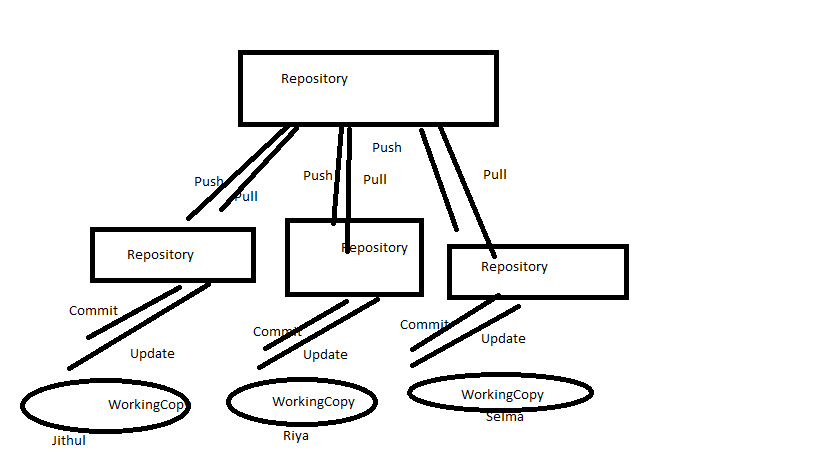
We have Two Types of Version Control Systems:

1)Centralized Version Control System

2)Distributed Version Control System

Centralized Version Control System:

2)Distributed Version Control System:



Git :Distributed Version Control System.

That is the why Git is famous and popular and it is also free and open source.

Git:https://git-scm.com/

GitHub: It is a website to upload your repositories online.

Provides Backup

Provides visual interface to your repo

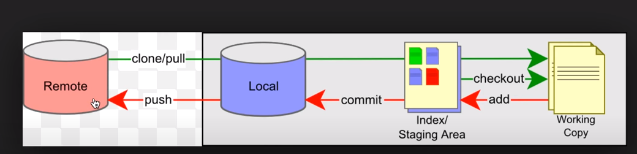
Makes collaboration easier

<https://github.com/>.

Git !=github

Git is a version Control System.

Git hub is a just website to put your repositories



This is a remote repository and this is a local repository Let us say our repository is put on a server its placed on a server which is in this remote server

We can do a clone or we can do a pull to take all the repositories present here into our local system.

So here we can get all the repositories Let us say we have all these four repositories or differenr branches of repositories now we can take or checkout any branch we will create a working copy for us and now we can do all our work on this working copy

And we can be offline during these changes So u can do all the changes and updates here,

Once we are done with our changes we can do our add command and what we will add command to is it will put all ur changes on a staging area it will still not commit it it will just put in staging area and then we have to executecommit commnd, which will make all our changes and committed to the local repository but at this point it is still not pushed it to the remote repository.

So we have to again do push command which will push our changes to the remote repositories

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

**There are two types of VCS:**

* Centralized Version Control System (CVCS)
* Distributed Version Control System (DVCS)

**Centralized VCS**

Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration. It works on a single repository to which users can directly access a central server.

Please refer to the diagram below to get a better idea of CVCS:



The repository in the above diagram indicates a central server that could be local or remote which is directly connected to each of the programmer’s workstation.

Every programmer can extract or **update** their workstations with the data present in the repository or can make changes to the data or **commit** in the repository. Every operation is performed directly on the repository.

Even though it seems pretty convenient to maintain a single repository, it has some major drawbacks. Some of them are:

* It is not locally available; meaning you always need to be connected to a network to perform any action.
* Since everything is centralized, in any case of the central server getting crashed or corrupted will result in losing the entire data of the project.

**Distributed VCS**

These systems do not necessarily rely on a central server to store all the versions of a project file.

In Distributed VCS, every contributor has a local copy or “clone” of the main repository i.e. everyone maintains a local repository of their own which contains all the files and metadata present in the main repository.

You will understand it better by referring to the diagram below:



As you can see in the above diagram, every programmer maintains a local repository on its own, which is actually the copy or clone of the central repository on their hard drive. They can commit and update their local repository without any interference.

They can update their local repositories with new data from the central server by an operation called “**pull**” and affect changes to the main repository by an operation called “**push**” from their local repository.

The act of cloning an entire repository into your workstation to get a local repository gives you the following advantages:

[[](https://www.edureka.co/devops-certification-training)](https://www.edureka.co/devops-certification-training" \t "_blank)

**[DevOps Certification Training Course](https://www.edureka.co/devops-certification-training" \t "_blank)**

[Explore Curriculum](https://www.edureka.co/devops-certification-training" \t "_blank)

* All operations (except push & pull) are very fast because the tool only needs to access the hard drive, not a remote server. Hence, you do not always need an internet connection.
* Committing new change-sets can be done locally without manipulating the data on the main repository. Once you have a group of change-sets ready, you can push them all at once.
* Since every contributor has a full copy of the project repository, they can share changes with one another if they want to get some feedback before affecting changes in the main repository.
* If the central server gets crashed at any point of time, the lost data can be easily recovered from any one of the contributor’s local repositories.

# Install Git – Git Installation On Windows And CentOS:

## ****Install Git On Windows****

**Step 1**:

To download the latest version of Git, click on the link below:

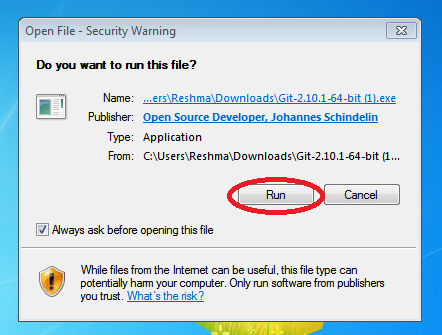
[***Download Git for Windows***](https://git-scm.com/download/win/)

[Git - Downloading Package (git-scm.com)](https://git-scm.com/download/win/)

Great! Your file is being downloaded.

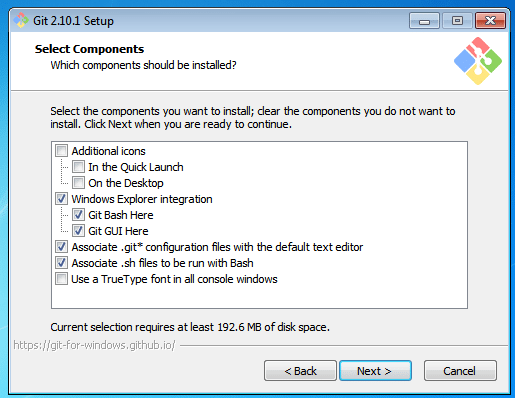
**Step 2:**

After your download is complete, **Run** the .exe file in your system.



**Step 3:**

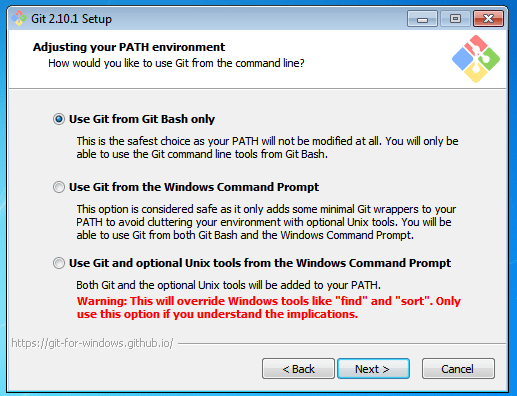
After you have pressed the **Run** button and agreed to the license, you will find a window prompt to select components to be installed.



After you have made selection of your desired components, click on **Next>**.

**Step 4:**

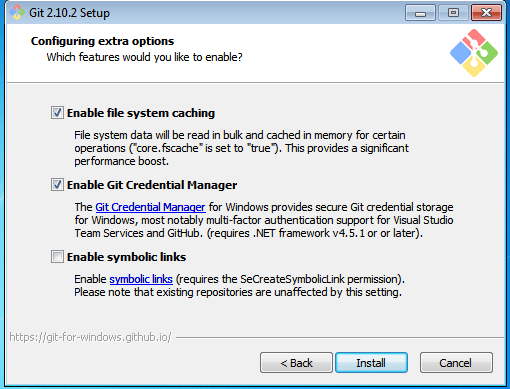
The next prompt window will let you choose the adjustment of your path environment. This is where you decide how do you want to use Git.



You can select any of the three options according to your needs. But for beginners, I recommend using **Use Git From Git Bash Only**

**Step 5:**

The next step is to choose features for your Git. You get three options and you can choose any of them, all of them or none of them as per your needs. Let me tell you what these features are:



The first is the option to **Enable file system caching**.

Caching is enabled through Cache manager, which operates continuously while Windows is running. File data in the system file cache is written to the disk at intervals determined by the operating system, and the memory previously used by that file data is freed.

The second option is to enable **Git Credential Manager**.

The **Git Credential Manager** for Windows (GCM) is a credential helper for Git. It securely stores your credentials in the Windows CM so that you only need to enter them once for each remote repository you access. All future Git commands will reuse the existing credentials.

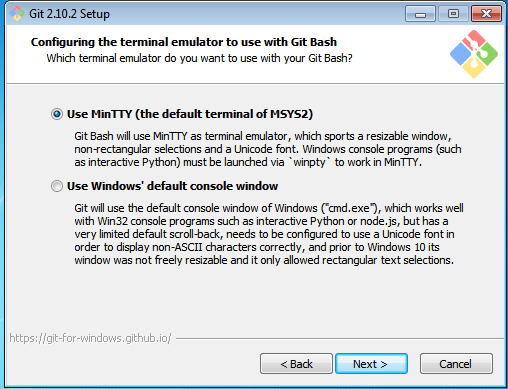
The third option is to **Enable symbolic links**.

Symbolic links or symlinks are nothing but advanced shortcuts. You can create symbolic links for each individual file or folder, and these will appear like they are stored in the folder with symbolic link.

I have selected the first two features only.

**Step 6:**

Choose your terminal.



You can choose one from the options.

The default terminal of MYSYS2 which is a collection of GNU utilities like bash, make, gawk and grep to allow building of applications and programs which depend on traditionally UNIX tools to be present.

Or you can choose the window’s default console window (cmd.exe).

**Step 7:**

Now you have got all you need. Select **Launch Git Bash** and click on **Finish**.

