

Practical 1

To perform Version Control using GIT

# Aim: To perform Version Control using GIT

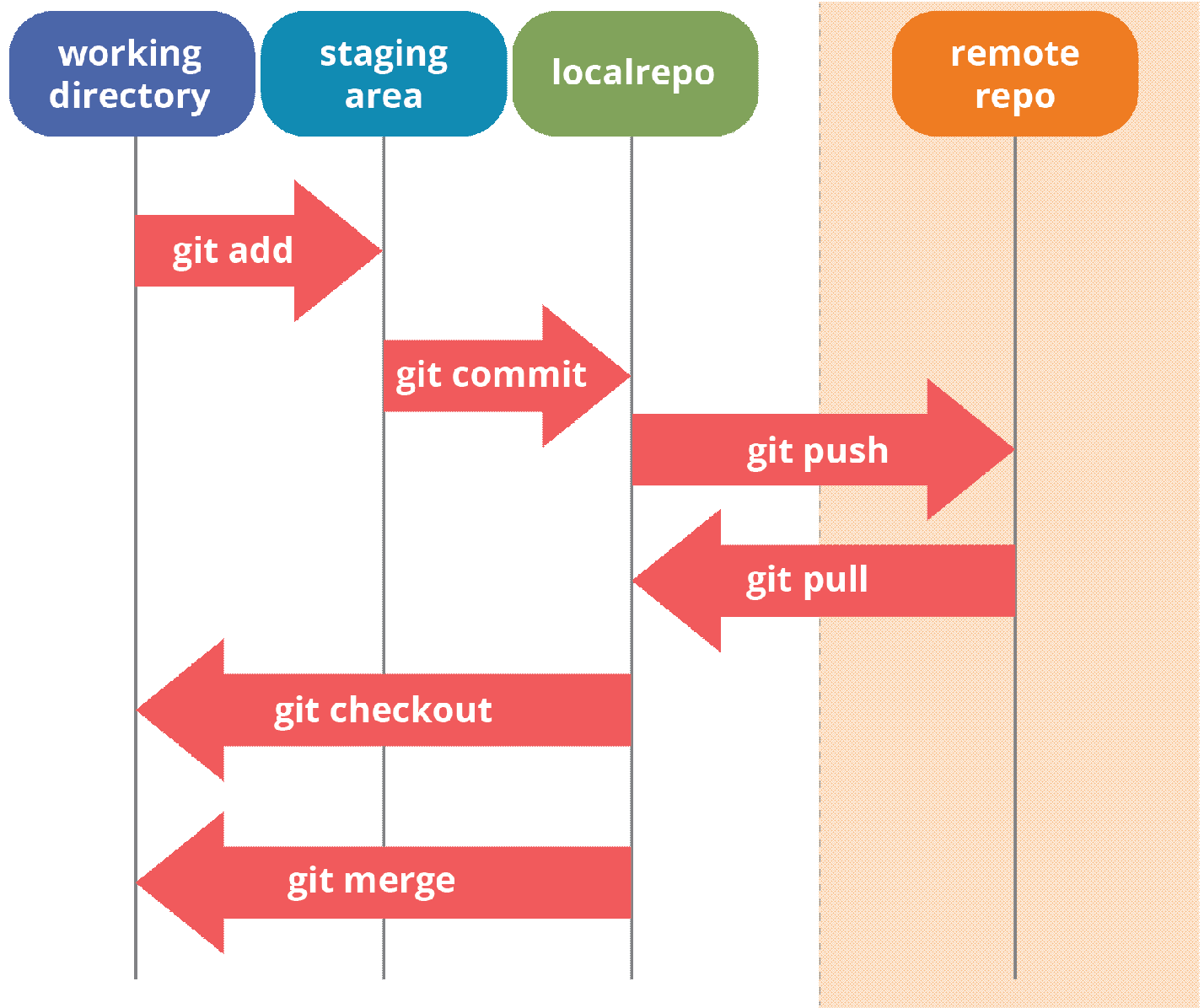
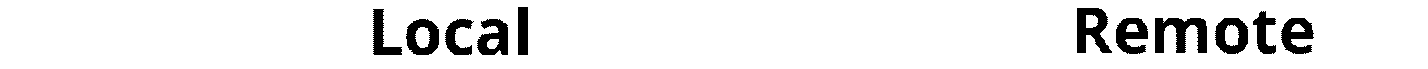
**Theory:**

* 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.
* Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.
* It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.
* Some of the basic operations in Git are:
  1. Initialize
  2. Add
  3. Commit
  4. Pull
  5. Push
* Some advanced Git operations are:

1. Branching
2. Merging
3. Rebashing

* The following diagram depict the all supported operations in GIT





**Installation of GIT**

* Installation of GIT can be done in both Windows and Ubuntu Operating Systems.
* The installation process is as follows:

# Windows Operating System:

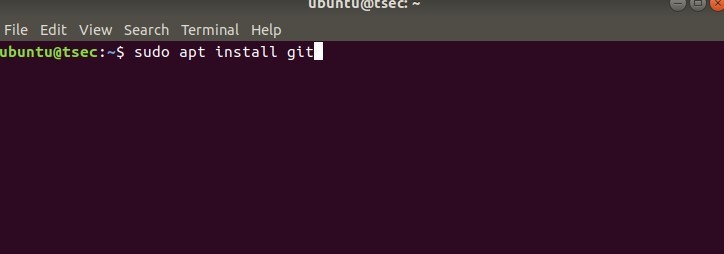
* Download GIT from <https://git-scm.com/> and perform straight forward installation.

# Ubuntu

* Install GIT using the following command

# $sudo apt install git



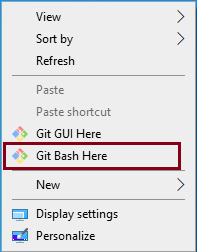


* Confirm the version after installation using command

# $git –version

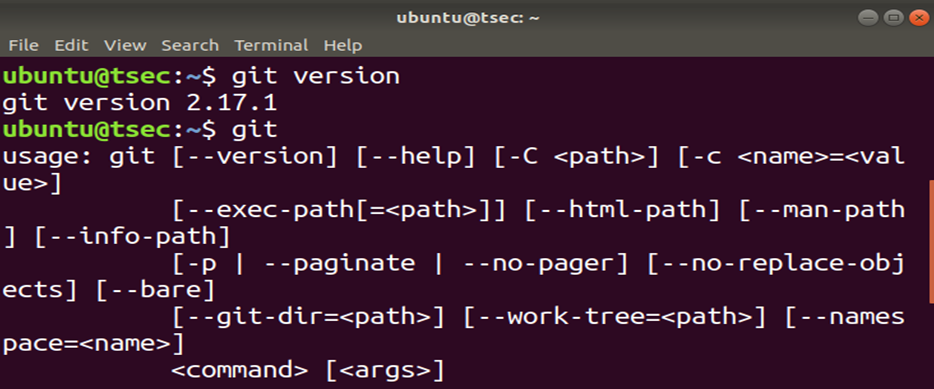


* Once installation is done, open the terminal in Ubuntu and perform the following steps or in windows Right click and select Git bash here.



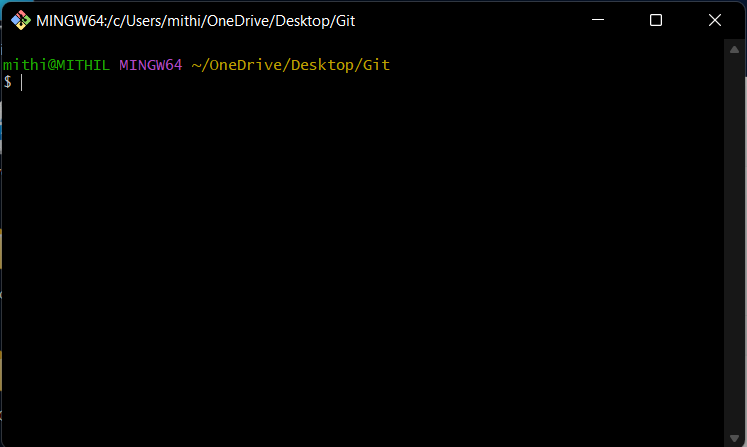
* The output of GIT Bash in windows and GIT shell in Ubuntu is shown below:





GIT Shell in Ubuntu

GIT Bash in Windows



* To perform version control, let us create a directory dvcs (Distributed version control system) and change directory to dvcs.

# $ mkdir git-dvcs

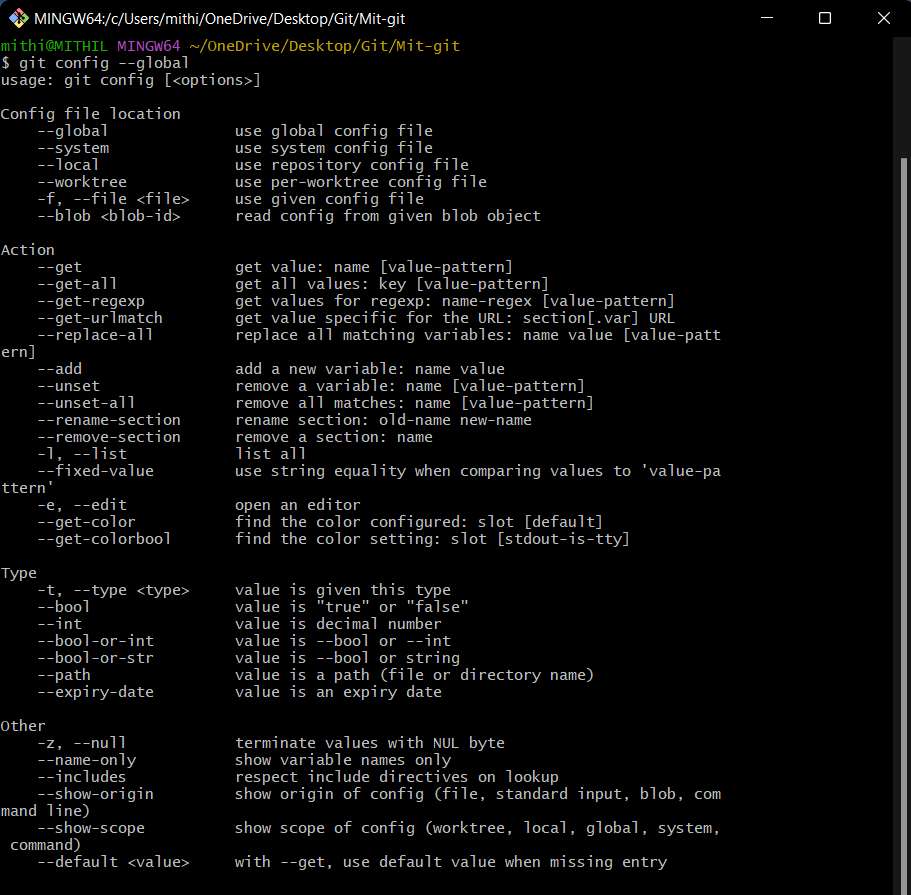
**$ cd git-dvcs/**



* Now check the user information using

# $ git config –global

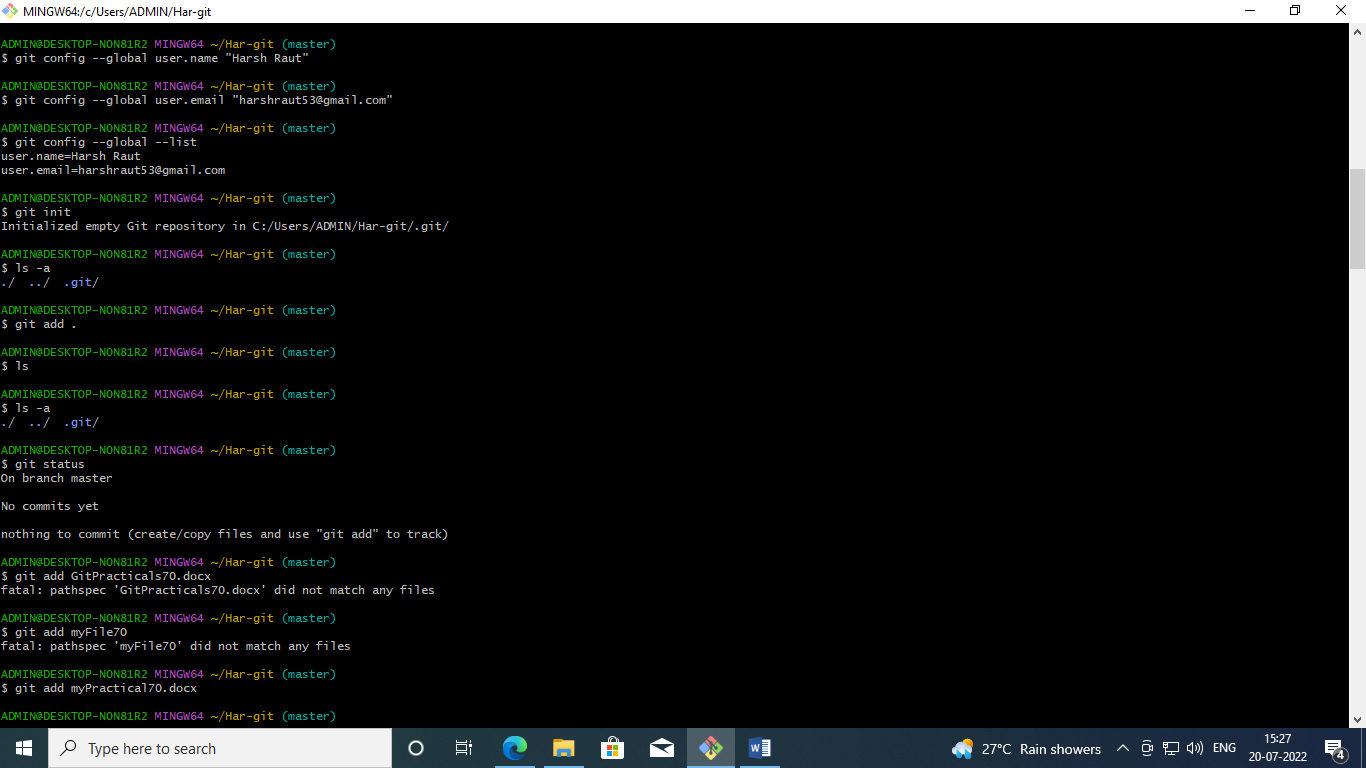




* As there are no users defined, let us define it using following two commands

# $ git config --global user.name "Mithil Ranpise"

**$ git config --global user.email "ranpisemithil@gmail.com”**

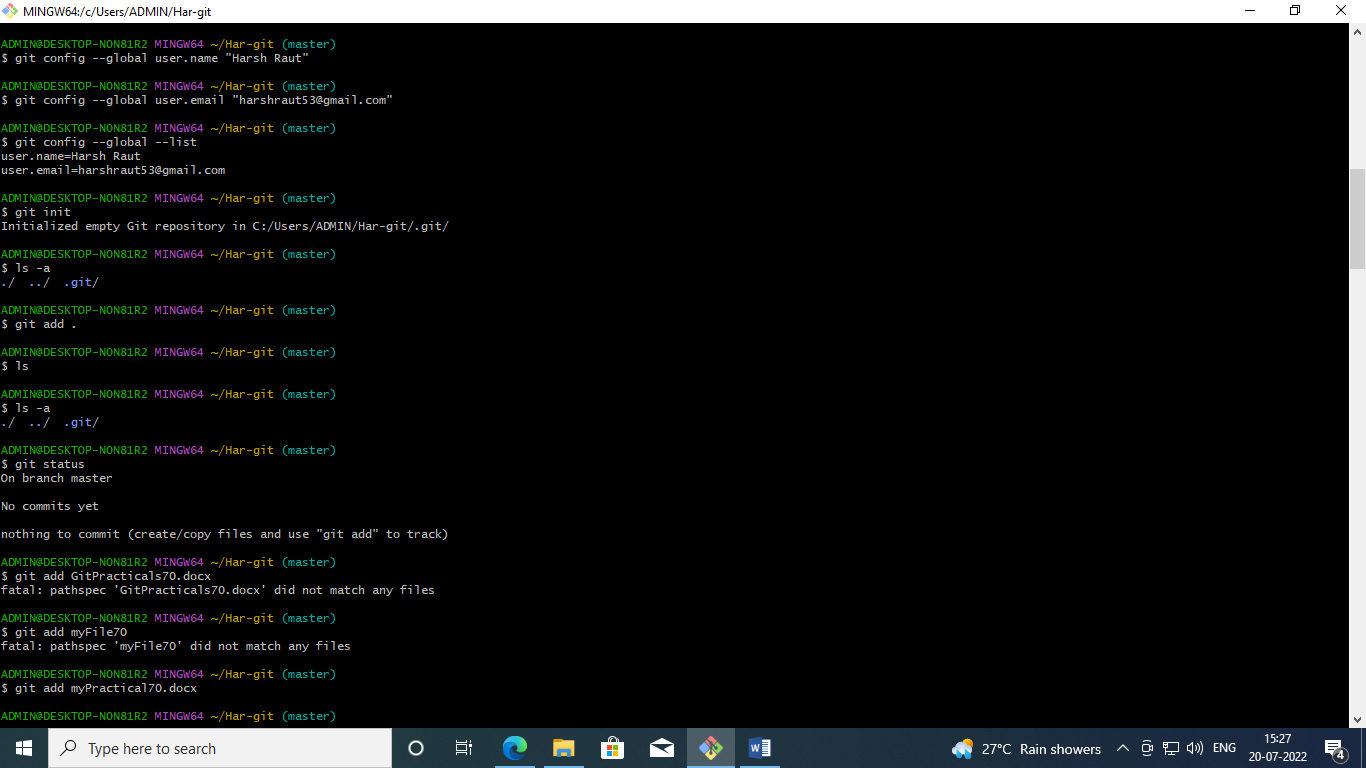




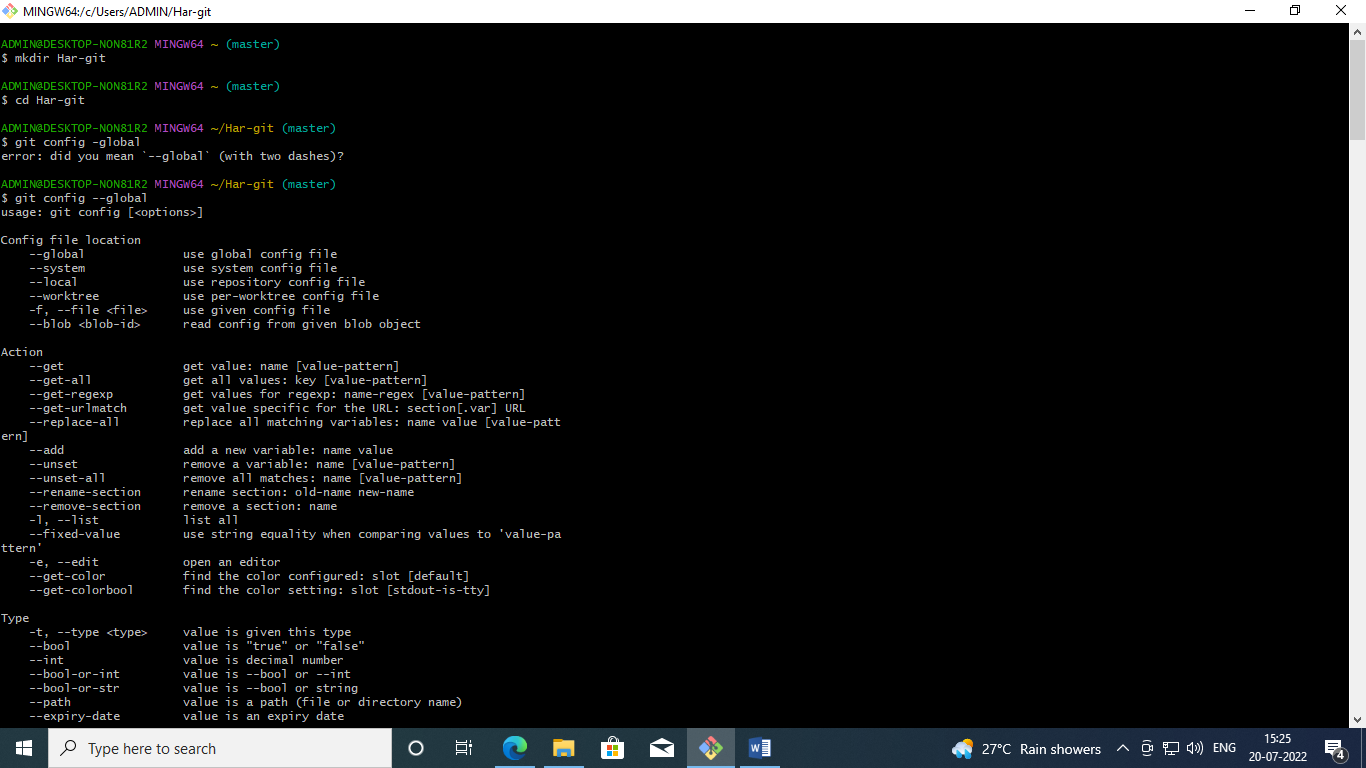
* Now, check the list of users

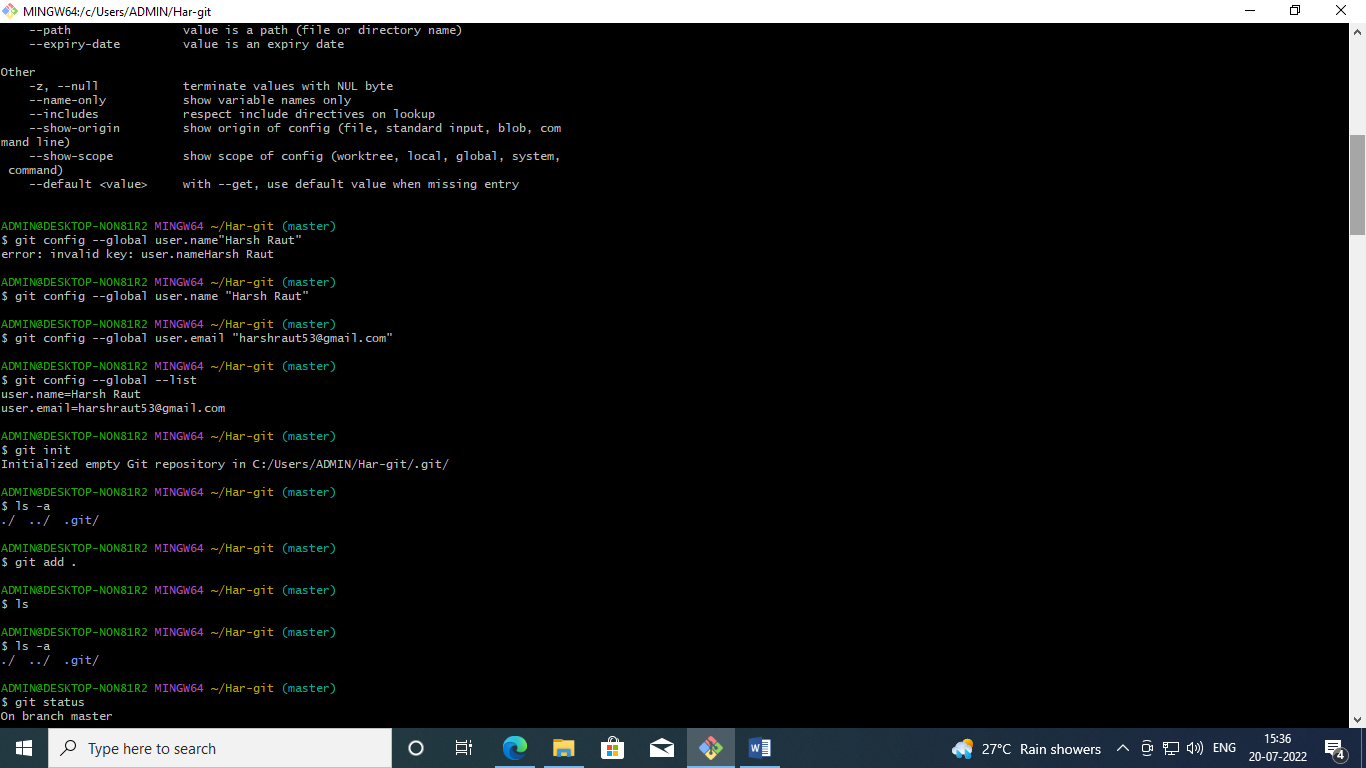
# $ git config --global –list user.name = Mithil Ranpise

**user.email = ranpisemithil@gmail.com**

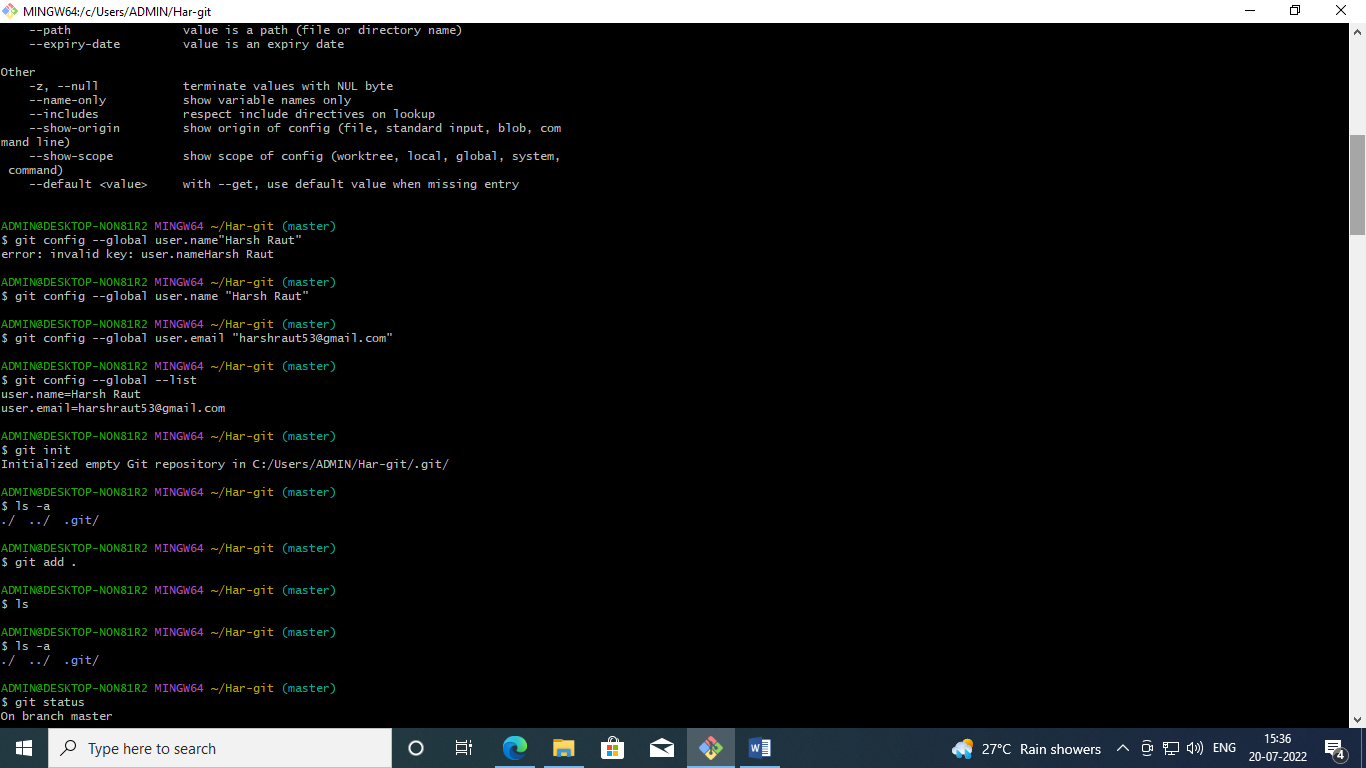


* Let us create a repository for version control named “Har-git”. The output of above command shown below which adds git hidden directory in current directory





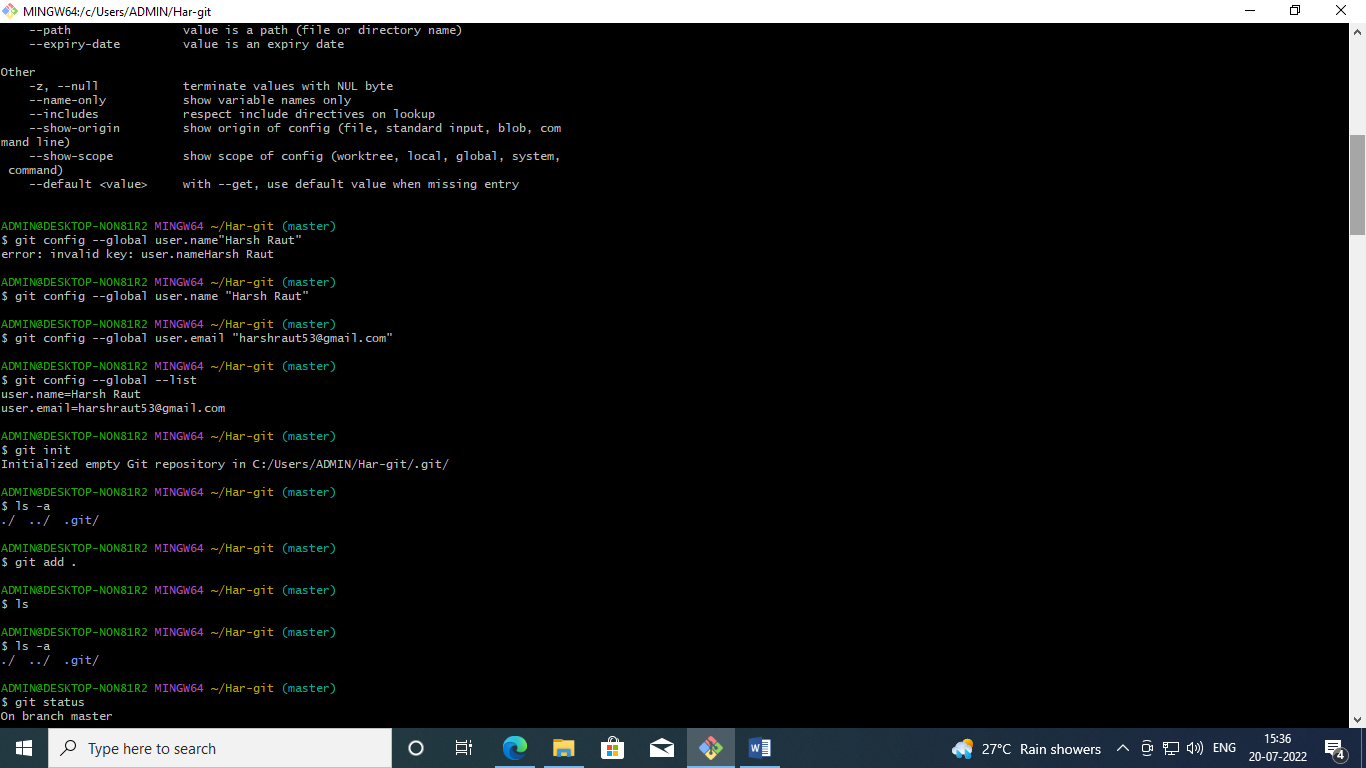
* If you have existing repository, then simply delete .git file and reinitialize it.





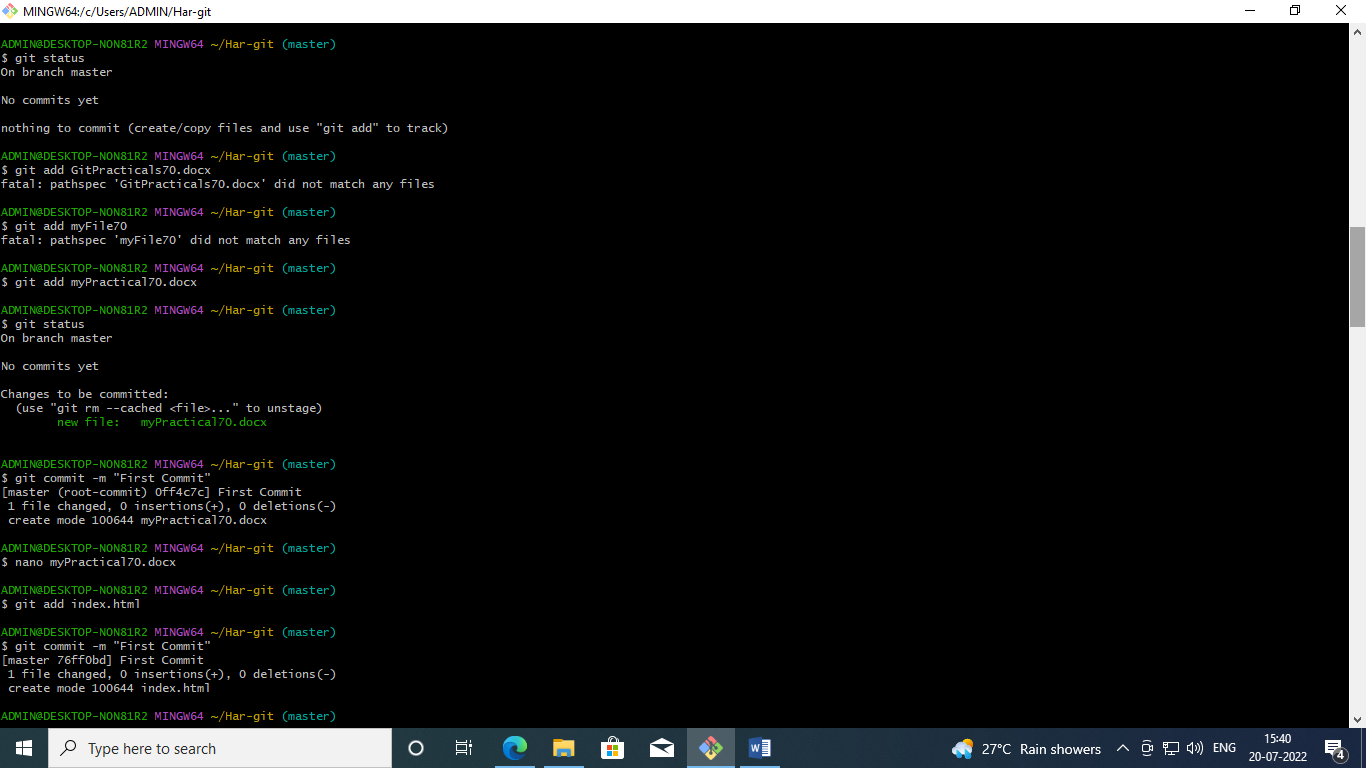
* Now, let us add some files inside our repository “git-demo-project”. To add files in the repository by create or copy some doc, html, image files inside current directory to see index and staging area.
* The add command is used along with dot (. Dot means current directory) for adding files in current repository i.e. making them in staging mode. They are untracked until we commit them.

**$ git add** (Index and staging area)



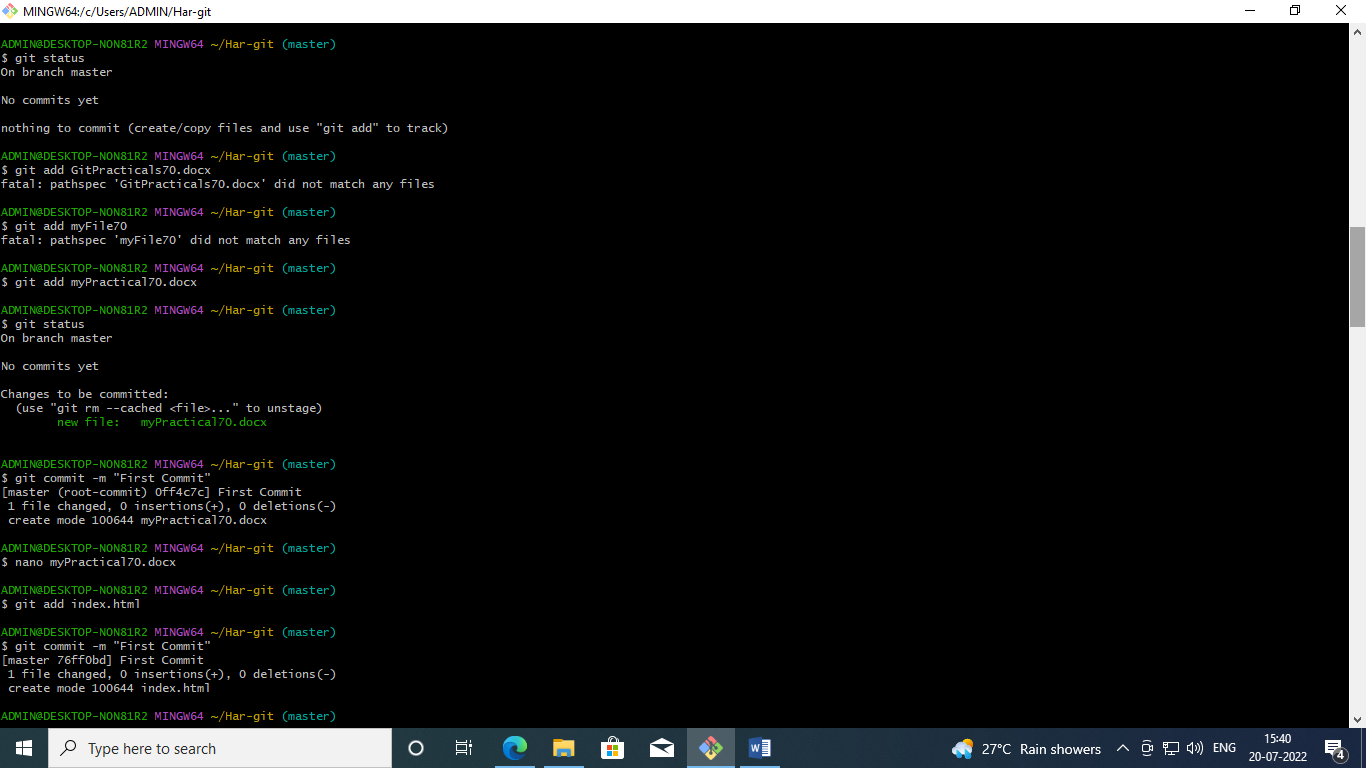
* To check the status of repository, use

**$ git status** (Which will show you some untrack files, so untracks files can be tracked using commit command.)



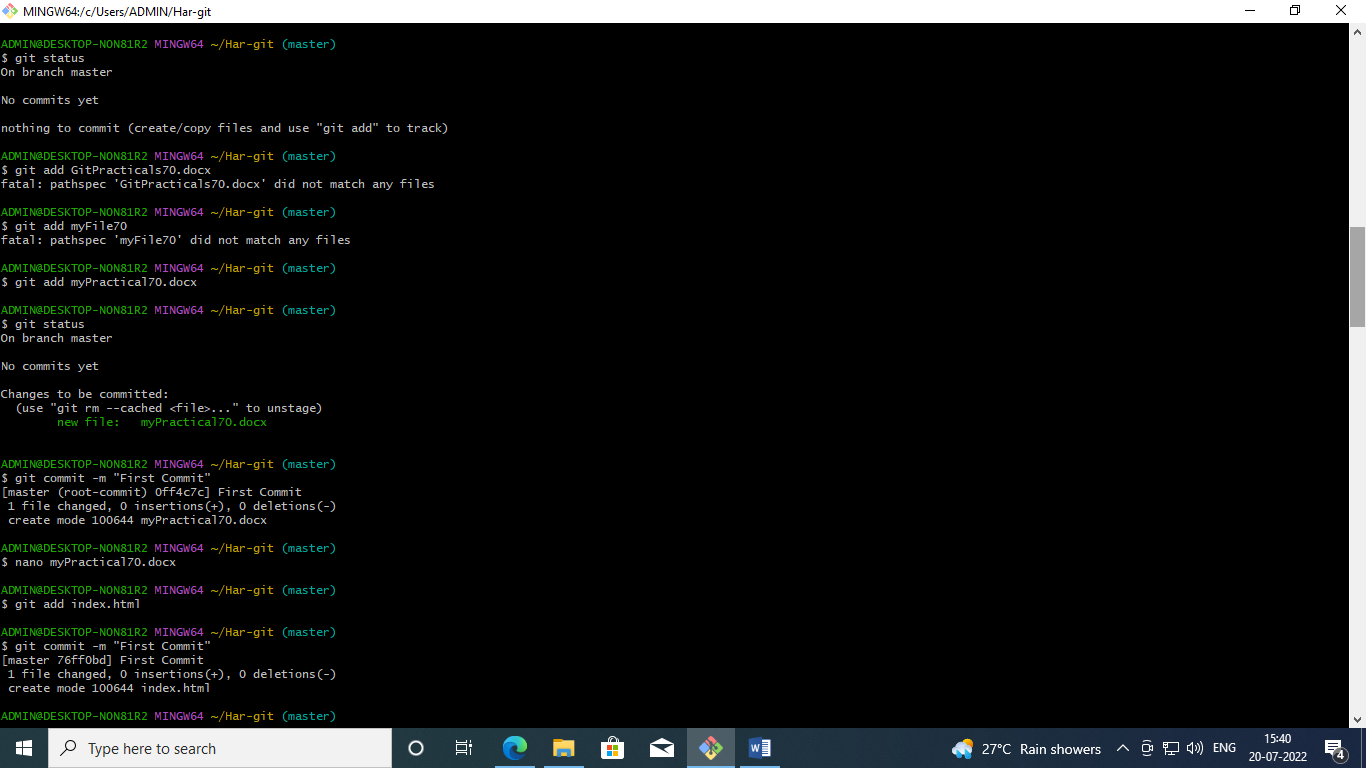


* After adding the files the status is as follows:

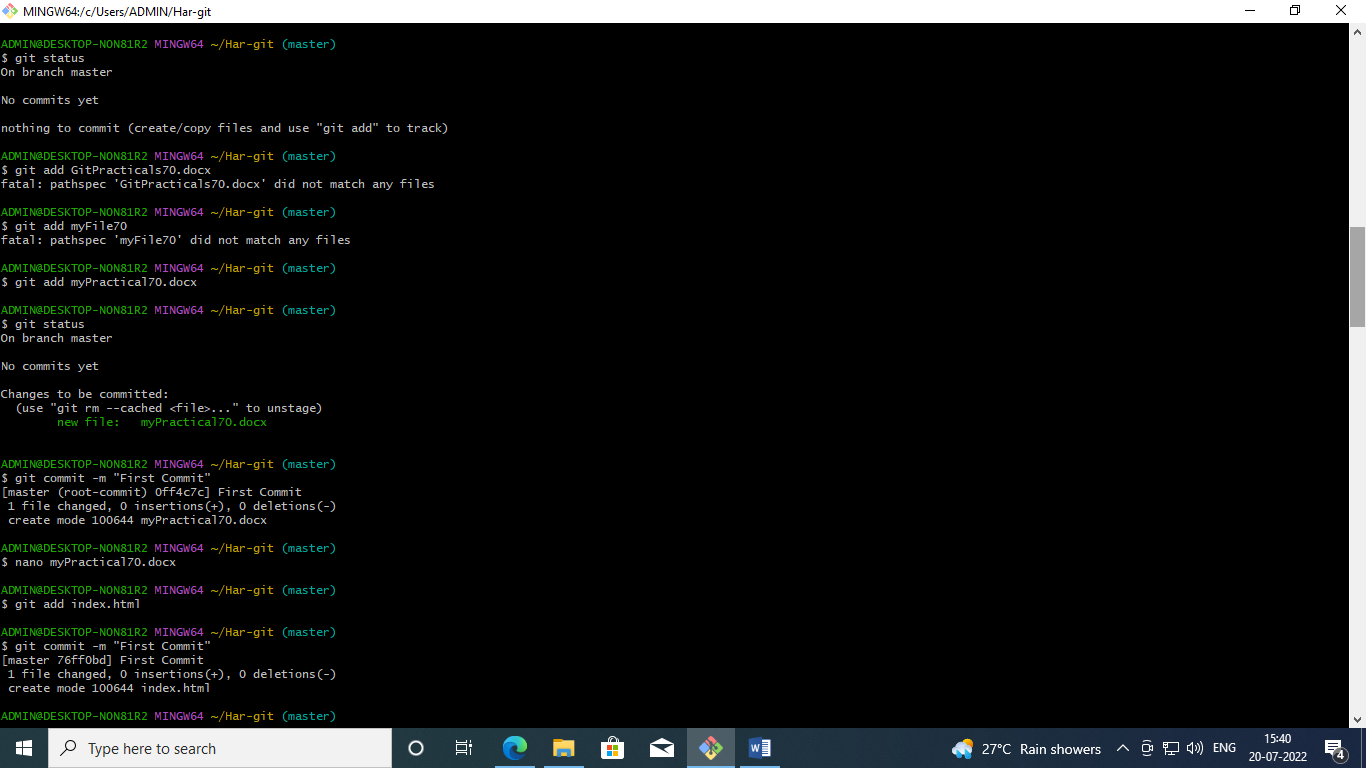


* Now, let us commit the changes

# $ git commit -m "First Commit" (#here -m for message)



* Add index.html in our directory

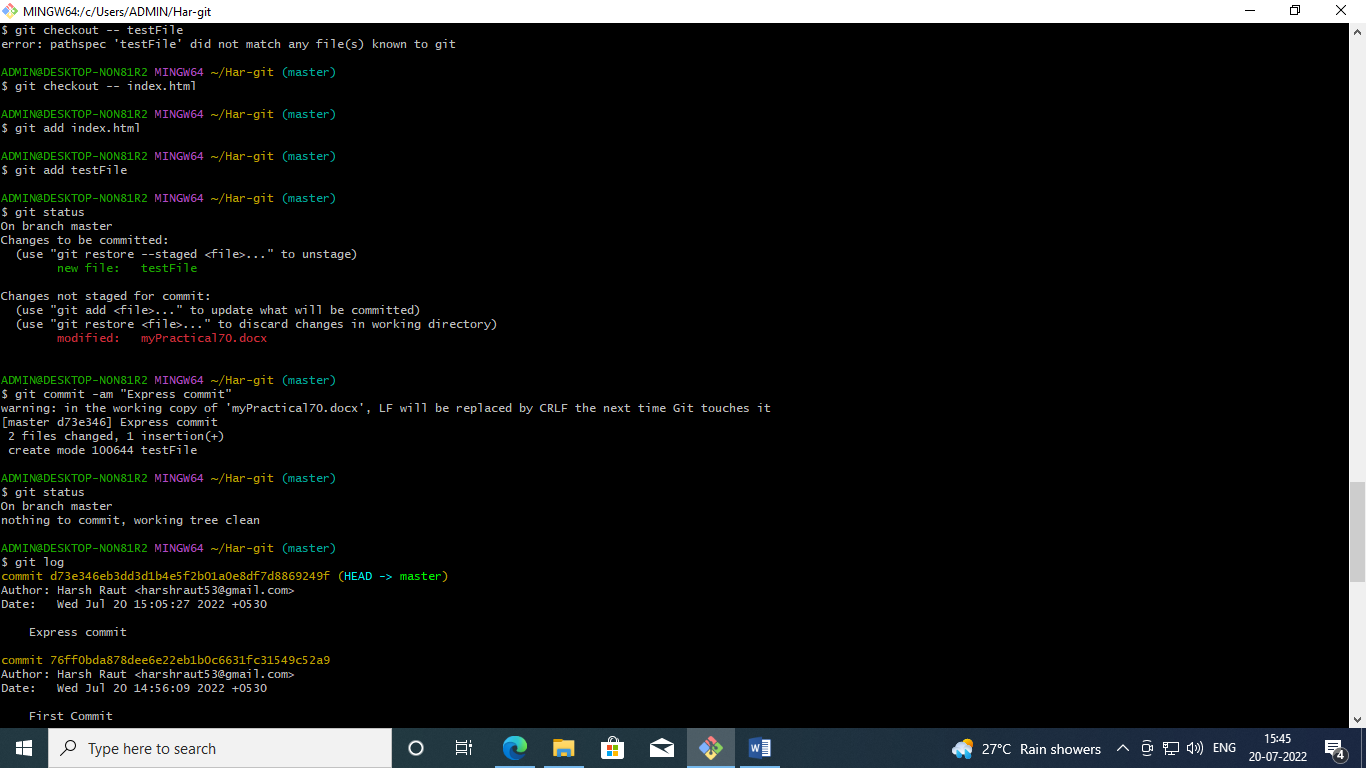


* To commit the index.html use the following commands.

# $ git add .

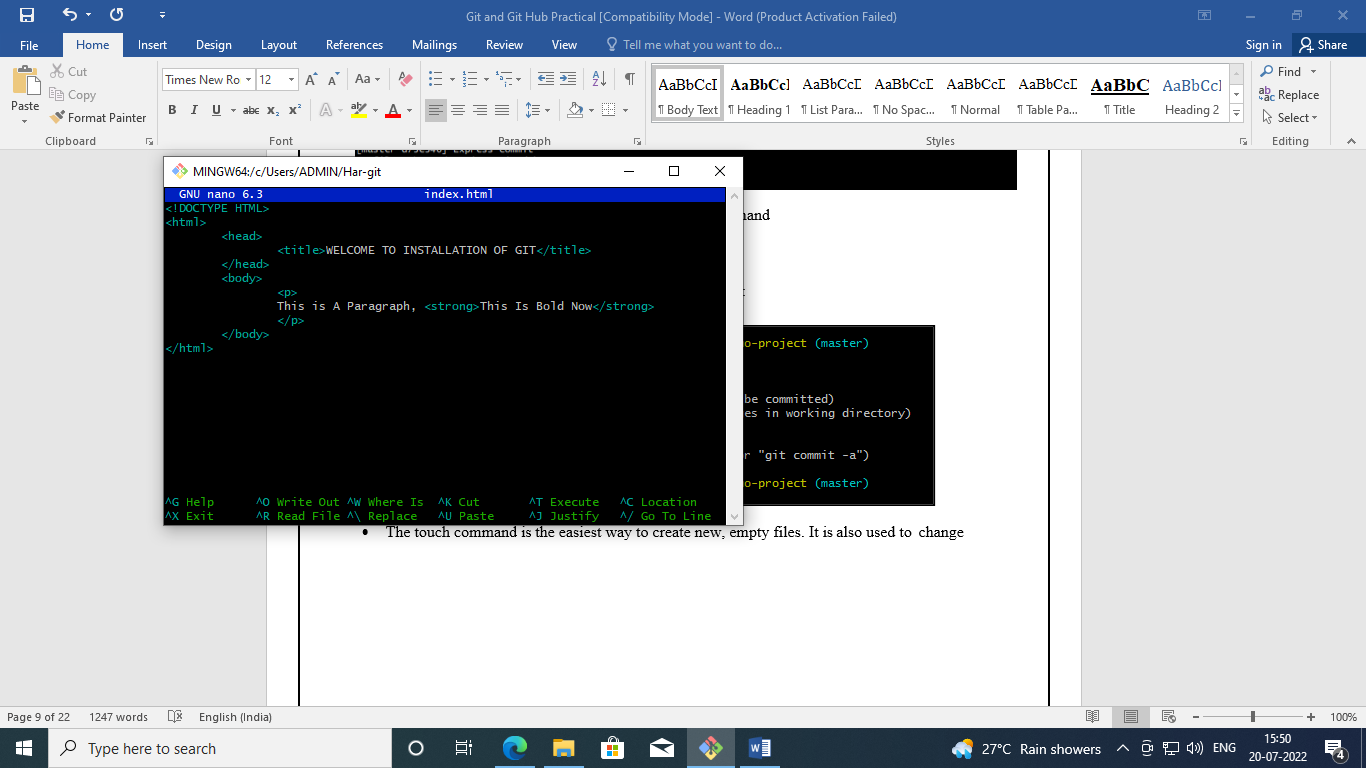
**$ git commit -am "express Commit" (#Here -a used for express commit)**



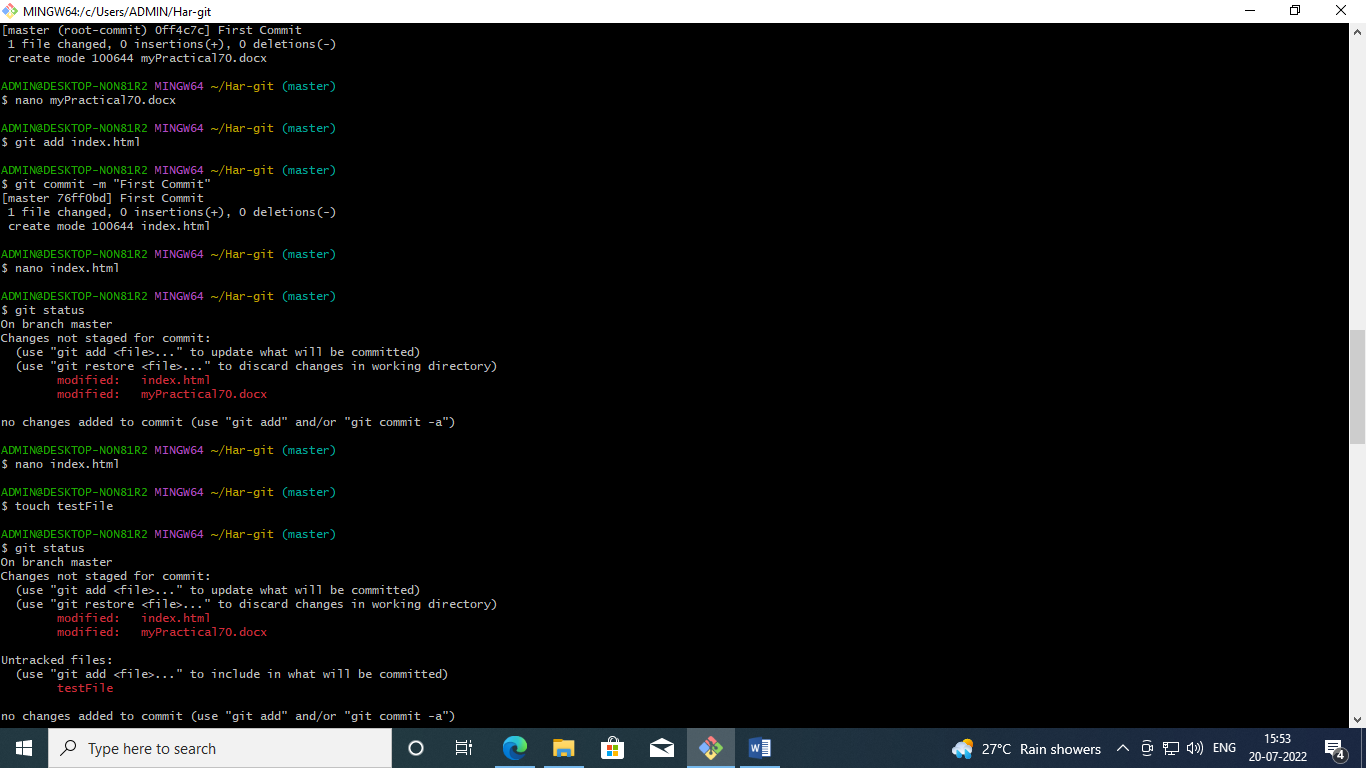


* To view or edit the index.html , use the following command

# $ nano index.html



* After editing the index.html , the git status is as follows:

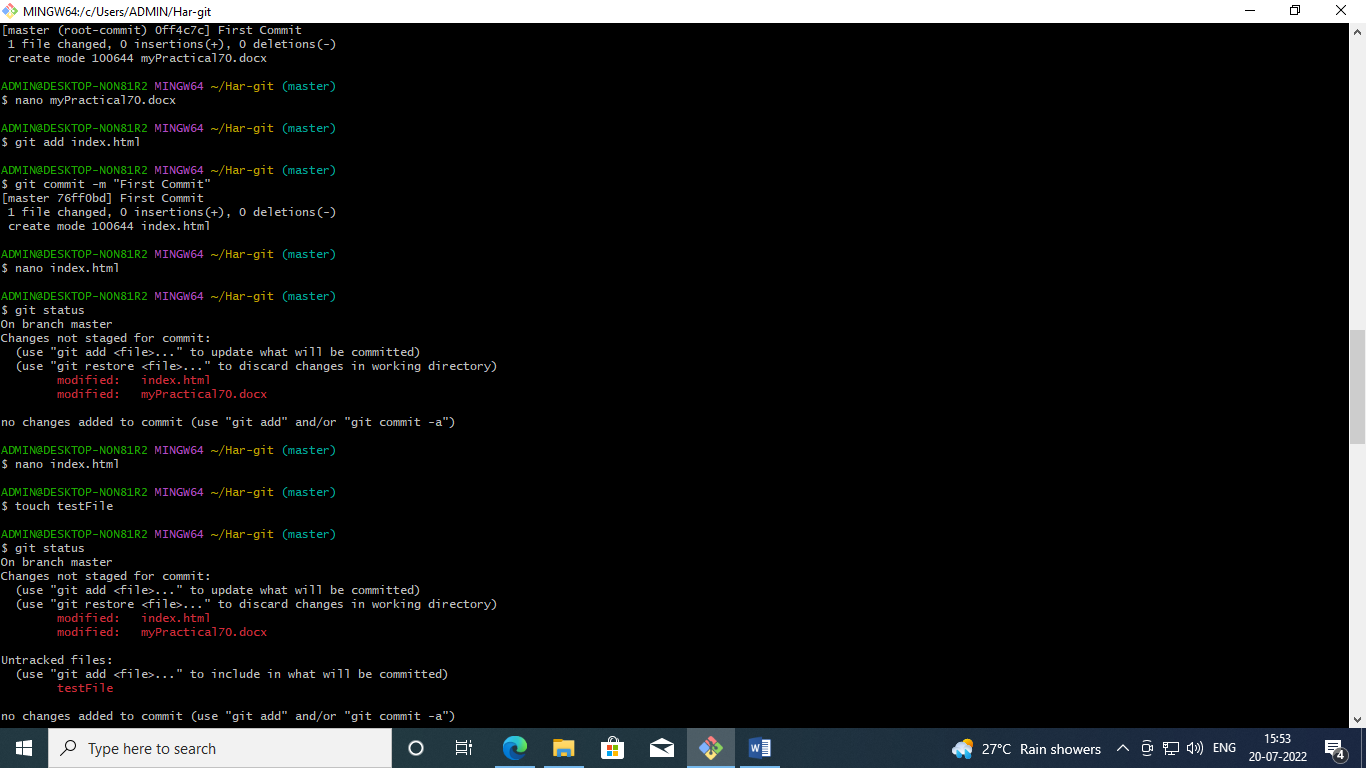


* The touch command is the easiest way to create new, empty files. It is also used to change

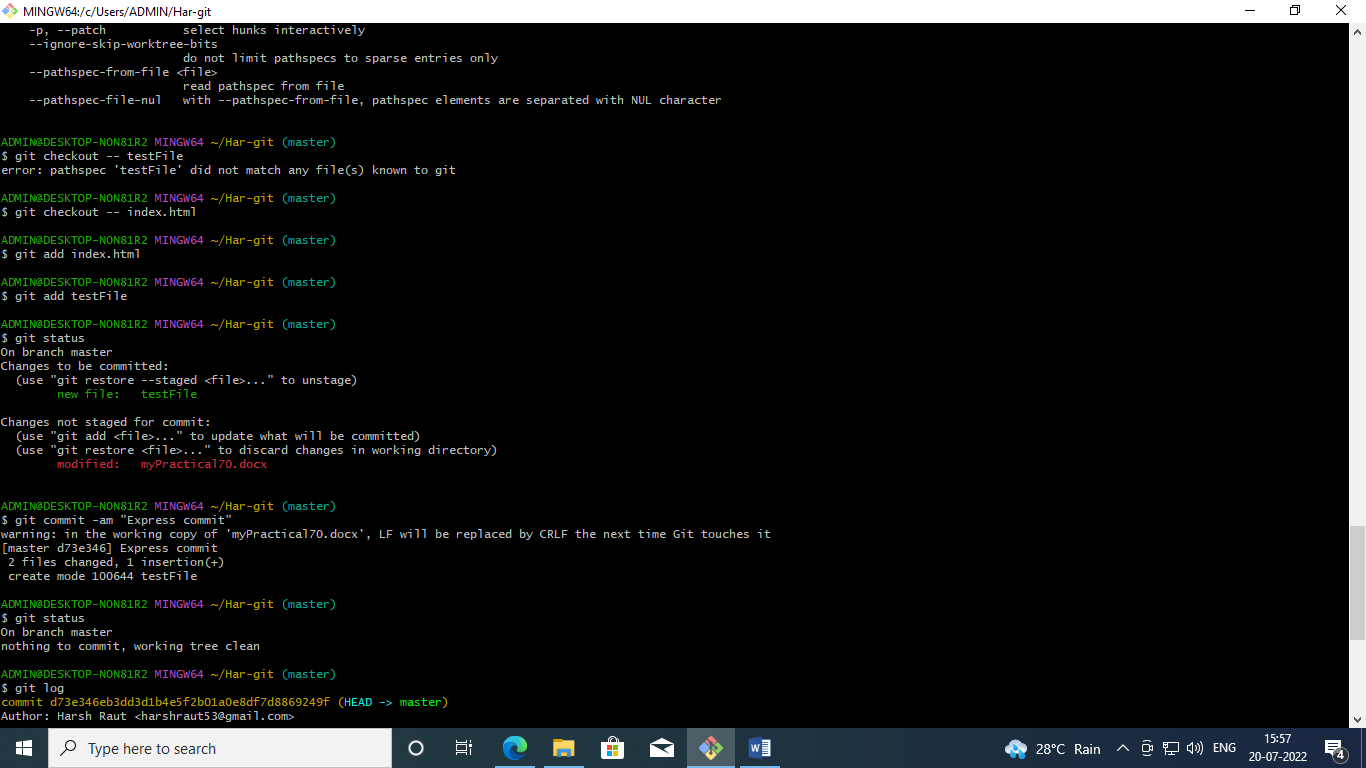


the timestamps (i.e., dates and times of the most recent access and modification) on existing files and directories.

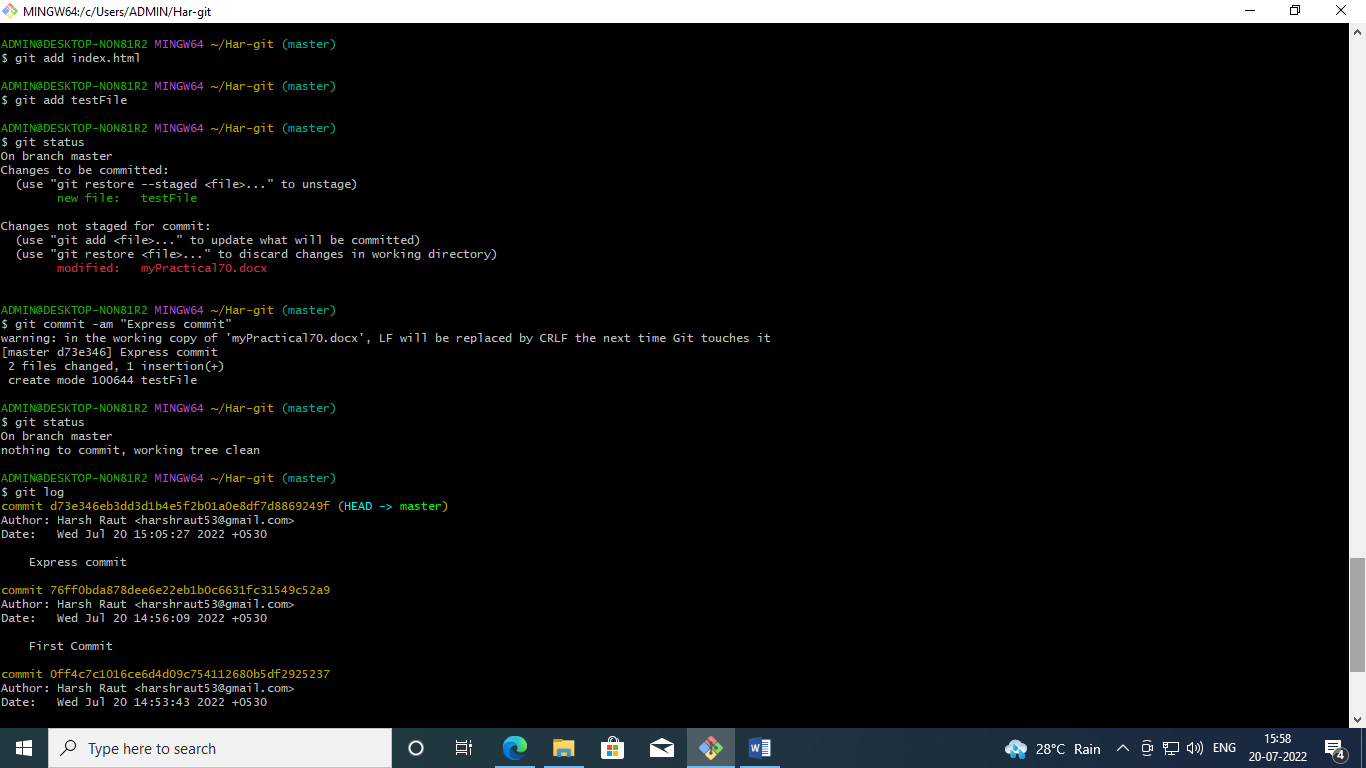
# $ touch testFile



* The git checkout command lets you navigate between the branches created by git branch. Checking out a branch updates the files in the working directory to match the version stored in that branch, and it tells Git to record all new commits on that branch.



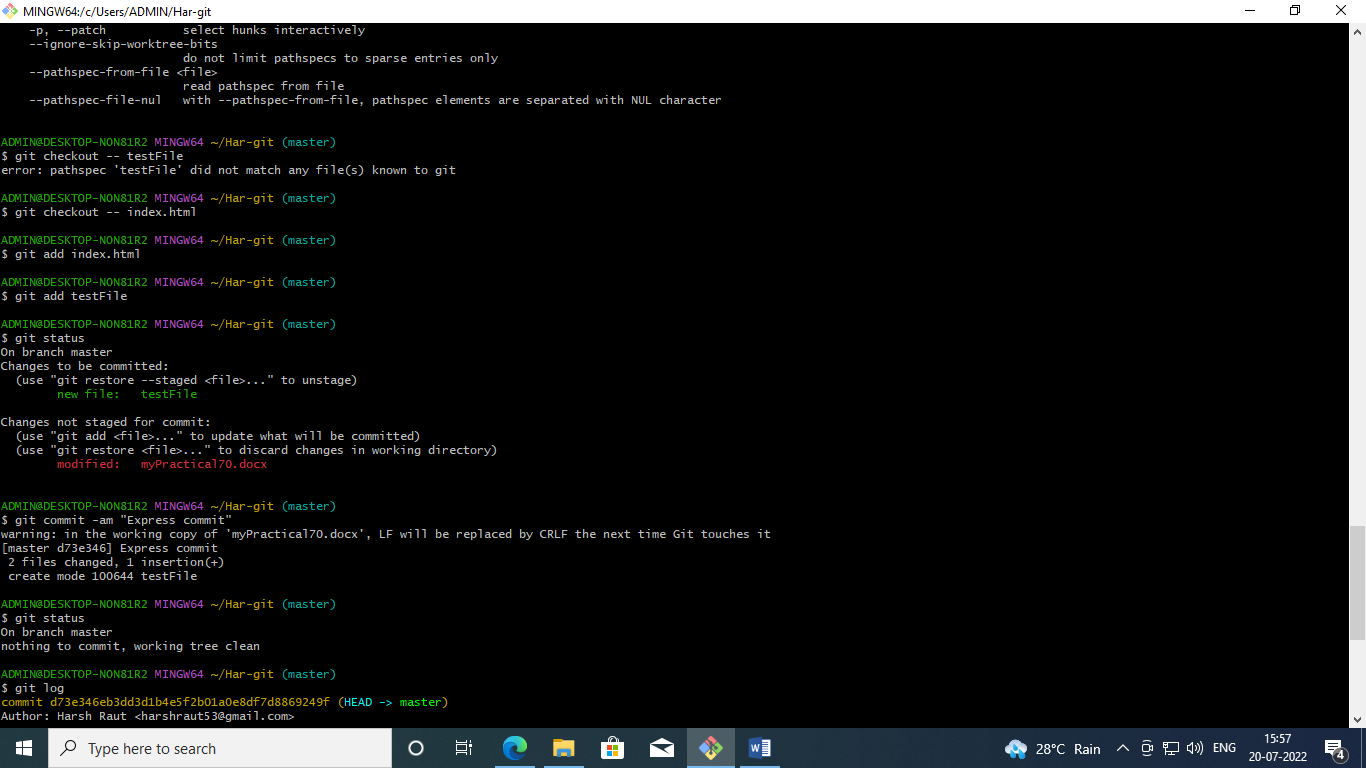
* Adding and committing the index.html and teststatus files



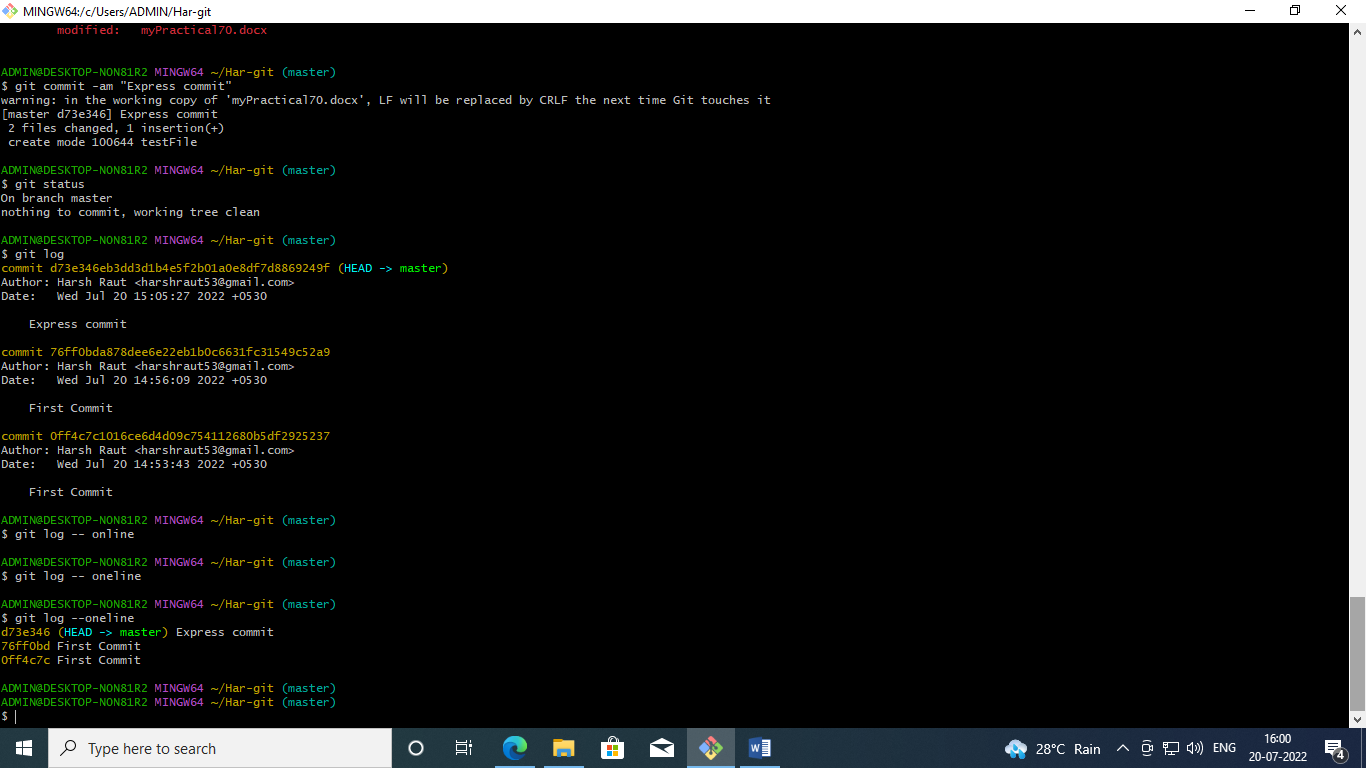


* Commiting the files

# $ git commit -am "Express commit"

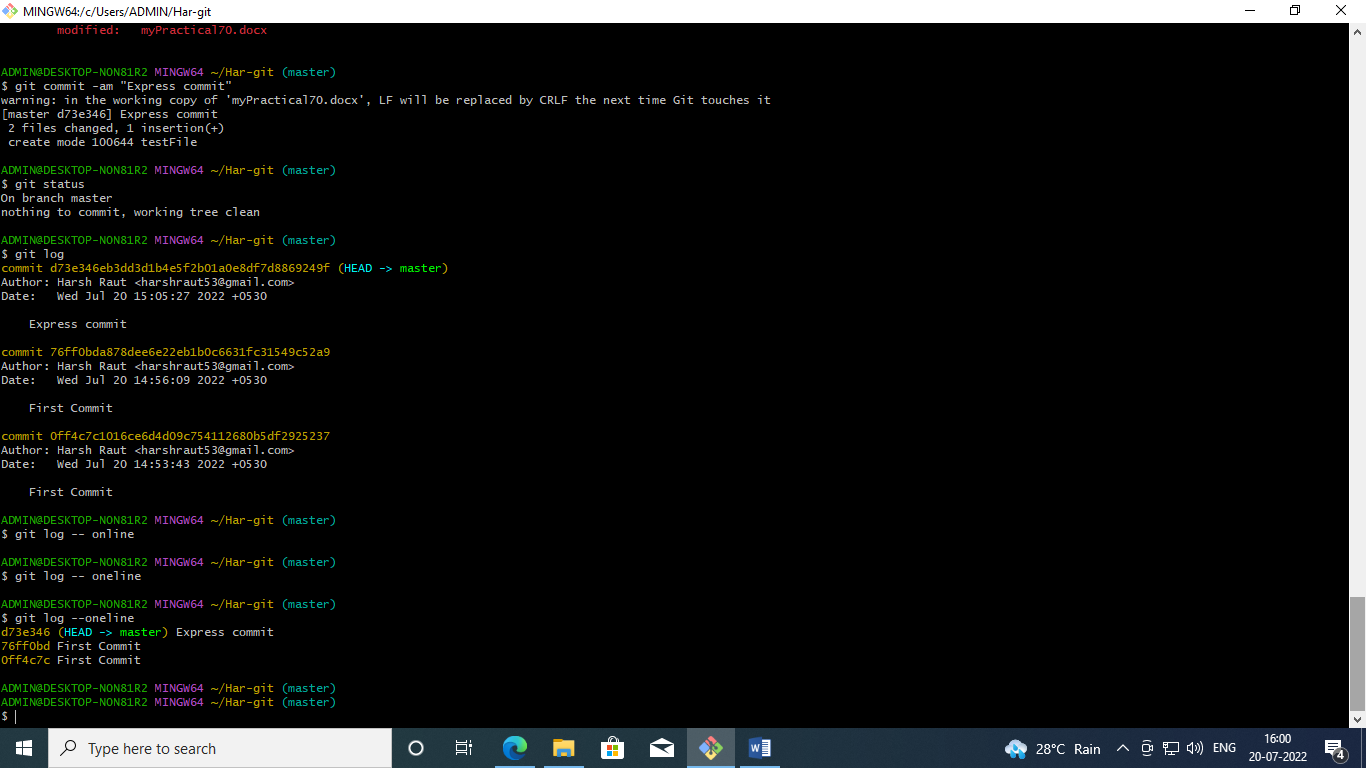
****

* To view all the commits , use the following command:





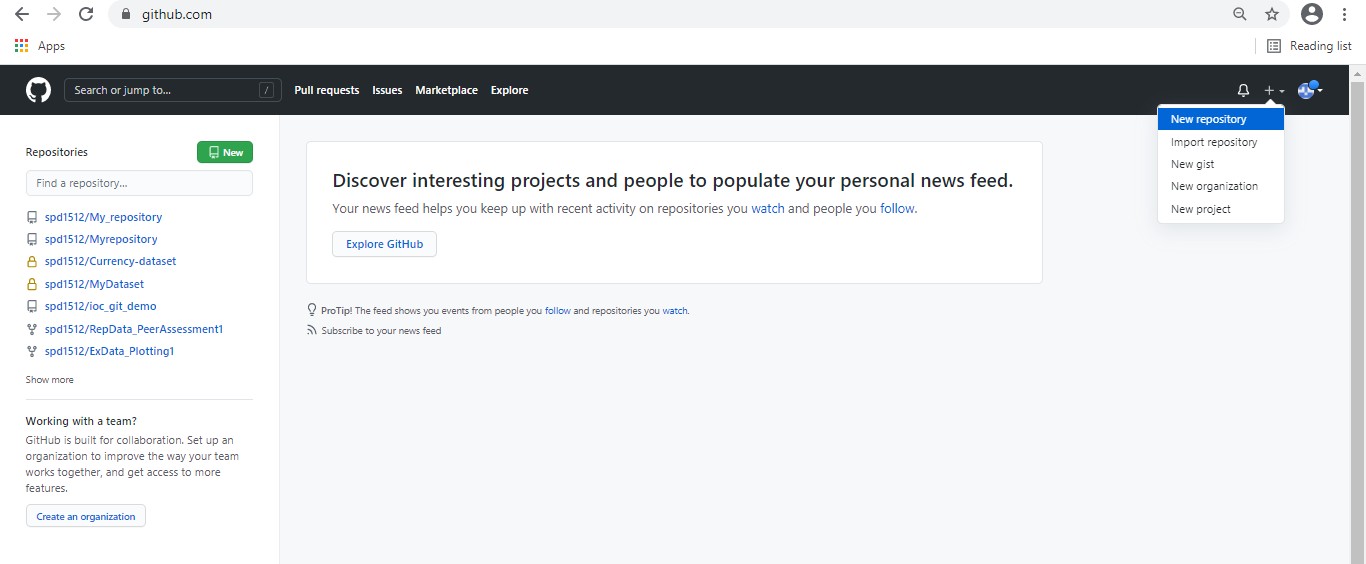
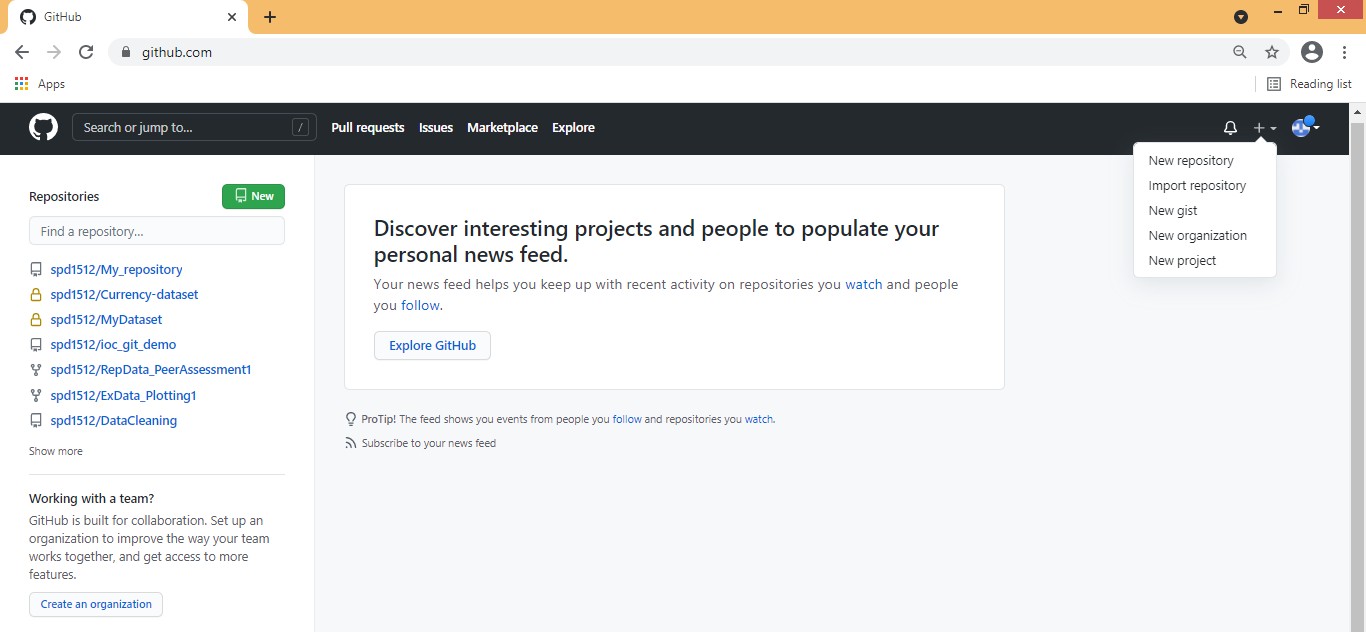
* To see all the operation in oneline use the –-oneline option in log command



* Open Github in desktop
* Log into your Github Account
* The following screen is displayed:

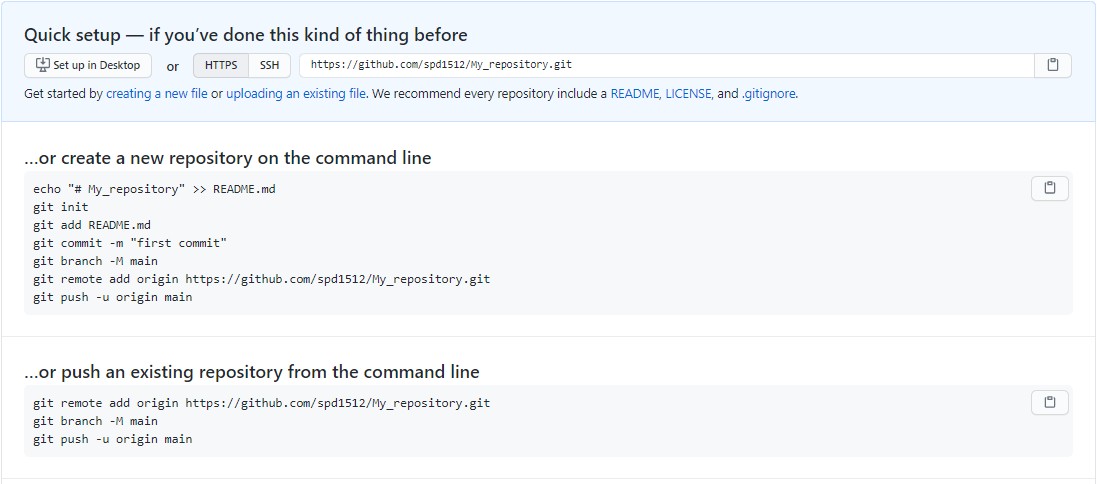
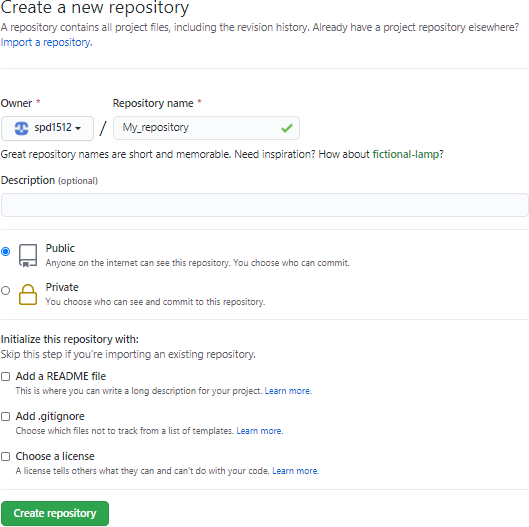


* Select the “ New Repository” option



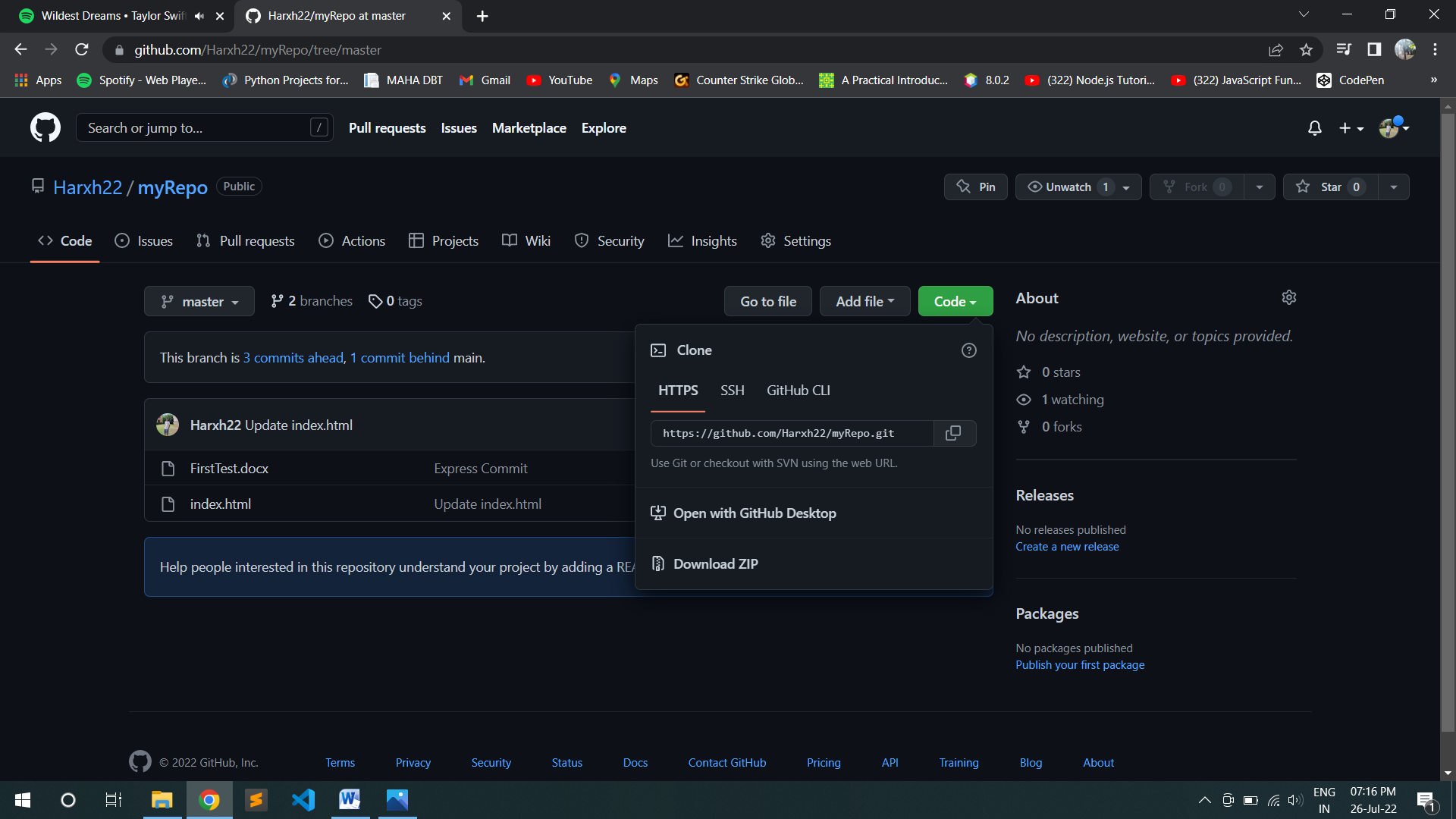
* Now Specify a Name to repository and select public option followed by create repository.





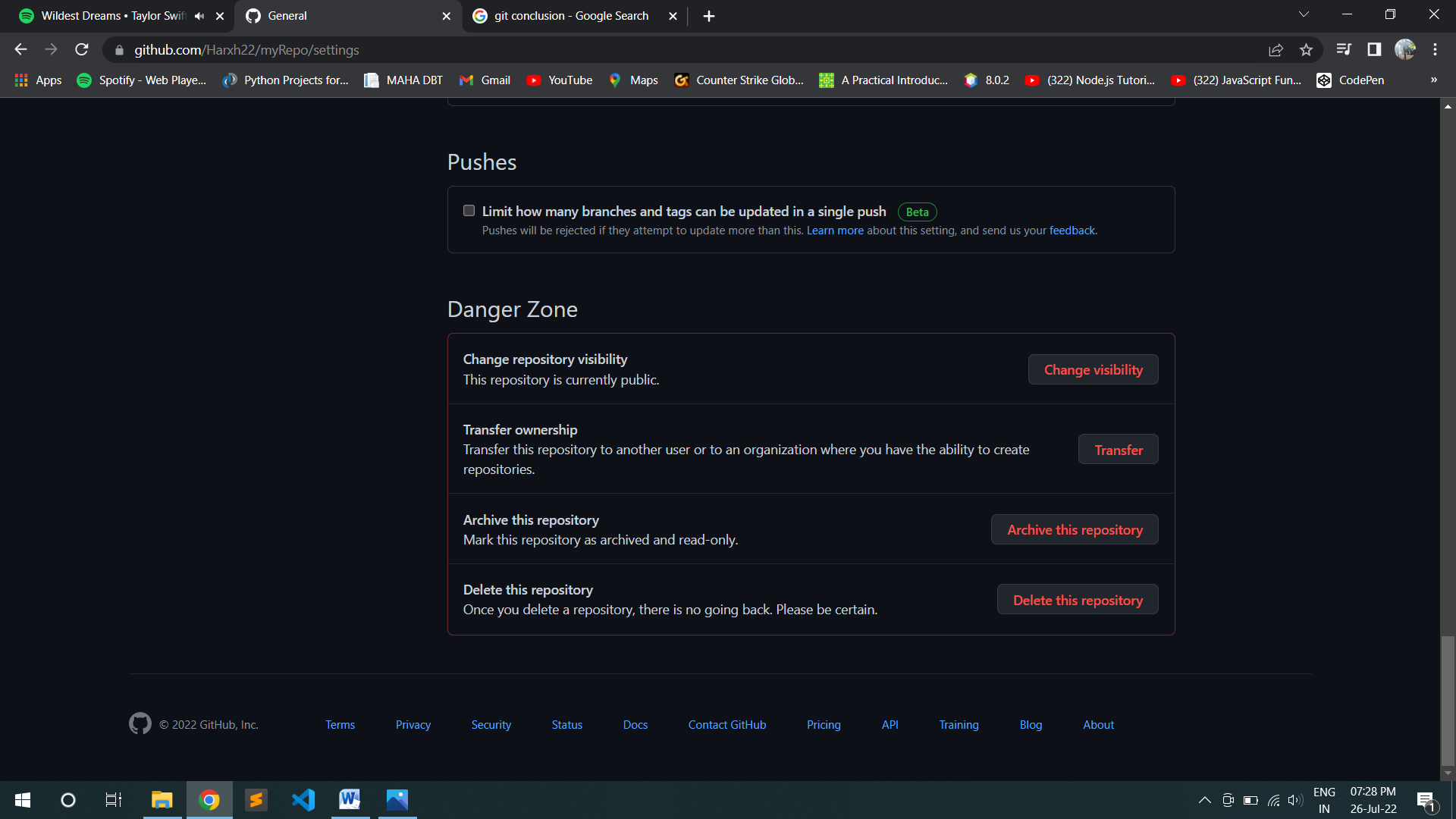


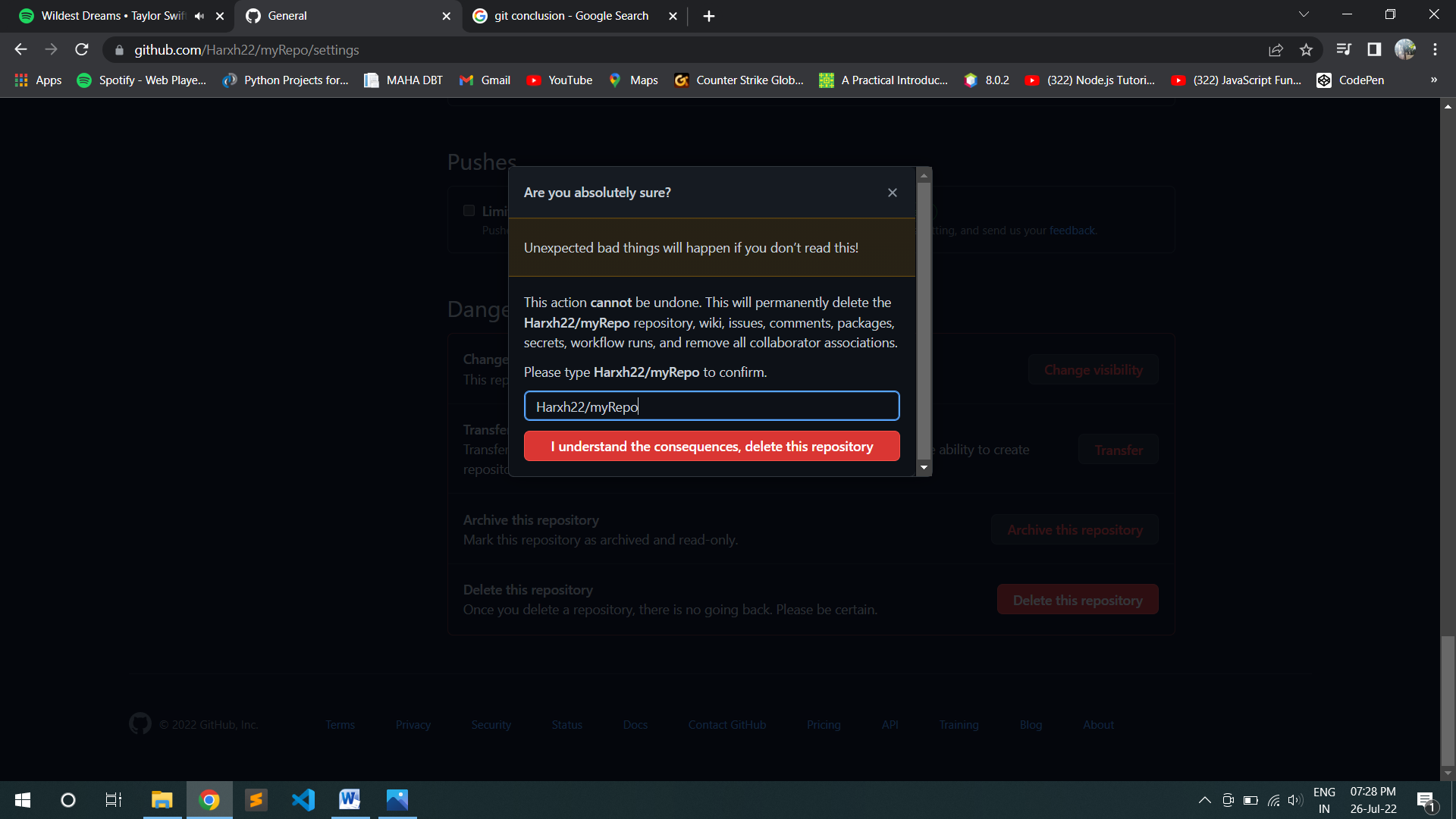
* By default, we can create public repository in Github. So we can copy the entire public repository of any other users in to own account using “FORK” Operation. Now fork the repository (Sharing with other users who wants to contribute).





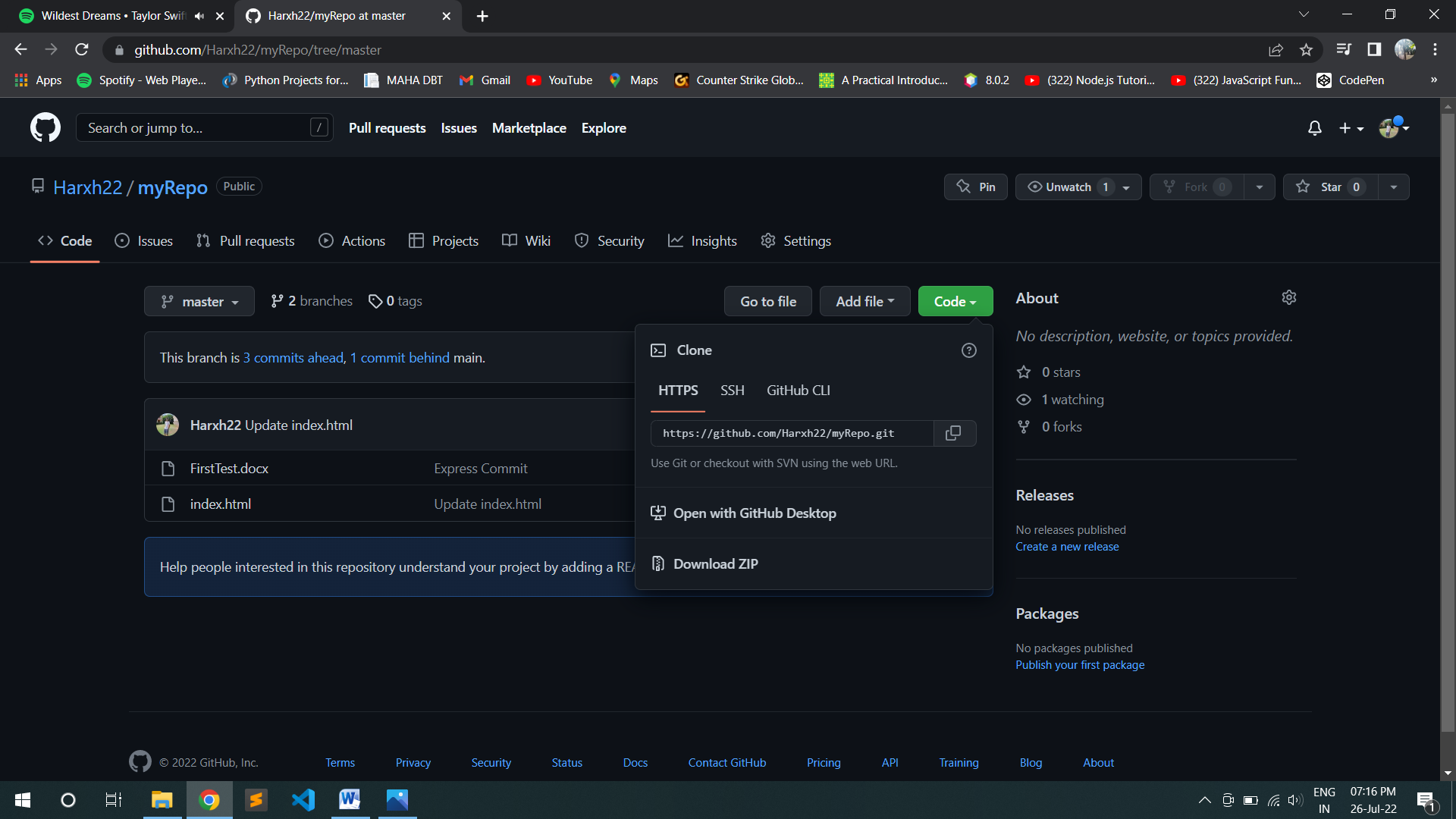
* To delete the repository, open the desired repository you want to delete and go to the settings option. There you will see delete repository button to delete it.
* Now, if you want to download a repository in local machine, then git clone command is used followed by path to repository.

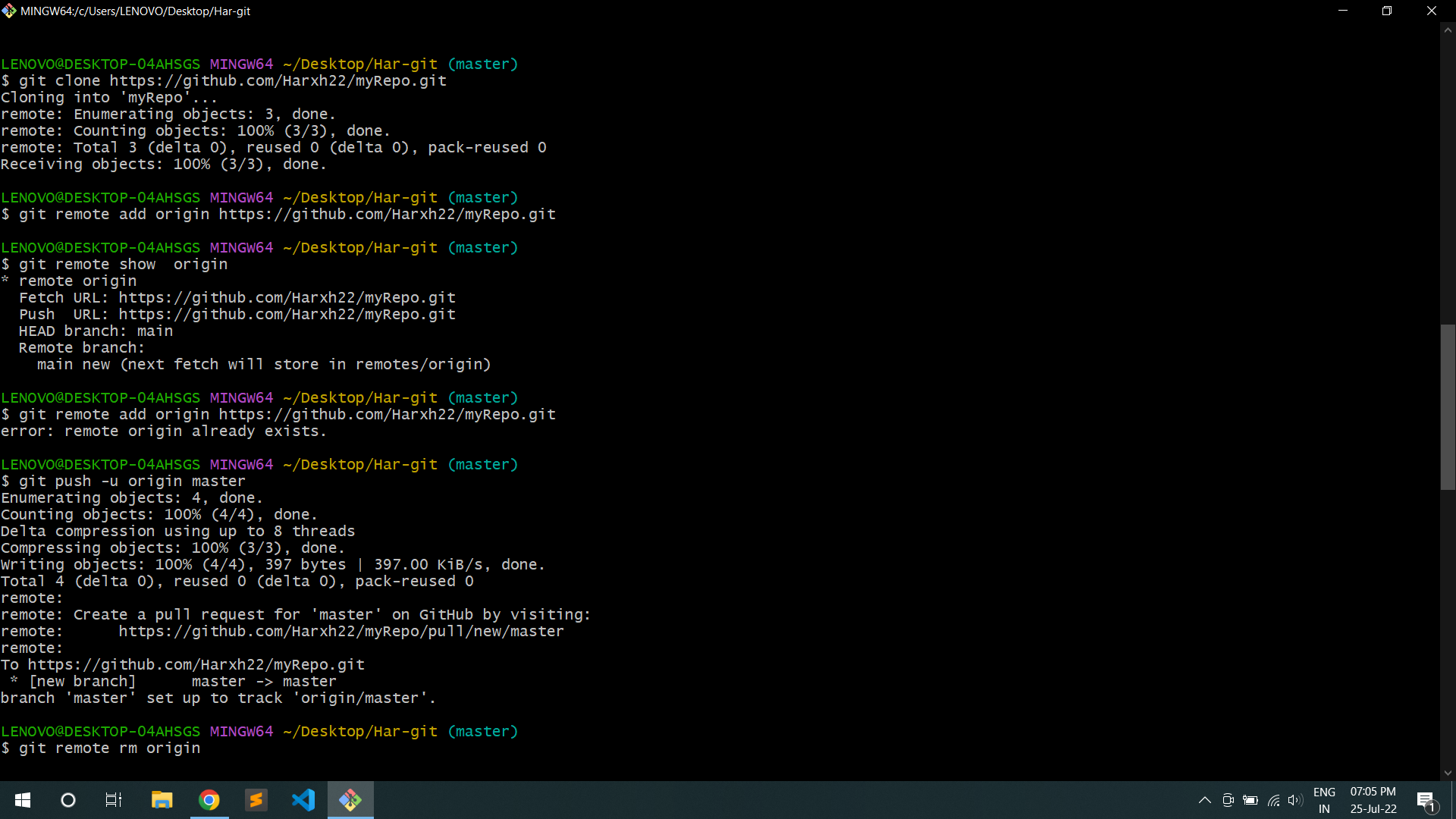






* In GitHub the path of repository can be known through clone or download button and it can be downloaded using git clone command as shown below.





# Pull and Push Processes

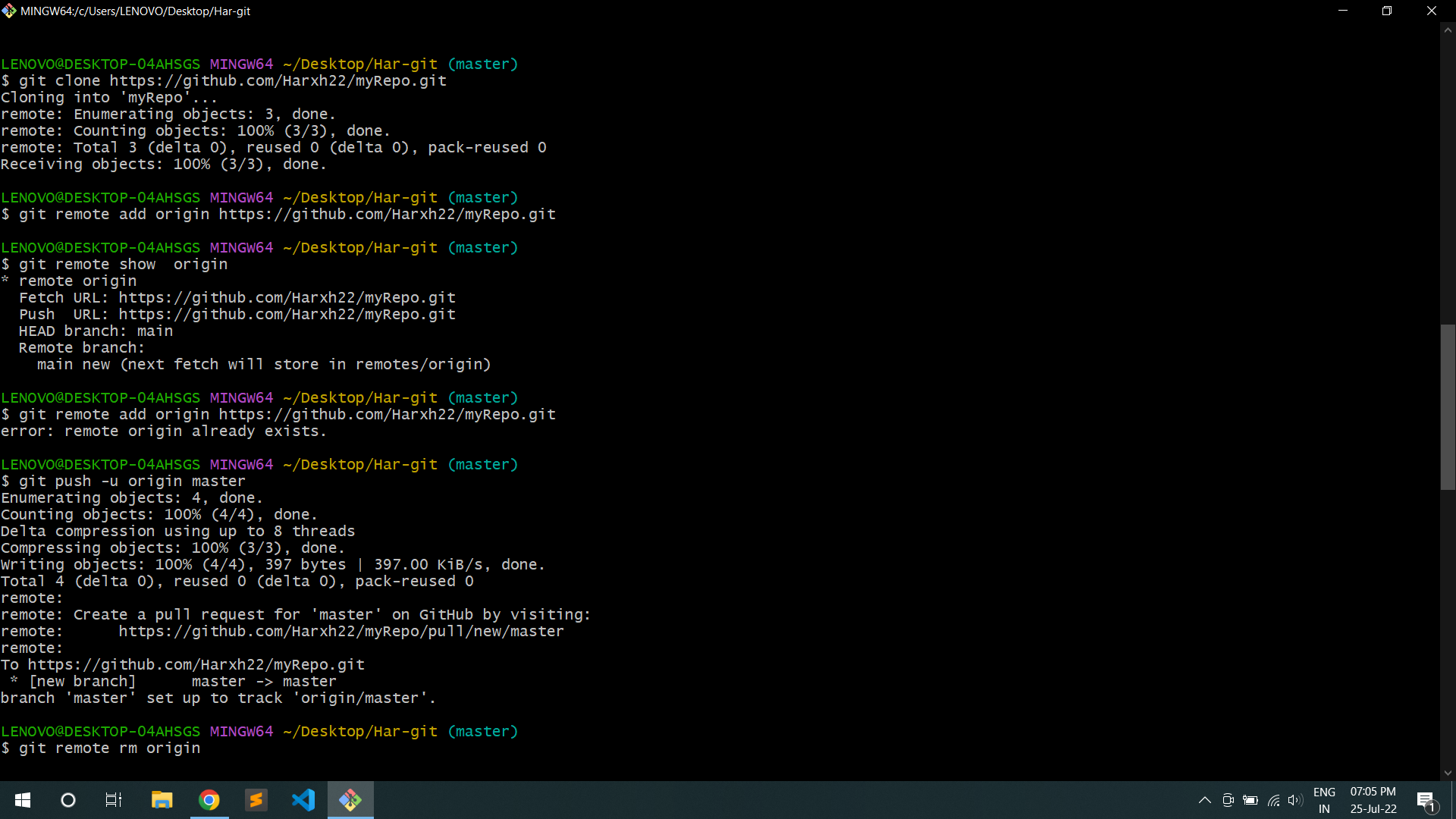
* The pull command used to fetch the repository from github to local while push is used to commit files from local repository to Github.
* **Push:** Push changes to Web repository
* **Pull:** Pull changes to Local repository
* The following commands are used for pull and push repositories:

1. **Push Command**

**$ git remote add origin** [**https://github.com/spd1512/DataCleaning.git**](https://github.com/spd1512/DataCleaning.git)

**$ git remote show origin**

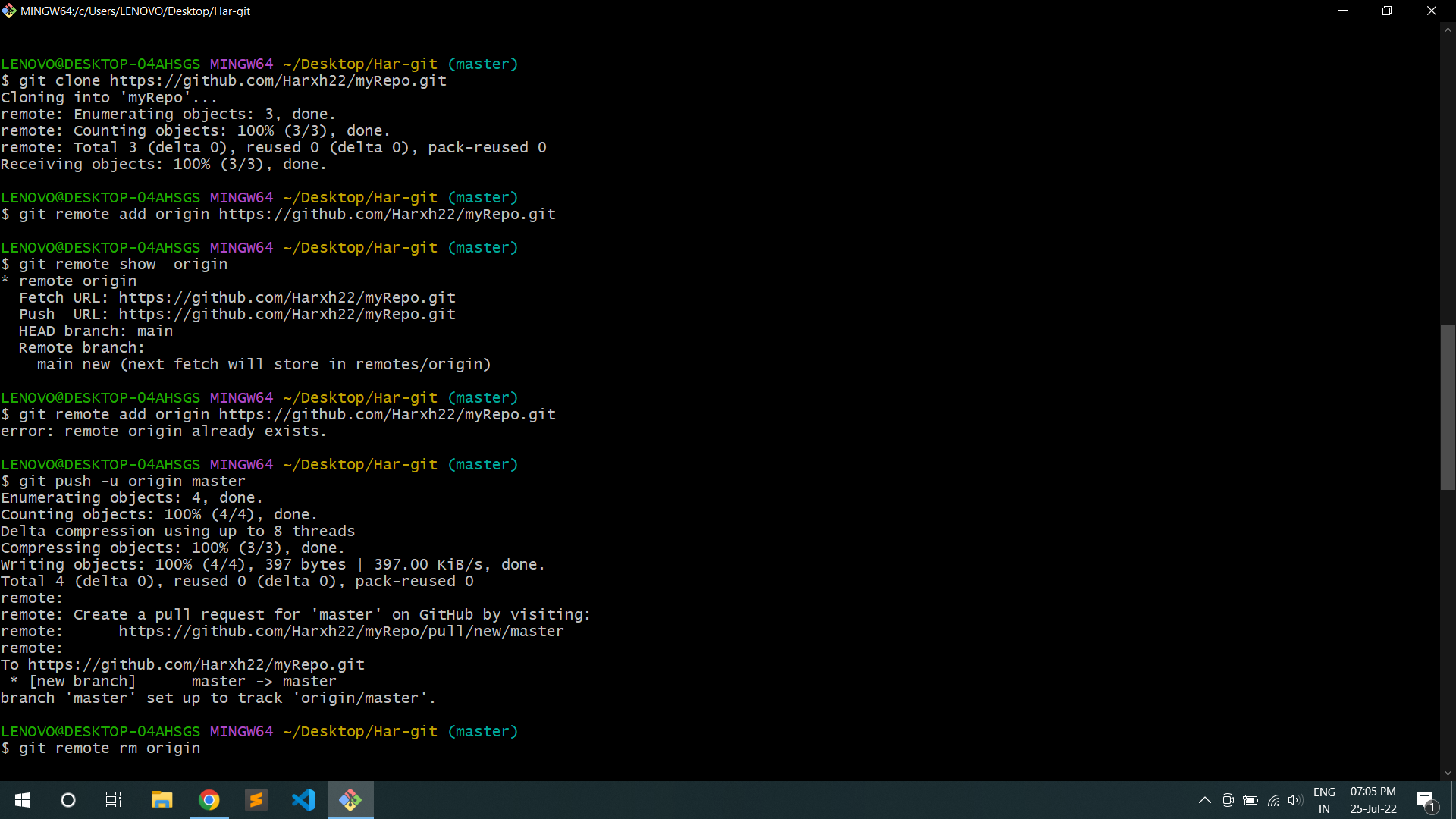




* If you add remote again then will show you fatal error.

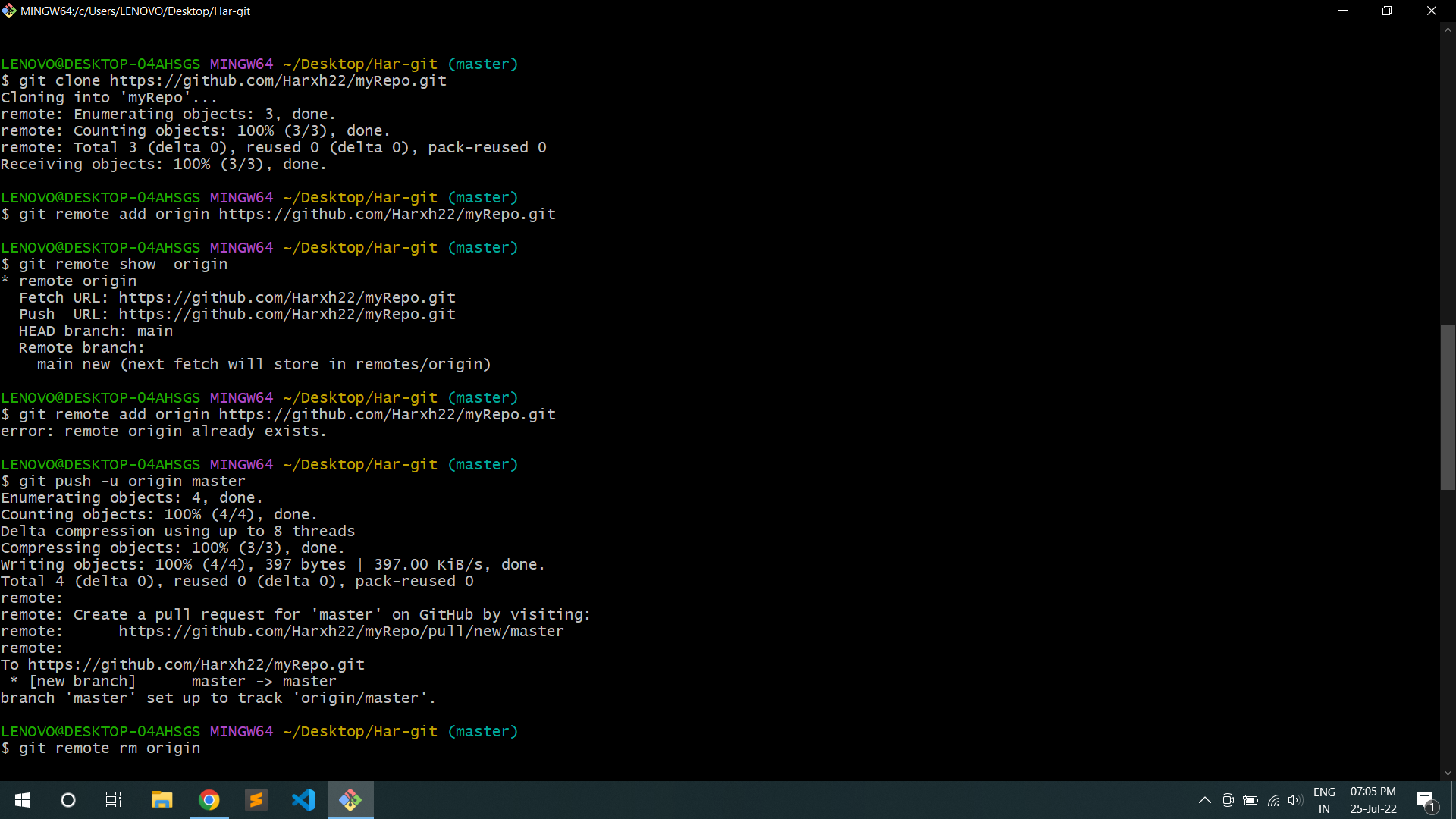
$ git remote add origin https://github.com/Harxh22/myRepo.git

fatal: remote originalready exists.



* Now, to push the local repository to remote github following command is used

# $ git push -u origin master

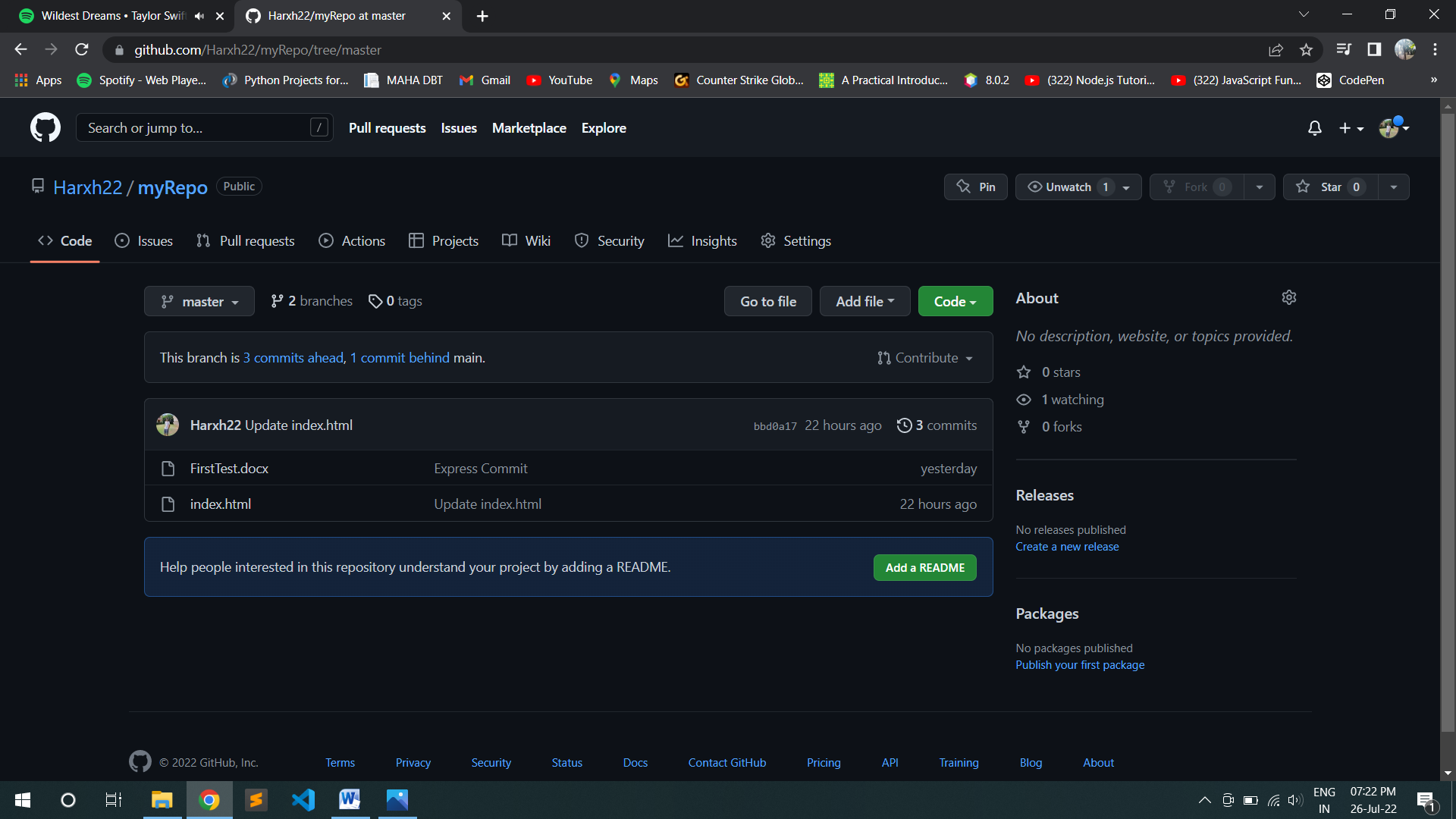




* So, to delete origin rm origin command is used

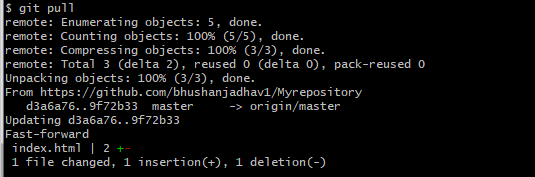
# $ git remote rm origin

* Now you can check the github for updated contents.



1. **Pull Command**

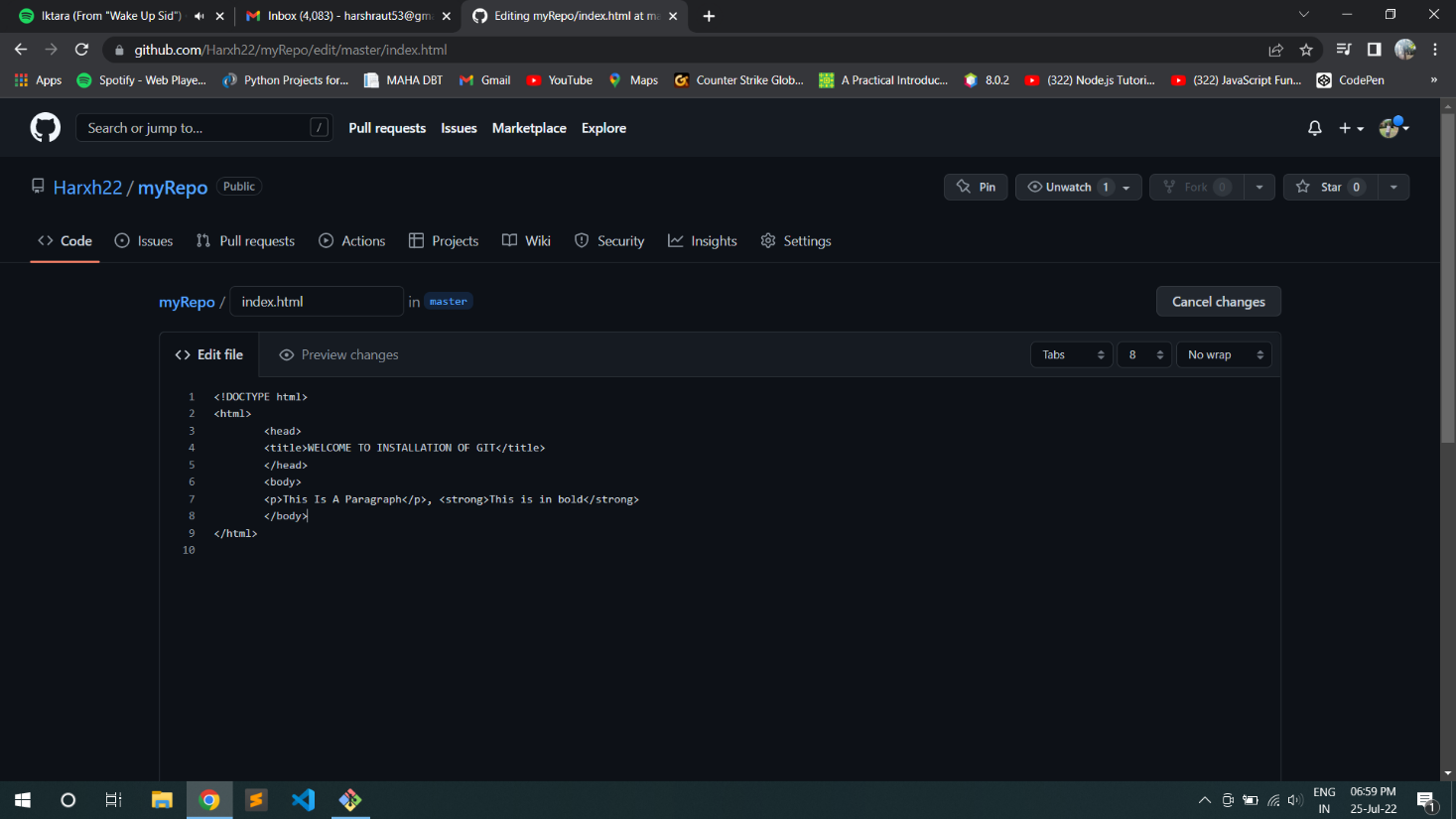
* Pull command is used to download the remote updated repository into local one.
* The command for download is **$ git pull**



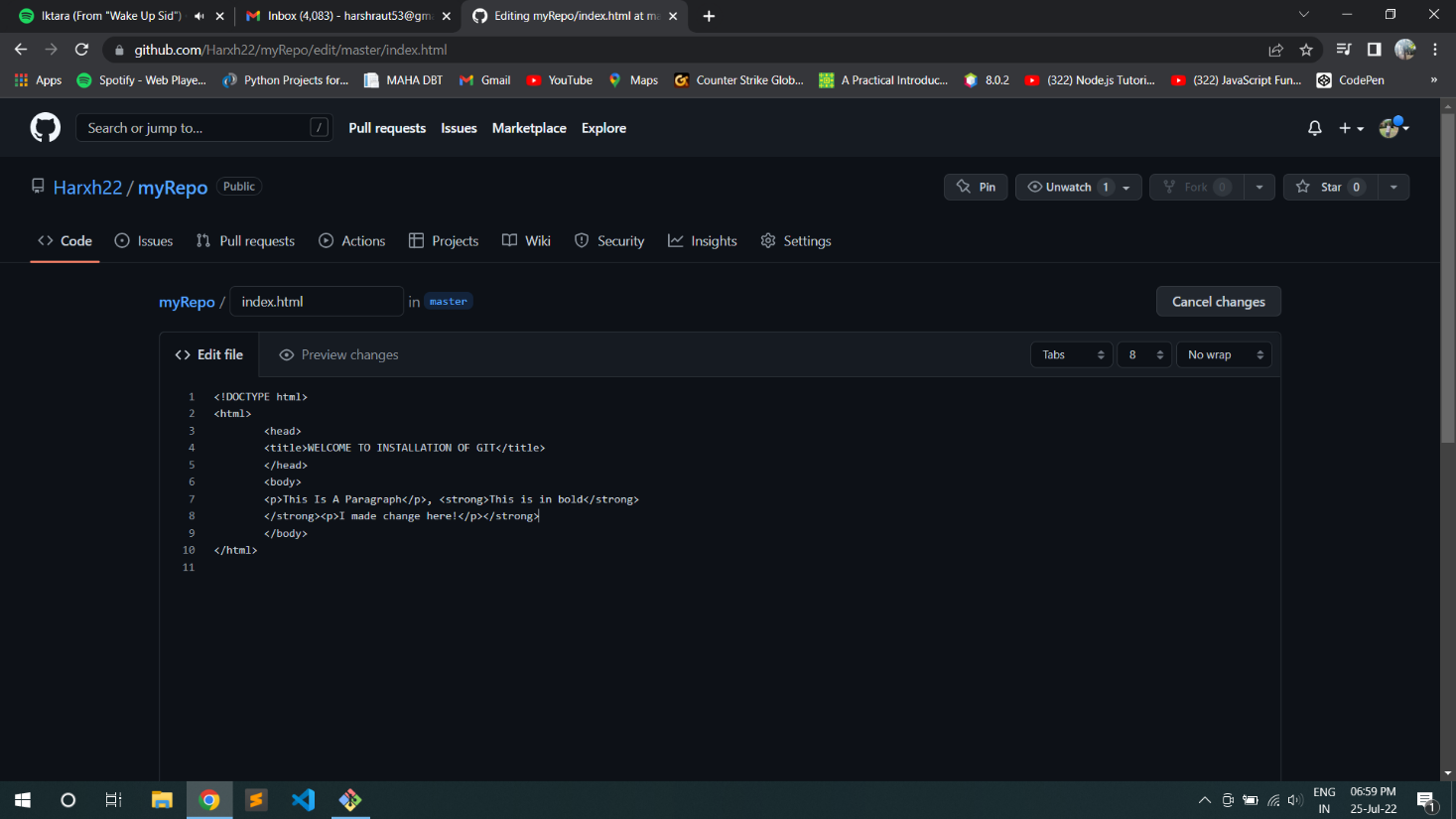


1. **Fetch Command**

* Suppose you have a file in github and you have changes that.



# Original File

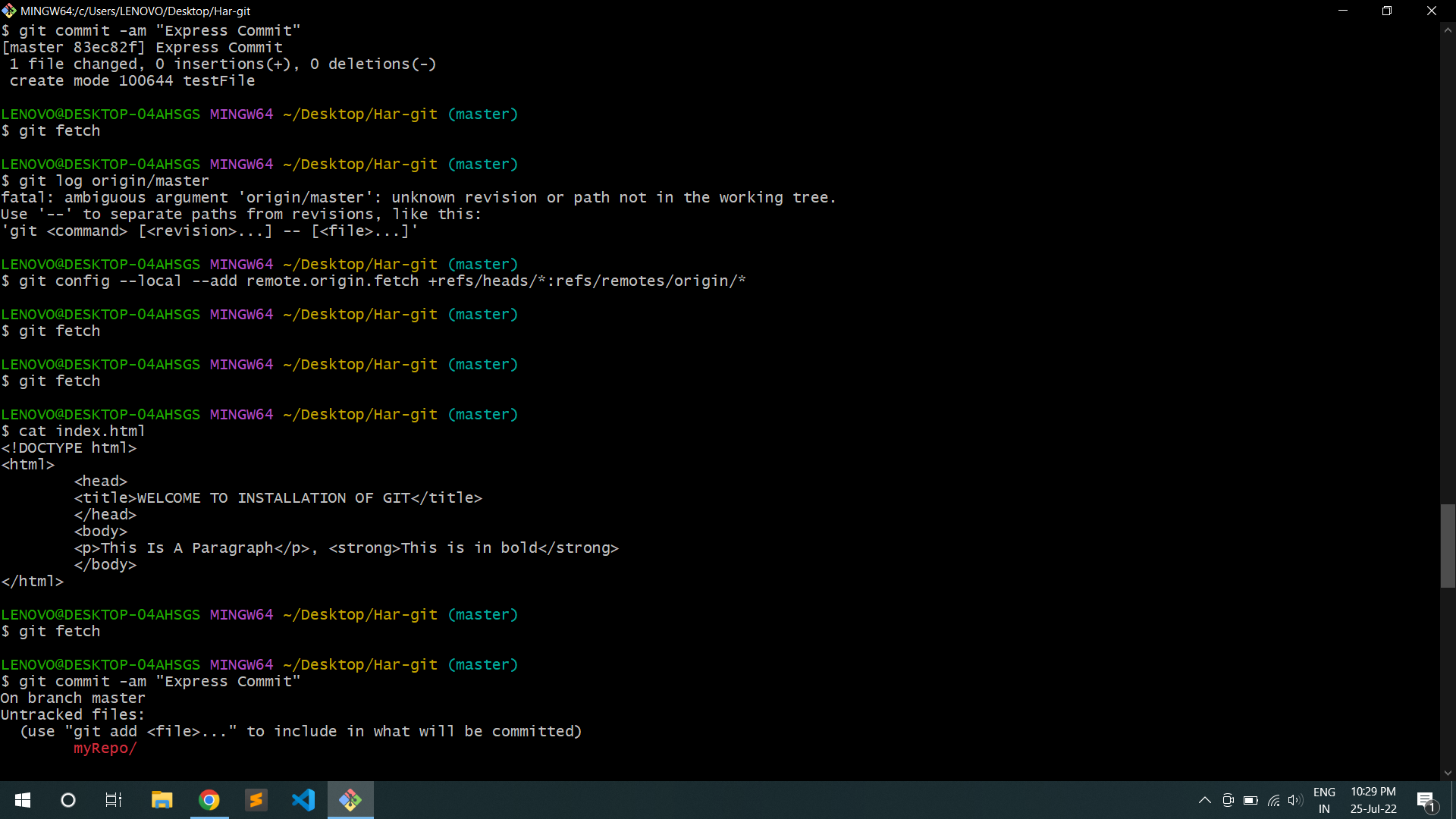


**Changed File**



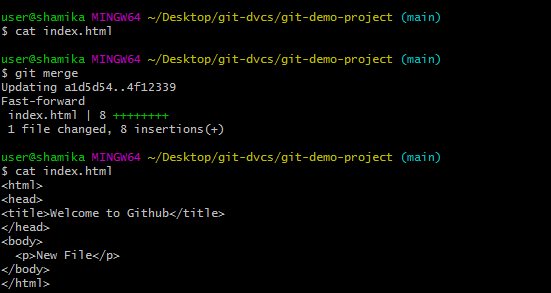
* Now we use fetch command to fetch the changes, which will show you both the files like original and changed in local repository.

# $ git fetch

****

* Here fetch will not show you like updated changes file as like push. So use merge command to merge the changes so use following command for merge.

# $ git merge origin/master





**Conclusion:**

Git and GitHub provide fast and convenient ways to track projects, whether the project is by one individual or a team of software developers. Although GitHub has many complex features available, it’s easily accessible for individual and small projects that need some kind of tracking mechanism. It has proven very useful to the open-source community and in academia as well.