Subject Name: Source Code Management

Subject Code: CS181

Cluster: Zeta

Department: CSE

CHITKARA
UNIVERSITY

Submitted By:	**Submitted To:**
Name: Archie	Ms. Swati Goel
Roll No.: 2110992425	
G33	

CHITKARA
UNIVERSITY

List of Tasks

Experiment 1

Aim: Setting up of Git Client

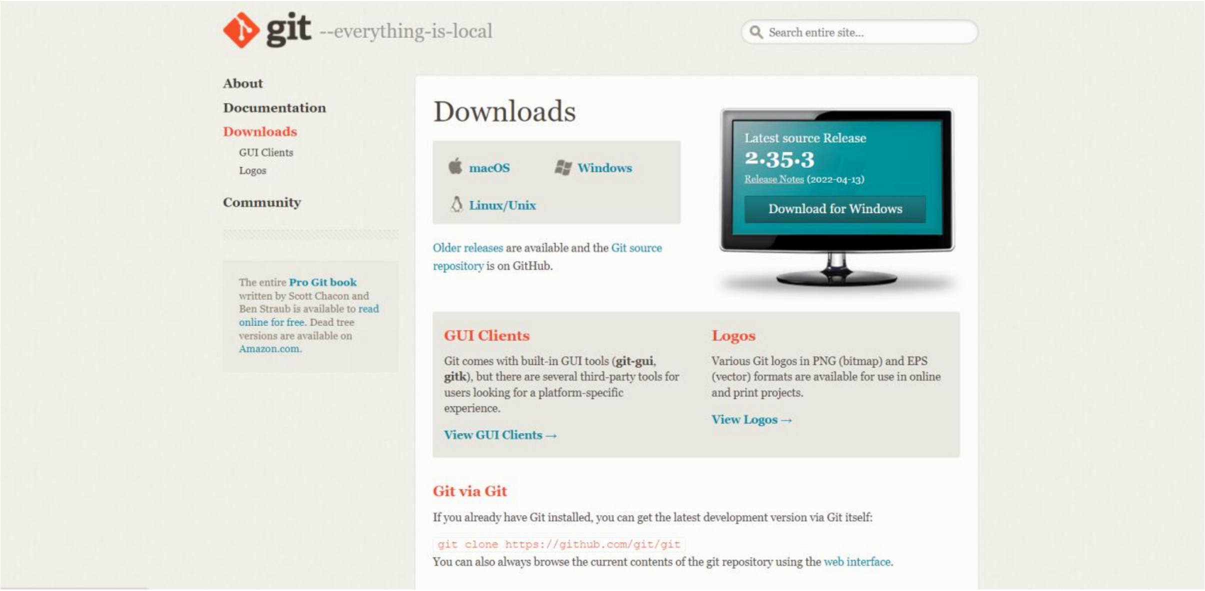
Theory:

Git: Git is useful to anyone who writes code or track changes to files. Git is used to tracking changes in the source code, enabling multiple developers to work together or non-linear development.

What is GIT? : Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently.

Procedure: We can download git in Windows, Linux, Mac. Download the latest version of Git and choose the 64/32 bit version. After the file is downloaded, install it in the system. Once installed, select Launch the Git Bash, then click on finish. The Git Bash is now launched.

Snapshots for Installation:



The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Older releases are available and the Git source repository is on GitHub.

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

Logos

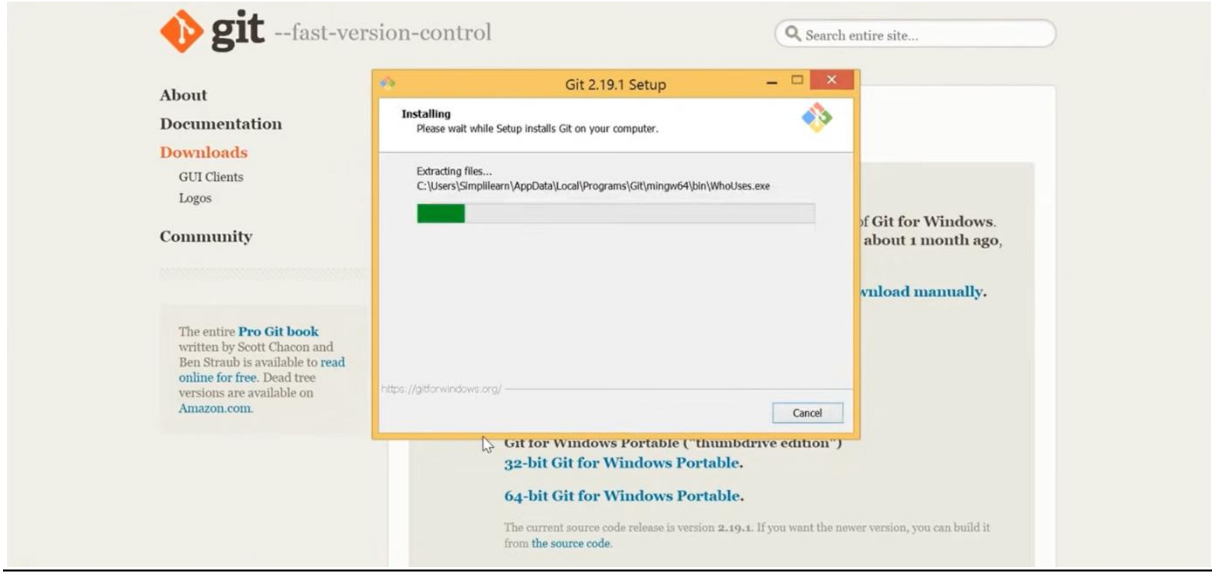
Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the web interface.



The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Installing
Please wait while Setup installs Git on your computer.

Extracting files...
C:\Users\Simplilearn\AppData\Local\Programs\Git\mingw64\bin\WhoUses.exe

https://gitforwindows.org/

f Git for Windows. about 1 month ago,

wnload manually.

Git for Windows Portable ("thumbdrive edition")

32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

The current source code release is version 2.19.1. If you want the newer version, you can build it from the source code.

Experiment 2

Aim: Setting up GitHub Account

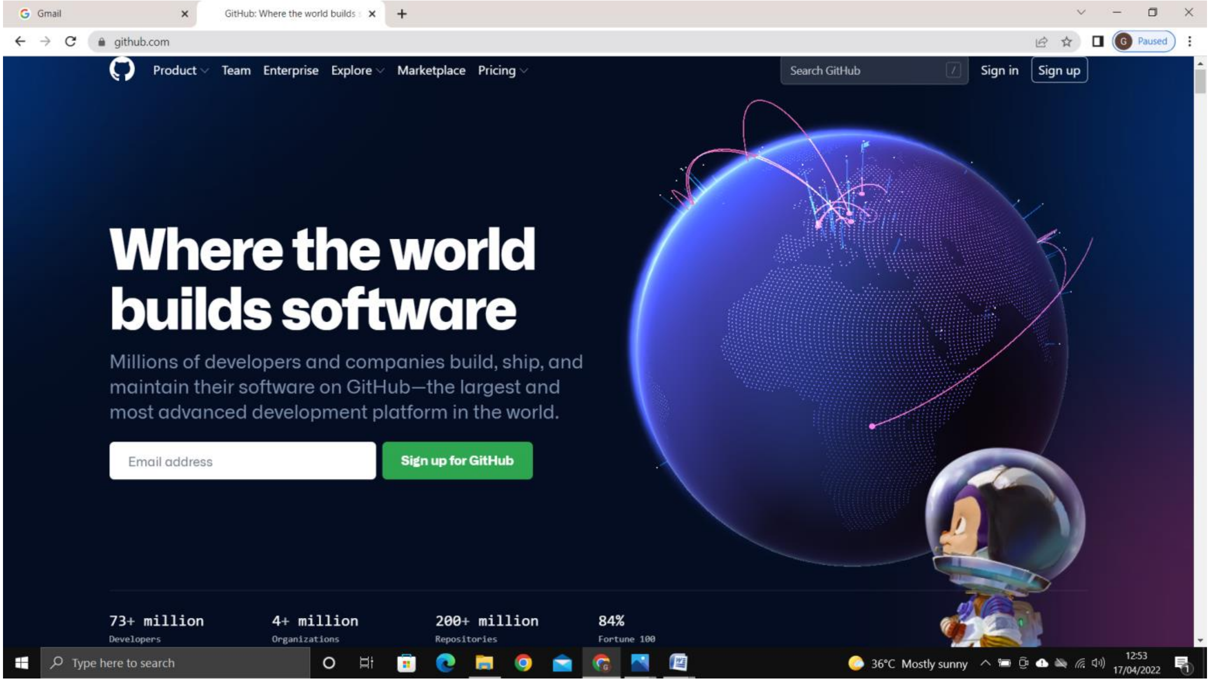
Theory:

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. GitHub lets multiple people make separate changes to web pages at the same time. GitHub encourages teams to work together to build and edit their site content.

Advantages of GitHub: GitHub allows multiple users to work on a single project at a time. It helps all the team members to work together on a single project at a time from different locations. Here chance of losing data is less as it provides multiple copies of it. GitHub is most preferred platform for developers especially whenever the developers are working on the single project at a time from different locations.

Procedure:

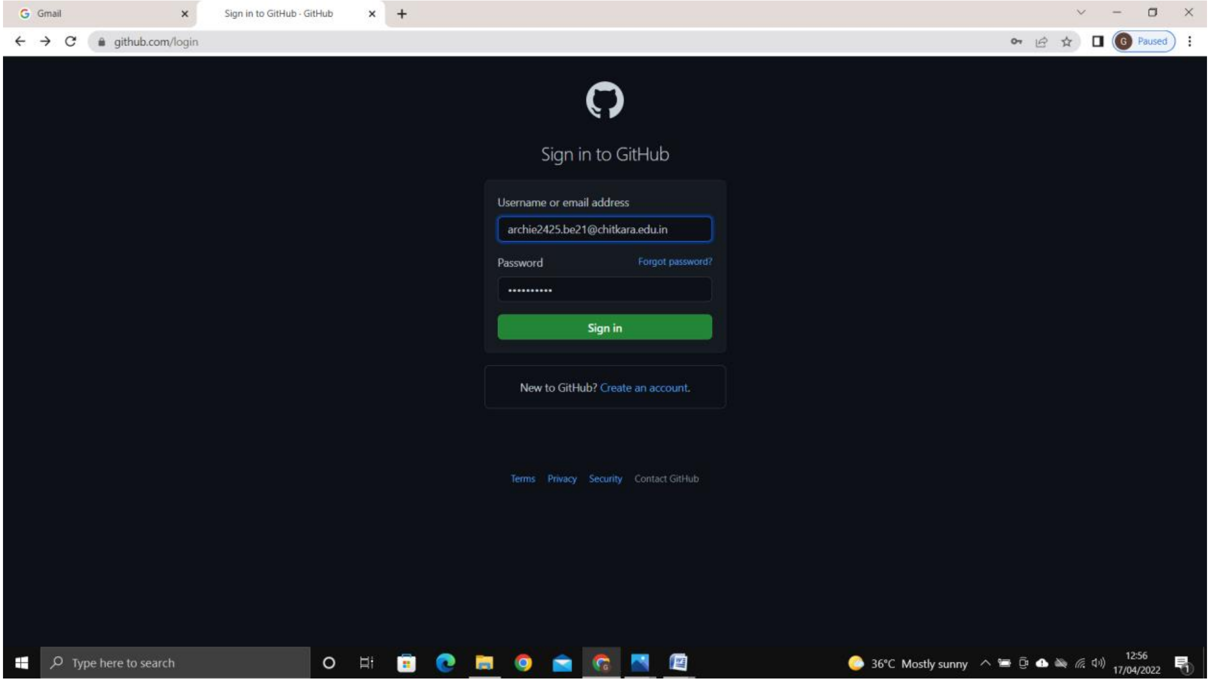
Step 1) Go to https://github.com/signup



Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

Step 2) After visiting the link you need to create an account. If you already have an account you can sign in else you need to create one.

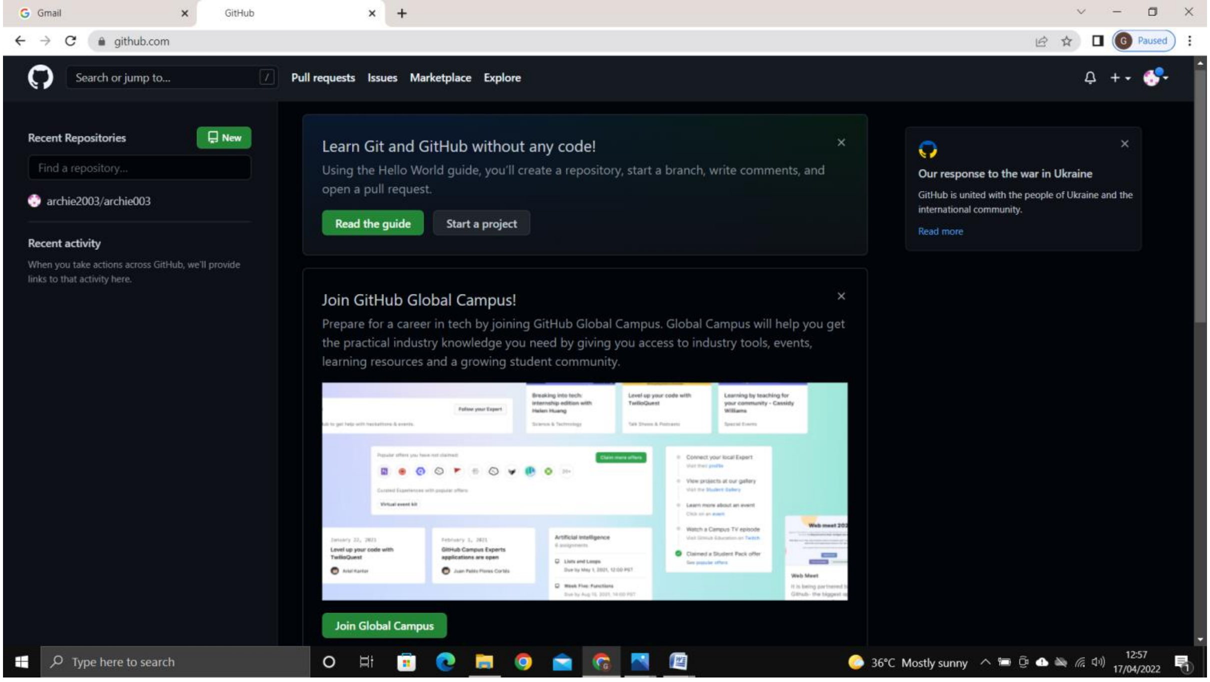


Sign in to GitHub

Username or email address: archie2425.be21@chitkara.edu.in

Password: ••••••••••

New to GitHub? Create an account.



Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Join GitHub Global Campus!

Prepare for a career in tech by joining GitHub Global Campus. Global Campus will help you get the practical industry knowledge you need by giving you access to industry tools, events, learning resources and a growing student community.

Our response to the war in Ukraine

GitHub is united with the people of Ukraine and the international community.

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

Linking GitHub Account with GitBash:

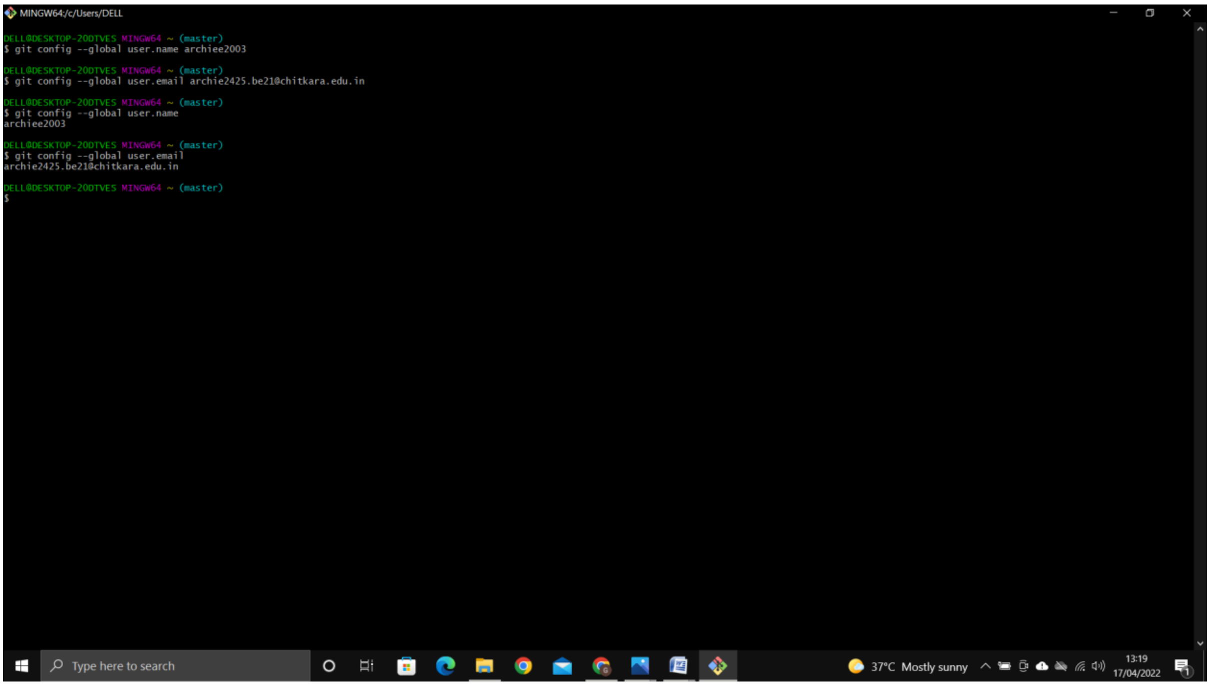
Username: git config --global user.name “username in git”

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

Verify username and email:

Username: git config --global user.name

Email: git config --global user.email



```
DELL@DESKTOP-20DTVES MINGW64 ~ (master)
$ git config --global user.name archiee2003

DELL@DESKTOP-20DTVES MINGW64 ~ (master)
$ git config --global user.email archie2425.be21@chitkara.edu.in

DELL@DESKTOP-20DTVES MINGW64 ~ (master)
$ git config --global user.name
archiee2003

DELL@DESKTOP-20DTVES MINGW64 ~ (master)
$ git config --global user.email
archie2425.be21@chitkara.edu.in

DELL@DESKTOP-20DTVES MINGW64 ~ (master)
$
```

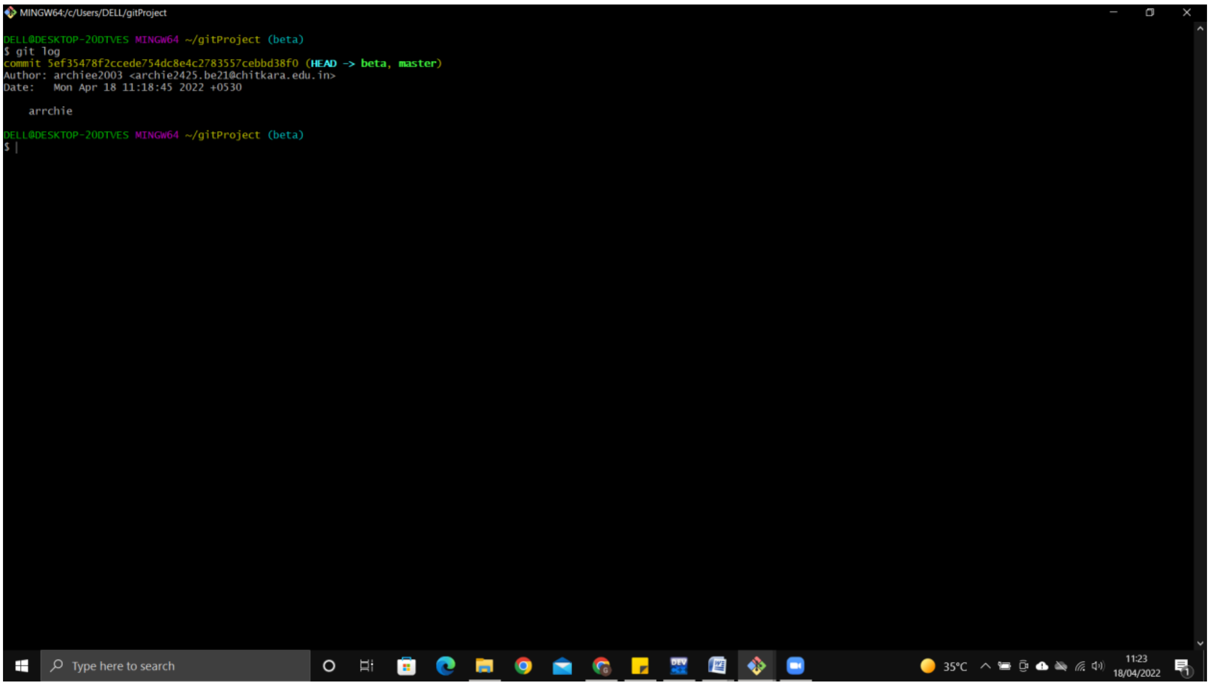
EXPERIMENT 3

Aim: Generate Logs

Theory:

Logs: Git log is a utility tool to review and read a history of everything that happens to a repository. Multiple options can be used with a git log to make history more specific. Generally, the git log is a record of commits.

Why logs? : Logs help to check what were the changes in the code or any other file and by whom. It also contains the number of insertions and a deletions including at which time it was changed.



```
DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ git log
commit 5ef35478f2ccede754dc8e4c2783557cebbd38f0 (HEAD -> beta, master)
Author: archiee2003 <archie2425.be21@chitkara.edu.in>
Date:   Mon Apr 18 11:18:45 2022 +0530

    arrchie

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ |
```

Experiment 4

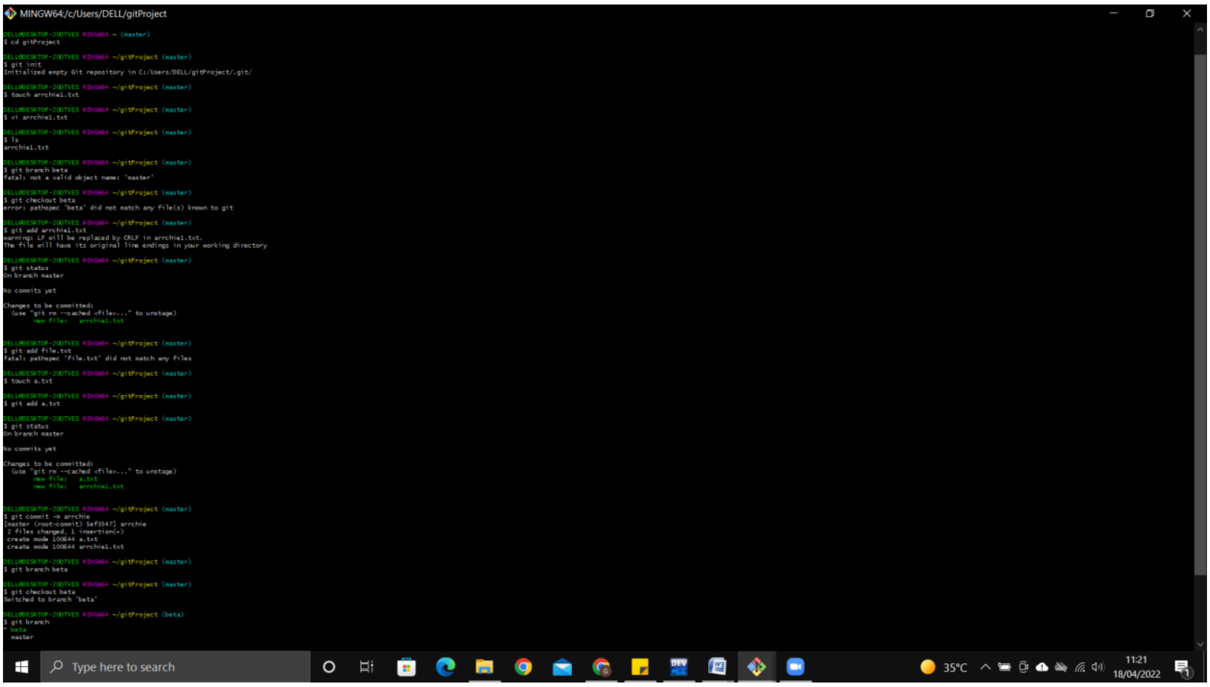
Aim: Create and visualize branches

Create Branches: A branch is a version of the repository that diverges from the main working project. It is a feature available in most modern version control systems. A Git project can have more than one branch. These branches are a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug, you spawn a new branch to summarize your changes. So, it is complex to merge the unstable code with the main code base and also facilitates you to clean up your future history before merging with the main branch.

1) For creating a new Branch: git branch "name of the branch"

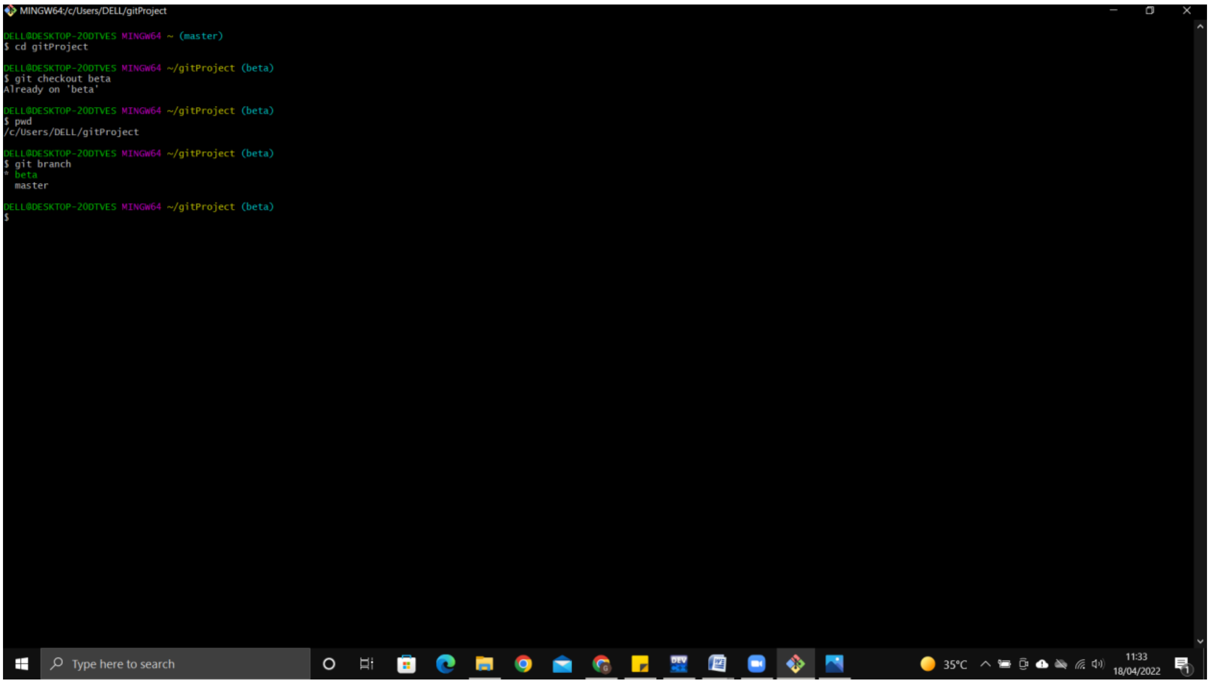
2) To check how many branches do we have: git branch

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



```
DELL@DESKTOP-20DTVES MINGW64 ~ (master)
$ cd gitProject

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git init
Initialized empty Git repository in C:/Users/DELL/gitProject/.git/

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ touch arrchiel.txt

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ vi arrchiel.txt

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ ls
arrchiel.txt

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git branch beta
fatal: not a valid object name: 'master'

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git checkout beta
error: pathspec 'beta' did not match any file(s) known to git

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git add arrchiel.txt
warning: LF will be replaced by CRLF in arrchiel.txt.
The file will have its original line endings in your working directory

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   arrchiel.txt


DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git add file.txt
fatal: pathspec 'file.txt' did not match any files

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ touch a.txt

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git add a.txt

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   a.txt
        new file:   arrchiel.txt


DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git commit -m arrchie
[master (root-commit) 5ef3547] arrchie
 2 files changed, 1 insertion(+)
 create mode 100644 a.txt
 create mode 100644 arrchiel.txt

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git branch beta

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git checkout beta
Switched to branch 'beta'

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ git branch
* beta
  master
```

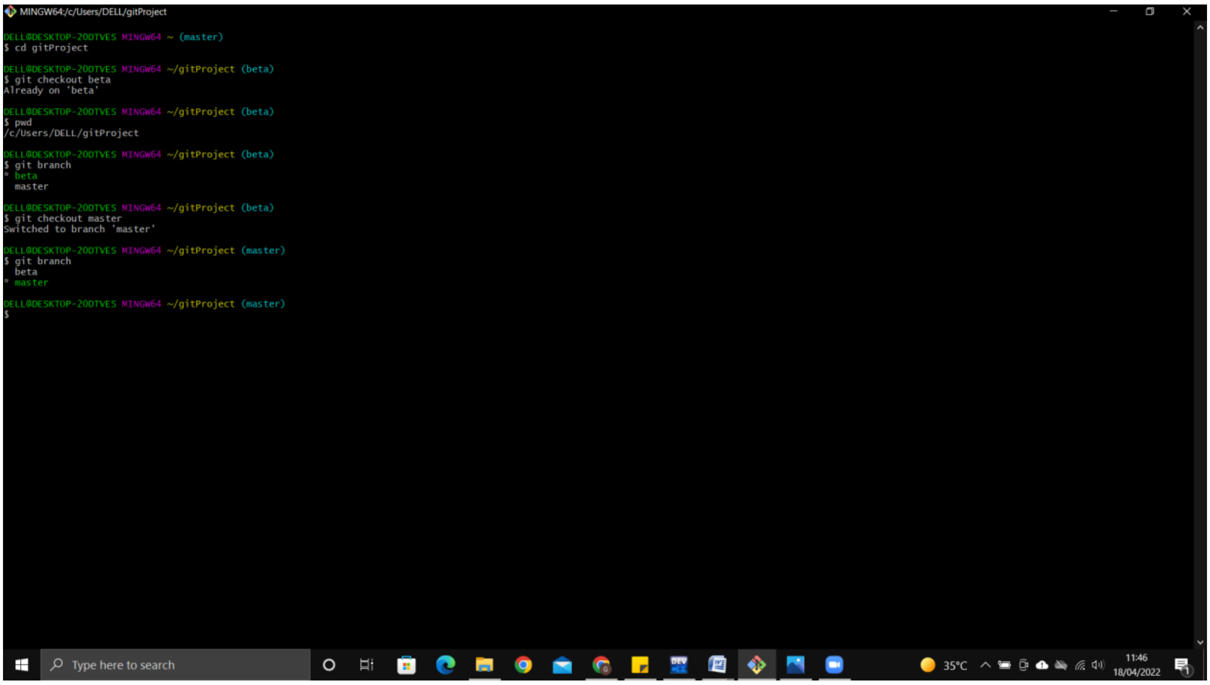
<u>Visualizing Branches:</u> To visualize I have created a new files in a new branch beta instead of master branch.



MINGW64:/c/Users/DELL/gitProject

```
DELL@DESKTOP-20DTVES MINGW64 ~ (master)
$ cd gitProject

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ git checkout beta
Already on 'beta'

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ pwd
/c/Users/DELL/gitProject

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ git branch
* beta
  master

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$
```

After this we will change the branch from activity1 to master, but when I will switch to the master branch there will not be the same file in the master , it will not show the new file in the master branch.



MINGW64:/c/Users/DELL/gitProject

```
DELL@DESKTOP-20DTVES MINGW64 ~ (master)
$ cd gitProject

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ git checkout beta
Already on 'beta'

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ pwd
/c/Users/DELL/gitProject

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ git branch
* beta
  master

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ git checkout master
Switched to branch 'master'

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git branch
  beta
* master

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$
```

Experiment 5

Aim: Git Lifecycle Description

Theory:

Stages in GIT lifecycle: Git is one of the premier distributed version control systems available for programmers and corporate. We will see details about how a project that is being tracked by git proceeds with workflow i.e. Git Life Cycle. As the name suggests is regarding different stages involved after cloning the file from the repository. It covers the git central commands or main commands that are required for this particular version control system.

The stages are:

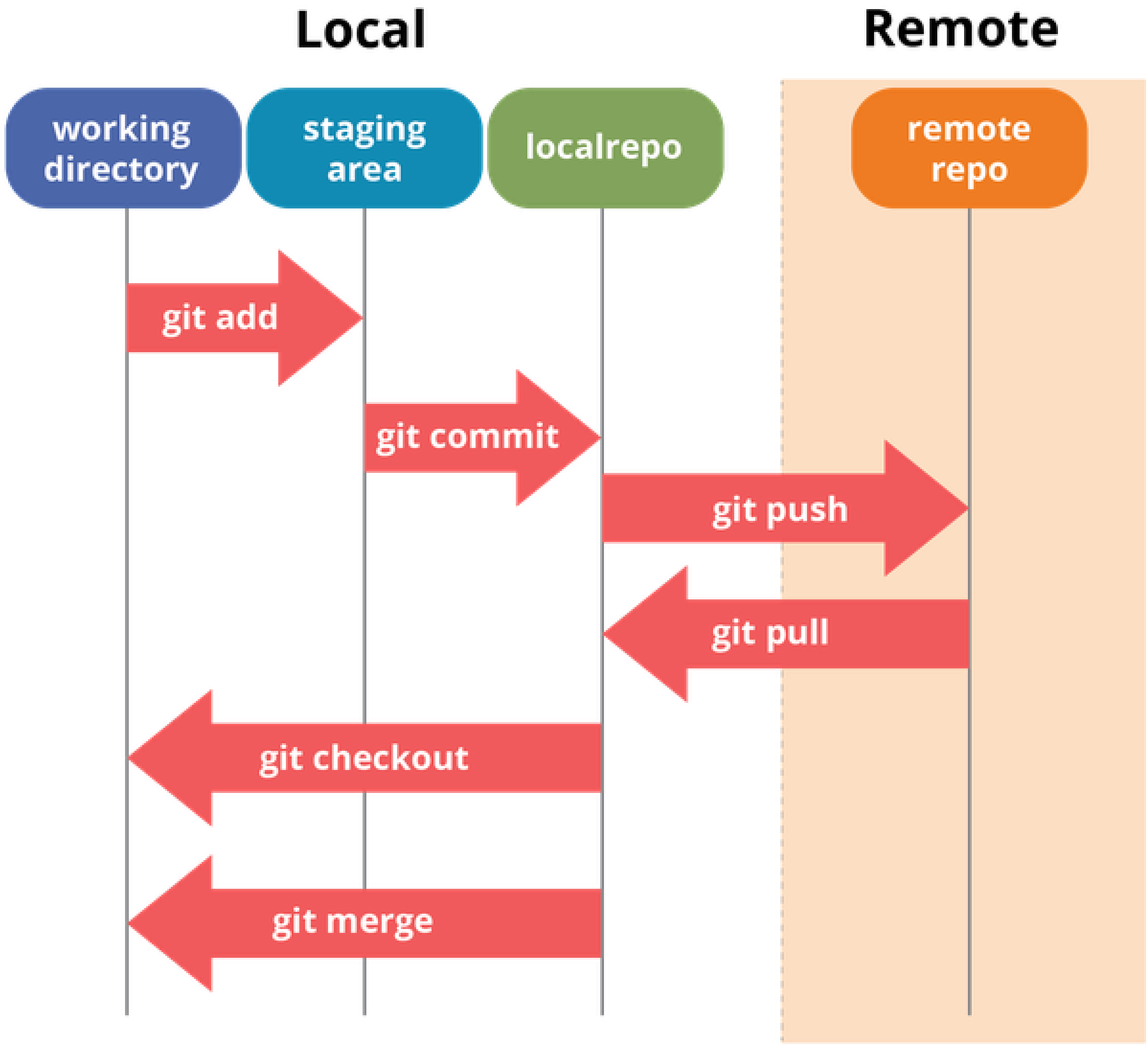
- Working Directory
- Staging Area
- Git Directory

1) Working Directory: Whenever we want to initialize our document to make it a git repository, we use the git init command. After this command, Git becomes aware of these files in the project although it doesn't track the files yet. The files are further tracked in the staging area.

2) Staging Area: Now, to track the different versions of our files we use the command git add. We can term the staging area as a place where different versions of our files are stored. Git add command copies the version of your file from your working directory to the staging area. We can, however, choose which files we need to add to the staging area because in our working directory there are some files that we don't want to br tracked.

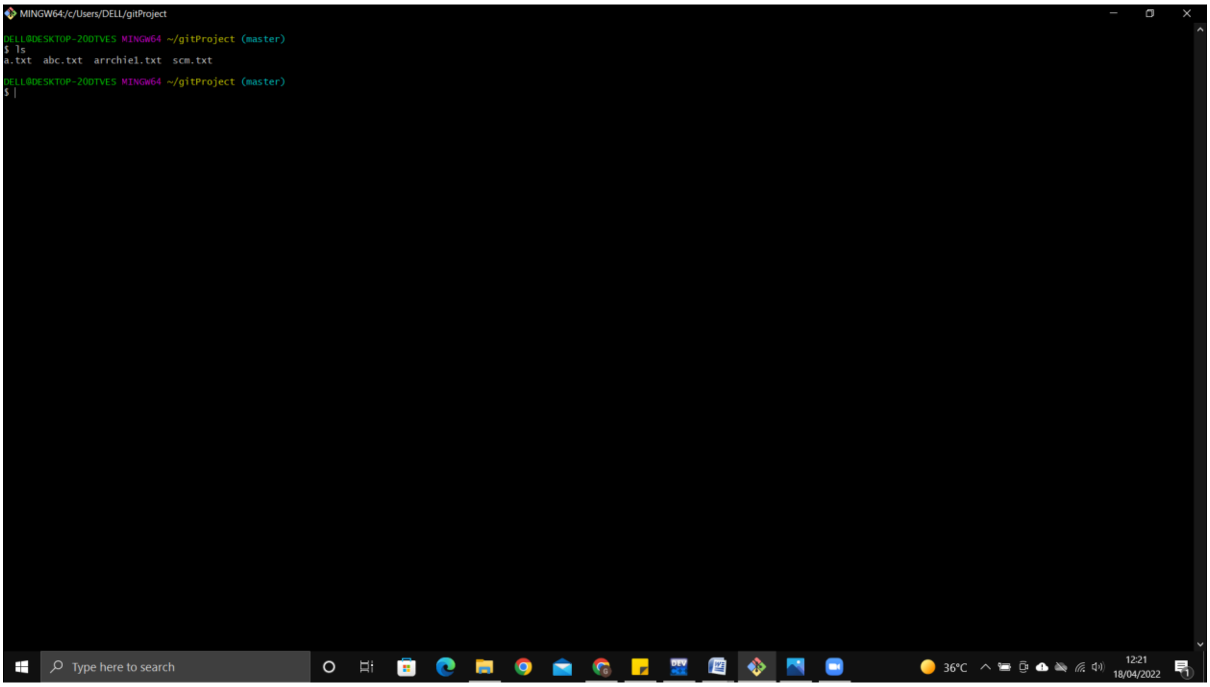
3) Git Directory: Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the git commit command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about. Git preserves the information or the metadata of the files that were committed in a Git Directory which helps Git in tracking files and basically it preserves the photocopy of the committed files. Commit also stores the name of the author who did the commit, files that are committed, and the date at which they are committed along with the commit message.

4) **Remote Repository**→ means mirror or clone of the local Git repository in GitHub. And pushing means uploading the commits from the local Git repository to a remote repository hosted in GitHub.



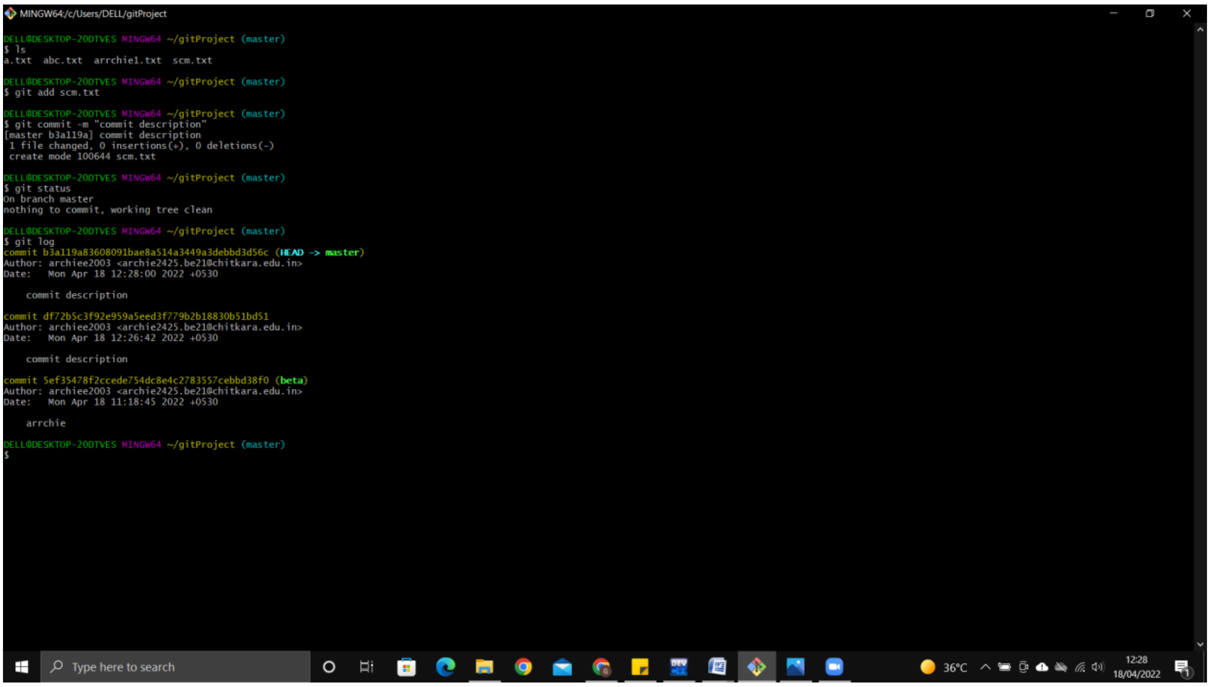
Procedure:

1) To see a list of available files in the git folder open the git bash terminal type the command ls to see the list of available files.



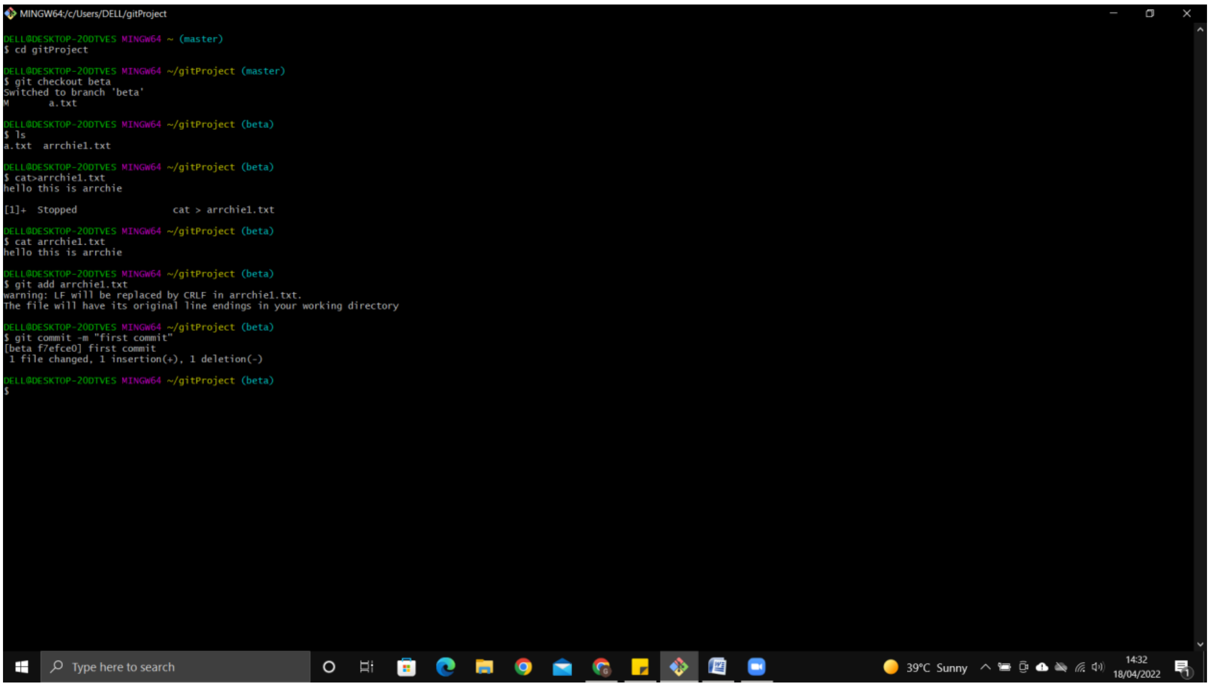
```
DELL@DESKTOP-2ODTVES MINGW64 ~/gitProject (master)
$ ls
a.txt  abc.txt  arrchie1.txt  scm.txt

DELL@DESKTOP-2ODTVES MINGW64 ~/gitProject (master)
$ |
```

2) To commit a file in git bash terminal add a file using the command git add filename.txt
After writing the above command write the following command to commit the file git commit –m “commit description”.
Now, type git log to see the history of the previous commit.



```
DELL@DESKTOP-2ODTVES MINGW64 ~/gitProject (master)
$ ls
a.txt  abc.txt  arrchie1.txt  scm.txt

DELL@DESKTOP-2ODTVES MINGW64 ~/gitProject (master)
$ git add scm.txt

DELL@DESKTOP-2ODTVES MINGW64 ~/gitProject (master)
$ git commit -m "commit description"
[master b3a119a] commit description
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 scm.txt

DELL@DESKTOP-2ODTVES MINGW64 ~/gitProject (master)
$ git status
On branch master
nothing to commit, working tree clean

DELL@DESKTOP-2ODTVES MINGW64 ~/gitProject (master)
$ git log
commit b3a119a83608091bae8a514a3449a3debbd3d56c (HEAD -> master)
Author: archiee2003 <archie2425.be21@chitkara.edu.in>
Date:   Mon Apr 18 12:28:00 2022 +0530

    commit description

commit df72b5c3f92e959a5eed3f779b2b18830b51bd51
Author: archiee2003 <archie2425.be21@chitkara.edu.in>
Date:   Mon Apr 18 12:26:42 2022 +0530

    commit description

commit 5ef35478f2ccede754dc8e4c2783557cebbd38f0 (beta)
Author: archiee2003 <archie2425.be21@chitkara.edu.in>
Date:   Mon Apr 18 11:18:45 2022 +0530

    arrchie

DELL@DESKTOP-2ODTVES MINGW64 ~/gitProject (master)
$
```

3) To add content in git file write the following command cat>fileName.txt and write something after pressing something after pressing the following command cat filename.txt After pressing the enter key on the terminal the line you wrote with the above cat command will be displayed. After following the above steps write the command git add filename.txt and ignore the warning displayed on the terminal. After that commit the file and check the status in the steps mentioned previously.



```
DELL@DESKTOP-20DTVES MINGW64 ~ (master)
$ cd gitProject

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (master)
$ git checkout beta
Switched to branch 'beta'
M       a.txt

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ ls
a.txt  arrchie1.txt

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ cat>arrchie1.txt
hello this is arrchie

[1]+  Stopped                 cat > arrchie1.txt

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ cat arrchie1.txt
hello this is arrchie

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ git add arrchie1.txt
warning: LF will be replaced by CRLF in arrchie1.txt.
The file will have its original line endings in your working directory

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$ git commit -m "first commit"
[beta f7efce0] first commit
 1 file changed, 1 insertion(+), 1 deletion(-)

DELL@DESKTOP-20DTVES MINGW64 ~/gitProject (beta)
$
```