Chitkara University Institute of Engineering and Technology, Punjab

Academic Year: 2021-2022

Department of Computer Science and Engineering

Course Name: Source Code Management

Course Code: CS181

Assignment: Task 1.1

Submitted to:

Dr. Monit Kapoor

Submitted by:

Hardik Sawhney

2110990546

G-01 (Beta Cluster)

INDEX

TASK: 1.1.1

SETTING UP OF GIT CLIENT

What is Git?

Git is a version control system for:

- Tracking code changes
- Tracking who made changes
- Coding collaboration

What does Git do?

- Manage projects with **Repositories**
- **Clone** a project to work on a local copy
- Control and track changes with **Staging** and **Committing**
- **Branch** and **Merge** to allow for work on different parts and versions of a project
- **Pull** the latest version of the project to a local copy
- **Push** local updates to the main project

How to install Git on Windows?

We can download Git for Windows, Mac and Linux from https://git-scm.com/downloads. In the steps given below we will see how to install Git on windows.

Step 1: To download the most recent maintained build 64 bit version of Git for Windows, click on "Download for windows" button.
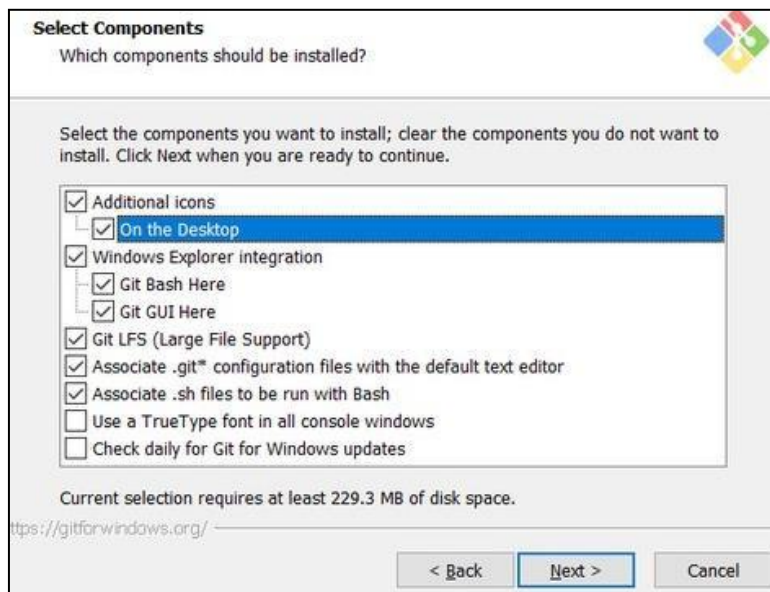
Step 2: Go to the folder where new downloads get store. Double click on the installer and Git Setup dialog box will open up.
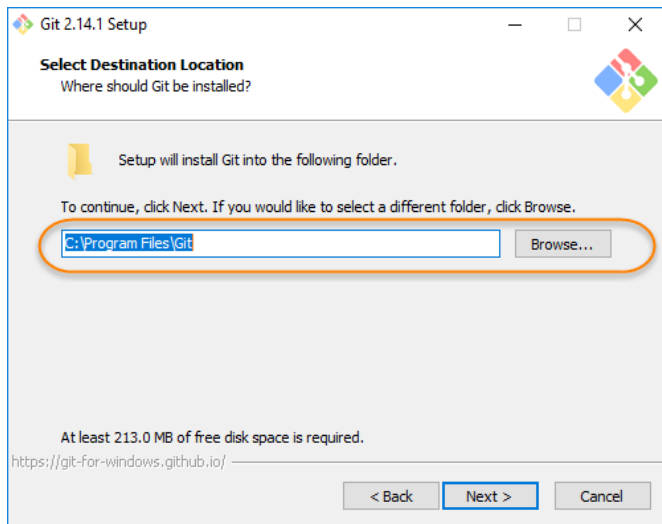

**Step 3:** Tick the following checkboxes and click on Next button.

Step 4: In the next step, you can choose the editor which suits you best and click on Next Button.
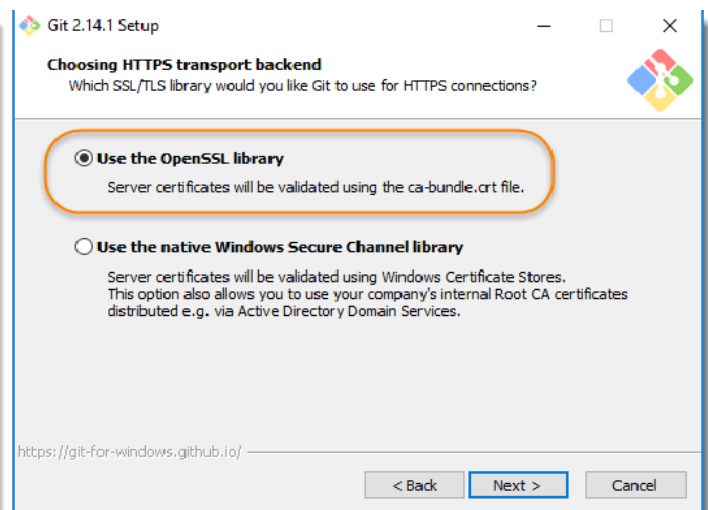
Step 5: Choose the folder where you want to install it. You can go with the default filled folder or you can change it. Click on Next button.
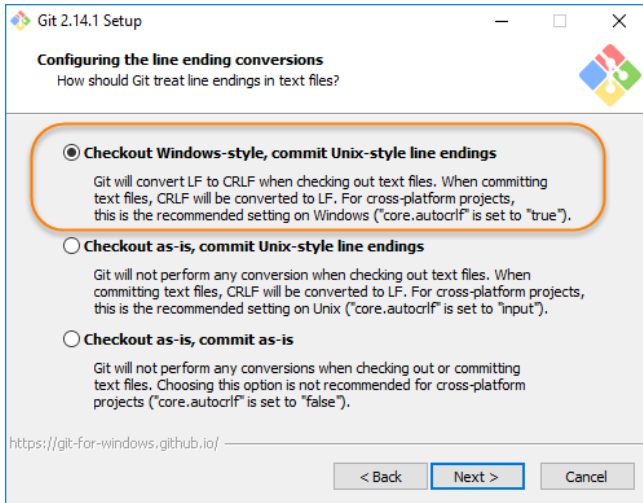

Step 6: Now select the option given below and click on Next button to finish prompts and successfully install Git on your system.
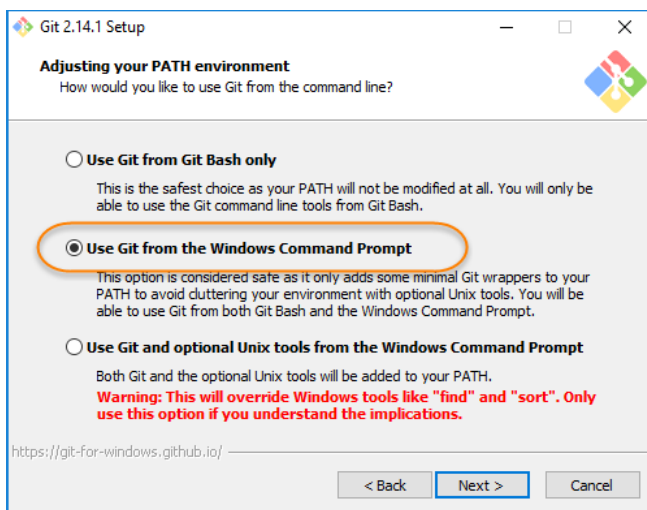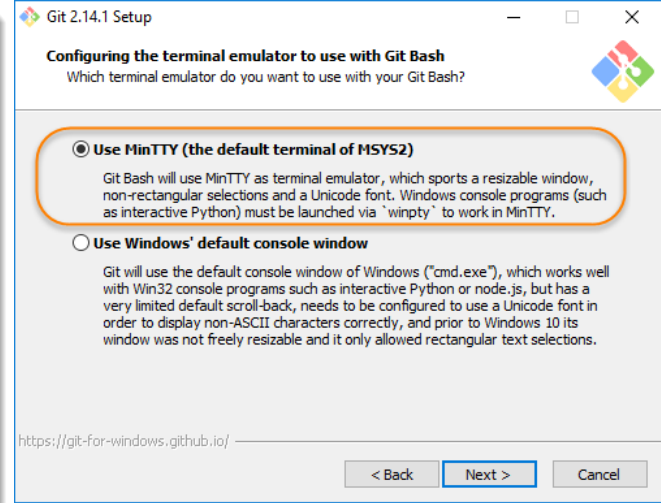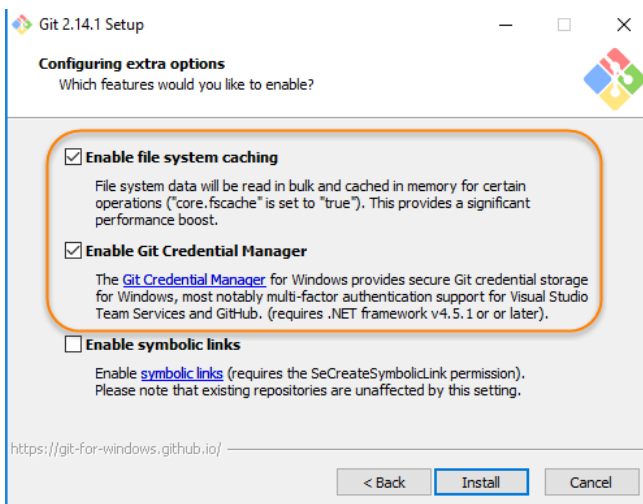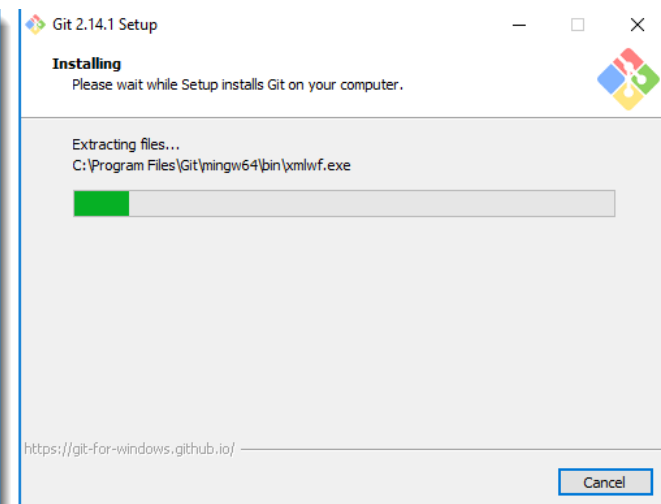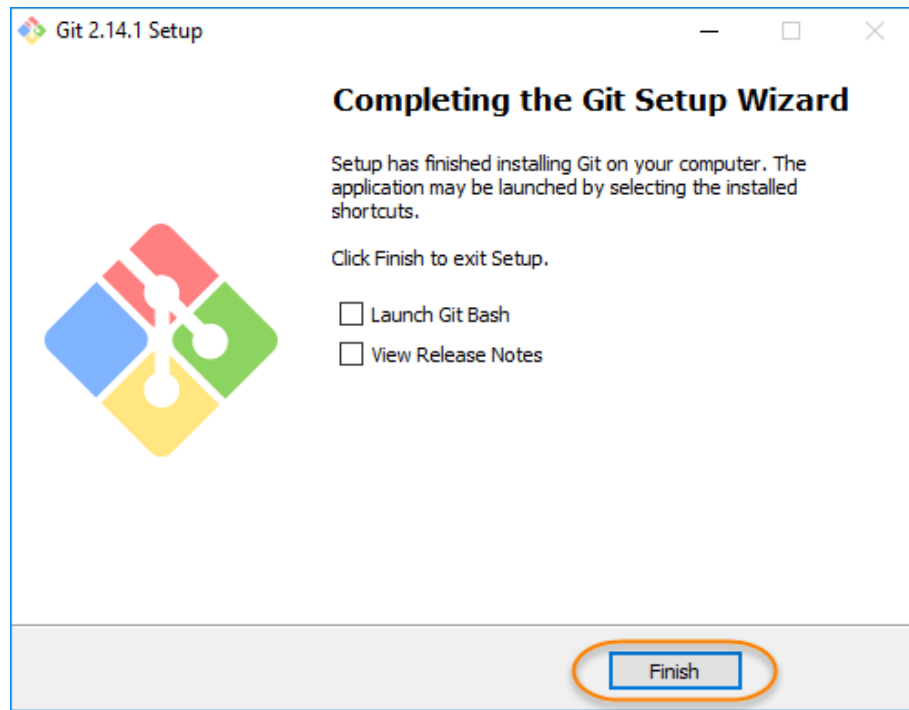

(1)

(2)

(3)

(4)

(5)

(6)

Step 7: This dialogue box will appear at the last. Click on Finish button and Git is successfully installed on your system.

These applications and documents will appear in the choosen folder.

SETTING UP GITHUB ACCOUNT

What is GitHub?

- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.
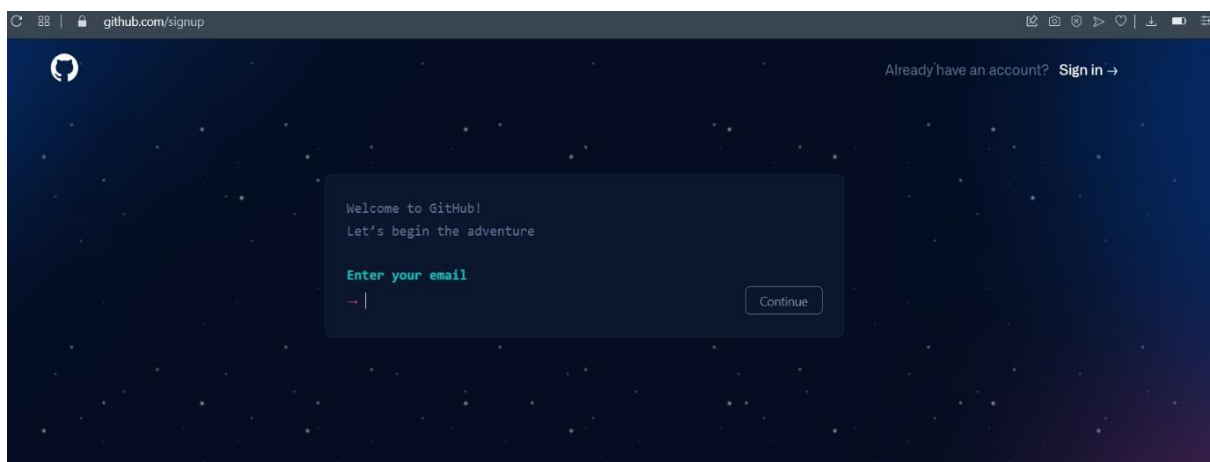
Advantages:

- It makes it easy to contribute to Open-Source projects.
- Track changes in your code across versions.

How to Sign up?
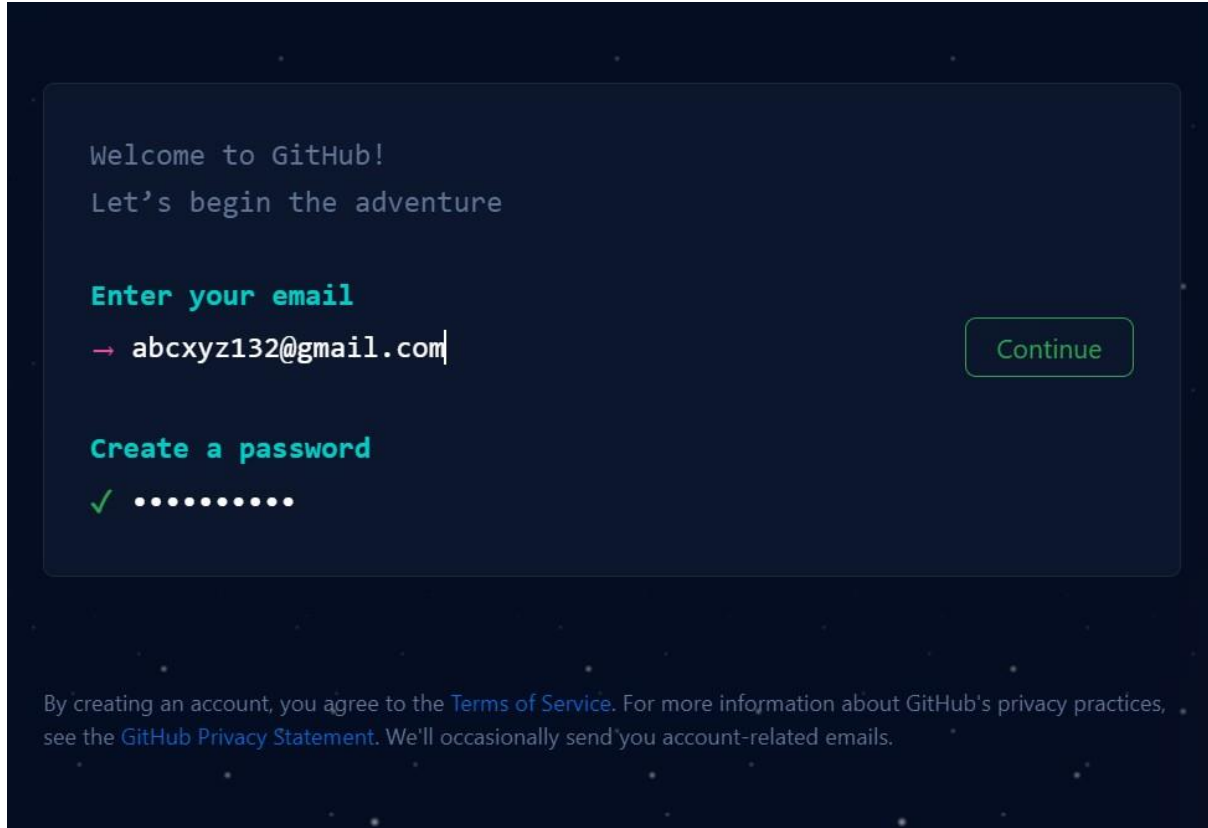
Step 1: In order to create an account on GitHub, visit https://github.com/signup.

Following page appears on the screen.

Step 2: Enter your e-mail and continue.

Step 3: Then create a strong password for your GitHub account.

Welcome to GitHub!
Let's begin the adventure

Enter your email
→ abcxyz132@gmail.com   Continue

Create a password
✓ ●●●●●●●●●

By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails.

Step 4: Then create a username.

Welcome to GitHub!
Let's begin the adventure
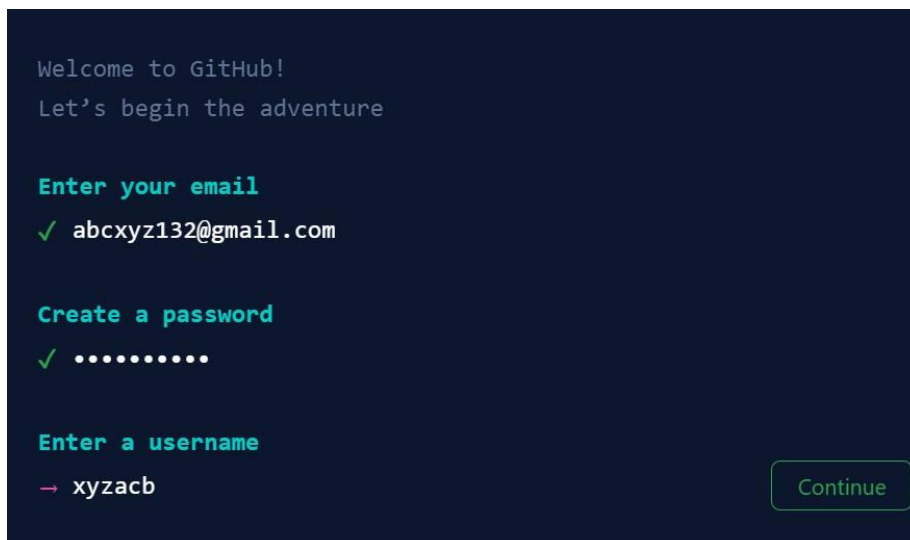
Enter your email
✓ abcxyz132@gmail.com

Create a password
✓ ●●●●●●●●●

Enter a username
→ xyzacb   Continue

Step 5: Fill in all the details that are required and click on create account and your GitHub account would be created

Now you should sign in to your GitHub account.

Step 6: Enter your email-address or username , then the password and continue at https://github.com/signin

Step 7: You have logged in in your account. Now you can create and edit any project.
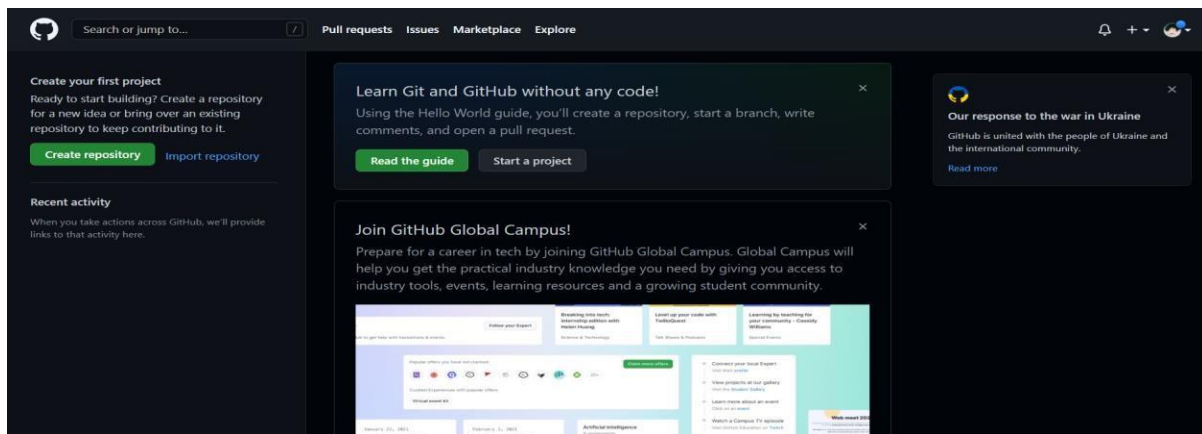
Linking GitHub account with Git Bash:

Username: git config --global user.name "username in github"

Email: git config --global user.email "your email in github"

TASK: 1.1.3
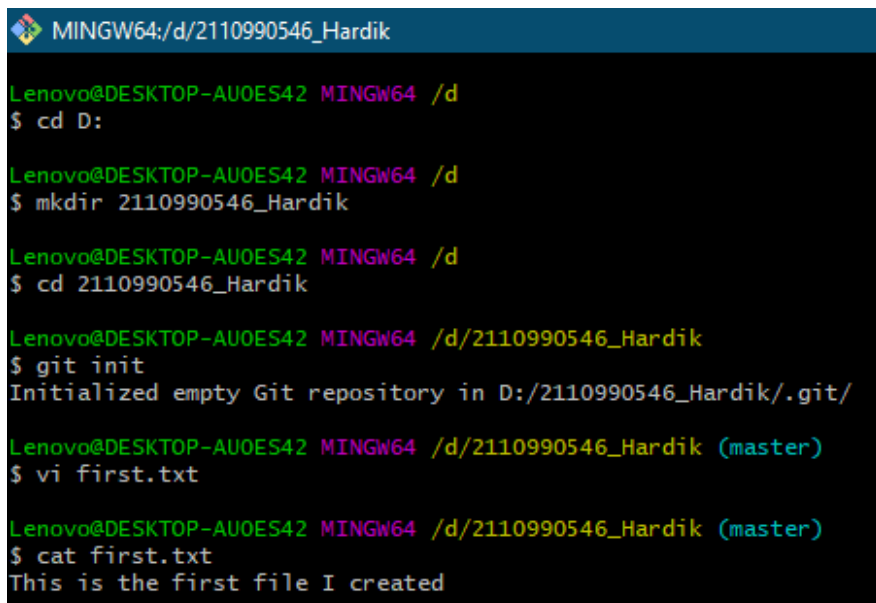
GENERATE LOGS

Logs are nothing but the history which we can see in git . It contains all the past commits, insertions and deletions in it which we can see any time.

Logs helps to check that what were the changes in the code or any other file and by whom. It also contains the number of insertions and deletions including at which time it was changed.

Commands used to generate Logs in Git are:

Step 1: Create a file in the folder.

Step 2: Check Status


```
Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ cat first.txt
This is the first file I created


Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        first.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Step 3: Staging Of File


```
Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git add .
warning: LF will be replaced by CRLF in first.txt.
The file will have its original line endings in your working directory

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git status
On branch master

No commits yet

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

Step 4: Commit the file and check git status

```
Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git commit  -m "First file created"
[master (root-commit) a6a966f] First file created
 1 file changed, 2 insertions(+)
 create mode 100644 first.txt

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Step 5: Now run the command $ git log to generate all the commits in the repository

```
Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git log
commit a6a966f49b9ea2f44931809f18d2a22da02133cb (HEAD -> master)
Author: hardiksawhney24 <hardik0546.be21@chitkara.edu.in>
Date:   Sat Apr 9 22:32:51 2022 +0530

    First file created
```

We can add more files as well or we can also modify the previously existing files.

But we have to stage the modified file again as we do for new files

We can more commits and check them by $ git log command.

```
MINGW64:/d/2110990546_Hardik

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ vi second.txt

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ vi third.txt

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        second.txt
        third.txt

nothing added to commit but untracked files present (use "git add" to track)

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git add second.txt

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git commit -m "Added another file"
[master d0c1255] Added another file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 second.txt

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        third.txt

nothing added to commit but untracked files present (use "git add" to track)

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git add third.txt

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git commit -m "added third file"
[master 7baee7b] added third file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 third.txt
```

```
Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ vi first.txt

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   first.txt

no changes added to commit (use "git add" and/or "git commit -a")

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git add first.txt
warning: LF will be replaced by CRLF in first.txt.
The file will have its original line endings in your working directory

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git commit -m "Modified first file"
[master 4771044] Modified first file
 1 file changed, 1 insertion(+)

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Now we can run $git log command to check all the commits in this repository.

The first commit shown is the most recent commit.

```
Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git log
commit 477104449d694b7feeef8d17070084359b6a752b (HEAD -> master)
Author: hardiksawhney24 <hardik0546.be21@chitkara.edu.in>
Date:    Sat Apr 9 22:45:19 2022 +0530

    Modified first file

commit 7baee7baf7148c6881d23d6f5a4c32d9ce22a4a4
Author: hardiksawhney24 <hardik0546.be21@chitkara.edu.in>
Date:    Sat Apr 9 22:43:57 2022 +0530

    added third file

commit d0c125549cc605411aa94685f77d79fe1efcc9f4
Author: hardiksawhney24 <hardik0546.be21@chitkara.edu.in>
Date:    Sat Apr 9 22:43:25 2022 +0530

    Added another file

commit a6a966f49b9ea2f44931809f18d2a22da02133cb
Author: hardiksawhney24 <hardik0546.be21@chitkara.edu.in>
Date:    Sat Apr 9 22:32:51 2022 +0530

    First file created
```

✓ As it can be observed, on using this command, the system displays all the changes made in the file or list of all the commits in the history along with the information of the user.

✓ **This commands clearly defines the git as the 'version-controlled system' as it allows us to rollback to any of the previous working states and keeps track of all the versions.**
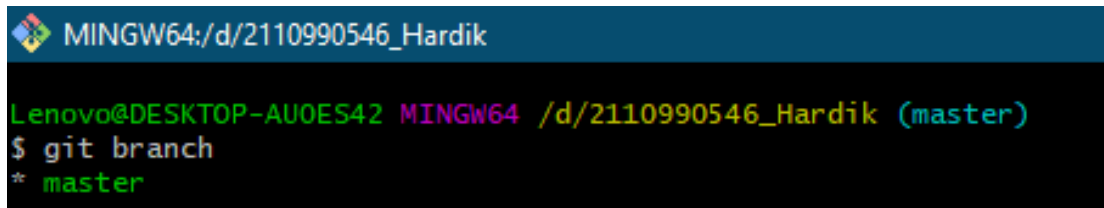
TASK: 1.1.4

CREATE AND VISUALIZE BRANCHES

The main branch in git is called as 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 file which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches).

FOLLOW THESE STEPS TO CREATE A SEPARATE BRANCH IN GIT.

1. Firstly, you can check the present branches in the master branch with the help of this command.

$ git branch

2. To create a new branch:

$ git branch <name of branch you want to create>

3. To move to the new branch created

$ git checkout <name of branch you want to move>

```
Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ git checkout new
Switched to branch 'new'
```

4. Create new file and stage them

```
Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (new)
$ git add fourth.txt

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (new)
$ git commit -m "fourth file"
[new 226779a] fourth file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 fourth.txt

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (new)
$ git status
On branch new
nothing to commit, working tree clean
```

5. Move to another branch and you can't see the additions of current branch in that

```
Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (new)
$ git checkout master
Switched to branch 'master'

Lenovo@DESKTOP-AUOES42 MINGW64 /d/2110990546_Hardik (master)
$ ls
first.txt  second.txt  third.txt
```

TASK: 1.1.5

<u>GET LIFECYCLE DESCRIPTION</u>

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**:**

1. **Working directory**
2. **Staging area**
3. **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. Working directory is the directory containing hidden .git folder.

Working directory is the directory containing hidden .git folder.

**git init** - *Command to initialize a Git repository*
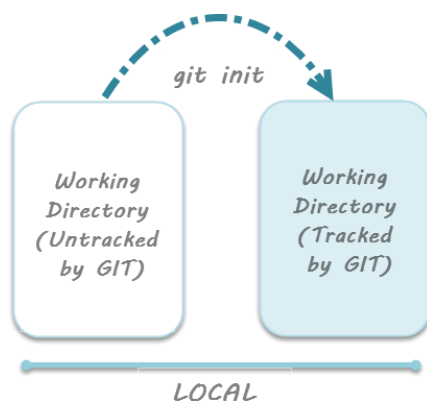
Staging area:

Some files in the project like class files, log files, result files and temporary data files are dynamically generated. It doesn't make sense to track the versions of these files.

Whereas the source code files, data files, configuration files and other project artifacts contain the business logic of the application. These files are to be tracked by Git are thus needs to be added to the staging area.

git add - *Command to add files to staging area.*
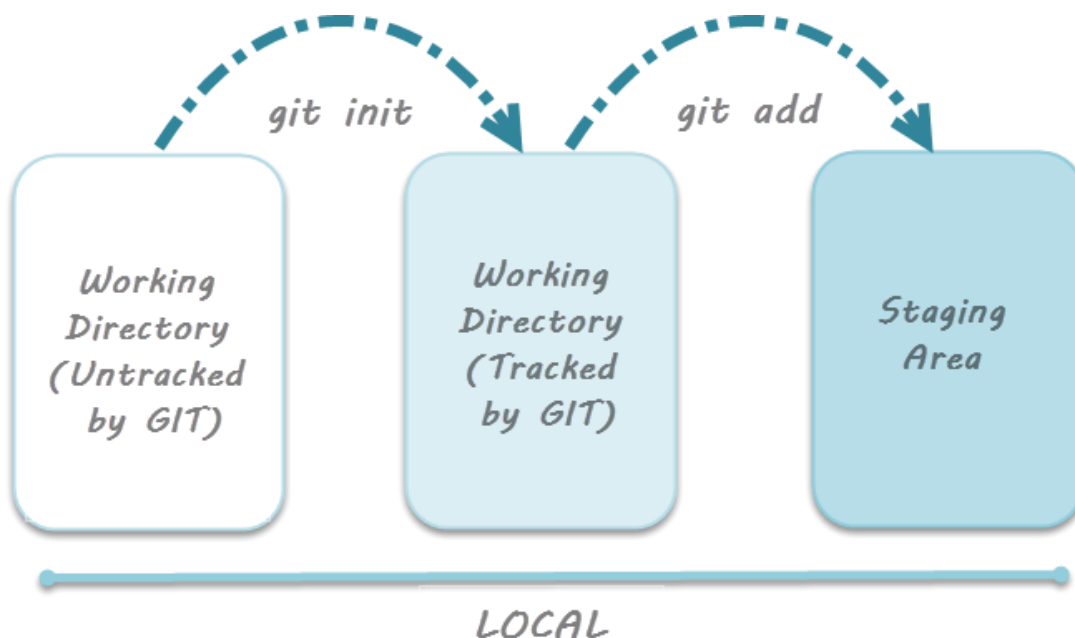
Git Directory:

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.

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.

Thus, Git directory is the database where metadata about project files' history will be tracked.

git commit -m"your message" - Command to commit files to Git repository with message.