TASK 1.1 REPORT FILE

Subject:- SCM

Submitted to:- Dr. Monit Kapoor

Submitted By:-

Name: Aakash Jha
Roll:-2110990005
Group:- G01-A
Cluster:- Beta

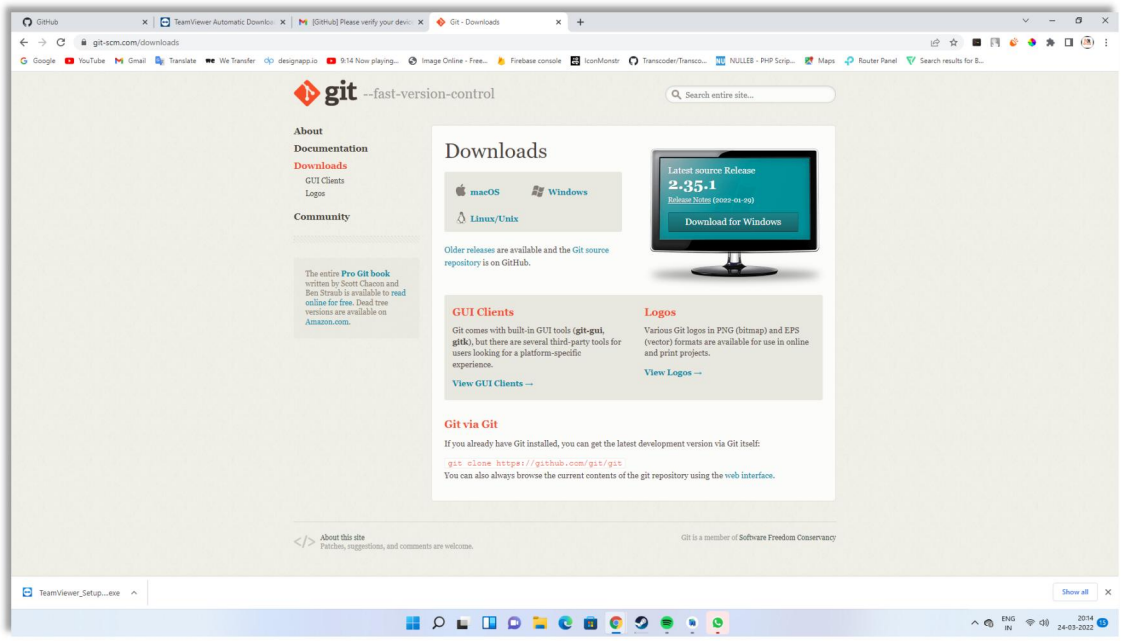
INDEX

Task 1.1

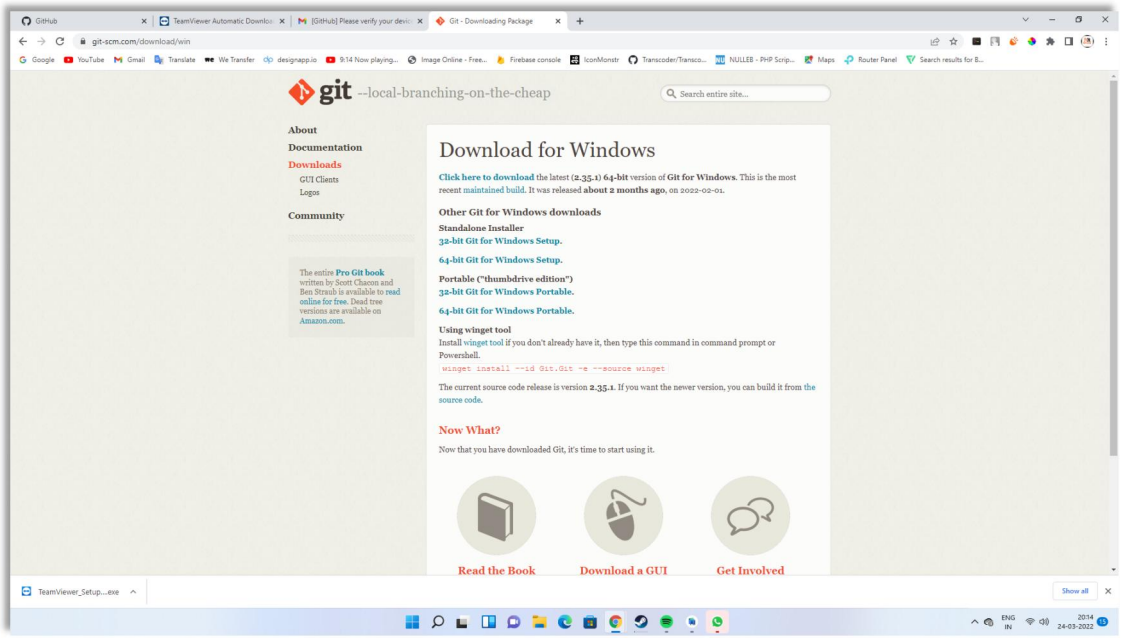
Exp. 01

Aim: Setting up of Git Client

✧ For git installation on your system, go to the linked URL.
https://git-scm.com/downloads



◆ You must first access this webpage and then choose your operating system by clicking on it. I'll walk you through the processes for the Windows operating system in this article.

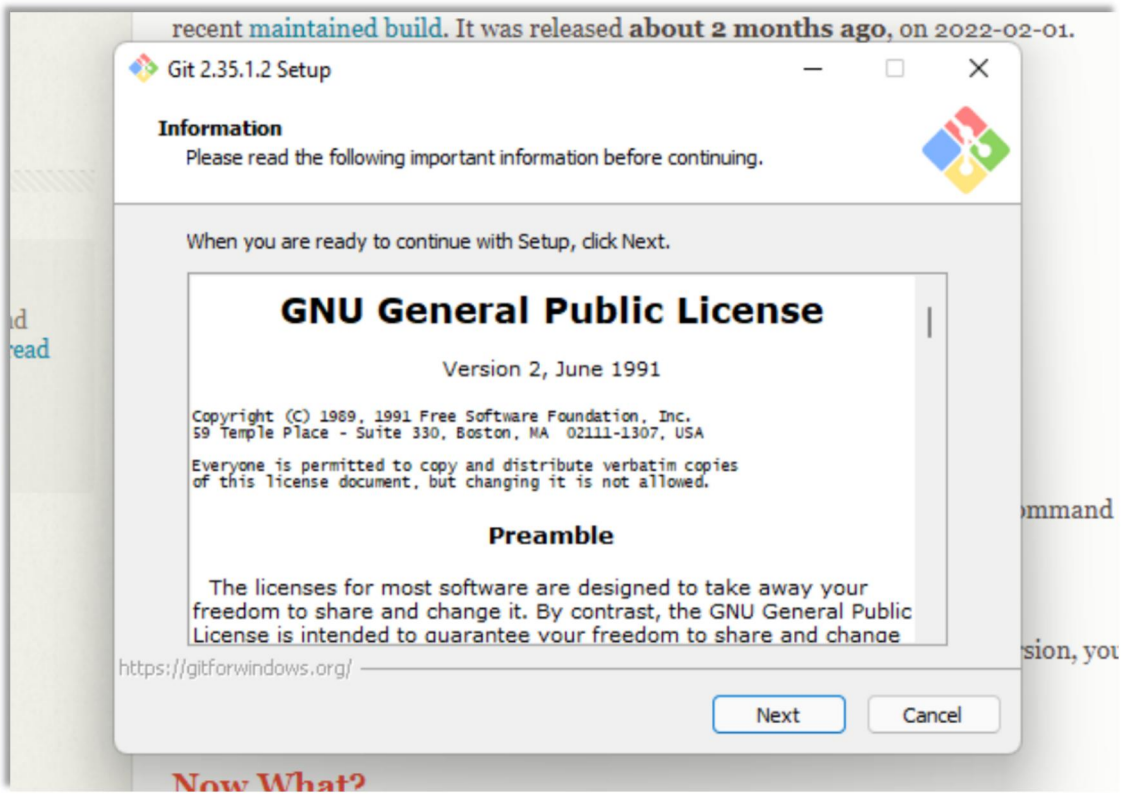


✧ Select the CPU for your system now. (Most of the system now runs on 64-bit processors.) Your download will begin when you pick a processor.

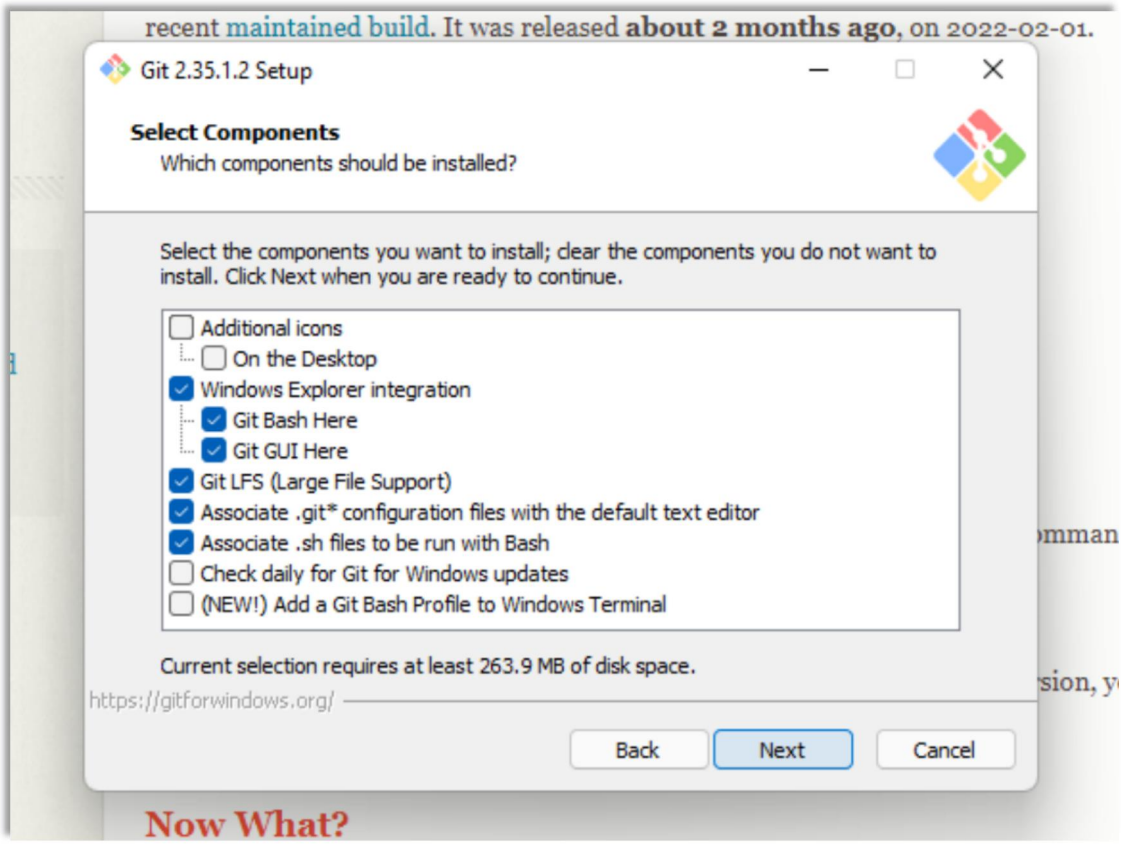
◆ You must now open the Git folder.

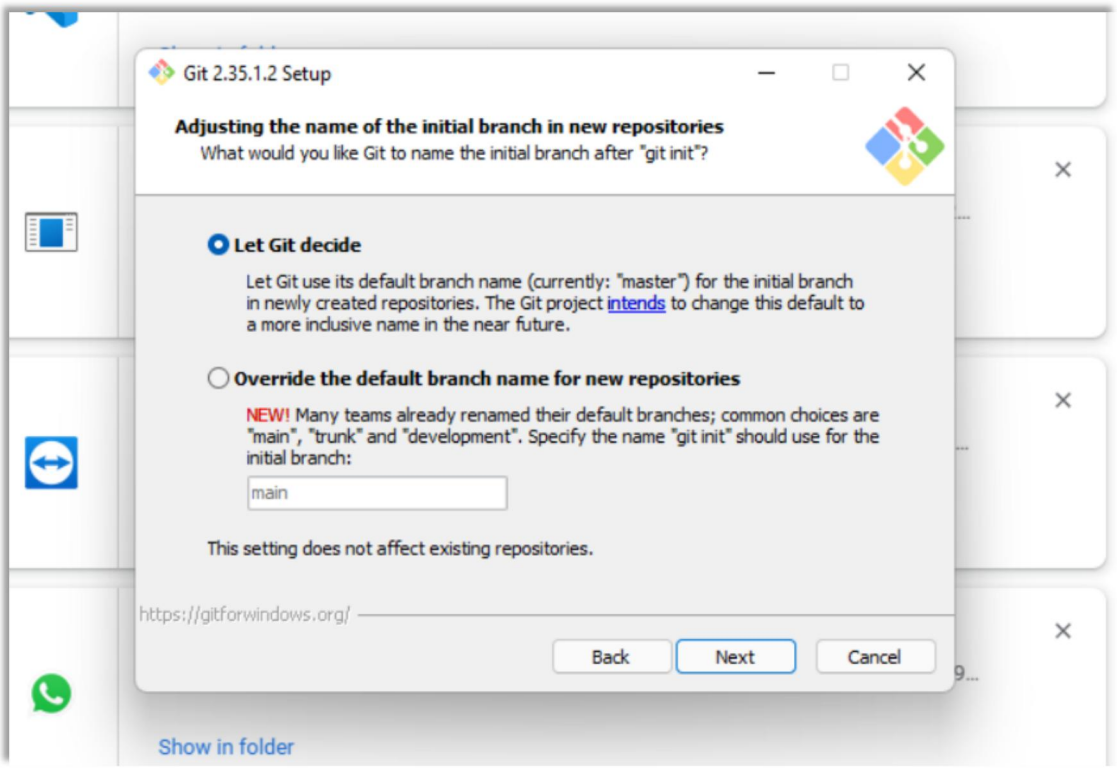
◆ You will be asked if you want to enable this programe to make modifications to your PC once you launch it.

◆ YES should be selected.

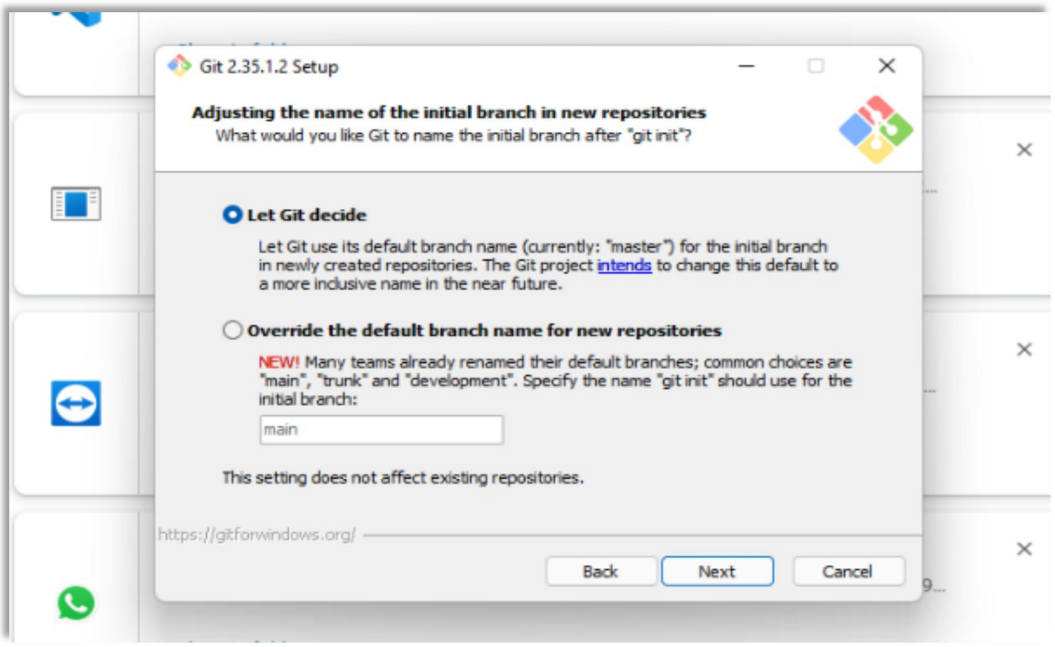


- Click on Next

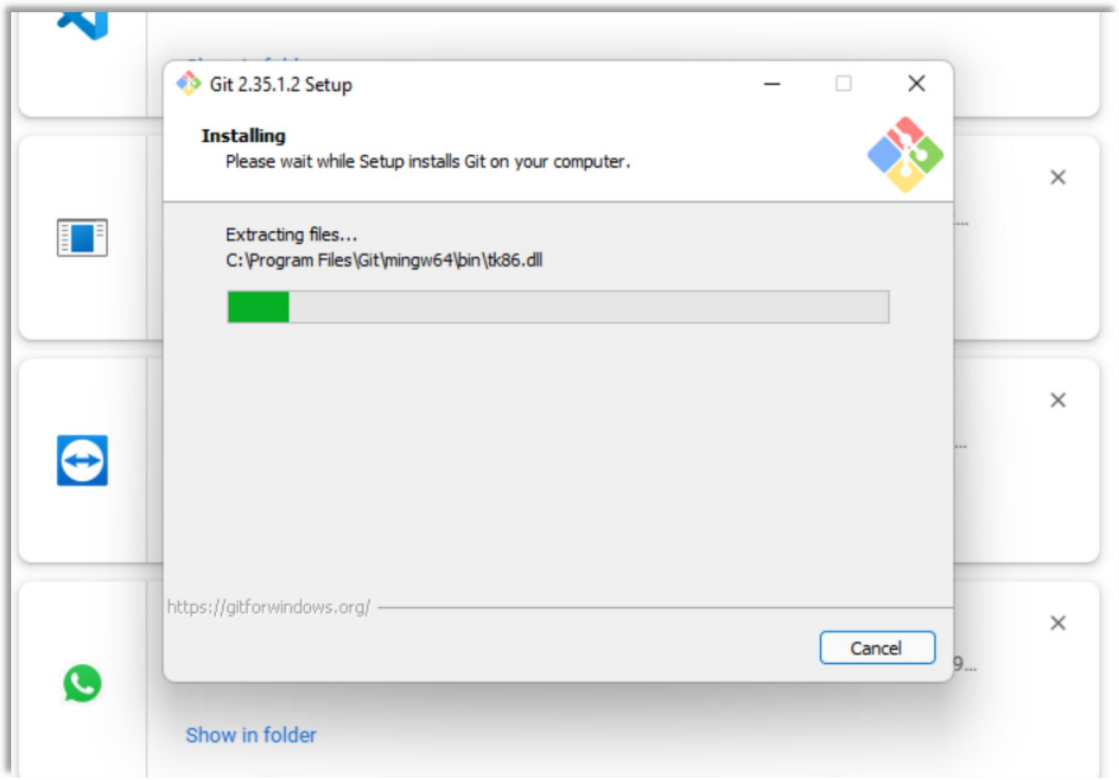




◆ Continue clicking on next few times more



◆ Now select the Install option.



◆ Click on Finish after the installation is finished.

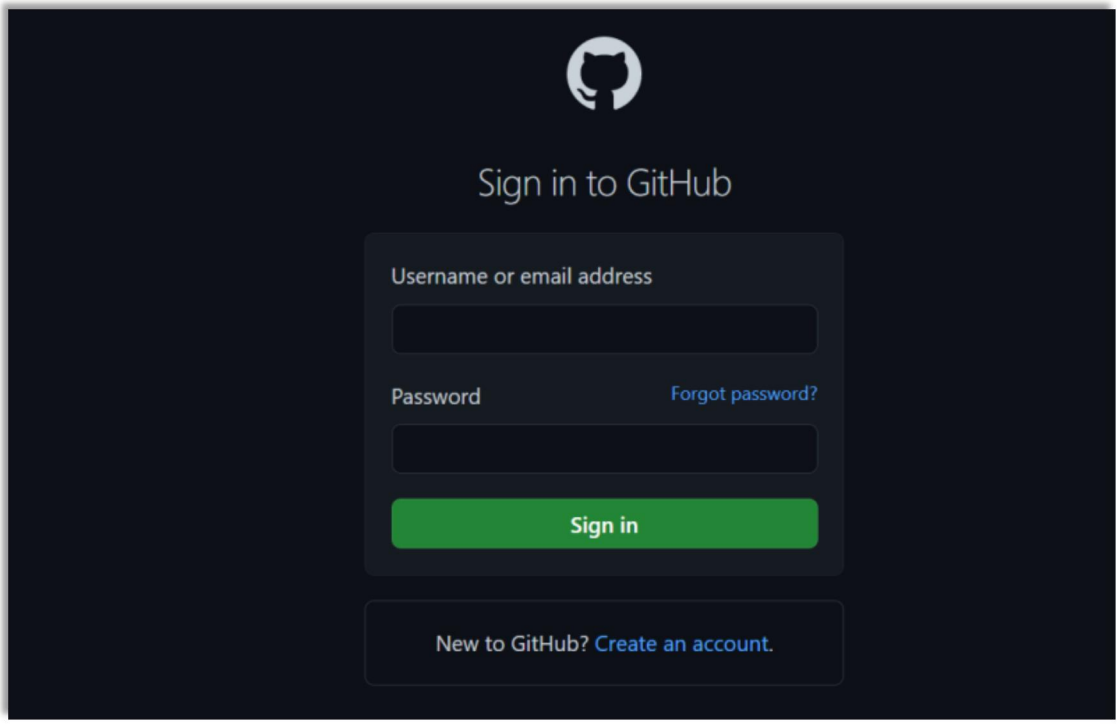
The installation of the git is finished and now we have to setup git client and GitHub account.

Exp. 02

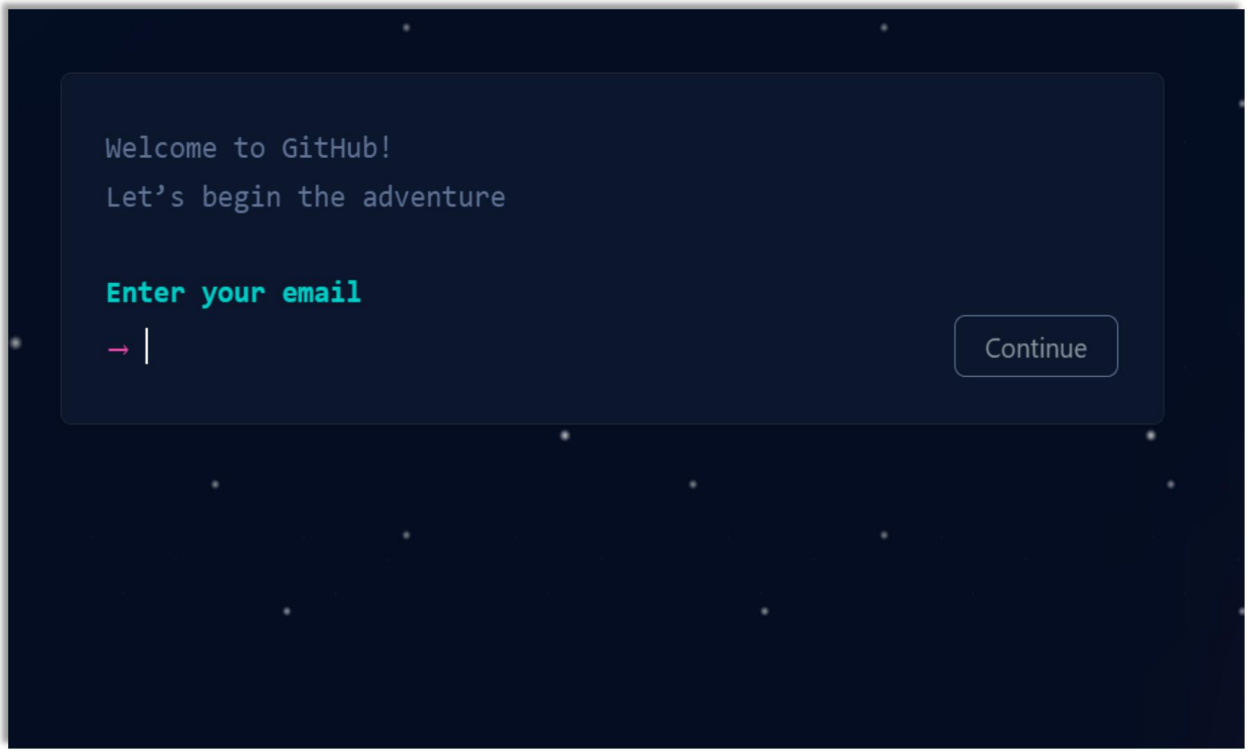
Aim: Setting up GitHub Account

- Open your web browser search GitHub login.

- Click on Create an account if you are a new user or if you have already an account, please login.



- After clicking on "Create a New Account," you will be sent to a new page where you must enter your email address for your account. Now type in the password you'd want to use for your GitHub account. Then you'll be prompted to enter your username.

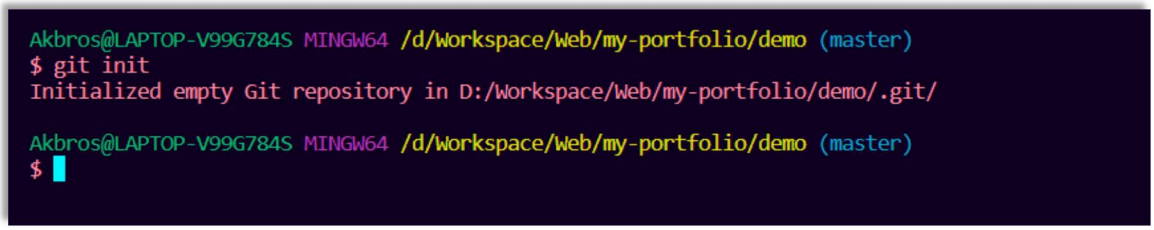


- You will be asked some questions just keep the defaults and continue.
- Now Click on Create Account.
- Verify it from your email and you are all set to go.

Exp. 03

Aim: Program to Generate logs

✧ First of all create a local repository using Git. For this, you have to make a folder in your device, right click and select "**Git Bash Here**". This opens the Git terminal. To create a new local repository, use the command "**git init**" and it creates a folder **.git**.



✧ When we use GIT for the first time, we have to give the user name and email so that if I am going to change in project, it will be visible to all.

For this, we use command →

"git config --global user.name *Name*"
"git config --global user.email *email*"

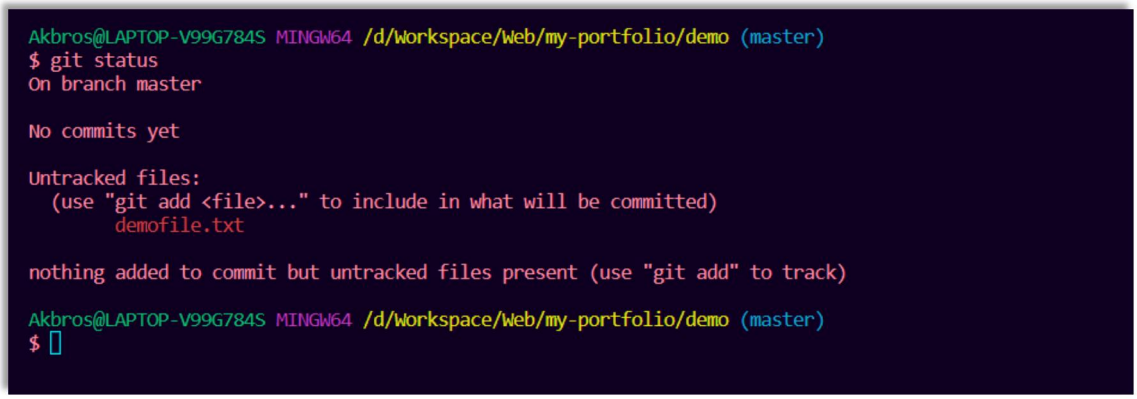
For verifying the user's name and email, we use →

"git config --global user.name"
"git config --global user.email"

Some Important Commands:

- **ls →** It gives the file names in the folder.
- **ls -lart →** Gives the hidden files also.
- **git status →** Displays the state of the working directory and the staged snapshot.
- **touch filename →** This command creates a new file in the repository.
- **Clear →** It clears the terminal.
- **rm -rf .git →** It removes the repository.
- **git log →** displays all of the commits in a repository's history
- **git diff →** It compares my working tree to staging area.

Now, we have to create some files in the repository. Suppose we created demofile.txt Now type git status:

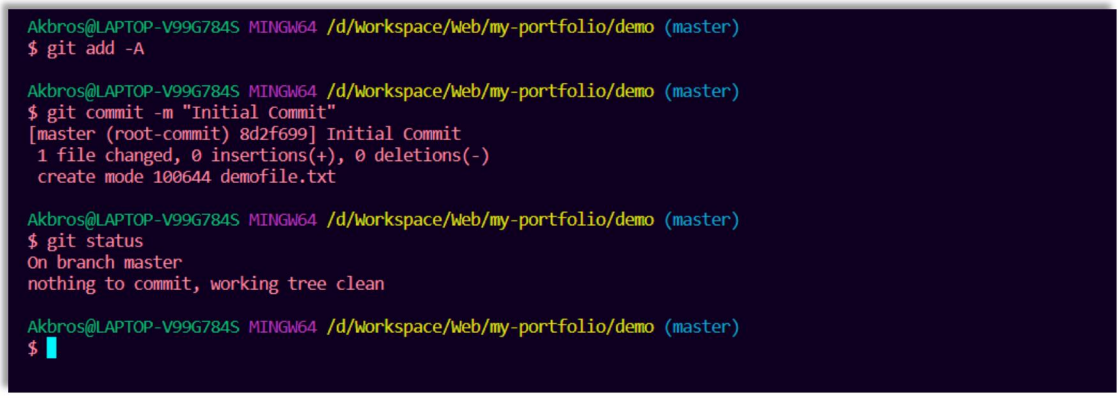


You can see that demofile.*txt* is in red colour that means it is an untracked file.

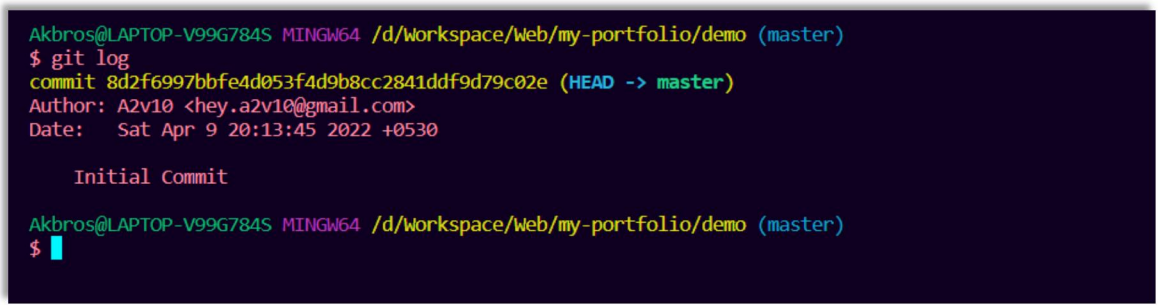
Now firstly add the file in staging area and then commit the file.

For this, use command →

git add -A [For add all the files in staging area.]
git commit -m "write any message" [For commit the file]



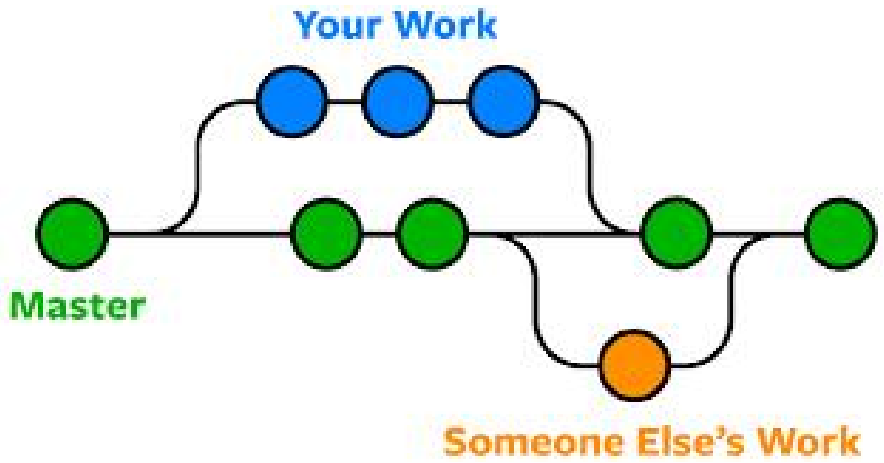
- **git log:** *The git log command displays a record of the commits in a Git repository. By default, the git log command displays a commit hash, the commit message, and other commit metadata.*



Exp.04

Aim: Create and visualize branches

- **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.



Let us see the command of it:
Firstly, add a new branch, let us suppose the branch name is branch1.
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]

```
Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (master)
$ git branch branch2

Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (master)
$ git branch
  branch1
  branch2
* master

Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (master)
$ git checkout branch1
Switched to branch 'branch1'

Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (branch1)
$
```

In this you can see that firstly ‘git branch’ shows only one branch in green color but when we add a new branch using ‘git branch branch1’, it shows 2 branches but the green colour and star is on master. So, we have to switch to branch1 by using ‘git checkout branch1’. If we use ‘git branch’, now you can see that the green colour and star is on branch1. It means you are in branch1 branch and all the data of master branch is also on branch1 branch. Use “ls” to see the files.

Now add a new file in branch1 branch, do some changes in file and commit the file.

```
Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (branch1)
$ touch demo2.txt

Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (branch1)
$ git add -A

Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (branch1)
$ git commit -m "Demo Commit"
[branch1 f22bd1a] Demo Commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 demo2.txt

Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (branch1)
$
```

If we switched to master branch, ‘demo.txt’ file is not there. But he file is in branch1 branch.

```
Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (branch1)
$ ls
demo2.txt  demofile.txt

Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (branch1)
$ git checkout master
Switched to branch 'master'

Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (master)
$ LS
demofile.txt

Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (master)
$
```

- To add these files in master branch, we have to do merging. For this firstly switch to master branch and then use command →

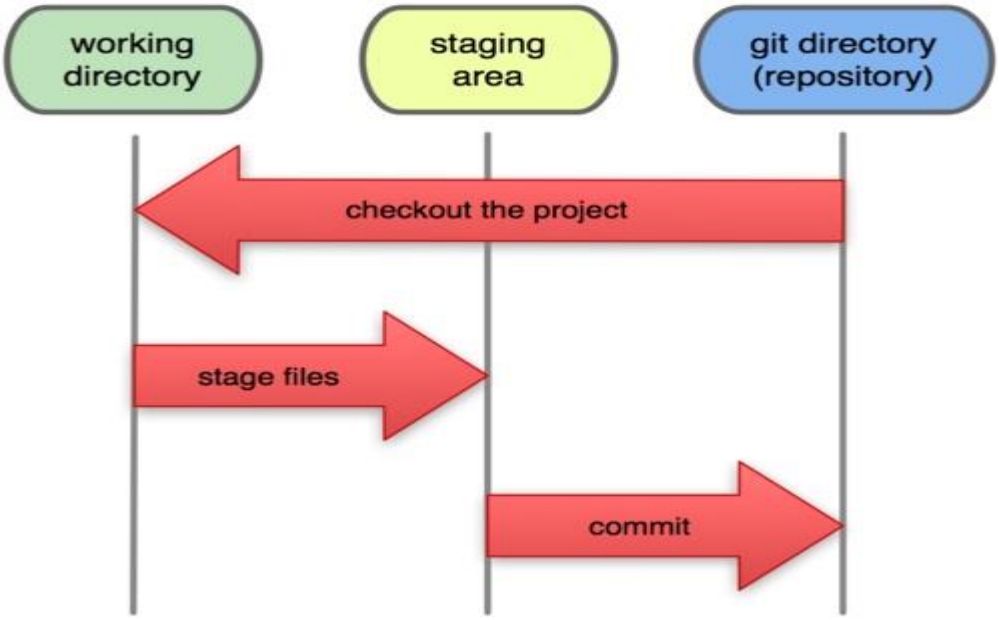
  git merge branchname [use to merge branch]

```
Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (master)
$ git merge branch1
Updating 8d2f699..f22bd1a
Fast-forward
 demo2.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 demo2.txt

Akbros@LAPTOP-V99G784S MINGW64 /d/Workspace/Web/my-portfolio/demo (master)
$ git log
commit f22bd1a15c0a896a1a761c71df3007d429d821e2 (HEAD -> master, branch1)
Author: A2v10 <hey.a2v10@gmail.com>
Date:   Sat Apr 9 20:47:18 2022 +0530

    Demo Commit

commit 8d2f6997bbfe4d053f4d9b8cc2841ddf9d79c02e (branch2)
Author: A2v10 <hey.a2v10@gmail.com>
Date:   Sat Apr 9 20:13:45 2022 +0530

    Initial Commit
```

Exp. 05

Aim: Git lifecycle description

Stages in GIT Life Cycle:
Now let's understand the three-stage architecture of Git:



- **Working Directory:** This is the directory that we've initialized, and here all the changes are made to commit on GitHub.

- **Staging Area:** This is where we first put out code or files of the working repository. The command that we use to stage code is, "git add --a", "git add FileName" or "git add -A". In simple terms, staging means telling Git what files we want to commit (new untracked files, modified files, or deleted files).

- **Git directory(repository):** This is where all the commits are stored whenever we make a commit. We can revert to an older version of or project using the "git checkout" command from this directory.