

Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **DCSE**

CHITKARA
UNIVERSITY



Submitted By:

Anmol Dhiman

2110990210

G3

Submitted To:

Dr. Deepak Thakur

List of Experiments

S. No	Aim	Page No.
1	Setting up of Git Client	1
2	Setting up GitHub Account	2-4
3	Program to Generate log	5
4	Create and visualize branches	6-8
5	Git lifecycle description	9-12

Aim: Setting up of Git Client

Theory:

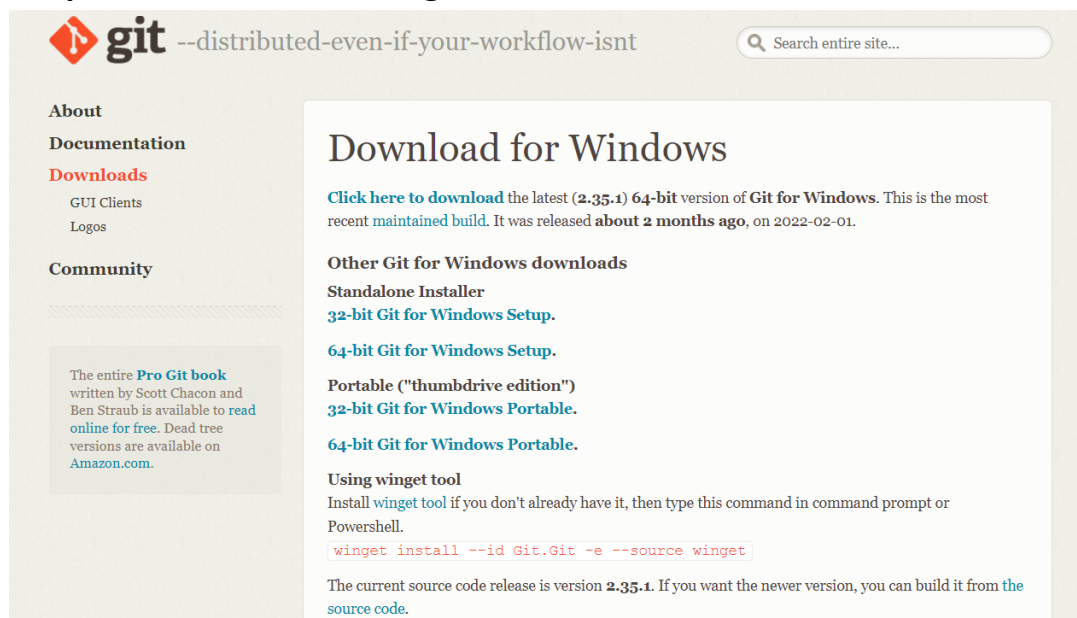
What is Git?

So, what is Git in a nutshell? This is an important section to absorb, because if you understand what Git is and the fundamentals of how it works, then using Git effectively will probably be much easier for you. As you learn Git, try to clear your mind of the things you may know about other VCSs, such as CVS, Subversion or Perforce — doing so will help you avoid subtle confusion when using the tool. Even though Git's user interface is fairly similar to these other VCSs, Git stores and thinks about information in a very different way, and understanding these differences will help you avoid becoming confused while using it.

Procedure to Install Git Client:

1. Browse to the official Git website: <https://git-scm.com/downloads>
2. Click the download link for Windows and allow the download to complete.
3. Browse to the download location (or use the download shortcut in your browser). Double-click the file to launch the installer.
4. Install it like a regular program.
5. Once the installation is complete, tick the boxes to Launch Git Bash, then click **Finish**.

Snapshots of Download Page:



Aim: Setting up GitHub Account

Theory:

What is GitHub?

GitHub is a code hosting platform for collaboration and version control. It lets you (and others) work together on projects.

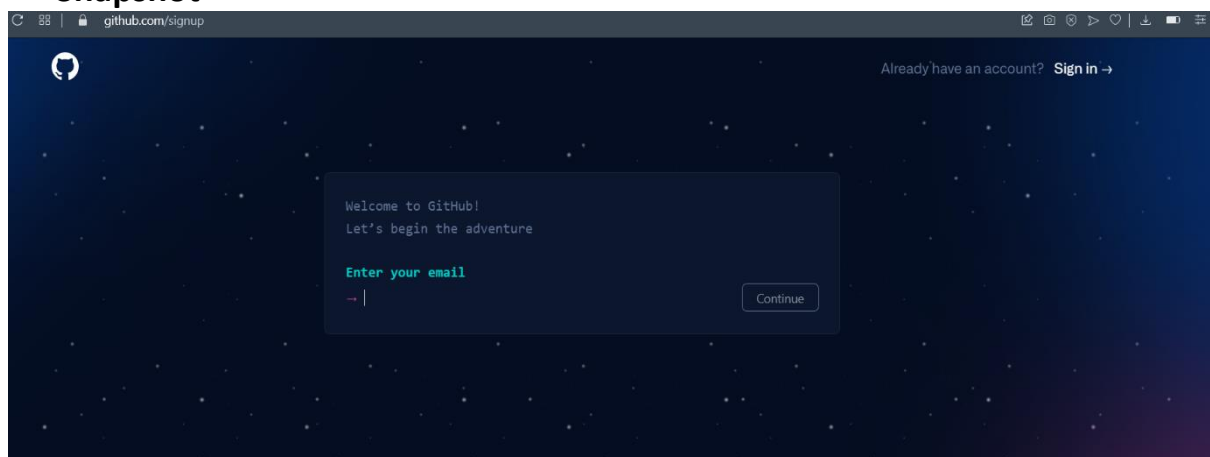
Advantages of GitHub?

- Version control your projects.
- Push your projects to GitHub and let the world know how nice and useful code you write.
- Explore other's projects on GitHub, get inspired code more, or contribute to their project.
- Collaborate with others, by letting other people contribute to your projects or you contributing to other's projects.
- Maintain useful lists. I didn't find anything better then creating lists on GitHub using markdown. I like how clean it looks.
- Learn Git.
- Link your nice GitHub profile while applying for grants, jobs, in your resume. It makes things better and helps.

Procedure:

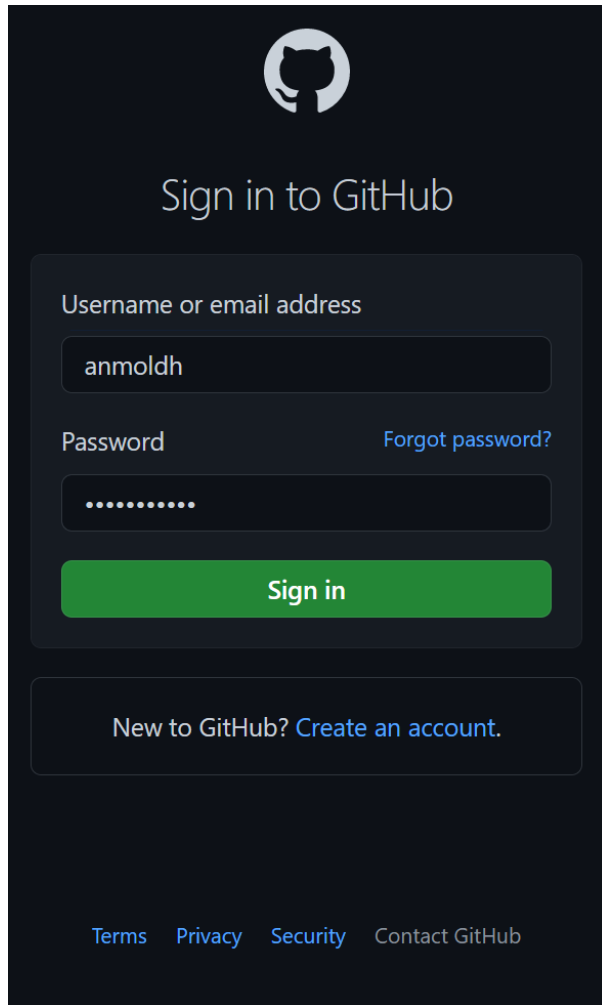
- Visit <https://github.com/>, Click Signup, if you don't have an account.

Snapshot



After clicking signup this kind of interface will appear, if you already have an account continue to login.

Signing into GitHub Account:



The image shows the GitHub sign-in page. At the top is the GitHub logo. Below it, the text "Sign in to GitHub" is centered. There are two input fields: "Username or email address" with the value "anmol dh" and "Password" with masked characters. A link "Forgot password?" is next to the password field. A green "Sign in" button is below the fields. At the bottom, there is a link "New to GitHub? Create an account." and a footer with links "Terms", "Privacy", "Security", and "Contact GitHub".

Sign in to GitHub

Username or email address

anmol dh

Password

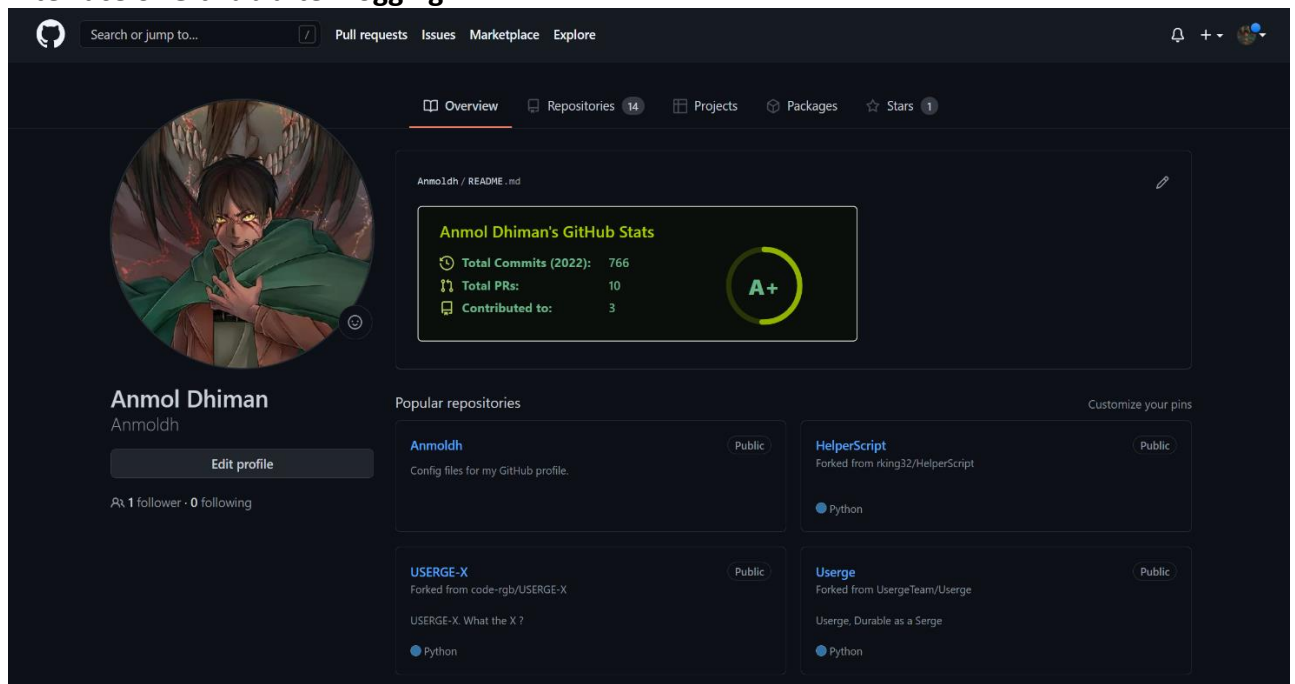
[Forgot password?](#)

Sign in

New to GitHub? [Create an account.](#)

[Terms](#) [Privacy](#) [Security](#) [Contact GitHub](#)

Interface of GitHub after Logging-in:



The image shows the GitHub user profile page for Anmol Dhiman. The page has a dark theme. At the top, there is a search bar and navigation links: "Pull requests", "Issues", "Marketplace", and "Explore". The user's profile picture is a circular avatar of a person with wings. Below the profile picture, the name "Anmol Dhiman" and the username "Anmol dh" are displayed. There is an "Edit profile" button. The profile statistics show "1 follower" and "0 following". The "Overview" tab is selected, showing "Anmol Dhiman's GitHub Stats":

Anmol Dhiman's GitHub Stats	
Total Commits (2022):	766
Total PRs:	10
Contributed to:	3

A green circular badge with "A+" is next to the stats. Below the stats, there is a section for "Popular repositories". The repositories listed are:

- Anmol dh** (Public): Config files for my GitHub profile.
- USERGE-X** (Public): Forked from code-rgb/USERGE-X. USERGE-X. What the X ?
- HelperScript** (Public): Forked from rking32/HelperScript.
- Userge** (Public): Forked from UsergeTeam/Userge. Userge, Durable as a Serge.

Each repository has a "Python" icon. There is also a "Customize your pins" link.

Linking GitHub account with Git Bash:

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

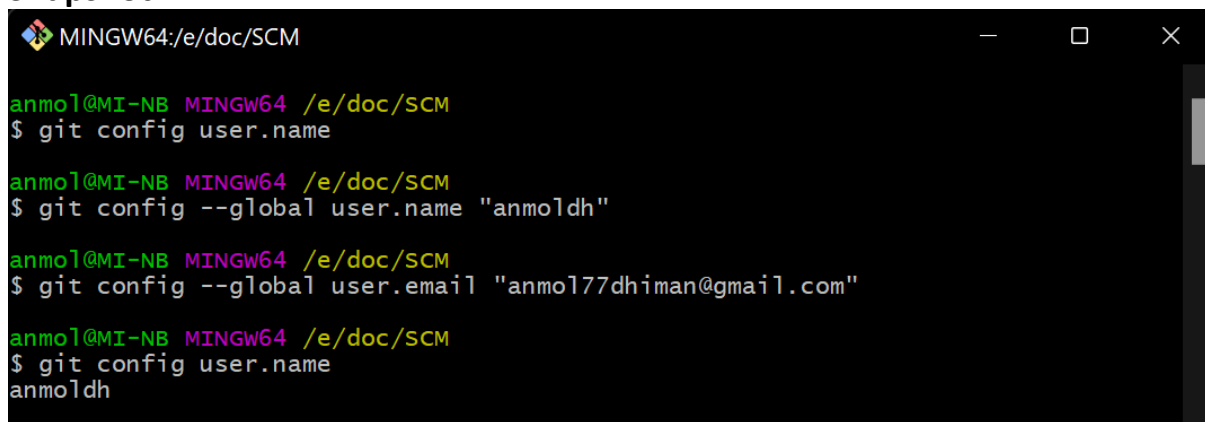
Email: git config --global user.email "your email in github"

Check Username & Email:

git config user.name

git config user.email

Snapshot:

A screenshot of a Windows terminal window titled "MINGW64:/e/doc/SCM". The terminal shows a series of commands and their outputs. The prompt is "anmol@MI-NB MINGW64 /e/doc/SCM". The first command is "\$ git config user.name", which returns "anmol". The second command is "\$ git config --global user.name 'anmol'". The third command is "\$ git config --global user.email 'anmol77dhiman@gmail.com'". The fourth command is "\$ git config user.name", which returns "anmol".

```
MINGW64:/e/doc/SCM
anmol@MI-NB MINGW64 /e/doc/SCM
$ git config user.name
anmol
anmol@MI-NB MINGW64 /e/doc/SCM
$ git config --global user.name "anmol"
anmol@MI-NB MINGW64 /e/doc/SCM
$ git config --global user.email "anmol77dhiman@gmail.com"
anmol@MI-NB MINGW64 /e/doc/SCM
$ git config user.name
anmol
```

Aim: Program to Generate log

Theory:

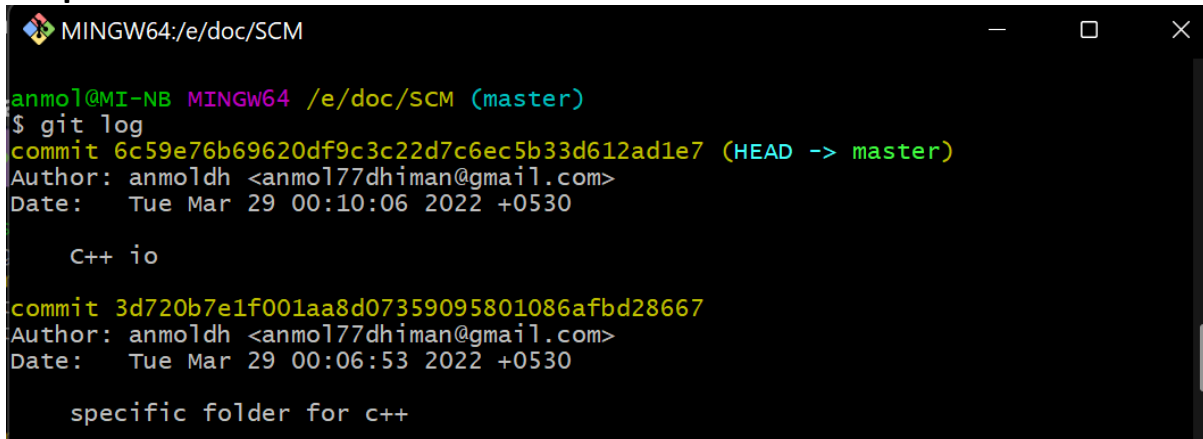
Git Logs:

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.

How Git Logs are useful?

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.

Snapshot:



```
MINGW64:/e/doc/SCM

anmol@MI-NB MINGW64 /e/doc/SCM (master)
$ git log
commit 6c59e76b69620df9c3c22d7c6ec5b33d612ad1e7 (HEAD -> master)
Author: anmol dh <anmol77dhiman@gmail.com>
Date: Tue Mar 29 00:10:06 2022 +0530

    C++ io

commit 3d720b7e1f001aa8d07359095801086afbd28667
Author: anmol dh <anmol77dhiman@gmail.com>
Date: Tue Mar 29 00:06:53 2022 +0530

    specific folder for c++
```

Aim: Create and visualize branches

Branches in Git:

In Git, branches are a part of your everyday development process. Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes.

Syntax:

- 1. To create a new Branch:** git branch "branch-name"

```
MINGW64:/e/doc/SCM

anmol@MI-NB MINGW64 /e/doc/SCM (master)
$ git branch
* master

anmol@MI-NB MINGW64 /e/doc/SCM (master)
$ git branch "sample"

anmol@MI-NB MINGW64 /e/doc/SCM (master)
$ git checkout "sample"
Switched to branch 'sample'
```

- 2. To check numbers of branches, We've:** git branch

```
MINGW64:/c/Users/anmol/Downloads/SCM

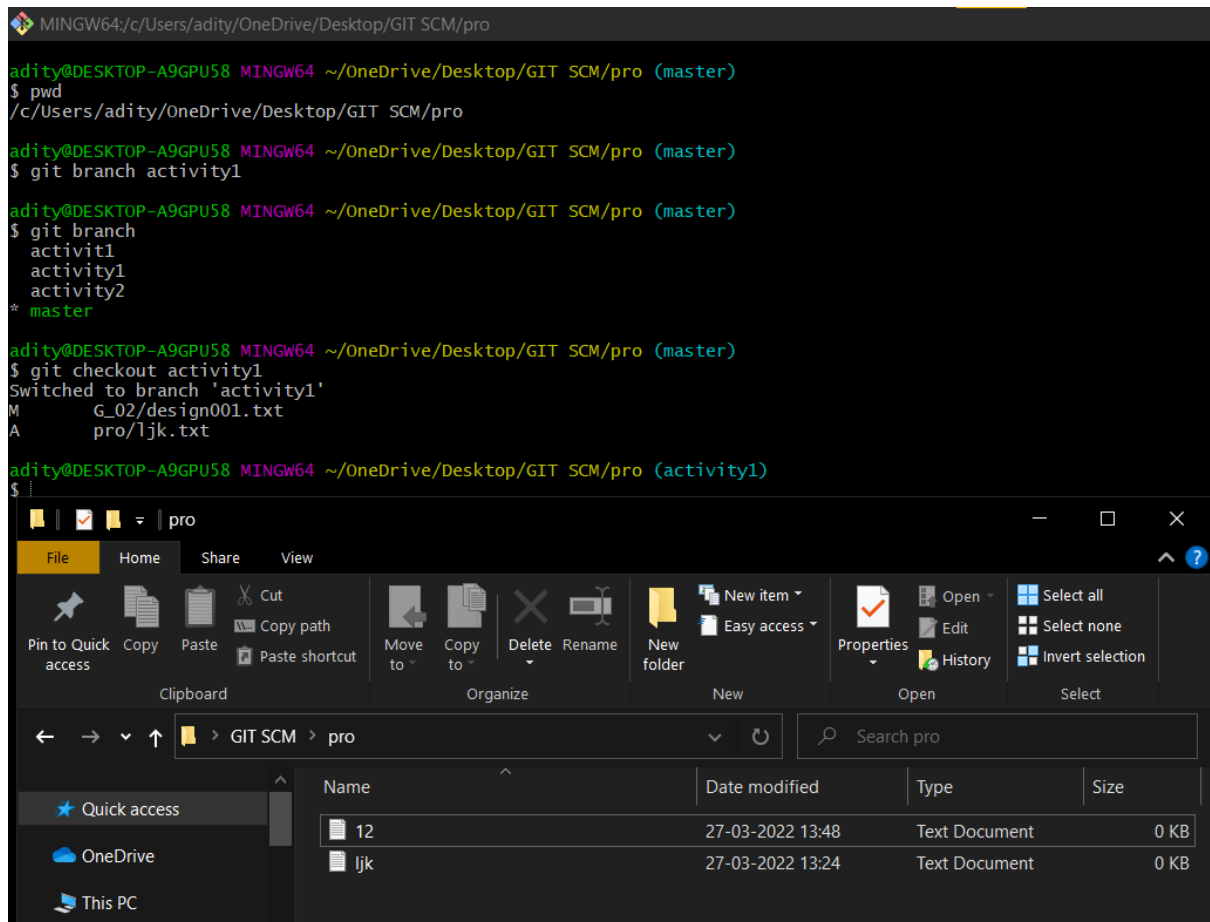
anmol@MI-NB MINGW64 ~/Downloads/SCM (sample)
$ git branch
master
* sample
```

- 3. To change current working branch:** git checkout "branch-name"

```
anmol@MI-NB MINGW64 /e/doc/SCM (master)
$ git checkout "sample"
Switched to branch 'sample'
```


Visualizing branches:

To visualize I have created a new file in a new branch activity 1 instead of master branch.

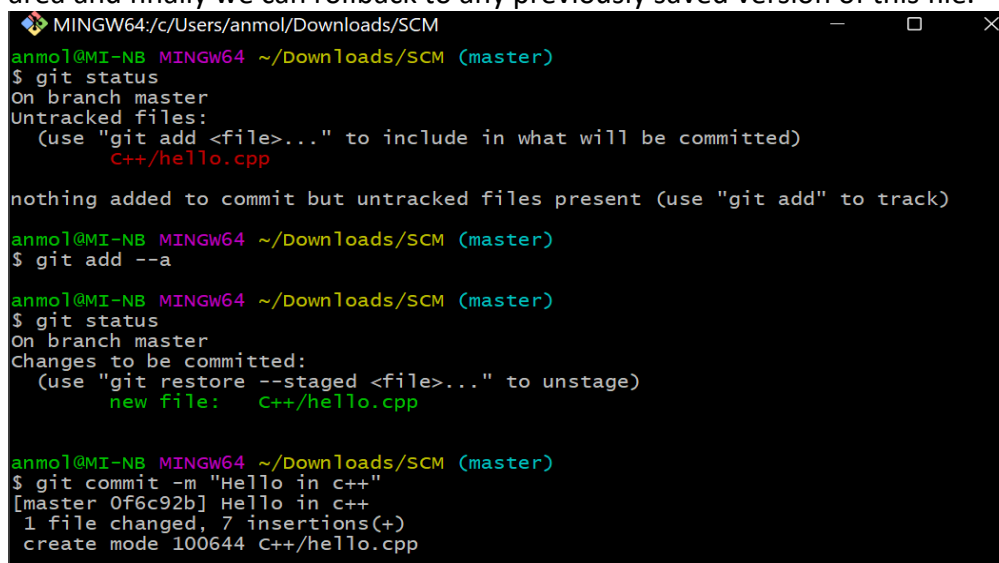


The screenshot shows a terminal window and a File Explorer window. The terminal window displays the following commands and output:

```
MINGW64:/c/Users/adity/OneDrive/Desktop/GIT SCM/pro
adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/GIT SCM/pro (master)
$ pwd
/c/Users/adity/OneDrive/Desktop/GIT SCM/pro
adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/GIT SCM/pro (master)
$ git branch activity1
adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/GIT SCM/pro (master)
$ git branch
  activit1
  activity1
  activity2
* master
adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/GIT SCM/pro (master)
$ git checkout activity1
Switched to branch 'activity1'
M       G_02/design001.txt
A       pro/ljk.txt
adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/GIT SCM/pro (activity1)
$
```

The File Explorer window shows the 'pro' folder. It contains two files: '12' and 'ljk'. The 'ljk' file is a Text Document, 0 KB, and was modified on 27-03-2022 at 13:24.

After this I have done the 3 Step architecture which is tracking the file, send it to staging area and finally we can rollback to any previously saved version of this file.

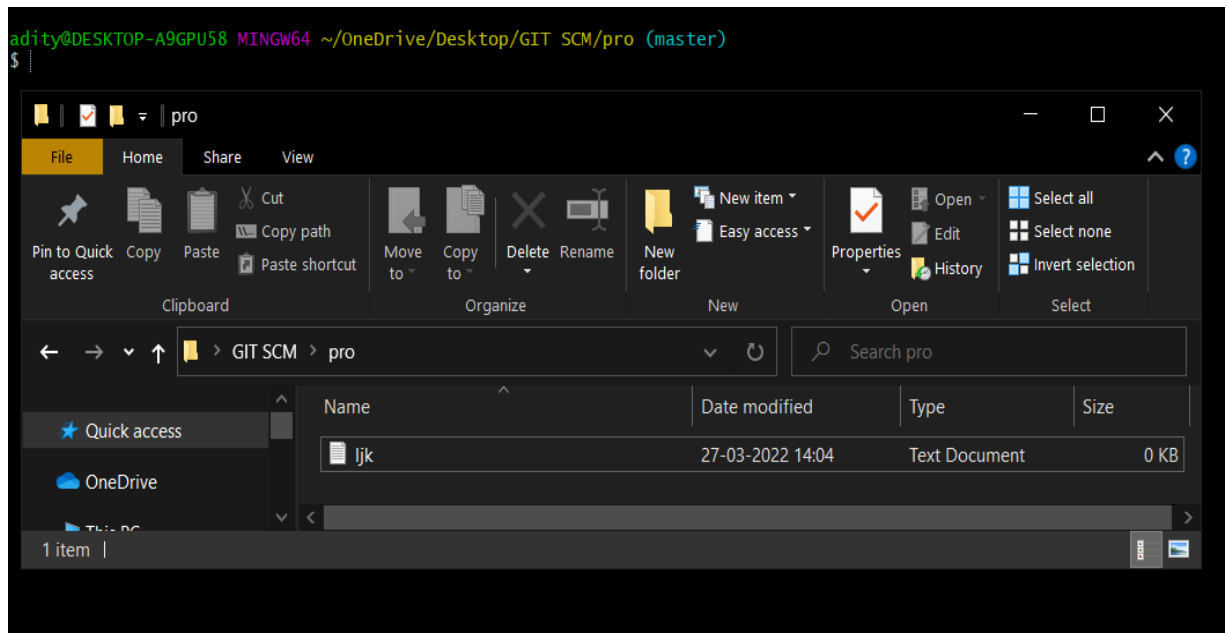


The screenshot shows a terminal window with the following commands and output:

```
MINGW64:/c/Users/anmol/Downloads/SCM
anmol@MI-NB MINGW64 ~/Downloads/SCM (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        c++/hello.cpp

nothing added to commit but untracked files present (use "git add" to track)
anmol@MI-NB MINGW64 ~/Downloads/SCM (master)
$ git add --a
anmol@MI-NB MINGW64 ~/Downloads/SCM (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   c++/hello.cpp
anmol@MI-NB MINGW64 ~/Downloads/SCM (master)
$ git commit -m "Hello in c++"
[master 0f6c92b] Hello in c++
1 file changed, 7 insertions(+)
create mode 100644 c++/hello.cpp
```

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.



In this way we can create and change different branches. We can also merge the branches by using git merge command.

Aim: Git lifecycle description

Theory:

Stages in GIT Life Cycle:

Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, 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:

- Working directory
- Staging area
- 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.

Staging Area:

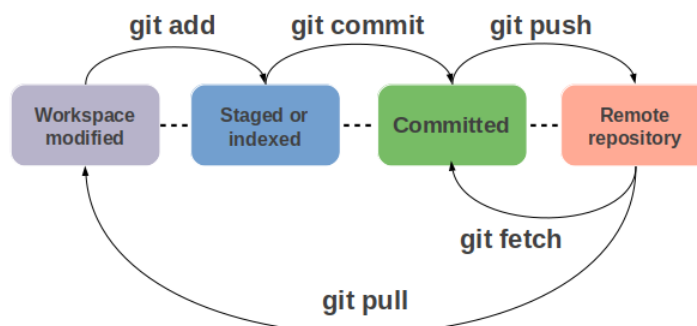
Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.

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. Apart from commit message, this step also records the author and time of the commit. 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.

Remote Repository: means mirror or clone of the local Git repository in GitHub. And pushing means uploading the commits from local Git repository to remote repository hosted in GitHub.

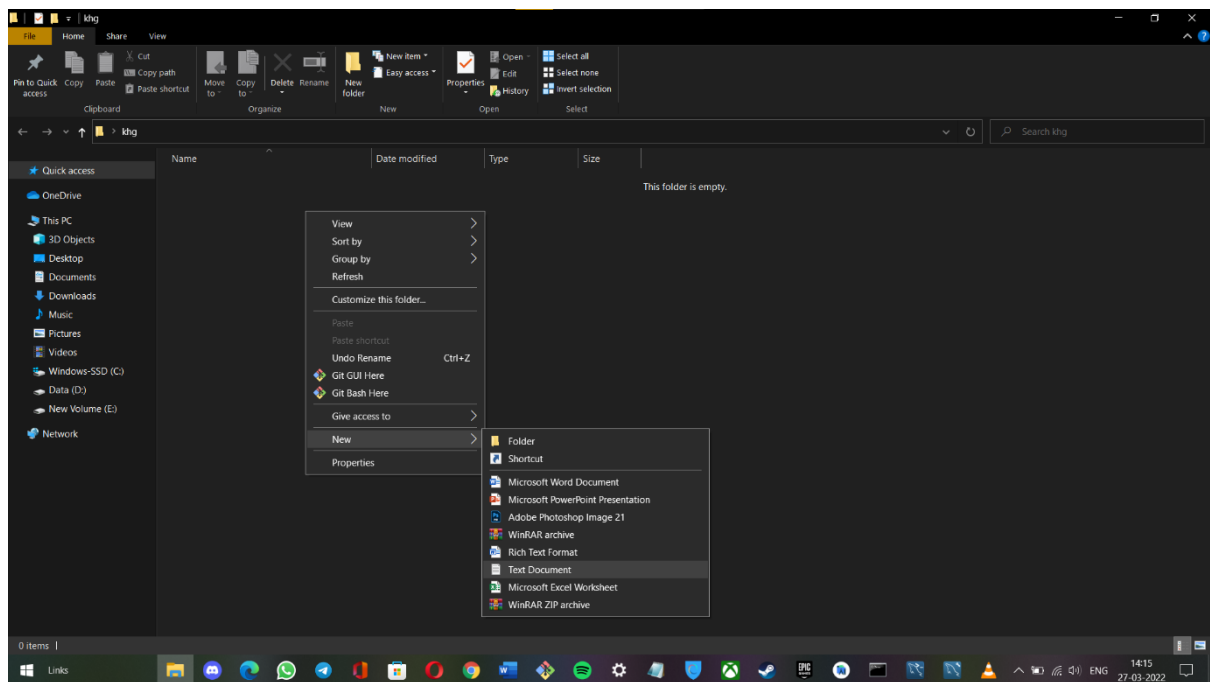
● It's the git life-cycle



Snapshots:

```
MINGW64:/c/Users/adity/OneDrive/Desktop/khg
adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/khg
$ git status
fatal: not a git repository (or any of the parent directories): .git

adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/khg
$ |
```



File	to	to	Folder	History	Invert selection
d	Organize	New	Open	Select	
khg					
Name	Date modified	Type	Size		
git	27-03-2022 14:16	Text Document	0 KB		

```

MINGW64:/c/Users/adity/OneDrive/Desktop/scm.git

adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git
$ ls
git.txt

adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git
$ git status
fatal: not a git repository (or any of the parent directories): .git

adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git
$ git init
Initialized empty Git repository in C:/Users/adity/OneDrive/Desktop/scm.git/.git/

adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git (master)
$ git status
On branch master

No commits yet

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

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

adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git (master)
$ |

```

```
adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git (master)
$ git add --a

adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git (master)
$ git status
On branch master

No commits yet

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

adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git (master)
$ |
```

```
new file:   git.txt

adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git (master)
$ git commit -m "New file added git.txt"
[master (root-commit) 9fc05da] New file added git.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 git.txt

adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git (master)
$
```

```
adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git (master)
$ git status
On branch master
nothing to commit, working tree clean

adity@DESKTOP-A9GPU58 MINGW64 ~/OneDrive/Desktop/scm.git (master)
$ |
```