

**Subject Name: Source Code Management**

**Subject Code: CS181**

**Cluster: Beta**

**Department: DCSE**



**Submitted By:**

AARYAN

PRABHAKAR

2110990018

G01

**Submitted To:**

Dr. Monit Kapoor

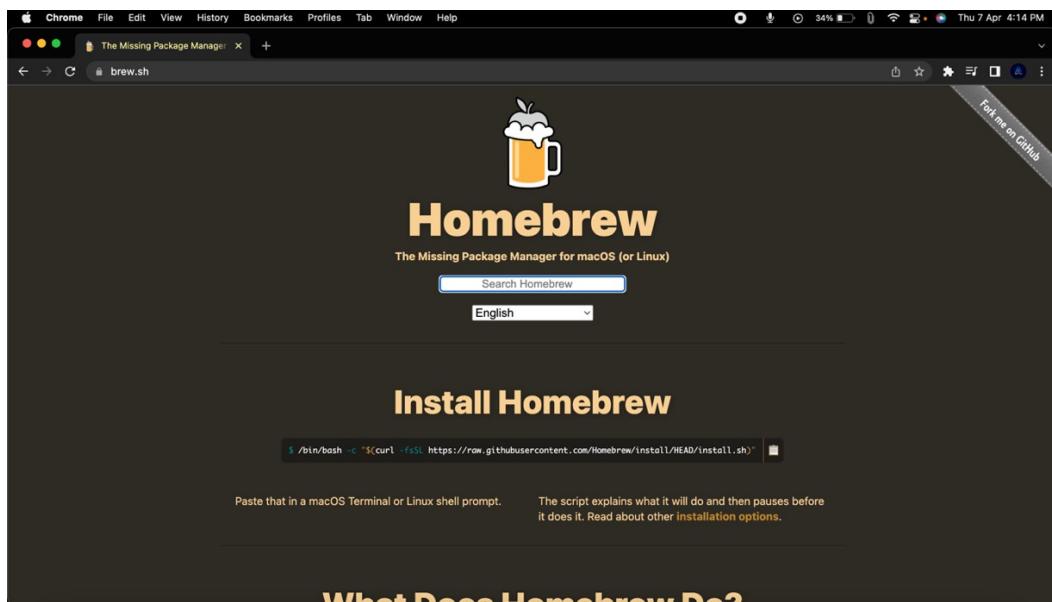
## Experiment No. 01

**Aim:** To set up Git Client.

**Procedure:**

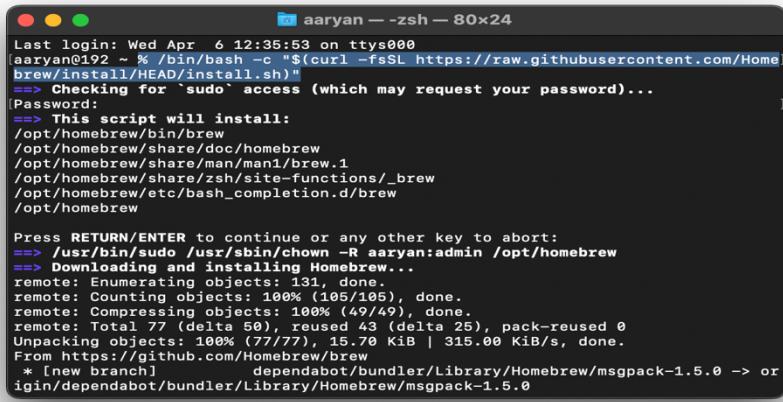
1. Visit the provided link for the installation of Homebrew in your system

(<https://brew.sh>)



2. After opening this site, copy the command and paste it in your terminal and wait for installation to complete.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

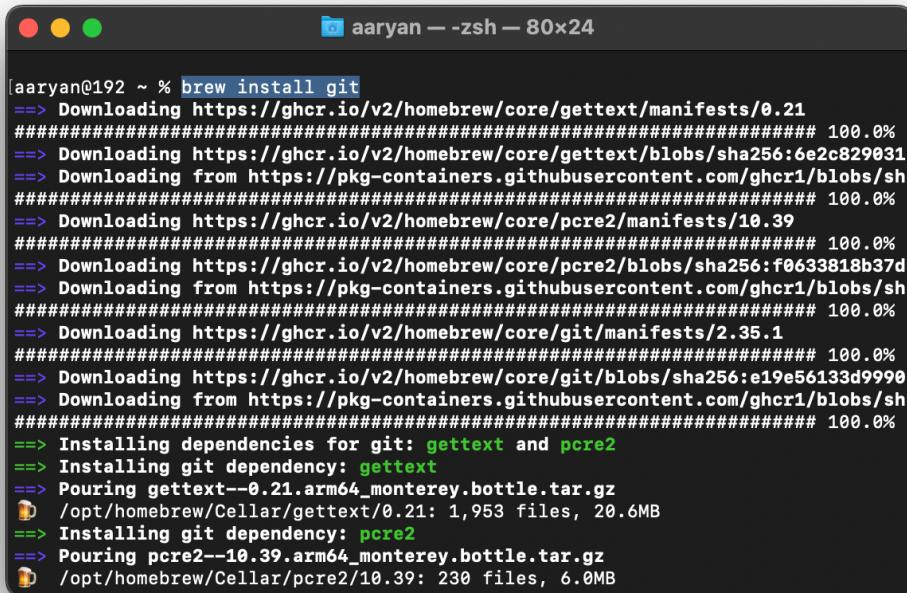


```
aaryan -- zsh -- 80x24
Last login: Wed Apr  6 12:35:53 on ttys000
[aaryan@192 ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
==> Checking for "sudo" access (which may request your password)...
>Password:
==> This script will install:
/opt/homebrew/bin/brew
/opt/homebrew/share/doc/homebrew
/opt/homebrew/share/man/man1/brew.1
/opt/homebrew/share/zsh/site-functions/_brew
/opt/homebrew/etc/bash_completion.d/brew
/opt/homebrew

Press RETURN/ENTER to continue or any other key to abort:
==> /usr/bin/sudo /usr/sbin/chown -R aaryan:admin /opt/homebrew
==> Downloading and installing Homebrew...
remote: Enumerating objects: 131, done.
remote: Counting objects: 100% (105/105), done.
remote: Compressing objects: 100% (49/49), done.
remote: Total 77 (delta 50), reused 43 (delta 25), pack-reused 0
Unpacking objects: 100% (77/77), 15.70 KiB | 315.00 KiB/s, done.
From https://github.com/Homebrew/brew
 * [new branch] dependabot/bundler/Library/Homebrew/msgpack-1.5.0 -> origin/dependabot/bundler/Library/Homebrew/msgpack-1.5.0
```

- Now type the following command in terminal to install git in your system

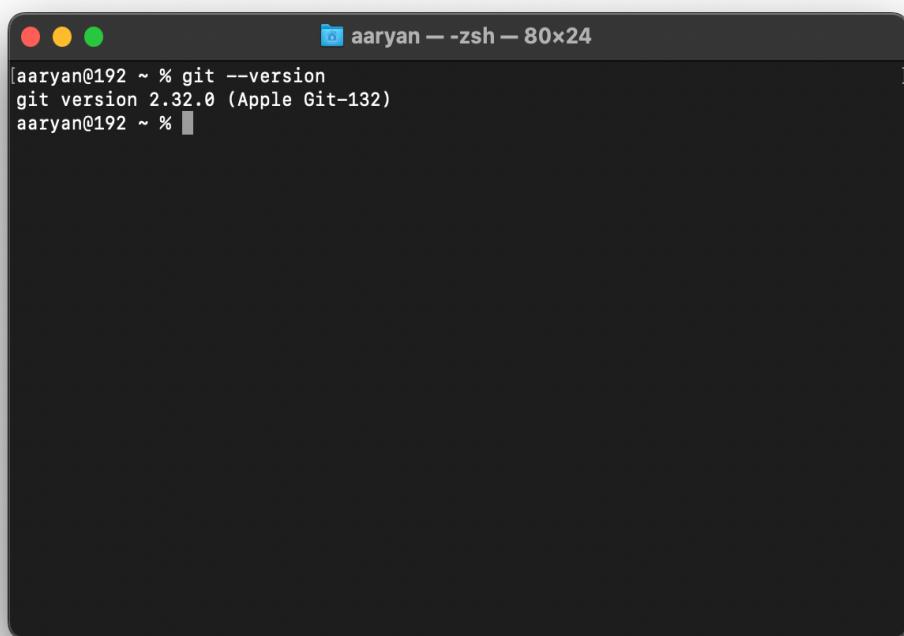
```
brew install git
```



```
[aaryan@192 ~ % brew install git
==> Downloading https://ghcr.io/v2/homebrew/core/gettext/manifests/0.21
#####
 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/gettext/blobs/sha256:6e2c829031
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:6e2c829031
#####
 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/pcre2/manifests/10.39
#####
 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/pcre2/blobs/sha256:f0633818b37d
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:f0633818b37d
#####
 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/git/manifests/2.35.1
#####
 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/git/blobs/sha256:e19e56133d9990
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:e19e56133d9990
#####
 100.0%
==> Installing dependencies for git: gettext and pcre2
==> Installing git dependency: gettext
==> Pouring gettext--0.21.arm64_monterey.bottle.tar.gz
🍺 /opt/homebrew/Cellar/gettext/0.21: 1,953 files, 20.6MB
==> Installing git dependency: pcre2
==> Pouring pcre2--10.39.arm64_monterey.bottle.tar.gz
🍺 /opt/homebrew/Cellar/pcre2/10.39: 230 files, 6.0MB
```

4. After the installation of git you can check whether it is installed or not by typing the following command in your terminal.

```
git --version
```



A screenshot of a macOS terminal window titled "aaryan — zsh — 80x24". The window shows the command "git --version" being run and its output: "git version 2.32.0 (Apple Git-132)". The terminal has a dark background with red, yellow, and green window control buttons at the top left.

```
aaryan@192 ~ % git --version
git version 2.32.0 (Apple Git-132)
aaryan@192 ~ %
```

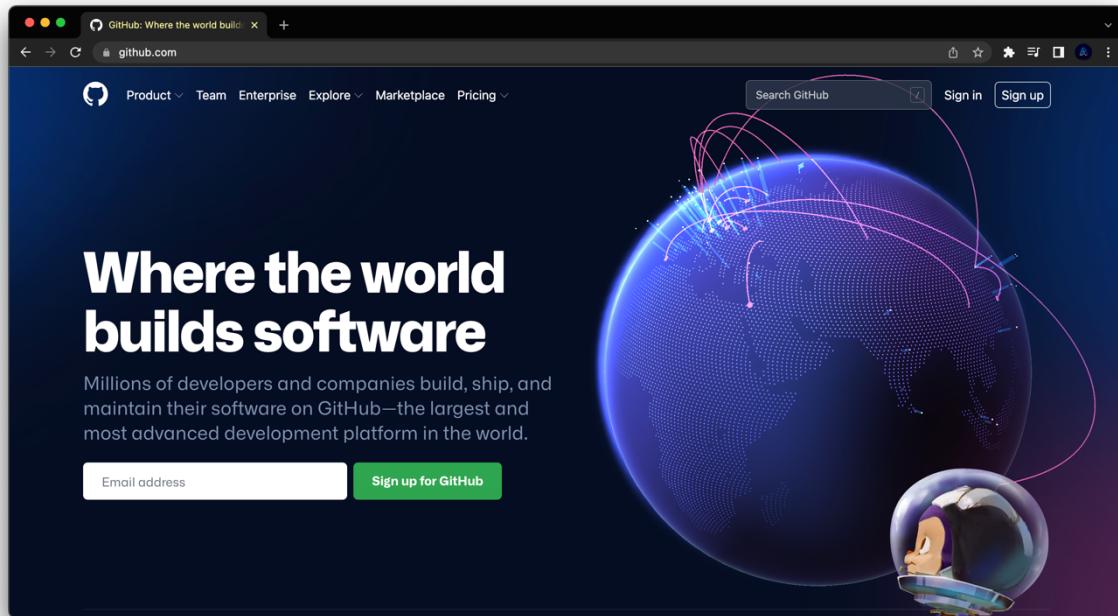
5) As we can see we have successfully installed git into our system.

## Experiment No. 02

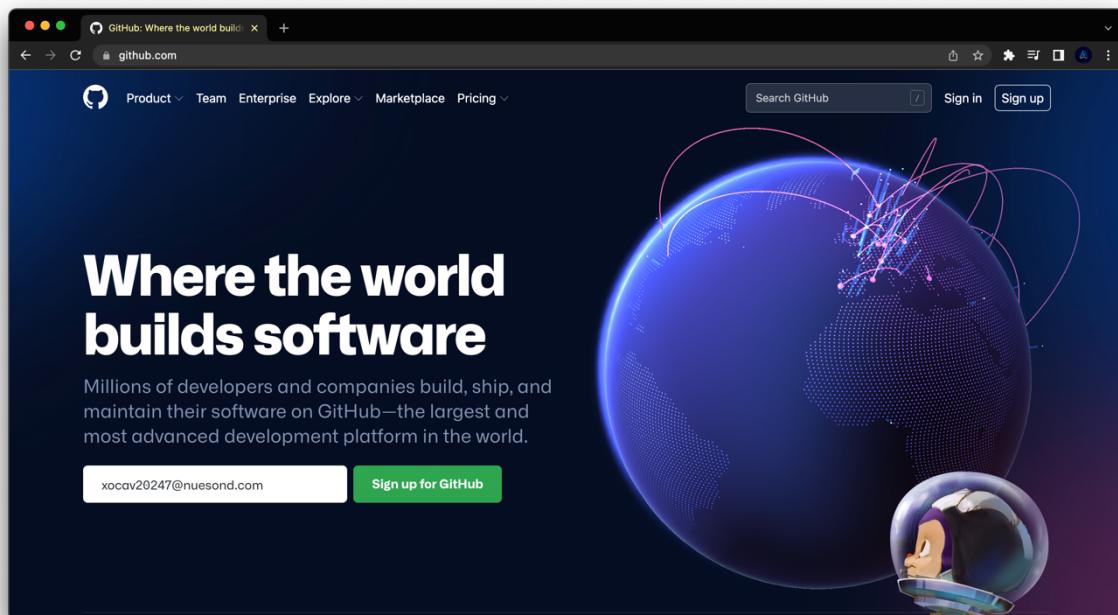
**Aim:** To set up GitHub Account

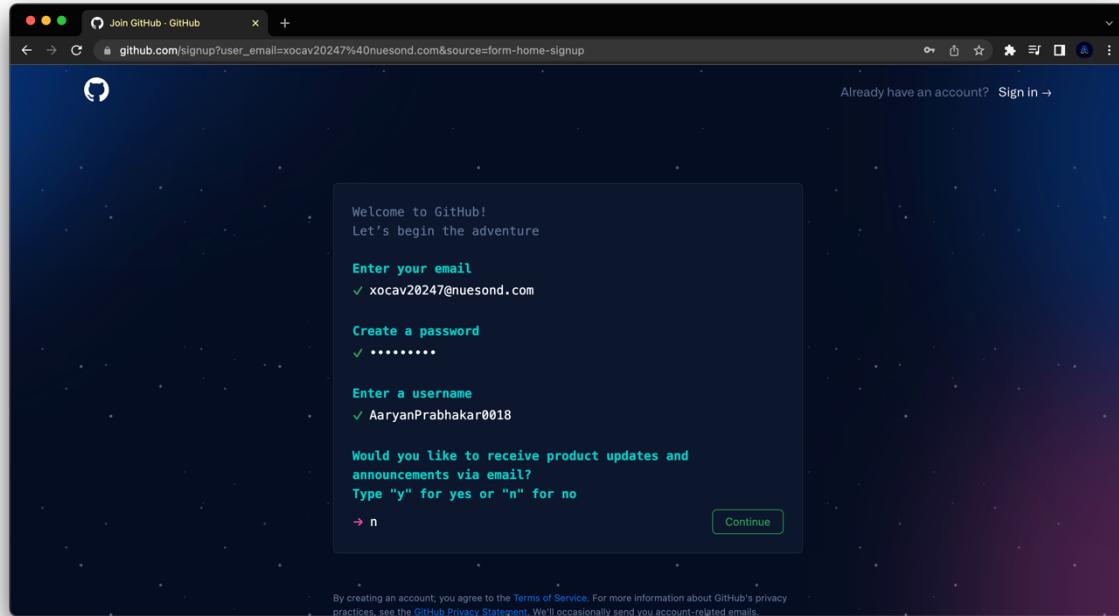
**Procedure:**

1. Click on <https://github.com> to open GitHub on browser.

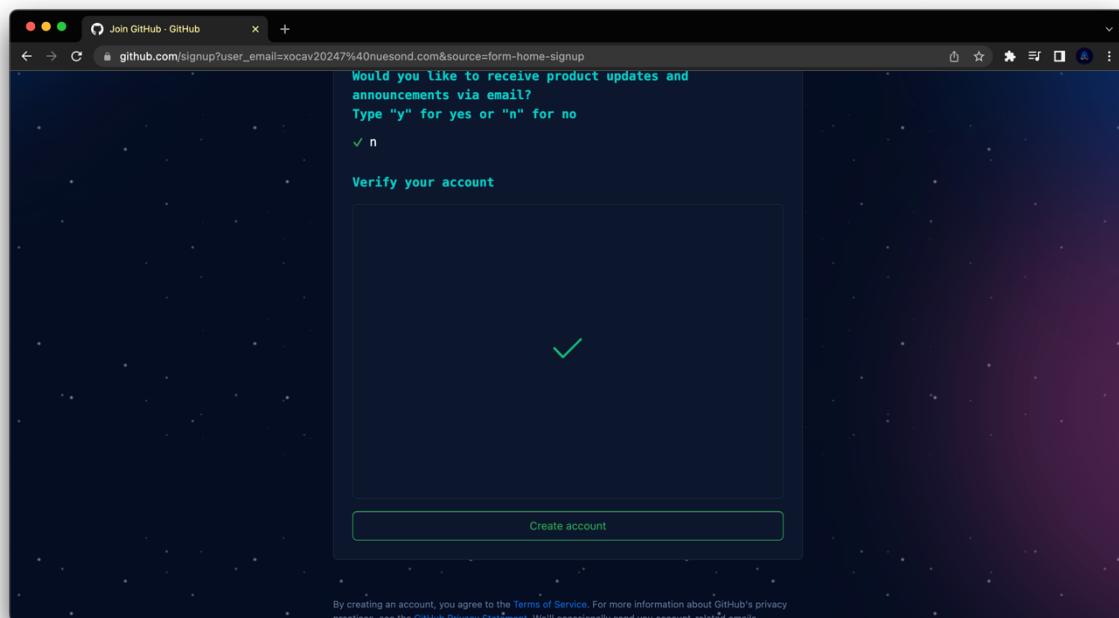


2. Type in your email in the Email address box and click on Sign up for GitHub. On the next page click on continue.





3. Solve the puzzle and Click on Create Account.



4. Your homepage of GitHub will be loaded and your GitHub account is created successfully.

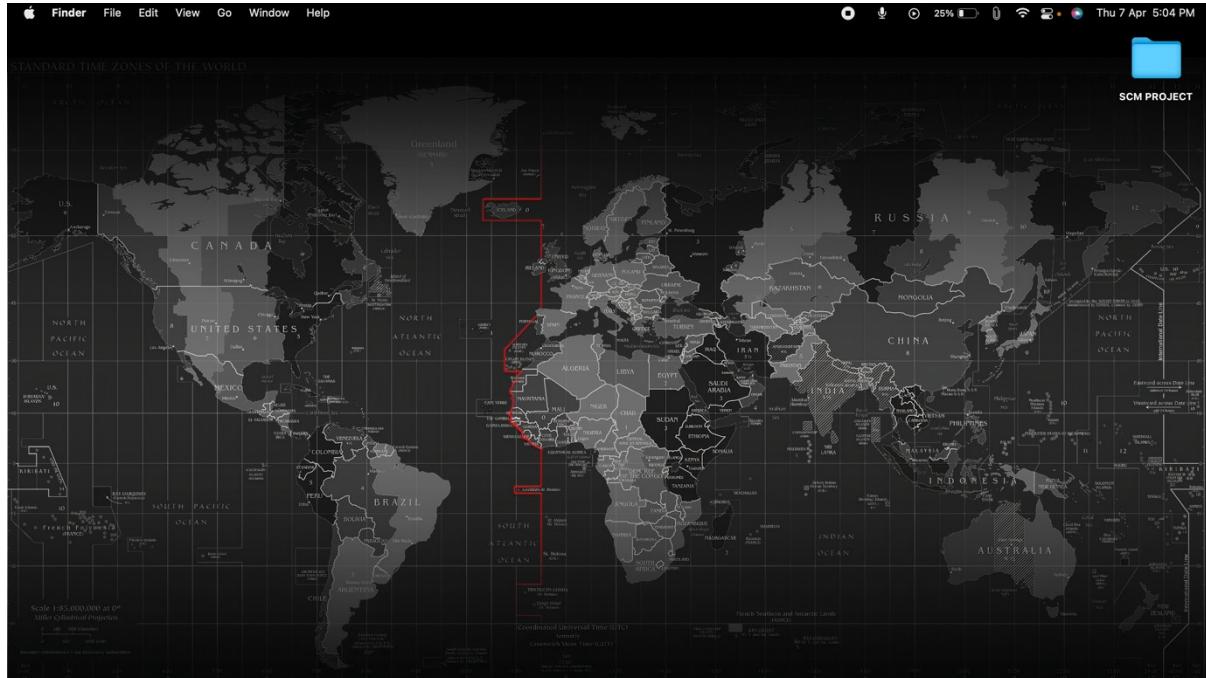
## Experiment No. 03

**Aim:** To create a program to generate logs.

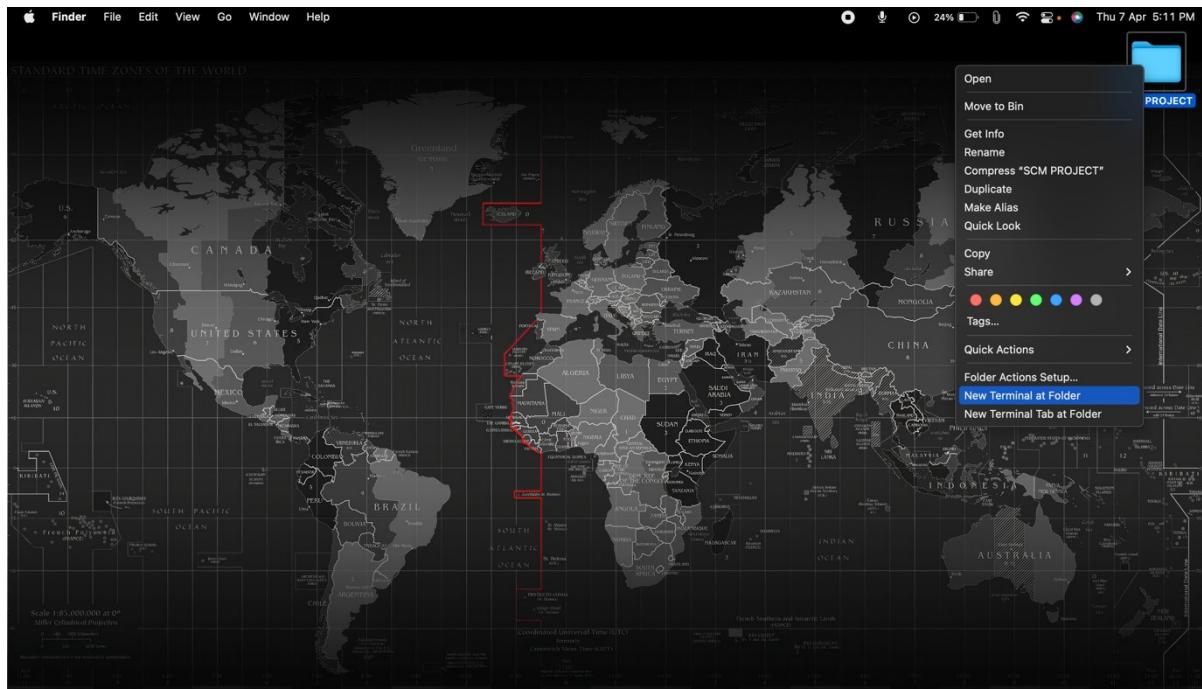
### **Procedure:**

#### ➤ **Initialising Git Repository:**

1. Make a new folder and name it SCM PROJECT.

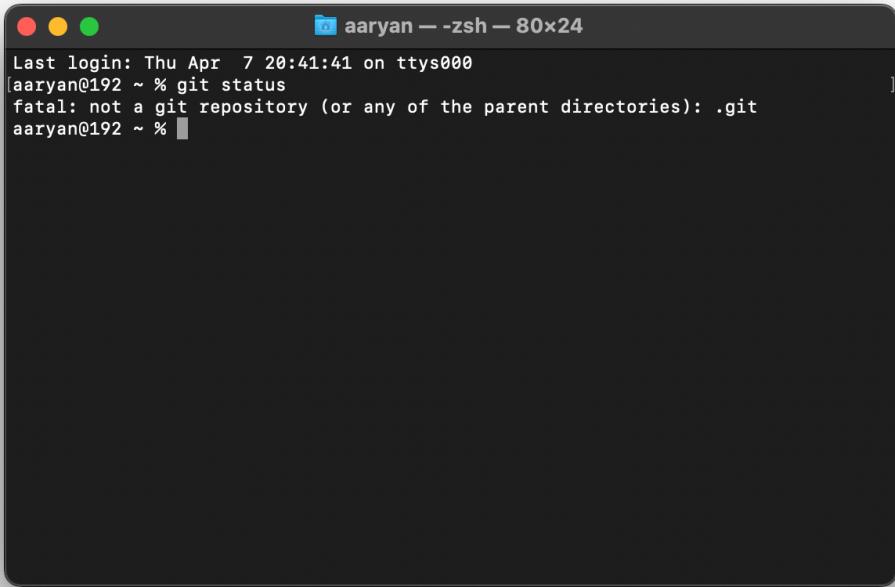


- 2) Now right click on the folder and select **new terminal at folder** option from the menu.



3) Firstly, check the Present Working Directory by typing the command **pwd** and press Enter.

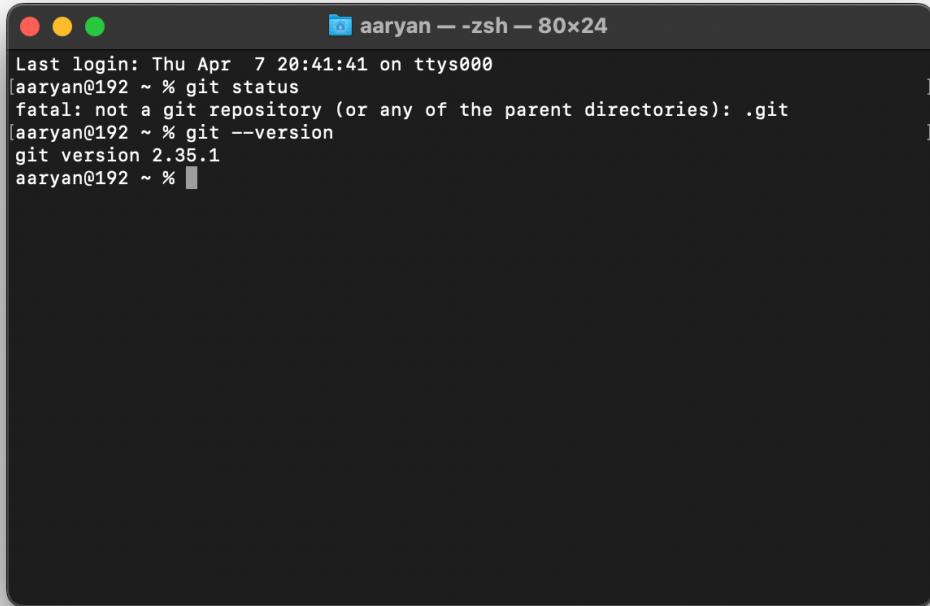
4) Now, to know every status about the files in the repository, type command **git status**.



aaryan -- zsh -- 80x24

```
Last login: Thu Apr  7 20:41:41 on ttys000
[aaryan@192 ~ % git status
fatal: not a git repository (or any of the parent directories): .git
aaryan@192 ~ % ]
```

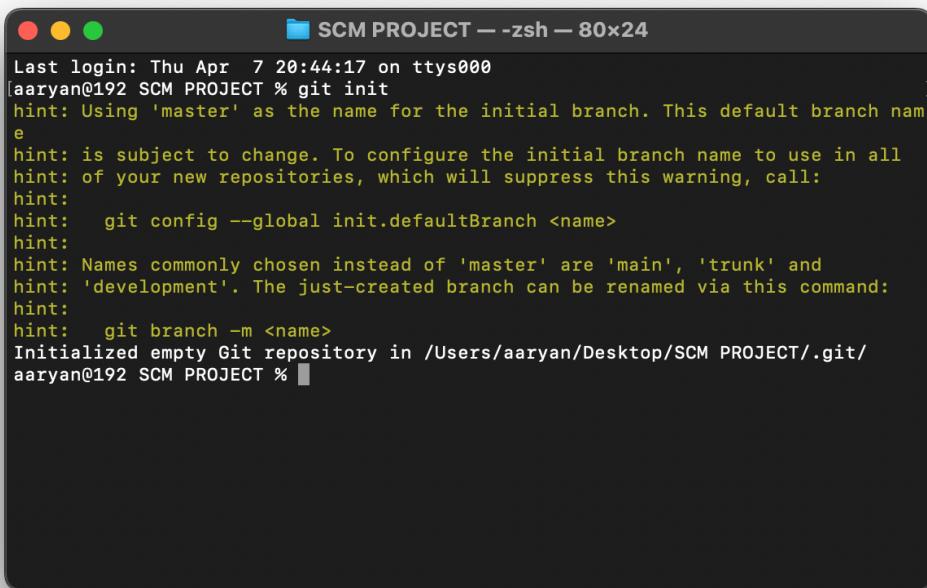
5) To get the information about the version of your git, type the command ***git --version***



aaryan -- zsh -- 80x24

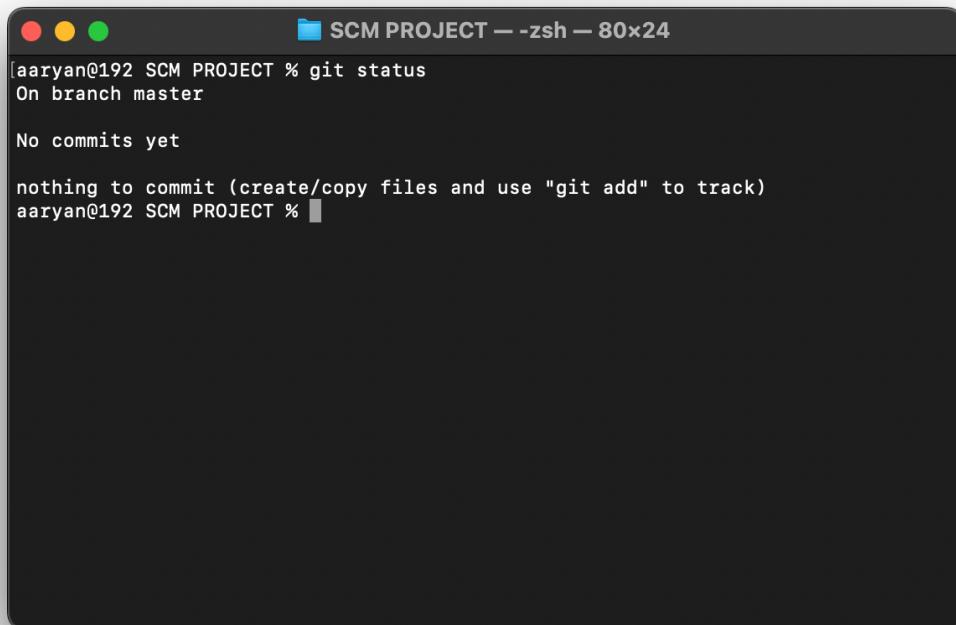
```
Last login: Thu Apr  7 20:41:41 on ttys000
[aaryan@192 ~ % git status
fatal: not a git repository (or any of the parent directories): .git
[aaryan@192 ~ % git --version
git version 2.35.1
aaryan@192 ~ % ]
```

6) Now, to initialize git repository type in command ***git init***



```
Last login: Thu Apr  7 20:44:17 on ttys000
[aaryan@192 SCM PROJECT % git init
hint: Using 'master' as the name for the initial branch. This default branch nam
e
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/aaryan/Desktop/SCM PROJECT/.git/
aaryan@192 SCM PROJECT %
```

7) Check the status of your repository by typing the command **git status**



```
[aaryan@192 SCM PROJECT % git status
On branch master

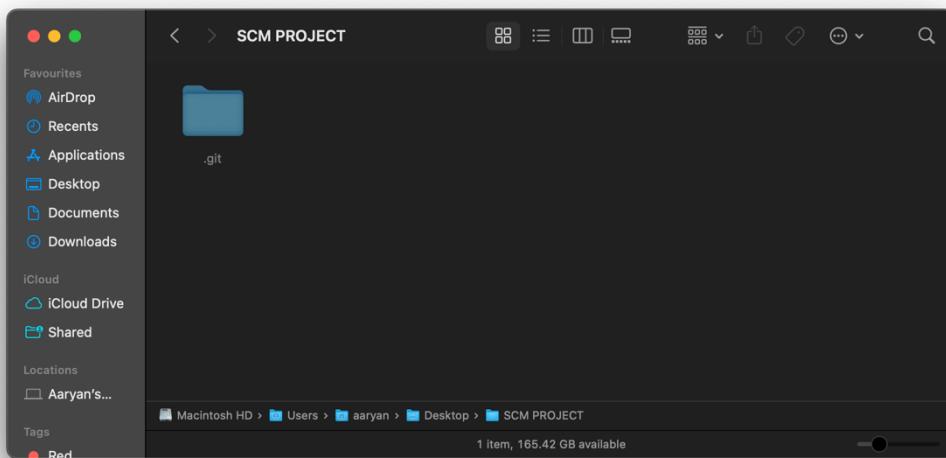
No commits yet

nothing to commit (create/copy files and use "git add" to track)
aaryan@192 SCM PROJECT %
```

8) Now go to your folder.

9) By default, the `.git` folder is hidden. To see the `.git` folder go inside the folder and press

**Command+shift+period(.)**



#### ➤ **Configuring Username and User Email:**

So, now that you have successfully created your GitHub account. It is best to use your username and email of GitHub in Git. So, let's update our username and user email

- 1) Type command `git config --global user.name "YOUR USERNAME"`. It will update your username.
- 2) Similarly, you can update your email using the command `git config --global user.email "YOUR EMAIL"`.

```
[aaryan@192 SCM PROJECT % git config --global user.name "AaryanPrabhakar0018"]
[aaryan@192 SCM PROJECT % git config --global user.email xocav20247@nuesond.com]
aaryan@192 SCM PROJECT %
```

- 3) To verify the updated username type command `git config user.name` .

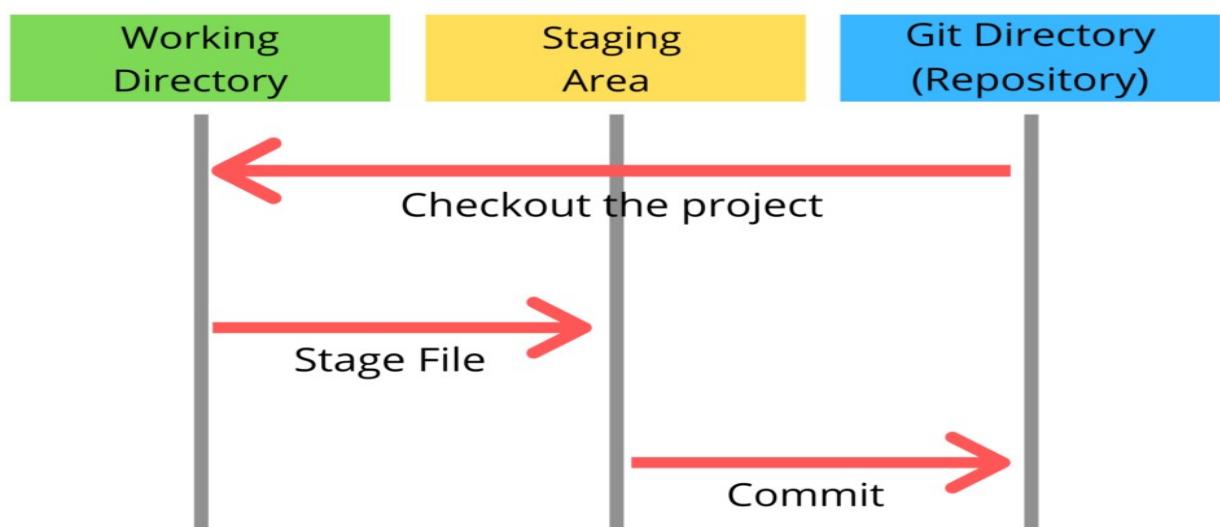
4) Similarly, email can be verified using ***git config user.email***

```
SCM PROJECT — zsh — 80x24
[aaryan@192 SCM PROJECT % git config user.name
AaryanPrabhakar0018
[aaryan@192 SCM PROJECT % git config user.email
xocav20247@nuesond.com
[aaryan@192 SCM PROJECT % ]
```

➤ **Three-Stage Architecture:**

Many VCS's use a two-stage architecture i.e., a repository and a working copy. Git uses three-stage architecture i.e., a working directory, staging area and local repository. Working Directory is the folder which we created i.e., SCM Project.

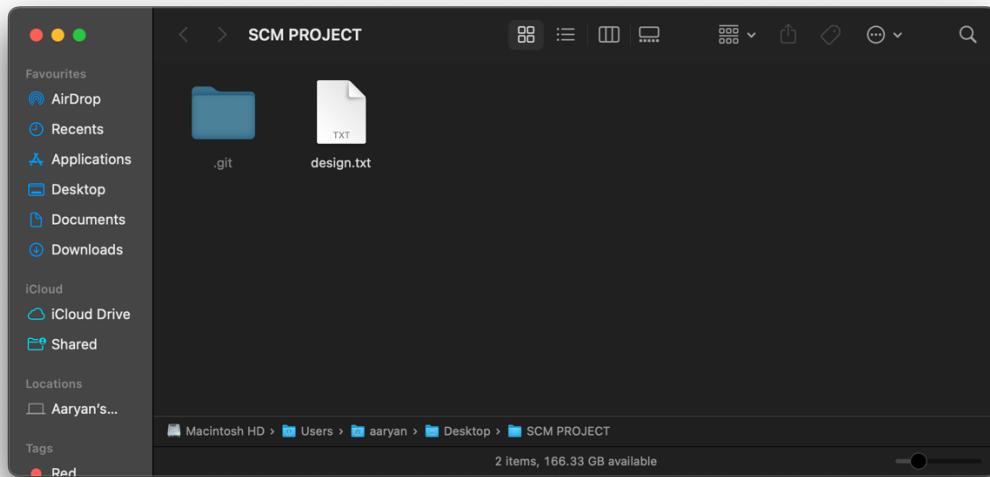
Working Directory was converted into Repository where we used ***git init*** command to initial the repository.



Working directory specifies the file explorer's folder where your files are stored, Staging area is an area where those files are present which you want to send to commit (to create snapshot of files). After commit is fired, files which are in staging area will move to Git Repository.

Now if you made any changes in your files which are in Git repository, those have to be added into the Staging area and Committed

1)Create a file in the empty Git Repository (SCM Project folder). Let's create a .txt file and name it as design.txt



2)Now, open Git Bash and type command *git status*.

We can clearly see that the design file is in Untracked files and it is colored in red

```
Last login: Fri Apr  8 21:10:08 on ttys000
[aaryan@192 SCM PROJECT % git status
On branch master

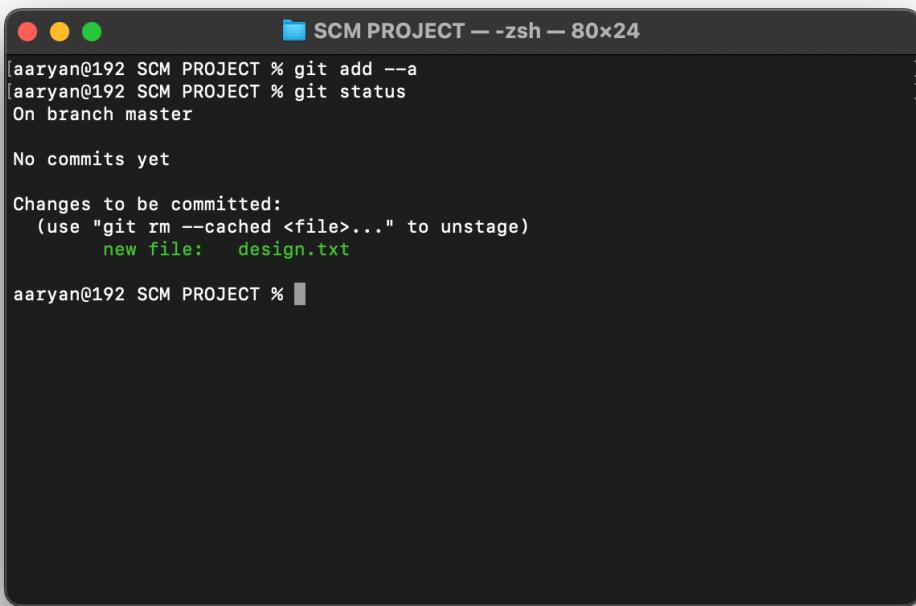
No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
aaryan@192 SCM PROJECT % ]
```

3) Now type ***git add --a*** which will move or add the files to the Staging area. Here --a means all the latest files changed or added. You can also specify a file like ***git add design.txt***.

4) After that check status using ***git status***

A screenshot of a terminal window titled "SCM PROJECT -- zsh -- 80x24". The window shows the following command-line session:

```
[aaryan@192 SCM PROJECT % git add --a
[aaryan@192 SCM PROJECT % git status
On branch master

No commits yet

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

aaryan@192 SCM PROJECT % ]
```

The terminal has three colored window controls (red, yellow, green) at the top left. The title bar is "SCM PROJECT -- zsh -- 80x24". The prompt "[aaryan@192 SCM PROJECT %" appears twice, followed by the execution of "git add --a", then "git status". The output shows the current branch is "master" and there are no commits. It lists a single change: a new file named "design.txt" that is being tracked for commit. The prompt ends with "% ]".

Observe that the design.txt has turned from red to green. This means that file has been added to the staging area and is ready to be committed

5) Now to commit the files in the staging area, use command ***git commit -m "Any Message"*** and check the status

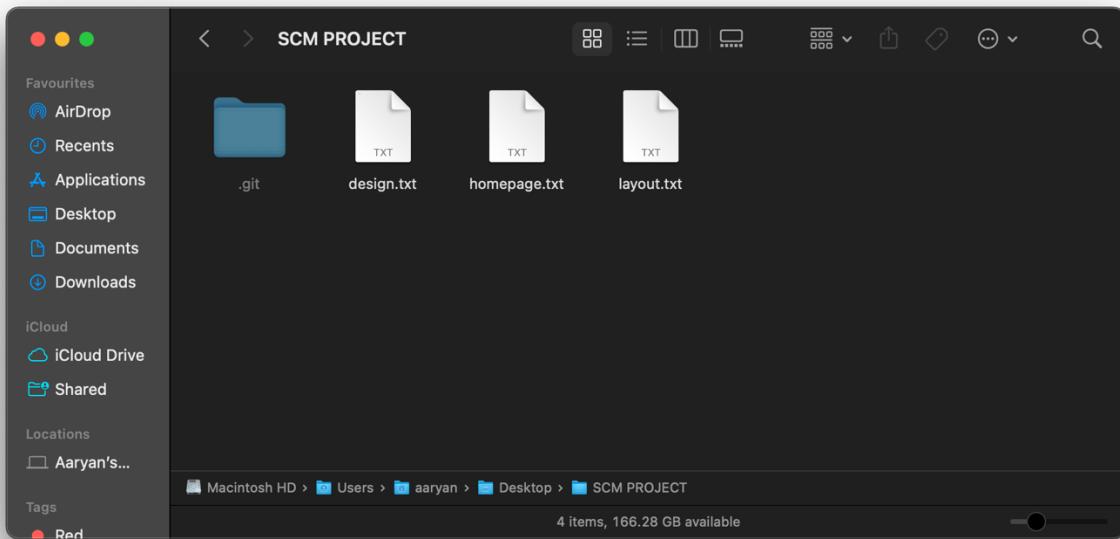
```
● ○ ● SCM PROJECT — zsh — 80x24
[aaryan@192 SCM PROJECT % git add --a
[aaryan@192 SCM PROJECT % git status
On branch master

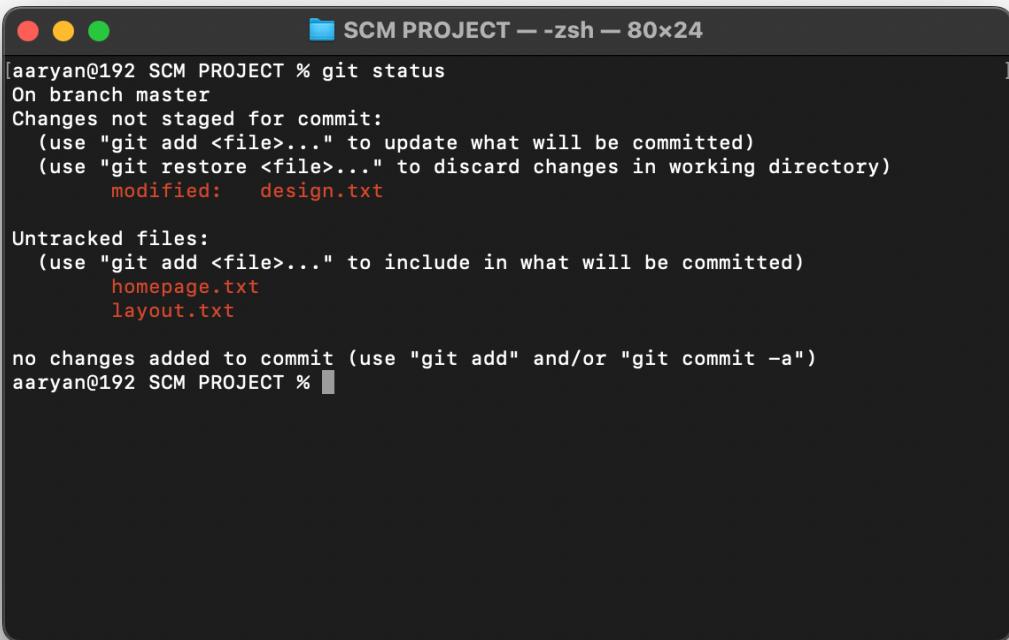
No commits yet

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

[aaryan@192 SCM PROJECT % git commit -m "first commit to design.txt"
[master (root-commit) 8c26962] first commit to design.txt
  1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 design.txt
[aaryan@192 SCM PROJECT % git status
On branch master
nothing to commit, working tree clean
aaryan@192 SCM PROJECT %
```

- 6) Now let's check the status. It shows that design.txt is modified and homepage.txt and layout.txt are untracked. All these changes have to be committed.



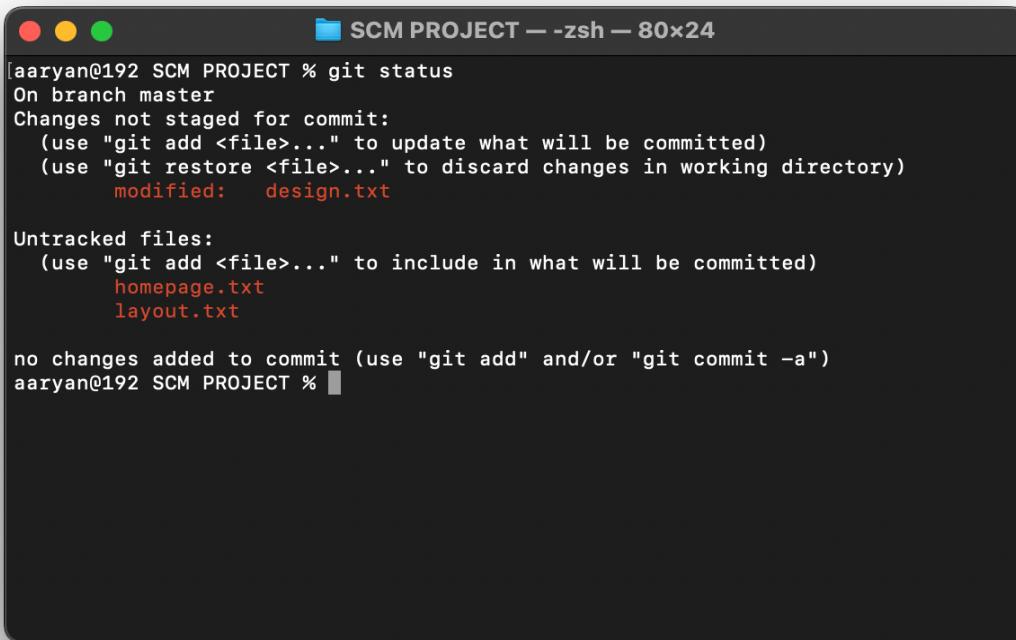


```
[aaryan@192 SCM PROJECT % git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   design.txt

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

no changes added to commit (use "git add" and/or "git commit -a")
aaryan@192 SCM PROJECT %
```

7)Now let's check the status. It shows that design.txt is modified and homepage.txt and layout.txt are untracked. All these changes have to be committed.

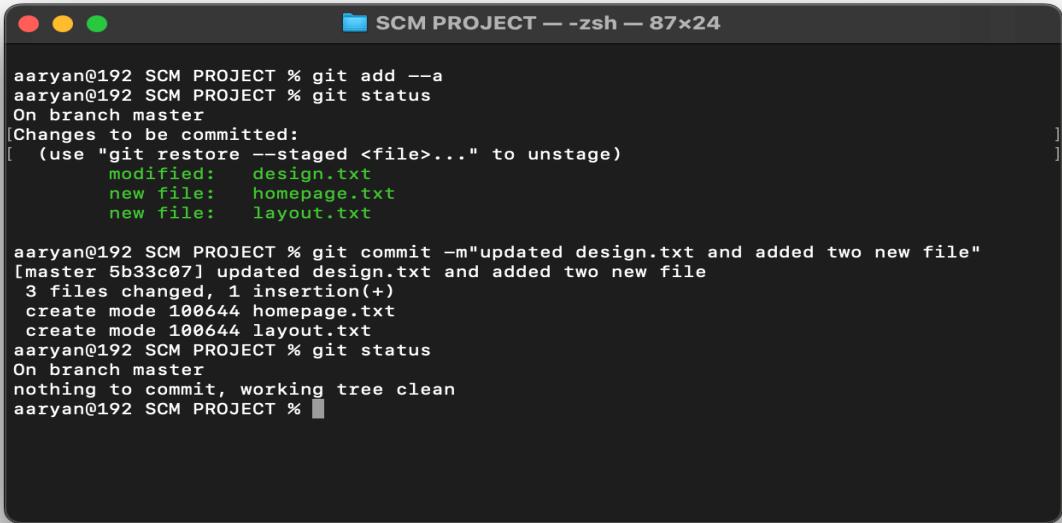


```
[aaryan@192 SCM PROJECT % git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   design.txt

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

no changes added to commit (use "git add" and/or "git commit -a")
aaryan@192 SCM PROJECT %
```

8)Now we have to commit these changes .



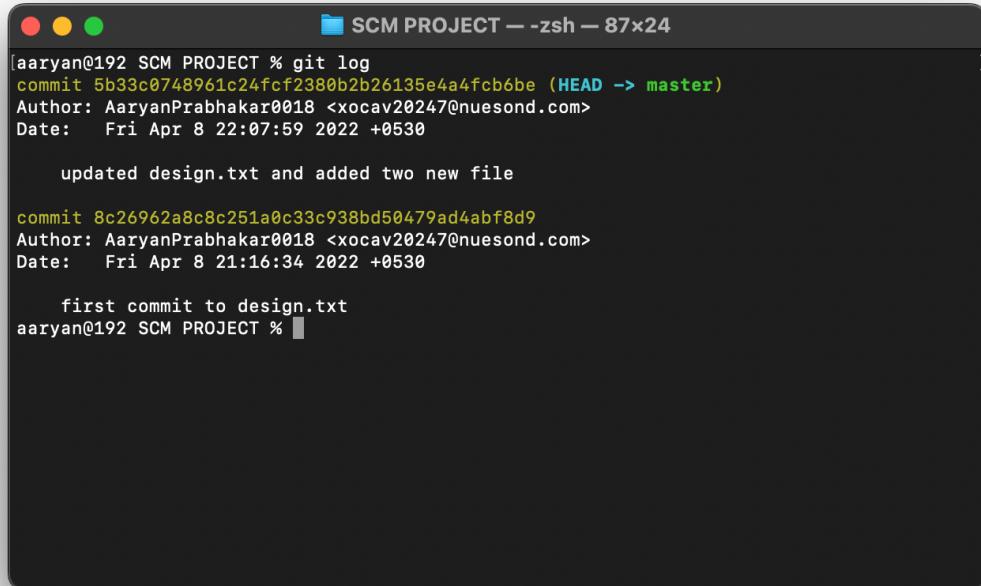
```
SCM PROJECT -- zsh -- 87x24

aaryan@192 SCM PROJECT % git add --a
aaryan@192 SCM PROJECT % git status
On branch master
[Changes to be committed:
 [ (use "git restore --staged <file>..." to unstage)
   modified: design.txt
   new file: homepage.txt
   new file: layout.txt

aaryan@192 SCM PROJECT % git commit -m"updated design.txt and added two new file"
[master 5b33c07] updated design.txt and added two new file
 3 files changed, 1 insertion(+)
 create mode 100644 homepage.txt
 create mode 100644 layout.txt
aaryan@192 SCM PROJECT % git status
On branch master
nothing to commit, working tree clean
aaryan@192 SCM PROJECT %
```

#### ➤ Logs:

Git log is a utility tool to review and read a history of everything that happens to a repository. Every change committed and the message corresponding to it is recorded and can be seen in the git log. **Command:git log**



```
SCM PROJECT -- zsh -- 87x24

[aaryan@192 SCM PROJECT % git log
commit 5b33c0748961c24fcf2380b2b26135e4a4fcb6be (HEAD -> master)
Author: AaryanPrabhakar0018 <xocav20247@nuesond.com>
Date:   Fri Apr 8 22:07:59 2022 +0530

    updated design.txt and added two new file

commit 8c26962a8c8c251a0c33c938bd50479ad4abf8d9
Author: AaryanPrabhakar0018 <xocav20247@nuesond.com>
Date:   Fri Apr 8 21:16:34 2022 +0530

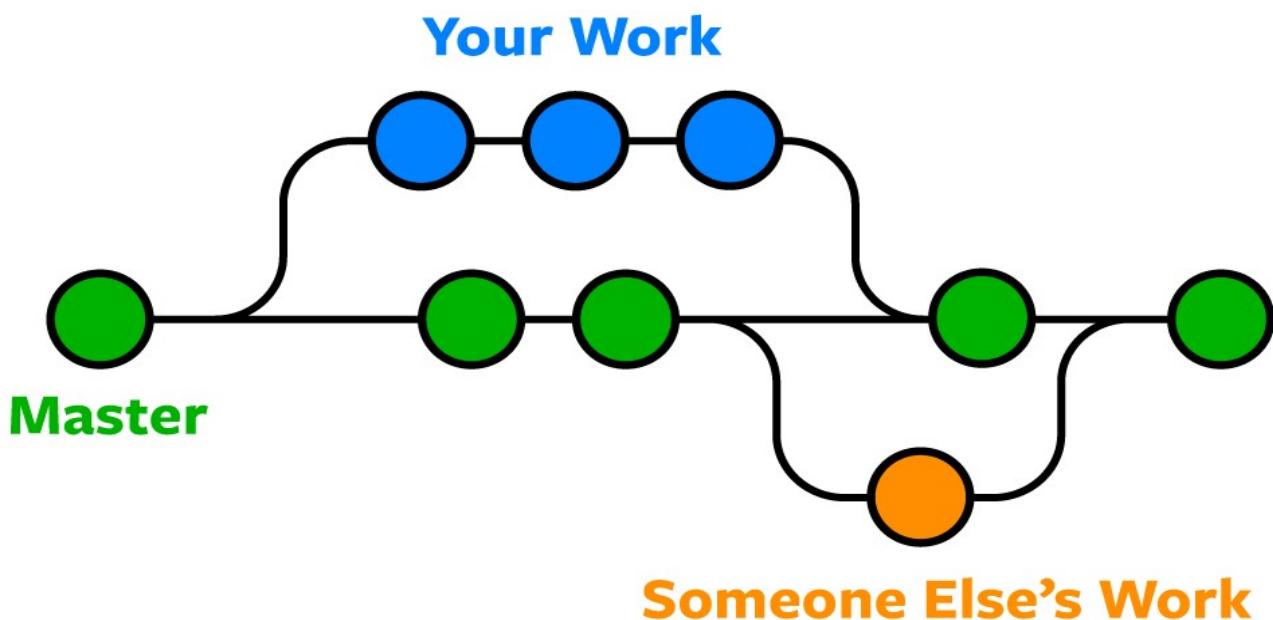
    first commit to design.txt
aaryan@192 SCM PROJECT %
```

## Experiment No. 04

**Aim:** To create and visualize branches.

**Theory:** Nearly every VCS has some form of branching support. Branching means you diverge from the main line of development and continue to do work without messing with that main line.

Branching simply means parallel development.

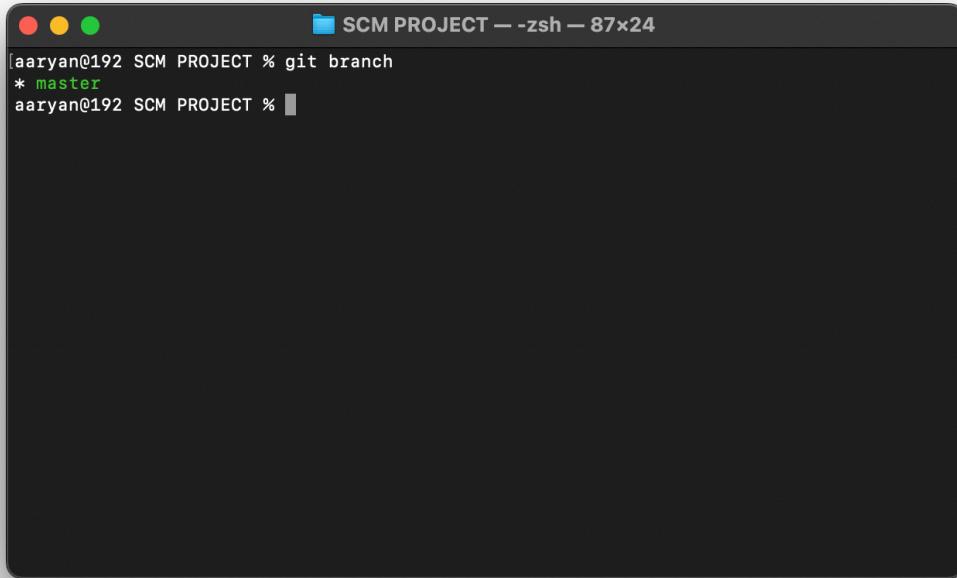


Suppose your team is making a website on the master. You did your work and you think that you can make the website better and you want to do some experiments. Without messing with the website, you can experiment with it using this amazing feature git provides us i.e., branching.

Whenever we create a child branch from a parent branch, all the data will be copied in child branch. But any changes after committing in child will not reflect in parent branch. Similarly, any changes in parent branch after creating child branch, won't be reflected in child branch.

### Procedure:

1. Open the folder(**SCM PROJECT**)in terminal and type the command **git branch**



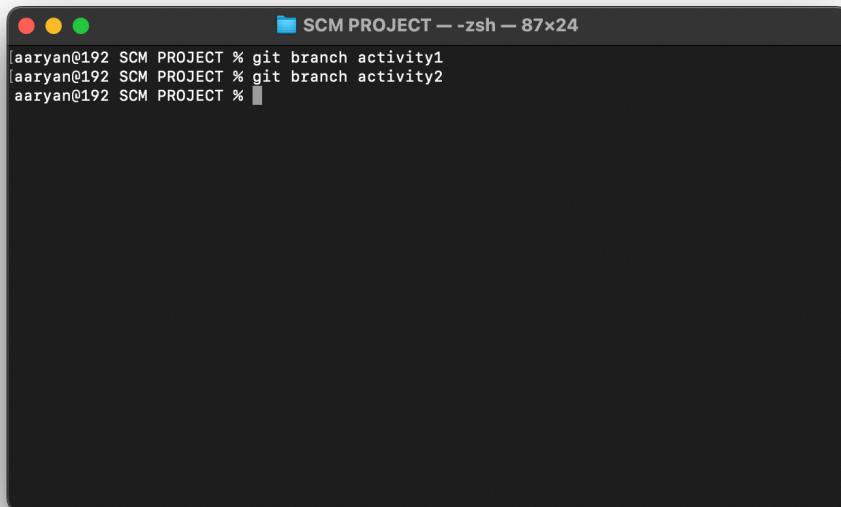
A screenshot of a macOS terminal window titled "SCM PROJECT -- zsh -- 87x24". The window shows the command "git branch" being run, with the output indicating a single branch named "master" marked with an asterisk (\*). The terminal has a dark background with light-colored text.

```
[aaryan@192 SCM PROJECT % git branch
* master
aaryan@192 SCM PROJECT % ]
```

Master branch is the default and main branch. **Green** colour means that this branch active at the moment. ***git branch*** command shows the list of all the branches.

2. To create a new branch from master, use the command ***git branch <Name of the branch>***

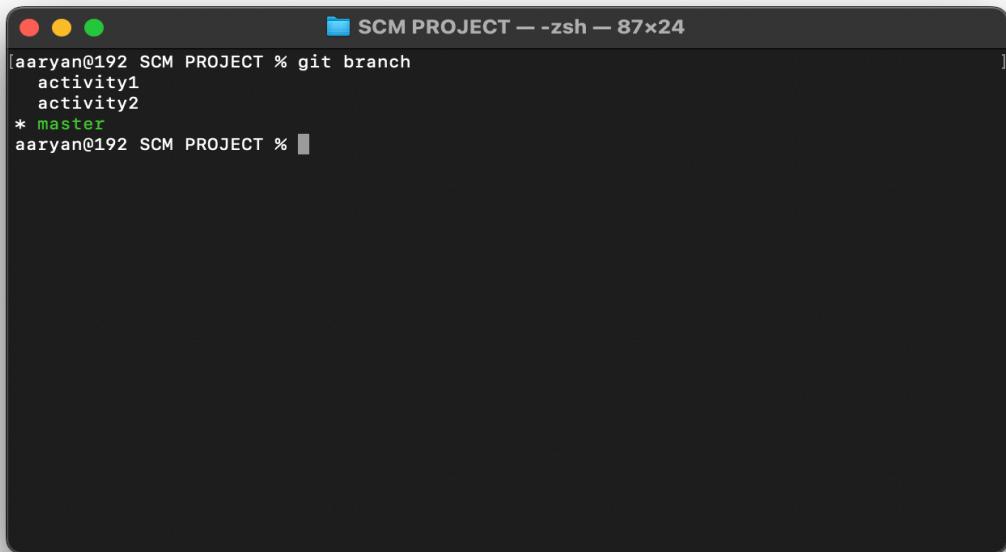
Let's take for example create two branches from master branch ***git branch activity1*** and ***git branch activity2***



A screenshot of a macOS terminal window titled "SCM PROJECT -- zsh -- 87x24". The window shows two commands being run sequentially: "git branch activity1" and "git branch activity2". Both commands return an empty list, indicating that no branches have been created yet. The terminal has a dark background with light-colored text.

```
[aaryan@192 SCM PROJECT % git branch activity1
[aaryan@192 SCM PROJECT % git branch activity2
[aaryan@192 SCM PROJECT % ]]
```

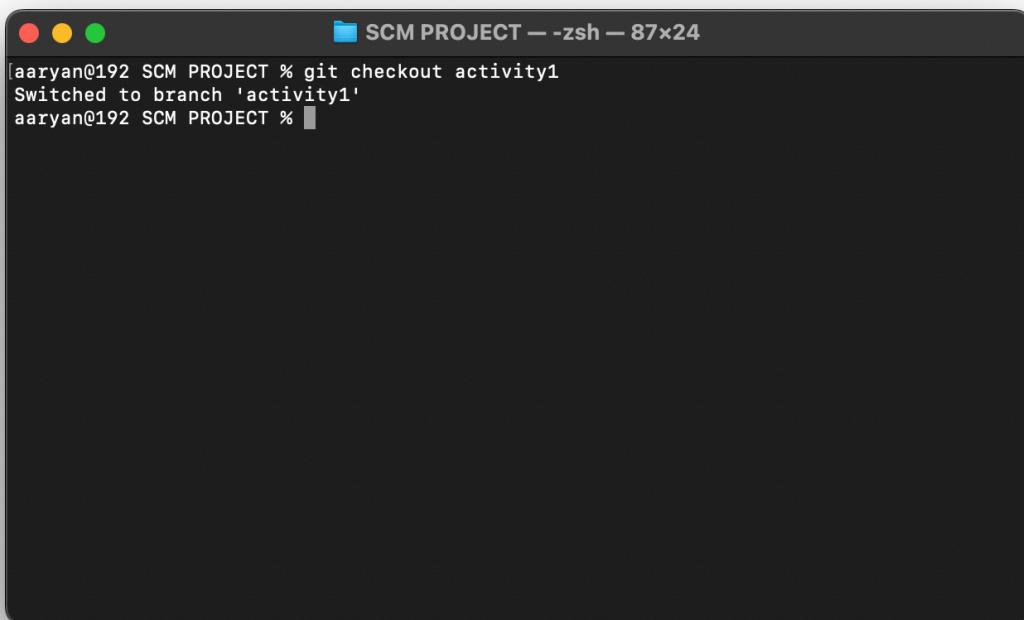
3. To see all the branches, use command ***git branch***



A screenshot of a macOS terminal window titled "SCM PROJECT -- -zsh -- 87x24". The window shows the command "git branch" being run, which lists three branches: "activity1", "activity2", and the current branch, "master". The "master" branch is indicated by an asterisk (\*).

```
[aaryan@192 SCM PROJECT % git branch
  activity1
  activity2
* master
aaryan@192 SCM PROJECT % ]
```

4. Now, to shift or go to the newly created branches use the command ***git checkout <branch name>***

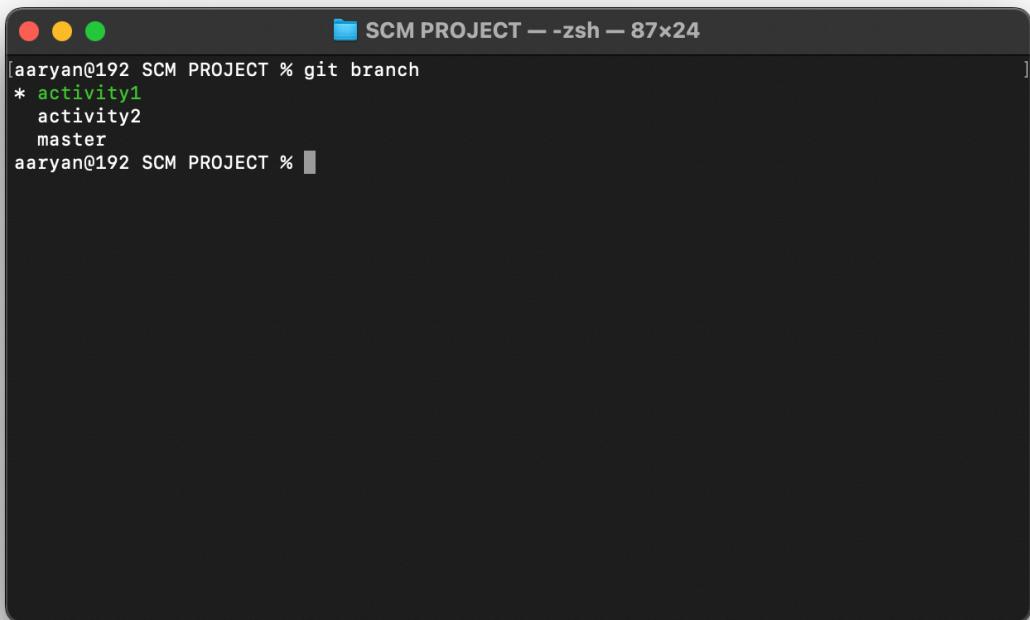


A screenshot of a macOS terminal window titled "SCM PROJECT -- -zsh -- 87x24". The command "git checkout activity1" is run, followed by a confirmation message "Switched to branch 'activity1'". The prompt then changes to "aaryan@192 SCM PROJECT %".

```
[aaryan@192 SCM PROJECT % git checkout activity1
Switched to branch 'activity1'
aaryan@192 SCM PROJECT % ]
```

Now, we can clearly see that it is switched to the branch activity1.

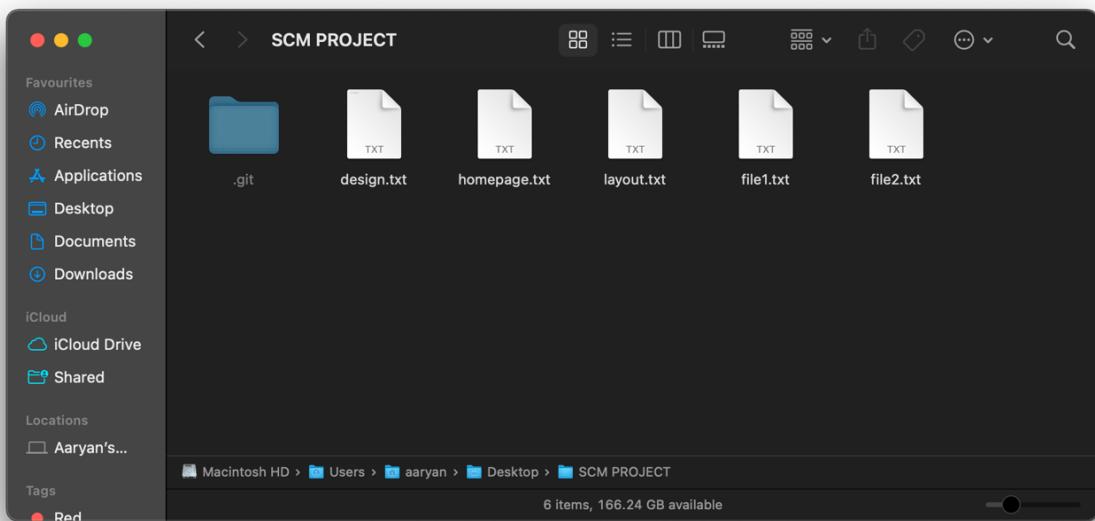
To double verify we can use the command ***git branch***

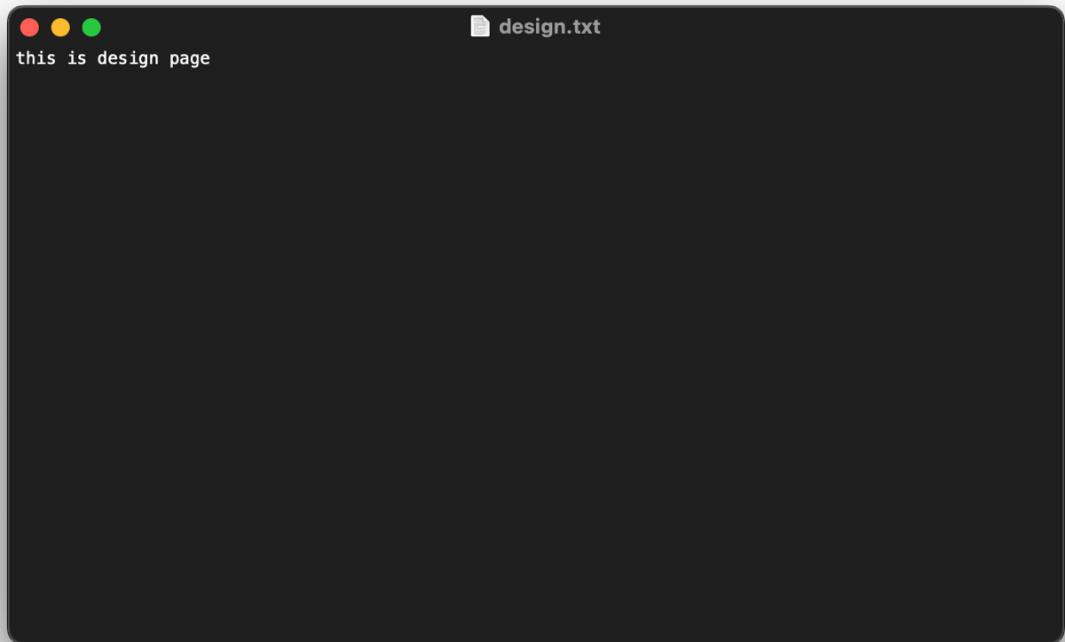


A screenshot of a terminal window titled "SCM PROJECT — -zsh — 87x24". The window shows the command "git branch" being run, with the output indicating three branches: "activity1" (marked with an asterisk), "activity2", and "master". The prompt "aaryan@192 SCM PROJECT %" is visible at the bottom.

It can be clearly seen that activity has turned green

5. Now make some changes in the folder. For example: Make two new txt files named file1.txt and file2.txt  
And make some changes in design.txt



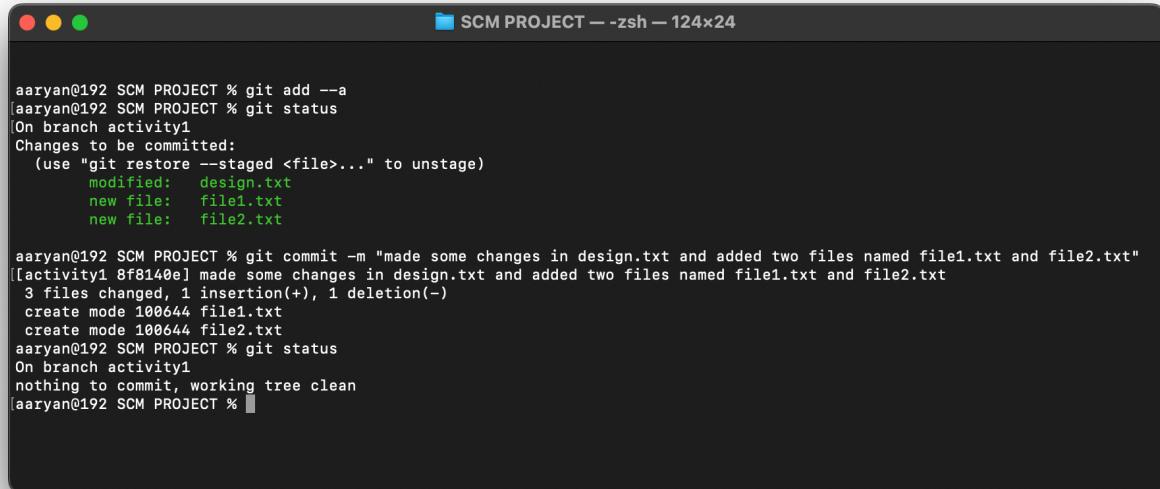


```
[aaryan@192 SCM PROJECT % git status
On branch activity1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   design.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file1.txt
    file2.txt

no changes added to commit (use "git add" and/or "git commit -a")
aaryan@192 SCM PROJECT % ]
```

6. Now we have to add the changes or new files to staging area and then commit them.

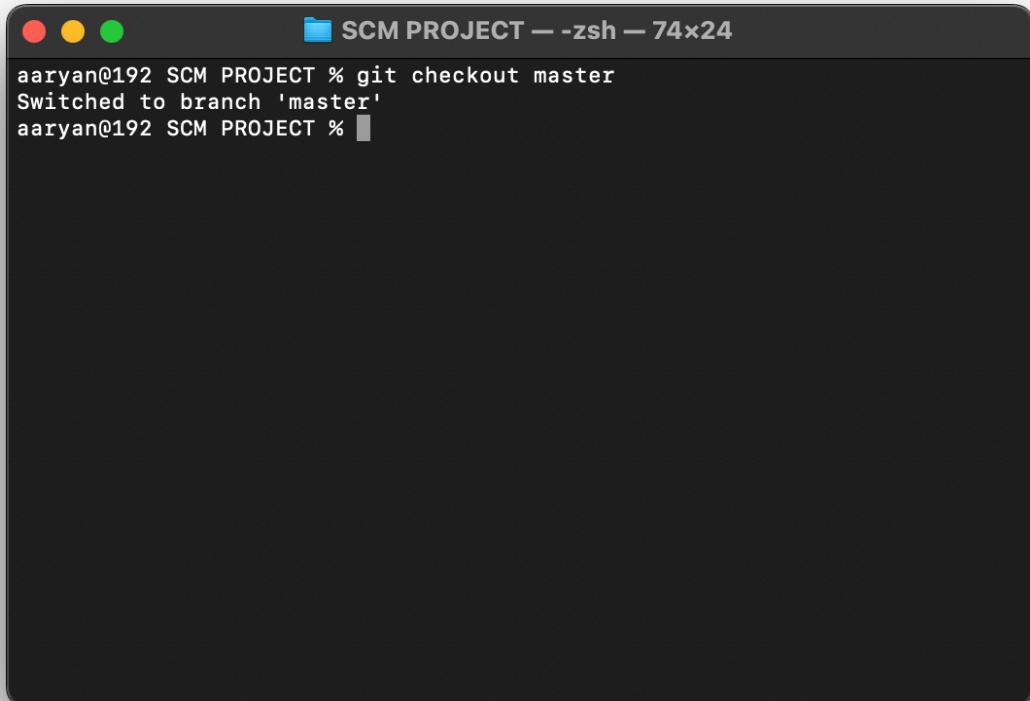


A terminal window titled "SCM PROJECT — zsh — 124x24". The session shows:

```
aaryan@192 SCM PROJECT % git add --a
[aaryan@192 SCM PROJECT % git status
On branch activity1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   design.txt
    new file:   file1.txt
    new file:   file2.txt

aaryan@192 SCM PROJECT % git commit -m "made some changes in design.txt and added two files named file1.txt and file2.txt"
[activity1 8f8140e] made some changes in design.txt and added two files named file1.txt and file2.txt
  3 files changed, 1 insertion(+), 1 deletion(-)
  create mode 100644 file1.txt
  create mode 100644 file2.txt
aaryan@192 SCM PROJECT % git status
On branch activity1
nothing to commit, working tree clean
[aaryan@192 SCM PROJECT %
```

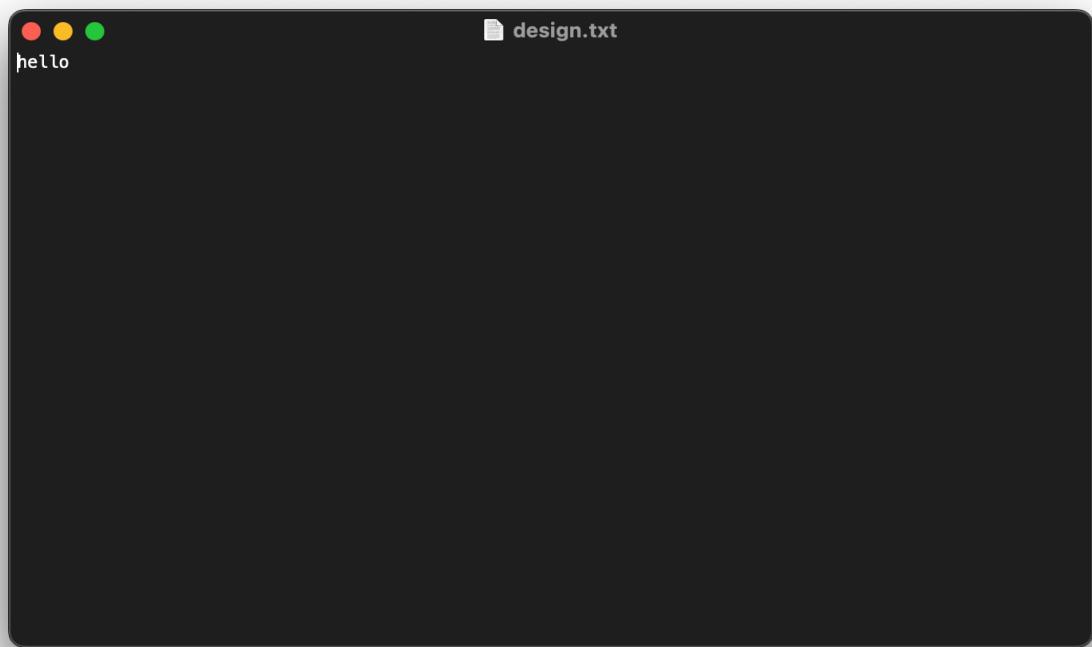
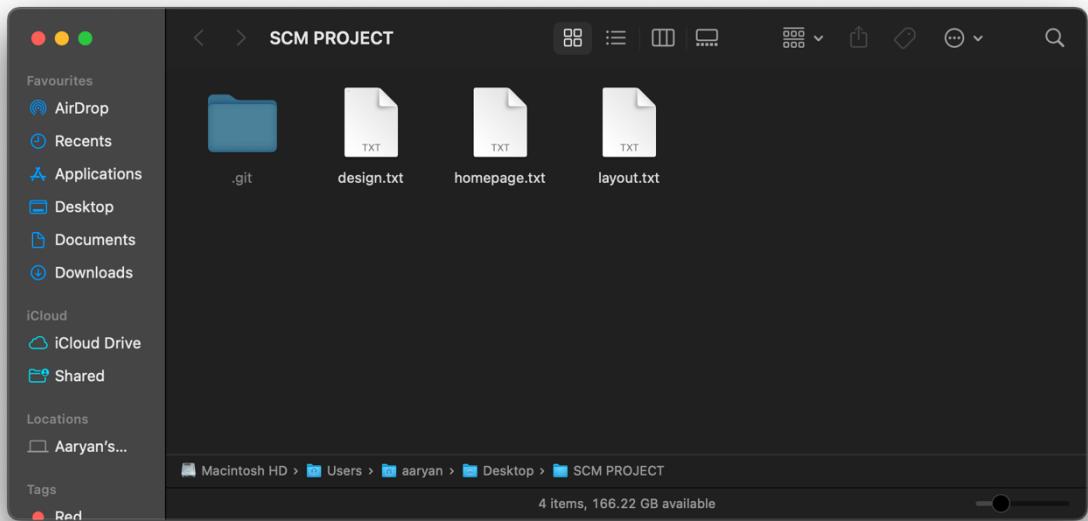
7. Now let's go back to the master branch using the command ***git checkout master***



A terminal window titled "SCM PROJECT — zsh — 74x24". The session shows:

```
aaryan@192 SCM PROJECT % git checkout master
Switched to branch 'master'
aaryan@192 SCM PROJECT %
```

8. Check the SCM Project folder, there are no file1.txt and file2.txt and design file has not changed like we did in activity1 branch.



**So, this proves that any changes done in child branch will not reflect in parent branch after committing the changes.**

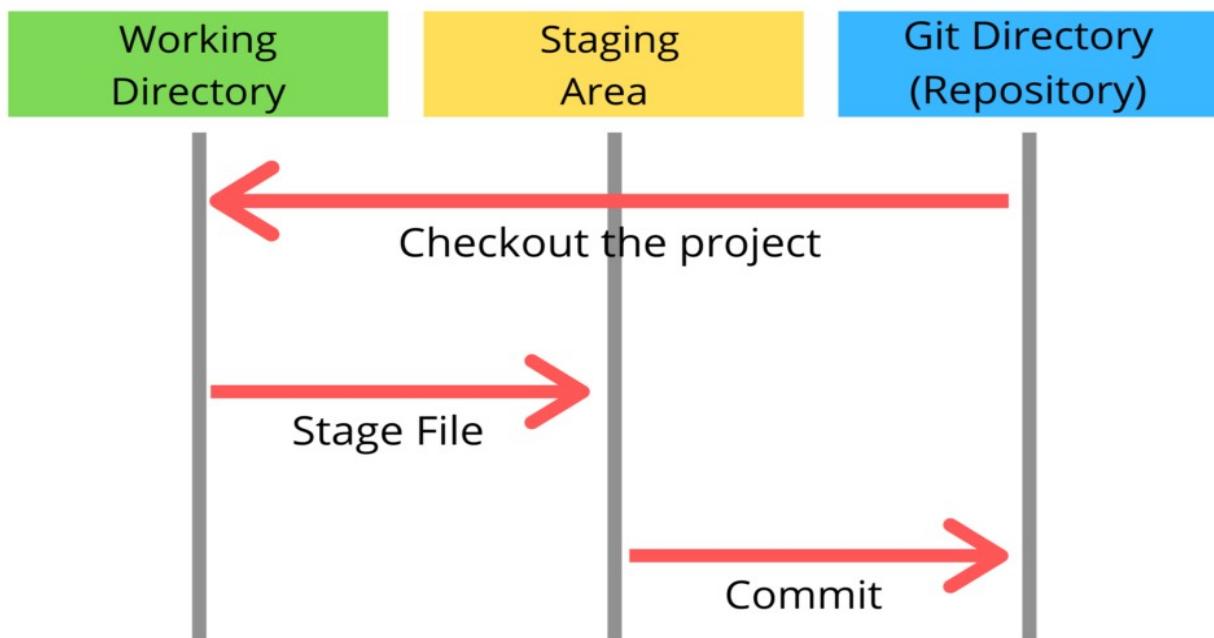
## Experiment No. 05

**Aim:** To describe Git Lifecycle.

Many VCS's use a two-stage architecture i.e., a repository and a working copy. Git uses three-stage architecture i.e., a working directory, staging area and local repository.

Working Directory is the folder which we created i.e., SCM Project.

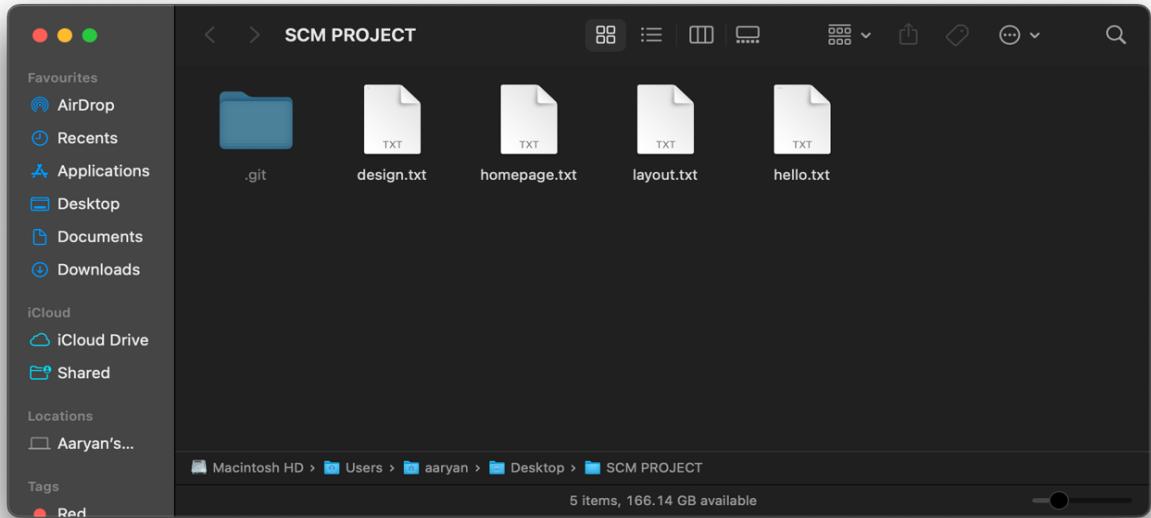
Working Directory was converted into Repository where we used **git init** command to initial the repository.



Working directory specifies the file explorer's folder where your files are stored, Staging area is an area where those files are present which you want to send to commit (to create snapshot of files). After commit is fired, files which are in staging area will move to Git Repository.

Now if you made any changes in your files which are in Git repository, those have to be added into the Staging area and Committed.

1. Create a file in the Git Repository (SCM Project folder). Let's create a .txt file and name it as hello.txt.



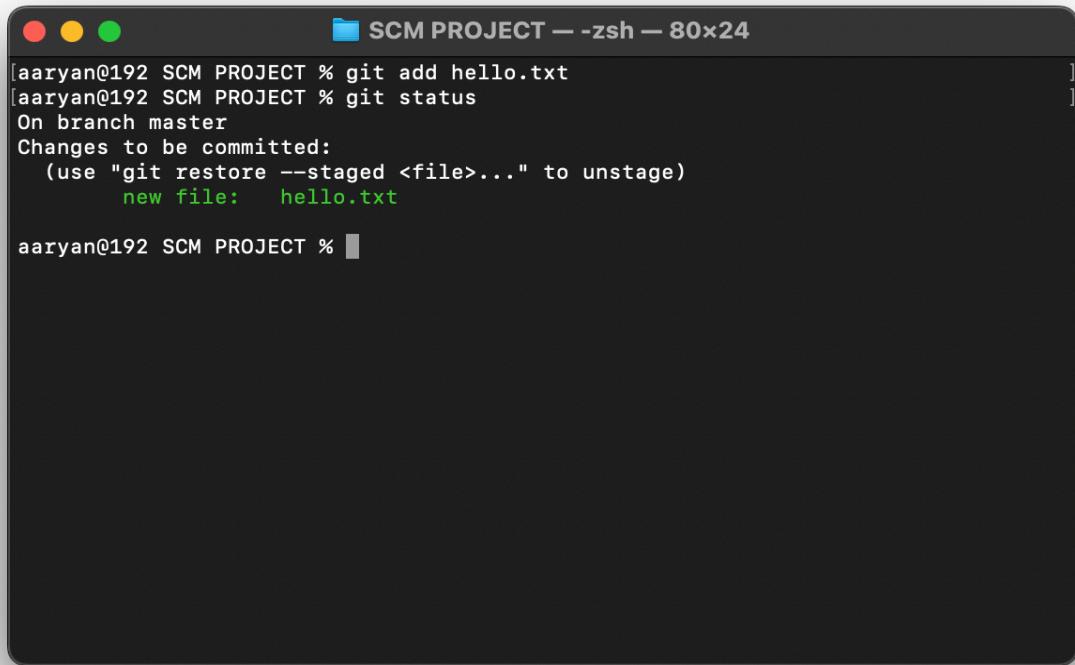
2. Now, open Git Bash and type command *git status*

We can clearly see that the **hello.txt** file is in Untracked files and it is coloured in red.

```
Last login: Sat Apr  9 16:03:58 on ttys000
[aaryan@192 SCM PROJECT % git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    hello.txt

nothing added to commit but untracked files present (use "git add" to track)
aaryan@192 SCM PROJECT %
```

3. Now type ***git add --a*** which will move or add the files to the Staging area. Here **--a** means all the latest files changed or added. You can also specify a file like ***git add hello.txt***.
4. After that check status using ***git status***



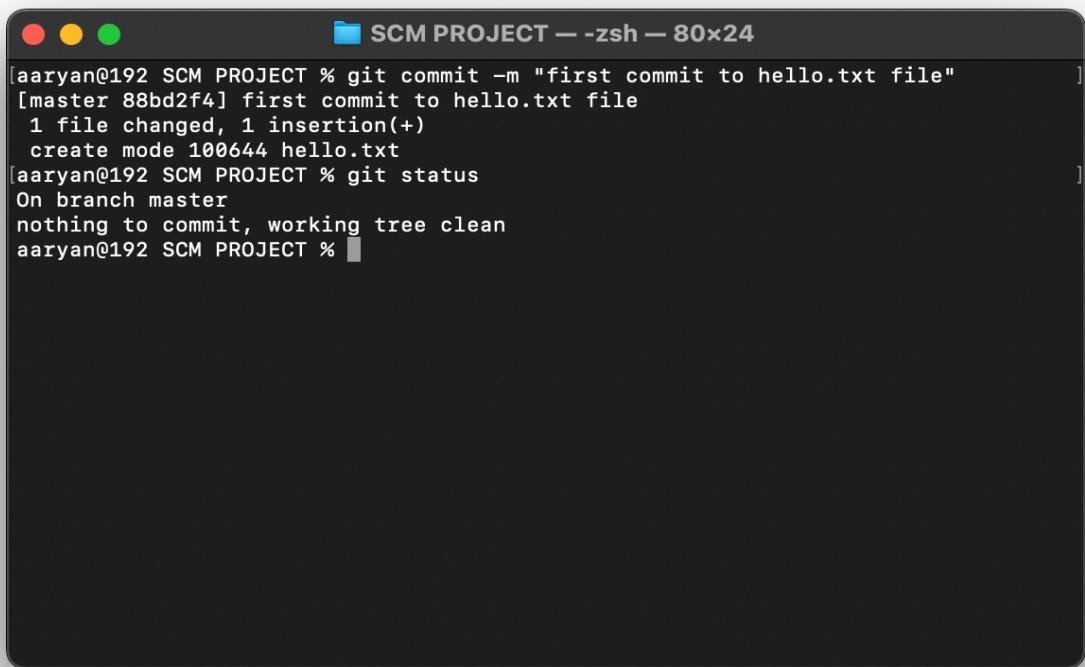
A screenshot of a macOS terminal window titled "SCM PROJECT — -zsh — 80x24". The window shows the following command-line session:

```
[aaryan@192 SCM PROJECT % git add hello.txt
[aaryan@192 SCM PROJECT % git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   hello.txt

aaryan@192 SCM PROJECT % ]
```

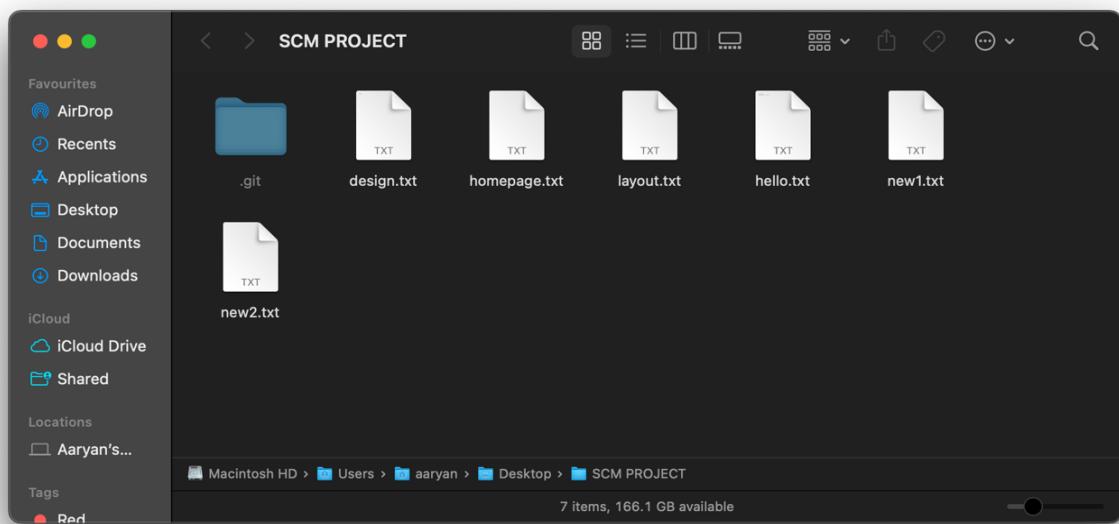
Observe that the **hello.txt** has turned from **red** to **green**. This means that file has been added to the staging area and is ready to be committed.

5. Now to commit the files in the staging area, use command ***git commit -m "Any Message"*** and check the status.

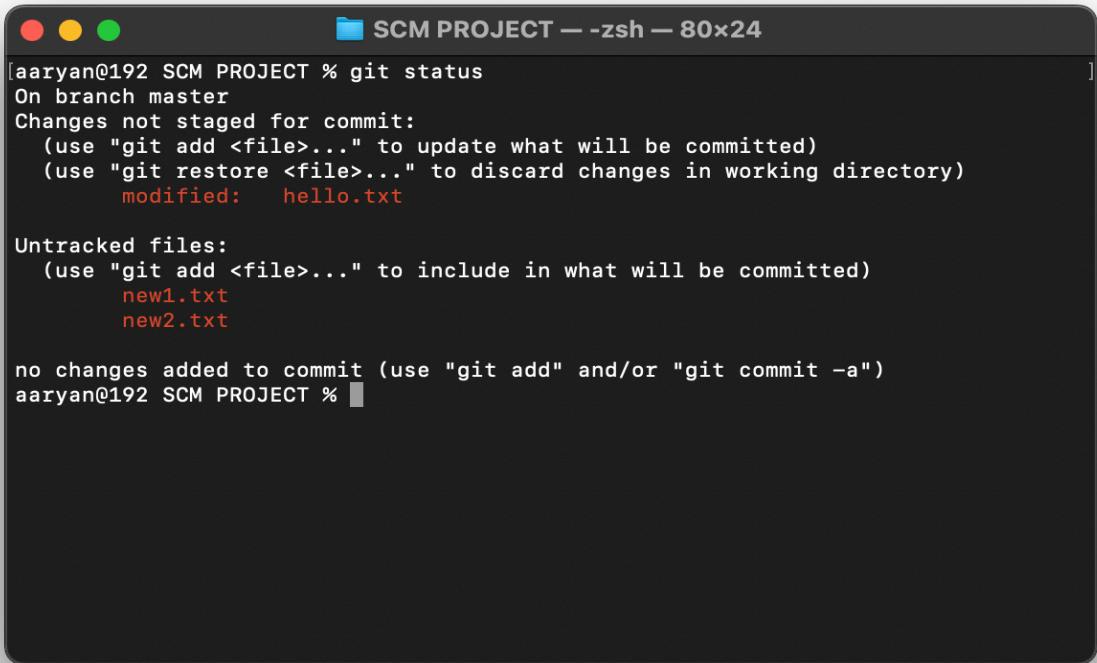


```
SCM PROJECT — -zsh — 80x24
[aaryan@192 SCM PROJECT % git commit -m "first commit to hello.txt file"
[master 88bd2f4] first commit to hello.txt file
 1 file changed, 1 insertion(+)
  create mode 100644 hello.txt
[aaryan@192 SCM PROJECT % git status
On branch master
nothing to commit, working tree clean
aaryan@192 SCM PROJECT %
```

6. Now create two new .txt files named **new1.txt** and **new2.txt** and typed something and changed the **hello.txt** file.



7. Now let's check the status. It shows that hello.txt is modified and new1.txt and new2.txt are untracked. All these changes have to be committed.



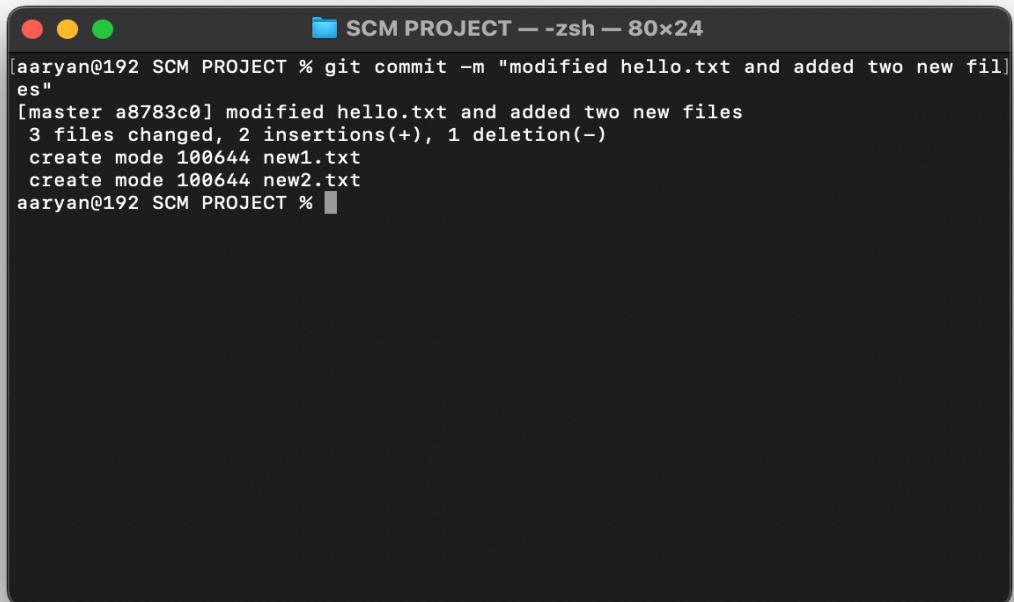
SCM PROJECT — zsh — 80x24

```
[aaryan@192 SCM PROJECT % 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:   hello.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    new1.txt
    new2.txt

no changes added to commit (use "git add" and/or "git commit -a")
aaryan@192 SCM PROJECT % ]
```

8. Now we have to commit these changes.



SCM PROJECT — zsh — 80x24

```
[aaryan@192 SCM PROJECT % git commit -m "modified hello.txt and added two new files"
[master a8783c0] modified hello.txt and added two new files
 3 files changed, 2 insertions(+), 1 deletion(-)
  create mode 100644 new1.txt
  create mode 100644 new2.txt
aaryan@192 SCM PROJECT % ]
```