

TASK 1.1

SOURCE CODE MANAGEMENT

(CS-181)

Submitted By:
Abhijit Singh
Roll no.-2110990041

Submitted To:
Dr. Monit Kapoor

Experiment 1

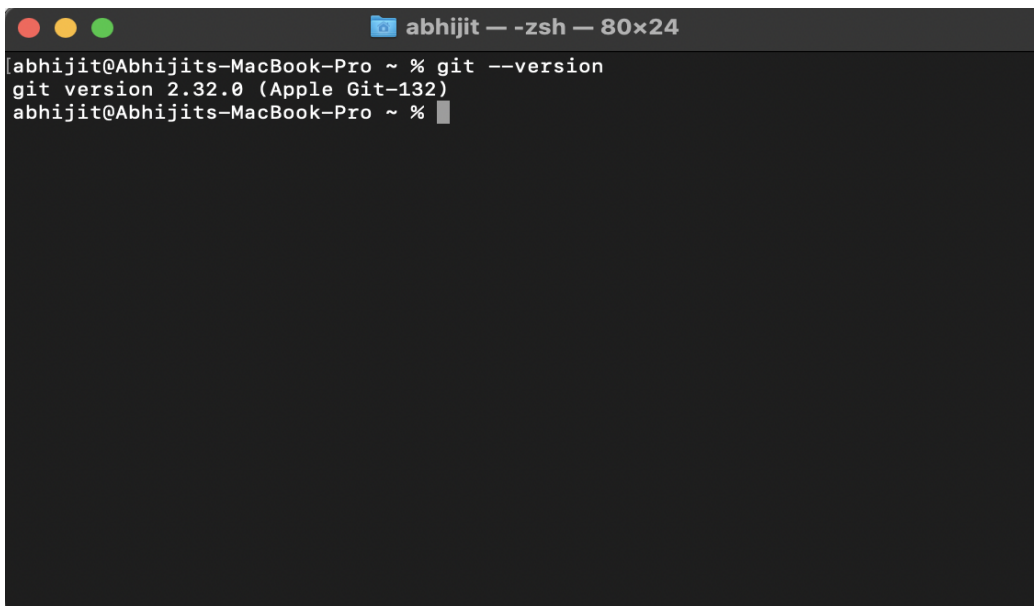
AIM: Setting up of GIT client.

THEORY:

- **GIT:** Git is a distributed, open-source version control system (VCS) that enables you to store code, track revision history, merge code changes, and revert to earlier code versions when needed. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

PROCEDURE:

- On Mac (running OS X or greater) GIT is pre installed.
- Check the version by running **git --version** in the terminal.

A screenshot of a macOS terminal window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left, followed by the text 'abhijit — zsh — 80x24'. The terminal content shows a user prompt '[abhijit@Abhijits-MacBook-Pro ~ %]' followed by the command 'git --version'. The output of the command is 'git version 2.32.0 (Apple Git-132)'. Below the output, the prompt '[abhijit@Abhijits-MacBook-Pro ~ %]' is shown again with a cursor, indicating the command has finished execution.

```
abhijit@Abhijits-MacBook-Pro ~ % git --version
git version 2.32.0 (Apple Git-132)
abhijit@Abhijits-MacBook-Pro ~ %
```

EXPERIMENT 2

AIM: Setting up a GitHub account.

THEORY:

- **GITHUB:** Github is an online software development platform used for storing, tracking, and collaborating on software projects. GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.

PROCEDURE:

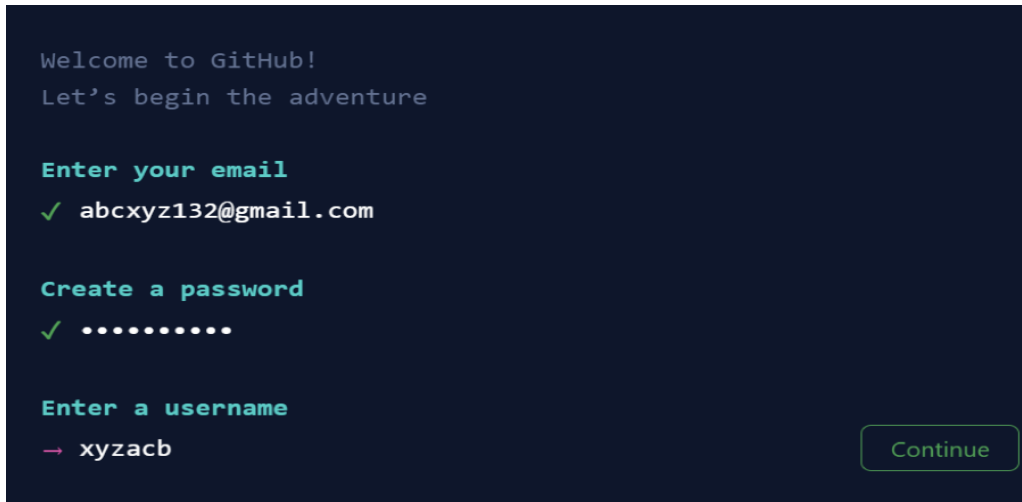
1. Search for GitHub on any search engine.
2. If you already have an account then click on **Sign in** or else



create an account by clicking on the **Sign up** option.

3. To create a new account:

- Click on the Sign **up** option.

A dark-themed screenshot of the GitHub account creation page. At the top, it says "Welcome to GitHub!" and "Let's begin the adventure". Below this, there are three sections: "Enter your email" with a green checkmark and the text "abcxyz132@gmail.com"; "Create a password" with a green checkmark and a series of dots; and "Enter a username" with a red arrow and the text "xyzacb". A green "Continue" button is located at the bottom right.

Welcome to GitHub!
Let's begin the adventure

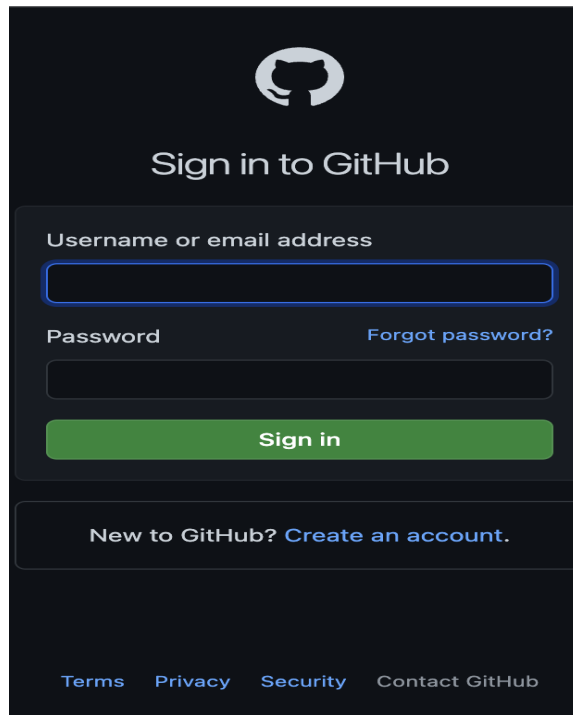
Enter your email
✓ abcxyz132@gmail.com

Create a password
✓

Enter a username
→ xyzacb

Continue

- Enter email address and then enter password.
 - Enter a username and solve the captcha.
 - Then enter the verification code sent on your email and your account will be created.
 - Personalize your account according to your needs.
 - Your account is set up.
4. If you already have an account then click on the Sign **in** option.

A dark-themed screenshot of the GitHub sign-in page. At the top is the GitHub logo and the text "Sign in to GitHub". Below this is a form with two input fields: "Username or email address" and "Password". There is a "Forgot password?" link next to the password field. A green "Sign in" button is at the bottom of the form. Below the form is a link that says "New to GitHub? Create an account.". At the very bottom, there are links for "Terms", "Privacy", "Security", and "Contact GitHub".

Sign in to GitHub

Username or email address

Password

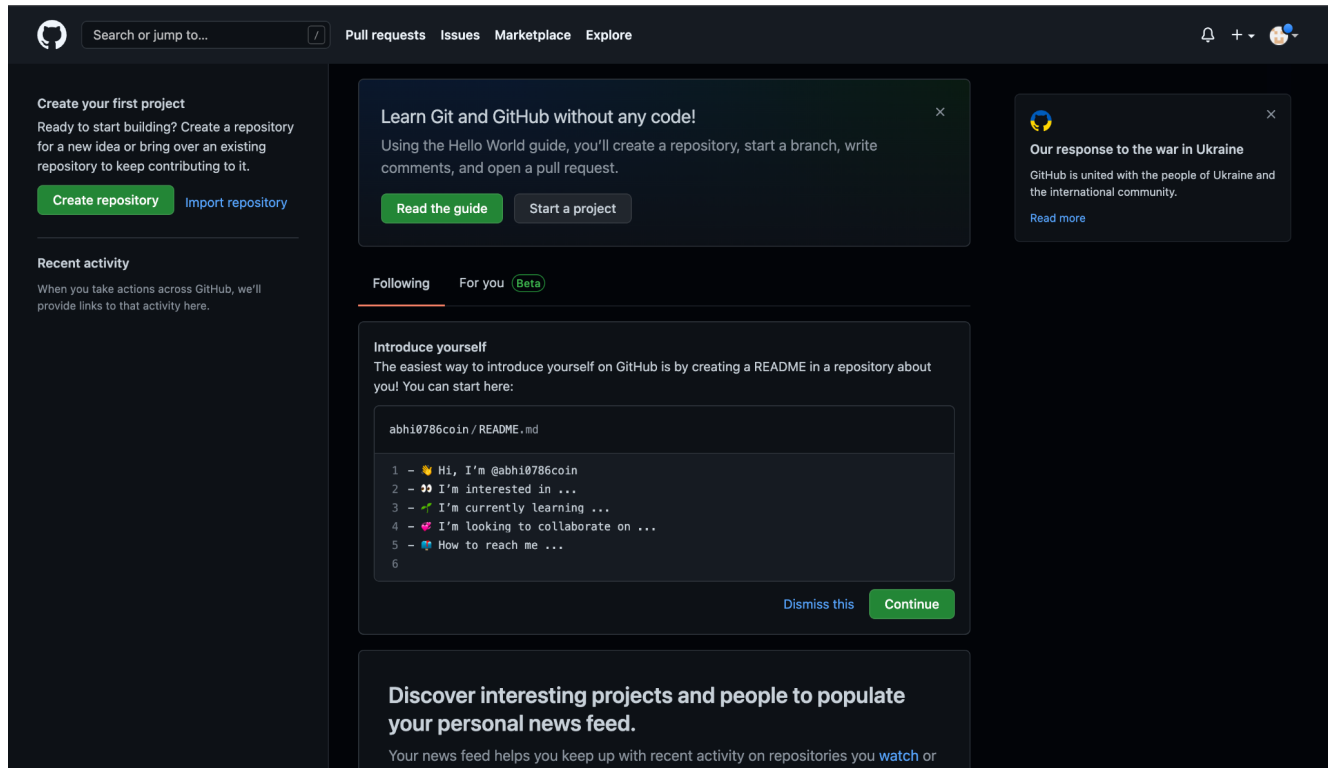
Forgot password?

Sign in

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

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

5. Interface of GitHub:



6. To link GitHub with Git Bash use the following commands.

- For username:
git config --global user.name "<Enter GitHub username>"
- For email:
git config --global user.email "<Enter GitHub email>"

EXPERIMENT 3

AIM: Generating Logs.

THEORY: 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 git log is a record of commits.

PROCEDURE:

1. Start a repo on your machine by:
 - Create a directory on your machine.

```
Last login: Sat Apr  9 18:46:31 on ttys000
[abhiжит@Abhijits-MacBook-Pro ~ % cd desktop
[abhiжит@Abhijits-MacBook-Pro desktop % mkdir task
[abhiжит@Abhijits-MacBook-Pro desktop % cd task
[abhiжит@Abhijits-MacBook-Pro task % touch abc.cpp
[abhiжит@Abhijits-MacBook-Pro task % git status
fatal: not a git repository (or any of the parent directories): .git
abhiжит@Abhijits-MacBook-Pro task %
```

- Initialize a repo and check the status of files. As abc.cpp is in red color, means that it is unstaged.

```
[abhiжит@Abhijits-MacBook-Pro task % git init
hint: Using 'master' as the name for the initial branch. This default branch name
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/abhiжит/Desktop/task/.git/
[abhiжит@Abhijits-MacBook-Pro task % git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
abhiжит@Abhijits-MacBook-Pro task %
```

- Add the file using **git add .** command and then check the status of the file. Now the file name is shown in green which means that the file is staged and ready to commit.

```

abhijit@Abhijits-MacBook-Pro task % git add .
abhijit@Abhijits-MacBook-Pro task % git status
On branch master

No commits yet

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

abhijit@Abhijits-MacBook-Pro task % █

```

2. Make some changes and commit at each step.
3. Run the **git log** command in the terminal to see the commits you made.

```

abhijit@Abhijits-MacBook-Pro task % git commit -m"sample commit"
[master (root-commit) 651aa13] sample commit
  Committer: Abhijit Singh <abhijit@Abhijits-MacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 abc.cpp
abhijit@Abhijits-MacBook-Pro task % git log
commit 651aa1394a7ad79541b4607a67e4f1644e104cea (HEAD -> master)
Author: Abhijit Singh <abhijit@Abhijits-MacBook-Pro.local>
Date:   Sat Apr 9 22:01:09 2022 +0530

    sample commit

```

EXPERIMENT 4

AIM: To create and visualize branches.

THEORY: The main branch in git is called the master branch.
But we can make branches out of this main master branch.

```
abhijit@Abhijits-MacBook-Pro SCM % git branch
* master
abhijit@Abhijits-MacBook-Pro SCM %
```

PROCEDURE:

1. To create a new branch use command:
git branch <name of branch>
2. To check branches use command:
git branch

```
abhijit@Abhijits-MacBook-Pro SCM % git branch
* master
abhijit@Abhijits-MacBook-Pro SCM % git branch example
abhijit@Abhijits-MacBook-Pro SCM % git branch
example
* master
abhijit@Abhijits-MacBook-Pro SCM %
```

3. To change present working branch use command:
git checkout <branch name>

```
abhijit@Abhijits-MacBook-Pro SCM % git checkout example
Switched to branch 'example'
abhijit@Abhijits-MacBook-Pro SCM % git branch
* example
  master
abhijit@Abhijits-MacBook-Pro SCM %
```

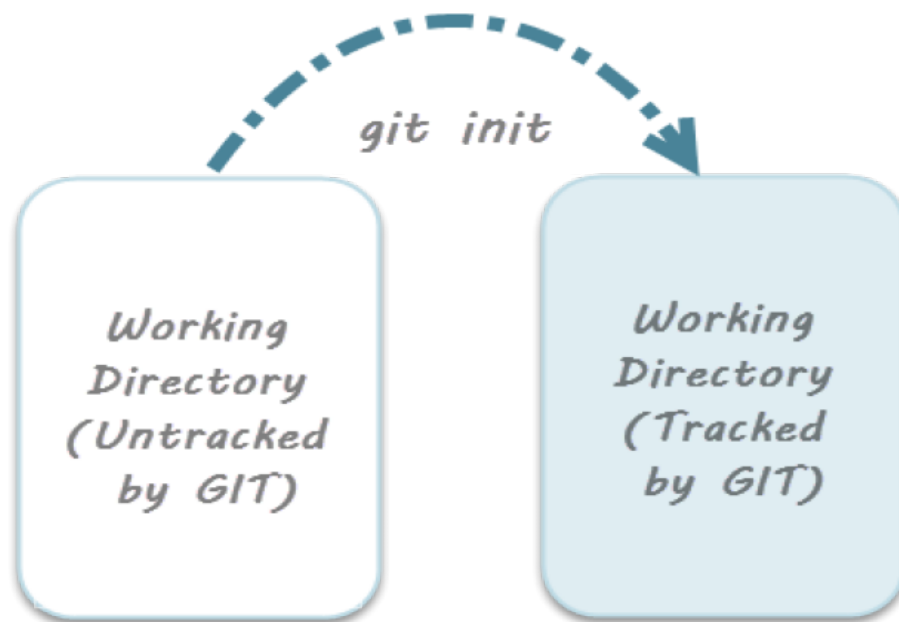

EXPERIMENT 5

AIM: To explain the GIT lifecycle.

THEORY: When a project is under the Git version control system, they are present in three major Git states in addition to these basic ones.

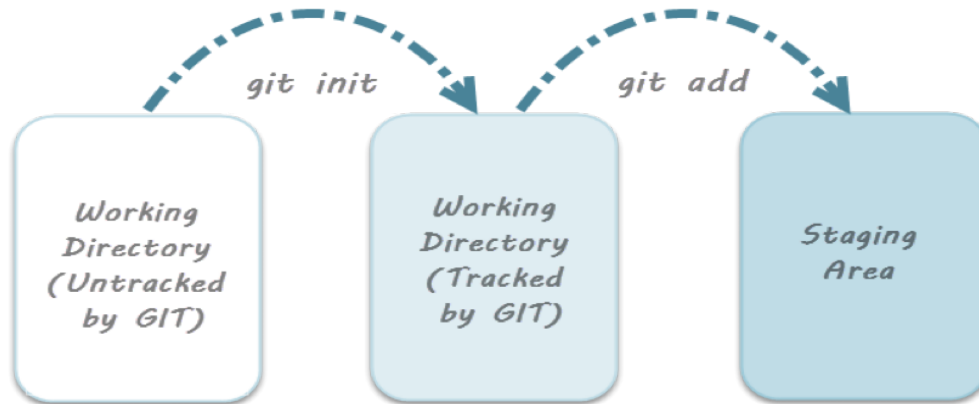
Here are the three Git states:

1. **Working Directory:** The git init command is used to initialize our local project directory to make it a git repository. After running this command, git becomes aware of the files in the project, though it does not yet track them. In the staging area, the files are tracked further.



2. **Staging Area:** To keep track of the various versions of our files, we now use the command git add. A staging area is a location where different versions of our files are kept. The git add command moves your file's version from your working directory to the staging area. We can, however, choose which files to add to the staging area because there are some files in our working directory that we don't want tracked, such as node modules, env

files, temporary files, and so on. Indexing in Git is the process by which Git determines which files need to be added or sent. Your staging area can be found inside the index file in the .git folder.



3. GIT Directory: Now that we have all of the files that need to be tracked ready in the staging area, we can commit them using the `git commit` command. Commit assists us in keeping track of the metadata associated with the files in our staging area. Every commit is accompanied by a message that describes what the commit is about. Git saves the information or metadata of the files that were committed in a Git Directory, which aids Git in file tracking and essentially saves a photocopy of the committed files. Commit also stores the name of the author who made the commit, the files that were committed, and the date they were committed, in addition to the commit message.

