

Source Code Management (CS181)

Cluster - Beta

Department - CSE

Submitted By - Ayushman Dixit

Group - G8

Roll no. - 2110990351

SCM Task File

Submitted To - Dr Monit Kapoor(Dean)

Task 1.1

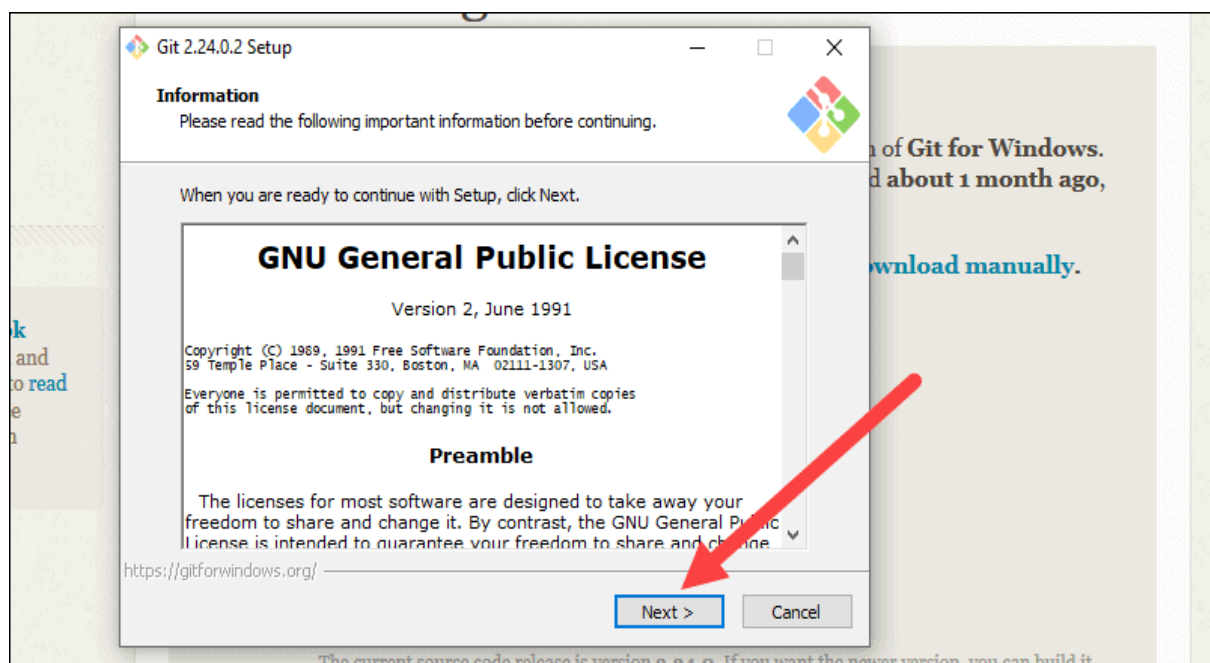
Experiment no.- 1

Installing Git

1. Browse the official Git website: <https://git-scm.com/downloads>
2. Click the download link for Windows and allow the download to complete.



3. Review the GNU General Public License, and when you're ready to install, click Next.



4. When running the installer, various screens appear (Windows screens shown). Generally, you can accept the default selections, unless you have reason to change it, and click Next.

5. Once the installation is complete, tick the boxes to view the Release Notes or Launch Git Bash, then click Finish.



First-Time Git Setup

Now that you have Git on your system, let's set up the Git environment. Git comes with a tool called `git config` that lets you get and set configuration variables that control all aspects of how Git looks and operates.

The first thing you should do when you install Git is to set your user name and email address. This is important because every Git commit uses this information, and it's immutably baked into the commits you start creating:

```
$ git config --global user.name "John Doe"
```

```
$ git config --global user.email johndoe@example.com
```

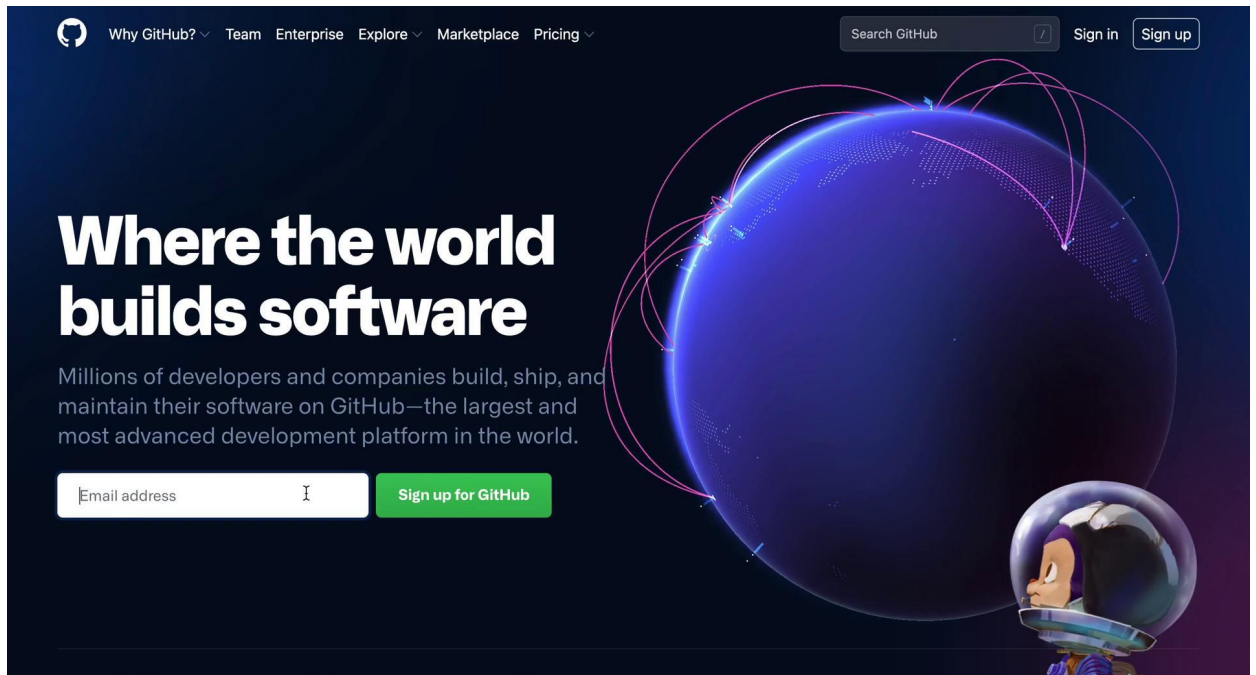
Again, you need to do this only once if you pass the `--global` option, because then Git will always use that information for anything you do on that system. If you want to override this with a different name or email address for specific projects, you can run the command without the `--global` option when you're in that project.

Many of the GUI tools will help you do this when you first run them.

Experiment - 2

Setting up Github Account

1. Go to <https://github.com>.



2. Click on signup

3. Fill out your email and make a strong password



Sign in to GitHub

Username or email address

Password

[Forgot password?](#)

Sign in

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

4. Complete the CAPTCHA.
5. Click Choose on your desired plan but for the first time, you can skip this.
6. Verify your email address.
7. Select your preferences.

Experiment - 3

Program to generate logs

Basic Git init :

The Git init command creates a new Git repository. It can be used to convert an existing, unsigned project to a Git repository or initialize a new, empty repository. Most other Git commands are not available outside of an initialized repository, so this is usually the first command you'll run in a new project.

Basic Git status:

The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git. The status output does not show you any information regarding the committed project history.

Basic Git add command:

The git add command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit. However, git add doesn't really affect the repository in

any significant way changes are not actually recorded until you run git commit.

Basic Git commit:

The git commit command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as “safe” versions of a project Git will never change them unless you explicitly ask it to. Prior to the execution of the git commit, The git add command is used to promote or 'stage' changes to the project that will be stored in a commit. These two commands git commit and git add are two of the most frequently used.

Basic Git log:

The Git log command is one of the most usual commands of git. It is the most useful command for Git. Every time you need to check the history, you have to use the git log command. The basic git log command will display the most recent commits and the status of the head.

Experiment - 4

Create and visualize branches in Git

How to create branches?

The main branch in git is called a master branch. But we can make branches out of this main master branch. All the files present in the master can be shown in the branch but the files which are created in the branch are not shown in the master branch. We also can merge both the parent(master) and child (other branches).

1. For creating a new branch: git branch "name of the branch".
2. To check how many branches we have: git branch.
3. To change the present working branch: git checkout "name of the branch".

Visualizing Branches:

To visualize, we have to create a new file in the new branch "activity1" instead of the master branch. After this, we have to do three-step architecture i.e. working directory, staging area and git repository.

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

After this, we will change the branch from activity1 to master, but when we switch back to the master branch the file we created i.e “hello” will not be there. Hence the new file will not be shown in the master branch. In this way, we can create and change different branches. We can also merge the branches by using the git merge command.

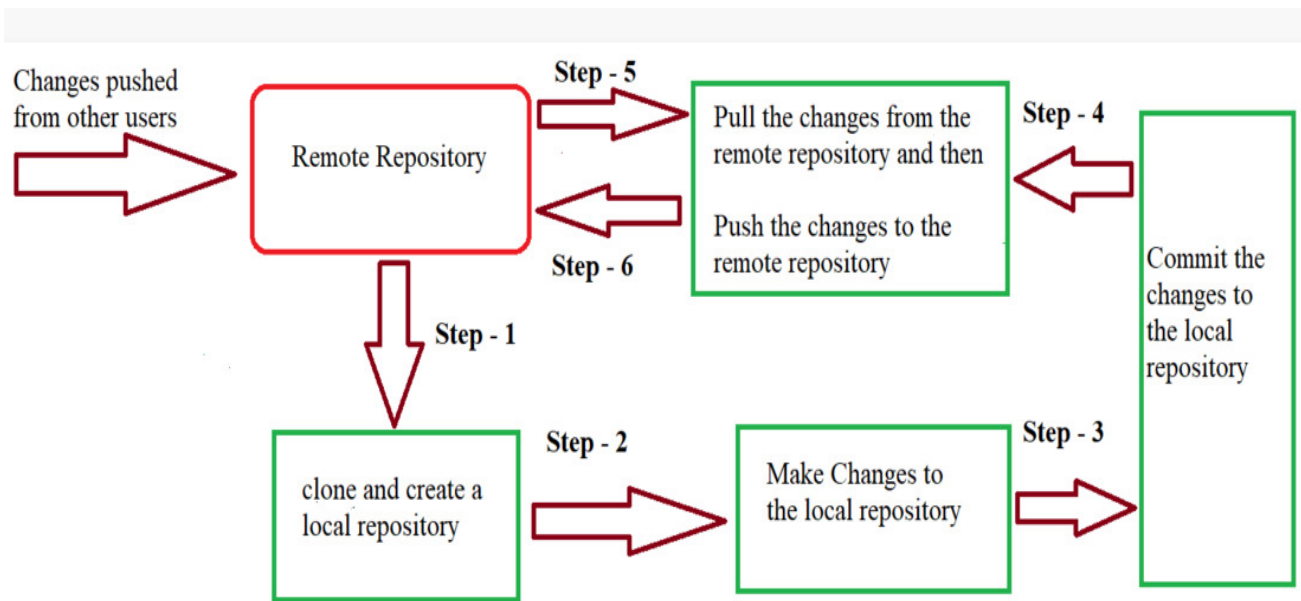
In this way, we can create and change different branches. We can also merge the branches by using git merge command.

Experiment - 5

Git lifecycle description

Git is used in our day-to-day work, we use Git for keeping a track of our files, working in a collaboration with our team, to go back to our previous code versions if we face some error. Git helps us in many ways. Let us look at the Lifecycle description that git has and understand more about its life cycle. Let us see some of the basic steps that we have to follow while working with Git.

- You clone the Git repository as a working copy.
- You modify the working copy by adding/editing files.
- If necessary, you also update the working copy by taking other developers' changes.
- You review the changes before committing.
- You commit changes. If everything is fine, then you push the changes to the repository.
- After committing, if you realize something is wrong, then you correct the last commit and push the changes to the repository.



When a directory is made into a git repository, there are mainly 3 states which make up the essence of the Git Version Control System. The three states are:

- Working Directory
- Staging Area
- Git Directory

1. Working Directory

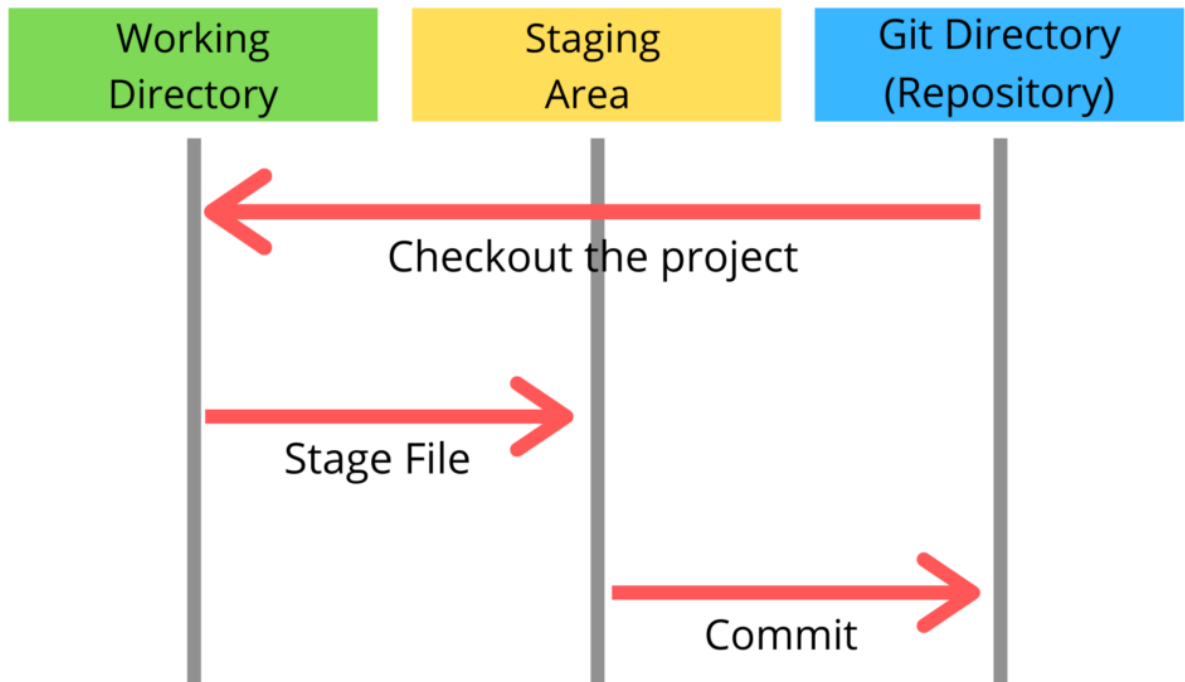
Whenever we want to initialize our local project directory to make it a git repository, we use the **git init** command. After this command, git becomes aware of the files in the project although it doesn't track the files yet. The files are further tracked in the staging area.

2. Staging Area

Now, to track the different versions of our files we use the command ***git add***. We can term a staging area as a place where different versions of our files are stored. ***git add*** command copies the version of your file from your working directory to the staging area. We can, however, choose which files we need to add to the staging area because in our working directory there are some files that we don't want to get tracked, examples include node modules, env files, temporary files, etc. Indexing in Git is the one that helps Git in understanding which files need to be added or sent. You can find your staging area in the ***.git*** folder inside the ***index*** file.

3. Git Directory

Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the ***git commit*** command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about. Git preserves the information or the metadata of the files that were committed in a Git Directory which helps Git in tracking files and basically it preserves the photocopy of the committed files. Commit also stores the name of the author who did the commit, files that are committed, and the date at which they are committed along with the commit message.



Task 1.2

Experiment - 6

Add collaborators on GitHub Repo

1. Create a New Repository. A repository is like a folder for your project. You can have any number of public and private repositories in your user account. Repositories can contain folders and files, images, videos, spreadsheets, data sets, and the revision history for all files in the repository. For more information, see "About repositories." When you create a new repository, you should initialize the repository with a README file to let people know about your project. For more information, see "Creating a new repository."

/

Pull requestsIssuesMarketplaceExplore

+

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner *

Repository name *

Ayushman0351 ▾

/ Scm project ✓

Great repository names are short and lowercase. Your new repository will be created as Scm-project. It solid-lamp?

Description (optional)

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

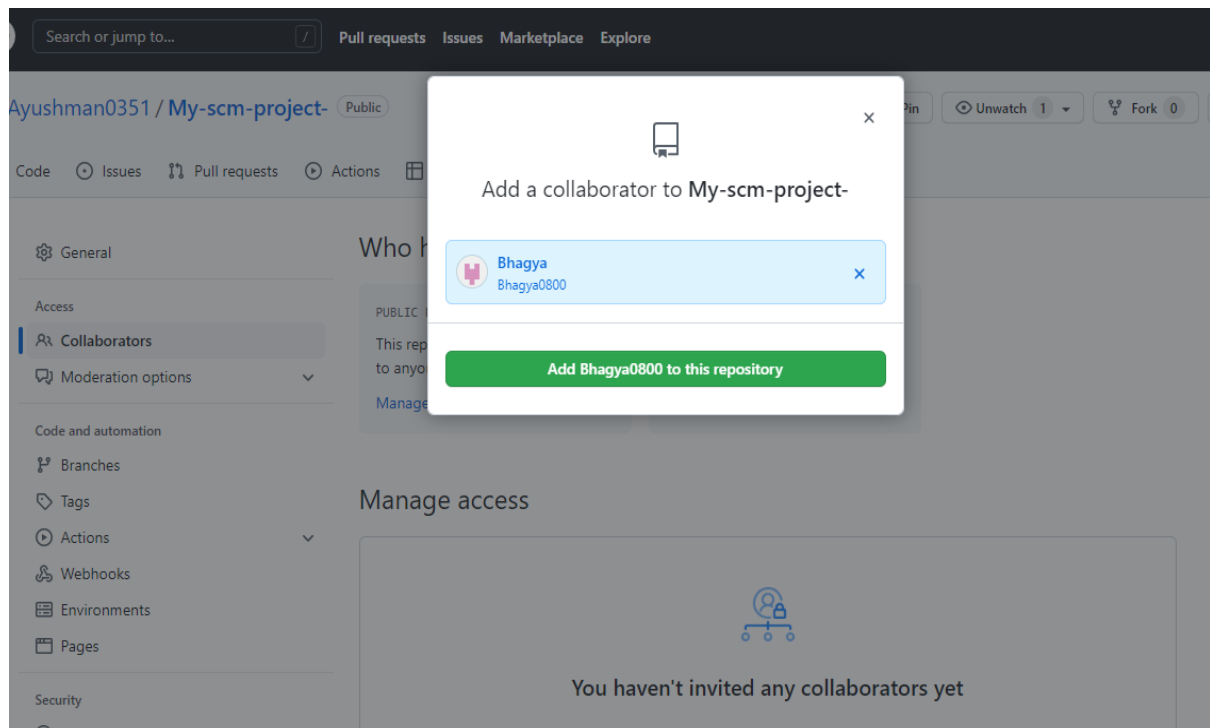
Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

...

2. Go to Collaborators in Repo Setting, Add the username or email of the Collaborator you want to add to your Repo.



3. Invitation Mail is sent to the Collaborator, the collaborator has to accept this Invitation.

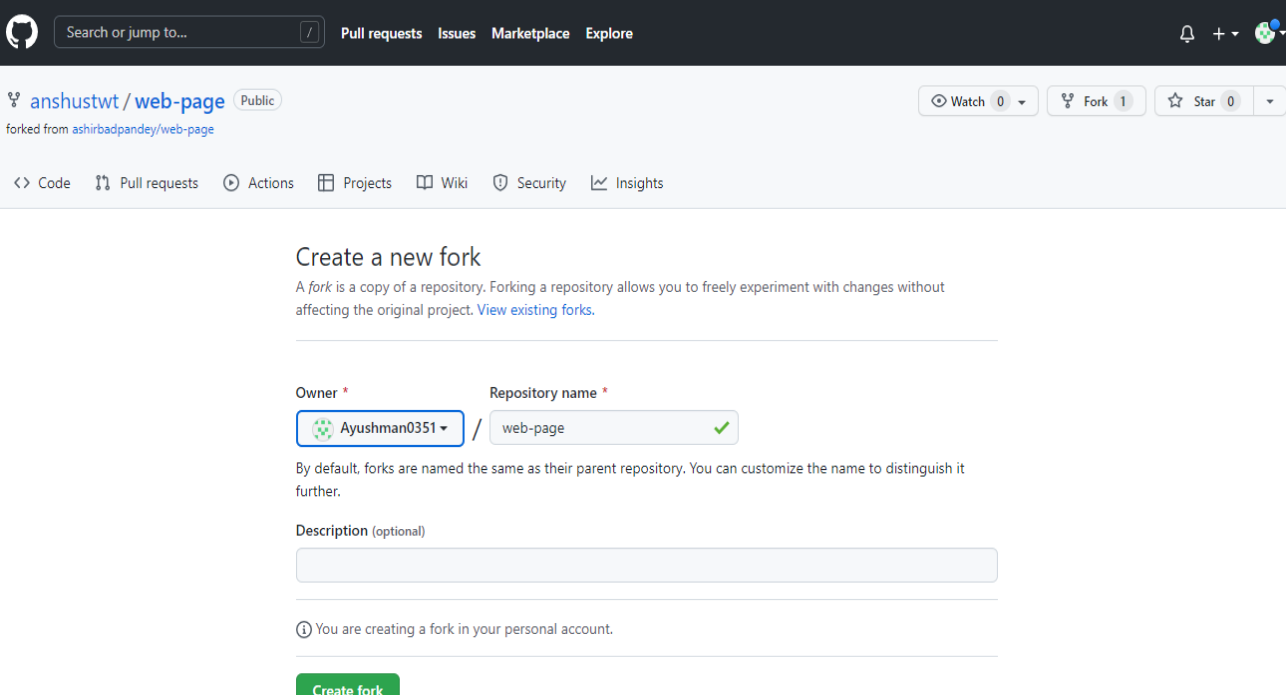
4. The new collaborator has now access to your repo.

Experiment - 7

Fork and Commit

Fork: A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original Project.

1. To fork a repository first thing, you need to do is, sign in to your GitHub account and then you came to the repository you want to fork, so here for demo purposes am using the repository.



The screenshot shows the GitHub interface for creating a new fork of a repository. At the top, the GitHub navigation bar is visible with the search bar and links for Pull requests, Issues, Marketplace, and Explore. Below this, the repository header shows 'anshustwt / web-page' as a public repository, with options to Watch (0), Fork (1), and Star (0). The main content area is titled 'Create a new fork' and includes a brief explanation of forking. The form fields for 'Owner' (Ayushman0351) and 'Repository name' (web-page) are filled out, with a green checkmark indicating the name is valid. A description field is also present but empty. At the bottom, a green 'Create fork' button is visible.

Search or jump to... Pull requests Issues Marketplace Explore

anshustwt / web-page Public Watch 0 Fork 1 Star 0

forked from ashirbadpandey/web-page

<> Code Pull requests Actions Projects Wiki Security Insights

Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Owner * Repository name *

Ayushman0351 / web-page ✓

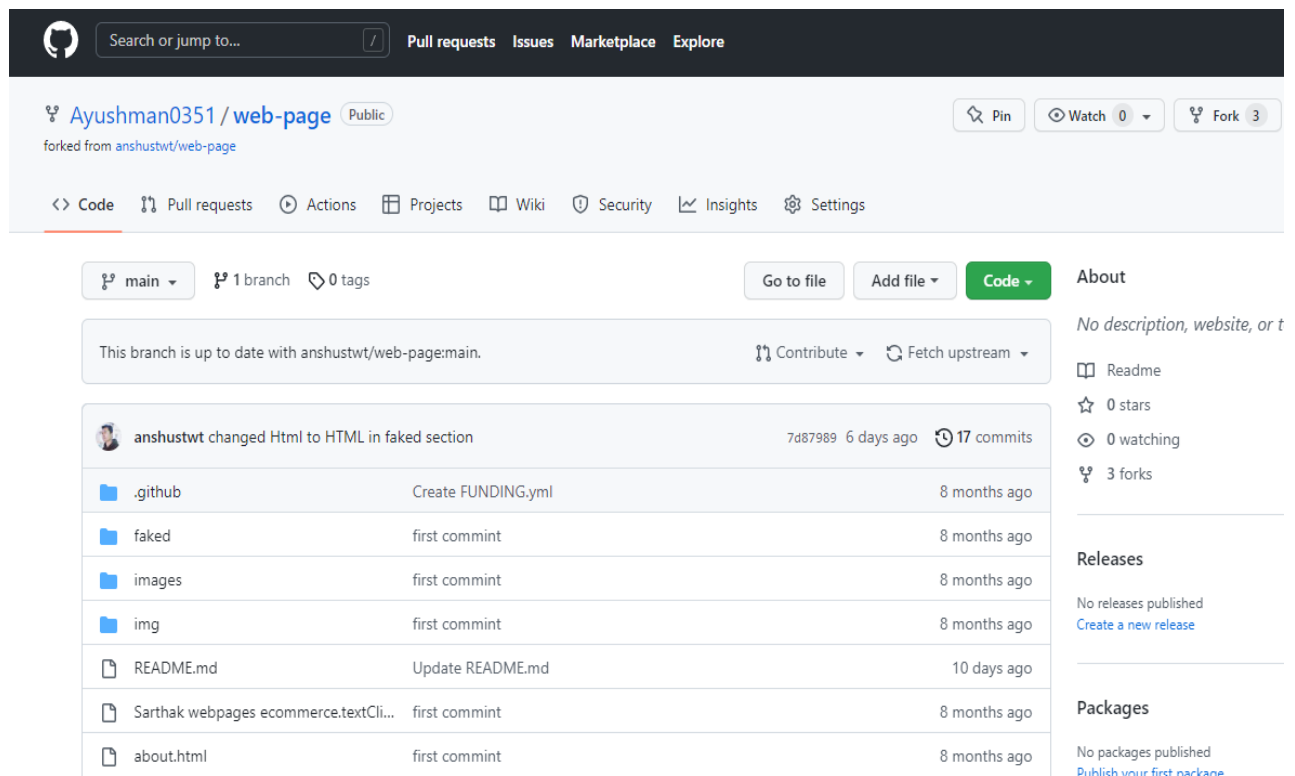
By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.

Description (optional)

ⓘ You are creating a fork in your personal account.

Create fork

2. Click on the Fork button on the right upside corner. Then it will ask to create a new fork, add a description if you want and then click on create a fork.



3. Now you will have a copy of the repo you have forked from another user. Now you can do any modification you want without making changes to the main source code.

3. Now type <https://github.com/Ayushman0351/web-page.git> git on CLI.

Git pull --> This command is used to fetch the remote repo or to clone the repo.

4. Now Open the file make changes to it and commit it and push it to remote.

5. Now go to GitHub and accept the merge request.

Experiment -8

Merge and Resolve Conflicts.

1. Do changes in the master branch and commit those changes. And check out to a different branch and again do changes and commit it. Now checkout to the master branch and merge that branch into the master.

```
MINGW64; c:/Users/HP/desktop/Fork
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (master)
$ cat >fork.txt

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (master)
$ cat >fork.txt
hello
hi bye bye
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (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:   fork.txt

no changes added to commit (use "git add" and/or "git commit -a")
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (master)
$ git add fork.txt
warning: LF will be replaced by CRLF in fork.txt.
The file will have its original line endings in your working directory
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (master)
$ git commit -m"Commit"
[master 0d12b98] Commit
1 file changed, 2 insertions(+), 7 deletions(-)
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (master)
$ |
```

COMMIT TO ANOTHER BRANCH

```

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (activity)
$ cat >fork.txt
lion
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (activity)
$ git status
On branch activity
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:   fork.txt

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

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (activity)
$ git add fork.txt

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (activity)
$ git commit -m"changes made"
[activity 270ae04] changes made
1 file changed, 1 insertion(+), 2 deletions(-)

```

2. Now try to merge it will give a conflict error.

```

$ git merge activity
Auto-merging file1.txt
CONFLICT (content): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.

```

3. Use Command “git merge tool” to solve the conflict. git - merge tool – Run merge conflict resolution tools to resolve merge conflicts.

4. Press “l” to insert, after insertion. Press “:wq”. The merge conflict is solved and our activity branch is merged with the master branch.

Experiment - 9

Reset and Revert

Git-revert – Revert some existing commits.

1. On Git Bash CLI, Type the command “git revert”. It reverts the changes that were done before Commit.

git revert HEAD~3:- Revert the changes specified by the fourth last commit in HEAD and create a new commit with the reverted changes

Git-reset - Reset current HEAD to the specified state

2. At a surface level, git reset is similar in behaviour to git checkout. Where git checkout solely operates on the HEAD ref pointer, git reset will move the HEAD ref pointer and the current branch ref pointer. To better demonstrate this behaviour, consider the following example. This example demonstrates a sequence of commits on the main branch. The HEAD ref and main branch ref currently point to commit d. Now let us execute and compare, both git checkout b and git reset b.



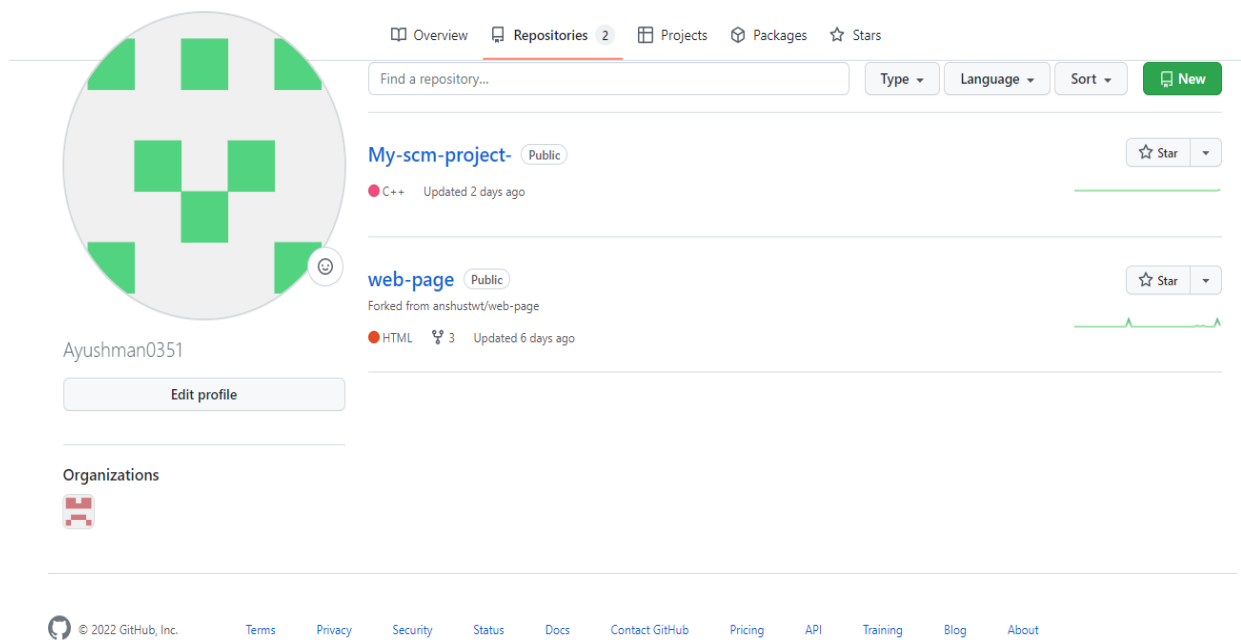
GIT RESET:

1. reset is the command we use when we want to move the repository back to a previous commit, discarding any changes made after that commit.

```
$ git reset --hard 7b98ccfc31e76b87eff7dbd7fed0275a14033d15  
HEAD is now at 7b98ccf hemlo
```

Task 2 - Project Work

1. Log into your Github account and you will land on the homepage as shown below. Click on the Repositories option in the menu bar.



2. Enter the Repository name and add the description of the repository. Select if you want the repository to be public or private.

Pull requests

Issues

Marketplace

Explore

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner *

Repository name *

Ayushman0351

/

Scm project

Great repository names are short and lowercase. Your new repository will be created as Scm-project. It solid-lamp?

Description (optional)

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

3. Now add collaborators in the repository.

Search or jump to...

Pull requests

Issues

Marketplace

Explore

Ayushman0351 / My-scm-project Public

Code

Issues

Pull requests

Actions

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Actions

Webhooks

Environments

Pages

Security

Who has access

Manage access

Add a collaborator to My-scm-project

Bhagya

Bhagya0800

Add Bhagya0800 to this repository

You haven't invited any collaborators yet

4. Now the collaborators will accept the invite that was sent to them.

5. To open a pull request we first have to make a new branch, by using the git branch name option. After making a new branch we add a file to the branch or make changes to the existing file. Add and commit the changes to the local repository.

6. Use the git push origin branch-name option to push the new branch to the main repository.

7. Now collaborator makes changes and creates a pull request.

8. Now after opening a pull request all the team members will be sent the request if they want to merge or close the request. If the team member chooses not to merge your pull request, they will close your pull request. To close the pull request simply click on the close pull request.

9. Now Do the required changes in the repository, add and commit these changes in the local repository in a new branch. Push the modified branch using git push -u origin branch name. Open a pull request. The pull

request will be created and will be visible to all the team members.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base repository: anshustwt/web-page ▾


base: main ▾

←

head repository: Ayushman0351/web-page ▾

compare: main ▾








✓ **Able to merge.** These branches can be automatically merged.



Update about.html


Write

Preview

H B I  <>     @  

I am editing this as a demo.

Attach files by dragging & dropping, selecting or pasting them.

☒ Allow edits by maintainers 

Create pull request ▾

10. Click on the pull request. Two options will be available, either to close the pull request or Merge the request with the main branch. By selecting the merge branch option the main branch will get updated for all the team members. By selecting close the pull request the pull request is not accepted and not merged with the main branch.

11. If the file is merged it will show a message to you.

12. Network graphs need to be recorded.

Excluding merges, **1 author** has pushed **1 commit** to master and **1 commit** to all branches. On master, **0 files** have changed and there have been **0 additions** and **0 deletions**.

