****

Subject Name: **Source Code Management**

Subject Code: **24CSE0106**

Cluster: **iAlpha**

Department: **DCSE**

Submitted By: Submitted To:

Name – Mansha Name – Dr. Sunil Gupta

2410991373 (Associate Professor)

G - 17

**Department of Computer Science and Engineering**

**Chitkara University, Rajpura**

**#IndexTask1.1#**

|  |  |  |
| --- | --- | --- |
| **S.**  **No** | **ProgramTitle** | **PageNo.** |
| **1.** | To install and configure Git Client on your local system. | 3 - 5 |
| **2.** | Setting up GitHub Account and Adding Collaborators on GitHub Repository. | 6 - 10 |
| **3.** | To merge two branches within a Git repository. | 11 - 14 |
| **4.** | To demonstrate push and pull operations in Git. | 15 - 17 |

**Practical 1: Setting up Git Client**

* **Aim:** To install and configure Git Client on your local system.
* **Theory:** Git is a distributed version control system used to track changes in source code. This practical focuses on setting up Git on your local system for effective version control.
* **Procedure:**

1.Download Git from git-scm.com.

2. Install Git by following the setup wizard.

3. Open Git Bash and verify installation using the command: git --version.

4. Configure user details using the commands: git config --global user.name "Your

Name" git config --global user.email "Your Email"

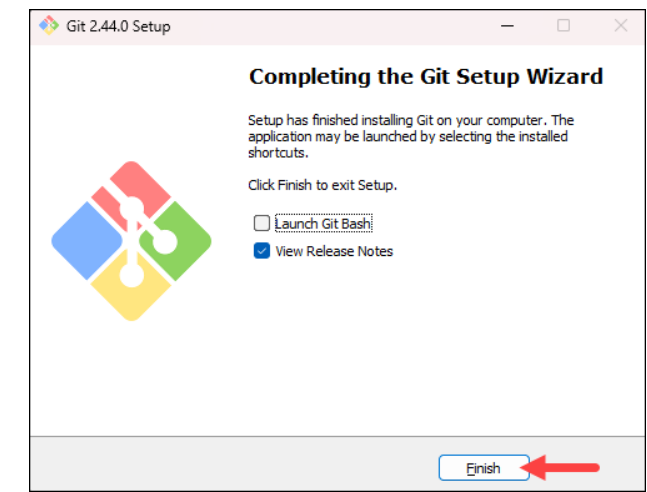
* **Solution:**

1. **Download Git from git-scm.com**

* Visit git-scm.com in your browser.
* Click on the "Download" button, and it will automatically detect your operating system (Windows, macOS, or Linux).

1. **Install Git Using the Setup Wizard**

* After downloading, run the installer (the .exe file on Windows or .dmg on macOS).
* Follow the Git Setup Wizard through the installation process. Some important choices during installation.
* Choose the default editor for Git(pick your preferred text editor, e.g., VSCode, Vim, or Notepad++ on Windows).
* Adjust your PATH environment(choose the recommended option).
* Choose HTTPS transport backend(choose "Use the OpenSSL library").
* The rest of the options can generally be left as default**.**



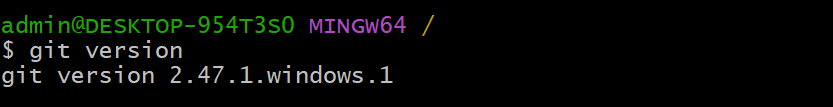
**3. Verify Git Installation Using git --version in Git Bash**

* Once Git is installed, open Git Bash. On Windows, you can find it in the Start menu by searching for **"Git Bash".**
* In the Git Bash terminal, type the following command and press Enter:

git --version This command will display the version of Git installed, something like:

git version 2.34.1.windows.1

**Take a screenshot of this output to include in your task.**

****

**4. Configure User Details**

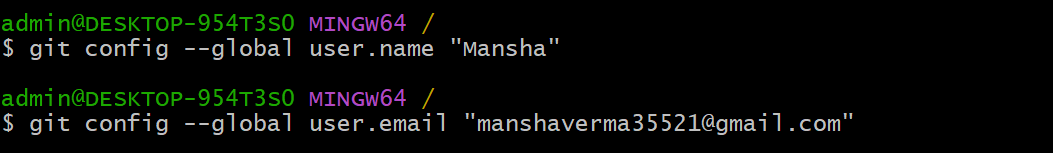
To configure your Git user details, use the following commands in Git Bash. Replace

"Your Name" and "Your Email" with your actual name and email address.

**Configure Git Username:**

git config --global user.name "Your Name"

**Configure Git Email:**

git config --global user.email "your-email@example.com”

**Practical 2: Setting up GitHub Account**

* **Aim:**  Setting up GitHub Account and Adding Collaborators on GitHub Repository.
* **Theory:**  Whenever you make a repository in GitHub, not everyone has the permission to change or push codes into your repository. The users have a read-only access. In order to allow other individuals to make changes to your repository, you need to invite them to collaborate to the project.

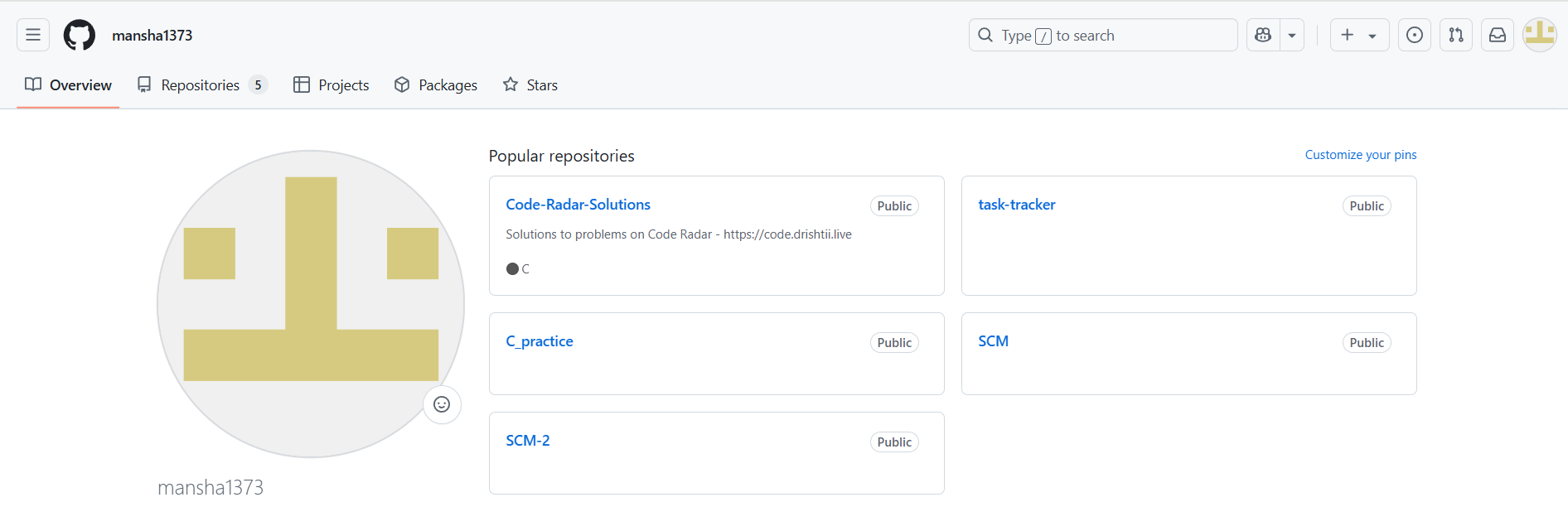
GitHub also restricts the number of collaborators we can invite within a period of 24 hours. If we exceed the limit, then either we have to wait for 24-hours or we can also create an organization to collaborate with more people.

Being a collaborator, the user can create, merge and close pull requests in the repository. They can also remove them as the collaborator.

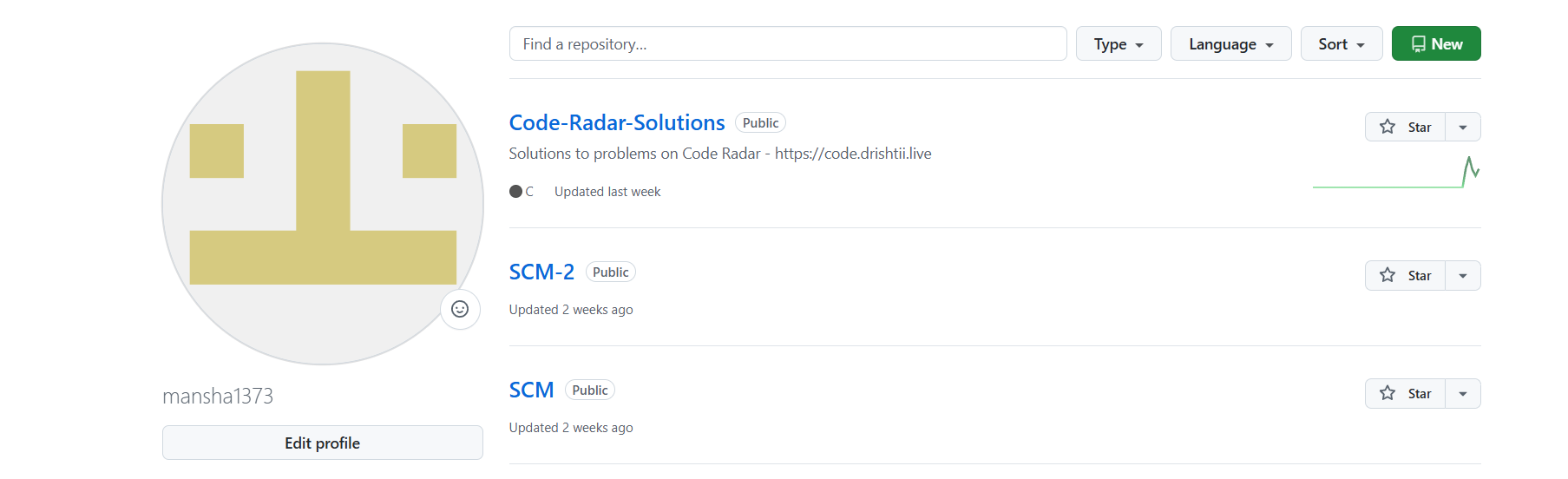
* **Procedure:**

1. Login to your GitHub account and you will land on the homepage as shown below.

Click on Repositories option in the menu bar.

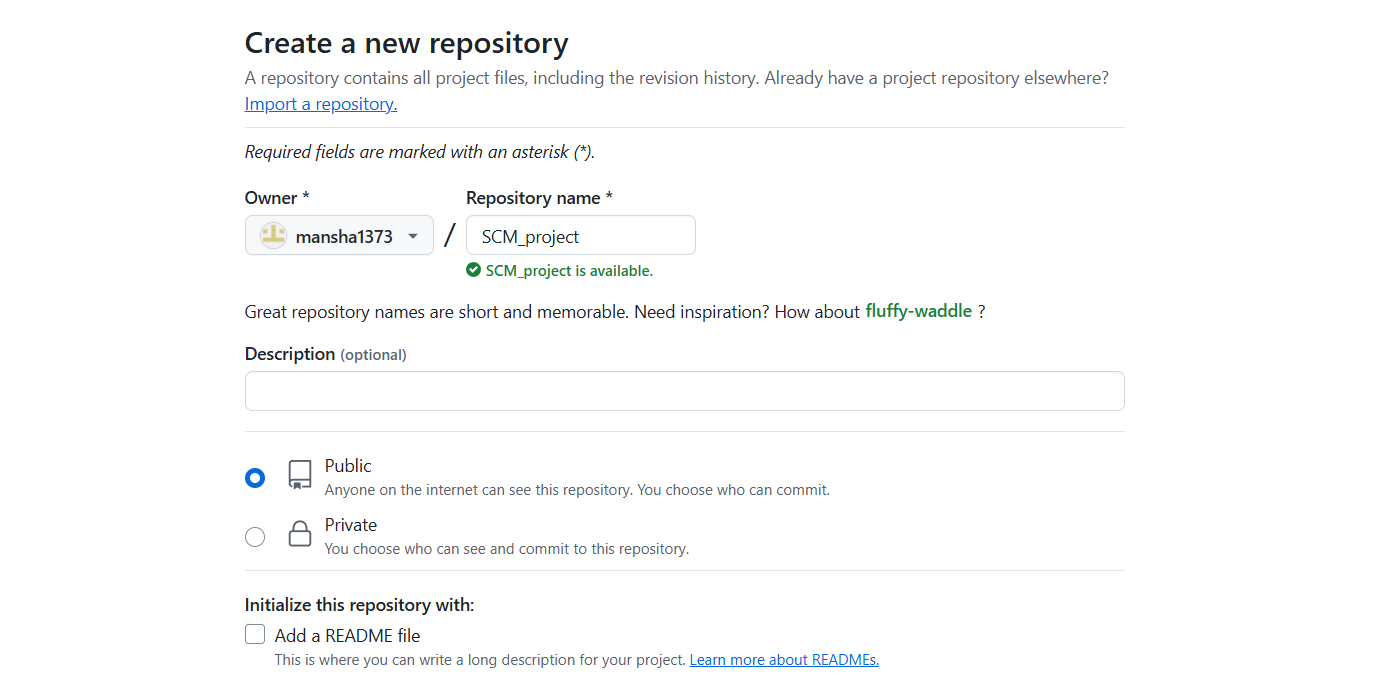


2. Click on the ‘New’ button in the the top right corner.

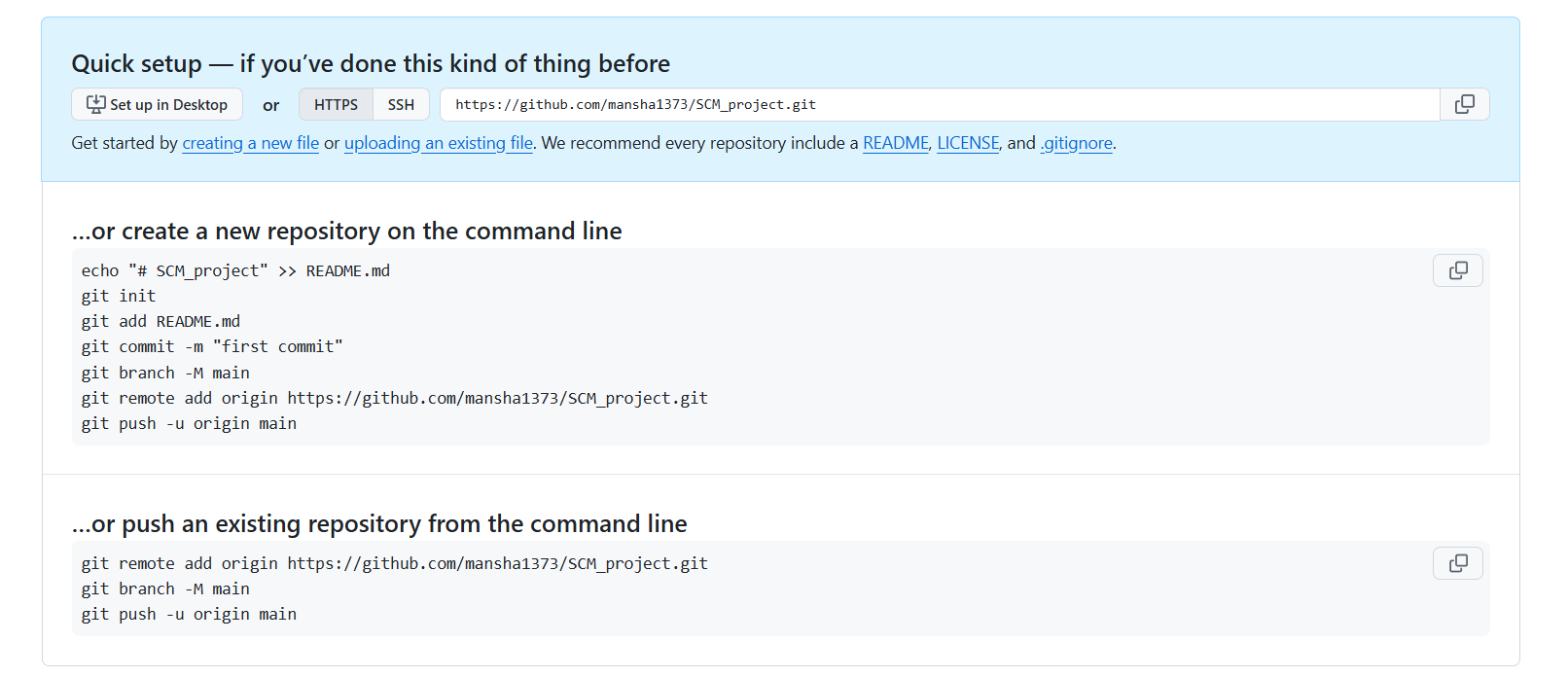


3. Enter the Repository name and add the description of the repository.

4. Select if you want the repository to be public or private.

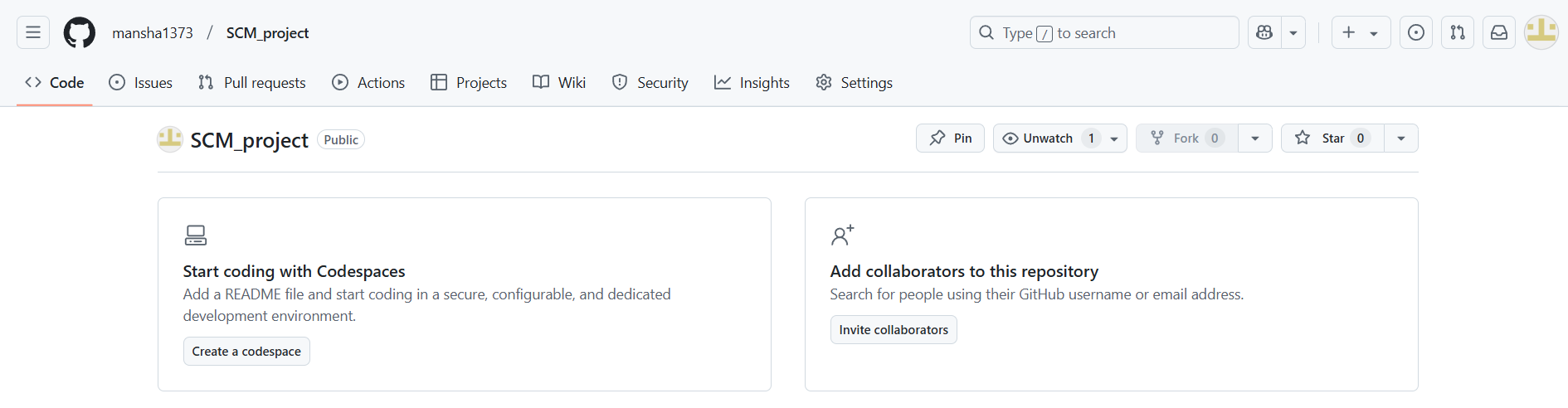


5. If you want to import code from an existing repository select the import code option.

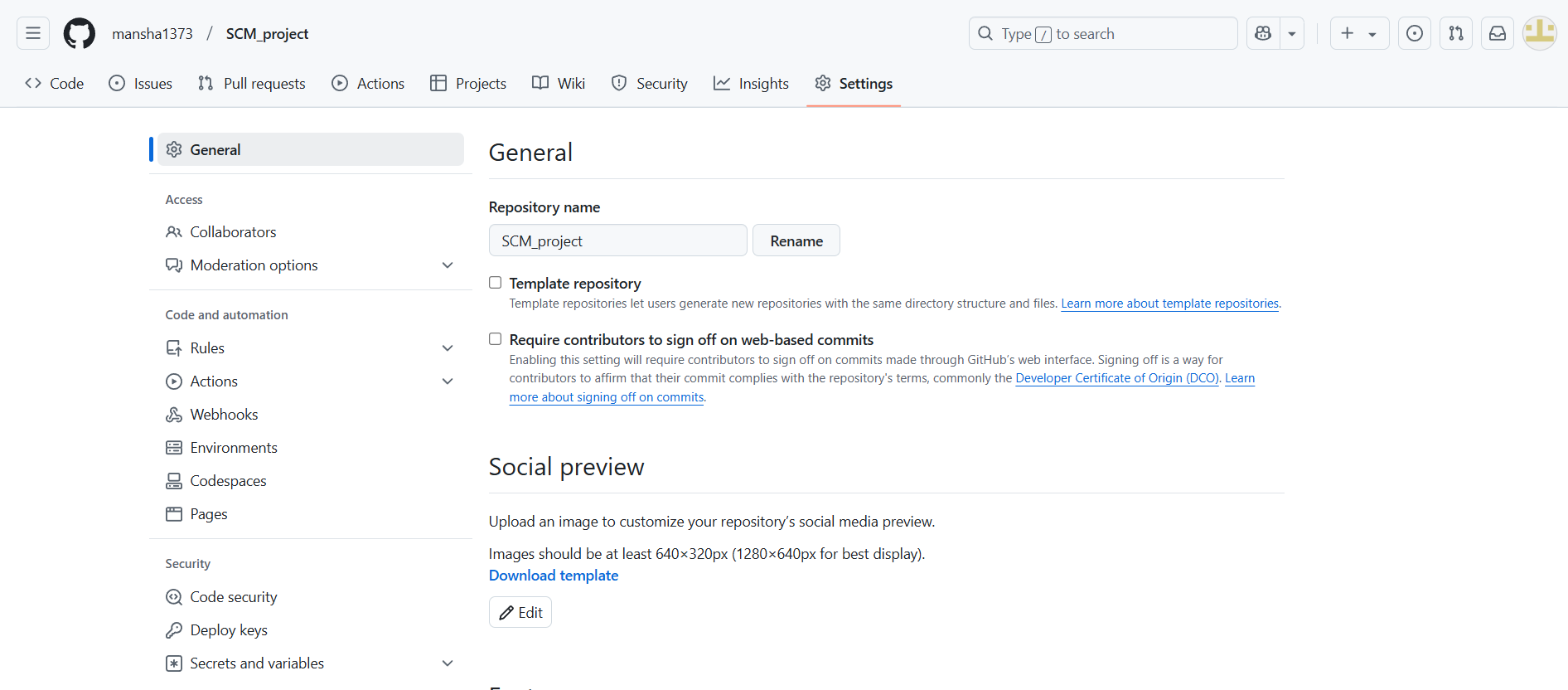


6. Now, you have created your repository successfully.

7. To add collaborators to your repository, open your repository and select settings option in the navigation bar.



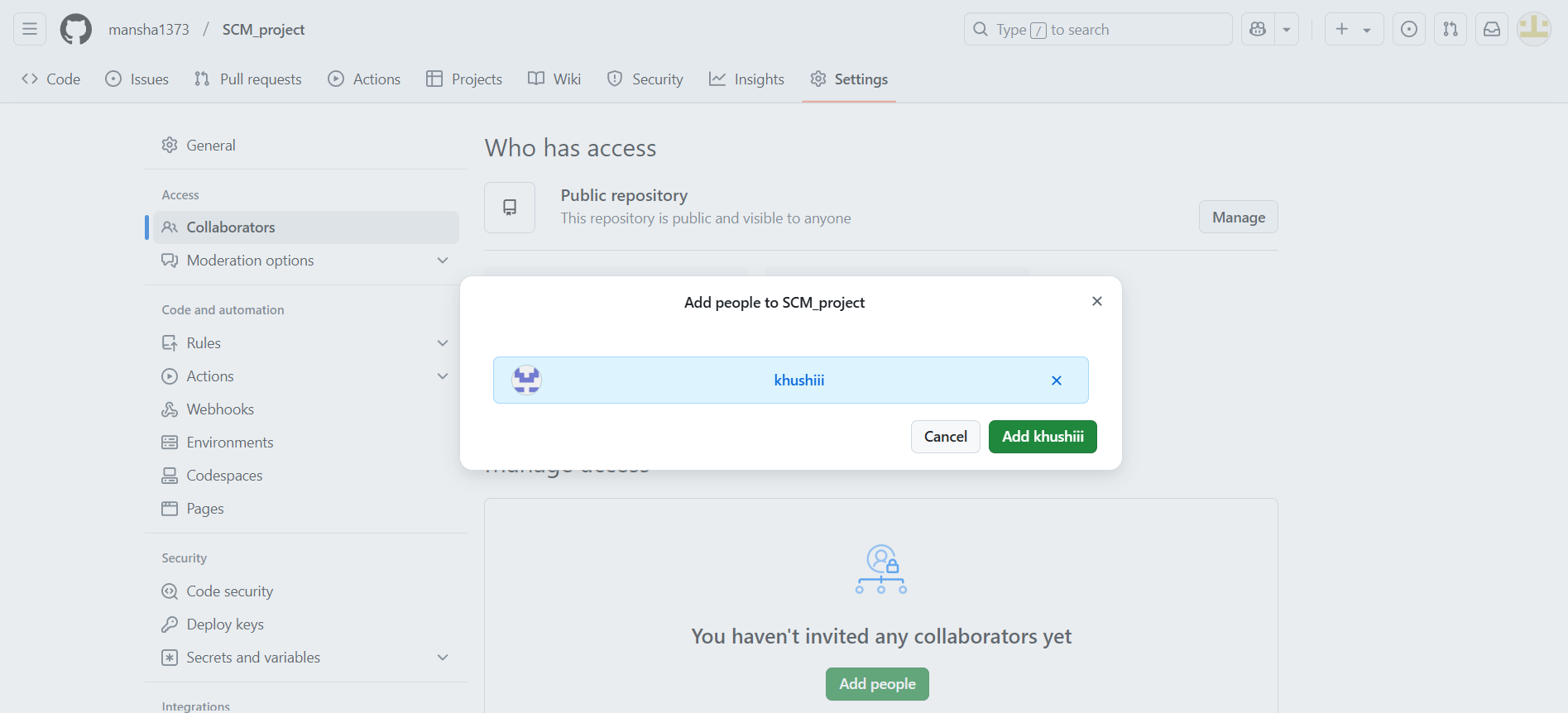
8. Click on Collaborators option under the access tab.



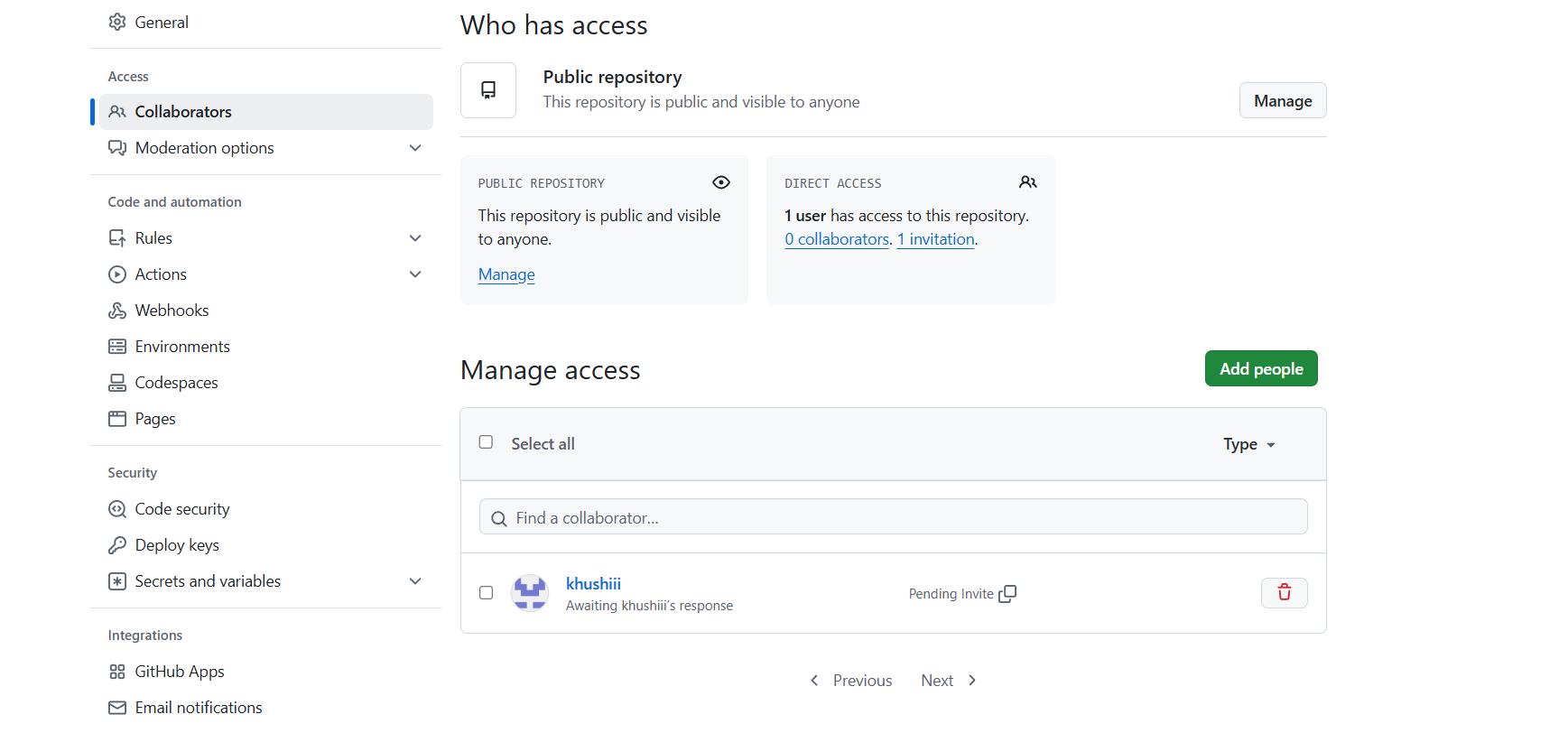
9. After clicking on collaborators, GitHub asks you to enter your password to confirm the access to the repository.

10. After entering the password, you can manage access and add/remove team members to your project.

11. To add members, click on the add people option and search the id of your respective team member.



12. To remove any member, click on remove option available in the last column of member’s respective row.



**Practical 3: Merging two branches**

* **Aim:** To merge two branches within a Git repository.
* **Theory:** Merging branches in Git allows you to combine changes from one branch into another. It is a fundamental process in collaborative workflows, ensuring all contributions are integrated into a single codebase.
* **Procedure:**

1. Create a new branch and switch to it: git checkout -b new-branch

2. Make changes to a file in the new branch and commit them: echo "New content" > file.txt git add file.txt git commit -m "Add changes in new branch" .

3. Switch back to the main branch: git checkout main.

4. Modify another file in the main branch and commit the changes: echo "Main branch changes" > another-file.txt git add another-file.txt git commit -m "Modify file in main branch" .

5. Merge the new branch into the main branch: git merge new-branch

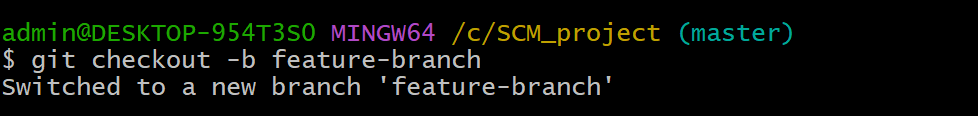
* **Tasks:** Provide the Git log showing the merge commit. Include screenshots of the merge process.
* **Solution:**

**1. Create a New Branch and Switch to It**

* First, you need to create a new branch from your current branch (usually main) and switch to it.

Run the following command to create and switch to a new branch (replace new-branch with your desired branch name):

* git checkout -b new-branch

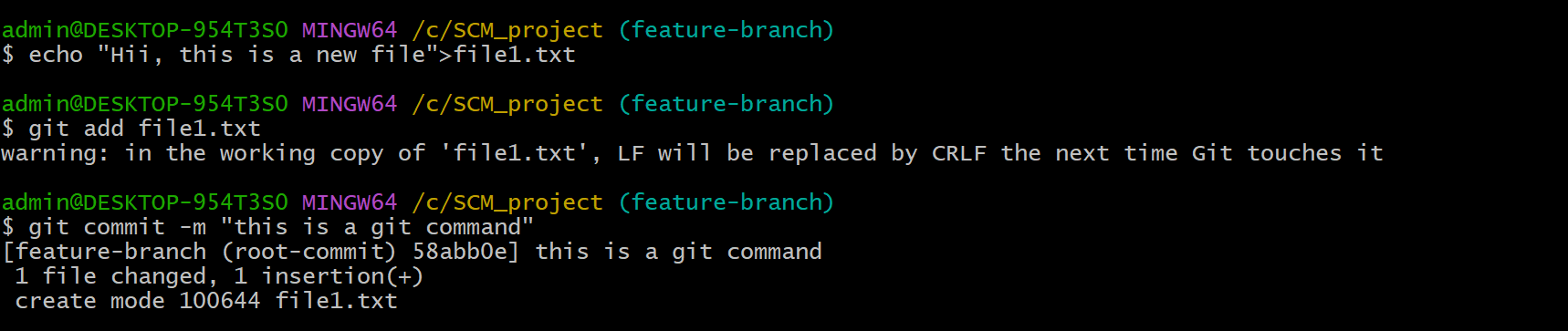


**Explanation:**

* b flag tells Git to create a new branch, and checkout switches you to that new branch.
* Make Changes to a File in the New Branch and Commit Them
* In your new branch, make some changes. For example, you can create or modify a file (file.txt).
* Run the following commands to add content to file.txt:

echo "New content" > file.txt

* git add file.txt
* git commit -m "Add changes in new branch"



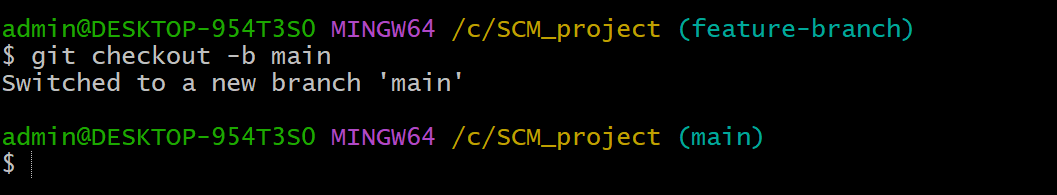
**2. Switch Back to the Main Branch**

After committing your changes on the new branch, you need to switch back to the main branch:

* git checkout main

**Explanation:**

* This command switches you back to the main branch.
* Switched to branch 'main' .



**3. Modify Another File in the Main Branch and Commit the Changes**

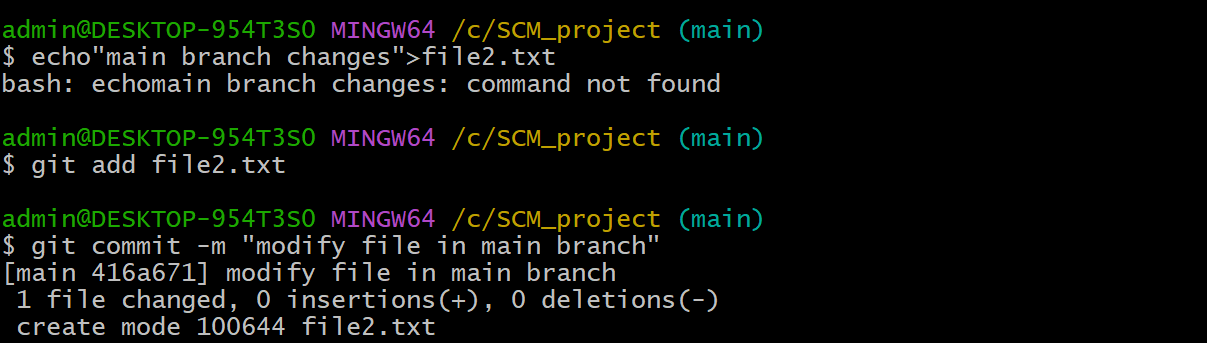
Now, on the main branch, make changes to a different file (e.g., another-file.txt).

Run the following commands:

* echo "Main branch changes" > another-file.txt
* git add another-file.txt
* git commit -m "Modify file in main branch"

**Explanation:**

* echo "Main branch changes" > another-file.txt: Creates or modifies the file another-file.txt with the text "Main branch changes".
* git add another-file.txt: Stages the changes for commit.
* git commit -m "Modify file in main branch": Commits the changes with a message.



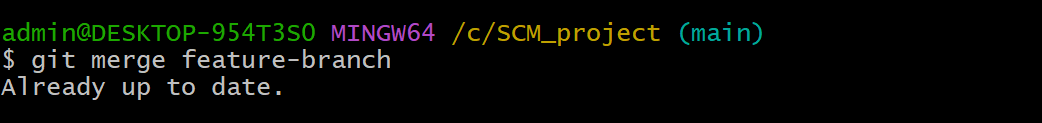
**4. Merge the New Branch into the Main Branch**

Now, you will merge the changes from your new-branch into the main branch.

Run the following command to perform the merge:

* git merge new-branch

**Explanation:** This command merges the changes from new-branch into the current branch (which is main).

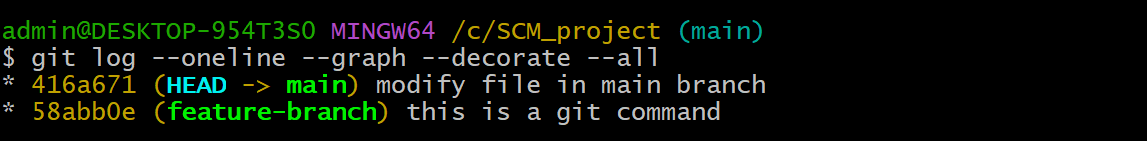


**5. View the Git Log Showing the Merge Commit**

Finally you can view the Git log to see the merge commit and the history of changes.

* Run the following command: git log --oneline --graph --decorate --all

**Explanation:** This will display the commit history in a simplified format, showing the branch history and the merge commit.



**Practical 4: Push/Pull Using Git**

* **Aim:**  To demonstrate push and pull operations in Git.
* **Theory:** Push transfers committed changes from the local repository to the remote repository, while pull retrieves updates from the remote repository.
* **Procedure:**

1. Make changes in the local repository and commit them.

2. Push the changes to the remote repository using git push.

3. Make changes directly on the remote repository (e.g., via GitHub interface).

4. Pull the changes to the local repository using git pull.

* **Tasks:**

1. Provide screenshots of the push and pull operations.

2. Include the updated commit log.

* **Solution:**

1. **Make Changes in the Local Repository and Commit Them .**

**Steps:**

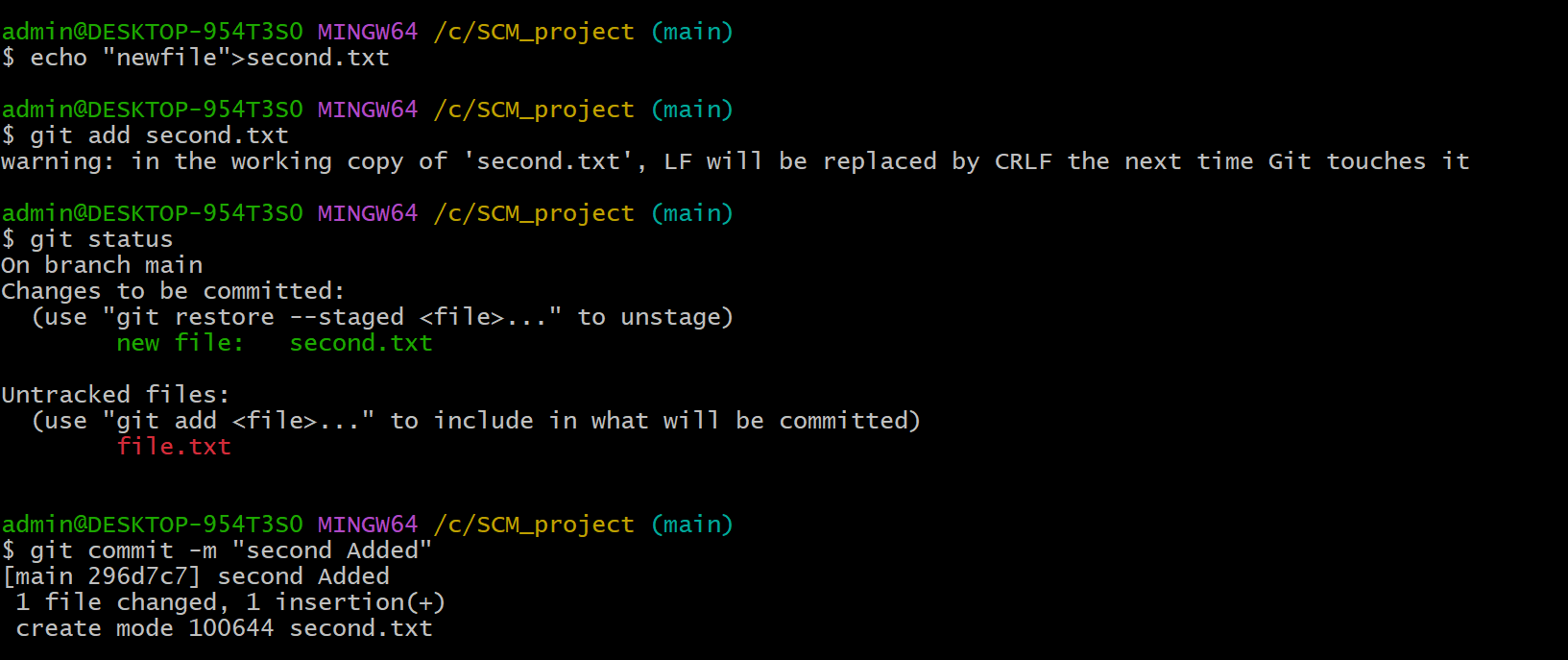
* If you don’t have a repository already, initialize a new repository in a folder:
* git init
* Create or modify a file. For example, let's create a file file.txt
* echo "Initial content" > file.txt

**Add and commit the changes:**

* git add file.txt
* git commit -m "Add file.txt with initial content"

**Explanation:**

* git add file.txt stages the file for commit.
* git commit -m "Add file.txt with initial content" commits the file with a message.



**2. Push the Changes to the Remote Repository Using git push**

* First, make sure your local repository is connected to a remote repository on GitHub (or any other Git hosting platform).

Run this command to link your local repo to a remote repository:

* git remote add origin <https://github.com/yourusername/repository-name.git>
* Replace your username and repository-name with your actual GitHub username and repository name.
* Now, push your changes to the remote repository:
* git push -u origin main.

**Explanation:**

* git push -u origin main pushes the changes to the main branch of the remote repository.

**3. Make Changes Directly on the Remote Repository (e.g., via GitHub Interface)**

* Go to your repository on GitHub.
* Click on the file (e.g., file.txt), then click on the pencil icon to edit the file.
* Make some changes (for example, add more content) and save the changes.

**4. Pull the Changes to the Local Repository Using git pull**

Once you've made changes directly in the remote repository (e.g., via GitHub), you need to **sync your local repository** with the remote repository using git pull.

Run this command to pull the changes:

* git pull origin main

**Explanation:**

* git pull origin main fetches and merges the changes from the main branch of the remote repository into your local repository.

View the Updated Commit Log

To check the commit log and ensure that both local and remote repositories are in sync, use the following command:

* git log --log --oneline --decorate --all

**Explanation:** This command will show a simplified commit history with the commit hashes and messages.

