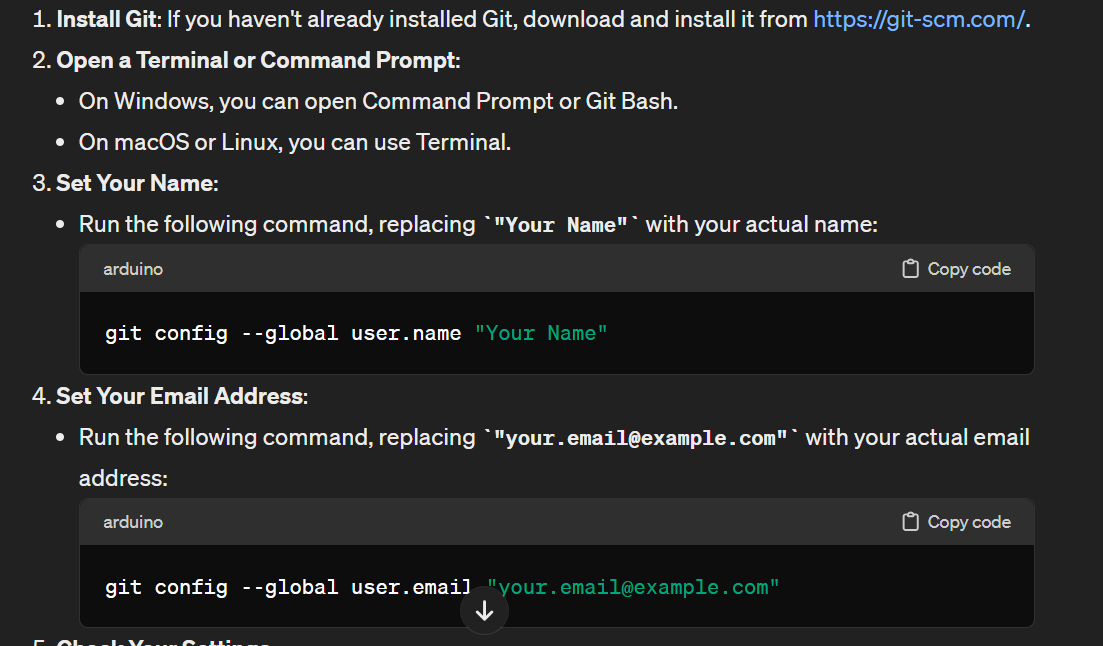
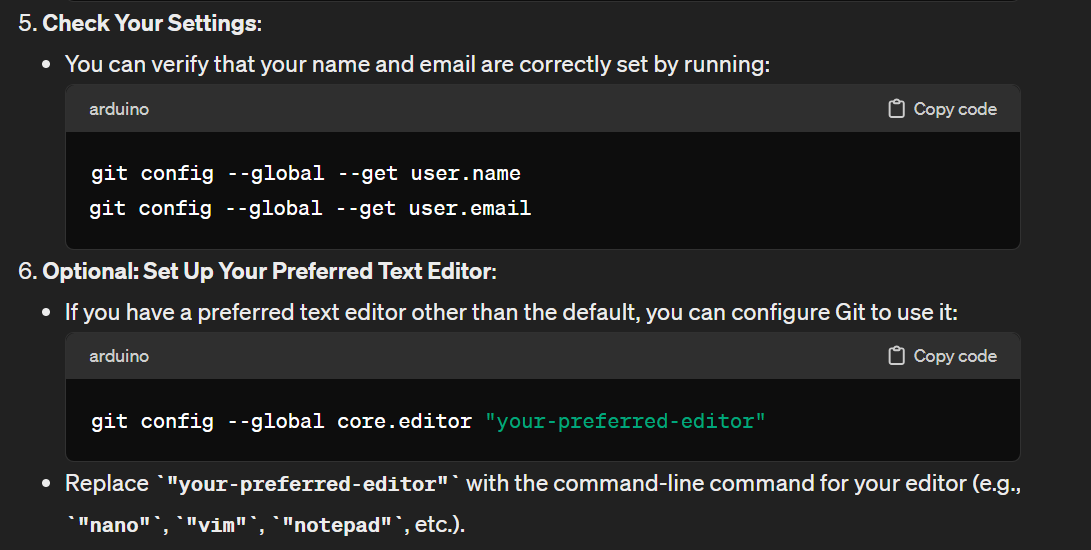
1. **installing the GIT**

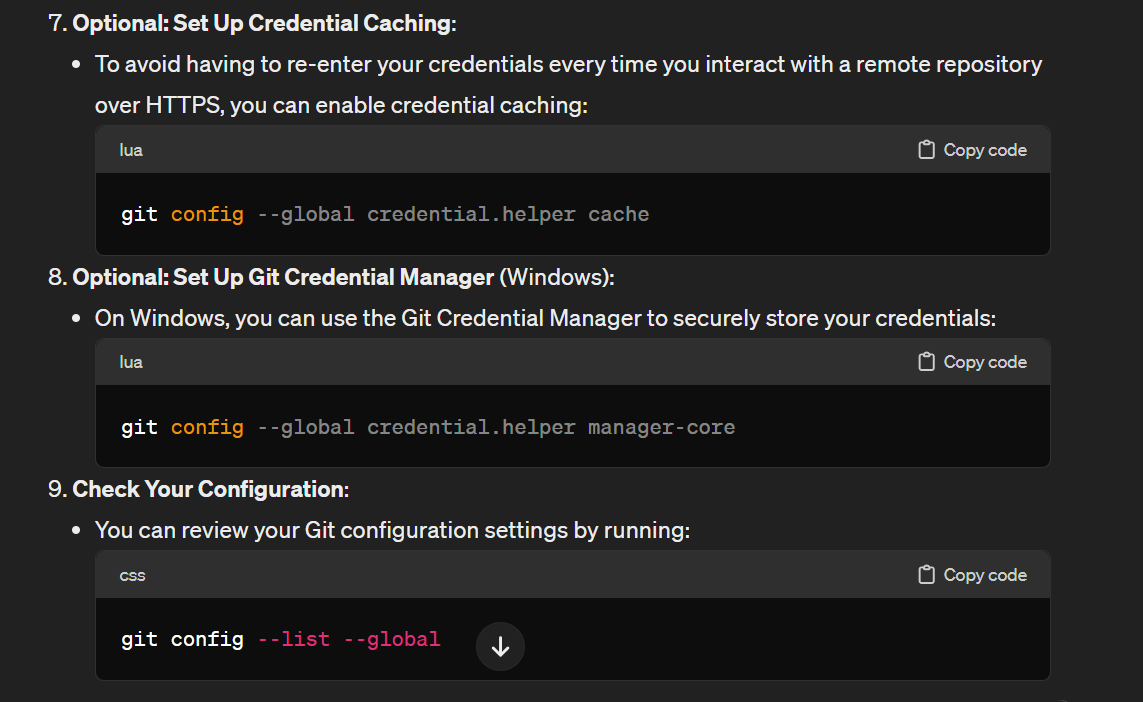
* Download Git: Go to https://git-scm.com/download/win and get the Git installer for Windows.
* Run Installer: Double-click the downloaded file to start the installation process.
* Follow Instructions: The installer will guide you through the process with simple prompts. Just click "Next" or "Install" as appropriate.
* Choose Options: During installation, select "Use Git from the Windows Command Prompt" so you can run Git commands from the Command Prompt or PowerShell easily.
* Finish Installation: Once the installation completes, click "Finish".
* Check Installation: Open Command Prompt or Git Bash and type git --version to make sure Git is installed correctly. You should see the installed version displayed.

1. **manage public and private repository**

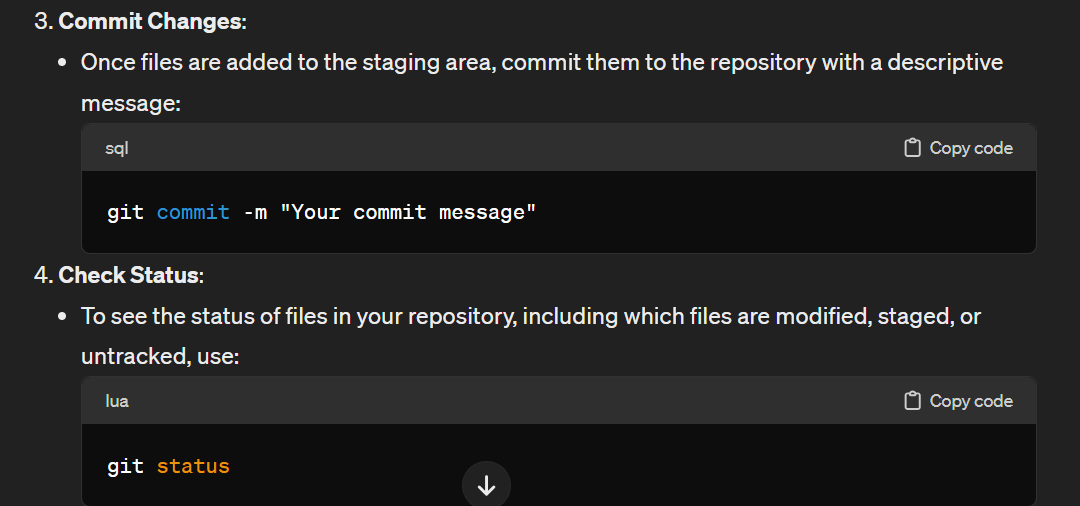
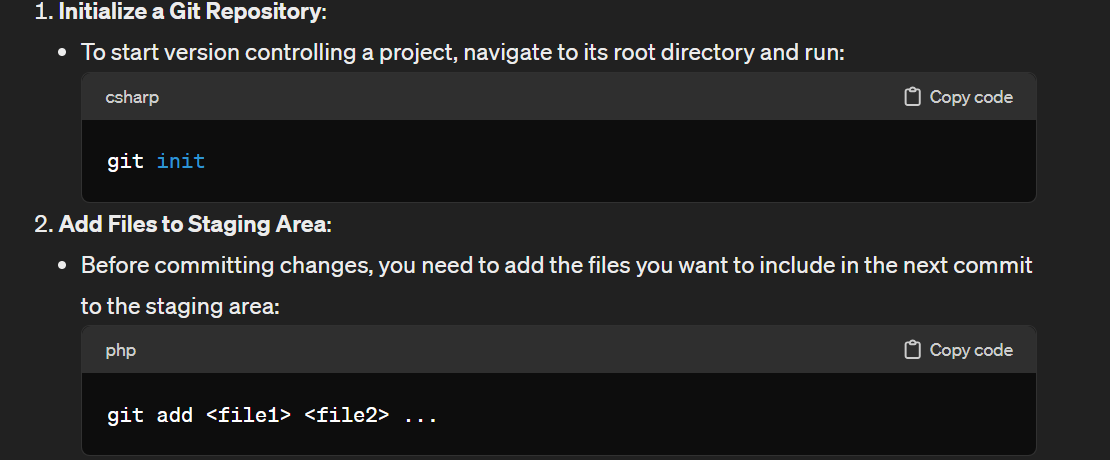
* Create a Repository: If you haven't already, create a repository on GitHub. Sign in to your GitHub account, click on the "+" icon in the top right corner, and choose "New repository". Fill in the necessary details like repository name, description, etc., and click "Create repository".
* Access Repository Settings: Once your repository is created, navigate to its main page. You'll see options like "Code", "Issues", "Pull requests", etc. Click on the "Settings" tab located near the top-right corner of the repository page.
* Change Repository Visibility: In the Settings page, scroll down until you find the "Danger Zone" section.
* Under "Danger Zone", you'll find the "Change repository visibility" option.
* Click on "Change repository visibility" and select "Private" from the dropdown menu.
* Confirm Changes: After selecting "Private", GitHub will prompt you to confirm the change. You might need to provide your GitHub password or re-enter your account credentials to confirm the action.
* Save Changes: Once confirmed, click on the "Make private" button to save your changes.
* Verify Repository Status: After making the repository private, its visibility will be updated accordingly. You'll notice a lock icon next to the repository name, indicating that it's now private.

1. **Write and execute the commands to set up and configure git on your local machine.** 

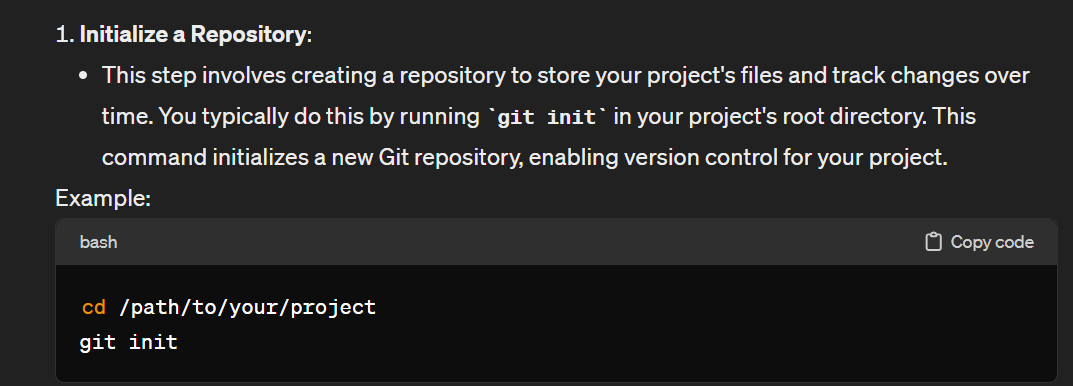


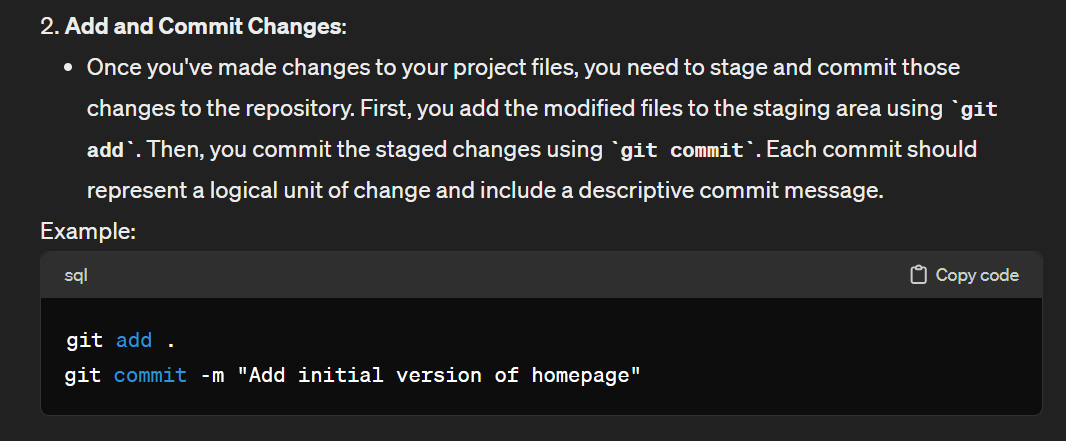


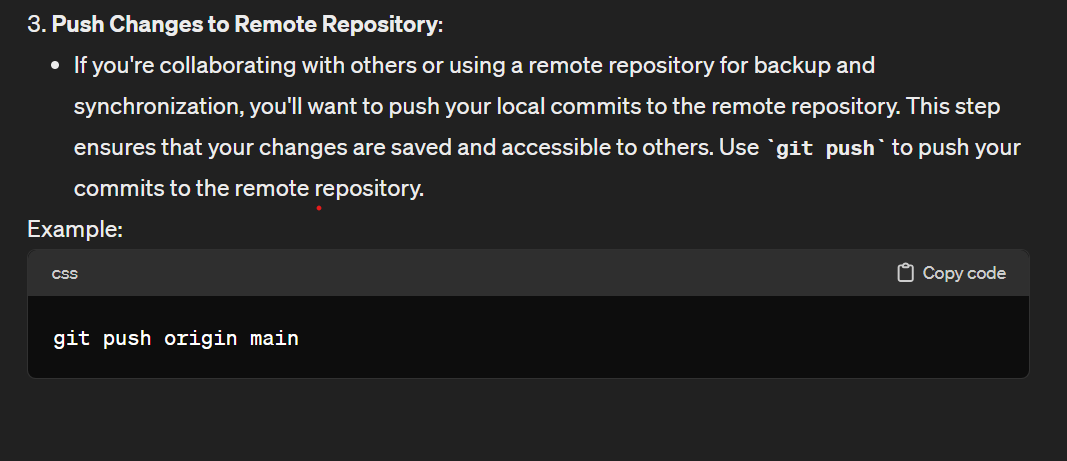
1. **performing basic operations in Git.**



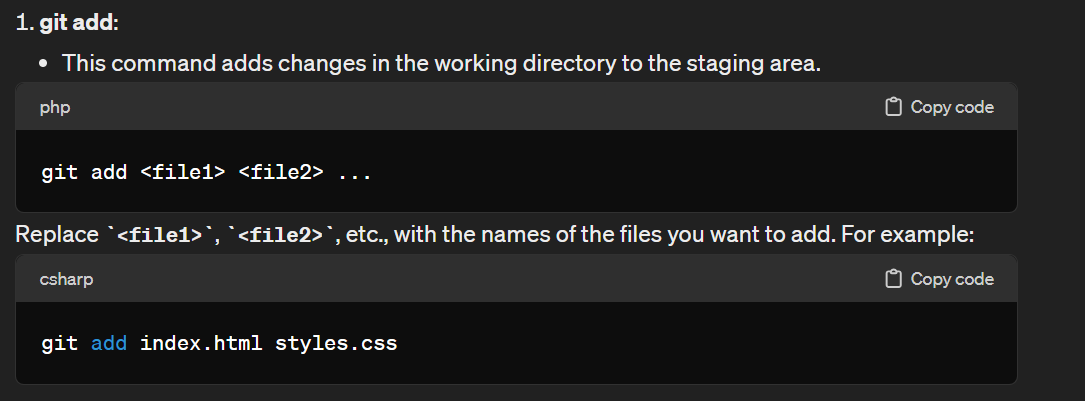
1. **Write and execute the three important steps of version control.**

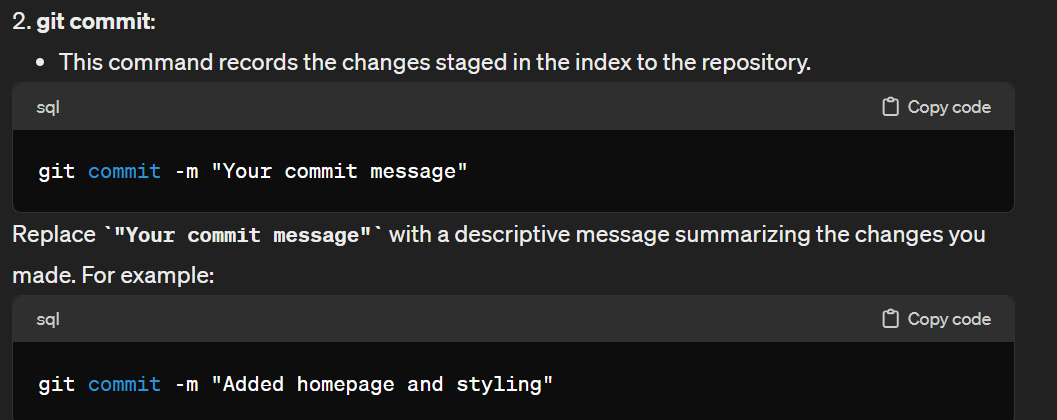


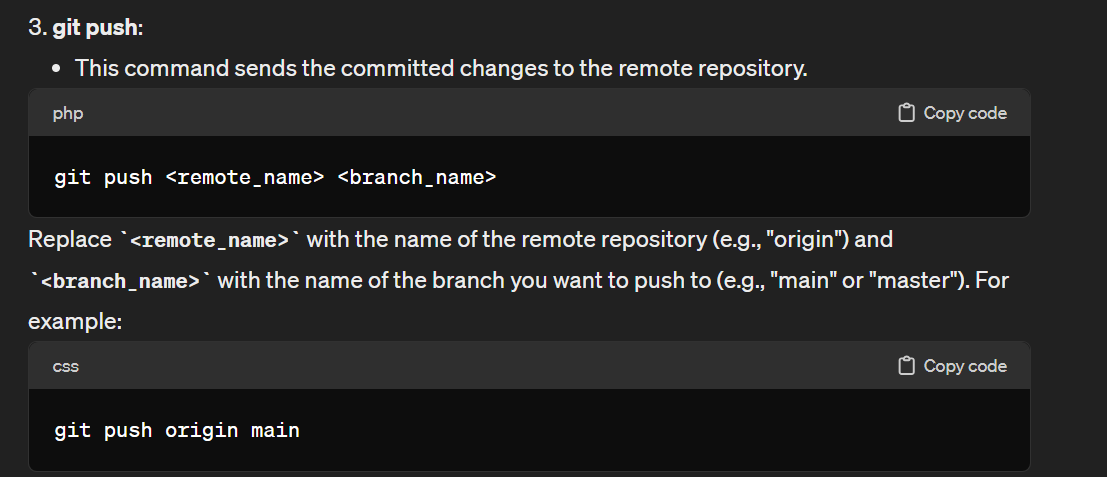


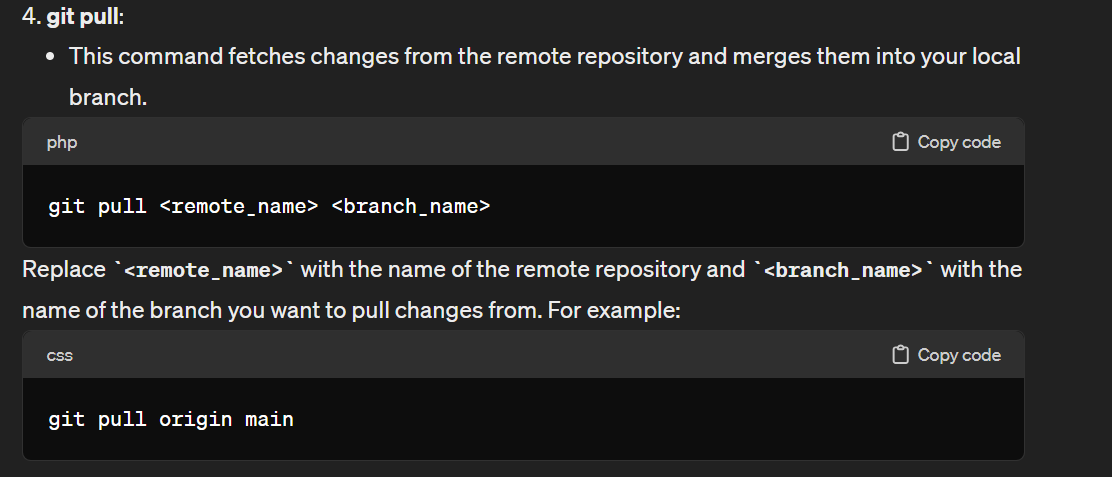


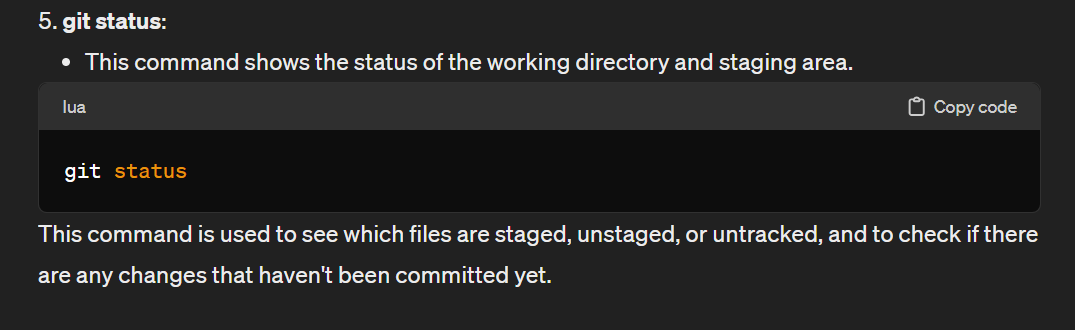
1. **execute git add, git commit, git push, git pull, git status.**









****

1. **Explain and execute how to create branches and merging branches.**

# Creating branches

To keep track of changes to this file using **git**, you need to:

1. Clone the repository.
2. Move into the cloned repository
3. Create a new branch using the command (replace *feature-branch* with your desired branch name).

*git checkout -b feature-branch*

1. Make modifications to files in your project.
2. Use git add to add the changes to the staging area
3. Commit the changes with a meaningful message

# Merging branches

To keep track of changes to this file using **git**, you need to:

1. Switch back to the main branch.

*git checkout main*

1. Merge the branch into the main branch

*git merge feature-branch*

1. Resolve conflicts (if necessary)
   1. Open the conflicting files and resolve the conflicts manually.
   2. After resolving conflicts, add the changes to the staging area and commit:

*git add .*

*git commit -m "Merge feature-branch into main"*

1. Push changes to github using the following command.

*git push origin main*

# 8.Create and Address GitHub Issues

1. Create a new issue in your repository:
   1. Go to your GitHub repository.
   2. Click on the "Issues" tab.
   3. Click the "New issue" button.
   4. Fill in the issue title and description.
   5. Optionally, assign the issue to a collaborator, apply labels, and set milestones.
2. Assign the issue to a collaborator:
   1. Within the issue, use the "Assignees" section to assign the issue to a collaborator.
   2. The assigned collaborator will receive notifications about the issue.
3. Label the issue with appropriate tags:
   1. Apply labels to categorize and prioritize issues.
   2. Create and use labels like "bug," "feature," or any relevant labels for your project.
   3. Labels can be added when creating the issue or edited later.
4. Close the issue using a commit message:
   1. Make changes to the codebase to address the issue.
   2. In your local repository, commit the changes with a commit message that references the issue number:

*git commit -m "Fix #1: Resolve issue with feature X"*

Replace "1" with the actual issue number.

* 1. Push the changes to GitHub:

*git push origin main*

The issue will be automatically closed when the commit is pushed.