Week 4 Solution: Bash Scripting & Git

# Section A: Project Template

Write a bash script called **project\_template.sh** that automates the creation of machine learning project structures with a consistent folder and file layout. The primary goal is to save time by automating the generation of five project templates. The script will give the user two options:

1. **Project name**
2. **Exit**

If a user selects option 1, they will be prompted to enter a project name, which will be used to generate the project folder and its subdirectories. If option 2 is selected, the script will quit. If an invalid option is selected, the screen will display "Invalid selection."

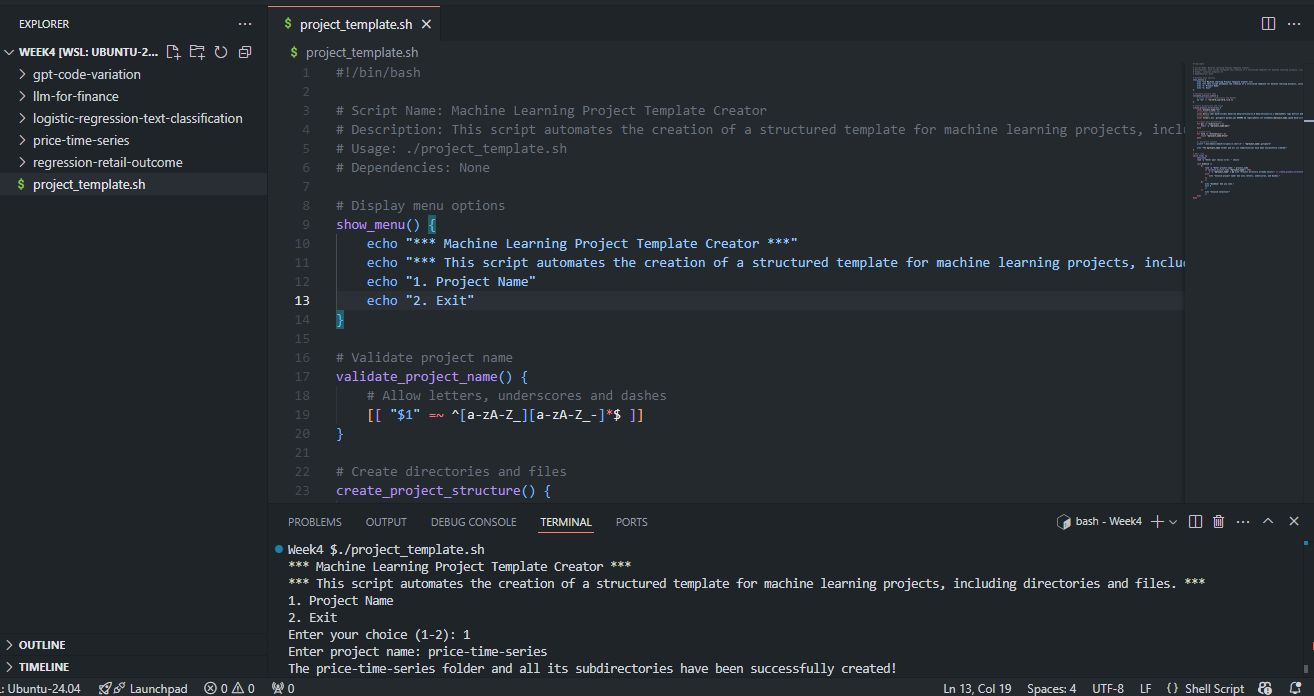
Once a valid project name has been entered (with no numbers or punctuation marks except for dashes or underscores), the script will create a root project directory that contains subfolders such as data/raw, data/versions, notebooks, models, metrics, deployments, logs, bash-scripts, src, and reports, as demonstrated in the screenshot. Files like .gitignore, README.md, requirements.txt, .env, params.yml, and notebooks/[project\_name].ipynb will also be generated. The script will also add .env, data/, and bash-scripts/ to the .gitignore file.

Additionally, empty Python source files such as \_\_init\_\_.py, eval.py, deploy.py, feature.py, train.py, cleaning.py, split.py, and processor.py will be created in the src directory. This setup will allow the user to quickly create consistent machine learning project structures with a single command, enhancing productivity and reducing manual effort.

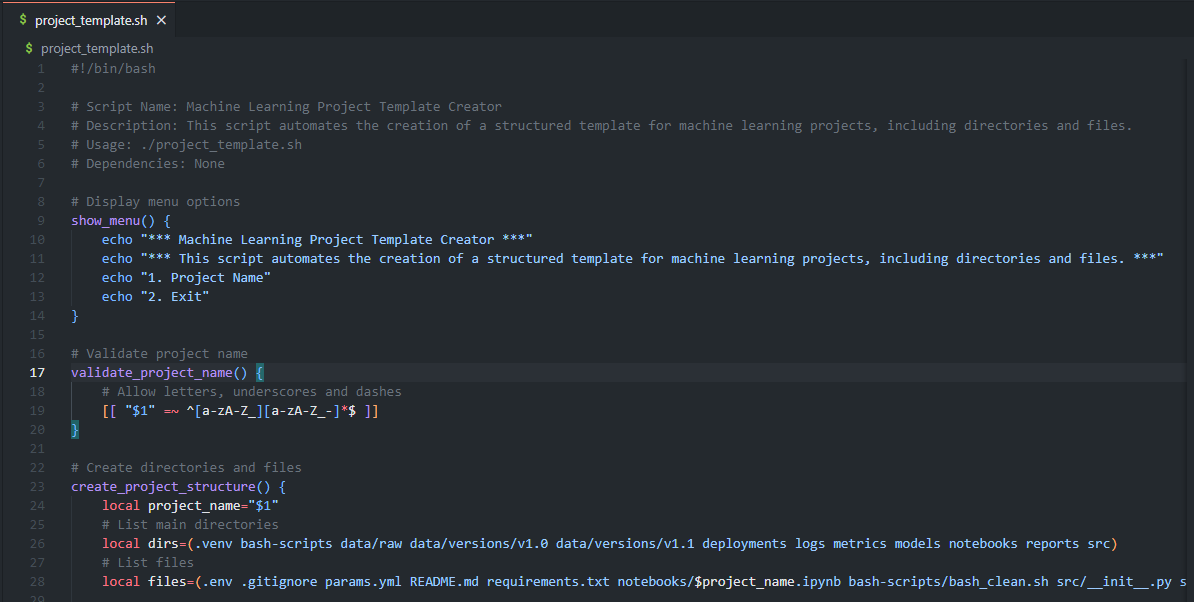
For demonstration purposes, create the following projects using the script you have written:

1. **price-time-series**
2. **regression-retail-outcome**
3. **gpt-code-variation**
4. **logistic-regression-text-classification**
5. **llm-for-finance**

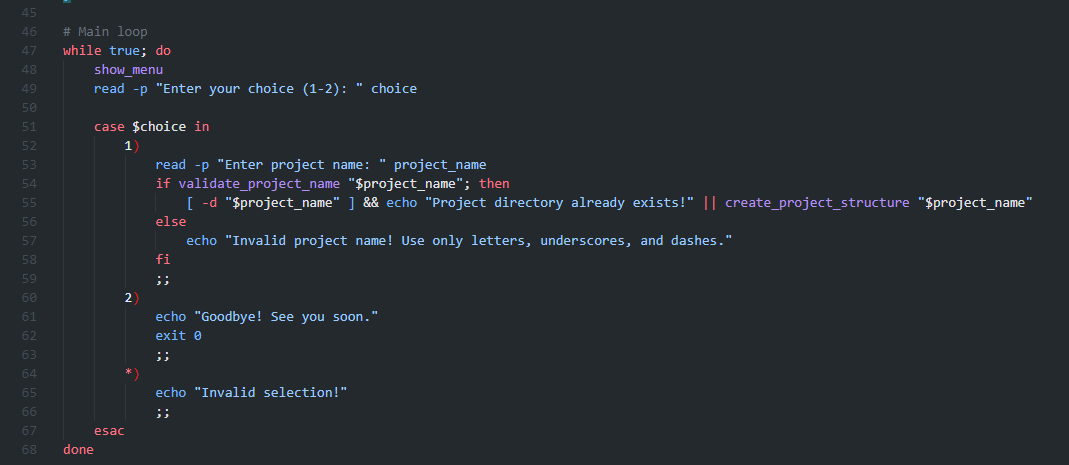
In this section, capture screenshots of the source code as well as one of the created projects, including its subfolders and files. Attach these screenshots in a Word document for submission.



Creating the **price-time-series** template on the terminal







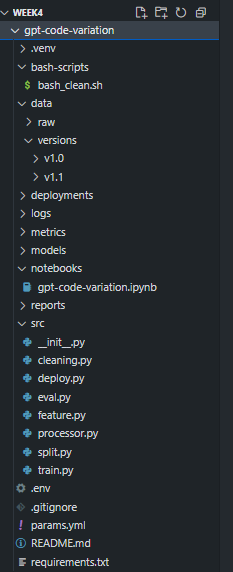


Image showing gpt-code-variation folder and subdirectory

# Section B: Introduction to Git & GitHub

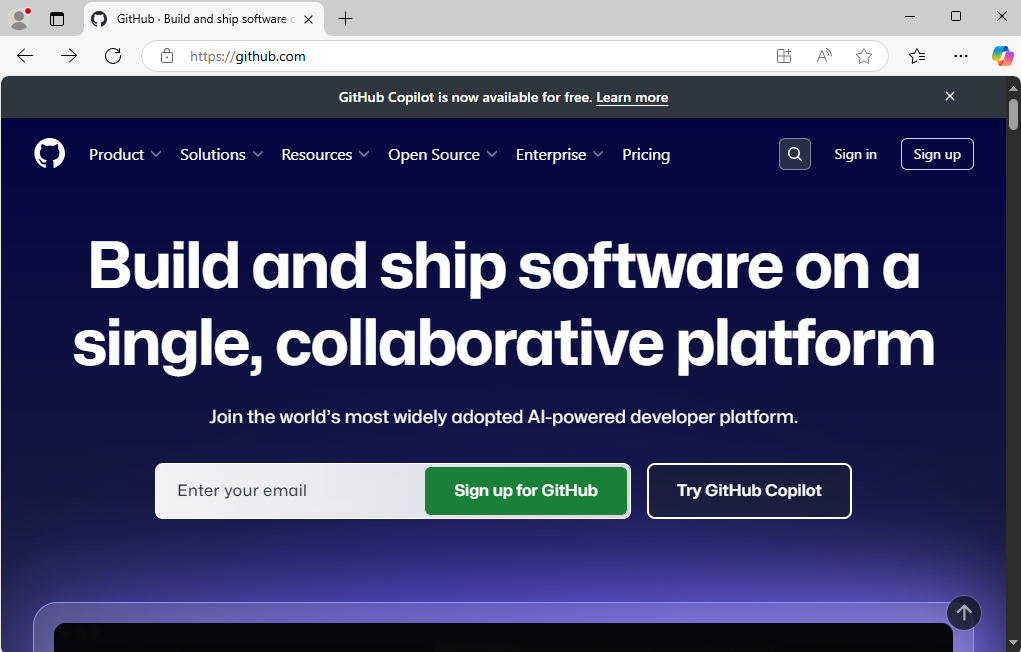
1. In your own words, what is the difference between Git and GitHub?

Git is the free open-source distributed version control system that runs locally on the computer. It tracks changes made in code/files, allowing you to revert, branch, and collaborate efficiently.

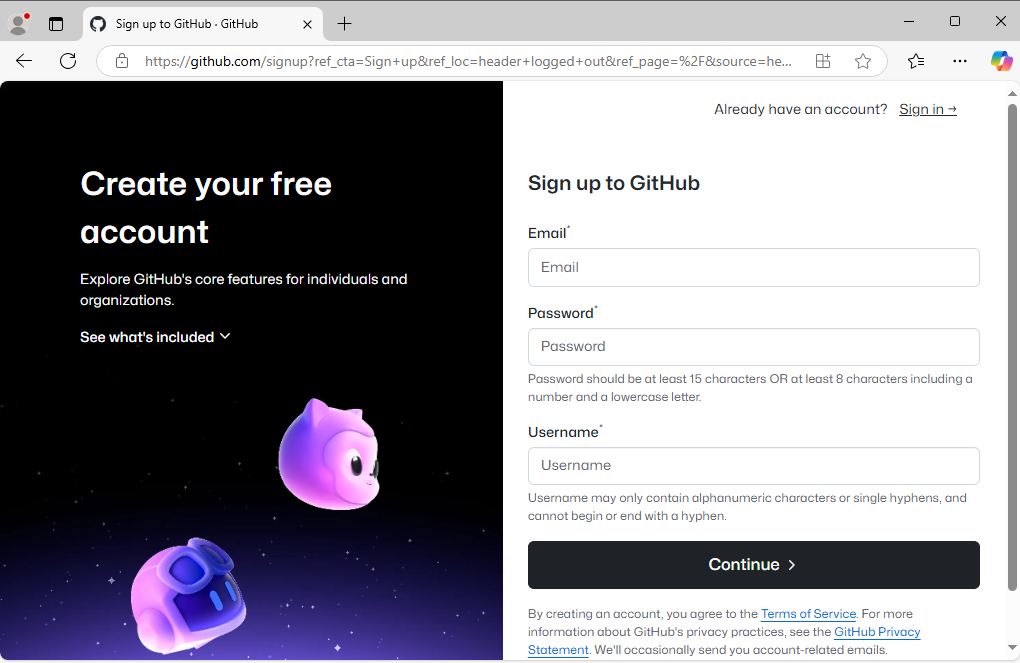
GitHub is a cloud-based platform for hosting Git repositories. It allows you to share your code, adapt existing projects and collaborate with teams. GitHub provides tools such as pull requests, issue tracking and continuous integration / continuous deployment (CI/CD).

1. Create a GitHub account. Explain how to create the account.

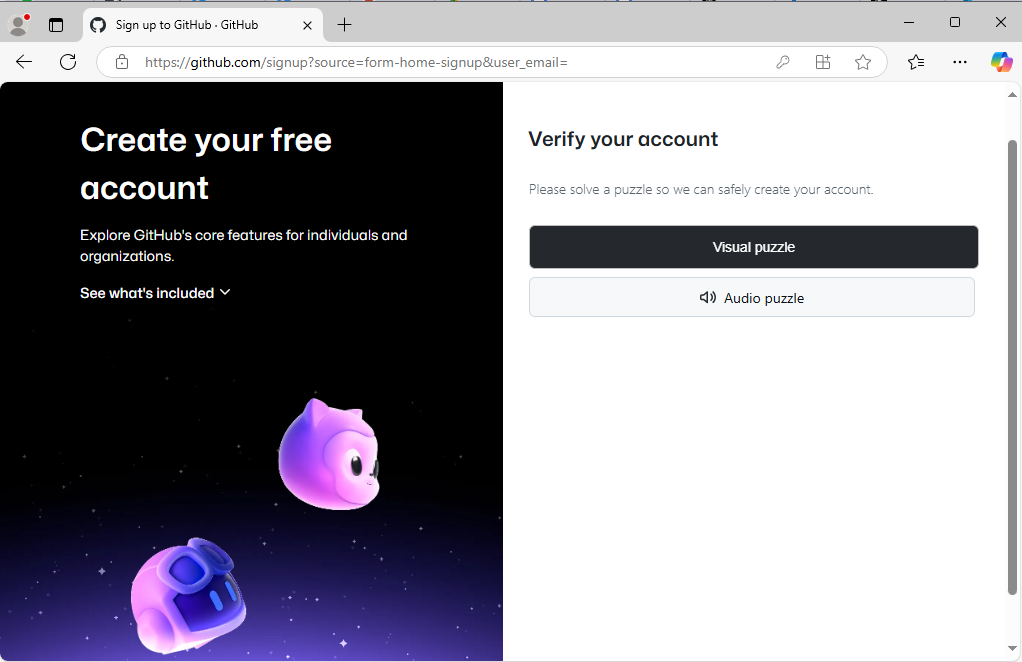
Step 1: Visit <https://github.com> on a web browser and click the big green “Sign up for GitHub” button.



Step 2: Provide an email address, a password and a user name that isn’t already taken. then click Continue



Step 3: Verify your account. GitHub will send you an email to verify the address you provided then sign in.



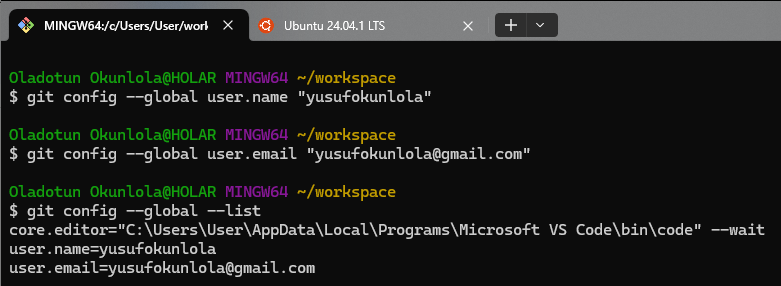
1. Assuming that you have created a GitHub account, install Git and configure it with your GitHub username and email. Show the code used for Git configuration on a local machine.

To configure Git on the local machine with your GitHub credentials, use the code below (see screenshot for example).

Set username: git config -–global user.name “[github-username]”

Set email: git config -–global user.email “[valid-email]”

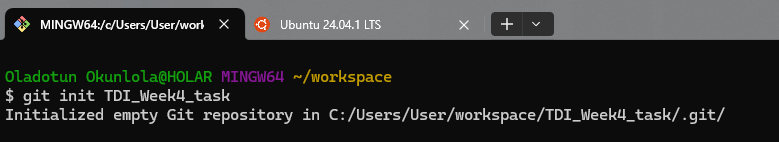
Confirm Git configuration: git config -–global –list



1. When should Git be configured? Support your answer.

Git should be configured immediately after installation. The basic configuration which involves identity settings is important for tracking who makes changes to projects which is required for committing changes to repositories. Early Git configuration prevents inconsistencies, ensures better collaboration and improves workflow efficiency.

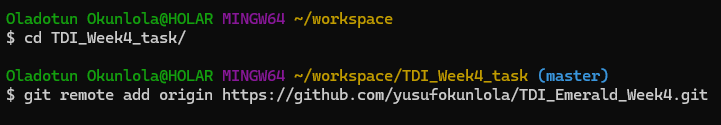
1. Provide the code to initialize Git locally in two different ways. Mention their differences.
2. git init: This creates a new subdirectory named .git that contains all of your necessary repository files in the present working directory.
3. git init test: This creates a folder named “TDI\_Week4\_task” with an empty Git repository subdirectory.



1. Provide the code to connect Git local and remote repositories.

The code below connects the Git local to a remote GitHub remote repository

git remote add origin <https://github.com/github-username/repository-name.git>



1. Give reasons why we should use Git tokens when connecting local and remote repositories.
2. Git tokens allow us to define the scope of what one wants to do (e.g., read-only, read-write) thus limiting access to sensitive data.
3. Git tokens can be revoked or regenerated after usage without having any effect on the account credentials.
4. Git tokens can be used for authentication in scripts instead of using a username and password thus providing enhanced security.
5. Explain the purpose of git add.

git add [file] : add a file as it looks presently to the next commit (stage) in Git's version control system.

1. Explain when we should use git status, git commit, and git push.

git status : shows modified files in working directory, staged for the next commit

git commit -m “[descriptive message]” : commit your staged content as a new commit snapshot

git push : transmits local branch commits to the remote repository branch

In summary, git status should be used before committing files to see what’s changed; git commit should be used to save a version of your work before pushing to a remote repository and git push should be used to share committed changes to a remote repository (like GitHub).

1. What is the use of branches in Git? Support your answer.

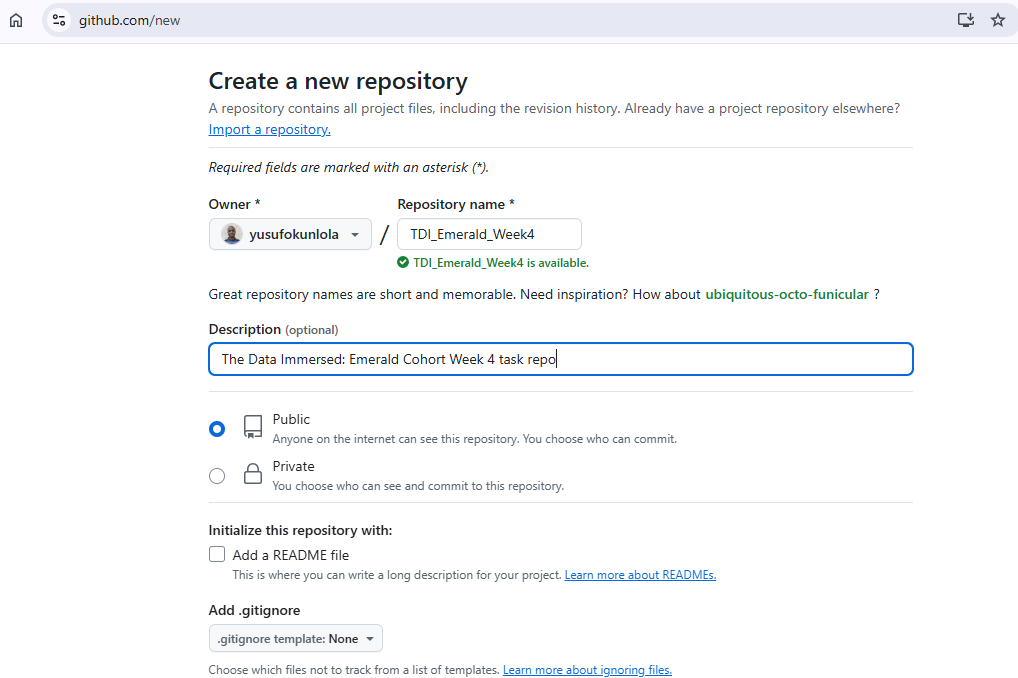
Branches in Git serve as isolated development environments that allow developers to work on different features or fixes independently without affecting the main codebase. Git branches are lightweight movable pointer to commits and fast to create.

1. What is the difference between git push and git push --set-upstream?

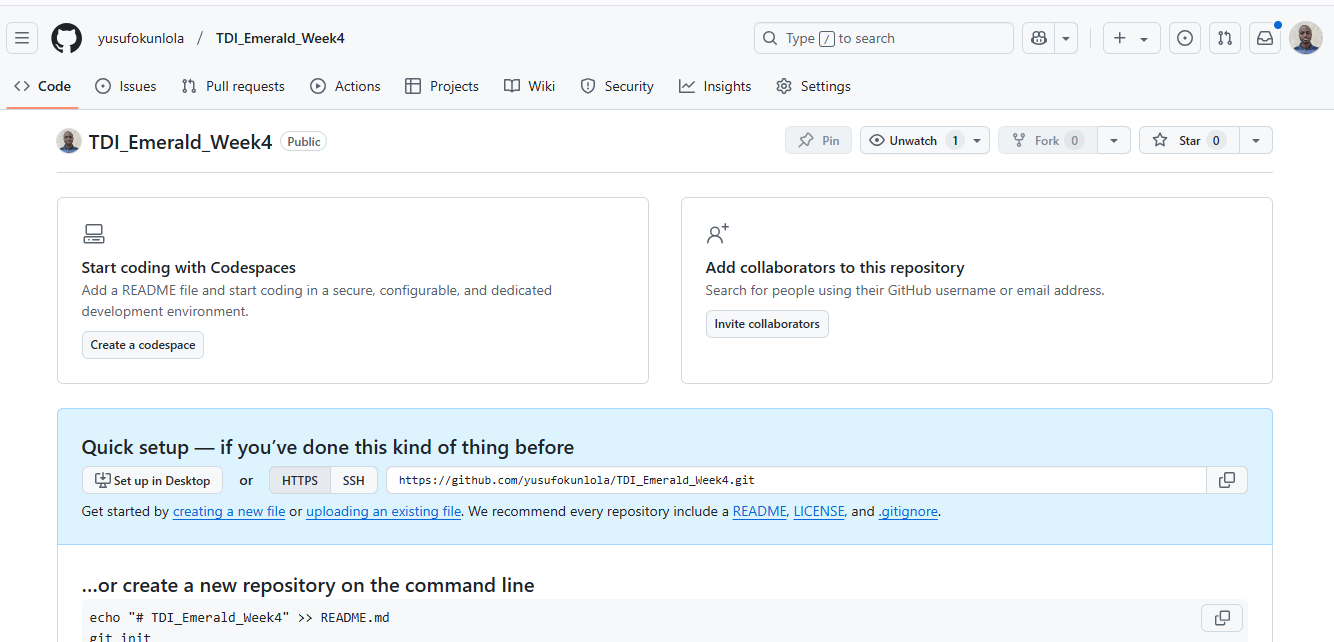
git push: This shares committed changes from the local repository to the remote repository. It works if the local branch is already linked to a remote branch.

git push --set-upstream: This is used to link the local branch to a remote branch so future git push and git pull commands can be done without specifying the remote and branch.

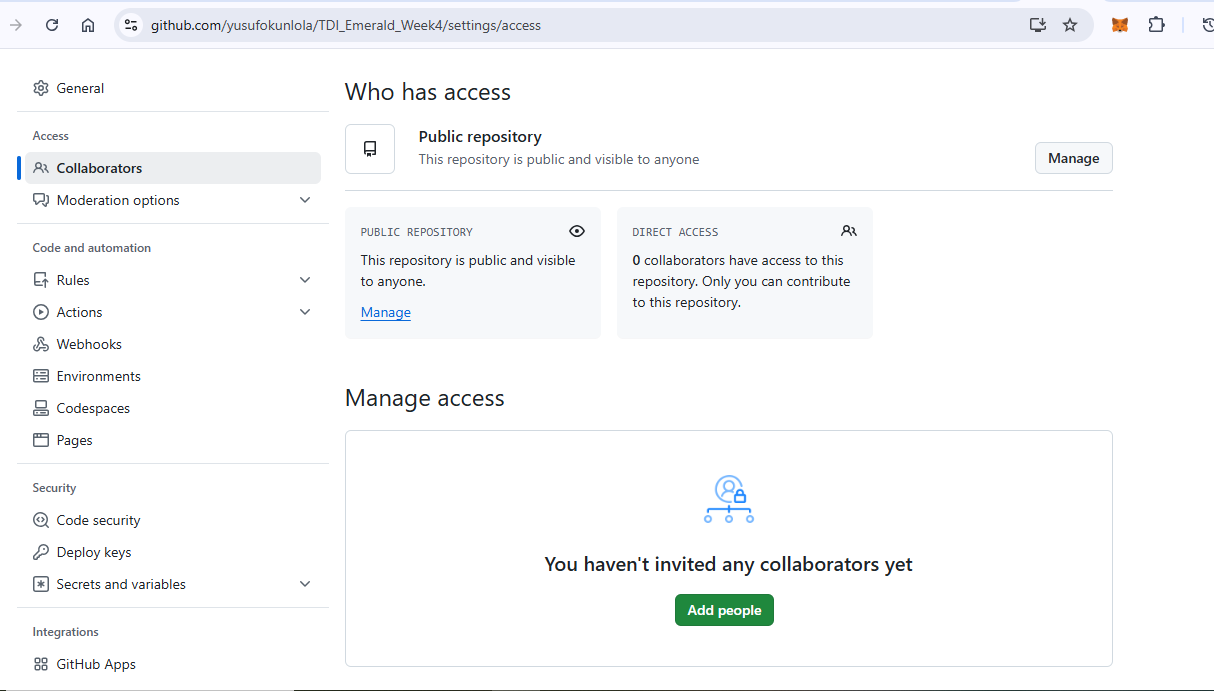
1. Create a GitHub repository and add a collaborator to your repository. Ask your classmate to be your collaborator.
2. Create a new repository and give it a name.



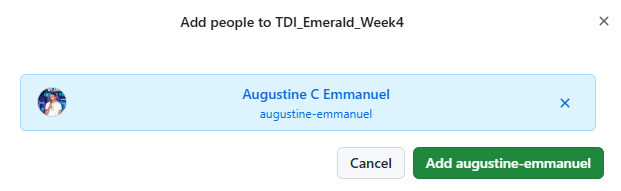
1. Click on **Add collaborators to this repository** to invite a collaborator



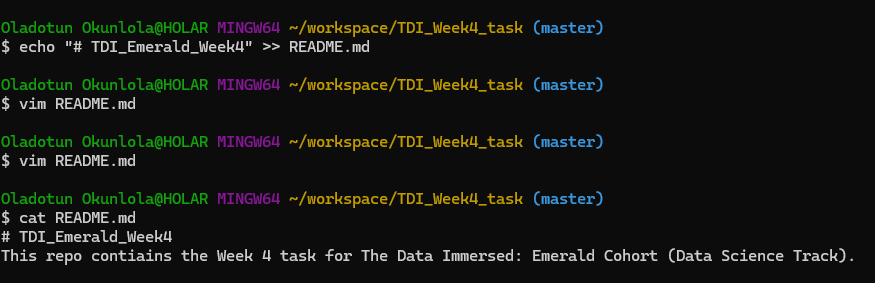
1. Click on **Add People** and type the username of the collaborator and select

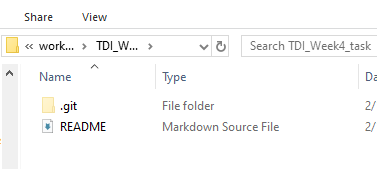


1. After selecting the collaborator, click on Add.



1. In your repository, add a README.md file using a local repository.





GitHub repository: <https://github.com/yusufokunlola/TDI_Emerald_Week4>