Python Machine Learning Project with Git & GitHub for version control

Parameswari Manthiramoorthi

IITP_AIMLTN_2602771

14-Feb-2026

Question:

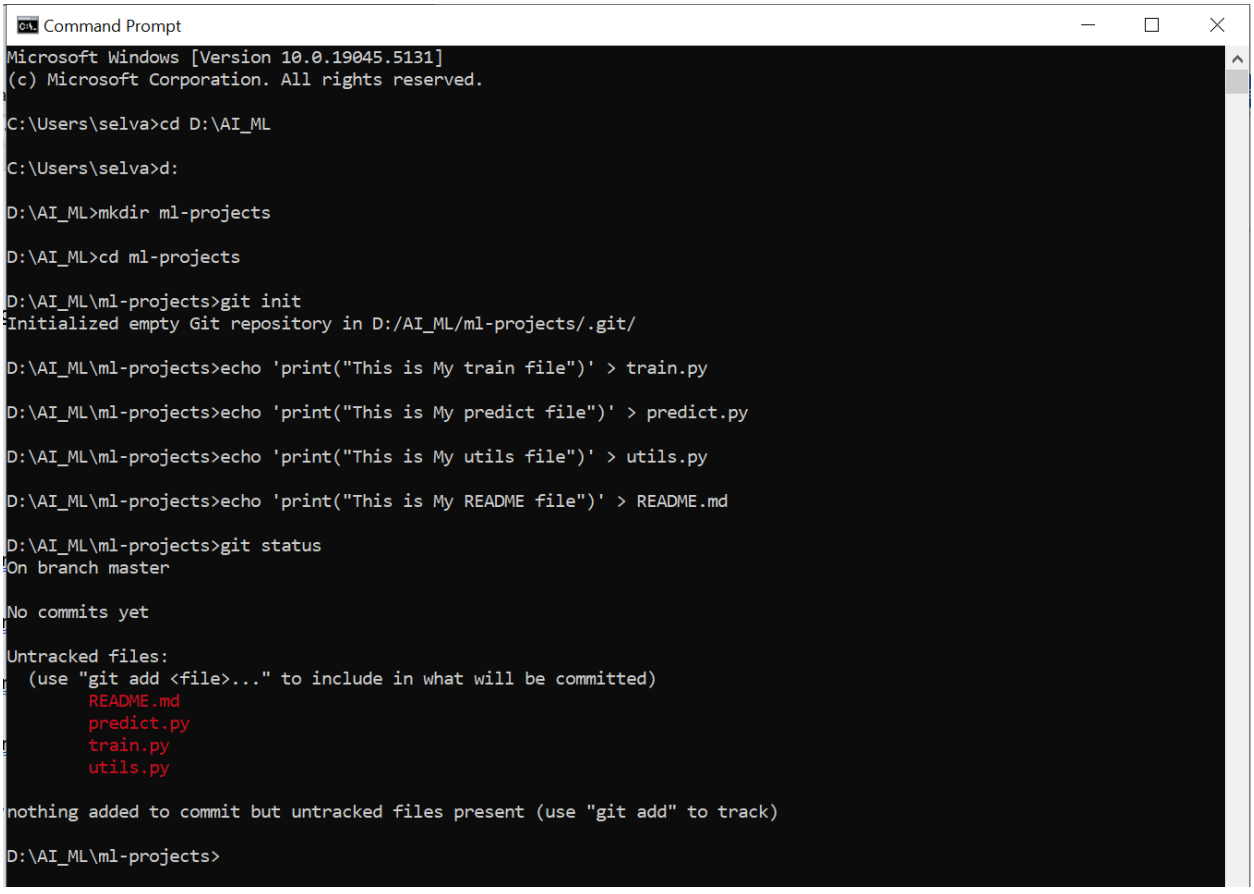
You are developing a Python machine learning project and want to use Git and GitHub for version control. The project will have the following files:

- train.py (model training script)
- predict.py (prediction script)
- utils.py (helper functions)
- README.md (project documentation)

Complete the following three stages:

Stage 1: Basic Setup (Foundation) :

1. create a new directory called ml-project
2. Navigate into that directory
3. Initialize a Git repository
4. Create all four files mentioned above (you can create empty files)
5. Check the status of your Git repository



Explanation:

Created project folder, initialized Git, added all files, and checked the status.

Stage 2: Version Control Workflow (Application)

```
D:\AI_ML\ml-projects>git add train.py utils.py
```

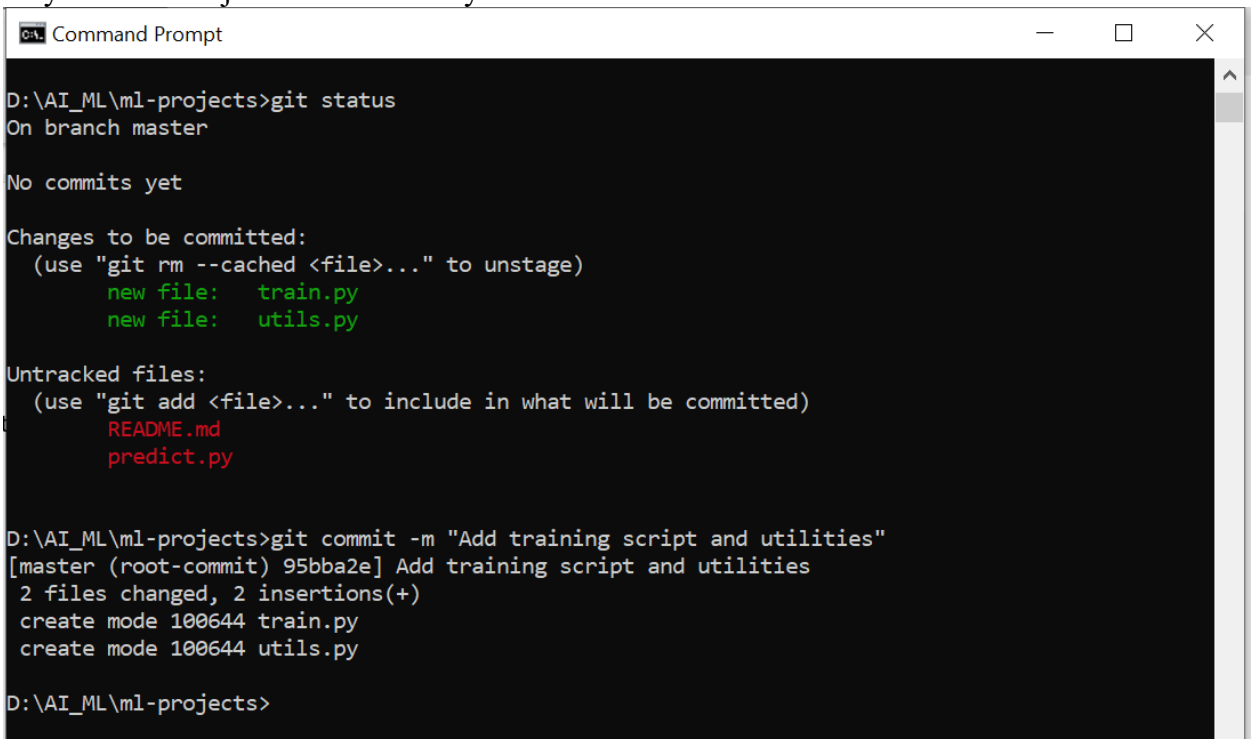
Explanation:

We tell Git to **track only the files we want to save** for now. in this case, `train.py` and `utils.py`. Other files like `README.md` and `predict.py` will not be included yet.

```
D:\AI_ML\ml-projects>git add train.py utils.py

D:\AI_ML\ml-projects>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   train.py
        new file:   utils.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        predict.py


D:\AI_ML\ml-projects>
```

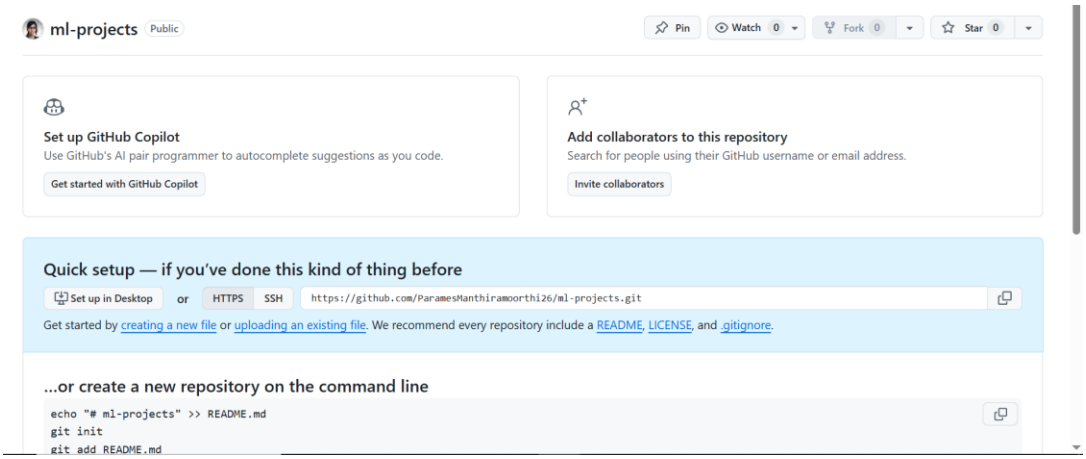
Explanation:

This command helps us **see which files are staged, unstaged, or untracked**. It confirms that only the files we just added are ready to commit.



We now **save our staged changes locally** in Git with a clear message describing what we did.

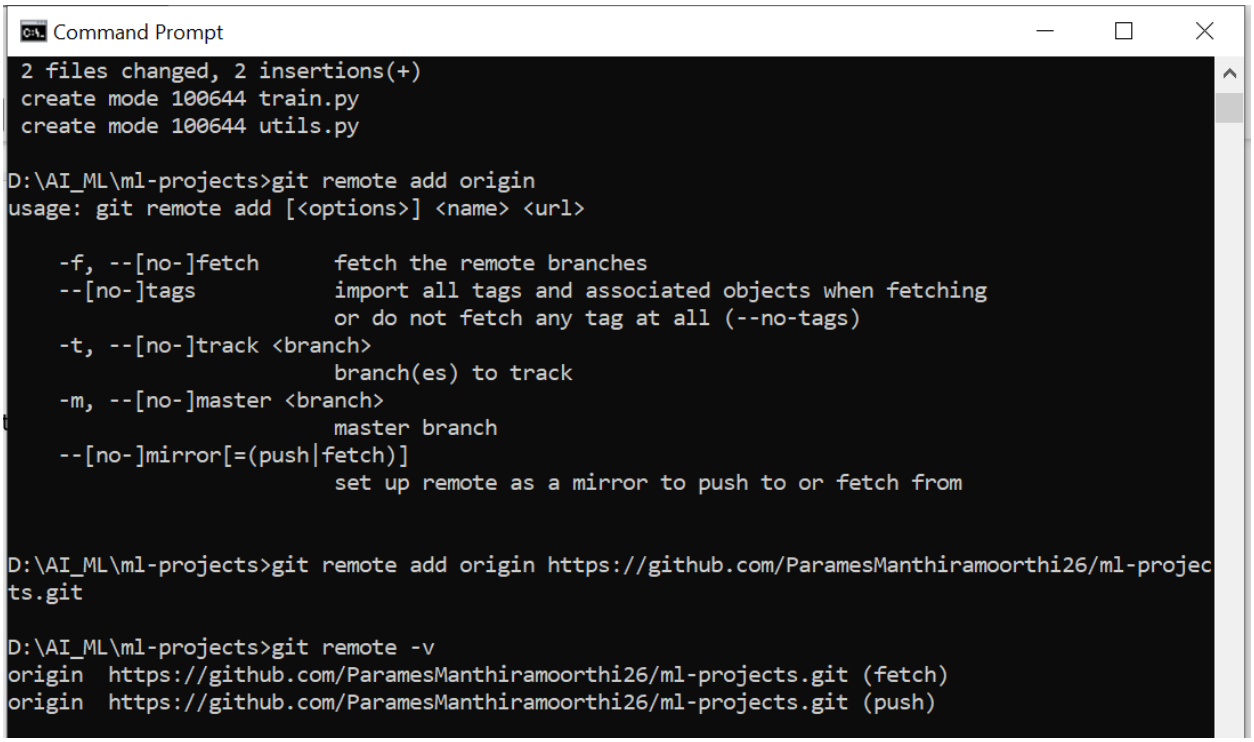
Create a repository on GitHub called ml-projects :



Explanation:

A Repository on GitHub called ml-projects Created successfully.

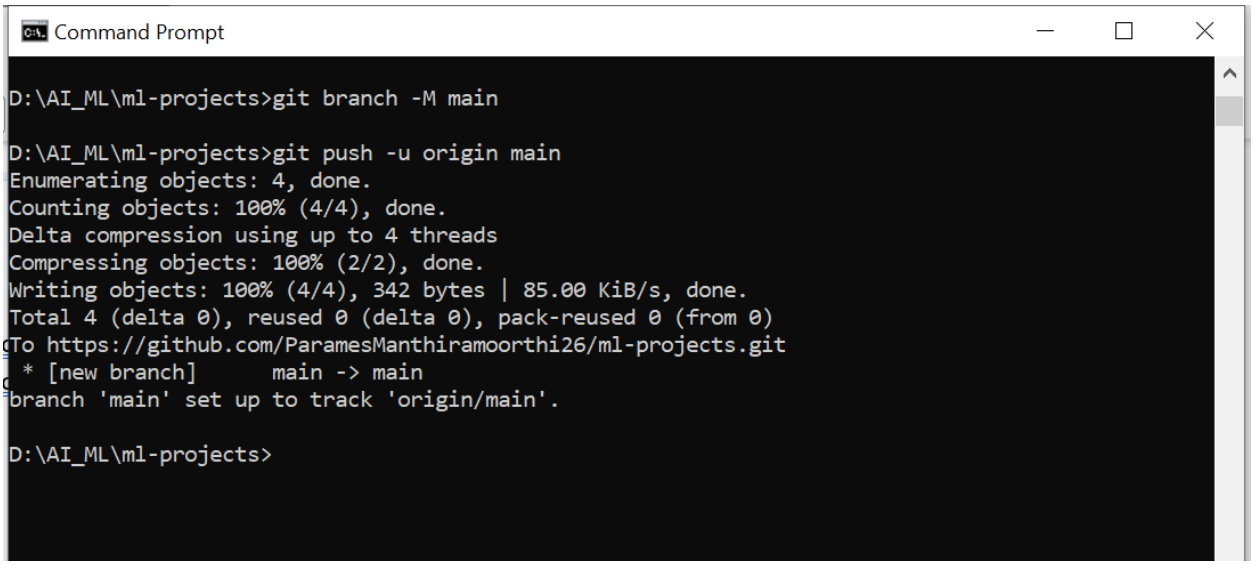
Link My local repository to this GitHub repository :



Explanation:

- Connect our local project to the remote repository on GitHub, giving it the name origin for easy reference. (git remote add origin https://github.com/ParamesManthiramoorthi26/ml-projects.git)
- This shows the URLs for our remote repositories. We can **confirm that Git is pointing to the correct GitHub project**.

Push my commits to GitHub :



Explanation:

- Rename our branch to main, which is now the standard default branch name in GitHub projects.
- Finally, we upload our local commits to GitHub.
- The -u flag makes future pushes easier because Git will remember that main tracks origin/main.

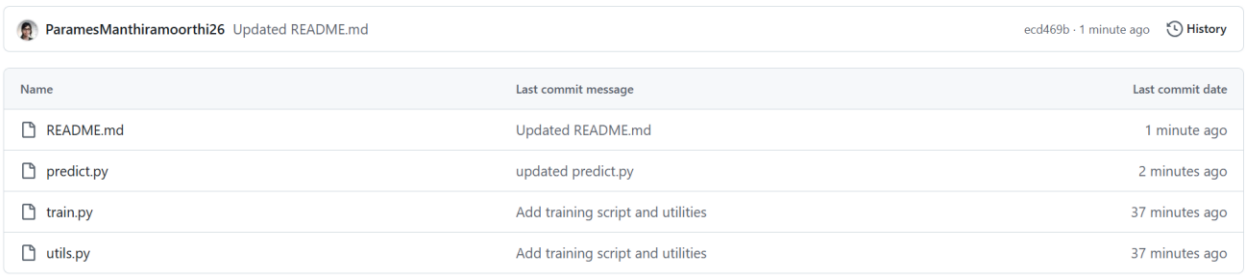
Expected Output:

- Only the train.py and utils.py scripts are committed.
- The local repository is connected to GitHub.
- Remote main branch now contains our initial work.

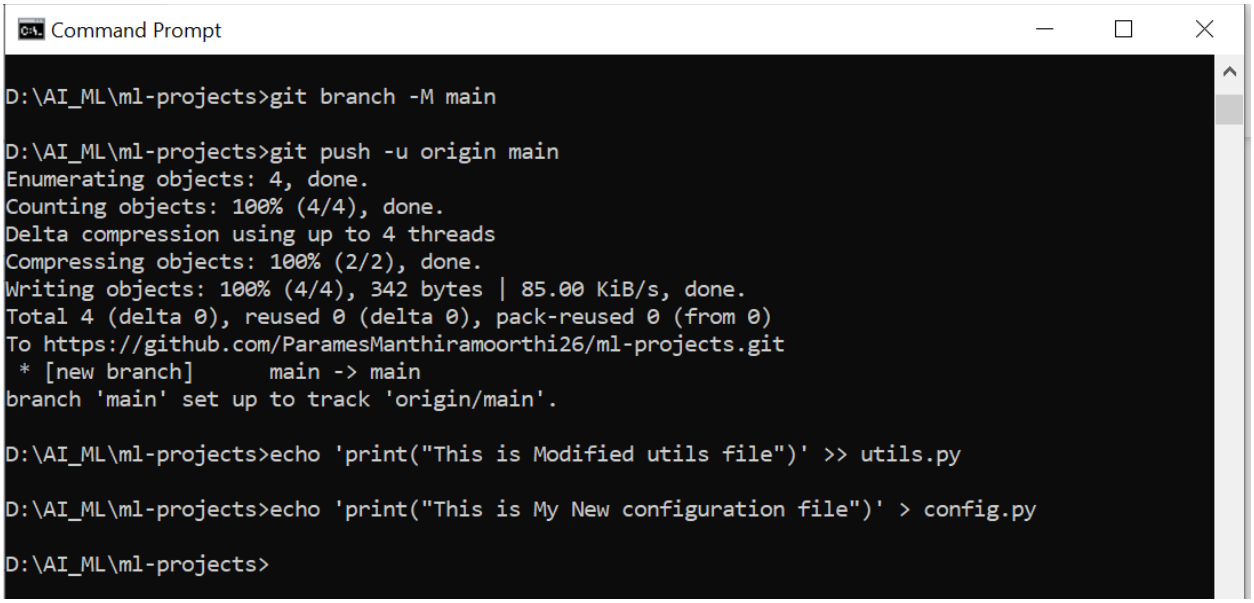
Stage 3: Mini-Project - Collaborative Workflow (Synthesis)

In this stage, we simulate a collaborative workflow:

- The remote repository on GitHub already has updates to predict.py and README.md (added manually on GitHub).



- Locally, we have modified utils.py and created a new file config.py.



The goal is to **safely integrate the remote changes into our local repository, commit local updates, and push everything back to GitHub**.

Step 1: Retrieve the latest changes from GitHub :

```
D:\AI_ML\ml-projects>git pull origin
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (6/6), 1.89 KiB | 80.00 KiB/s, done.
From https://github.com/ParamesManthiramoorthi26/ml-projects
   95bba2e..ecd469b  main       -> origin/main
Updating 95bba2e..ecd469b
error: The following untracked working tree files would be overwritten by merge:
        README.md
        predict.py
Please move or remove them before you merge.
Aborting

D:\AI_ML\ml-projects>
```

Explanation:

- This command downloads the latest commits from the remote main branch and merges them into our local branch.
- Even though we made these changes ourselves on GitHub, Git still treats them as remote updates.
- Pulling first ensures our local branch is up-to-date, avoiding push errors.

Step 2: Stage local changes :

```
Aborting

D:\AI_ML\ml-projects>git add utils.py config.py
```

Explanation:

- We tell Git to track our modified and new files locally.
- Staging ensures only the files we want are included in the next commit.

Step 3: Commit the staged changes:

```
D:\AI_ML\ml-projects>git add utils.py config.py

D:\AI_ML\ml-projects>git commit -m "Update utils.py and add config.py"
[main 7162a48] Update utils.py and add config.py
 2 files changed, 2 insertions(+)
 create mode 100644 config.py

D:\AI_ML\ml-projects>
```

Explanation:

- This saves our staged changes permanently in the local repository.
- The commit message clearly describes what was done for future reference.

Step 4: Push changes to GitHub:

```
D:\AI_ML\ml-projects>git push origin main
To https://github.com/ParamesManthiramoorthi26/ml-projects.git
 ! [rejected]        main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/ParamesManthiramoorthi26/ml-projects.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. If you want to integrate the remote changes,
hint: use 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

D:\AI_ML\ml-projects>
```

- We got following issues , because git prevent overwrite to github rebo

```
D:\AI_ML\ml-projects>
D:\AI_ML\ml-projects>git pull origin main
From https://github.com/ParamesManthiramoorthi26/ml-projects
 * branch            main       -> FETCH_HEAD
error: The following untracked working tree files would be overwritten by merge:
        README.md
        predict.py
Please move or remove them before you merge.
Aborting
Merge with strategy ort failed.
```

- **This is a real merge conflict ,**I created **README.md** and **predict.py** locally and ALSO created/edited the same files directly on GitHub Repository, that's why Git doesn't know **which version is correct**, so it asks me to decide.

- **Now Check Which Files Have Conflicts**

```
D:\AI_ML\ml-projects>git status
On branch main
Your branch and 'origin/main' have diverged,
and have 2 and 2 different commits each, respectively.
  (use "git pull" if you want to integrate the remote branch with yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both added:      README.md
        both added:      predict.py

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

D:\AI_ML\ml-projects>
```

- **Now we clearly see these both files have conflicts (README.md , predict.py).**
- **Open README.md ,we will see something like this inside (<<<<<<< HEAD, =======, >>>>>>> origin/main)**

```
D:\AI_ML\ml-projects>type README.md
<<<<<<< HEAD
'print("This is My README file")'
=======
This is the updated ReadMe file.
>>>>>>> origin/main
```

- We must **edit the file manually** and remove the markers.
- Open notepad then remove conflict markers and save the correct code.
- Next Tell Git the Conflict Is Resolved.

```
D:\AI_ML\ml-projects>git add README.md predict.py
```

- Complete the Merge Commit

```
D:\AI_ML\ml-projects>git commit -m "Resolved merge conflict in README and predict"
[main bdecb25] Resolved merge conflict in README and predict
```

- Push to GitHub

```
D:\AI_ML\ml-projects>git push origin main
Enumerating objects: 18, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 4 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.28 KiB | 187.00 KiB/s, done.
Total 12 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/ParamesManthiramoorthi26/ml-projects.git
   ecd469b..bdecb25  main -> main
```

Final Workflow Summary

Step 1: Pull the Latest Changes git pull origin main Start by downloading the newest updates from GitHub. This ensures your local project is up to date before you add your own work. It helps avoid conflicts and keeps everyone aligned.

Step 2: Check Your Current Changes git status Review which files you have modified or created. This step helps you confirm that you are committing only the intended changes.

Step 3: Stage the Required Files git add utils.py config.py Add only the specific files you want to include in this update. Staging allows you to control what goes into each commit and keeps the history clean.

Step 4: Commit Your Work git commit -m "Update utils and add config" Save the staged changes with a meaningful message. A clear commit message helps others understand what was updated and why.

Step 5: Push Changes to GitHub git push origin main Upload your committed changes to GitHub so the remote repository reflects your latest work and stays synchronized.

Best Practice: Follow this order every time — **Pull → Review → Stage → Commit → Push** — to maintain a smooth and organized collaborative workflow.