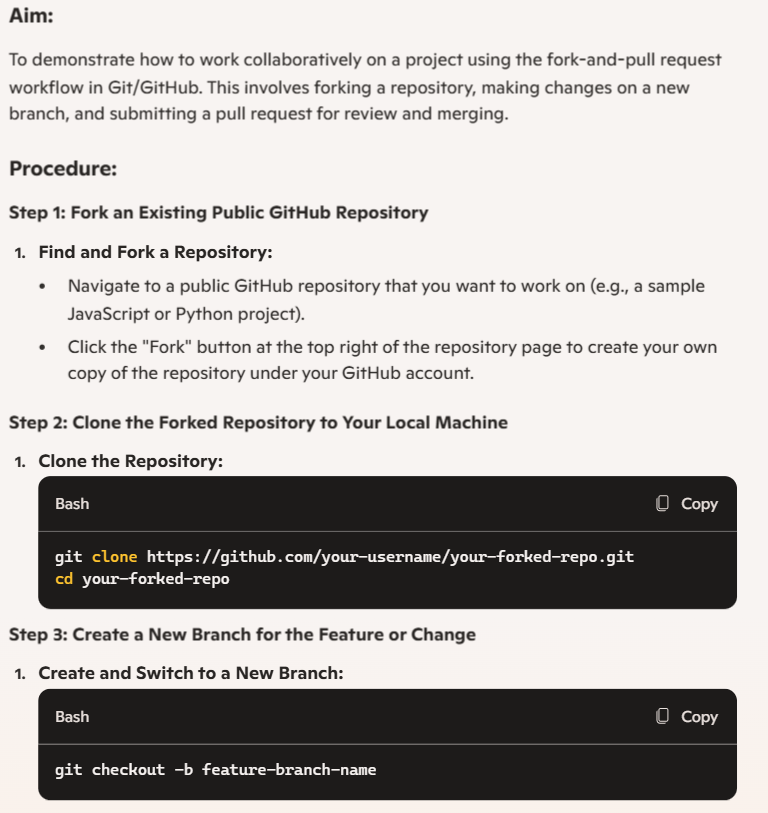
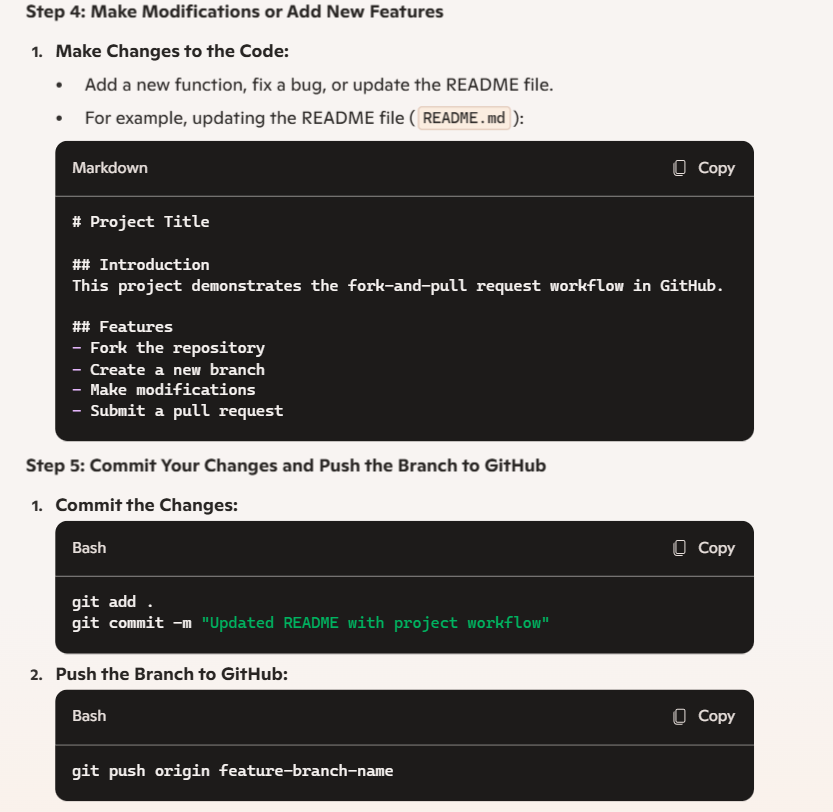
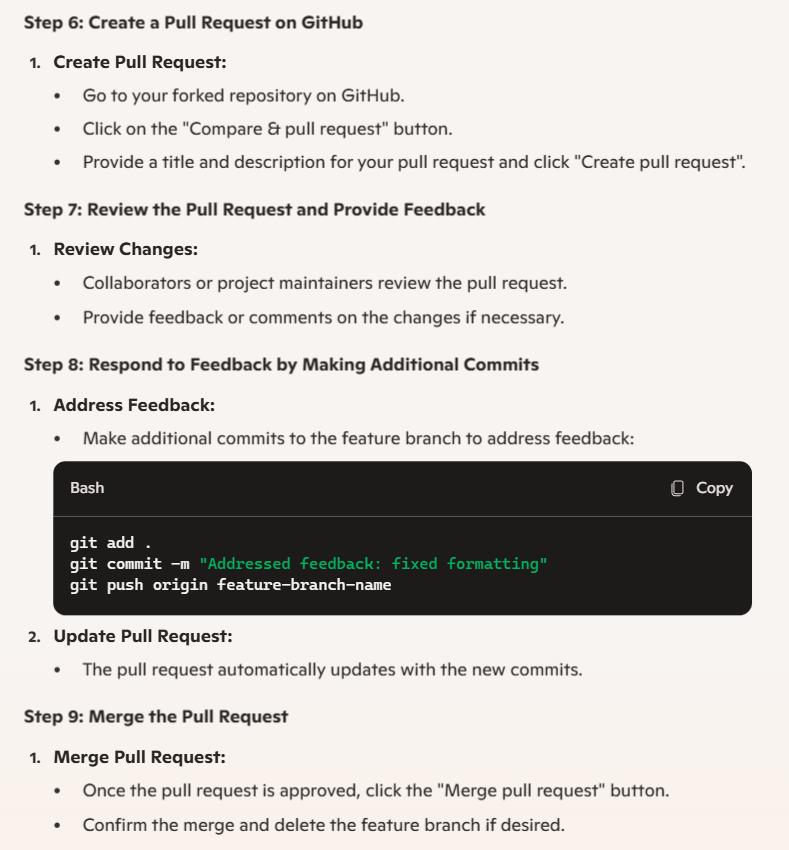
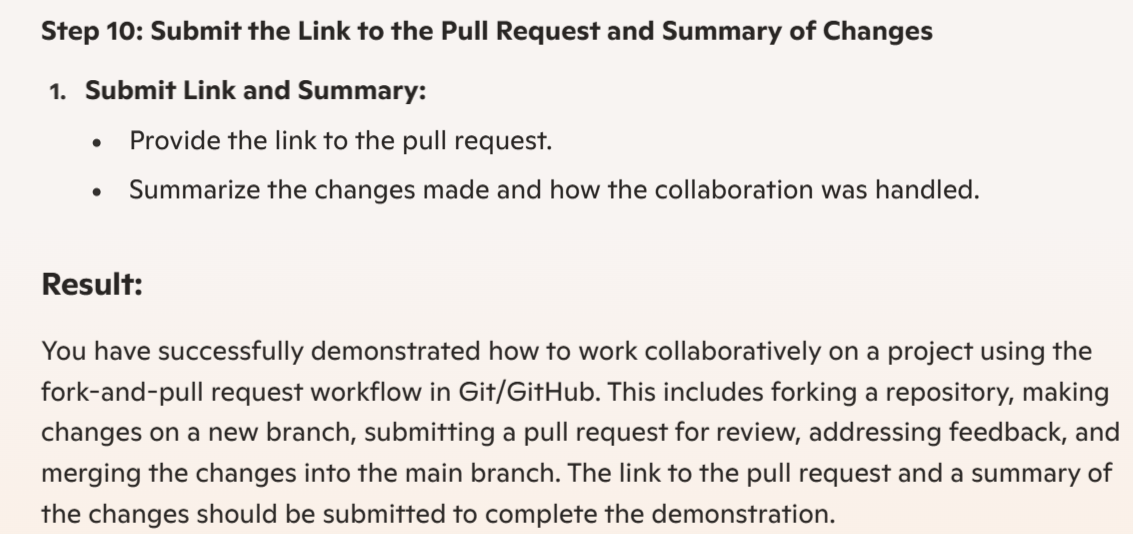
**GIT & GITHUB**

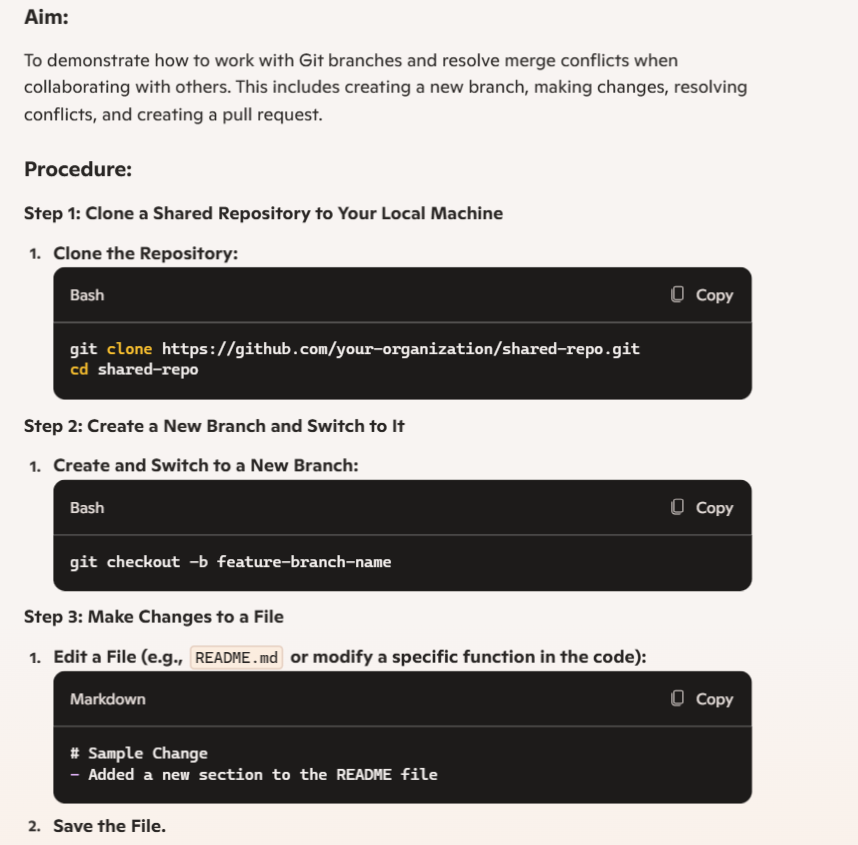
1. Demonstrate how to work collaboratively in Git/GitHub on a project using the fork-and-pull request workflow. Tasks: 1. Fork an existing public GitHub repository (e.g., a sample JavaScript or Python project). 2. Clone the forked repository to your local machine using Git. 3. Create a new branch for the feature or change you want to work on. 4. Make modifications or add new features (e.g., add a function, fix a bug, or update the README). 5. Commit your changes and push the branch to GitHub. 6. Go to the GitHub repository and create a pull request to merge your feature branch into the main branch. 7. Review the pull request and provide feedback on the changes. 8. Respond to feedback by making additional commits to the Git/GitHub CO3 feature branch if necessary. 9. Once the pull request is approved, merge it into the main branch. 10. Finally submit the link to the pull request along with a summary of the changes you made and how you collaborated.

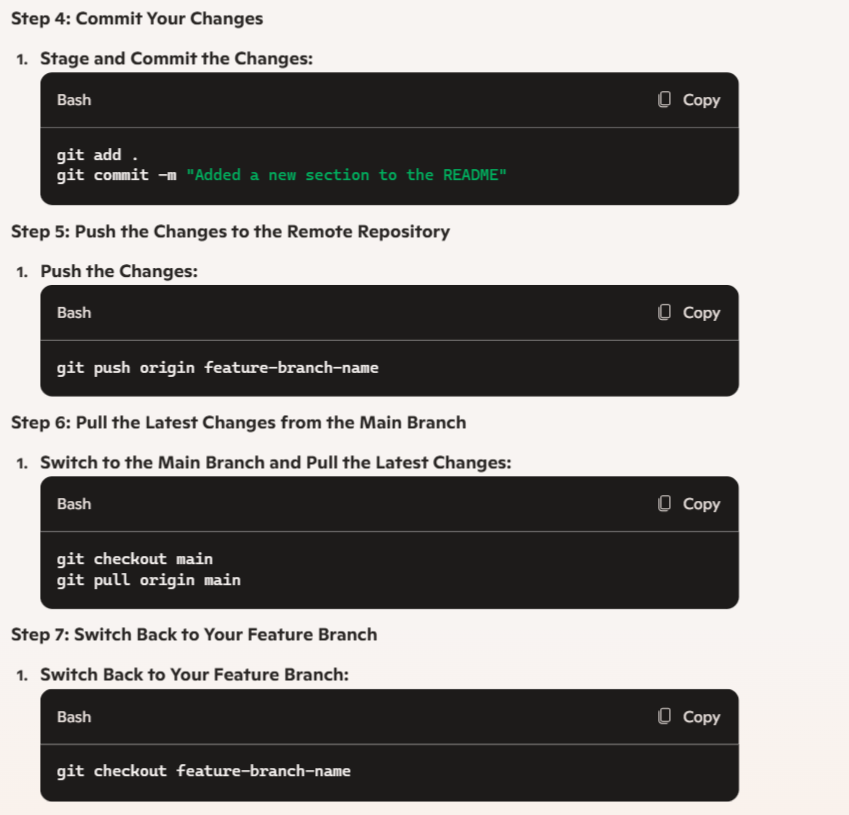


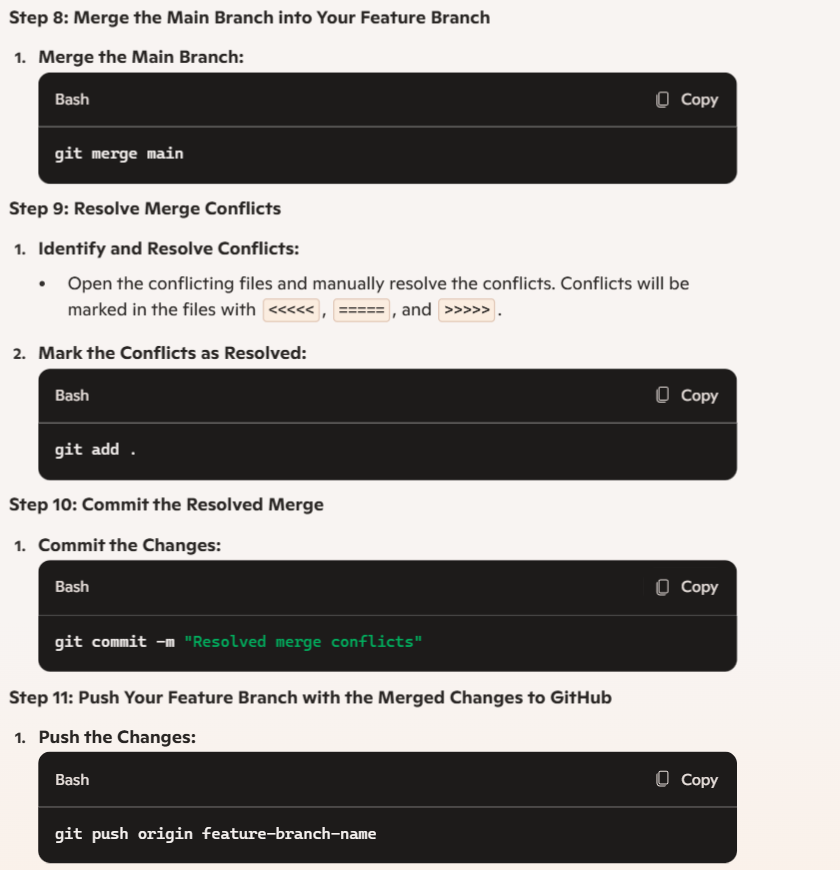


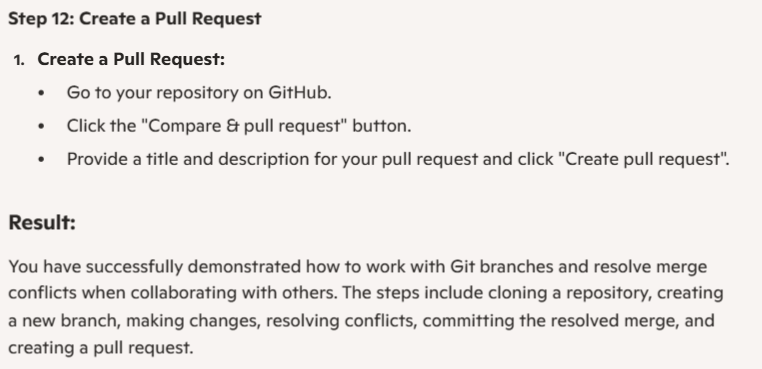
 

1. Demonstrate how to work with Git branches and resolve merge conflicts when collaborating with others. Tasks: 1. Clone a shared repository to your local machine. 2. Create a new branch and switch to it. 3. Make changes to a file (e.g., update a README or modify code in a specific function). 4. Commit your changes. 5. Push the changes to the remote repository. 6. Before merging, pull the latest changes from the main branch. 7. Switch back to your feature branch. 8. Merge the main branch into your feature branch. 9. If there are conflicts, resolve them manually by editing the conflicting files. After resolving conflicts, mark the conflicts as resolved. 10. Commit the resolved merge. 11. Push your feature branch with the merged changes to GitHub and create a pull request. 12. Submit a summary of the steps you performed, the conflicts you encountered, and how you resolved them.

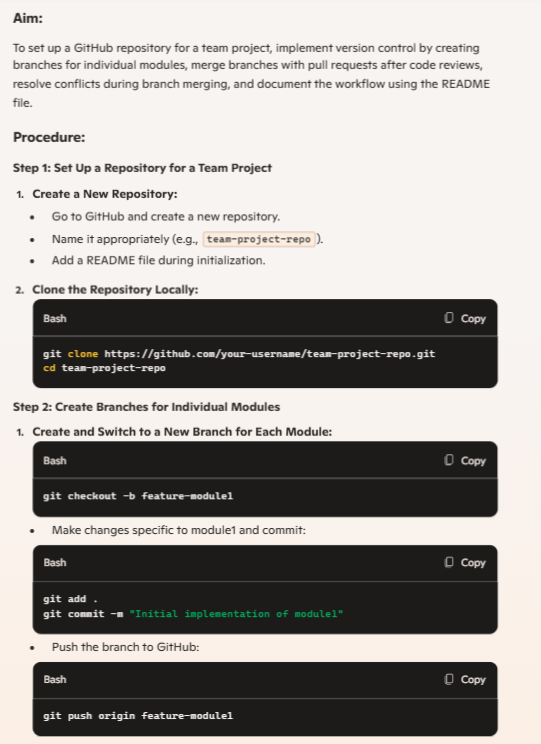


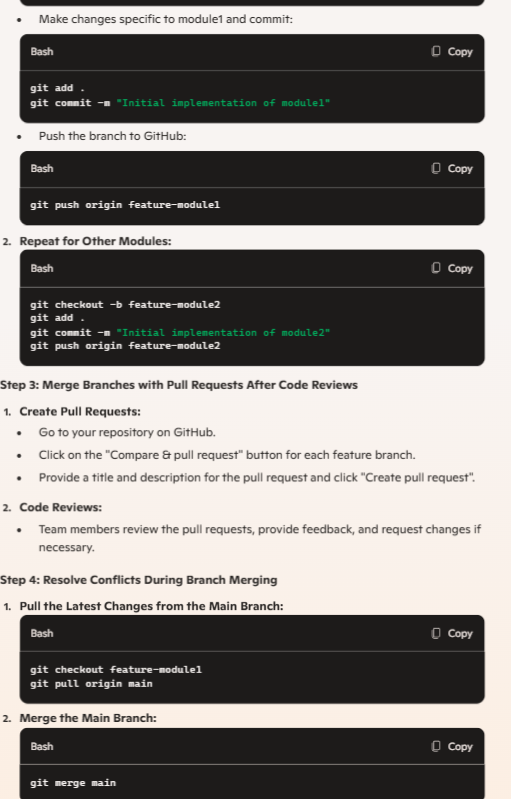
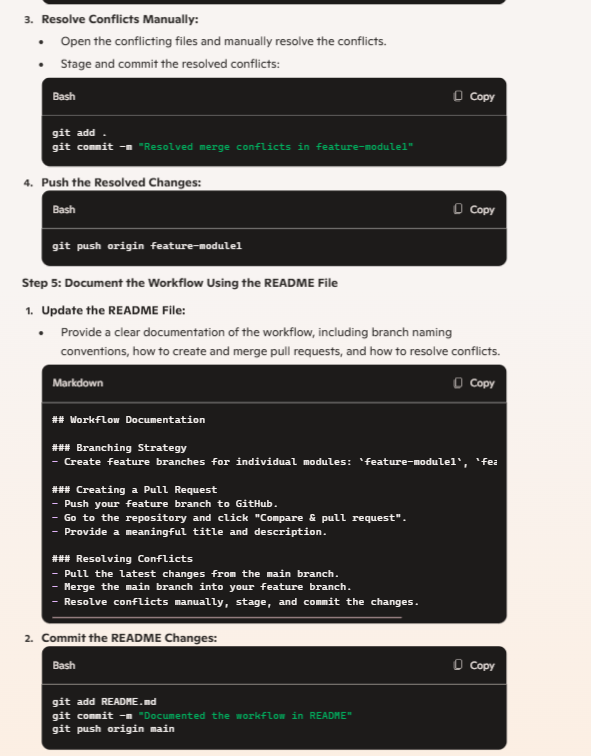
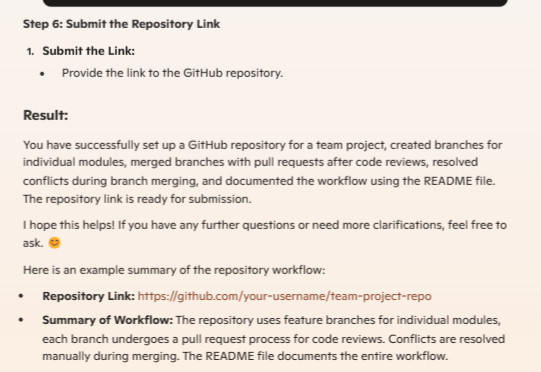




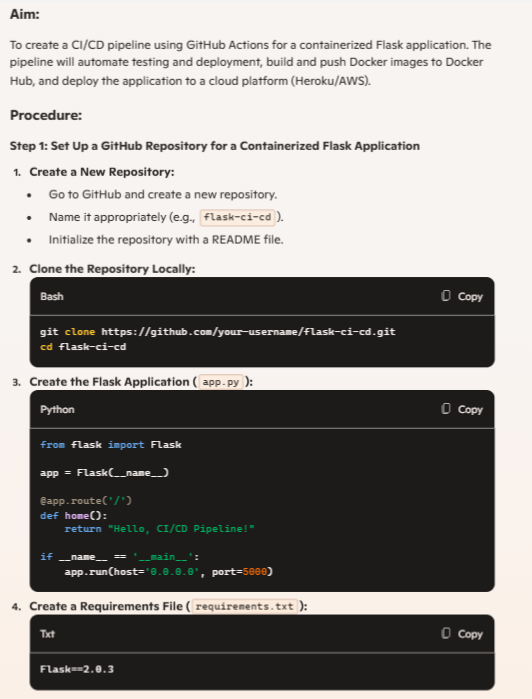


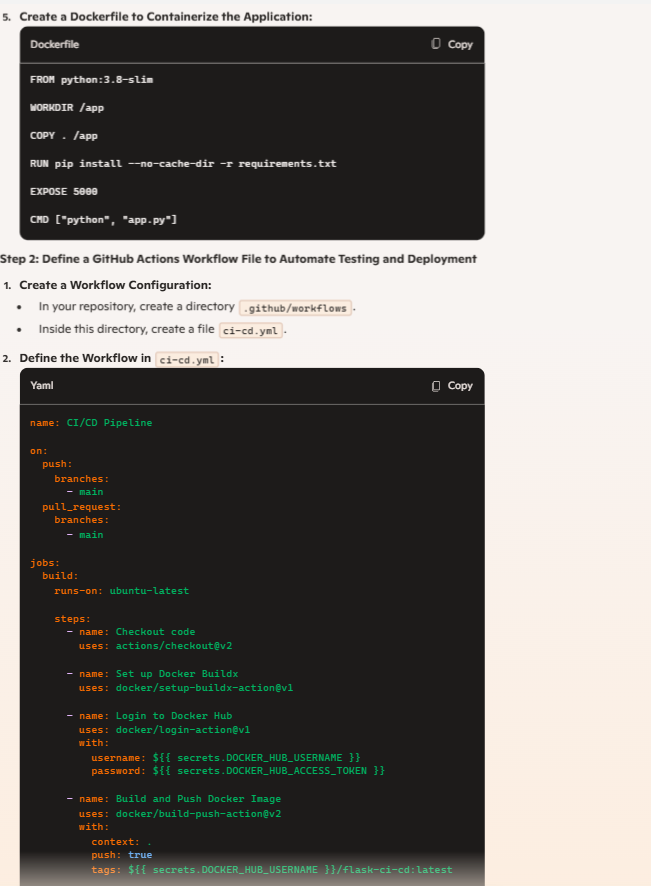
1. Create a GitHub Repository and Implement Version Control Tasks: • Set up a repository for a team project. • Create branches for individual modules. • Merge branches with pull requests after code reviews. • Resolve conflicts during branch merging. • Document the workflow using the README file. • Submit the repository link



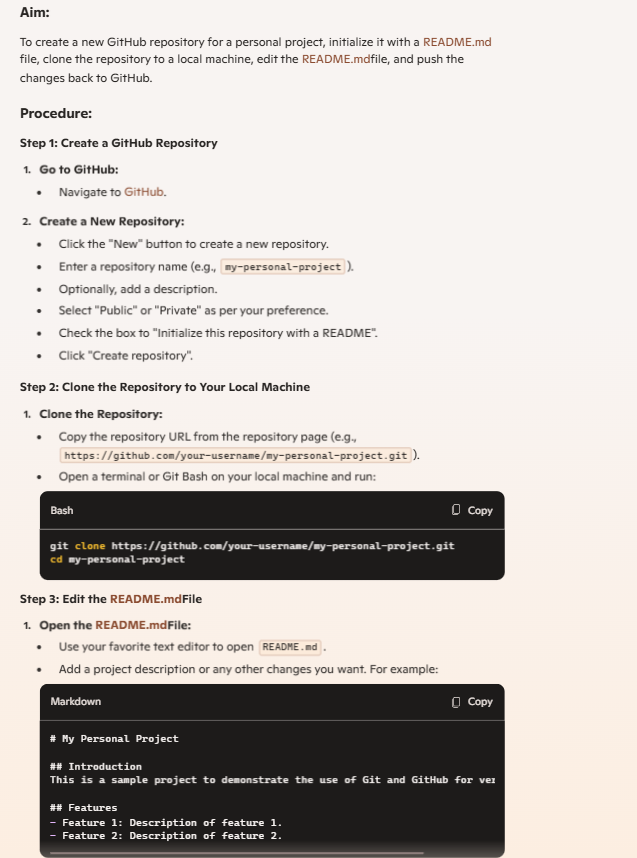
1. Create a CI/CD Pipeline Using GitHub Actions Tasks: • Set up a GitHub repository for a containerized Flask application. • Define a GitHub Actions workflow file to automate testing and deployment. • Add steps to build, test, and push Docker images to Docker Hub. • Deploy the application to a cloud platform (Heroku/AWS). • Submit the workflow file and deployment link.

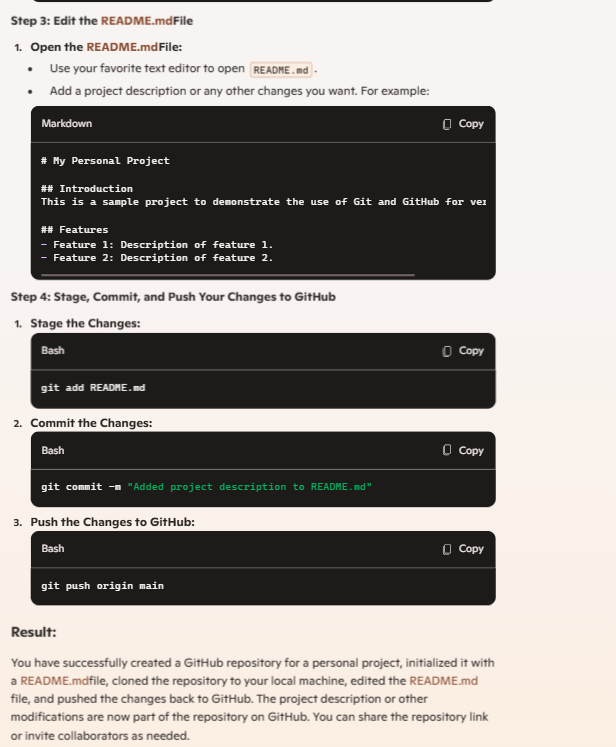




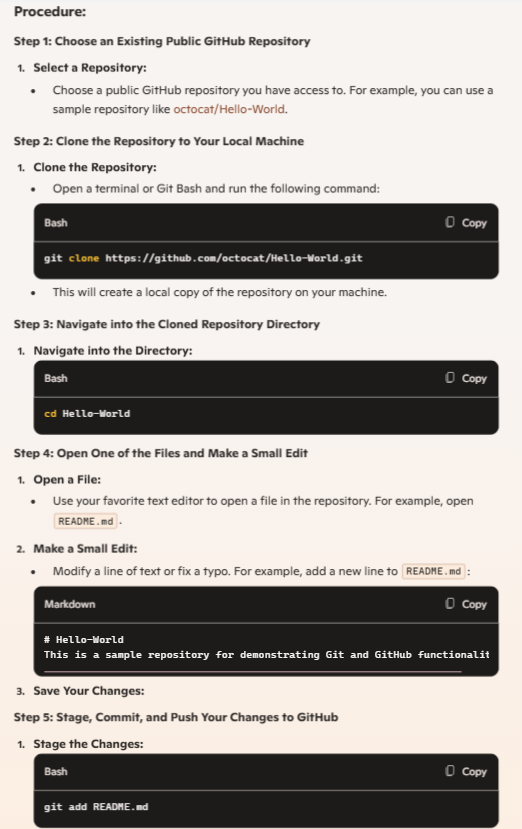


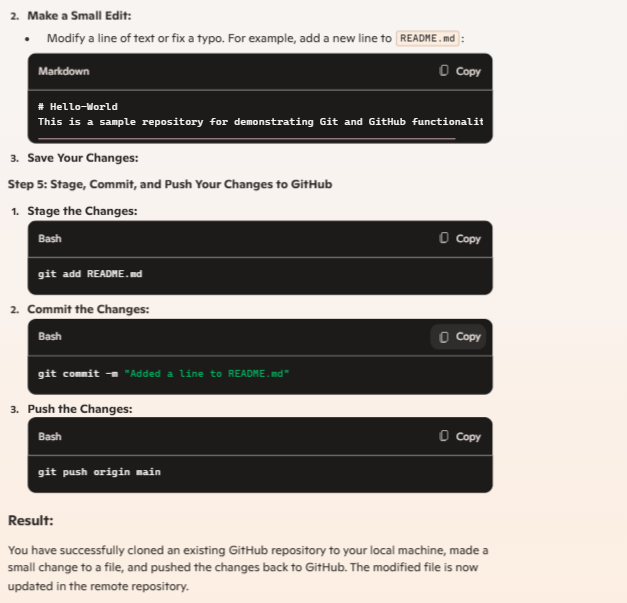
1. Create a new GitHub repository for a personal project. Initialize the repository with a README.md file. Then, clone the repository to your local machine, make some changes to the README.md (e.g., add a project description), and push the changes back to GitHub. Tasks: • Create a GitHub repository. • Clone the repository to your local machine. • Edit the README.md file. • Stage, commit, and push your changes to GitHub.



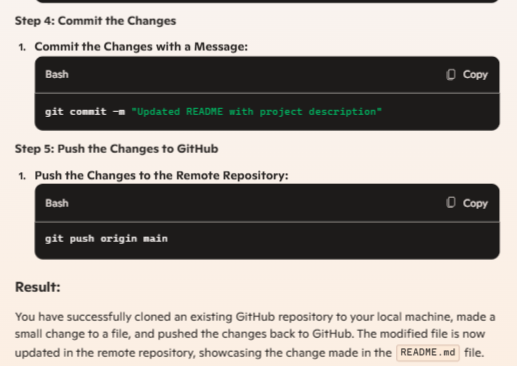


1. Clone an existing GitHub repository to your local machine and make a small change in one of the files. Tasks: • Choose an existing public GitHub repository (e.g., your own or a sample repository). • Clone the repository to your local machine using git clone . • Navigate into the cloned repository directory and open one of the files. • Make a small edit, such as modifying a line of text or fixing a typo. • Save your changes.

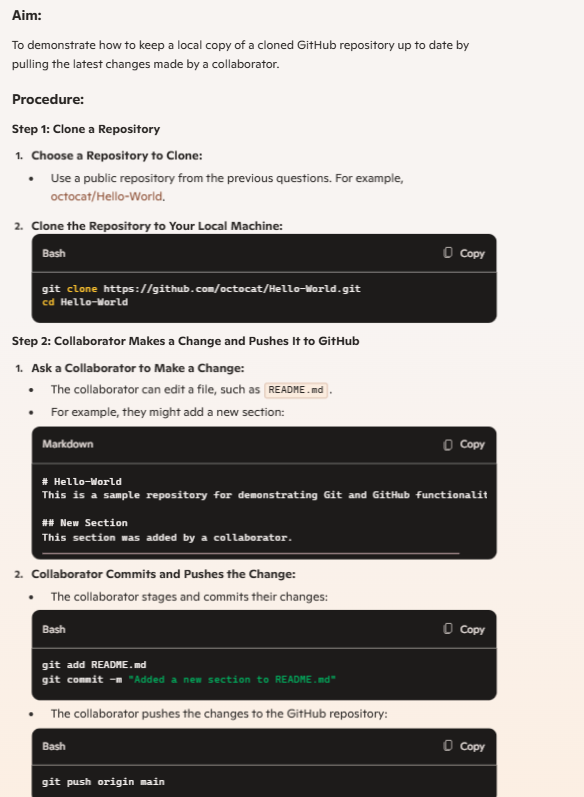


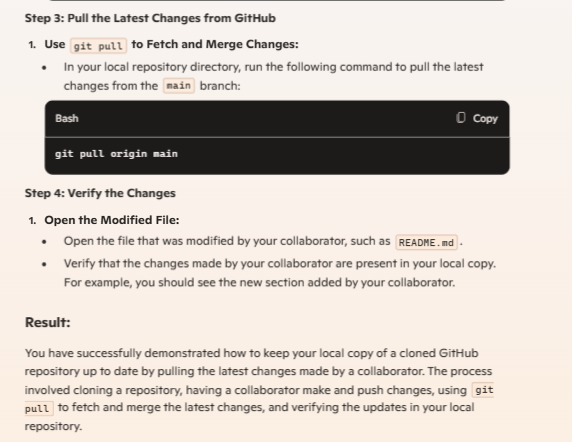


1. After modifying a file in a cloned repository, commit the changes locally and push them to GitHub. Tasks: • Clone a repository (from question 1). • Modify a file (e.g., README.md). • Stage the changes with git add. • Commit the changes with git commit -m "Updated README". • Push the changes to GitHub using git push.

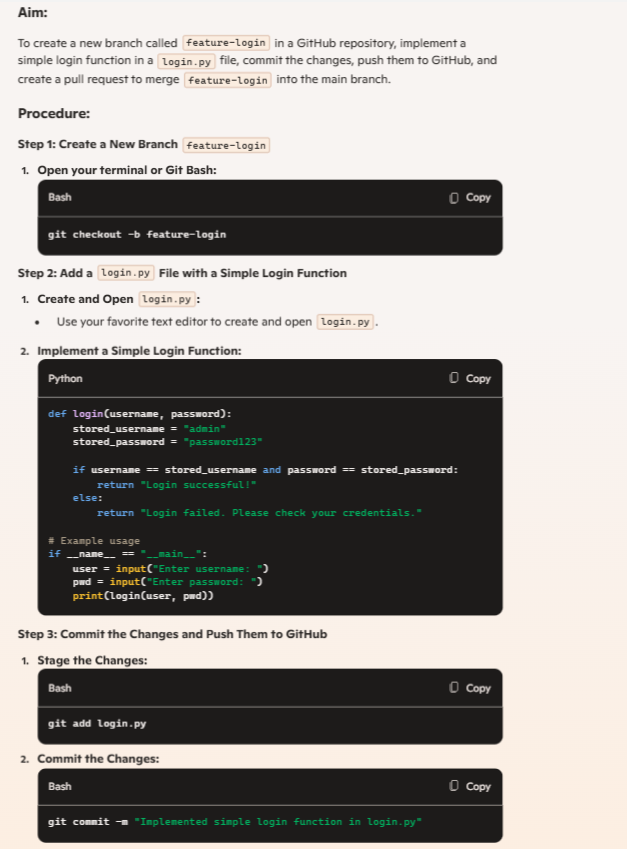
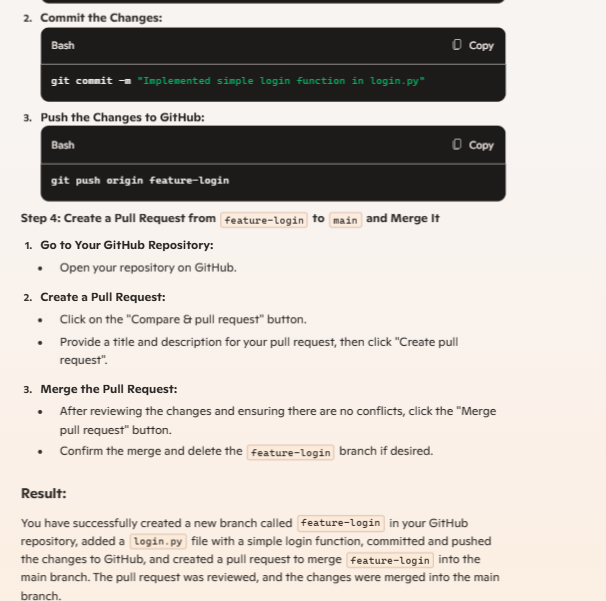
 

1. You cloned a repository earlier, and now another collaborator has made changes to it. You need to pull the latest changes from GitHub to keep your local copy up to date. Tasks: • Clone a repository (from previous questions). • Ask a collaborator to make a change and push it to the GitHub repository. • Use git pull origin main to pull the latest changes. • Verify that the changes made by your collaborator appear in your local repository.





9) In your GitHub repository, create a new branch called feature-login. Implement a simple login function in a login.py file. After completing the function, create a pull request to merge feature-login into the main branch. Tasks: • Create a new branch feature-login. • Add a login.py file with a simple login function. • Commit the changes and push them to GitHub. • Create a pull request from feature-login to main and merge it.

1. You and your team are working on a project. One team member adds a new feature to the project. You need to fetch the latest changes, create a new branch, and implement your feature without affecting the existing code. Tasks: • Fork a repository (you can use an existing repository or a partner’s repository). • Clone the repository to your local machine. • Create a new branch for your feature. • Implement your changes and push them to your forked repository. • Submit a pull request to the original repository.

