Git Help

/alanabarrettfrew

teacherturnsturtle81

My First Project

As I transitioned into the corporate world, I faced the challenge of adapting from working independently on GitHub during my bootcamp and personal projects to collaborating in a larger, more complex environment in GitLab. Both use git commands, but initially, I was concerned about making mistakes in this new setting and doing something to spoil shared code. All imposter syndrome really.

My early projects have all involved cloning the repository, installing all necessary requirements, creating my own branch, resolving issues or tickets, and then merging my code. This process has helped build my confidence, allowing me to focus more on the quality of my code for subsequent tasks.

I hope this reference guide proves helpful. For context I am cloning the code from GitLab, however the same would apply to clone the code from Github. I am using VS Code as my IDE.

/alanabarrettfrew

teacherturnsturtle81

Project Set Up

Here is a guide for setting up a full-stack AI Virtual Assistant app using Git, Python, and React in your local environment:

- Create and Name a Folder in C Drive
  - Navigate to your C drive.
  - Right-click to create a new folder and name it appropriately, such as AI_Virtual_Assistant.
- Open Git Bash in the Folder
  - Right-click on the newly created folder and select "Open Git Bash Here" to open Git Bash in that location.
- Clone the Repository Using HTTPS. If the repos are separate, repeat the command for each folder
  - In Git Bash, execute the following command to clone the repository. in this case there are 3 repos to clone:
  - backend
    - git clone <repo web url>
  - frontend
    - git clone <repo web url>
  - preprocessing
    - git clone <repo web url>

/alanabarrettfrew

teacherturnsturtle81

- Once all cloned, open in VS Code
    - code .
  - Open the terminal (this will open in powershell which I have all sorts of issues with, so I open a git bash terminal.
- Setup for Backend (Python)
  - Create and Activate Virtual Environment in root folder
    - python -m venv venv
    - venv\Scripts\activate
  - Install Backend Requirements
    - pip install -r <backend path>/requirements.txt
  - Open a New Terminal and Rename it to 'backend'
  - Navigate to Backend Folder and Run App
    - cd <backend path>
    - python app.py
  - Ensure the backend is running correctly.
- Setup for Frontend (React)
  - Open a New Terminal and Rename it to 'frontend'
  - Navigate to Frontend Folder and Install Packages
    - cd <frontend path>
    - npm install
  - Run the Frontend to Ensure it Works
    - npm run dev
- Setup for Preprocessing (if applicable)
  - Install Preprocessing Requirements
    - pip install -r <preprocessing path>/requirements.txt

/alanabarrettfrew

teacherturnsturtle81

- Remember to run npm run build for any changes made to the frontend to ensure they are properly stored in the static files of the backend.
- By following these steps, you will have a fully operational environment for your AI Virtual Assistant app, with both backend and frontend components running smoothly. ctrl +c to stop running them. In this example I have 3 repos in one folder. In others all the code for the 3 sections might be in one repo. It depends on the structure.
- When you can start to work on the project make sure you are in the repo main branch to start with and you have pulled latest code changes.
  - Create a new branch for whichever repo you are in
    - git checkout -b <new branch name>
  - Pull changes from main branch (or other selected branch to your feature branch)
    - git pull origin <branch name>
  - Start working on new branch
  - Commit code and when ready got to GitLab repo and create a merge request
    - git add .
    - git commit –m "<commit message>"
    - git push
- NOTE: on first push you may need something like this:
    - git push --set-upstream origin ABF-persistChat
- After that you should just be able to use git push.

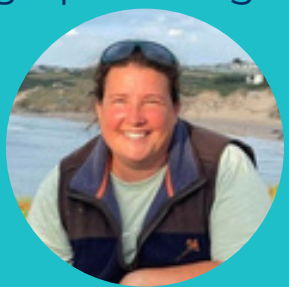
/alanabarrettfrew

teacherturnsturtle81

Git Issues and Resolutions

- Check which branch you are on
    - git branch
- Switch branch
    - git checkout <branch name>
- Pull latest code changes to local computer from remote repo
    - git pull origin <target branch>

- Delete a Local Branch
    - List branches to confirm the branch names:
        - git branch
    - Delete a local branch:Use the -d option to delete a branch. If the branch has unmerged changes, you might need to force delete it using the -D option.
        - git branch -d <branch_name>
    - If the branch has unmerged changes and you want to force delete it:
        - git branch -D <branch_name>
- Delete a Remote Branch
    - Use the following command to delete a branch from the remote repository:
        - git push origin --delete <branch_name>



/alanabarrettfrew

teacherturnsturtle81

Git Issues and Resolutions

IF ISSUES and you want to revert to HEAD
Reset to a clean state: To ensure your working directory is in a clean state, reset any changes. This will get rid of any changes you made:

- git reset --hard HEAD

Pull the latest changes from the main branch: Now, you can safely pull the latest changes from the main branch:

- git pull origin <branch name>

Abort the ongoing operation (if any):If there was a merge or rebase in progress, you might need to abort it:

- git merge --abort
- OR
- git rebase --abort

/alanabarrettfrew

teacherturnsturtle81

Documentation

GitHub

GitHub.com Help Documentation

Get started, troubleshoot, and make the most of GitHub. Documentation for new users, developers, administrators, and all of GitHub's products.

GitHub Docs

GitLab Documentation

GitLab product documentation.

gitlab.com

/alanabarrettfrew

teacherturnsturtle81