GITting Started:

**Submitting a pull request:**

1. Contact the git repository(repo) owner to get “collaborator” access to their repo. Once permissions have been granted, you can clone the repo to your local machine to make any changes.
2. Begin cloning the repo by accessing the green “code” button from the main repo page. Copy the repo URL. Use the terminal (MacOS) or Powershell (Windows) to navigate to your desired directory where you’d like to store the local version of this repo. Initialize a git repository using the command: $git init Once you’ve initialized a local repository you can pull down or “clone” from the remote using: $git clone {url}

Graphical user interface, text, application

Description automatically generated

1. Now we have everything from the remote repository in our local environment. We can begin to make our own changes without wrecking our friend’s work by creating our own “branch” to work on. You can check branches of the repo by entering: $git branch This will list all the branches with an asterisk next to the currently active branch (also highlighted in green). Create a new branch to work with by using the command: $git branch {branchName} We then activate this branch with the command: $git checkout {branchName} and make all the changes we need to make.
2. Once we’re happy with changes and we’d like our collaborator to take a look, we have to add, commit, and push changes for them to take effect. You can view a list of modified files by using the command: $git status All modified files that have not been added to the “staging” area will be listed in red. From here, you can select files to add one by one, or you can add all changed files to staging using ‘.’ The two git command options are as follows: $git add {filename} or $git add . Once files have been added to staging, they need to be “committed”. When committing changes to the repository, a message is wrapped up with the changes so anyone looking at the repo in the future has an idea of what these changes were. Commit changes with a message using the following command: $git commit -m “{message}” Obviously the -m flag is to designate the message.
3. Now the changes are ready to be “pushed” to the remote repository for the collaborator to review. To do this use: $git push You’ll need to specify the location for the “upstream” repo, or “origin” where these changes are supposed to go. After all, the branch we made isn’t located in the remote, so we have to tell git to put it there. An example from another project:

Graphical user interface, text, application

Description automatically generated

Figure - This details the message output by the console when trying to "push" a new branch to a remote repo

In this case, the branch name is “scheduler” and the git command to use to push the branch is listed. git push –set-upstream origin {branchName}

1. To complete this workflow, we have to make a pull request (PR) so the collaborator can review, accept, or reject the changes we’ve proposed. To do this, track back to the repo URL via your web browser. Select the branch you just uploaded, and click the pull request button:

Graphical user interface, text, application, chat or text message

Description automatically generated

1. After you’ve submitted your pull request, it’s up to your collaborator to review the code and either accept or reject the PR. When another user submits a branch with changes to YOUR repo, you can see the changes between files by inspecting them where you’ll have a side-by-side display of the file before changes and the file after. Green highlighted text are additions and red are deletions.

A screenshot of a computer

Description automatically generated with medium confidence

1. Code review is done by looking at the changes and selecting the “Review Code” button. Here, the reviewer has options to leave comments for the PR. They can either just leave straight comments, give comments and accept the PR, or give comments and reject the PR.

Graphical user interface, text

Description automatically generated