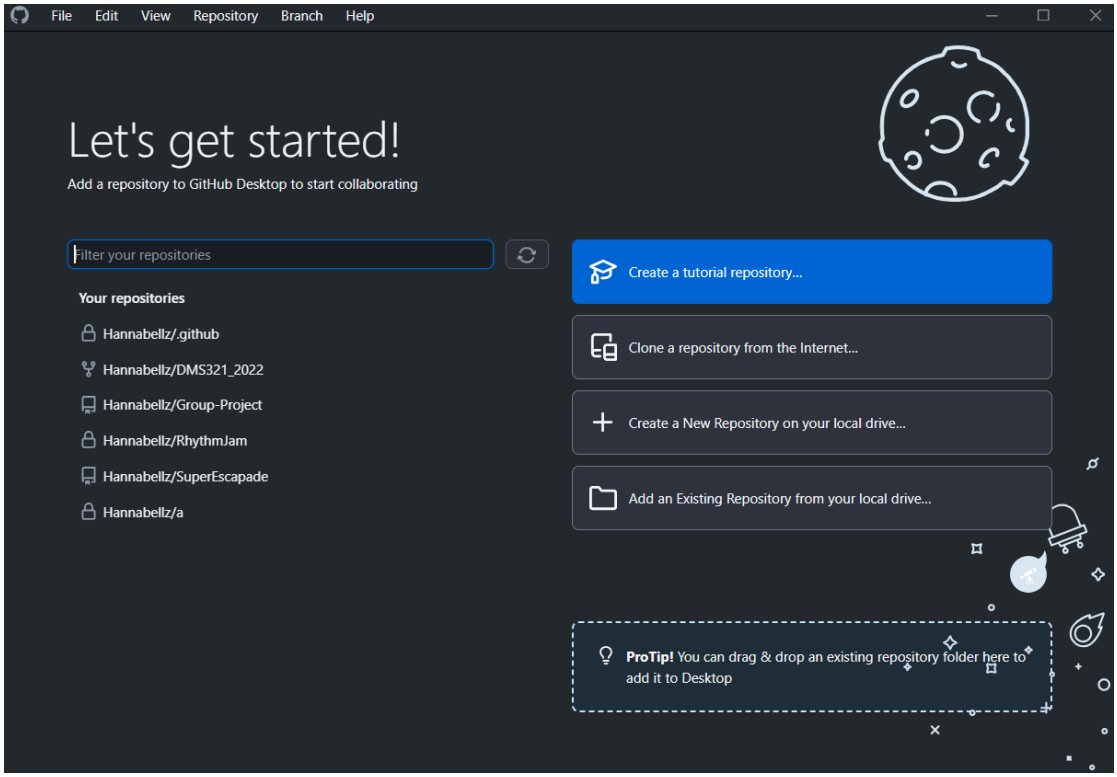
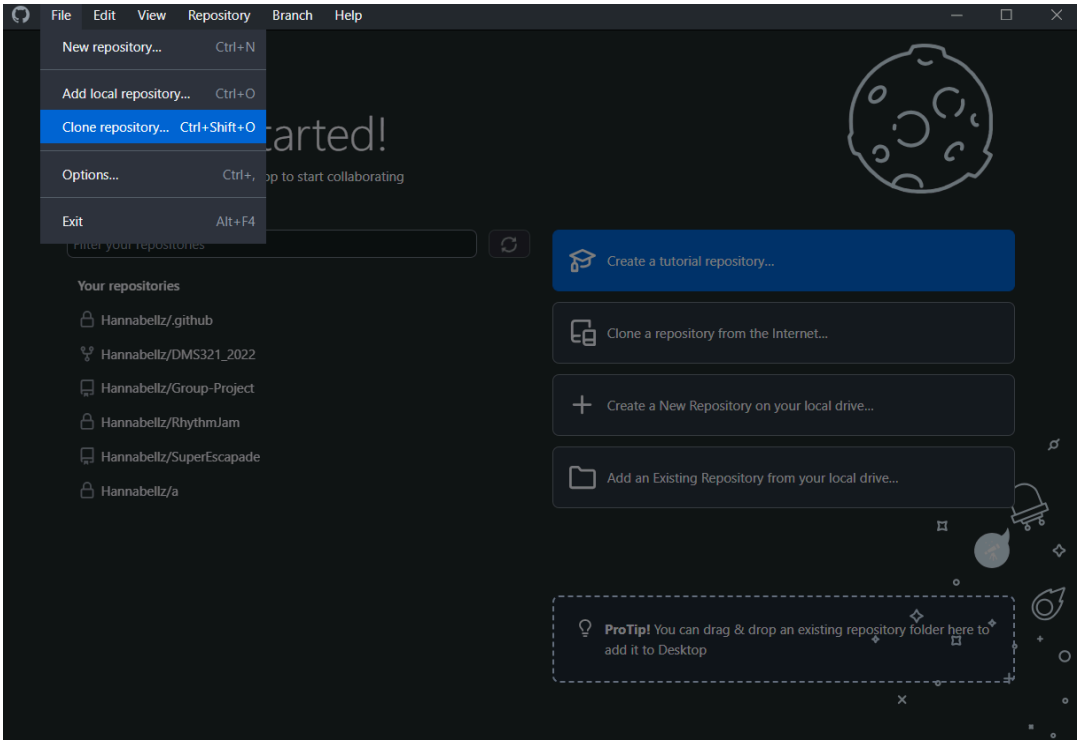
Setting up GitHub Desktop

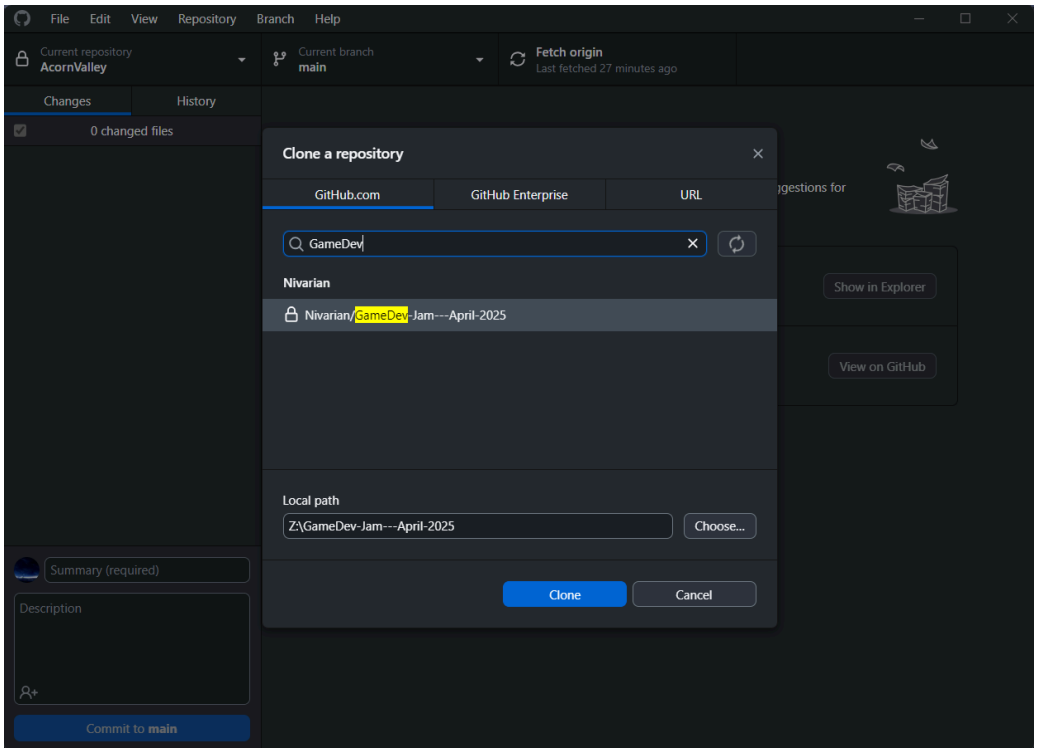
1) Install git from here https://git-scm.com/downloads.
2) If you aren't comfortable with Command Prompt/Terminal stuff, download GitHub Desktop from here https://desktop.github.com/download/.
3) If it's freshly downloaded and once you're logged in, GitHub Desktop may look something like this. You'll want to click on the Clone a repository from the Internet…



   a) Otherwise, you can go to File to click on the Clone repository option there.

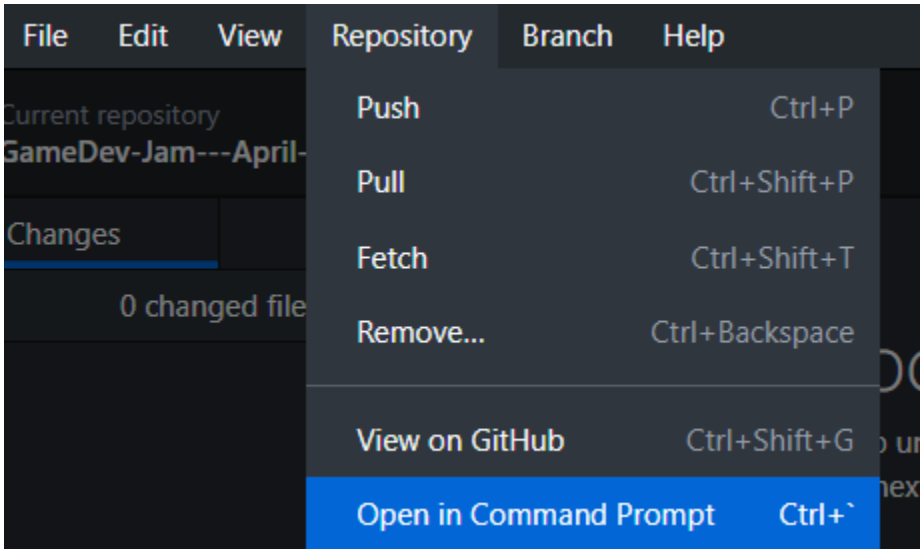


4) Search up the repository you wish to clone, and select where you want it to be on your computer. Then you're good to click Clone!

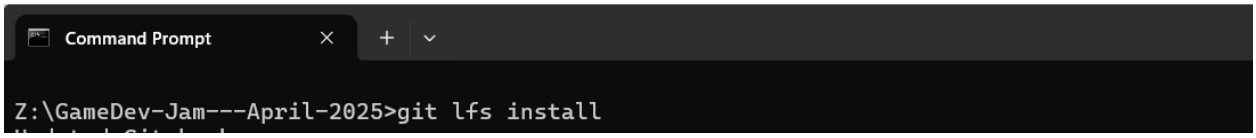


It is perfectly normal if it takes a bit! Bigger projects have a tendency to do so on that initial clone because you're trying to get all of the files that are included.

5) Once it's set, head to Repository and select Open in Command Prompt



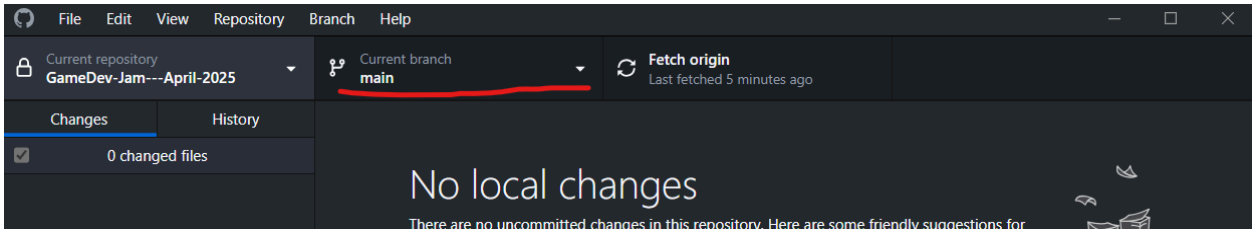
6) Type in git lfs install and hit enter



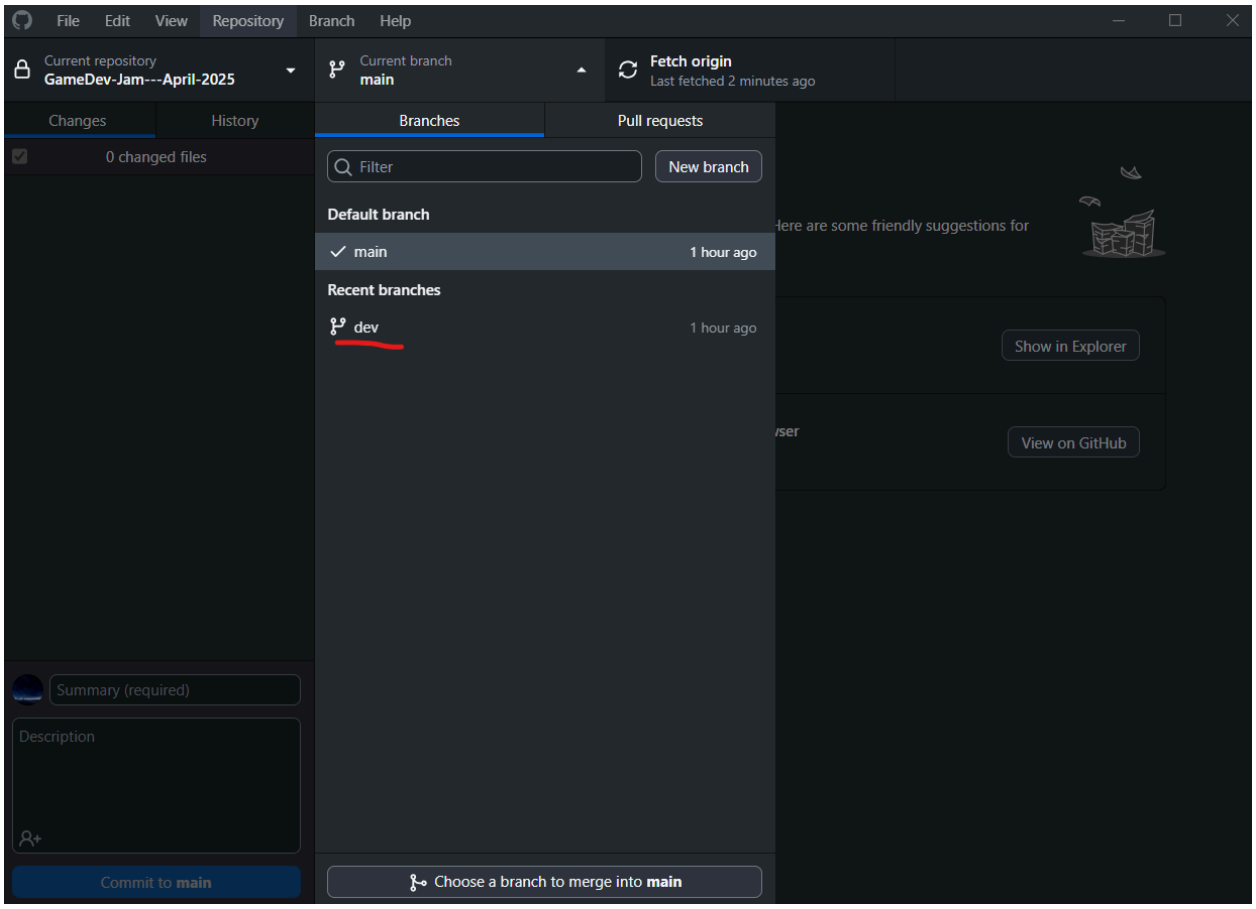
Best Practices (GitHub)

So, you've got the Repository cloned from. You want to get started on some work in-engine– what should you do?

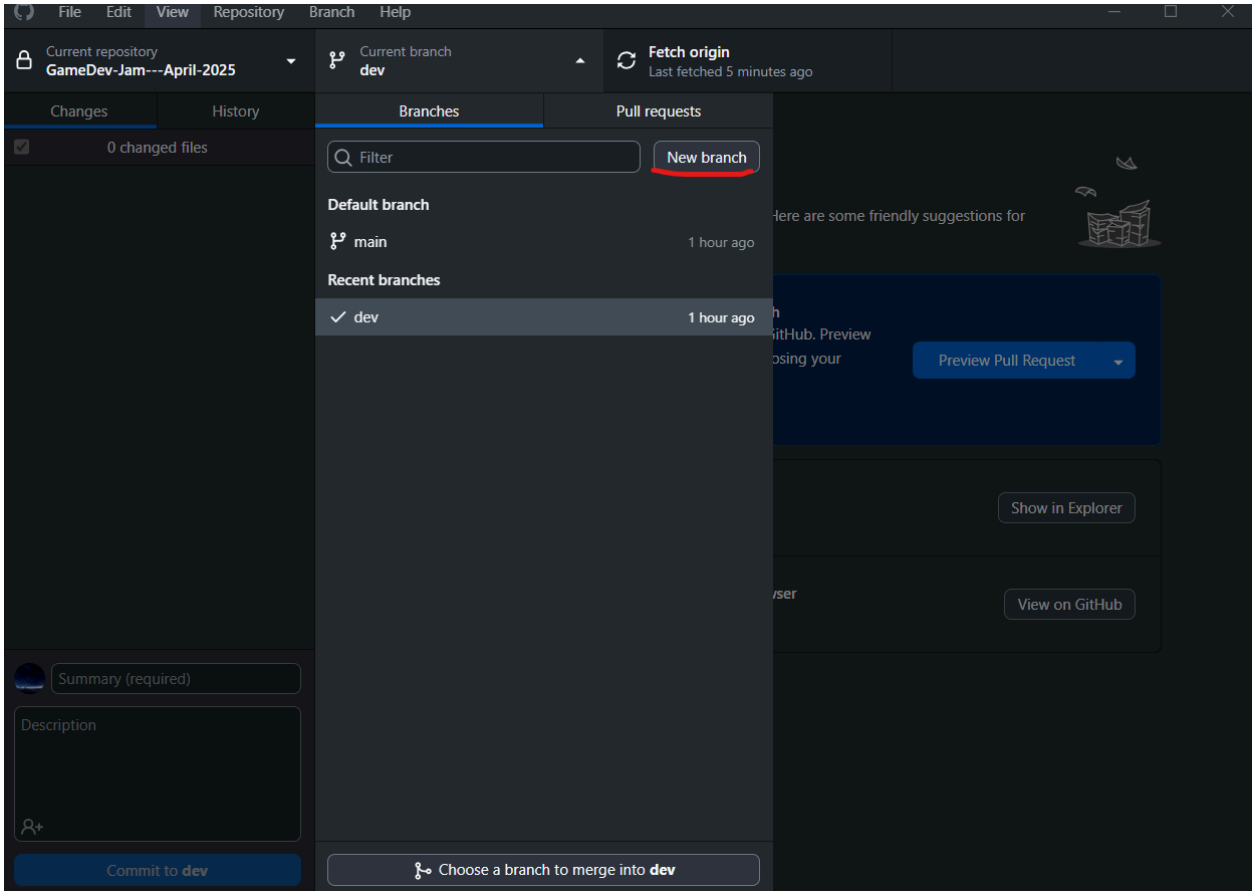
Initially, you'll be in the main branch. You DO NOT EVER want to push your changes to here!



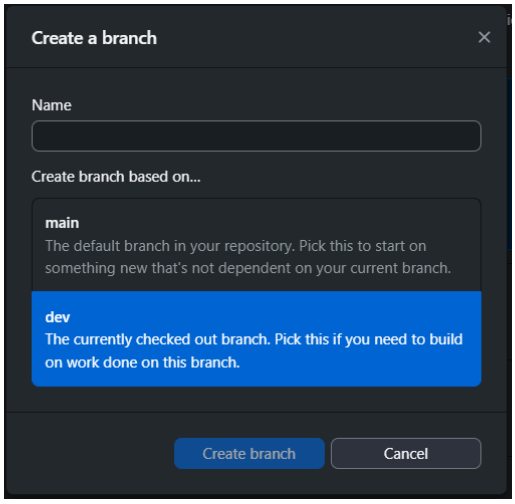
To avoid some major problems, the first thing you want to do before even opening the project is creating your branch. You will want to click on the Current branch, and *switch* to the dev branch. (If it's your first time doing so, it may display as origin/dev)



I'll touch more on the *why* later, but this will represent the project at its most up-to-date, functioning version. When you create a new branch, it uses the one you are currently on, so as a premise you want to be in dev before hitting "New branch."



In the window that pops up, make sure that dev is still selected (this option didn't pop up when I was in main)

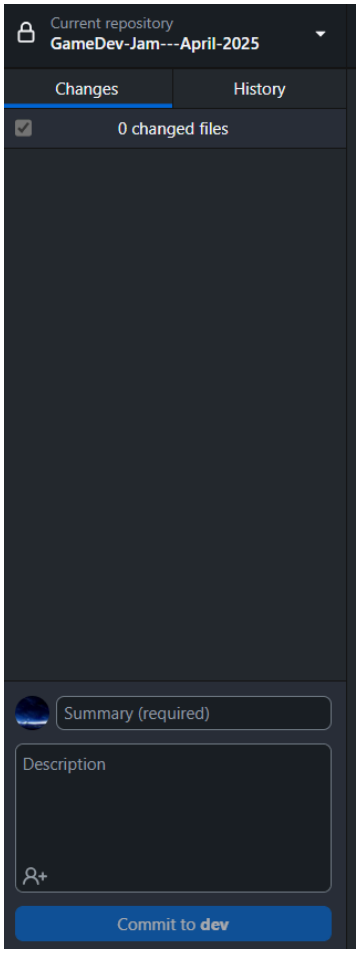


The main thing I will ask for branch naming conventions is that you put a description of what you're working on followed by your first name. So, if I was working on enemy behavior, I would name my branch: "EnemyBehavior_Hannah". Something that is easy to read, and still pretty succinct so there can be an idea at a glance.

And once you're in that branch, you are good to open up the Unreal Project and start cooking! Make sure that you follow naming and file conventions as described in the next section and be sure to stop back in here when you're ready to push your changes!

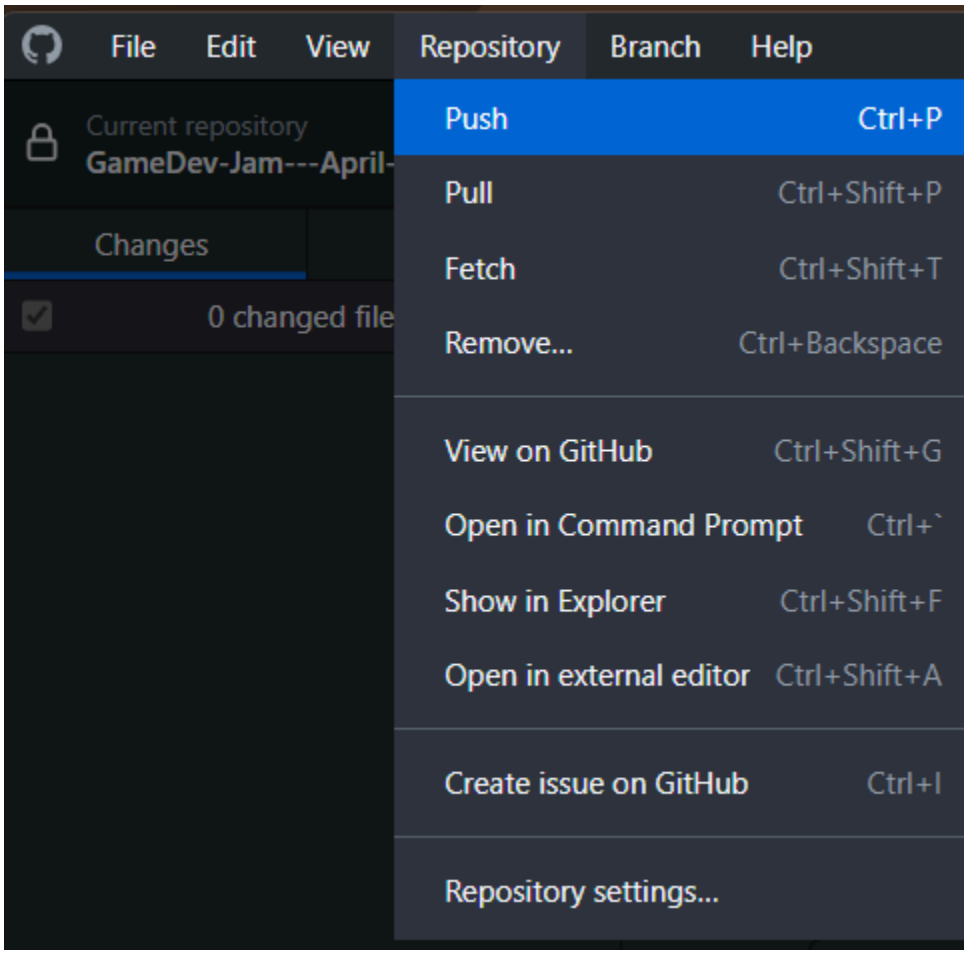
Pushing and Merging – What *you* need to worry about

You have your files all saved. You closed out of Unreal, and see in GitHub Desktop that the files you made changes to/added/removed are listed out to the left.



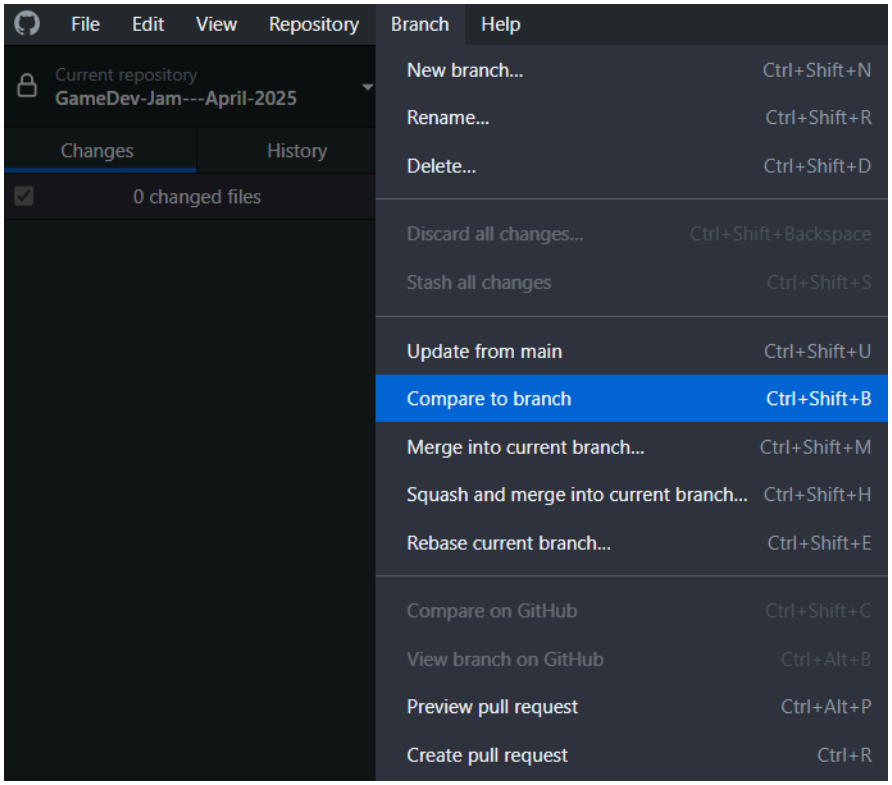
Toward the bottom where it says “Summary (required)” (it may give an example commit message for you) you will have to type a short explanation of what you did. If you want to write more in-depth, that will be in the description, but the summary is best kept short. So, if I were to be working on Enemy Behavior and figured out its regular movement and chasing, I may write something like: “enemy roam and chase functionality.” Upon typing something into the summary, the blue commit button will be clickable. That’s you committing your changes locally to your system, but they aren’t pullable for other people yet.

One way you push your committed changes is going to Repository and selecting the first option. And with that, you should be able to see it from GitHub in your browser! I suggest pushing your changes often throughout the day.

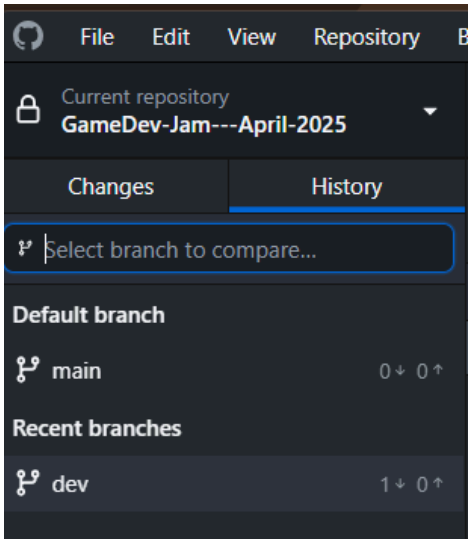


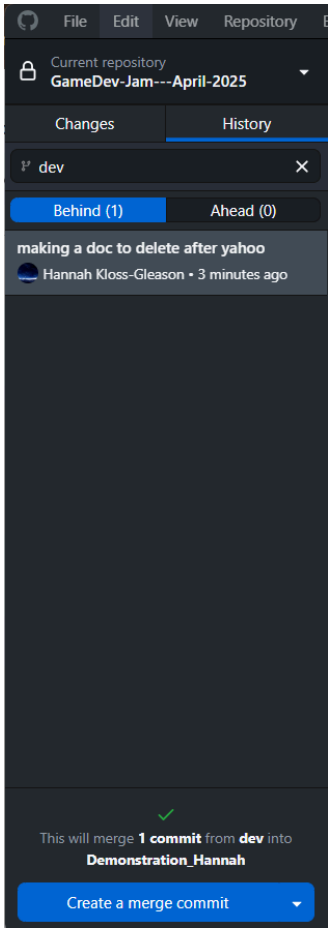
However, if you are done with a specific task, there will be extra steps for you to take. Remember when I said dev is important– here is the reason why.

First, go to Branch, then Compare to branch.



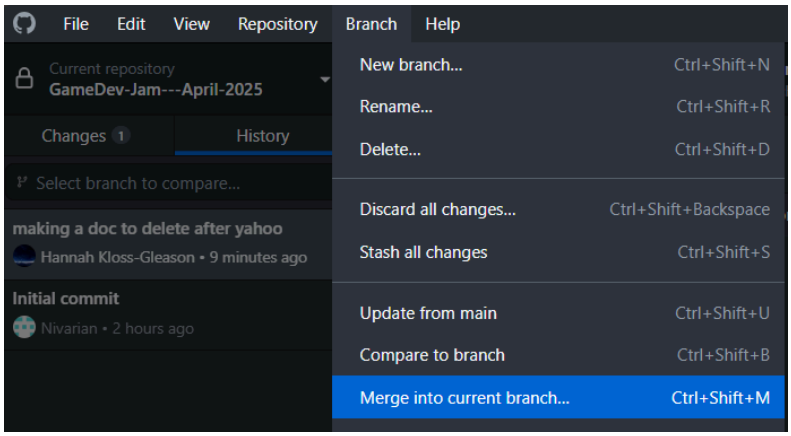
This will bring up the following, where you'll want to select dev





Select “Create a merge commit” so that your branch is up to date with dev. (As a side note, you can do this part as often as you’d like whenever you see changes being put to dev). Make sure the commit gets pushed!

Now, you can switch to dev to be your current branch. Then go to Branch > Merge into current branch...



Select your branch from the pop-up, choose to create a merge commit, and then make sure it gets pushed!

You should never be pushing straight to dev. Always make sure you are in the branch you mean to be prior to opening Unreal and starting to work!

Let me know (either through pinging me in Discord or DMing me) when you have merged something into dev. (This should be after testing it on your side!) Add in a small message detailing what you've completed for it/what I should be expecting to see, and I will take a look and verify it to merge it into main. You will not be touching the main branch.

Best Practices (Unreal)

Do not do not do NOT make changes to a file that is not your own unless you specifically have asked the person who originally made it if they or anyone else are making changes to it. As lovely as I find GitHub to be– if two people try to work on the same blueprint at the same time, it can be disastrous. Ideally, especially within the first few days, things should be able to be worked on independently before we try to bring it all together. Whenever you go to commit and push on GitHub Desktop or whatever you use, make sure that the changed files *only* list out the ones you changed. Discard the changes of the rest.

With that being said, some important things for you to know:

1) Save and save *often*. Unreal can be a bit finicky sometimes, and there's nothing worse than when you go to test your code by running a level and the engine crashes instead. Do yourself a favor. Save your progress regularly. TEST your progress regularly.
    a) Make sure that things are doing what you expect them to, piece by piece. It will make it easier if you keep on building onto one thing, if something breaks, you'll be able to pinpoint the *where* easier. Breakpoints can also be handy for this!
    b) There is a folder for you to create your own personal level to test your things in! Create a level in the _TO_PLACE_TESTING_LEVELS, so that way it's also easier to delete all these extra levels for the final build.
2) Take a look at the Recommended Asset Naming Conventions for creating different things– but some pretty common ones:

For	**Prefix**	**Example**
Blueprints	BP_	BP_PlayerCharacter_YourFirstName
Widget Blueprints	WBP_	WBP_MainMenu_YourFirstName

Enum	E_	E_Evidence_YourFirstName
Struct	F_	F_Dialogue_YourFirstName
Data Table	DT_	DT_Narration_YourFirstName

3) Prior to people branching off dev, I will make sure that some initial folders are in place. The primary ones I currently have planned (more can be added as necessary) are: Blueprints (with subfolders for Player, Enemy, and Environment) and UI. If there are any more that you can foresee being useful from the get-go, please either ping me or DM me.
4) Make sure that Unreal is closed when you are pulling/merging changes from dev into your branch. Things are probably not going to properly update, otherwise, and I'd rather be safe than sorry.