



# USING GITHUB WITH UNREAL ENGINE

BY: REBECCA ALOISIO

# CONTEXT OF THIS PRESENTATION

- This presentation was created using:
  - Git version: 2.30.0
  - Git LFS version: 2.13.2
  - GitHub Desktop version: 2.6.3
  - Unreal Engine: 4.25.4
- It was also created on 2/12/2021
- Elements of this presentation may carry over to other versions of the software, but there are no guarantees.
- Any prices and/or data plans mentioned were valid on the date of creating this presentation.

The background is a dark blue gradient. In the corners, there are white line-art illustrations of circuit boards or neural networks, consisting of lines and small circles.

# NEEDED SOFTWARE AND TOOLS

# GITHUB ACCOUNT

- Seems obvious, but make sure you have one.
- If you are working in a group, make sure you get your teammates usernames. You'll need them later.
- Go to: <https://github.com/>

# GIT

- Git is version control system.
- Useful for:
  - backing up your projects
  - Collaboration
- Download at: <https://git-scm.com/>
- Website should look like this->
- Circled is the download button.

**git** --distributed-is-the-new-centralized

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient staging areas, and **multiple workflows**.

**About**  
The advantages of Git compared to other source control systems.

**Documentation**  
Command reference pages, Pro Git book content, videos and other material.

**Downloads**  
GUI clients and binary releases for all major platforms.

**Community**  
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release  
**2.30.0**  
Release Notes (2020-12-27)  
**Download 2.30.0 for Windows**

Pro Git by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Windows GUIs Tarballs  
Mac Build Source Code

# GIT LFS

- Short for “Git Large File Storage”
- Needed for files that are too large for standard GitHub
  - GitHub’s limit is 100 MB for a file
- Download at: <https://git-lfs.github.com/>
- Website should look like this->

The screenshot shows the Git Large File Storage (LFS) website homepage. At the top, there is a navigation bar with links for "Docs", "Downloads", and "Source". The main heading is "Git Large File Storage". Below this, there is a sub-heading: "An open source Git extension for versioning large files". A paragraph explains that Git LFS replaces large files (audio, videos, datasets, graphics) with text pointers inside Git, storing the actual file contents on a remote server like GitHub.com or GitHub Enterprise. A prominent blue button labeled "Download v2.13.2 (Windows)" is visible. To the right, a diagram illustrates the workflow: a "Local" repository (represented by a computer icon) contains "code" files and a "file.psd". The "code" files are pushed to a "Remote" repository (represented by a server icon). The "file.psd" is pushed to a "Large File Storage" service (represented by a database icon). A security update notice states: "Git LFS security update: Windows users should update to 2.13.2 or newer". The page is divided into two columns: "Getting Started" and "Features". The "Getting Started" section includes a numbered list: 1. Download and install the Git command line extension. Once downloaded and installed, set up Git LFS for your user account by running: `git lfs install`. Below this, it says "You only need to run this once per user account." 2. In each Git repository where you want to use Git LFS, select the "Features" section, which includes: "Large file versioning" (Version large files—even those as large as a couple GB in size—with Git.) and "More repository space" (Host more in your Git repositories. External file storage makes it easy to keep your repository at a manageable size.)

# GIT LFS CONSIDERATIONS

- You only get 1 GB of Git LFS storage and bandwidth for free with GitHub.
- After that, it costs \$5 / month for every 50 GB of storage and bandwidth you need.
  - You can purchase this under “Billing & plans” in your GitHub Account settings.



# WHAT ARE GIT LFS STORAGE AND BANDWIDTH AND HOW IS USE DETERMINED?

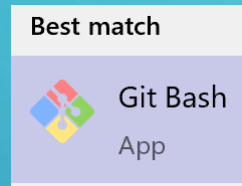
- **Storage** is used whenever a file is uploaded to the repository.
- **Bandwidth** is used whenever a file is downloaded.
- The amount used is based on the files size.
  - i.e. a file that is 500 MB will use that amount of your allotted storage and/or bandwidth.
- This applies for any small change, so USE WISELY!



# INITIALIZE GIT LFS

- Using Git Bash (Installed with Git):

- Open Git Bash



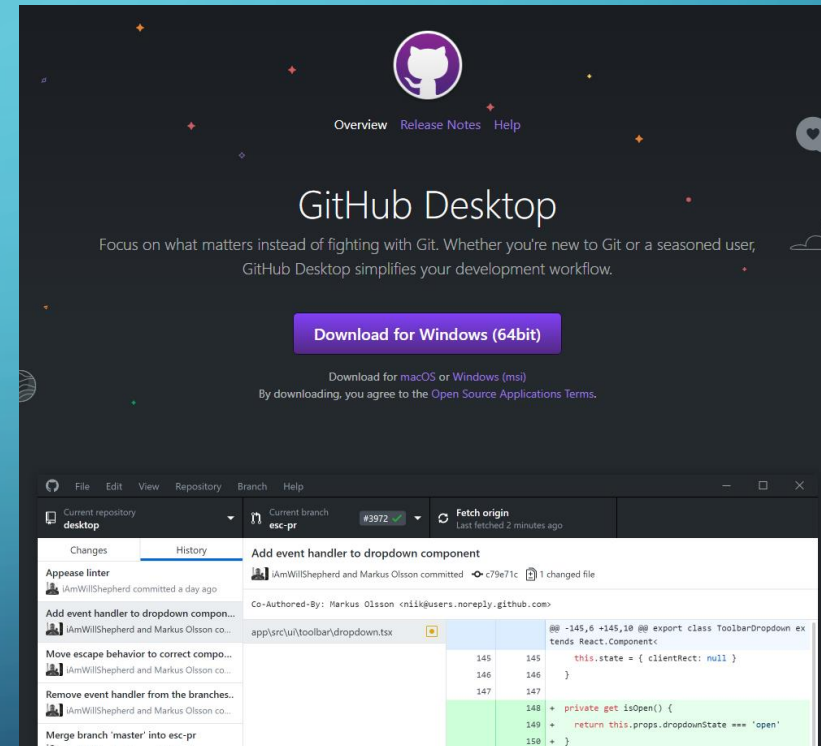
- Type: git lfs install

```
MINGW64:/c/Users/becky  
  
becky@Rebeccas-Surface-Book-2 MINGW64 ~  
$ git lfs install  
git LFS initialized.  
  
becky@Rebeccas-Surface-Book-2 MINGW64 ~  
$ |
```

- Note: If opening GitHub Desktop it may ask if you want to initialize Git LFS. Accept that as well.

# GITHUB DESKTOP (OPTIONAL, BUT RECOMMENDED)

- If you don't want to mess with console commands, use this.
- Gives a GUI for git and quickly connecting projects with GitHub.
- Download at:  
<https://desktop.github.com/>
- Website should look like this->

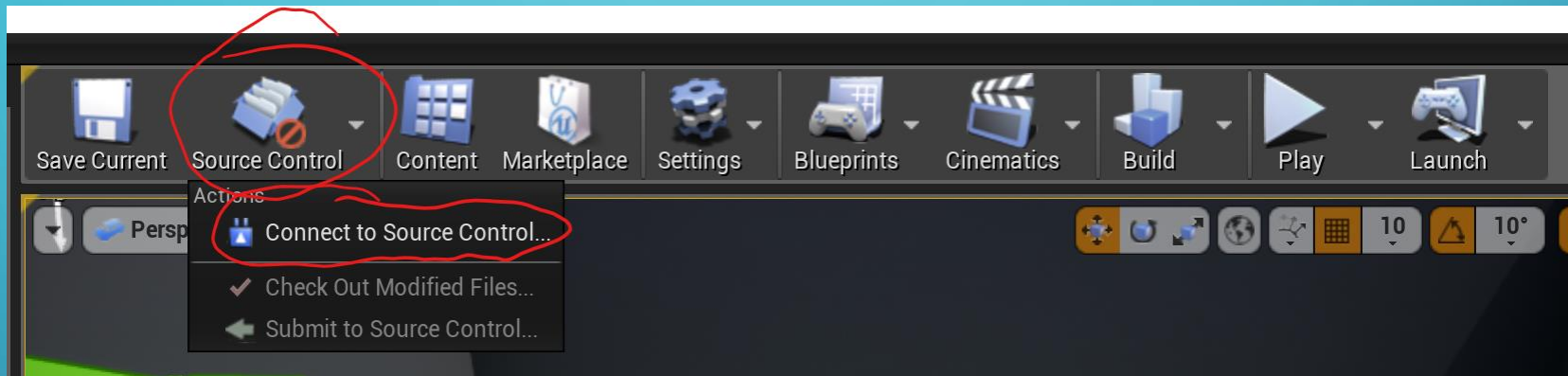




# CONNECTING AN UNREAL PROJECT TO SOURCE CONTROL

# FIRST TIME SET UP – PART 1

- For any project, fresh or established, open the project and:



# FIRST TIME SET UP – PART 2

- Make sure this is clicked.

- Uncheck this.

Source Control Login

Provider: Git (beta version)

Git Path: C:\Program Files\Git\bin\git.exe

Root of the repository: C:/Users/becky/Documents/Unreal Projects/MyProject/

User Name: Rebecca Aloisio

E-Mail: beckyaloisio@gmail.com

Current Project is not contained in a Git Repository. Fill the form below to initialize a new Repository.

URL of the remote server 'origin':

Add a .gitignore file

Add a basic README.md file

Add a .gitattributes file to enable Git LFS

Make the initial Git Commit

# MyProject

Developed with Unreal Engine 4

Initial commit

Initialize project with Git

Source Control Log

Accept Settings Run Without Source Control

- If this is not filled, you need to navigate to your installation of git.

- Click this
- Then this when done.

# FIRST TIME SET UP – PART 3

- The “.gitattributes” file contains the settings for Git LFS for a particular project.
- If you want the simplest set-up, track the files in the text editor example.
- There are two ways to modify the file:
- Edit .gitattributes with a text editor.

```
.gitattributes - Notepad
File Edit Format View Help
# UE file types
*.uasset filter=lfs diff=lfs merge=lfs -text
*.umap filter=lfs diff=lfs merge=lfs -text

# Raw Content types
*.fbx filter=lfs diff=lfs merge=lfs -text
*.3ds filter=lfs diff=lfs merge=lfs -text
*.psd filter=lfs diff=lfs merge=lfs -text
*.png filter=lfs diff=lfs merge=lfs -text
*.mp3 filter=lfs diff=lfs merge=lfs -text
*.wav filter=lfs diff=lfs merge=lfs -text
*.xcf filter=lfs diff=lfs merge=lfs -text
*.jpg filter=lfs diff=lfs merge=lfs -text
```

- Using console commands:

In each Git repository where you want to use Git LFS, select the file types you'd like Git LFS to manage (or directly edit your .gitattributes). You can configure additional file extensions at anytime.

```
git lfs track "*.psd"
```

Or any  
file type

Now make sure .gitattributes is tracked:

```
git add .gitattributes
```



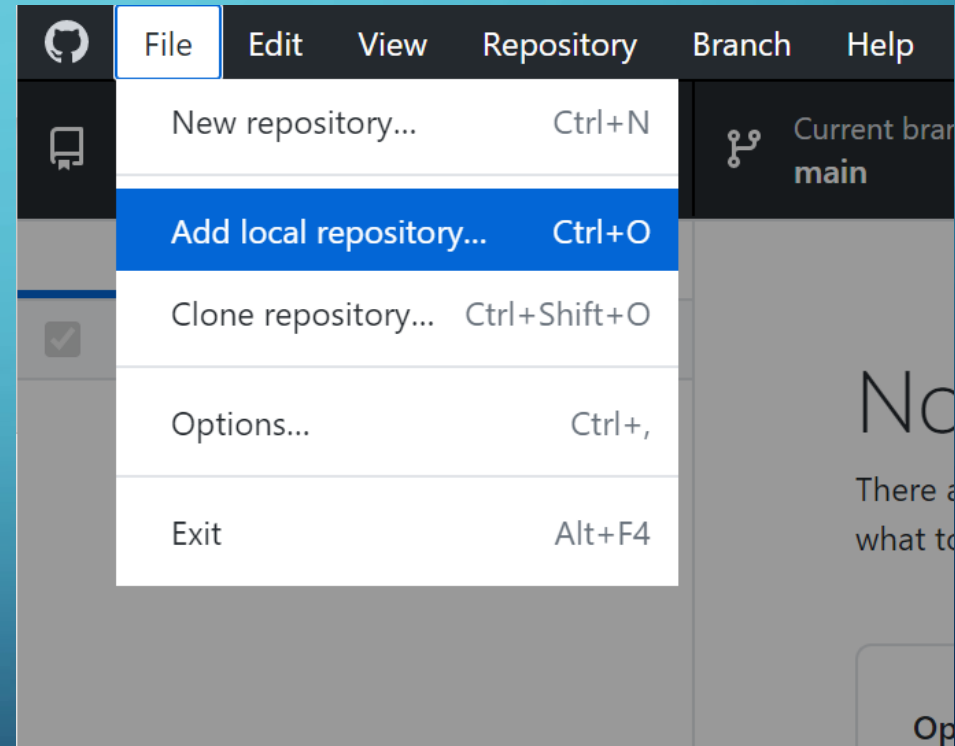
# WHY MODIFY?

- By default, Unreal will set the project to add everything in “Content” to Git LFS.
- That means unnecessary files are using precious Git LFS space.
  - For greater control of tracking (and storage use), set it to track individual files rather than files of a specific type.



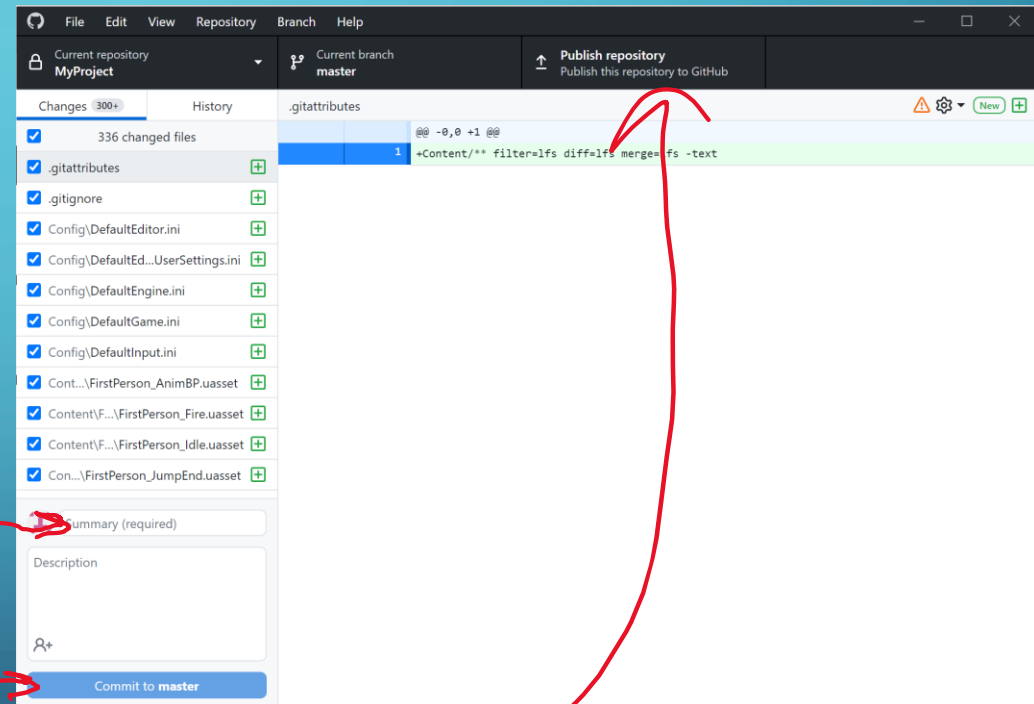
# FIRST TIME SET UP – PART 4

- Save and close out of the project.
- Open GitHub Desktop
- Click “File->Add a local repository”
- Navigate to the game project folder



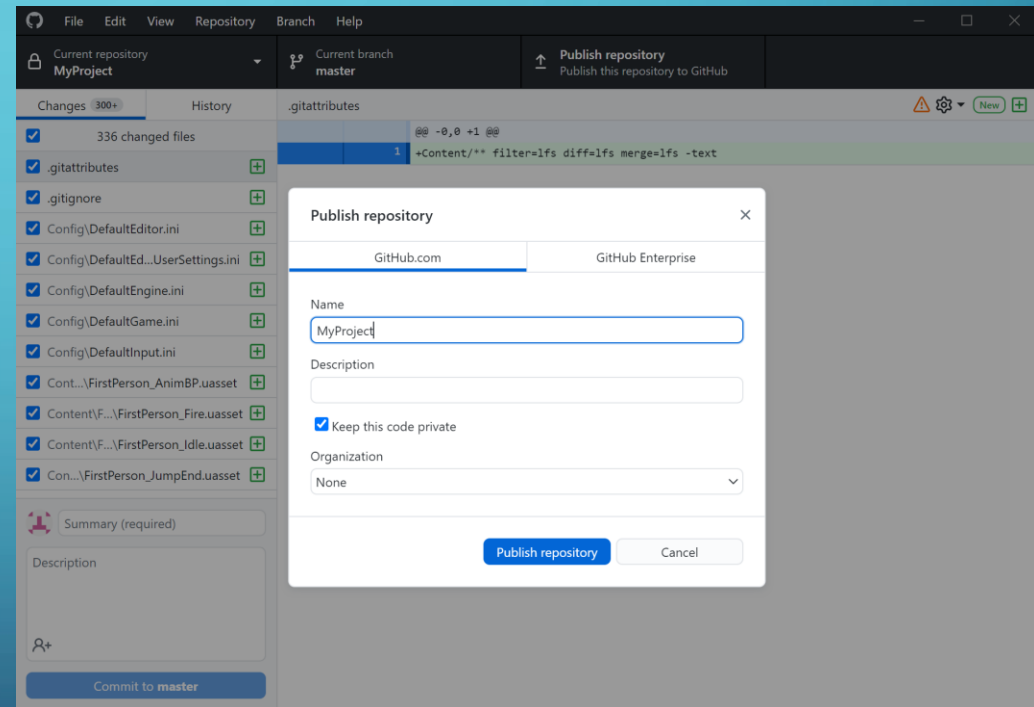
# FIRST TIME SET UP – PART 5

- Uploading your project to GitHub.
- You should see something like the side image.
- Traditionally, the first commit is called “Initial commit” so type that here
- Then click the “Commit to...” button.
- Then click the “Publish repository” button.



# FIRST TIME SET UP – PART 6

- You'll get this pop-up.
- Leave the “Name” section alone.
- “Description” is what will show up in the “About” page of the repository.
- “Keep this code private” will determine if you have a public or private repository.
  - Look at GitHub’s policy on private repositories before using them.
- Click “Publish repository”



# FIRST TIME SET UP - FINAL

- You're done!
- Once done uploading, go to <https://github.com/> to check that it uploaded okay.

The screenshot shows a GitHub repository page for 'skeletora / MyProject' (Private). The repository is on the 'master' branch and has 1 commit. The commit history shows an initial commit by 'skeletora' 5 minutes ago, with a commit hash of 'e4183b8'. The commit includes files: Config, Content, .gitattributes, .gitignore, MyProject.uproject, and README.md. The README.md file is expanded, showing the repository name 'MyProject' and the text 'Developed with Unreal Engine 4'. The right sidebar contains sections for 'About' (No description, website, or topics provided), 'Releases' (No releases published, Create a new release), and 'Packages' (No packages published, Publish your first package).

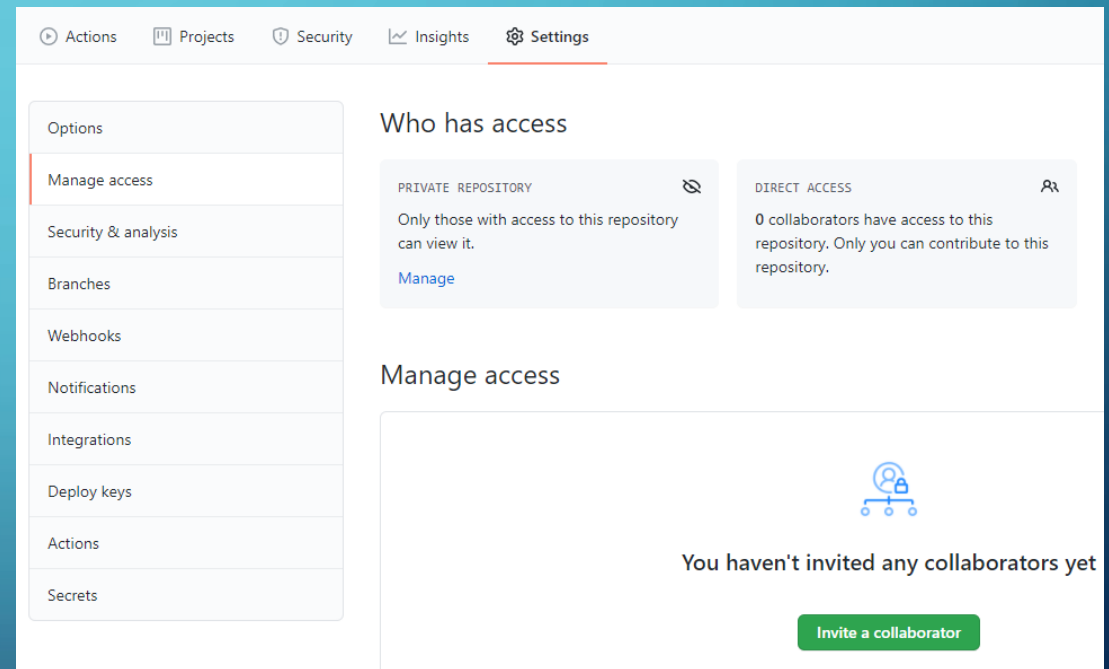
File	Commit	Time
Config	Initial commit	5 minutes ago
Content	Initial commit	5 minutes ago
.gitattributes	Initial commit	5 minutes ago
.gitignore	Initial commit	5 minutes ago
MyProject.uproject	Initial commit	5 minutes ago
README.md	Initial commit	5 minutes ago

The background is a dark blue gradient. In the corners, there are white line-art graphics resembling circuit traces or data paths, with small circles at the end of the lines. These graphics are located in the top-left, top-right, bottom-left, and bottom-right corners.

OTHERS ACCESSING THE PROJECT

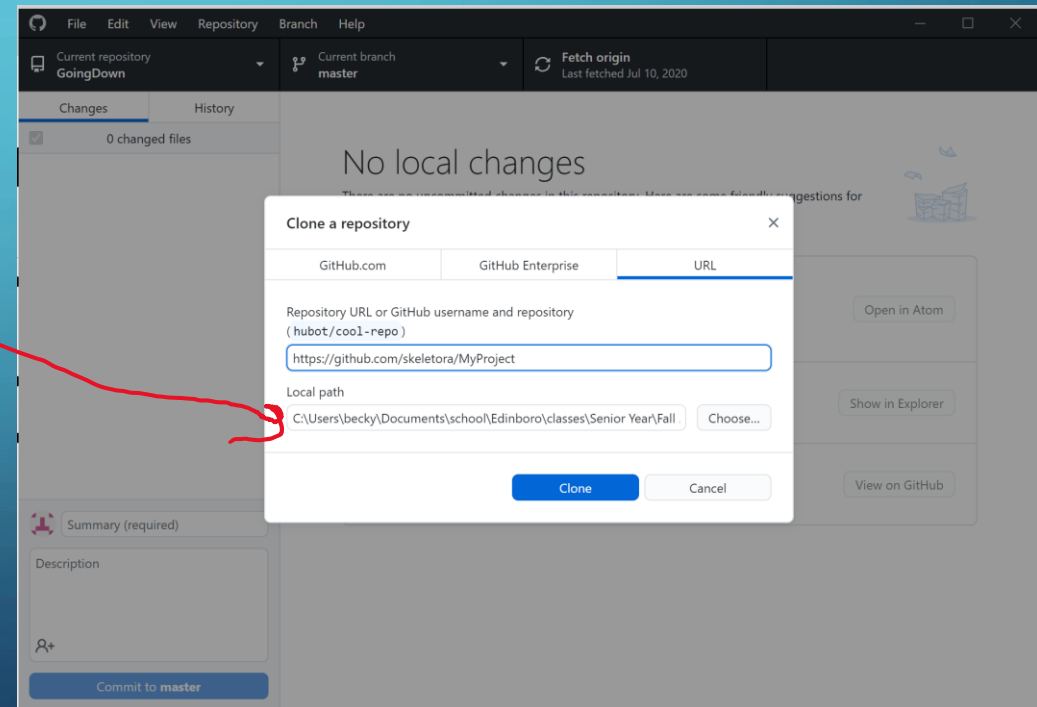
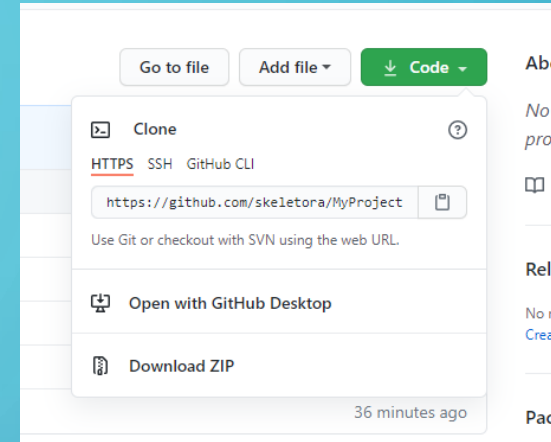
# ADD TEAMMATES AS COLLABORATORS

- Only collaborators can modify a project.
- Open the project repository.
- Go to “Settings”
- Click “Manage access”
- Click “Invite a collaborator”
- Enter their username.
- They will receive an email invite.



# DOWNLOADING THE PROJECT

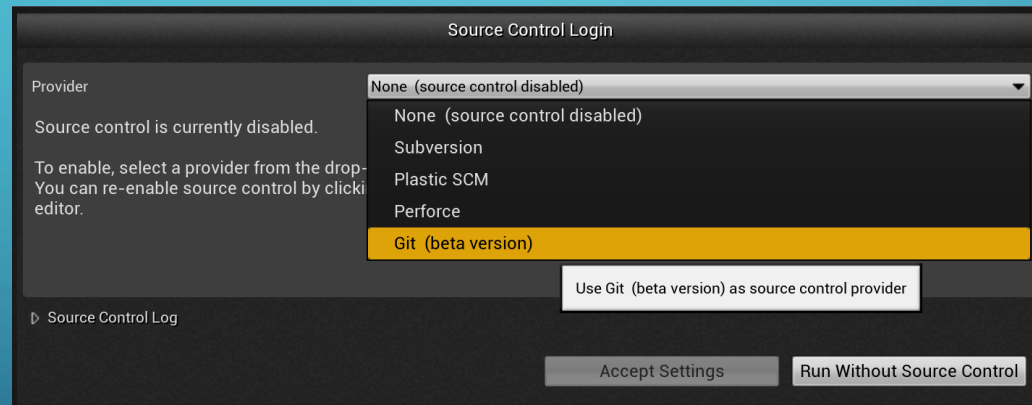
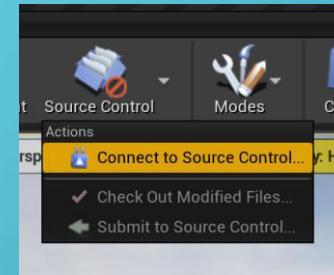
- Navigate to the project repository on GitHub.
- Click the green “Code” button.
- Click “Open with GitHub Desktop”
  - Give the browser permission.
- Set the file location to download the project to here
  - It will create a project folder there.
- Click “Clone”
- Agree to initialize Git LFS.



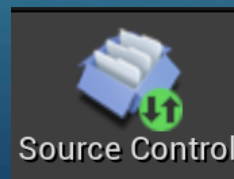


# CONNECT TO SOURCE CONTROL

- Open the downloaded project in Unreal Engine.
- Click “Source Control->Connect to Source Control...”
- Select “Git (beta version)” under “Provider”



- Make sure it has detected your git installation (just like in the set-up section) and click “Accept Settings”





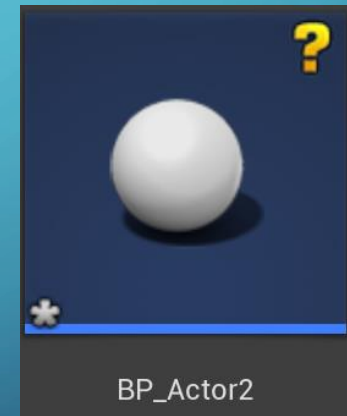
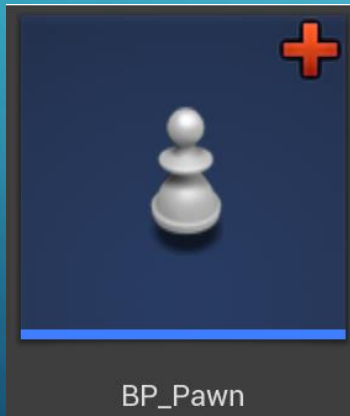
WHAT NOW?

# SHARING CHANGES

- Project modifications on your machine are not automatically shared with team.
- When ready to upload to GitHub, you have two options:
  - Unreal Engine
  - GitHub Desktop
- I suggest using the Unreal Engine method if the project is still open.

# SHARING CHANGES – UNREAL ENGINE PT. 1

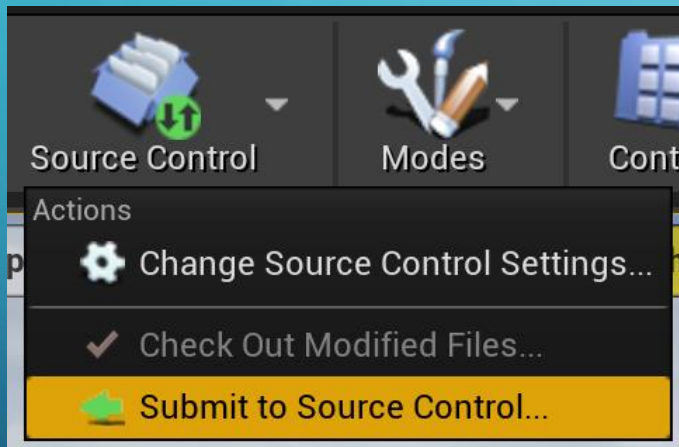
- Changed items should have one of the following symbols in the top right corner:
- If they don't, they won't be uploaded. They might have this:



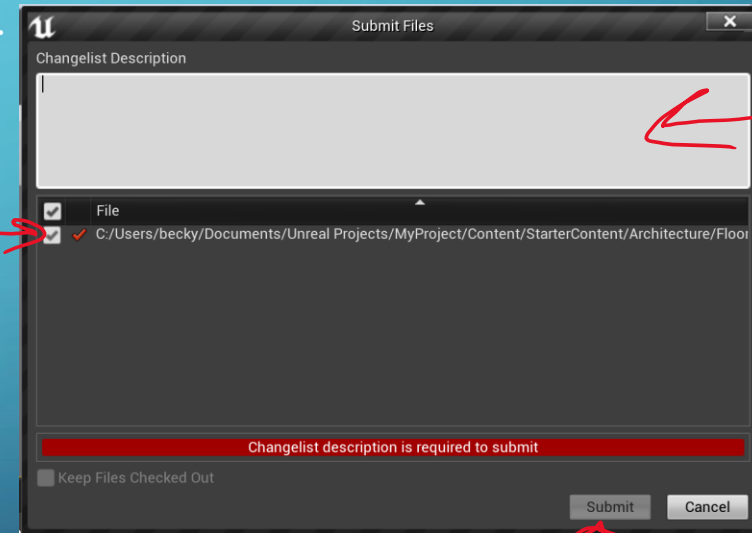
- If that is the case, then save it.

# SHARING CHANGES – UNREAL ENGINE PT. 2

- Click “Source Control->Submit to Source Control...”



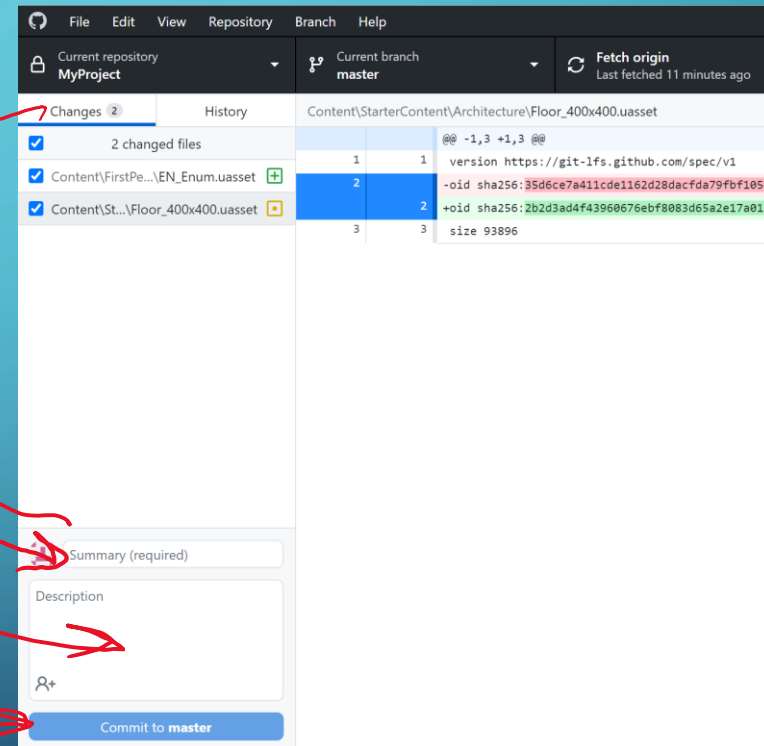
- Enter a description for the commit in “Changelist Description”
- Check the files included. Uncheck if you don’t want to include in commit.



- Click “Submit” when done.

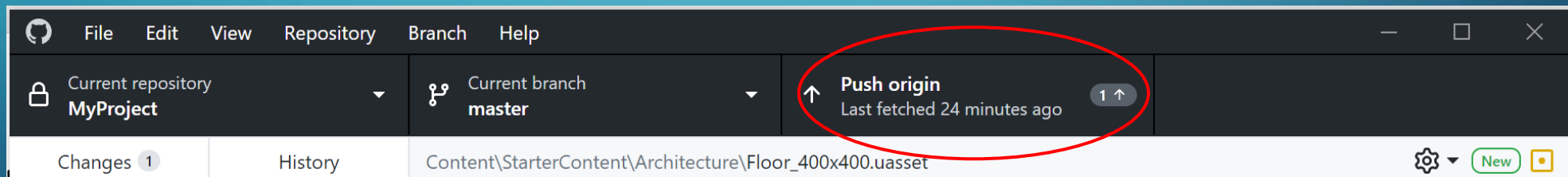
# SHARING CHANGES – GITHUB DESKTOP

- Everything that has been changed since the last commit is listed here
  - Make sure only the things you want to include are checked.
- Type a short descriptor for the commit here
- You can give a more detailed description here
- Click “Commit to...” when done.



# UPLOADING THOSE CHANGES

- You need to upload the changes to GitHub.
- Open GitHub Desktop (previous method doesn't matter)
- Click “Push to Origin”





# CHECKING FOR & DOWNLOADING UPDATES

- To check for changes, open GitHub Desktop.
- Click “Fetch origin”



**Fetch origin**

Last fetched 6 minutes ago

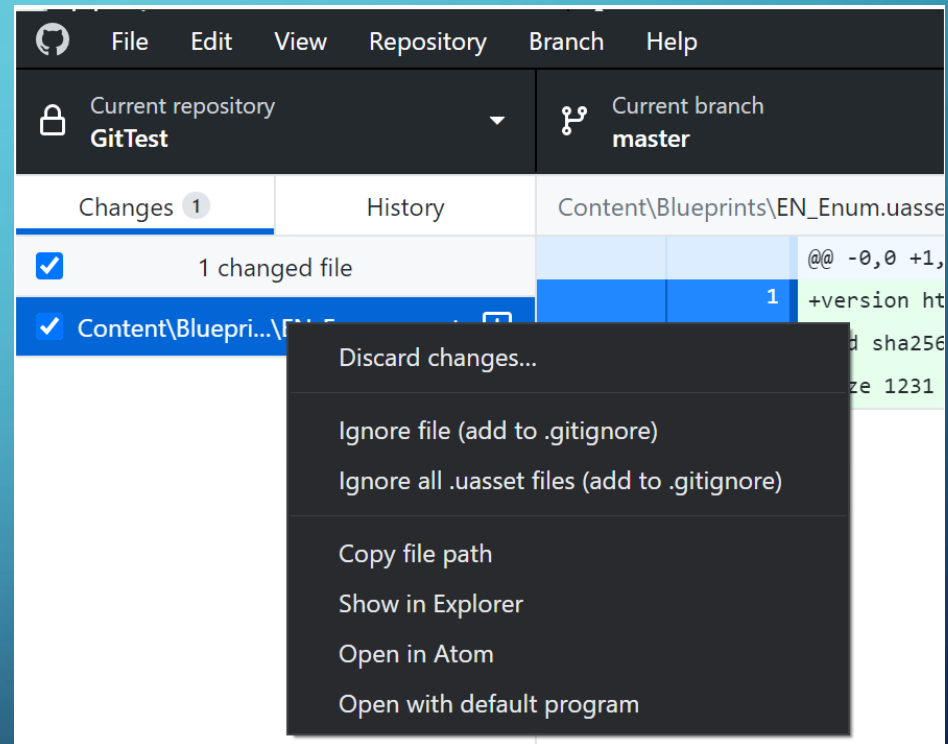
- If there is a change, the button will change to say “Pull origin”
- Click the button to download it.

# A WORD OF CAUTION

- Carefully delegate work.
  - If two people are modifying the same portion of the project, you may run into a conflict that can't be merged. Someone will have to try and resolve that before a merge can go through.
- Levels are a pain
  - Levels have the greatest tendency to show up in your changes (and therefore commits) when all you may have done is move the camera. Be careful to not include them if you didn't want them or you may run into a conflict.

# A FIX THAT SOMETIMES WORKS

- If changes are popping up that you didn't do, there is a way to discard them.
- CLOSE OUT OF THE PROJECT.
- Open GitHub Desktop
- Right-click on the offending object
- Click "Discard changes..."
- This can prevent future conflicts.



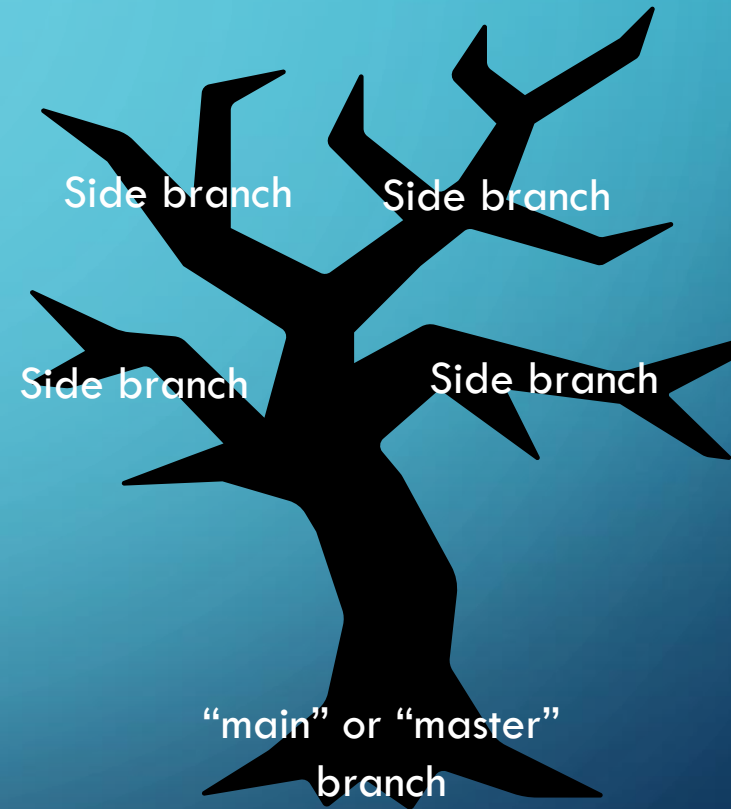
The background is a dark blue gradient. In the corners, there are decorative white line-art patterns resembling circuit traces or data paths, with small circles at the end of the lines.

# ADVANCED TOPICS

PRIMARILY RELATING TO GIT AND GITHUB

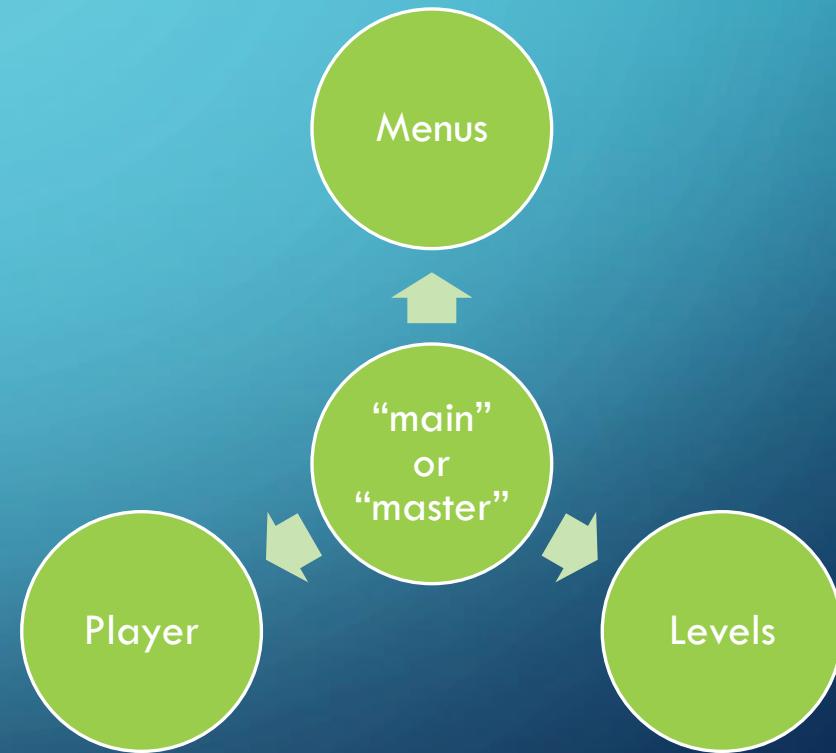
# BRANCHES – PT. 1 – WHAT ARE THEY?

- Branches are a way divide a project up to safely test and implement features.
- You have your “main” or “master” branch.
  - This is the tree trunk of the project that all other branches shoot off from.
- Then you have custom side branches.



# BRANCHES – PT. 2 – HOW THEY CONNECT

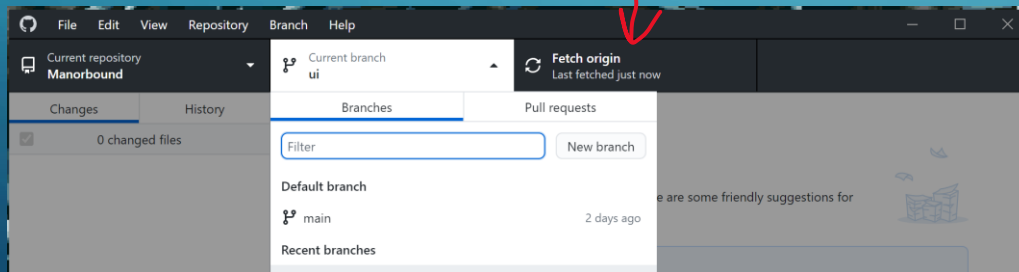
- Functionally, branches are like separate copies of your project.
- Creating a branch from another branch (i.e. “main”) makes the new branch a copy of the previous.
- Branches cannot access anything exclusive to another branch.
  - i.e. branch “menus” has the main menu. The branch “player” doesn’t so it cannot access the main menu.



# BRANCHES – PT. 3 – MAKING A BRANCH

- GitHub Desktop

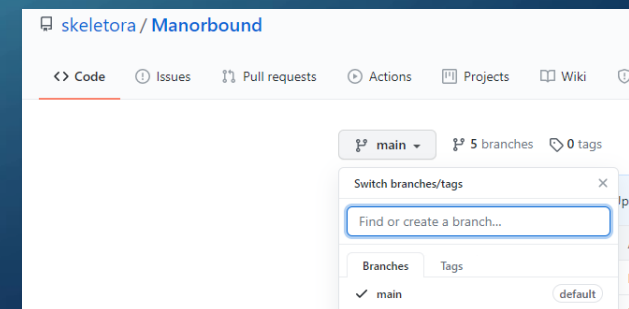
- Click “Current branch”
- Type the name in “Filter”
- Click “New branch”
- Click “Publish branch” (it will change)



- GitHub

- Under “Code” click “main” or “master”
  - Note: this may have a different branch name. Make sure it says “main” or “master”
- Type the name of the branch
- It should now be listed on GitHub Desktop

## Desktop





# BRANCHES – PT. 4 – THEIR USE

- The “main” or “master” branch should be treated as the current working version of the project.
- All other branches are used for implementing and testing new features before adding them to “main” or “master.”
  - Ex. Testing out a double jump for the player class before adding it to the “main” or “master” branch.
- What do you do when you are done implementing and testing on a branch? Afterall, branches can't access each other.
  - You submit a “pull request.”
- This can be done with GitHub or GitHub Desktop.

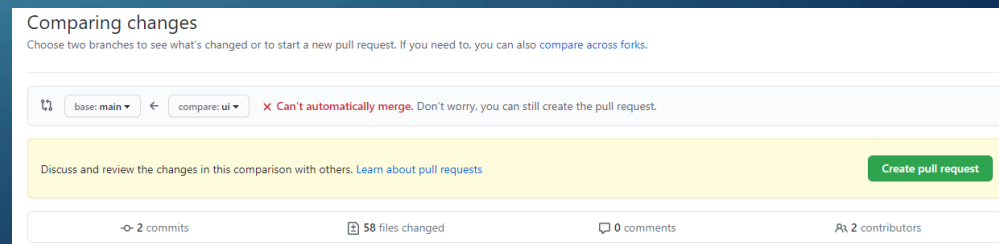
# BRANCHES – FINAL - MERGING

- GitHub Desktop

- Switch to the branch
  - Click “Current branch” and select the branch
- Click “Create Pull Request”
- You’ll be taken to github.com
- Fill out info
- Click “Create pull request”
- Someone will then need to go in and apply the pull request

- GitHub

- Click the “Pull Requests” tab
- Click “New pull request”
- The “compare: <branch>” tab is the branch that is being merged into “base: <branch>”
- You want a green “Able to merge”
  - If you don’t, you have a conflict.
- Click “Create pull request” when done.



The background is a dark blue gradient. In the corners, there are white line-art graphics resembling circuit traces or data paths, with small circles at the end of the lines. These are located in the top-left, top-right, bottom-left, and bottom-right corners.

QUESTIONS?