**GitHub and its Applications in Unity: A Hate Letter to Flash Drives**

**Common Terms**

|  |  |
| --- | --- |
| Term | Description |
| Repo | Repository. A Git repo is a place where your project is stored. Public repos can be viewed by anyone whereas private repos can only be viewed by your account. |
| LFS | Large File Storage. Some files are too large for Git to work with (100mb+). LFS is a system that solves this issue by 1. Uploading the initial large file, 2. Only updating parts of the large file that changed. This keeps git from constantly needing to move the entire large file. |
| Branch | A branch is what it sounds like. If you create a branch from your main branch, you can edit anything you want in that branch without messing up your main branch. The branch can then be merged. |
| Merge | Merging happens when you merge an off-shoot branch back into your main branch. |
| Merge Conflict | A merge conflict happens when code between your main branch and other branch are different. Merge conflicts must be resolved before a merge can be completed. |
| Push | Whenever you make client-side changes to a branch, the branch must be pushed to be committed to the repo in Git’s servers. |
| Commit | After pushing your updates, you must commit them. This finalizes changes making them no longer local to your machine. |
| Pull | Any time an update is committed to a repo, you can pull request, which applies those changes to the version of the repo on your machine. Note: if your machine was the most recent to commit, a pull request is unnecessary. |
| Clone | Before you start work on a repo, you must clone it. This creates a copy of the repo locally on your machine. |
| .gitignore | A file to be placed at the root folder of your repo. This file is useful as it contains a list of file types to be ignored during commits. Many files are build-specific—especially with Unity, and are unnecessary to keep in your repo. This saves space and time needed to commit. |
| README | A README file stores text information about your project. It can be useful to explain the project’s end goal if you are working collaboratively with internet strangers. |

**Creating a repository**

1. Create a GitHub account or sign into your existing one.
2. In the top right, click the “+” by your profile picture, then select “New Repository.”
3. Give your repo a name.

Add a .gitignore file. Search the templates for Unity, or add your own .gitignore later. I will provide a fleshed out Unity .gitignore here: 

2. Congratulations! You have a repo… so what now?

**Adding your Unity project to your new repo**

1. You can view your repo now via github.com, but there won’t be much there. Possibly a README and .gitignore if you chose to add those.
2. Open up GitHub Desktop.
3. In the top left, you’ll see “Current Repository.” Click that, then select “Clone Repository.”
4. If you’re signed in, you’ll see a list of repositories on your account. Select the one you recently created.
5. If you are working with a pre-existing unity project, go ahead and find where it’s stored (this can be done via Unity Hub) in your file browser. Then, click the root folder. Copy it (ctrl+c), then navigate to Documents/GitHub/<repository name>. If you’ve pulled your repository, you’ll see whatever files you saw on github.com. Go ahead and paste your Unity project there.
6. If you are creating a new Unity project, simply change the location the project will be created to Documents/GitHub/<repository name>.
7. Navigate back to Github Desktop. Here, you should see a list of changes made. If you added an existing project, there will be a lot of changes.
8. Congratulations! Your project is now accessible from any machine!

**Unless.. Uh-oh. The files are TOO STRONG.**

It’s no secret that Unity projects can get large. Some assets alone can reach sizes of 100+mb. Git doesn’t like big files unless it’s told how to deal with them. Thankfully, there’s a tool for this called Git LFS. It’s easy to set up and will teach Git how to work with large files. First, you’ll need to understand that Git is a system, GitHub is just a way to view that system. They are two separate things! So, with that in mind, it’s time to install Git Bash.

1. Go here: <https://git-scm.com/downloads> and download Git.
2. Now, navigate to your project folder. Right click, then select “Git Bash Here.”
3. You’ll be shown a terminal window. In that window, type **git lfs install**.
4. This will initialize LFS, but it doesn’t track file types on its own. In your terminal window, type **git lfs track “\*.psd”**.
5. If you look in your project, you’ll see a file named .gitattributes. This is similar to .gitignore, but instead of listing files that should be ignored, it lists files that Git can track for LFS.

Paste this into your .gitattributes file: 