**TEMASEK POLYTECHNIC**

**SCHOOL OF INFORMATICS & IT**

**DIPLOMA IN IMMERSIVE MEDIA & GAME DEVELOPMENT**

**AY2024/2025 OCTOBER SEMESTER**

**GADV (CGE2C25)**

**Introduction to Unity**

**Working with GitHub**

To see the additional comments and resources, make sure you select **All Markup** in the **Review/Tracking** pane



**Objectives**

You might have used GitHub before, but that was a while ago, so let’s revise the main concepts here!

In this worksheet, you’ll learn how to use GitHub, a cloud-based version control system, to manage your Unity game development projects. By the end of this worksheet, you will be able to:

* Understand what GitHub is and why it is essential for game development.
* Create a GitHub account and set up **GitHub Desktop**.
* Create, publish, and manage a GitHub **repository** for your Unity projects.
* Understand and use essential **Git commands** such as commit, push, fetch, and pull.
* Implement a **.gitignore** file to manage files effectively in Unity.

By following these step-by-step instructions, you will gain hands-on experience in using GitHub to track changes, collaborate, and maintain a secure version of your project. It can really be a lifesaver! 😊

1. **What is GitHub?**

GitHub is a cloud-based platform for managing code *repositories*, which are structured collections of files that can be stored online, downloaded (or *cloned*) to your computer, and updated with changes. Repositories can be *public* (visible to everyone) or *private* (restricted access). Each repository can have multiple *branches*, allowing you to work on different features or versions of your project without affecting the main codebase.

GitHub is also an amazing resource, with many public repositories available to browse, e.g. the C# source code for Unity:



https://github.com/Unity-Technologies/UnityCsReference

**Why is GitHub Important for Game Development?**

This example shows how useful GitHub is:

* **Repositories**: Imagine you are working on a GADV project called 2D Platformer. You create a repository named GADV\_Platformer to store all your game scripts, assets, and scenes.
* **Cloning**: You switch to another computer in the lab and need access to your project. You clone the repository from GitHub, so you have the latest version on your local machine.
* **Branches**: You want to add a double-jump mechanic but aren’t sure if it will work. You create a branch called double\_jump\_test, allowing you to experiment without affecting the main game code. If it works, you merge it back into the main branch.
* **Visibility**: You can make your repository private while developing to prevent others from copying your work, but when submitting assignments, you might set it to public so your lecturer can access it.
* **Version Control**: Halfway through the assignment, your character movement breaks due to an unintended change. Because you committed your work regularly, you can revert to an earlier version before the bug occurred.

Using GitHub ensures that your assignments and projects are safely backed up, enables collaboration with teammates, and helps track every change you make, allowing you to revert if something goes wrong.

This can be a real lifesaver! (It’s actually a good feeling to know that all your work is safely backed up on GitHub, and that any changes can be reversed if things go wrong—just remember to make regular backups!)

By using GitHub, you gain professional version control skills that are widely used in the game industry.

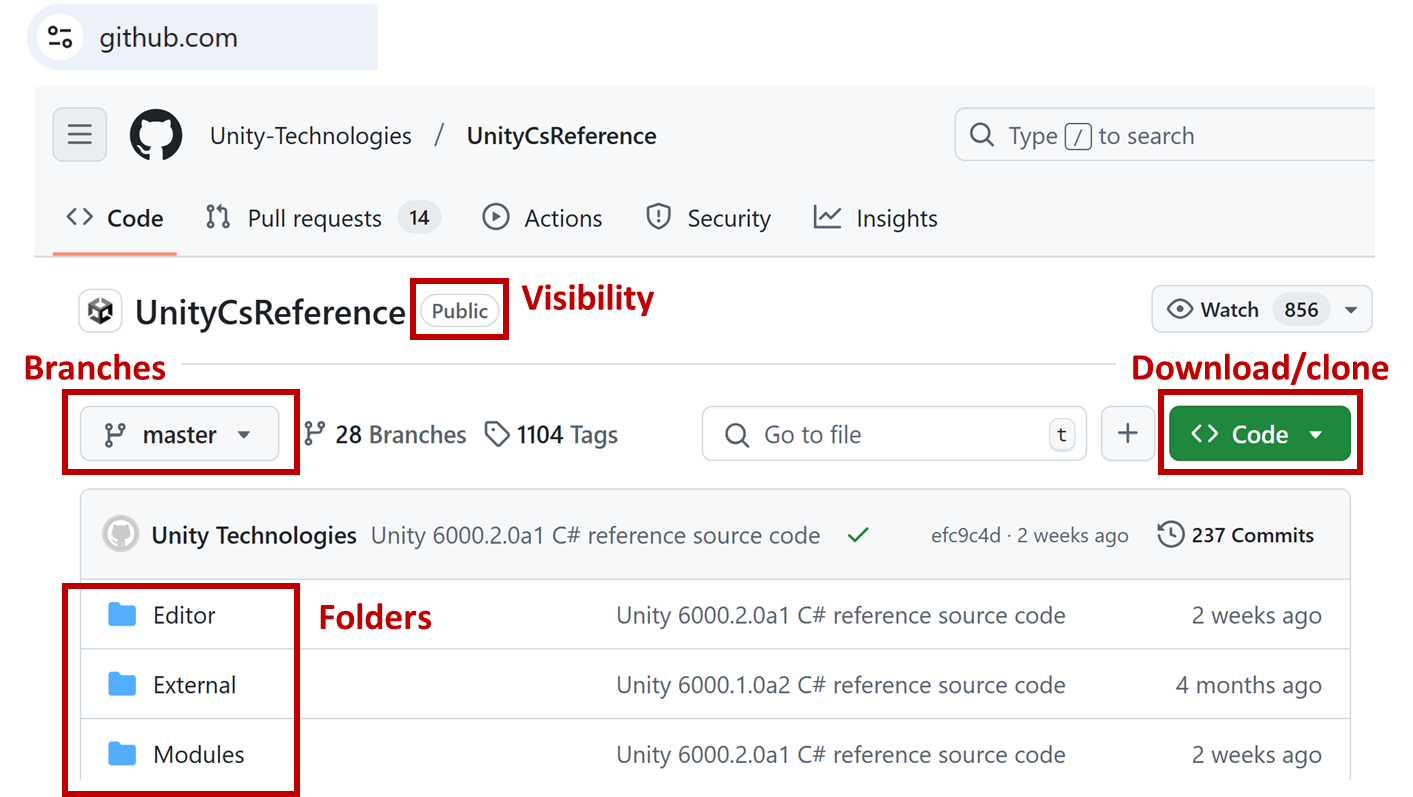
You should get into the habit of using GitHib for everything you do 😊

1. **Getting Started: Signing Up for GitHub**

**Step 1: Sign Up for a GitHub Account**

1. Go to [github.com](https://github.com/).
2. Click **Sign Up** and enter your email, username, and password.
3. Follow the verification steps and set up your profile.
4. Choose the free plan.
5. You can now access GitHub from any browser or through GitHub Desktop.

This is a typical GitHub page:



1. **Understanding Key GitHub Concepts**

**3.1 What is a Repository?**

A **repository** (or **repo**) contains all the project’s files, as well as the history of changes made to them. When you create a repo, GitHub maintains a **remote** repo and a **local** repo.

* **Local**: A Git repository stored on your computer, tracking changes locally.
* **Remote**: A Git repository hosted online.

A **local repo** and a **remote repo** are linked to **sync** changes between them.

* You **commit** changes locally in your local repo.
* You **push** changes to the remote repo (GitHub) to back up or share them.
* You **pull** updates from the remote repo to sync new changes from teammates.

**3.2 Understanding Version Control Terms**

* **Commit:** A snapshot of your project at a specific point in time.
* **Push:** Uploads your local changes to the GitHub repository.
* **Fetch:** Checks for new updates in the remote repository.
* **Pull:** Retrieves and merges updates from GitHub into your local copy.
* **Branch:** A separate version of your project, useful for testing features.

**3.3 Understanding the .gitignore File**

A **.gitignore** file tells Git which files or folders to ignore when committing changes to a repo.

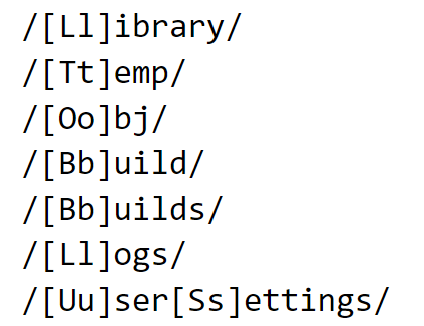
You will generally have a different .gitignore file for different types of projects.

In Unity projects, this is crucial because Unity generates many temporary files that don’t need to be tracked.

**Common Entries in a Unity .gitignore File**

* Library/ (Unity cache files)
* Temp/ (Temporary files)
* Logs/ (Log files)
* \*.csproj, \*.sln (Visual Studio project files)
* Build/ (Generated game builds)
* UserSettings/ (User preferences)

Here’s the first few lines of a typical .gitignore for Unity projects:



Let’s look at the first two lines to get an idea of how it works.

1. **/[Ll]ibrary/**

* Ignores the **Library** folder (case-insensitive, [Ll], so it matches both Library/ and library/).
* The **Library folder** contains cached and imported assets, which Unity can regenerate, so tracking it is unnecessary.

1. **/[Tt]emp/**

* Ignores the **Temp** folder, which stores temporary files that Unity creates while running.
* The files change frequently and don’t need to be committed to Git.

Why Ignore these?

* The Library and Temp folders are **not needed** in Git since Unity **rebuilds them automatically**.
* Ignoring them **reduces repo size** and **prevents unnecessary changes** from cluttering commits.

GitHub provides a default **.gitignore** file for Unity projects, but you can customise it to fit your needs.

GitHub can be managed via the website or a command line terminal window (e.g. Windows Powershell).

But it’s generally easier to use **GitHub Desktop**.

1. **Getting Started with GitHub Desktop**



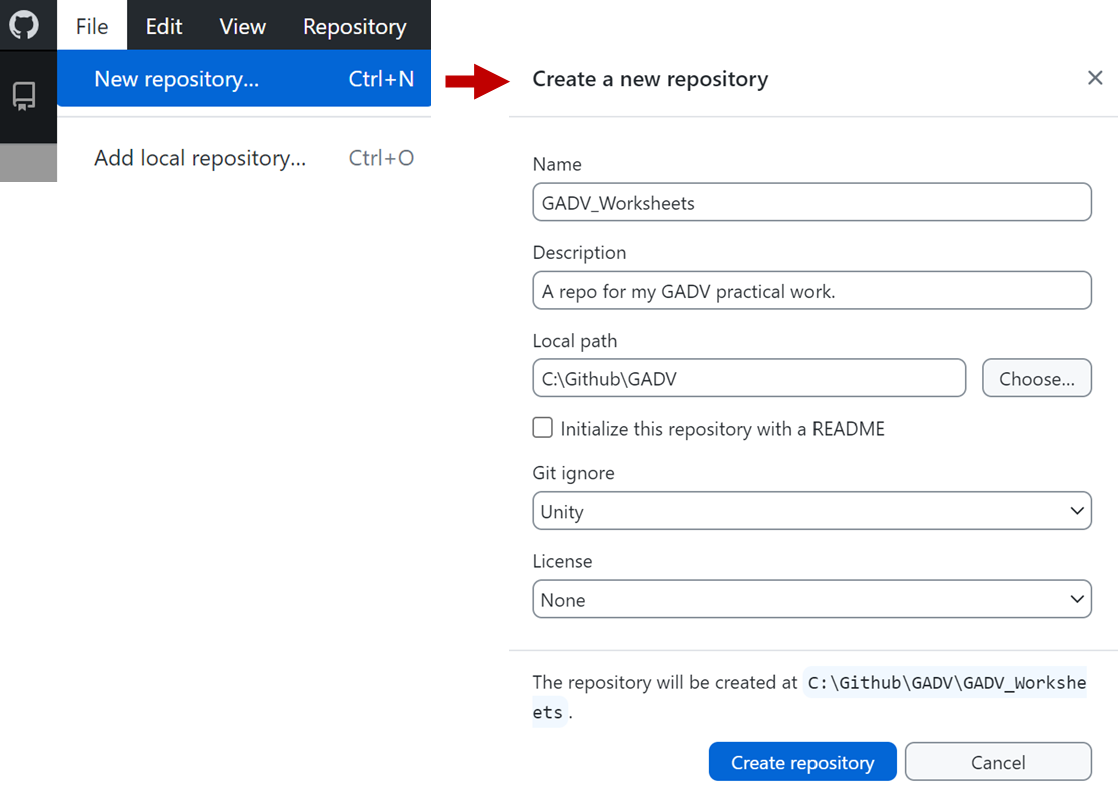
GitHub Desktop provides an easy way to use Git without needing to type commands in a terminal. You can watch this brief YouTube [video](https://www.youtube.com/watch?v=PvUexC0-D2s) about the basics of GitHub Desktop before reading further:

**Step 1: Install GitHub Desktop**

1. Go to [desktop.github.com](https://desktop.github.com/).
2. Download and install GitHub Desktop for your operating system.
3. Open GitHub Desktop and sign in with your GitHub account.

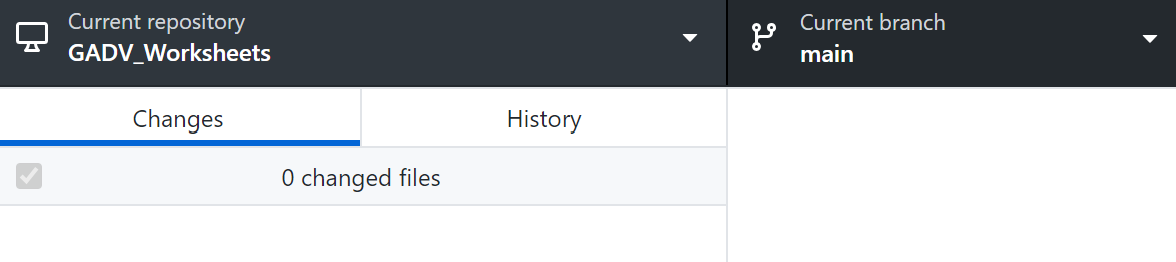
**Step 2: Create a New Repository from Scratch**

1. Click **File > New Repository**.



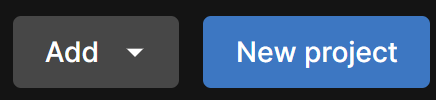
1. Name the repository **GADV\_Worksheets**.
2. Choose a local path where your Unity project files will be stored, e.g. **C:\Github\GADV** as shown in the screenshot above.
3. Select **Git Ignore: Unity** to generate a Unity-specific **.gitignore** file.
4. Click **Create Repository**.
5. GitHub Desktop will create the **GADV\_Worksheets** inside the specified local path, e.g. inside the **C:\Github\GADV** folder. You will now have the following folder setup: **C:\Github\GADV\GADV\_Worksheets**.

The repo is empty, since you haven’t added or created any files yet:

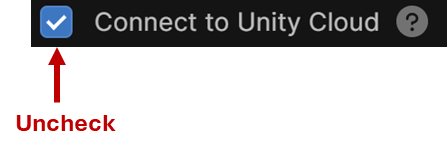


Now you should create a new Unity project in this GitHub repository folder.

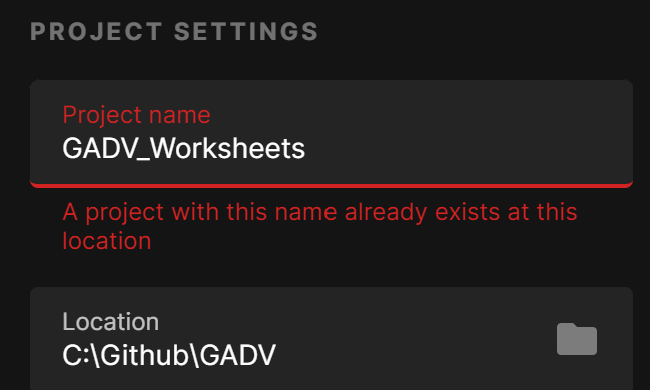
1. Create a new Unity project using Unity Hub.



There’s no need to connect to Unity cloud, so uncheck this:



1. Set the **project name and location** as shown below:

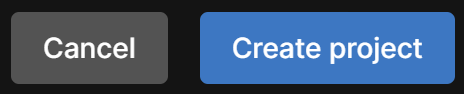


But there’s an error! Unity sees the **GADV\_Worksheets** repo folder and

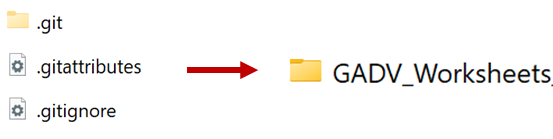
To get around this, open Explorer and rename the repository folder to:



Then in Unity Hub create the Unity **GADV\_Worksheets** project (you might have to retype the project name to remove the error):

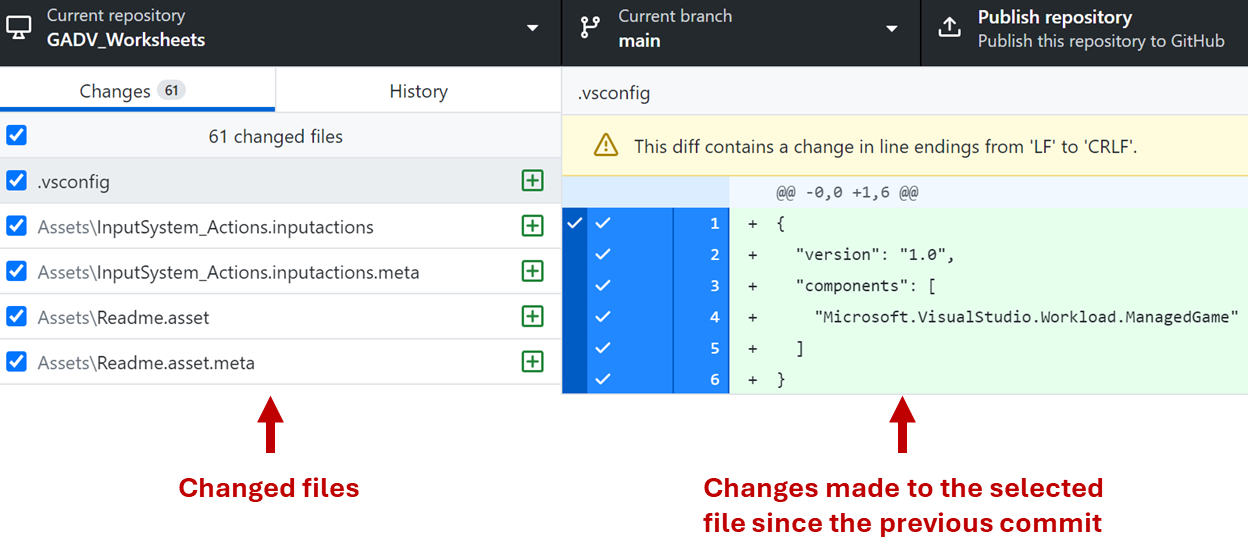


Once the Unity project is created (this might take a while!), copy the GitHub related files from **GADV\_Worksheets\_Temp** to the Unity project folder, **GADV\_Worksheets**.



Then delete the **GADV\_Worksheets\_Temp** folder.

In GitHub Desktop, your repo should now look like this:



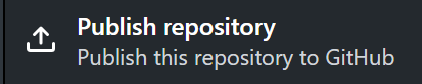
The changes to the project (newly added and modified files) are stored inside the **.git** folder inside **GADV\_Worksheets**.

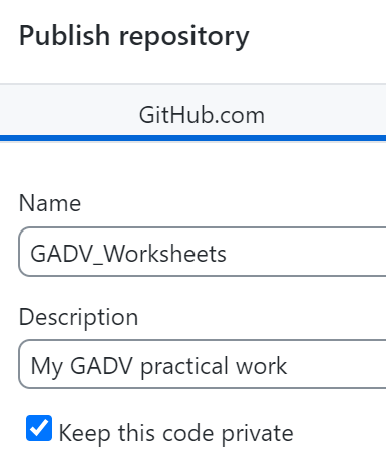
This is the local repositiory.

You must now publish these changes to the remote repository in the GutHub cloud so that GitHub can store and track them.

**Step 3: Publish the Repository to GitHub**

1. In GitHub Desktop, click **Publish Repository** (at the top, near the menu bar). Ensure the repository is set to **Private**.

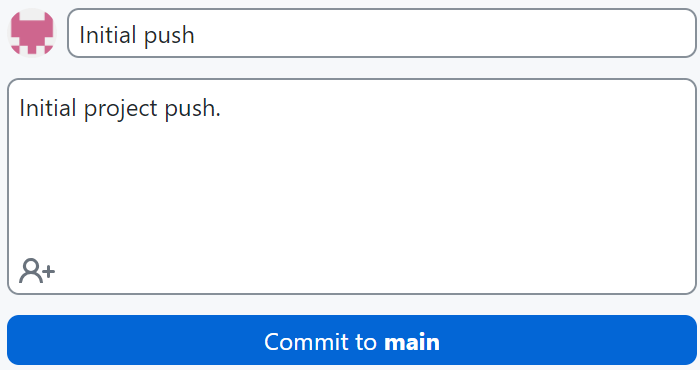




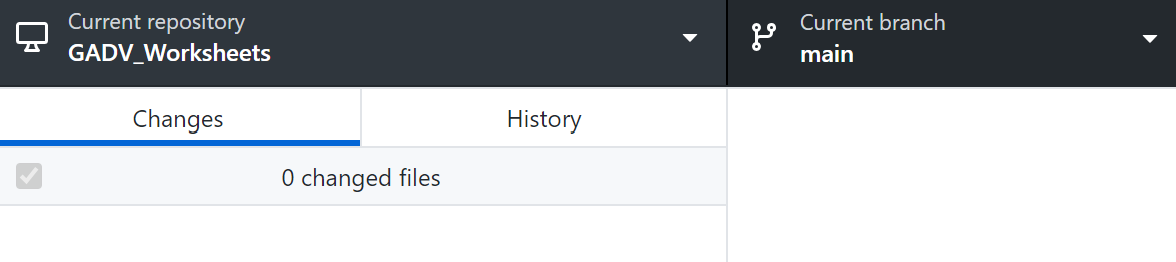
1. Click **Publish Repository** to upload your project to GitHub.

**Step 4: Committing Changes**

1. Open your Unity project and make some changes (e.g., create a new scene or script, or edit some code).
2. In GitHub Desktop, you will see the modified files listed.
3. Write a brief commit message (e.g., "Added main scene").
4. Click **Commit to main** to save changes locally.



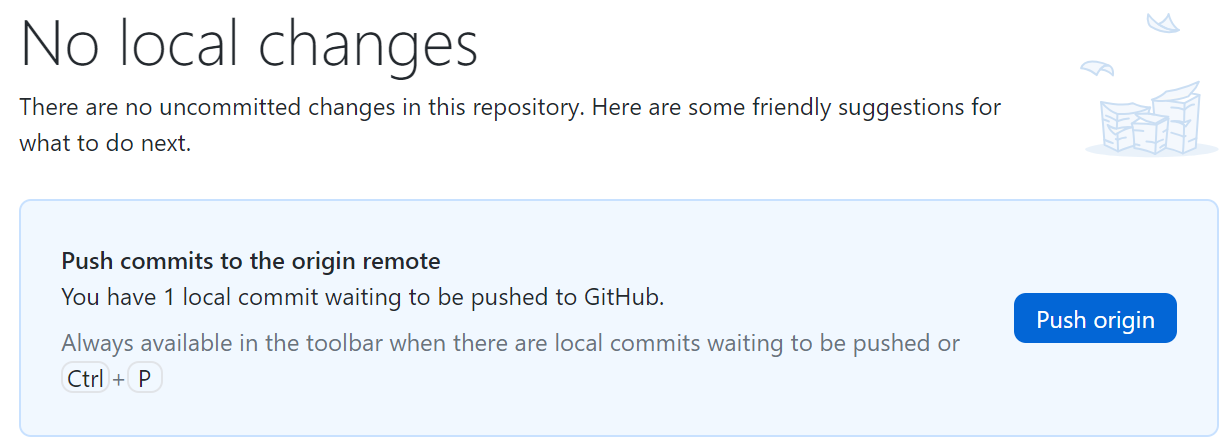
You’ll see that the **Changes** list becomes empty again.



This is because GitHub Desktop has recorded all the changes.

**Step 5: Pushing Changes to GitHub**

1. After committing, click **Push Origin** to upload the changes to GitHub.



**Step 6: Fetching and Updating from GitHub**

If working in a team, you need to sync changes:

1. Click **Fetch Origin** to check for updates.
2. If there are updates, click **Pull Origin** to download the latest changes.

By following these steps, you have successfully set up GitHub for version control in your game development projects!

This sets up a new repo. But what if you have an existing project that you want to upload to GitHub? As mentioned earlier, it’s better to create your Unity project first and then add this as a local repository to GutHib.

1. **Setting Up a Repo from an Existing Unity Project**

If you already have a Unity project and want to track its changes using GitHub:

**Step 1: Prepare Your Unity Project**

1. Open your Unity project in Unity Hub.
2. Ensure your project is inside a well-organised folder.
3. Remove any unnecessary temporary files by deleting the **Library**, **Temp**, and **Logs** folders (these will be ignored later).

**Step 2: Initialise Git in Your Project Folder**

1. Open **GitHub Desktop**.
2. Click **File > Add Local Repository**.
3. Click **Choose…** and select your Unity project’s folder.
4. Click **Create a Repository** and ensure these settings:

* **Name:** Use a meaningful name (e.g., **GADV\_Project**).
* **Git Ignore:** Select **Unity**.
* **Description (Optional):** Briefly describe your project.

1. Click **Create Repository**.

**Step 3: Publish Your Repository to GitHub**

1. In GitHub Desktop, click **Publish Repository**.
2. Set the repository **to private** unless you want it to be public.
3. Click **Publish Repository** to upload your project to GitHub.

**Step 4: Committing and Pushing Changes**

1. Open Unity and make a change (e.g., add a new GameObject or script).
2. In **GitHub Desktop**, you’ll see the modified files.
3. Write a **commit message** (e.g., "Initial commit with game setup").
4. Click **Commit to main** to save changes locally.
5. Click **Push Origin** to upload changes to GitHub.

**Step 5: Keeping Your Repository Updated**

1. Whenever you make changes in Unity, repeat the **commit and push** steps.
2. If working in a team, click **Fetch Origin** to check for updates.
3. Click **Pull Origin** to get the latest updates from teammates.

**Summary**

You should now understand the basics of GitHub and how it helps with version control in game development.

You have learned:

* The difference between local and remote repositories.
* How to commit and push changes to track your work.
* How to set up GitHub for a new or existing Unity project.
* The importance of a .gitignore file in Unity projects.

Using GitHub ensures your work is backed up, organised, and easy to revert if something goes wrong.

It also makes collaboration with others much easier. However, GitHub offers many more features beyond what we covered.

For example:

* Pull requests, which allow teams to review changes before merging them.
* Issues and project boards, which help manage tasks in game development.

You are encouraged to do some independent learning to explore GitHub further and develop good version control habits early—these skills are widely used in the game industry.

Congratulations on setting up your GADV GitHub repo! 😊

