

COMP1150/MMCC1011 Week 2 Prac

Topics covered:

- Version control
- GitHub, GitHub Desktop
- Simple Git Workflow (commit, push, pull)

Version Control

A **Version Control System** (VCS) is a way to keep track of changes to a collection of files in a project, such as a Unity project. It works like cloud storage such as Google Drive or Dropbox, but includes several extra features:

- A complete version history is kept for every individual file in the project.
- Ability to roll back changes to individual files.
- Ability for multiple developers on a team to work on the same set of files.

Version control works by having one central **repository** (or 'repo') on a server, which contains the master copy of all the files in the project. Developers can **pull** (or download) copies of the files onto a local machine, edit them, then **push** (or upload) the changes back to the repository. The VCS will automatically track which files have been changed and only upload the changes. Another developer on a different machine can then pull those changes onto their machine, without having to re-download the entire project.

The VCS also allows you to document the history of your project. Every time you push changes to the repository, you are asked to include comments describing your change, and these comments are kept in a log. Good commenting is a vital practice for working on large projects, as it will make tracking your changes (and reverting to the correct version) much easier.

The VCS also acts as a backup of all your files. If ever you lose any work, or make a change that accidentally breaks your code, you can roll back to an earlier version.

VCS tools can be difficult to learn, but extremely useful once you've mastered them. Any large software development project is bound to use version control to keep track of their projects, including video game development.

There are several different version control systems to choose between. In this course, we will be using one called [Git](#), which is widely used in the software development industry. Git is a powerful and complex tool, but there are GUI-based clients that make it more user friendly. We'll be using a client called [GitHub Desktop](#). We'll also be using [GitHub](#) to host your pracs and assignments. The Unity portion of your Game Design Task assessment will be distributed and submitted through GitHub (rather than iLearn).

Getting started

The first things you will need are:

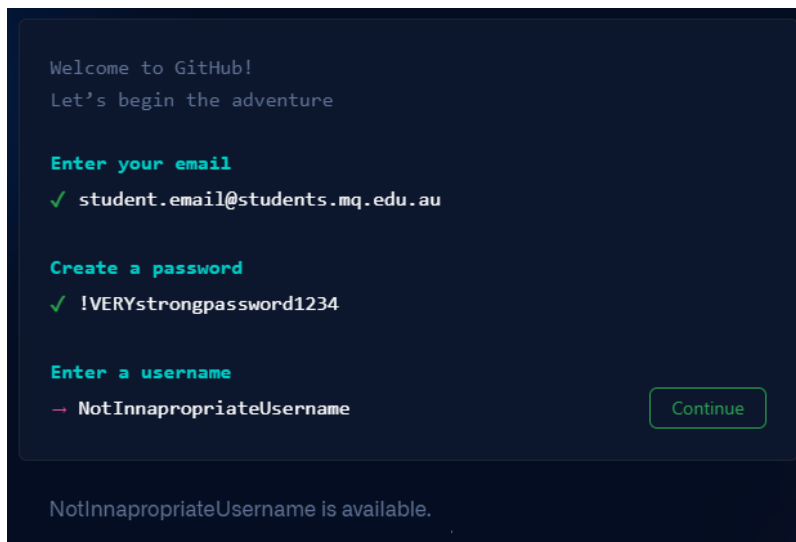
- 1) A GitHub account
- 2) A copy of GitHub Desktop (a Git client)

Note: If you are already familiar with Git and wish to use a different client, you are welcome to do so, however for the pracs we will assume you are using GitHub Desktop.

GitHub

To get started with today's prac, you'll first need to make a GitHub Account:

1. Go to the GitHub web page: <https://github.com/signup>.
2. Type in a username, your MQ email address, and a password.
 1. While you may use any username you want, this will be publicly visible so keep it appropriate
3. Click **Sign up for GitHub**.



Wellcome to GitHub!
Let's begin the adventure

Enter your email
✓ student.email@students.mq.edu.au

Create a password
✓ !VERYstrongpassword1234

Enter a username
✗ NotInnapropriateUsername

Continue

NotInnapropriateUsername is available.

Once your account is set up you may encounter a brief survey so GitHub knows what you are using the account for. Finally, you will need to verify your email to finish setting up the account.

Accepting an assignment

Each time we need to give you a new project, we will be creating an assignment for you in GitHub Classroom to accept and download. This allows us to keep track of your work (provided you make sure to update your repository as you go!).

For today's class, we'll be accepting the **COMP1150 2D Prac** assignment. Click on the following link:

<https://classroom.github.com/a/dNMPUYkz>

Find your **Student ID** in the list of 'Identifiers' and click on it to link it with your GitHub account.

Join the classroom:

COMP1150-22s1

To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? [Skip to the next step](#) →

Identifiers

It is important that you click on your own student ID and not someone else's. Double check this! If after **checking twice** you can't find your number in the list, choose 'Skip to the next step' and then make sure you email the lead prac tutor (oliver.smith@mq.edu.au) so that your ID can be properly linked to your GitHub account.

You'll now need to **accept** the assignment.

COMP1150-22s1

Accept the assignment —

COMP1150-22S1-2D-Prac

Once you accept this assignment, you will be granted access to the `comp1150-22s1-2d-prac-OliverSmithMQStaff` repository in the [MQGames](#) organization on GitHub.


Accept this assignment

This will create your assignment repository for you. This may take a couple of minutes, so you should wait before refreshing the page to check if it's ready.

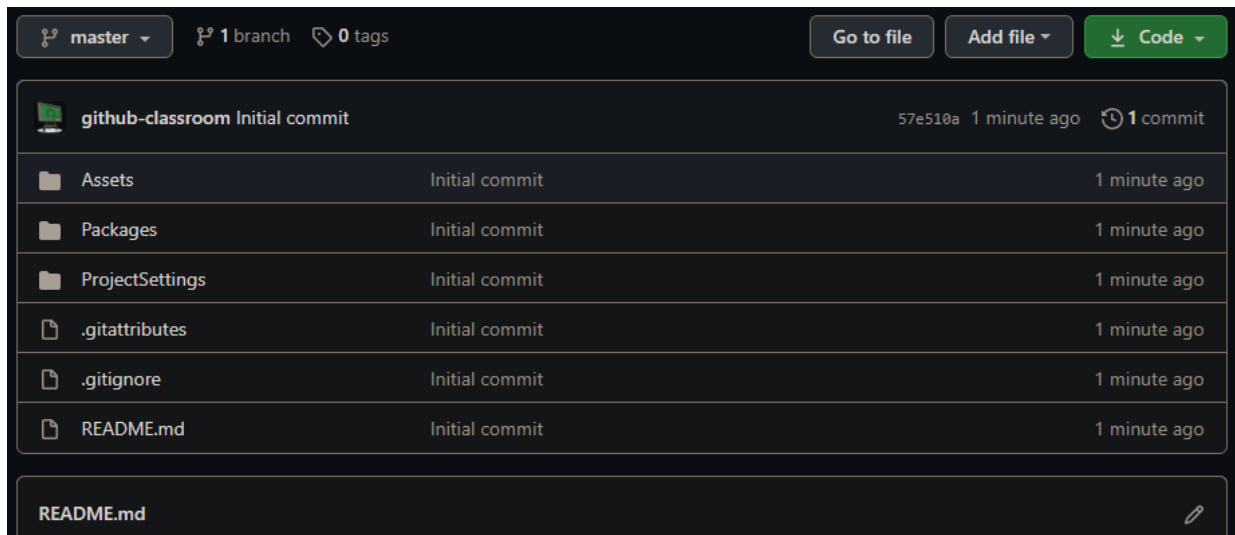
You're ready to go!

You accepted the assignment, **COMP1150-22S1-2D-Prac**.

Your assignment repository has been created:

 <https://github.com/MQGames/comp1150-22s1-2d-prac-OliverSmithMQStaff>

This repository contains a Unity project with some 2D sprites, as well as some important meta files (a `.gitignore` and a `.gitattributes`). You can click on the assignment link to view the project in GitHub. It should look something like this:



GitHub Desktop

Now that we've got our repository set up, it is time to pull it to your computer. To download, back-up and update our work, we will be using a Git Client. For this class, the client we're using is **GitHub Desktop**. If you are using the lab computers, open GitHub Desktop. If you are using your own device, you'll need to download and install GitHub Desktop from:

<https://desktop.github.com/>.

Note: the website will automatically detect your OS for the main download button, however if you are using an Apple Silicon (i.e. M1) Mac you may want to download the native version via the link below the main download button.

Once you've opened **GitHub Desktop**, you'll need to log-in by choosing 'Sign in to GitHub.com'. Github will launch your default internet browser and attempt to log you in (it may require you to log back into Github.com).

After you're logged in, GitHub Desktop will ask you to configure the name and email that will be associated with your commits, which can be copied from your account details or entered manually. We advise using your name, StudentID and student email as shown in the image below, so if those details are not correctly shown, you should update manually, before pressing **Continue**.

Configure Git

This is used to identify the commits you create. Anyone will be able to see this information if you publish commits.

Name

Email

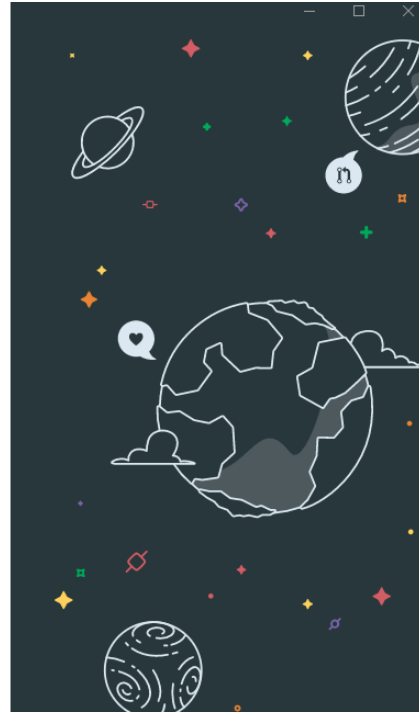
Continue

Cancel

Example commit

Fix all the things

FirstnameLastnameStudentnumber • 30m



Finally, you can untick share usage data (if you wish) and click 'Finish' to finalise setup.

Now that GitHub Desktop is set up we will work on **cloning** your repository. Cloning is the process of copying a repository from the server onto your computer. You'll need to do this every time you start working on a **new** computer. There are several ways to do this.

One way is to select your repository from the list in GutHub Desktop, then press the blue **Clone** button at the bottom of the screen to begin cloning your project.

Let's get started!

Add a repository to GitHub Desktop to start collaborating



Create a tutorial repository...



Clone a repository from the Internet...



Create a New Repository on your hard drive...



Add an Existing Repository from your hard drive...



ProTip! You can drag & drop an existing repository folder here to add it to Desktop

Your repositories

OliverSmithMQStaff/**COMP1150-3D-Pracs-2020S1**

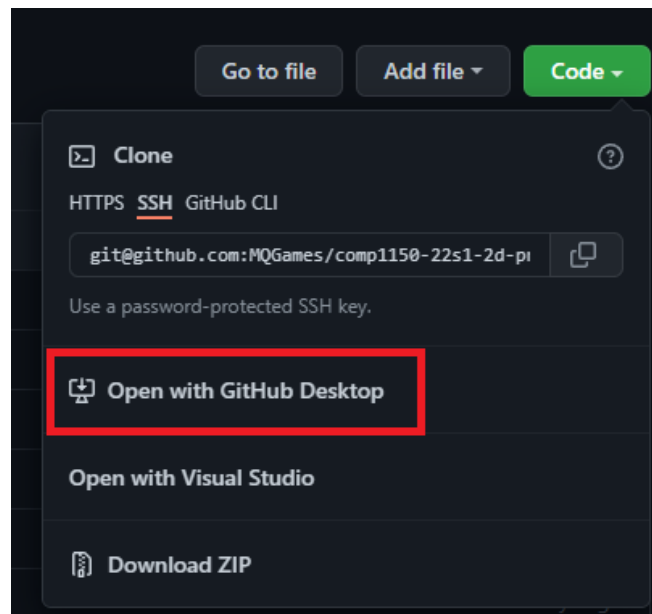
MQGames

MQGames/**COMP1150-22S1**-2D-Prac

MQGames/**comp1150-22s1**-2d-prac-OliverSmithMQStaff

Clone OliverSmithMQStaff/COMP1150-3D-Pracs-2020S1

Alternatively, you can return to the repo created for you from the classroom link on github.com, select the green **Code** button, and choose 'Open with GitHub Desktop'.



Either of these methods will then launch the correct clone details in GitHub Desktop, where you can press the blue **Clone** button to begin copying the project.

A screenshot of the 'Clone a repository' dialog box in GitHub Desktop. The dialog has a title bar with a close button. It features three tabs: 'GitHub.com', 'GitHub Enterprise', and 'URL', with the 'URL' tab selected. Below the tabs, there is a label 'Repository URL or GitHub username and repository' followed by '(hubot/cool-repo)'. A text input field contains the URL 'https://github.com/MQGames/comp1150-22s1-2d-prac-OliverSmithMQStaff.git'. Below this, there is a label 'Local path' and a text input field containing 'C:\Users\'StudentNumber'\Documents'. To the right of the local path input is a 'Choose...' button. At the bottom of the dialog, there are two buttons: a blue 'Clone' button and a grey 'Cancel' button.

The first address represents the link to the repository online, while the 'Local path' is where the project is going on your computer. The local path will automatically be set to create a new folder with the assignment name and your username at the designated location.

Important: On the Lab PC's it is critical that you place the repository in the following location:

C:\Users\<Student number>\Documents
(not the regular Documents folder).

For students on their own device we recommend creating a GitHub or separate folder in your Documents folder within which you can save all your projects.

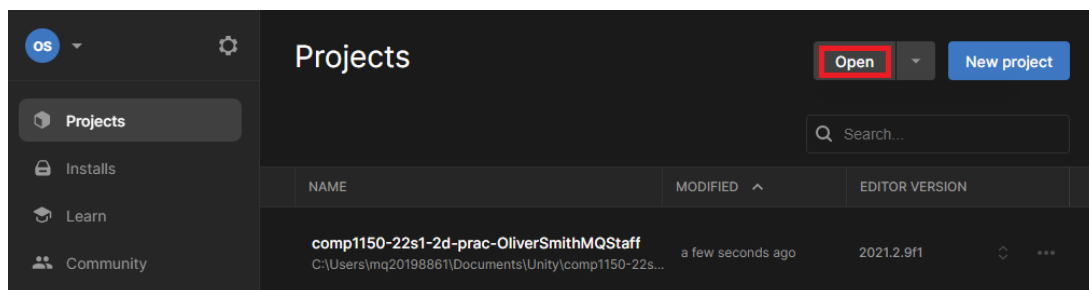
Make sure your repository is being cloned to the correct location, and then press **Clone**. Once the cloning process is complete, the project will now be downloaded to your computer.

Version Control in Unity

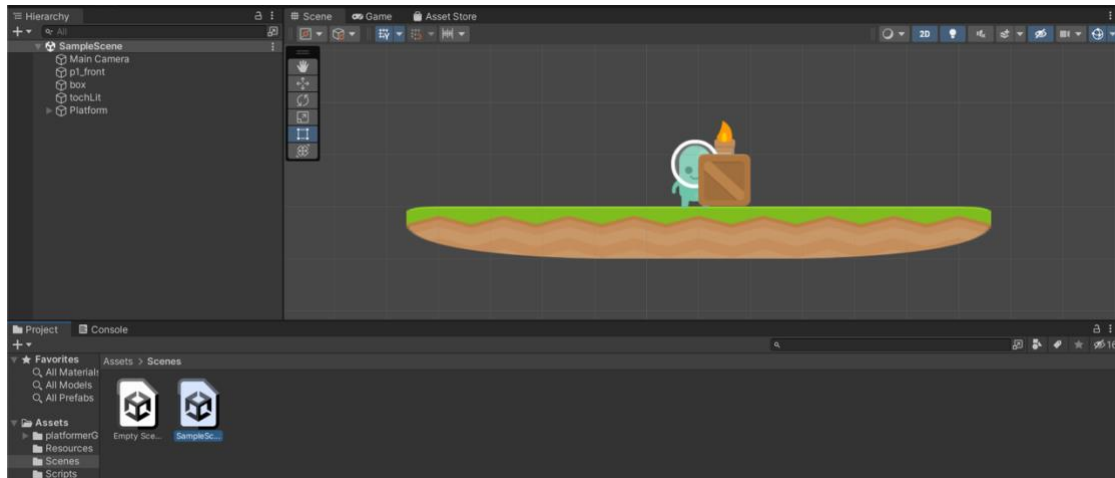
The project you've just cloned is a Unity project containing the 2D sprites package from last week. We'll be editing a scene to demonstrate how changes work with version control. Since you created a basic scene last week, let's first import that into the new project. Navigate to the scenes folder in your project files from last week (within the project folder, this should be located within Assets). You will see the scene from last week, e.g. 'SampleScene.unity' or whatever you saved it as.

Copy last week's scene (.unity file) to the Scenes folder in the new project. If you don't remember where the project is, you can check this in GitHub Desktop by clicking 'Repository > Show in Explorer/Finder'. If you can't locate your files from last week don't worry, you'll just need to recreate a basic scene in the new project to work in.

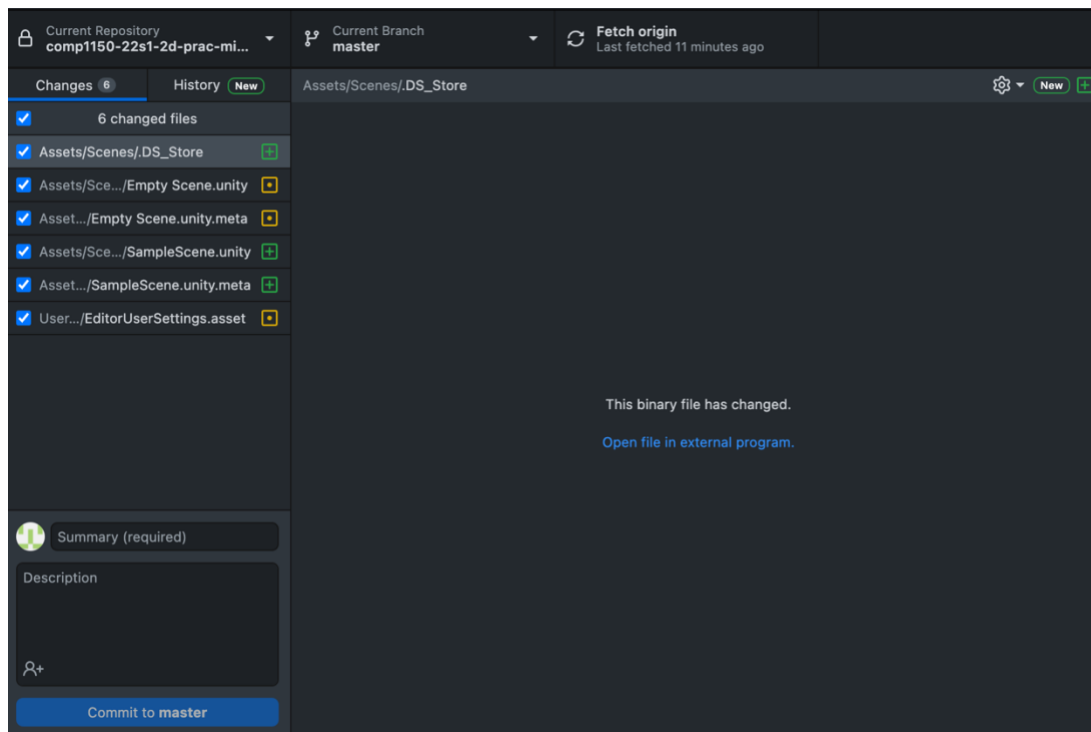
Next, open **Unity Hub**, make sure you are on the Projects Tab and press the **Open** button in the top right. Then navigate to where your repository folder is on the computer (if you kept the default settings this will be called 'comp1150-2d-prac-username';). Select the repository folder and press **Select Folder**. If it doesn't automatically open then click on the project in Unity Hub to open it in the Unity Editor.



Once the project has finished opening, in the Project panel within Unity navigate to the Scenes folder and double click on the scene from last week to open it. You'll notice that in the Hierarchy panel the top of the hierarchy changes to the name of the scene you just opened, and it should reveal the scene as you left it last week. If you couldn't locate your files from last week, just work in the existing 'Empty Scene' scene and quickly create a basic platformer scene (see example below).



Once you have a basic platformer scene ready, save it (call the scene something obvious like 'Platformer'), close Unity, and head back into GitHub Desktop. You'll now see something like this (depending on which files you changed):

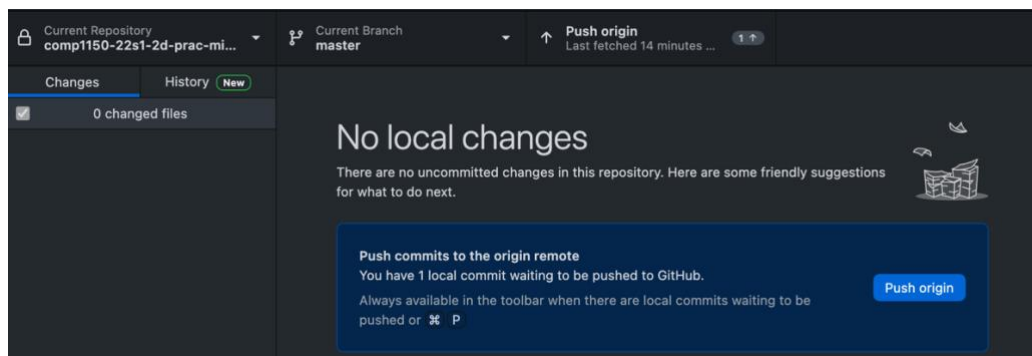


Under 'Changes', you'll see the total number of changed files, including your scene file that you added, and any additional files that you modified. GitHub Desktop has detected that you've made a new file, and so presents it here to prompt you to send the files up to GitHub and back them up. To send files to GitHub, we must do the following:

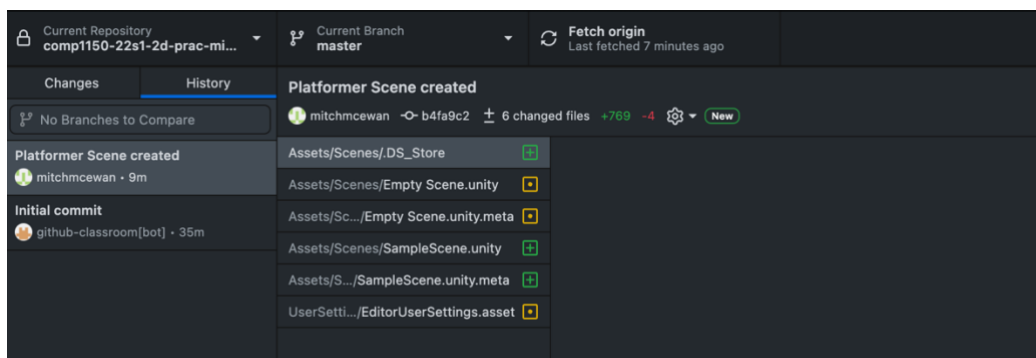
- Make sure the files are ticked in the 'changed files' list. New files are ticked by default.
- Add a **commit summary (required)**. Commit summaries are small messages we add to our updates so that we can keep track of our changes. It is **crucial** to use clear commit

messages. Type your message in the 'Summary' box – I wrote "Platformer Scene created". Then press **Commit to main** (this may also be called commit to master).

- Now our changes are locally recorded (committed). The last thing we need to do is **Push** our changes (to copy them to the server). You should now see a small '1↑' next to 'Push origin' at the top of the screen, and a blue notice in the main window that you have a local commit waiting to be pushed to GitHub. By pushing your changes, you'll update your repository, meaning everything is backed-up. Press **Push origin** and wait for it to update.




Once the push is complete, you can double-check that it has gone through by clicking on the **History** tab, where you can see the history of all commits made to the repository.



Seeing Your Changes Online

To verify that our changes went through to the server we can check they have been pushed to [GitHub.com](https://github.com).


Find your repository on Github.com. This can be found in a few ways. The simplest way is to open the repository directly from Github Desktop by clicking Repository > View on Github. The same button is also visible on the main GitHub window if your project doesn't have any uncommitted changes. Once you have found your repo on the web, you'll notice that on the right hand side there is a list of how long ago each of the files in your project were last updated. This is a useful column to look to in order to check if the most recently changed files in your last push have come through. It also indicates one of the core functions of version control systems, in that only the files that were modified in a project are updated.












mitchmcewan Merge branch 'master' of <https://github.com/MQGames/comp...>







a20c567

10 minutes ago

 5 commits

<div></div> Assets	Platformer Scene created	23 hours ago
<div></div> Packages	Initial commit	23 hours ago
<div></div> ProjectSettings	Initial commit	23 hours ago
<div></div> UserSettings	Another change	11 minutes ago
<div></div> .gitattributes	Initial commit	23 hours ago
<div></div> .gitignore	Initial commit	23 hours ago
<div></div> .vsconfig	Initial commit	23 hours ago
<div></div> README.md	Initial commit	23 hours ago
<div></div> test.text	Created a change that would force a Pull origin (for screencap purpo...	22 hours ago

Click the 'Commits' link in the top right above the list of last updated times. Clicking this link will take you to your commit history, which will look similar to the History tab from Github Desktop. This is a quick and easy way to verify that the changes were actually pushed to the server.

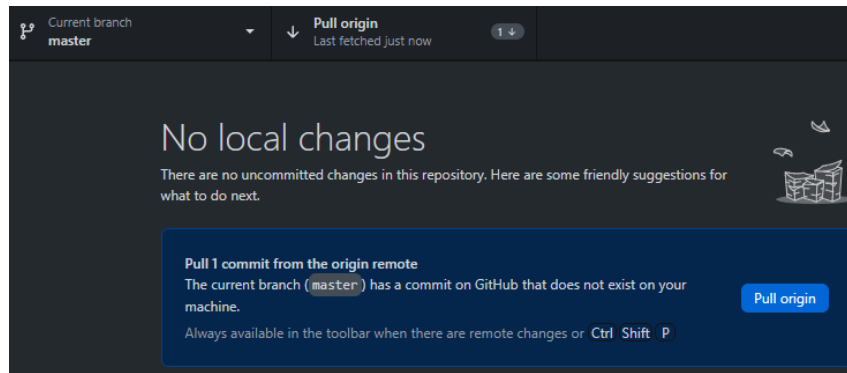
Platformer Scene created	 b4fa9c2 
 mitchmcewan committed 23 hours ago	
Initial commit	 adeb476 
 github-classroom[bot] committed 23 hours ago	

Working Across Different Devices

You can also test if a project has been successfully pushed to the cloud by downloading the updated repository onto another device. If you have another personal device that you can test this on, try cloning the repository to that computer by following the instructions earlier in the prac to pull your repository (including installing Github Desktop). Once that is done, open Github Desktop and check the History tab to confirm that your previous push was successful.

Once Unity is installed on this device, you can open your updated project on it, as well as make additional changes to your scene. Try making a small change (like adding a sprite, or slightly moving a sprite), save your scene, and push your changes.

Open GitHub Desktop on your original device, and you should see **'Pull origin'** and a **1↓** symbol where it notified you to push origin previously (note: you may need to 'Fetch origin' to refresh the page if you didn't close and reopen GitHub). This is Github Desktop's way of letting you know that it has confirmed a change on the server, and that your version (saved locally) is now out of date. Pressing 'Pull origin' will update this project.



When the pull is complete, open your project again through Unity Hub. The project should now reflect the changes from your last push (on your other device). Remember, you can always double-check your pushes by clicking on the History tab in Github Desktop, or by opening your repository on GitHub.com.

Important: You should always make sure you have the most up-to-date version of your Unity project on your computer before opening Unity, or you can end up with conflicting versions. **Always** fetch your project with GitHub Desktop to check that you have the most up-to-date version on that device before opening Unity. **Always pull before you push.**

Finishing up

You should now know how to do the following:

- Accept assignments through GitHub Classroom.
- Clone a project onto your computer.
- Commit and push changes to a repository.
- If you have a second device, pull changes to update a project on multiple devices.

Version control can be tricky and confusing at times, so we recommend you practice the steps of committing, pushing, and pulling as much as possible. You should be pushing your project frequently, as it will come in handy throughout this unit (e.g. the power goes out and you forgot to save, or you've been working on your project on a laptop at home, but have come into the lab without your laptop).

If you're a bit unsure about how to perform certain actions in Github you can always ask your prac demonstrator, ask on the [iLearn forums](#), look up [the Github Documentation](#), or check out [Github's help page](#).

Show your demonstrator:

To demonstrate your understanding of this week's content to your prac demonstrator you might show:

- Your GitHub repository and its history on Github Desktop, on one or multiple devices.
- That you can verify and understand changes to your repository on GitHub.com
- That you know how to push and pull files changed in Unity.