Class Software

Sebastian Grygorczuk - STEM Institute at CCNY

This Today's Agenda

- Go over how to create accounts for Unity, and GitHub.
- Go over the installation process of Unity Hub, Unity Editor, Visual Studio, and GitHub Desktop.
- Go over Version Control System and how it works.
- Clone and Fork the project files.

What Software Are We Using

Unity: Our Game Engine.

Visual Studio: The IDE we will be using

GitHub Desktop: The way we'll connect to the use of Version Control System.

GitHub

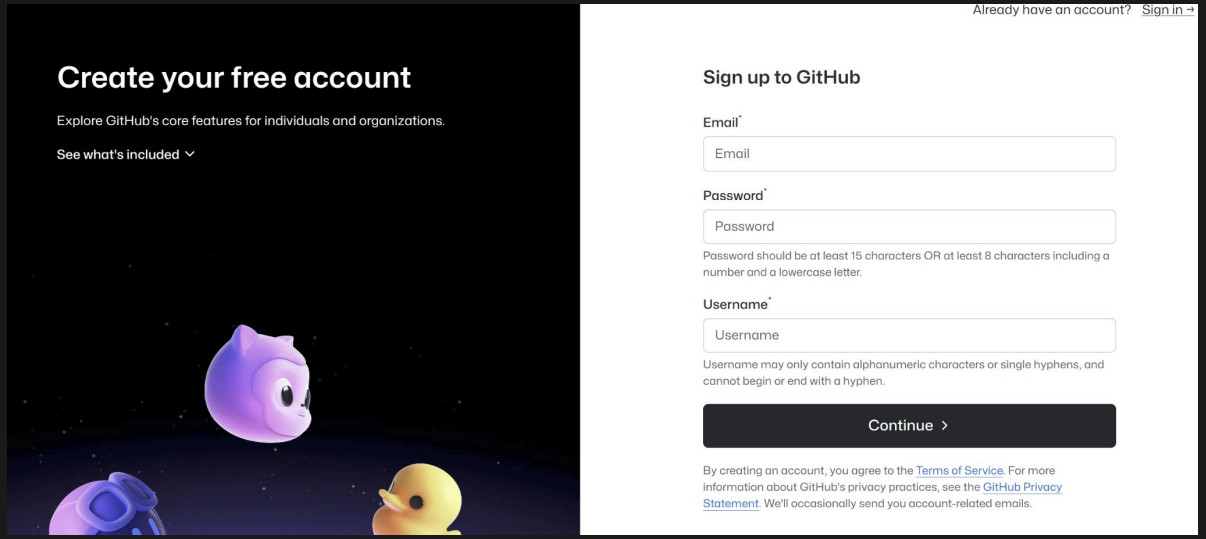
Creating A GitHub Account

Start by creating a GitHub account at https://github.com/. GitHub is a Version Control System (VCS) that helps you manage project files and track changes without conflicts. After setting up your account, you'll be able to connect it to Unity.

Creating A GitHub Account

Fill out the required information or use an existing account to sign in to Unity.

You'll need access to your email or phone to complete the sign-in process, so make sure you use credentials you have access to.



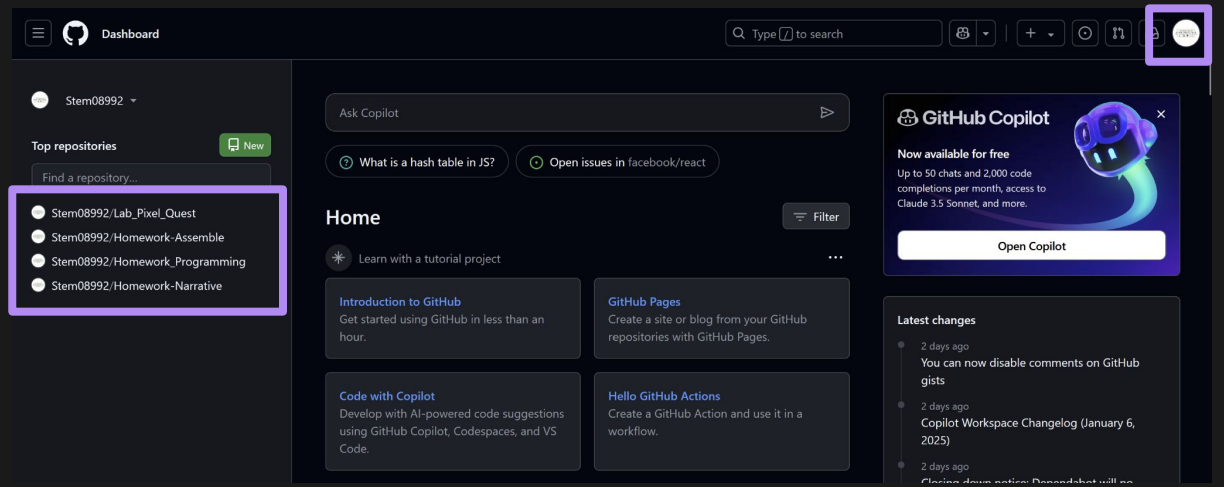
GitHub

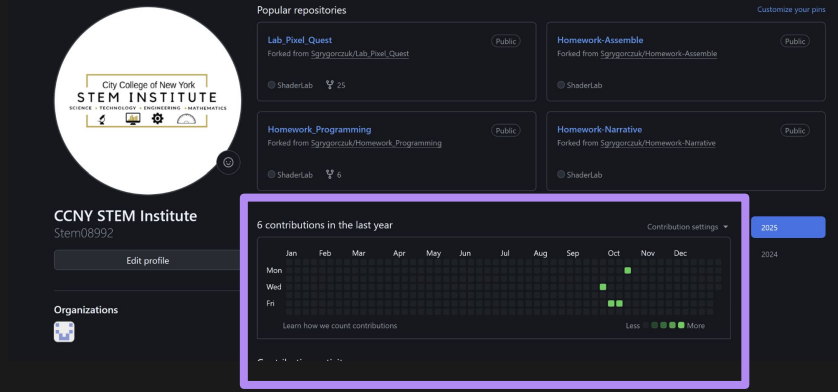
Once you've finished setting up your details, you'll be taken to the Landing Page.

On the sidebar, you'll see all the repositories you have access to.

You can view your profile by clicking on the user profile icon in the top-right corner.

Your profile page will display your pinned projects, commit history, and below that, your most recent commits.





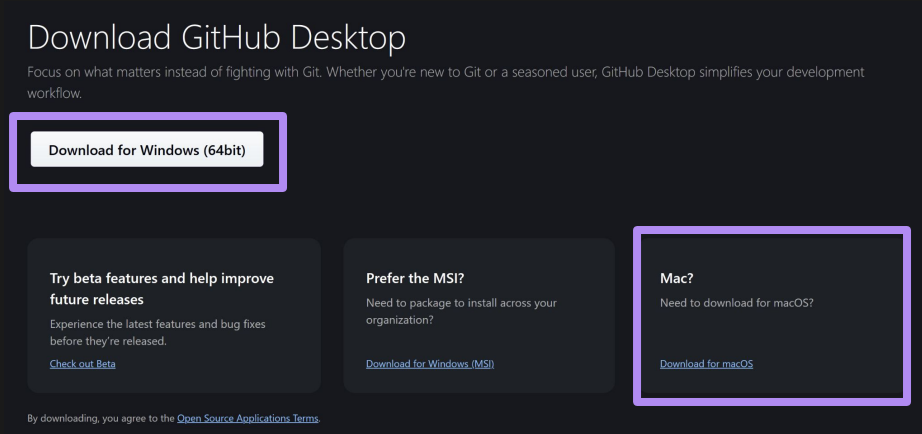
Downloading GitHub Desktop

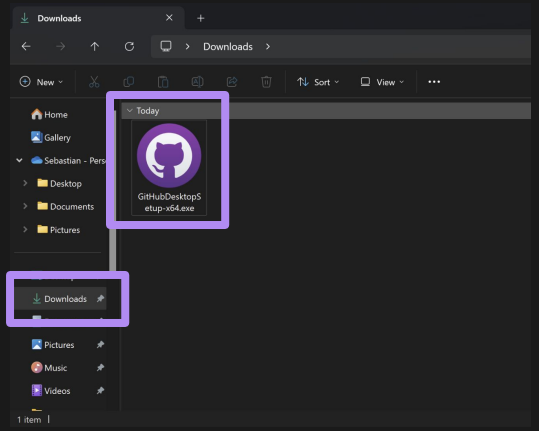
Once your GitHub account is set up, we can proceed to downloading GitHub Desktop.

Go to the following link to download the installer for your platform: https://desktop.github.com/download/.

After downloading, you should find the .exe file in your Downloads folder.

Simply double-click on it, and the installation process will complete in a few minutes.





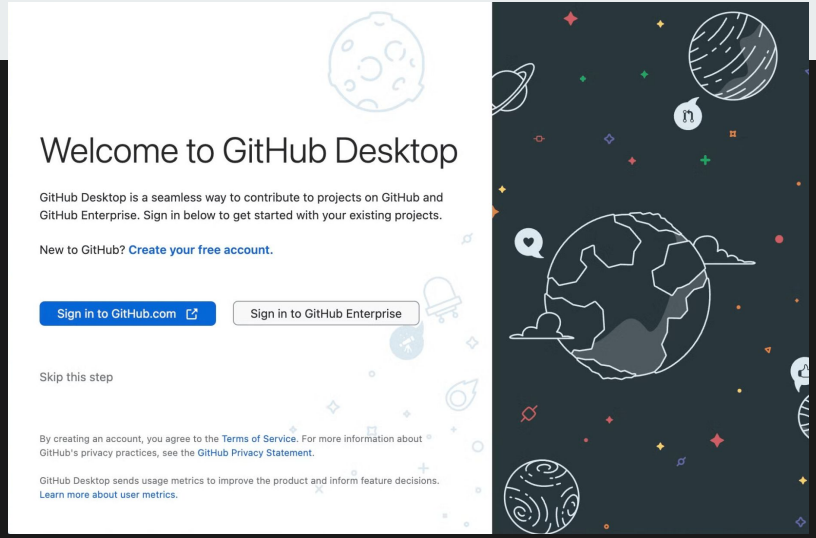
First Sign In

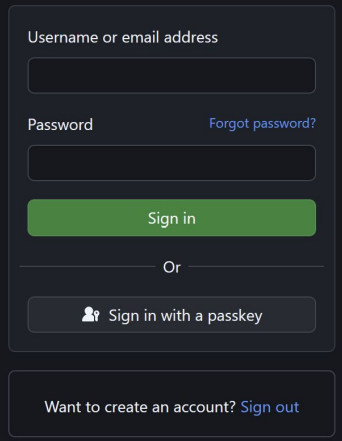
When GitHub Desktop first opens, you'll see a sign-in screen.

Click "Sign in to GitHub.com," and you'll be redirected to a web browser.

Enter your account details and, if necessary, verify your login via email or phone.

Note: The computers in the lab default to Internet Explorer, which doesn't support this login. Copy the URL from the browser that pops up and paste it into a more compatible browser like Chrome or Firefox.





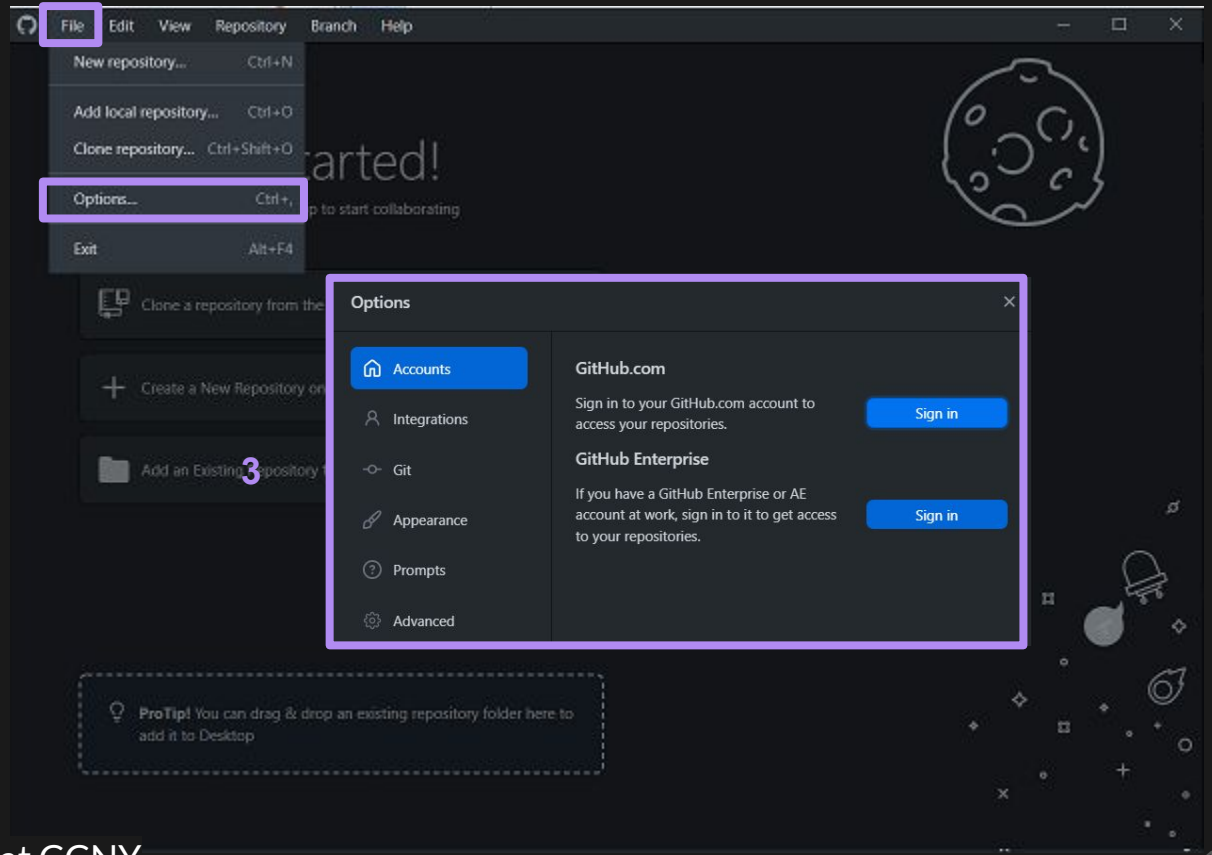
Sebastian Grygorczuk - STEM Institute at CCNY

GitHub Desktop

In case you logged in by skipping the sign-in process, or if the computer you're using has been used before, you can sign out of the previous account.

To do this, go to **File**, then select **Options**.

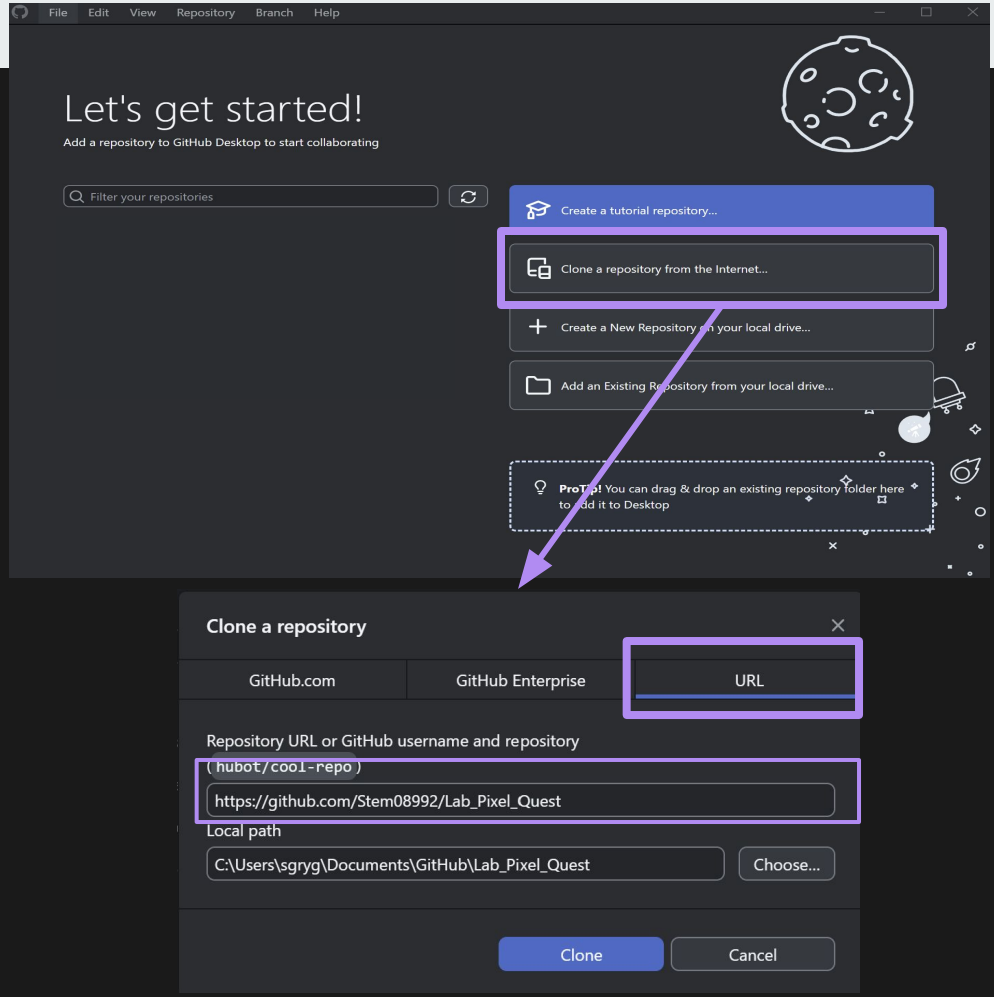
Under the **Accounts** tab, you'll have the option to sign out the previous person and log in with your own account.



First Clone

Once you log in, you will see the GitHub Desktop main screen. Since you don't have any repositories saved to your account yet, we'll need to clone (download) a project from the class account.

1. Click the **Clone a repository from the Internet** button. A pop-up window will appear.
2. In the pop-up, switch to the **URL** tab.
3. In the repository field, enter the following URL: https://github.com/Stem08992/Lab_Pixel_Quest
4. Leave the **Local Path** as is. This will create a new **GitHub** folder in your **Documents** folder, where the **Lab_Pixel_Quest** repository will be downloaded.
5. Click **Clone**, and the project files will be downloaded to your computer.

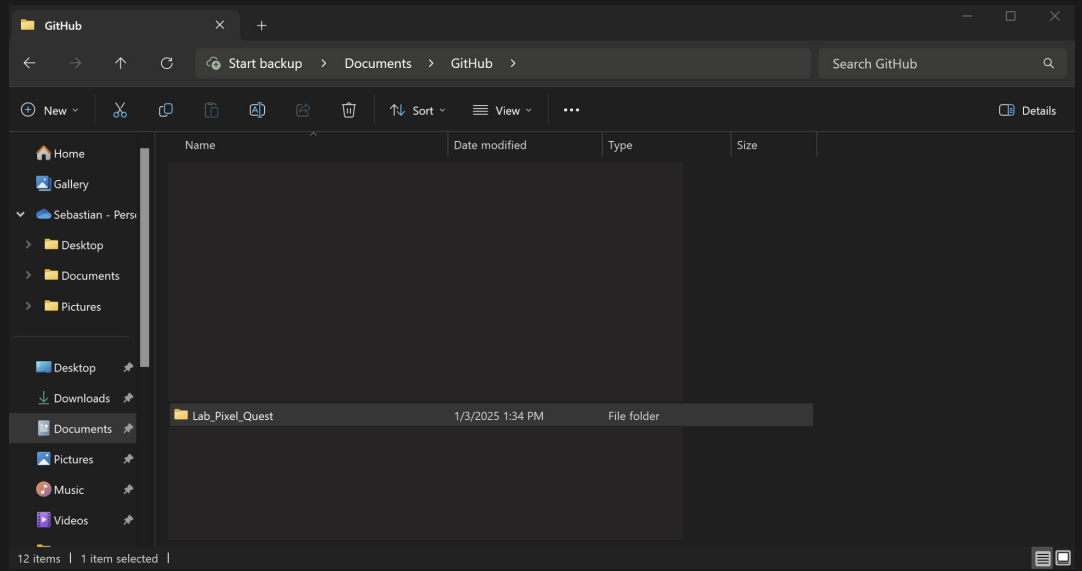


Sebastian Grygorczuk - STEM Institute at CCNY

Documents/GitHub/Lab_Pixel_Quest

Locating the Cloned Project Files: Once the project has been downloaded, follow these steps to find the files:

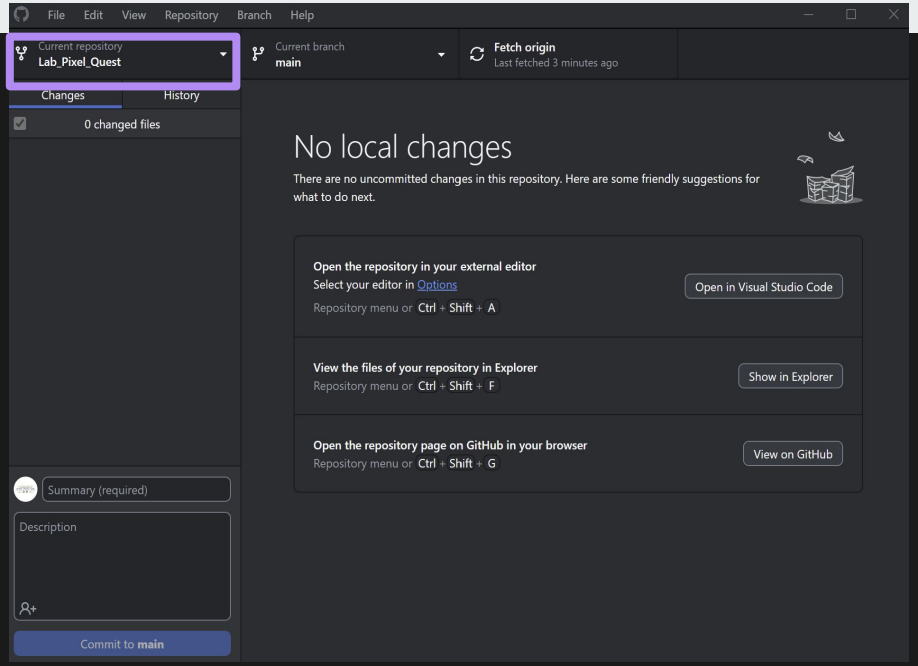
1. Open **File Explorer**.
2. Use the sidebar to navigate to the **Documents** folder.
3. Inside the **Documents** folder, open the **GitHub** folder.
4. This is where all the repositories you clone will be stored, including the **Lab_Pixel_Quest** project.

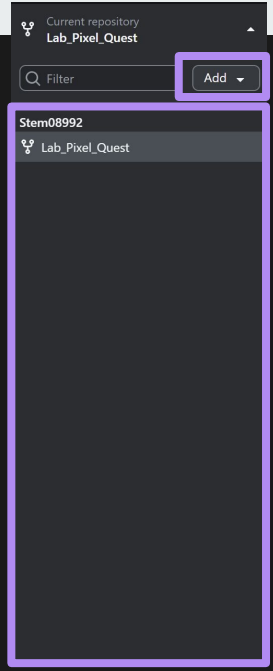


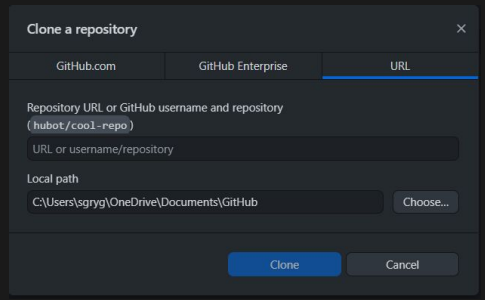
Project Directories

Desktop: Once the project is cloned, GitHub Desktop will look a lot different. Here's a breakdown of the key sections:

- **Project Name:** Clicking on this will allow you to switch to a different project. You can click it again to return to the "Changes" view.
- **Project List:** This list will appear, showing all the projects currently connected to GitHub Desktop. You can switch between them as needed.
- **Add Button:** Clicking this button opens the cloning pop-up, which lets you download other project files, like homework or additional projects.





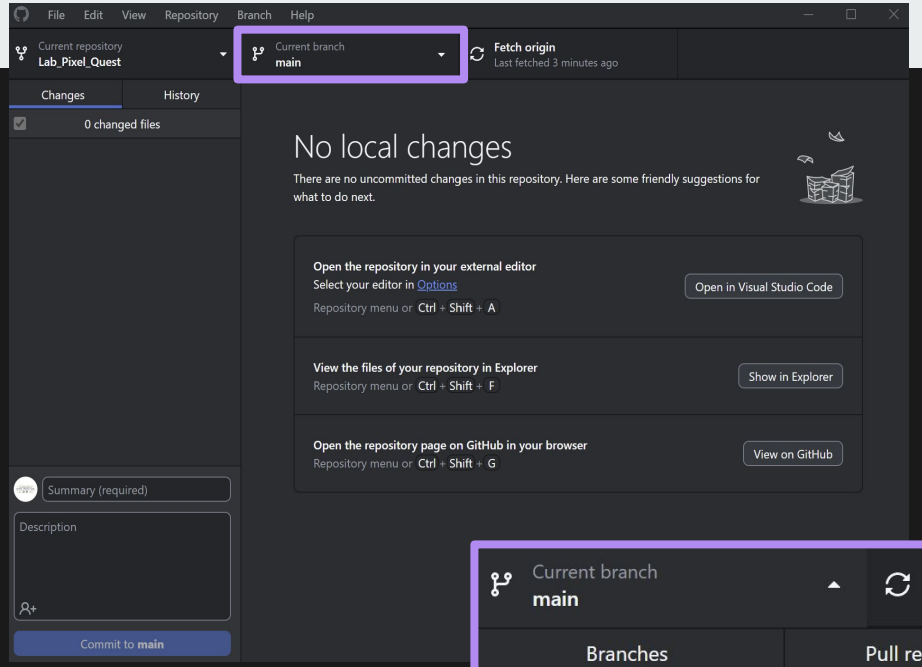


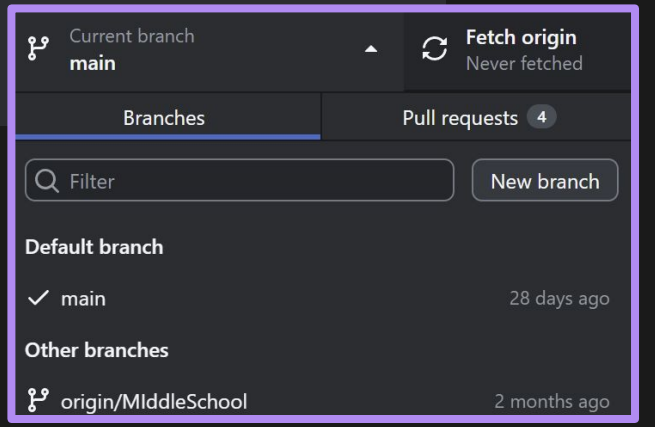
Branches

If you're using the class account, you may need to switch to a branch to see the most up-to-date version of the project.

- **Branches:** Think of branches as separate folders with different setups. When working on a team project, each person has their own branch where they can make changes independently. Later, those changes can be merged using GitHub's merge function.
- **Current Branch Button:** Clicking this button shows the branch you're currently working on and lets you switch to another branch.
- **Drop-down Menu:** This menu shows all the available branches. You can select the one you need to switch to.

Important Note: If you switch branches while Unity Editor is open, it will reload the project files and may cause issues or break the project. Make sure to save your work before switching branches.





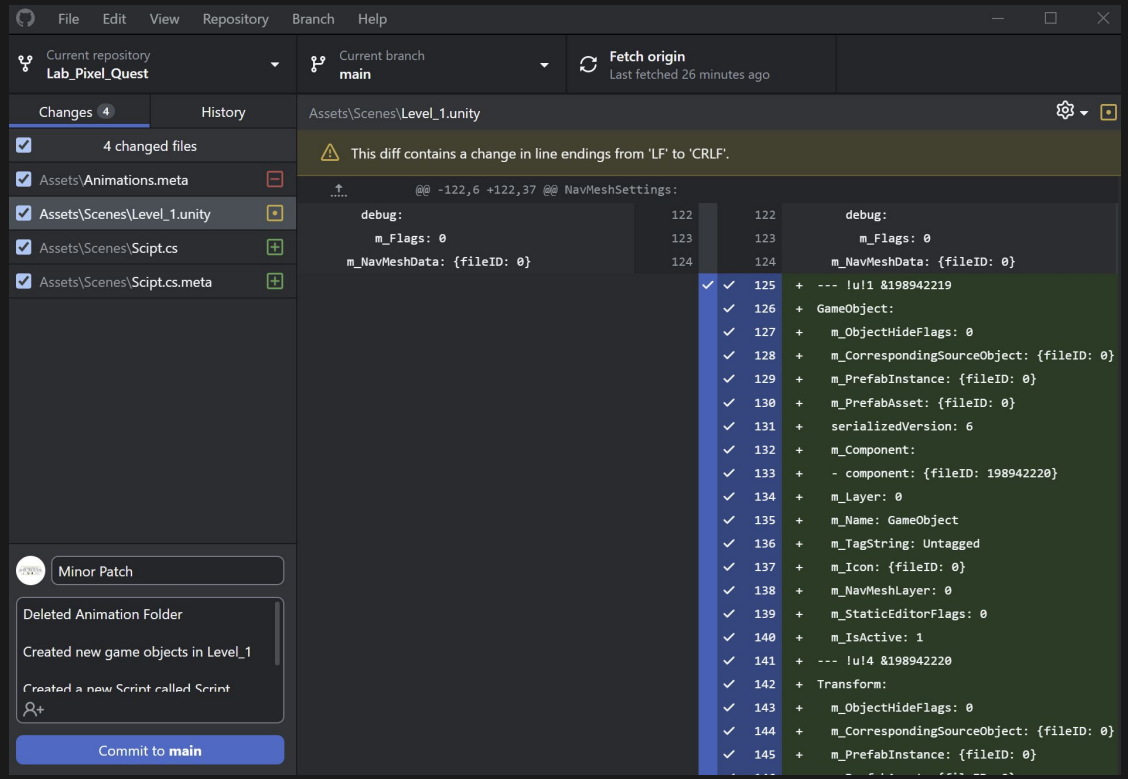
Sebastian Grygorczuk - STEM Institute at CCNY

Changes

When you make changes to your project, GitHub Desktop will show those changes in the **Changes** tab.

- **Red:** Indicates a file has been deleted.
- **Yellow:** Shows that a file has been modified.
- **Green:** Means a new file has been added to the project.

You can click on any file to see a detailed breakdown of the changes. The view will show you the differences between the file before and after your edits, with a split view for easier comparison.



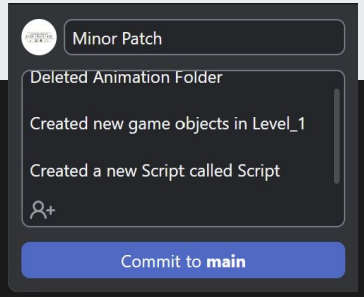
Commit & Push

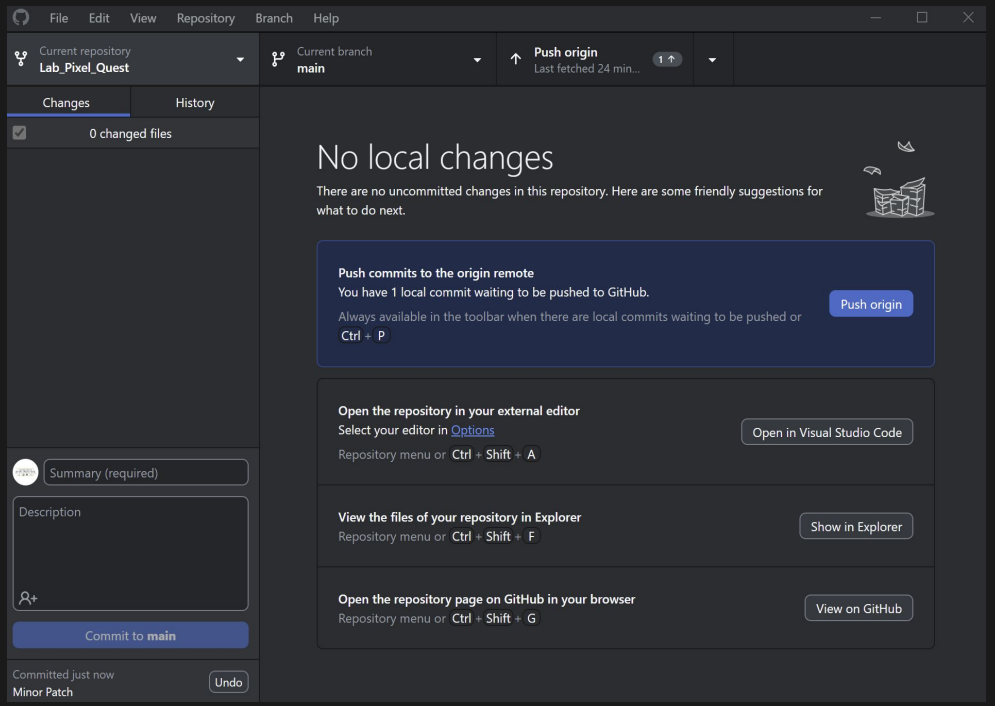
Once you've finished making changes, you need to save them online. All changes are initially saved locally on your machine, so you'll need to **commit** them. A commit is essentially a snapshot of the project at that specific point in time.

- **Commit often:** It's best to commit smaller changes frequently. This helps you pinpoint and revert back to earlier versions if any problems arise.
- **Write meaningful commit messages:** Clearly describe the changes made in each commit. This will help you (and others) understand what has been done when reviewing the commit history.

After committing, you'll need to **push** your changes to the online repository. This allows you to access your project from any other computer.

In GitHub Desktop, you'll find buttons to help you push your commits to the online repository and save your work.





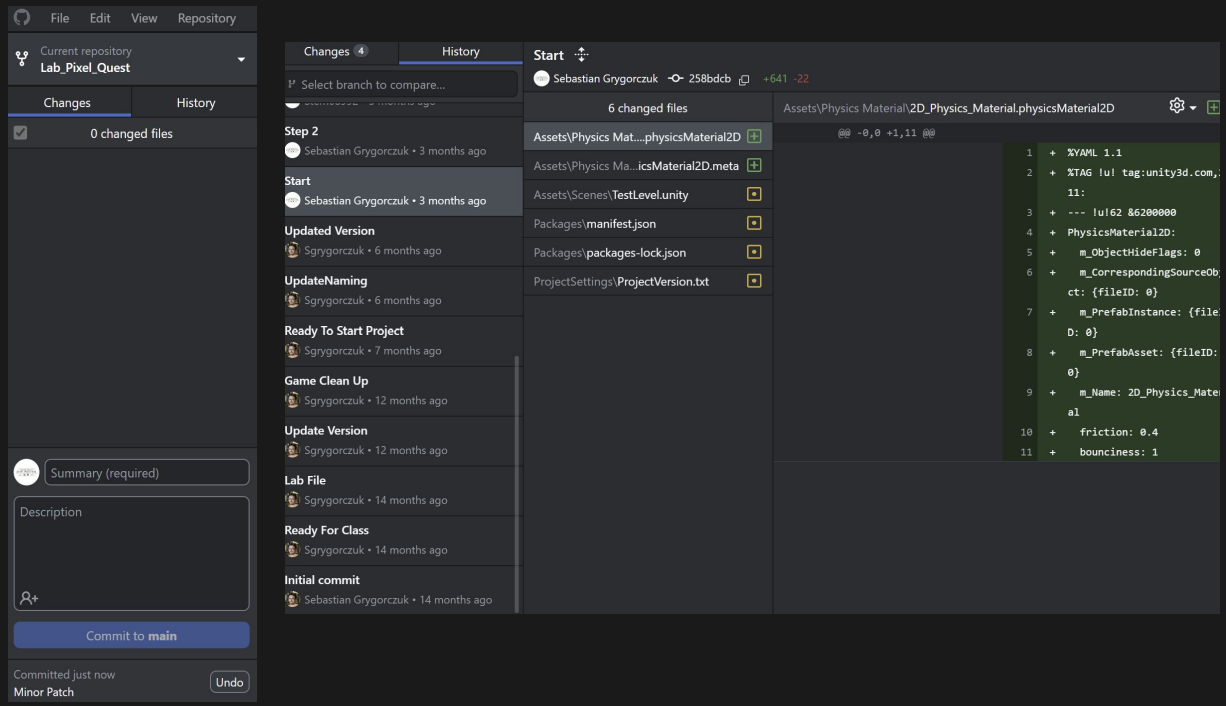
Sebastian Grygorczuk - STEM Institute at CCNY

Undo Commit - History

Sometimes, after committing changes and testing your game, you might encounter an unexpected bug caused by those changes. Fortunately, you can undo recent commits and revert the project to its previous state.

- **Use the History tab:** The History tab shows all your past commits along with the changes made in each one. By reviewing the commit history, you can identify the specific commit that caused the issue and revert to a previous version of the project.

This allows you to fix issues without losing your entire progress.

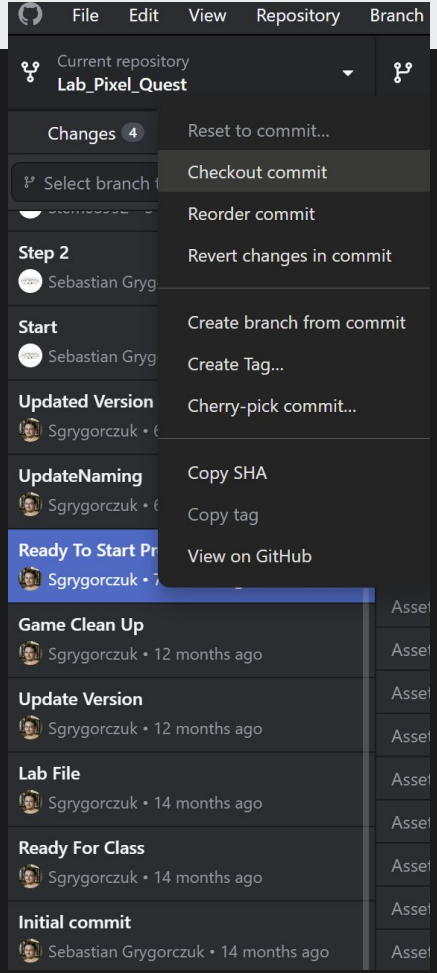


Reverting Commits

If you right-click on a commit in the **History** tab, you'll see several actions you can take:

- **Checkout Commit:** Allows you to view the project files as they were at that specific commit.
- **Revert Changes:** Reverts the changes from that commit, but this might lead to additional unforeseen issues.
- **Create a Branch:** Lets you create a new branch to safely investigate and address the changes in an isolated environment.

In any case, these actions will modify the project files to match the selected commit.
If you have unsaved changes in your current project, a pop-up will appear asking if you want to **stash** those changes to avoid losing them.



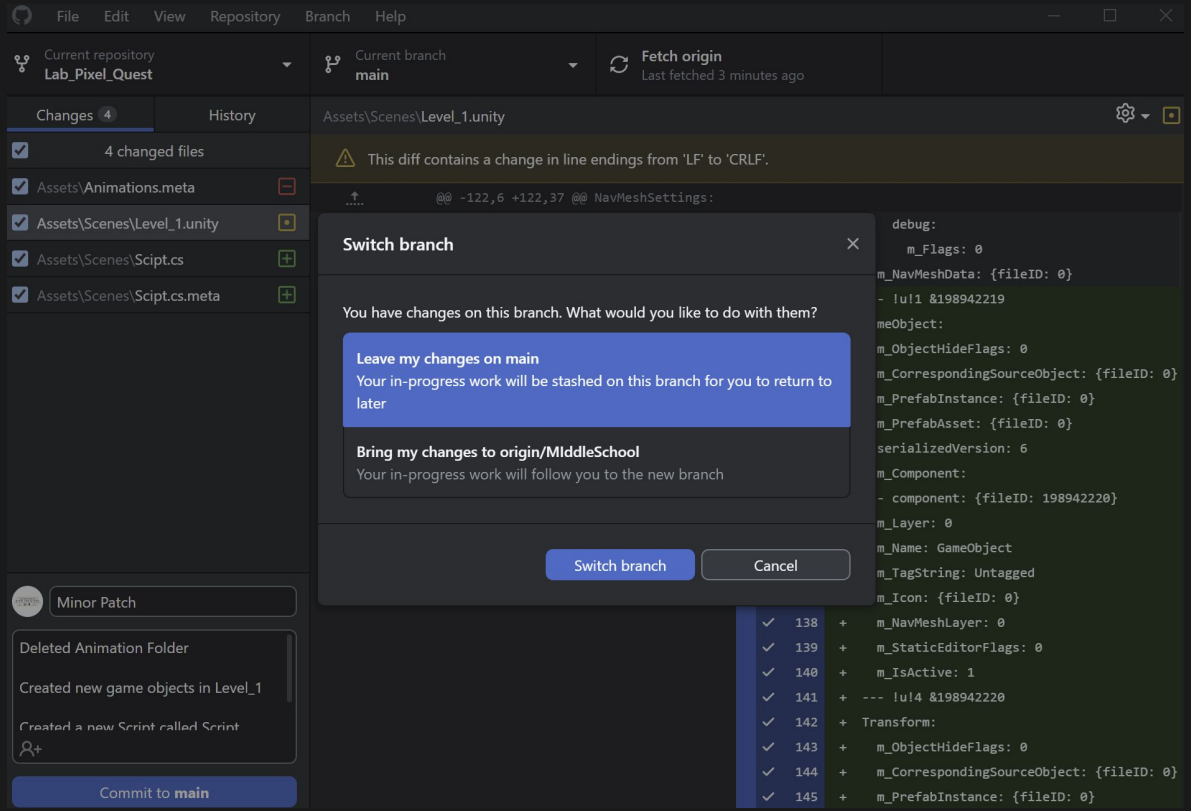
Stashing

Stashing lets you save changes locally without committing them, but you can only stash one set of changes at a time.

When faced with unsaved changes, you have three options:

1. **Commit the changes** before switching branches.
2. **Stash the changes** and retrieve them later when you return to this branch.
3. **Bring the changes** to the new branch you're switching to.

To avoid losing progress or creating conflicts, it's best to either commit or stash your changes before switching branches.



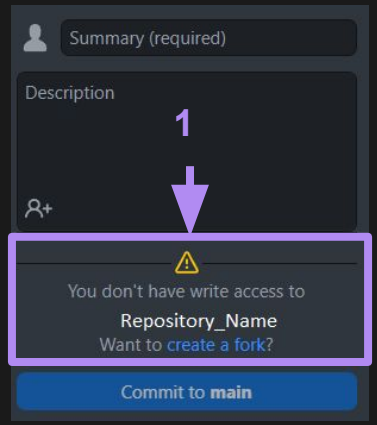
Forking Your Own Copy

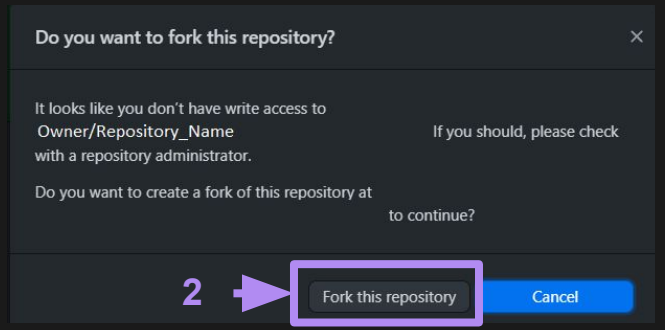
Since the project files you downloaded are mine, you won't be able to commit changes directly to my version.

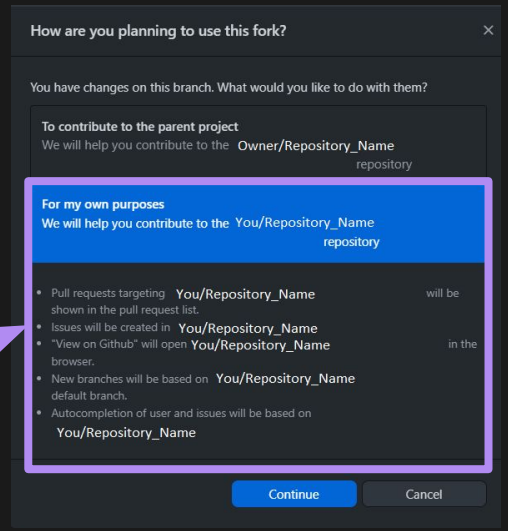
To create your own version of the project, we will **Fork** it. GitHub Desktop will show a warning during the commit process that you don't have access to the repository, suggesting that you fork it.

1. Click **Create a Fork.**
2. In the pop-up menu, select **Fork this Repository.**
3. Choose **For My Own Purposes.**

This will create a separate copy of the project, which only you can edit. The other option would fork directly from my branch.





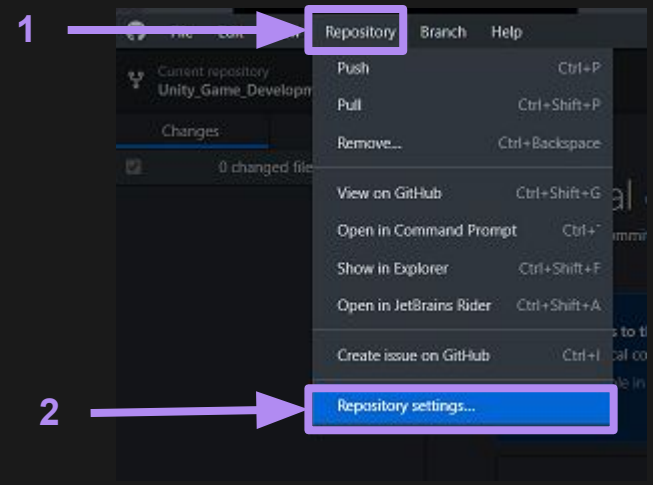


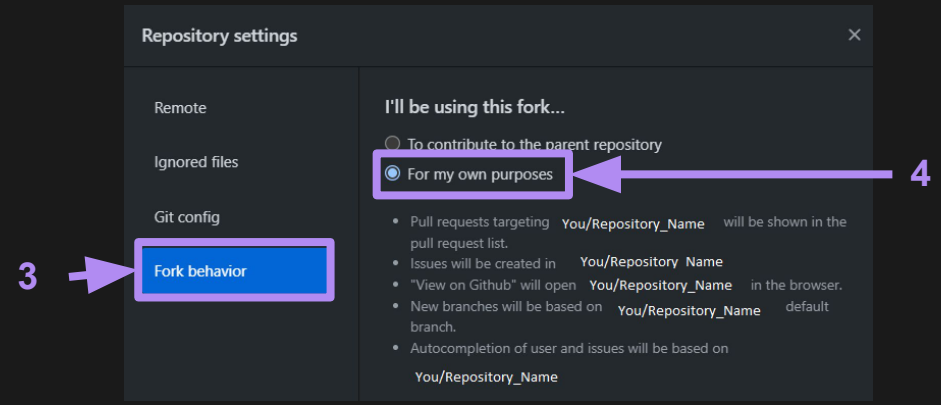
Forking Your Own Copy

If the warning doesn't appear in the commit area, you can manually set it up by:

1. Going to the **Toolbar**, selecting **Repository**.
2. Choose **Repository Settings**.
3. In the pop-up, go to **Fork Behavior**.
4. Select **For My Own Purposes**, then click **Save**.

This will allow you to create your own copy of the repository and commit and push changes without any issues, just like the warning would prompt you to





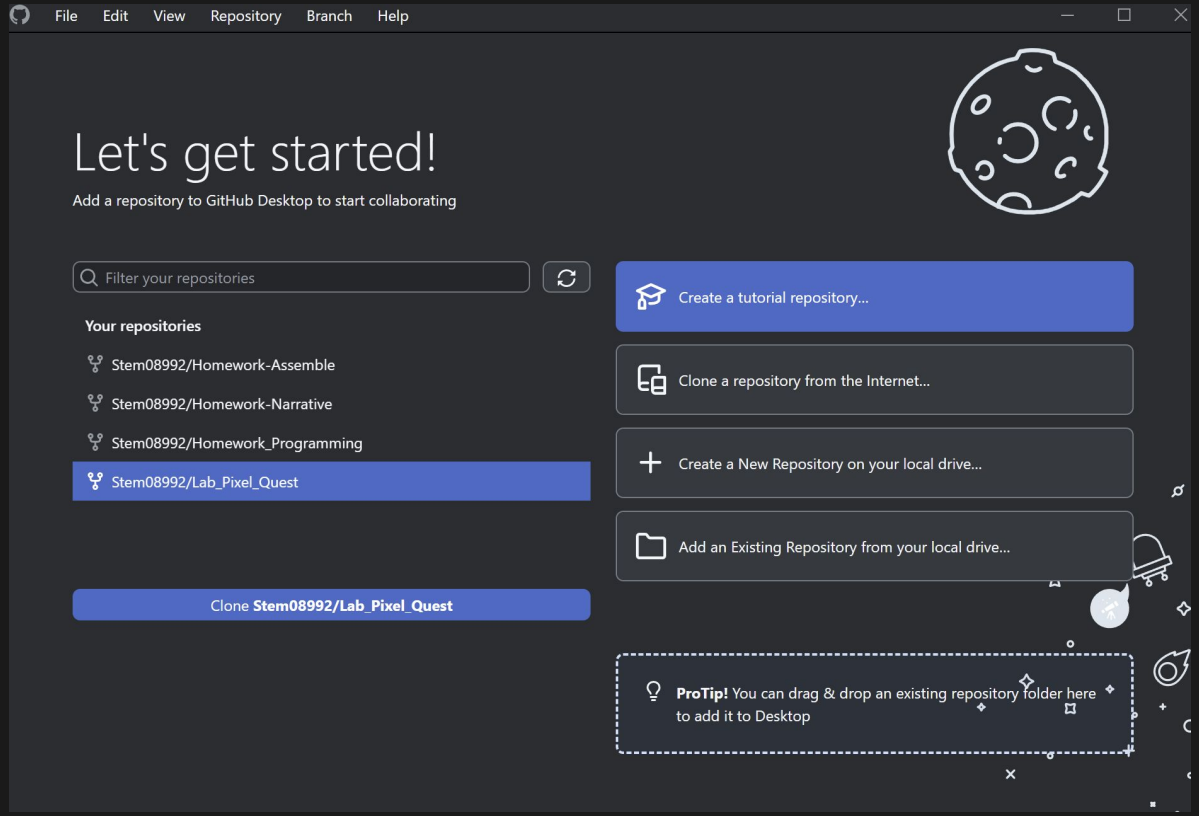
Cloning on New Computers

Since the school computers are wiped after each use, you'll need to re-login to all software.

However, once you've forked a project, it becomes part of your GitHub account.

After that, you no longer need to clone it using the URL link.

The project will appear on the left side in GitHub Desktop—just click on it, and the information will auto-fill for cloning.

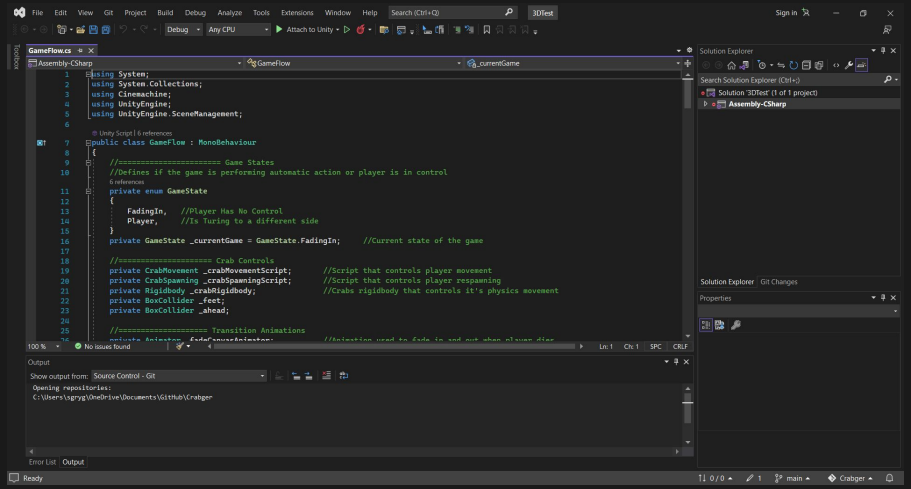


Visual Studio

Visual Studio

What is an Integrated Development Environment (IDE)?

An Integrated Development Environment (IDE) is software that helps programmers write, edit, and debug code. It’s essentially a powerful text editor. With Visual Studio, you'll write C# scripts to bring your Game Objects to life in Unity.

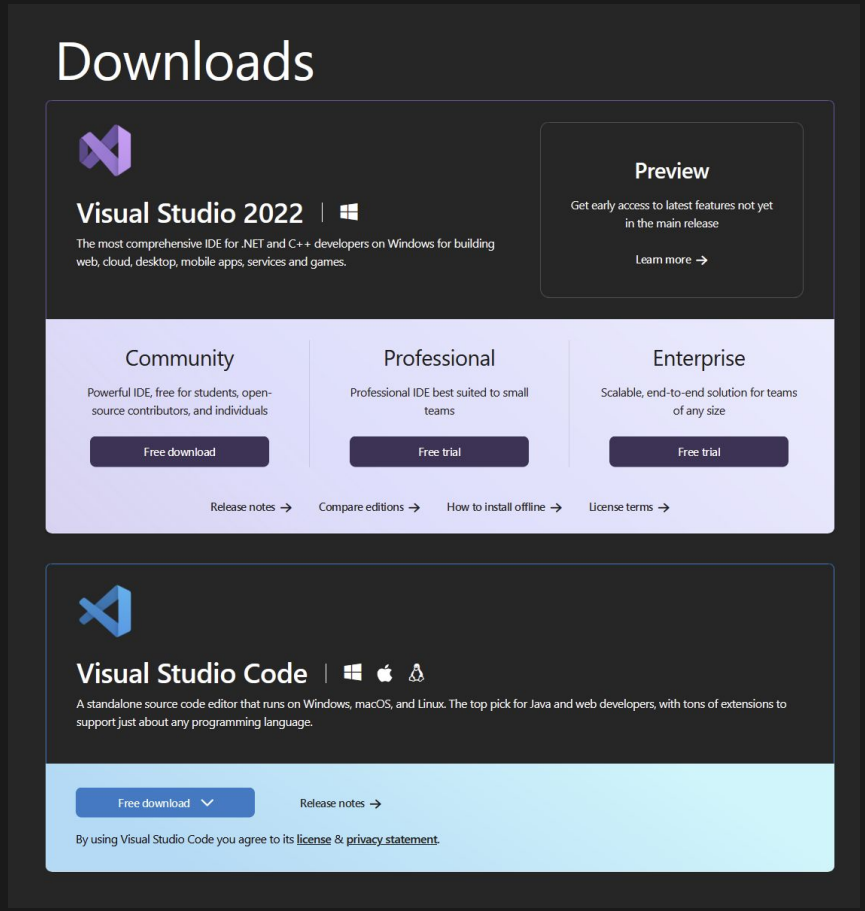


Downloading Visual Studio

Head over to download: https://visualstudio.microsoft.com/downloads/

For Windows: Download Visual Studio 2022 Community Edition. This will install the Visual Studio Installer, which manages versions and add-ons.

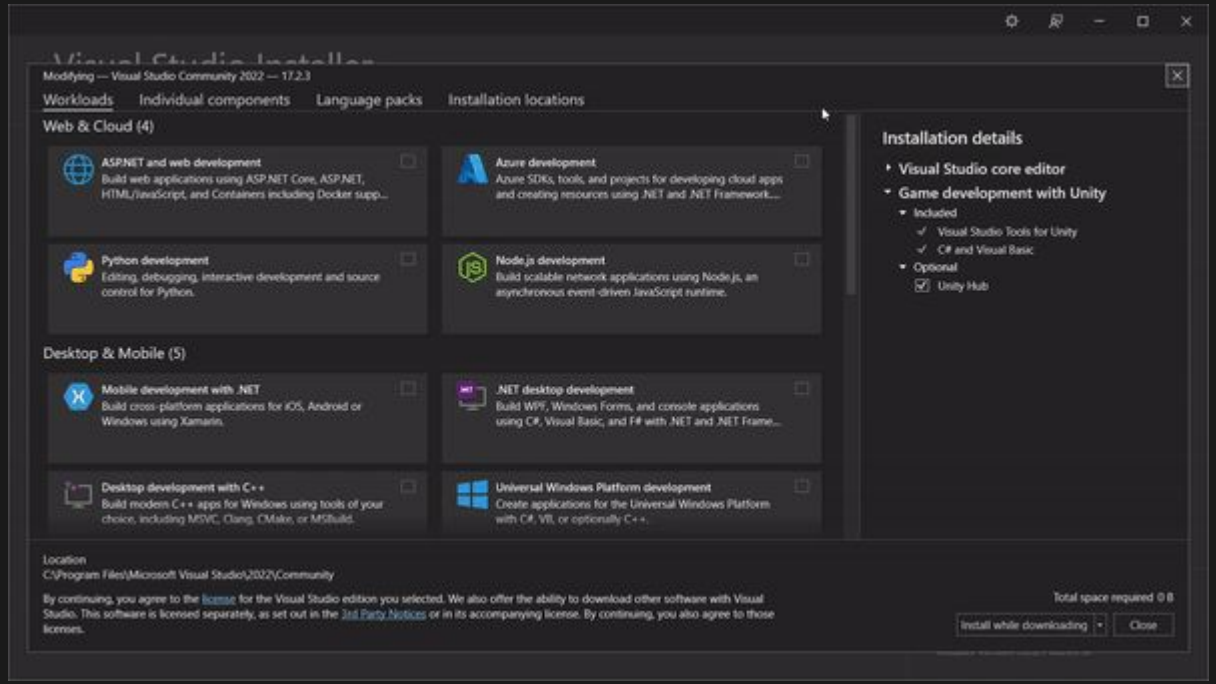
For Mac: Download Visual Studio Code instead.



Visual Studio Hub - Workload Installs

During installation, the Visual Studio Installer will prompt you to select workloads to download.

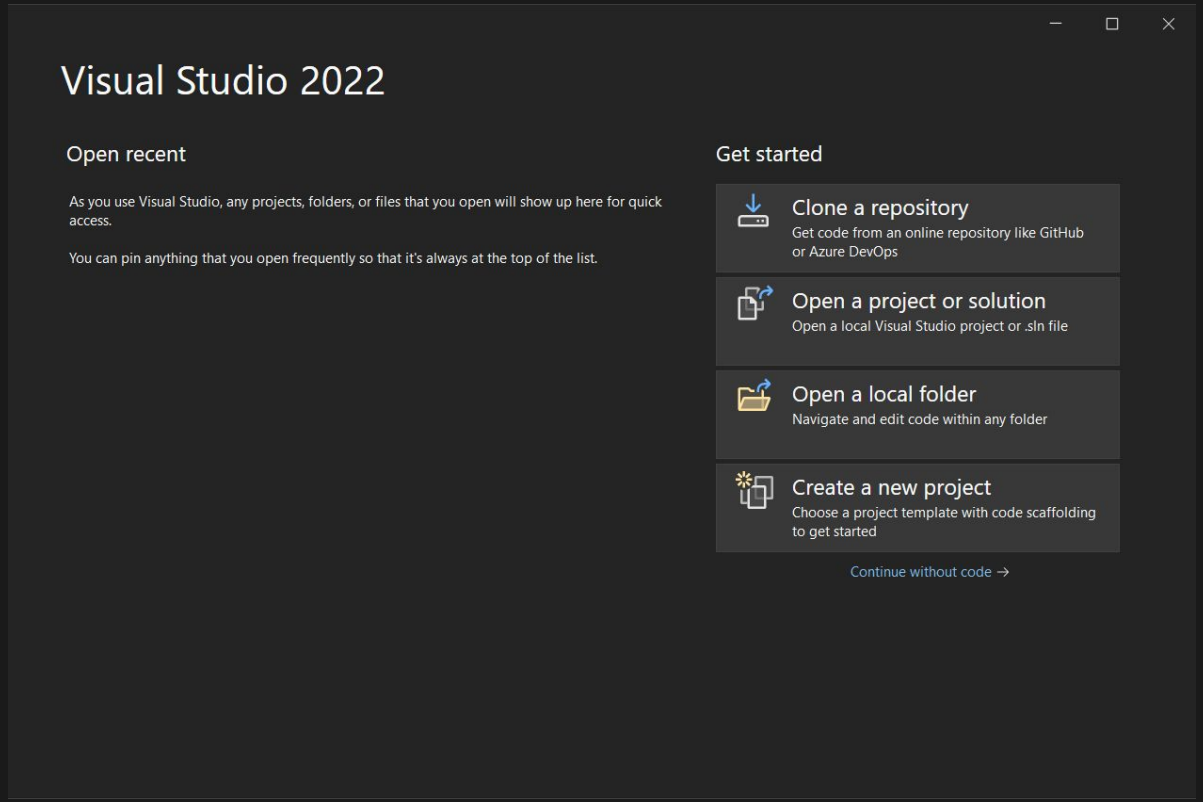
Scroll down and check the "Game Development with Unity" option. This ensures smooth integration between Unity and C# libraries, which is essential for writing scripts in Unity.



Visual Studio Community

Once you finish the installation, you will be met with this screen where you can select a script.

For now, we'll stop here, as we will become much more familiar with it once we get into our Introduction to C# lesson.

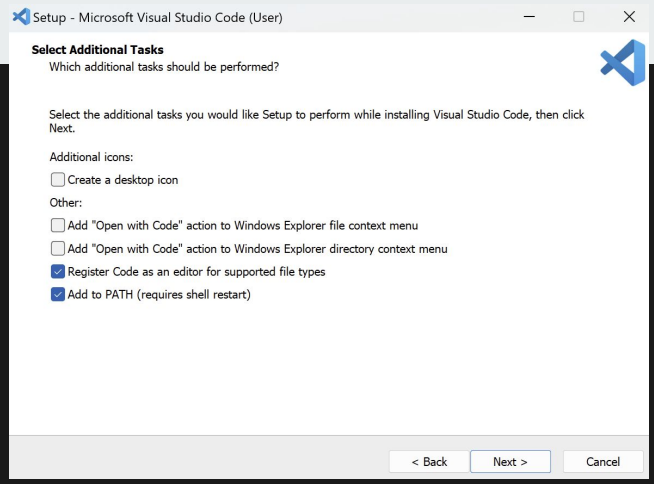


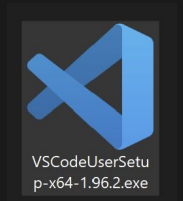
Visual Studio Code

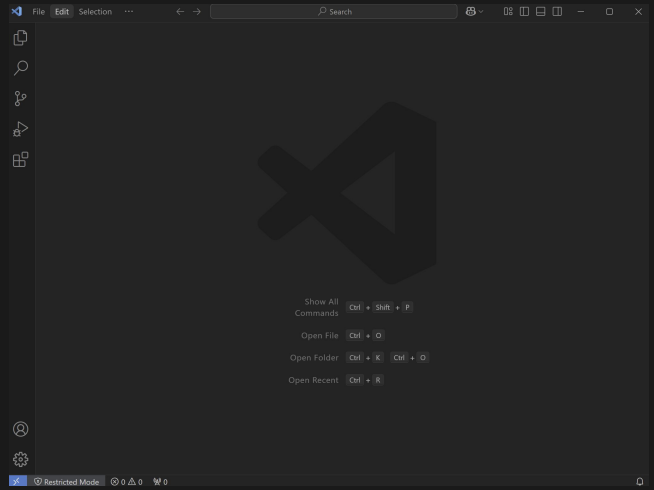
When downloading Visual Studio Code, it will ask you for the settings you'd like.

Simply use the default setup it provides, as shown in the picture.

There is no separate installer for Visual Studio Code, so it will open immediately after the installation is complete.





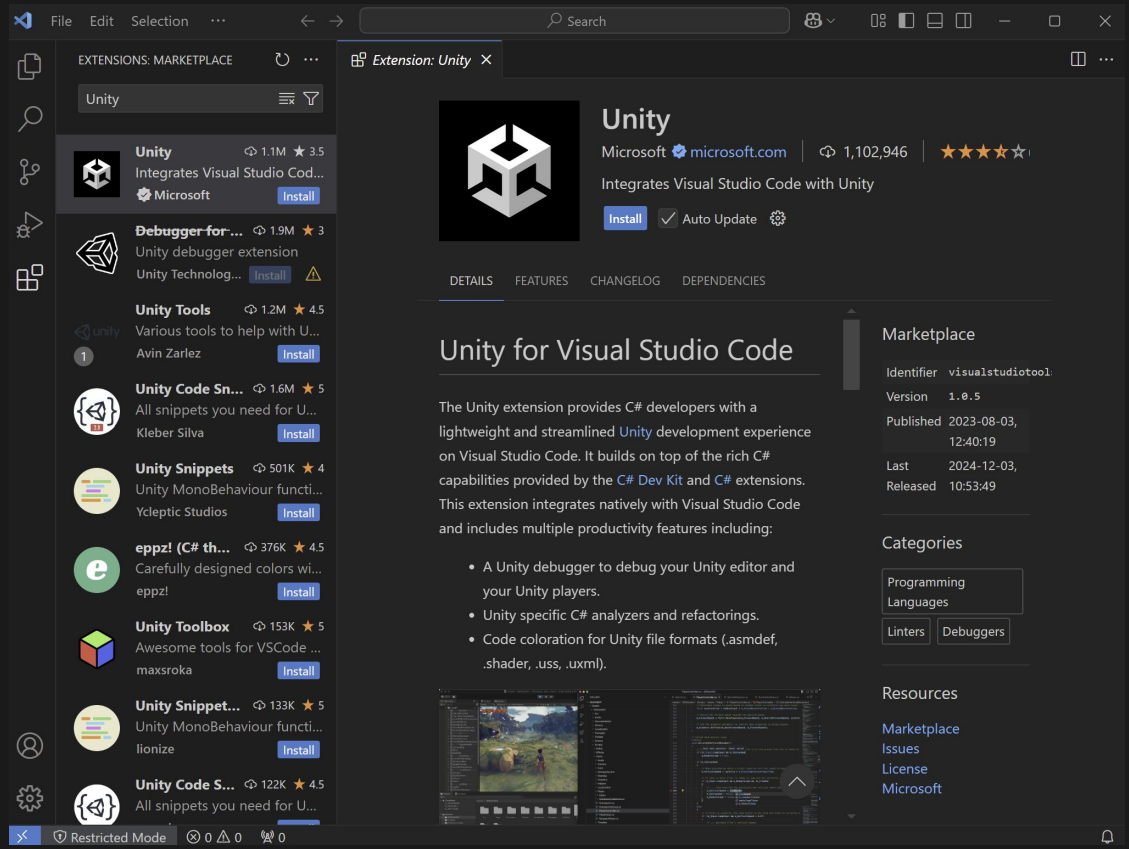


Unity Extension

To have Visual Studio Code work with Unity, you will need to go to the Extensions tab.

Type in "Unity" and then install the Unity Extension.

Once this is installed, you should be able to interact with Unity code without any issues.



Unity

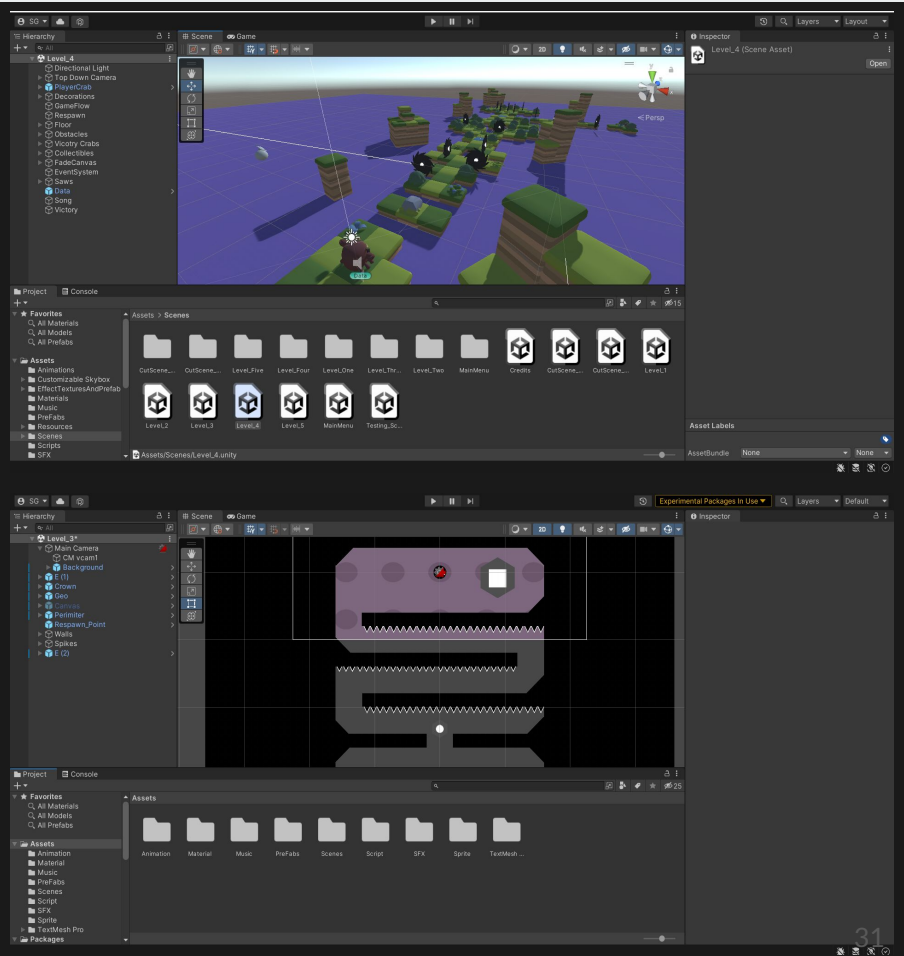
Unity

What is a Game Engine?

A game engine is software that comes with built-in libraries and features to help you create games.

It includes things like physics, image processing, and sound, which are already set up for you.

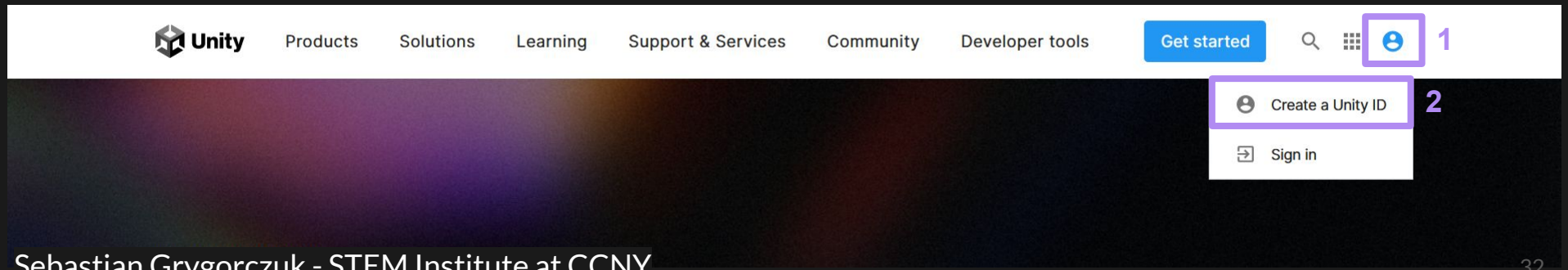
Instead of programming everything from scratch, you just need to make the connections between these features to build your game.



Making A Unity Account

Before we install any software, we'll first create an account with Unity. This account gives you access to the Unity Asset Store, where you can download 3D and 2D assets to use in your projects during this class and potentially for your final project.

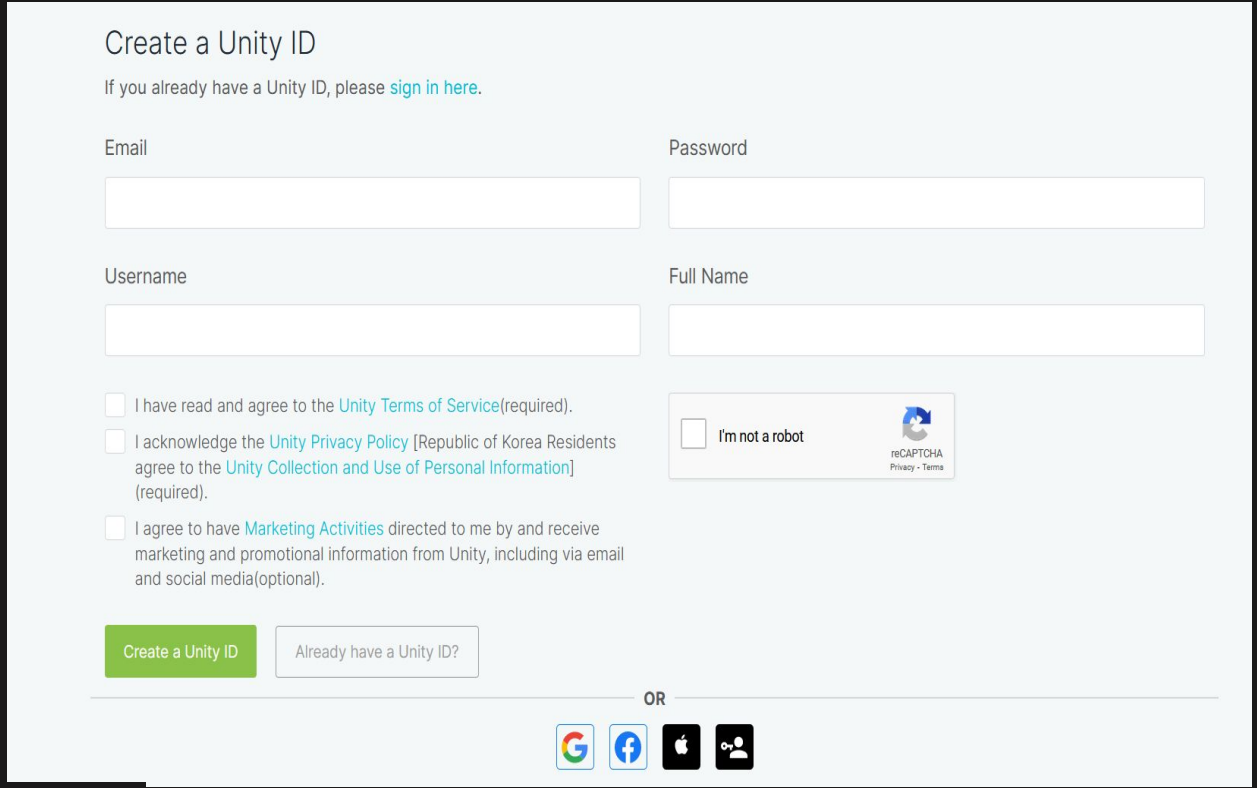
Head over to https://unity.com/ and click on the avatar in the top right corner. Then, click "Create a Unity ID" to start the process.



Making A Unity Account

Fill out the necessary information or use an existing account to connect to Unity.

You may need access to your email or phone for verification when signing in, so ensure you use credentials you have access to.



Installing Unity

- To access Unity, you will first need to download Unity Hub.
- Unity Hub is a central location to manage your projects and Unity versions. As Unity releases new versions with updated features, Unity Hub will help you easily manage them.
- To begin the download, go to https://unity.com/download#how-get-started.
- Scroll down a bit to find the box that allows you to download Unity Hub for your specific platform.

Signing Into Unity Hub

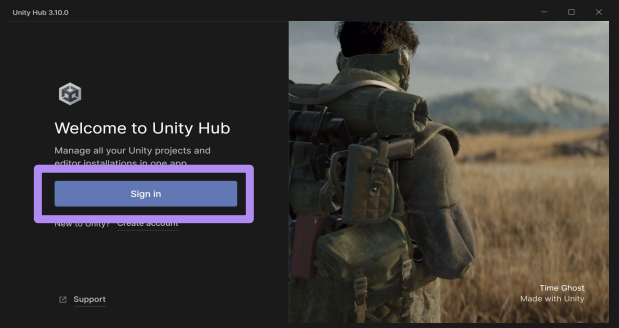
When you download Unity Hub, you will need to sign in.

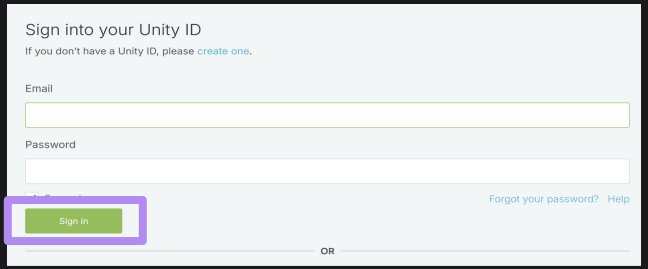
Click on the 'Sign In' button, and you'll be redirected to a web browser.

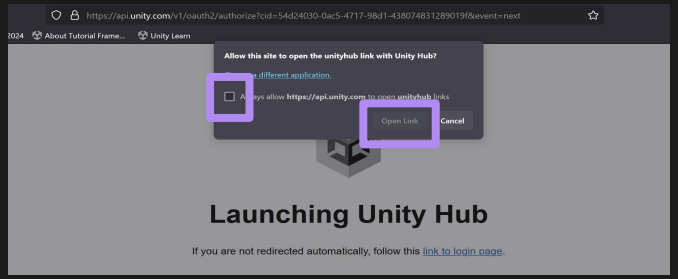
Enter the account information you've created and click 'Sign In.'

This will bring you back to Unity Hub.

A pop-up will appear—click 'Open Link,' and your account should be connected, giving you access to Unity Hub.







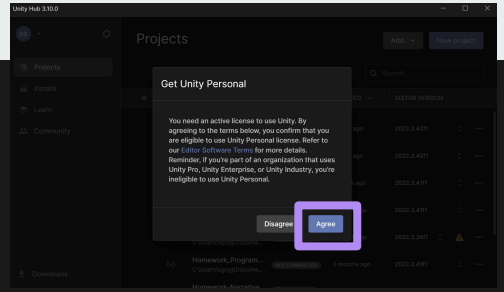
Sebastian Grygorczuk - STEM Institute at CCNY

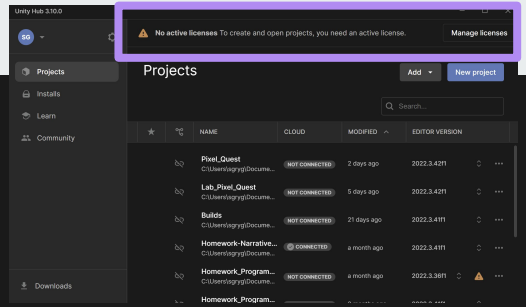
Unity License

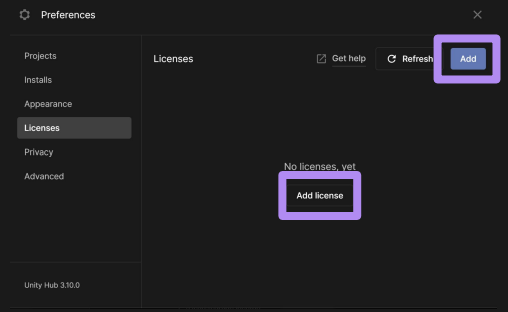
When you log into Unity for the first time on a computer, you'll need to choose what kind of license your account is operating under. A pop-up will appear—click **'Agree'**, and this will give you a **Personal License**, allowing you to use Unity for free.

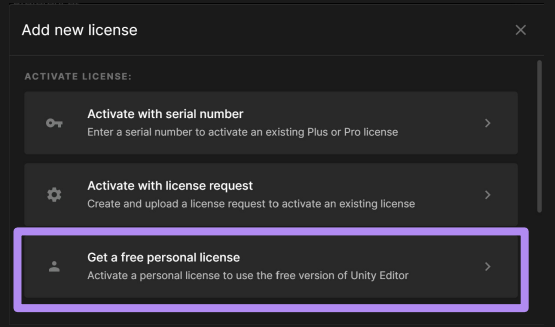
If you click **'Disagree'**, a warning banner will appear at the top of Unity. In this case:

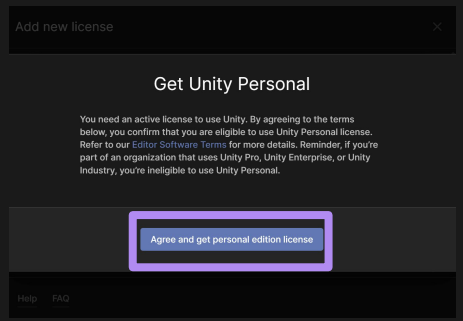
1. Click **'Manage Licenses'**.
2. This will open a new window—click **'Add'** or **'Add Licenses'**.
3. Then select **'Get a free Personal License'**.
4. Agree to the license, and you'll have full access to Unity.











Sebastian Grygorczuk - STEM Institute at CCNY

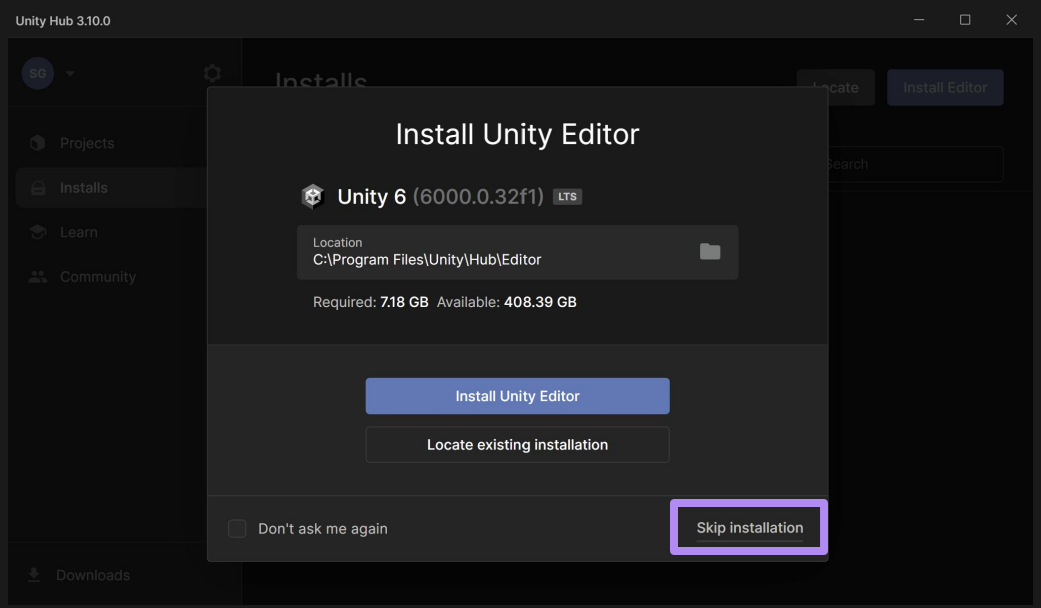
Installing Unity Editor - Skip Newest Version

After accepting the license, a second pop-up will appear, prompting you to install a version of the Unity Editor.

Unity updates frequently, but because of this, we cannot maintain the computers with the most up-to-date version.

Instead, we'll be using a specific version that matches the project files provided.

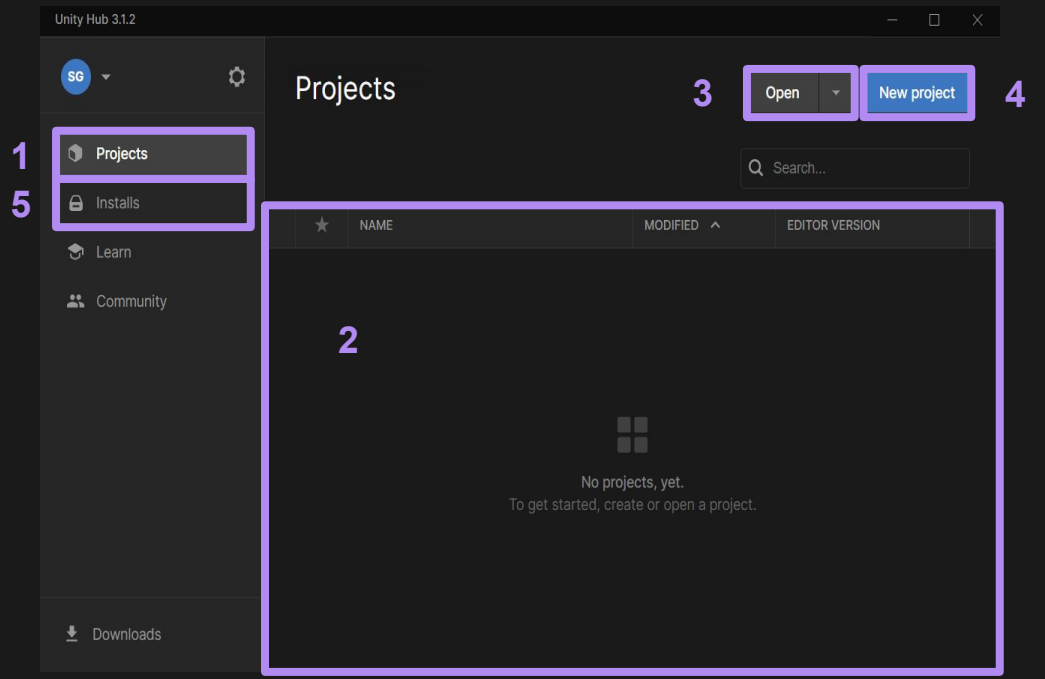
When this pop-up appears, **click 'Skip Installation'** to proceed.



Unity Hub

When you launch Unity Hub, you'll be presented with a clean interface that lets you manage your projects and Unity versions. Here's a breakdown of the key sections:

1. **Projects Tab:** This section displays all of the projects you've recently worked on, so you can easily pick up where you left off.
2. **Project Selection Area:** This is where you'll select the project you want to work on from the available list of projects.
3. **Open Project:** If you have projects that aren't displayed in the main list, you can use this option to browse for and open them directly from your computer.
4. **Create a New Project:** This button allows you to start a brand-new project. It's the gateway to setting up a fresh game or application in Unity.
5. **Install Tab:** Here, you can manage the versions of Unity that are installed on your computer. If you need a different version of Unity, you can add it here for use.

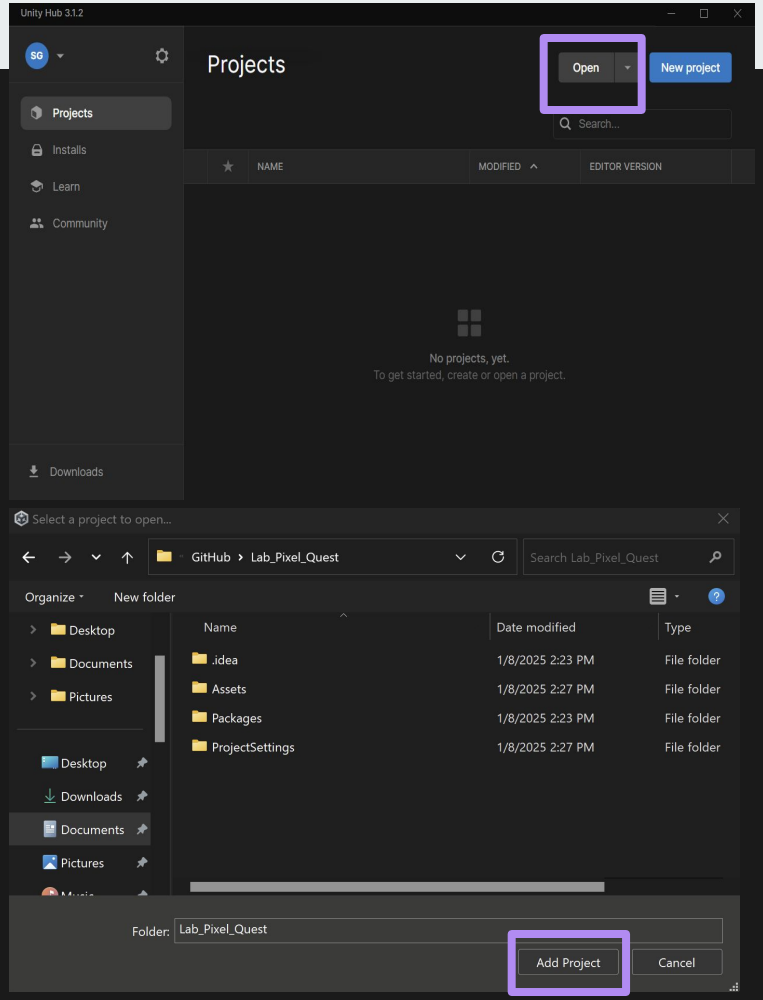


Import the Project Files

We will start by importing the files cloned using GitHub Desktop.

1. Click on **Add** and select **Add Project from Disk**.
2. This will open the File Explorer. Navigate to Documents/GitHub/Lab_Pixel_Quest.
3. Ensure the folder contains .idea, Assets, Packages, and ProjectSettings. If you see these, you're in the right place.
4. Don’t select any specific folder—just click **Add Project**.

This will add the project to your Unity Hub selection window.

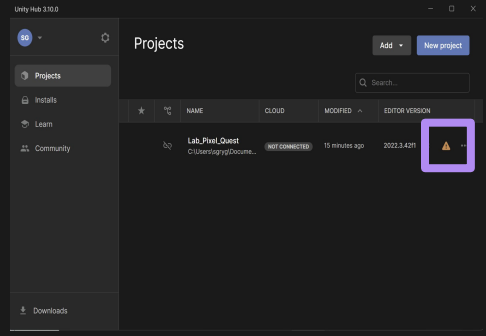


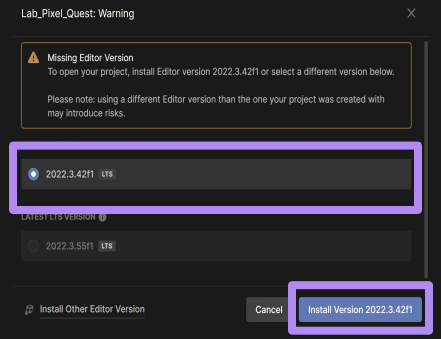
Installing Unity Editor

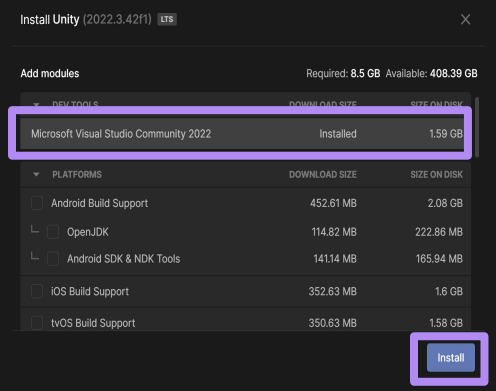
Now the project should appear in your list of projects, but you might see a yellow warning triangle. This indicates that the version of Unity used to set up the project is missing from Unity Hub.

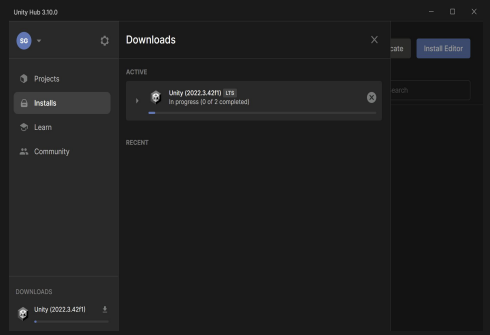
1. Click on the warning triangle, and a pop-up will appear showing the available versions for download.
    - The **Missing Version** will be listed at the top, and the latest version will appear below.
    - Make sure to download the **Missing Version** (e.g., 2022.3.42f1).
2. Once selected, another pop-up will appear asking which additional features to include with this Editor.
    - Ensure Visual Studio is already installed before proceeding.
    - Select **Install** and let the installation process complete.

Note: The Unity Editor is a large file (~7–8GB), so the download and installation time may vary depending on your internet speed.









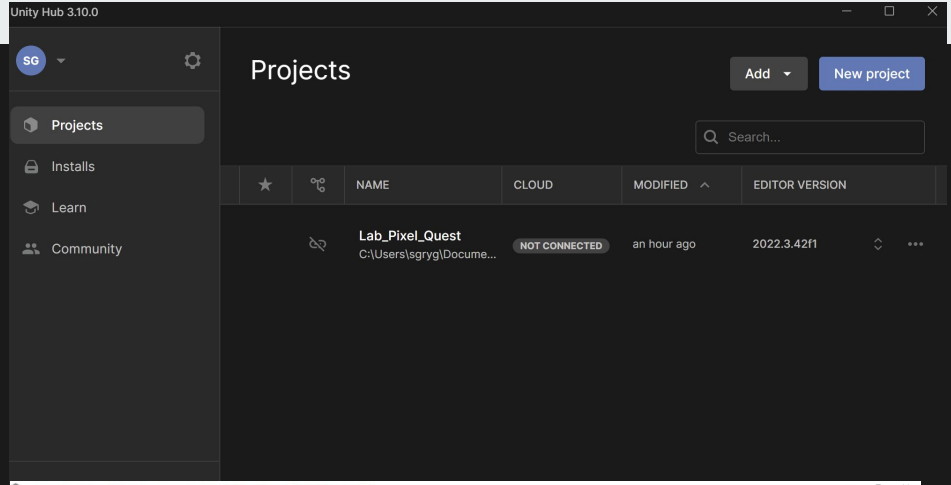
Sebastian Grygorczuk - STEM Institute at CCNY

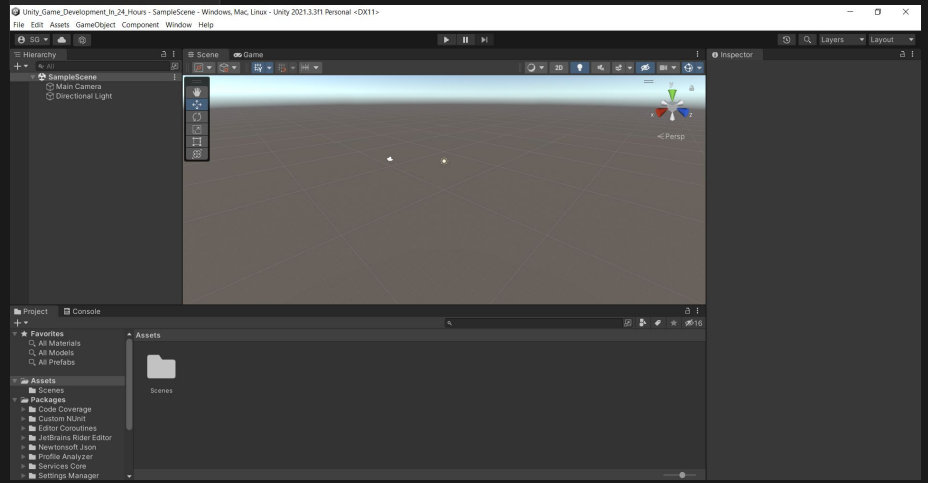
Loading Project

Once the Unity Editor has been successfully downloaded, the yellow warning triangle will be replaced with the version number. Click on the project name to open it.

Be patient, as the first time you open a project, Unity needs to configure some settings, which can take a little while. However, future loading times will be much quicker.

Once everything is set up, the Unity Editor will open, and you'll be ready to start creating your game.





Version Control System

GitHub - Version Control System

A Version Control System (VCS) is a tool that helps manage and track changes in a project's files, whether you're working alone or with a team.

Think of it like a Google Doc: VCS lets multiple people edit the same project at the same time while recording all the changes. It saves the history of those changes, and if something goes wrong, you can easily roll back to a previous version.

VCS also helps when two or more people edit the same part of the project. If there's a conflict, it highlights the differences so you can compare the versions and decide which one to keep.

[1]

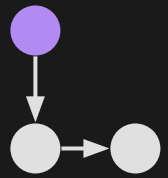
The Project Repository: Imagine the project files we download for the class are represented by a purple dot. This is the original repository.

[2]

Cloning the Project: You clone this repository to your personal computer (represented by a white dot), creating an independent copy.

Commit & Push

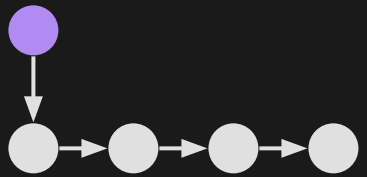
[3]



Making Changes Locally: After making changes on your local copy, you create a *commit*.

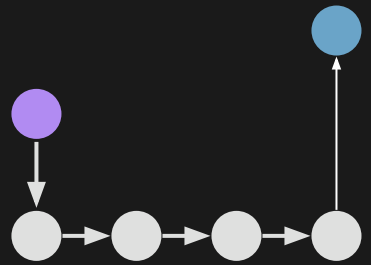
A commit is essentially a snapshot of your changes, saved locally.

[4]



You can create as many commits as you want without affecting the online repository.

[5]



Pushing the Project: When you're ready, you *push* your changes to GitHub.

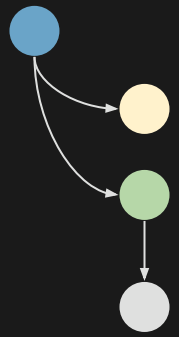
However, because the original repository wasn't yours, you need to *fork* it first.

Forking creates a copy of the repository under your account.

Now, any future commits and pushes will go to your version of the repository.

Branch and Merge

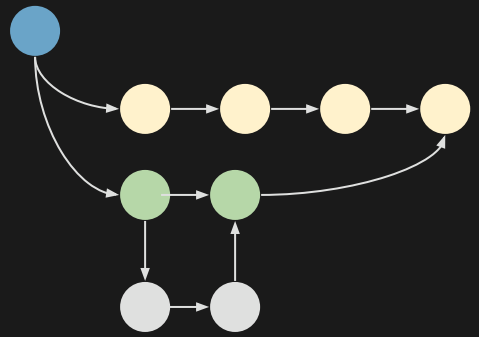
[6]



Working with a Teammate: If you're working with a teammate, you both create separate branches.

This allows each person to make their own changes without affecting the other person's work.

[7]



When both of you are ready, you'll *merge* the changes.

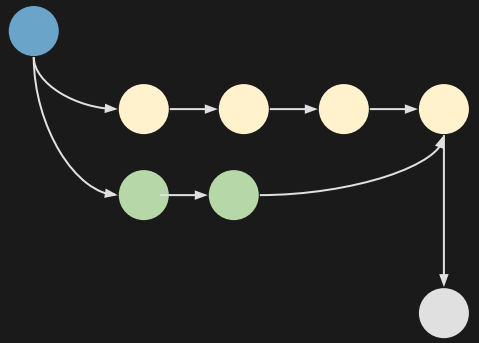
GitHub compares the differences between the two branches and applies any changes.

If both of you edited the same file, GitHub will flag this as a *merge conflict*.

Depending on the file type, you may need to manually choose which version of the changes to keep.

Pull

[8]



Pulling Changes: After merging online, you'll need to *pull* the changes to bring your local project up to date.

This ensures your local copy has the combined updates from both branches.

Here's a quick review of the vocabulary we just went through.

Clone: Copying/downloading the repository to create a local copy.

Fork: Creating your own copy of a project from another user's repository for personal use.

Commit: Creating a snapshot of your local project's current state.

Push: Uploading your local commits to the cloud (GitHub).

Branch: A copy of the project in a separate repository.

Merge: Combining the changes from different branches.

Pull: Bringing updates from the online repository to your local machine.