Introduction

[Projects](https://www.pluralsight.com/product/projects) are a new hands-on way to learn on Pluralsight. They follow real-world practices and workflows which may be new if you’ve never worked with a certain technology before.

The flow of every project follows the same structure:

1. Learn what you’ll build in this project.
2. Setup and install all dependencies on your local computer.
3. Download the code for this project (by forking and cloning it from GitHub).
4. Get this project working on your local computer.
5. Complete some number of code tasks.
6. Upload your code (by pushing it to GitHub).
7. Pluralsight checks your work and verifies everything is correct! If it’s not, we’ll also give you feedback about what needs to be fixed to make each code task correct.

One of the common themes for all Pluralsight Projects is the use of Git and GitHub, which is a requirement for all projects. This guide walks you through everything you need to know to use Git and GitHub in order to complete a Pluralsight Project.

What are Git and GitHub?

Before going too far, let’s explain what these are! Git is a version control system (VCS), similar to Subversion, CVS, Team Foundation Version Control, Mercurial, Harvest and other VCS’s.

If you’ve never used a version control system before directly, I bet you’ve used them indirectly. Time Machine for macOS and Google Docs version history are two widely used applications of version control. In both cases, and with Git, you have a current version and the ability to go back in time and see previous versions. That’s the power of Git - being able to save multiple copies of a file and track history over time.

Where Does GitHub Come In?

Having a bunch of past versions of code on your local computer will only get you so far. Eventually you’ll want to collaborate with other people - maybe building on someone else’s code, collaborating with another developer, or even just getting feedback in a code review.

GitHub is a platform that allows you to upload your code and allows others (if granted permission) the ability to contribute changes back to it.

The end result of this is that you can have multiple people working out of the same codebase without them constantly deleting each other’s work. Version control systems are a tremendously powerful tool!

Installing And Configuring Git

In order to work with Git, you’ll need to install it locally (on your computer). You can grab the latest version from the official site for Git, [git-scm.com](https://git-scm.com/).

Once git is installed locally, you should be able to run the gitcommand from your terminal (macOS, Linux) or command line (Windows).

You’ll want to set two configuration settings right away - your name and your email. Every time you make a code change, that change will be stamped with this user information, so make sure you’re setting this correctly.

1

2

$ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com

bash

Now whenever you make a commit to git, it’ll be associated with your name and email!

Creating a GitHub Account

Creating a GitHub account is as simple as visiting [github.com](https://github.com/) and finding the *Sign up* link at the top of the page. On that page, you’ll be able to set a username, email, and password, and that’s it! Keep that information handy, because you’ll need it in the next step. You won’t need more than a free account, so don’t worry about signing up for a paid plan unless you have a need for those features.

Authenticating with GitHub

For Pluralsight Projects to work, we need to connect your Pluralsight account to your GitHub account - where your code will be for this project. The first step to doing this is to authenticate with GitHub.

Pluralsight isn’t granted any private permissions as part of this. If you have private repositories, we can’t see them, or repositories of the companies you work for. We also can’t make changes to your code.

What this permission does is allow us to know that you are you, and give you credit for working on the Project within Pluralsight.

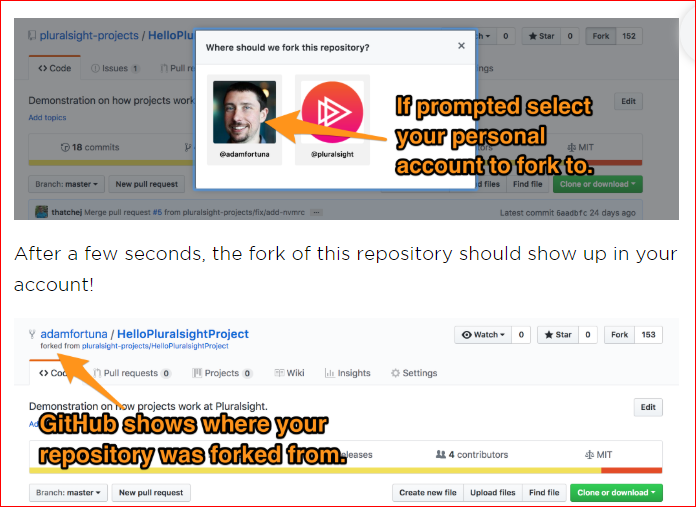
Forking a Repository on GitHub

When you start working on a Project, you’ll be asked to fork a Project repository that already exists at one of Pluralsight’s organizations on GitHub. “Forking a repository” means making a copy of someone else’s repository (Pluralsight’s) into your own account on GitHub. You can think of a “repository” as a folder of code annotated with all the version history with it.

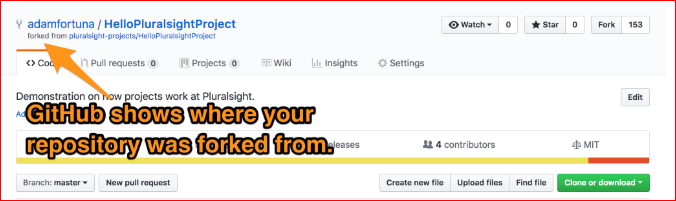
To fork a repository, sign into GitHub and visit the original webpage for the repository on GitHub (not Pluralsight). For example, if the Project asked you to fork a repository at the URL [GitHub - pluralsight-projects/HelloPluralsightProject: Demonstration on how projects work at Pluralsight.](https://github.com/pluralsight-projects/HelloPluralsightProject), then you would open that up in a new browser tab or window and click the “Fork” button.



If you are a member of any organizations (companies) on GitHub, you may be prompted with a question asking where to fork this repository to. All Pluralsight Projects assume you’re working out of your personal account, so make sure to choose to fork to that location.

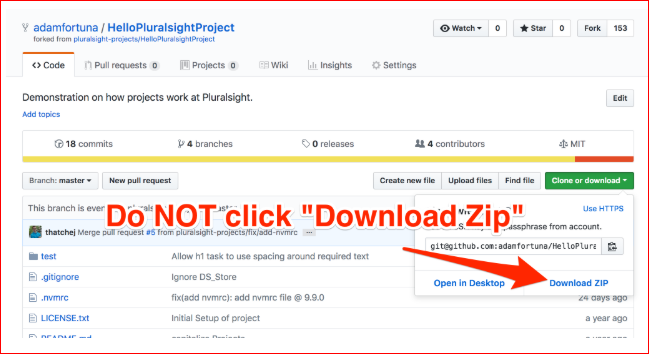


After a few seconds, the fork of this repository should show up in your account!



Cloning a Repository from GitHub

Once you have your fork created, you’ll need to get a copy of this code locally. You may initially think "I'll click the 'download zip'" option - but don't! That will cause trouble later on.



The reason for this is because downloading the zip will download the most current version of the files in this repository now, but it will not allow us to easily upload the files back to GitHub once we've made changes.

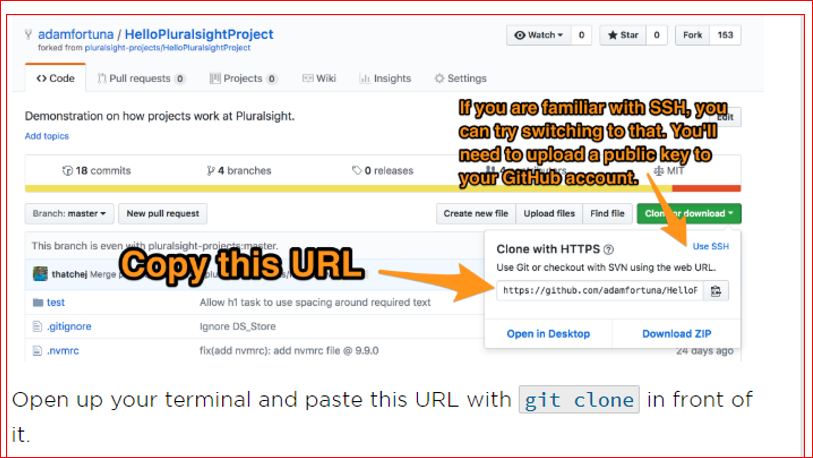
Instead, we'll need to "Clone" to repository down from GitHub onto our local machine. There are three common ways to clone the repository. You can use whichever one you are most familiar with.

1. Use the git command line interface (CLI) and a terminal.
2. Use a git client like [GitHub Desktop](https://desktop.github.com/) or [Tower](https://www.git-tower.com/).
3. Use git integration in your code editor of choice, like [Visual Studio Code](https://code.visualstudio.com/) or [Atom](https://atom.io/).

Behind the scenes, all of these are using the command line to interface with git and GitHub. Here's a super quick walkthrough of how to clone using one from each group.

Option 1: Cloning a Repository Using the Command Line

Cloning from the command line is probably the easiest to explain, but requires the deepest knowledge of git. There are only two steps which is nice!

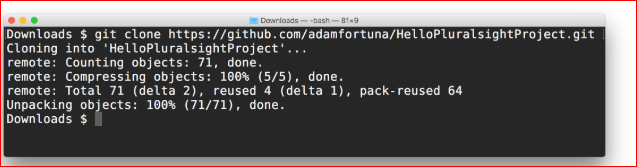


Open up your terminal and paste this URL with git clone in front of it.

1

git clone https://github.com/pluralsight-projects/HelloPluralsightProject.git

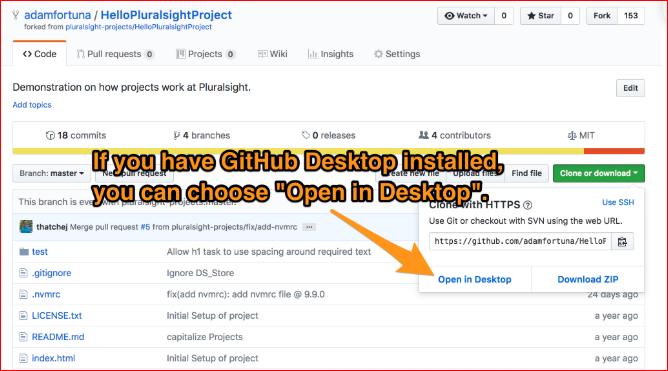
When you run this command in your terminal, you'll see something like this.

After the git clone command completes, you'll have a folder matching the name of your project -- in this case "HelloPluralsightProject".

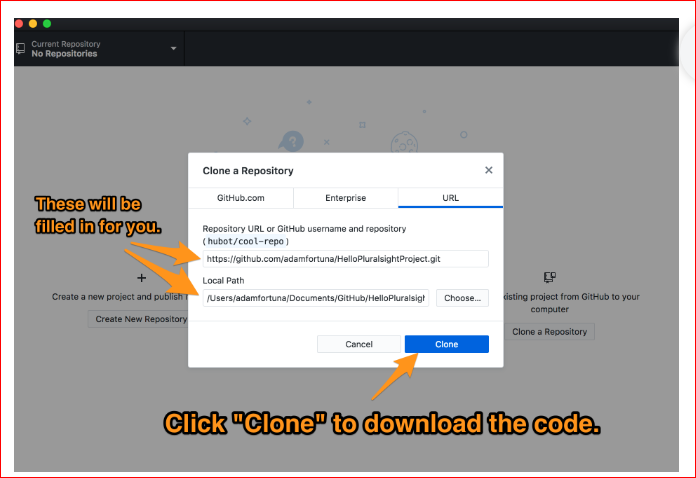
Option 2: Cloning a Repository Using GitHub Desktop

[GitHub Desktop](https://desktop.github.com/) is my favorite recommendation for beginners learning git. It has a user interface rather than a command line, so it requires less research into how git works. If you've installed GitHub Desktop, try this flow for downloading your code.

GitHub makes using GitHub Desktop extremely easy by providing this single click way to clone the repository.



Clicking this button will launch the GitHub Desktop Application and prefill it with all information needed to clone the repository.

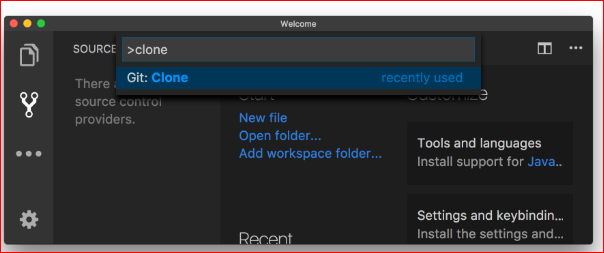


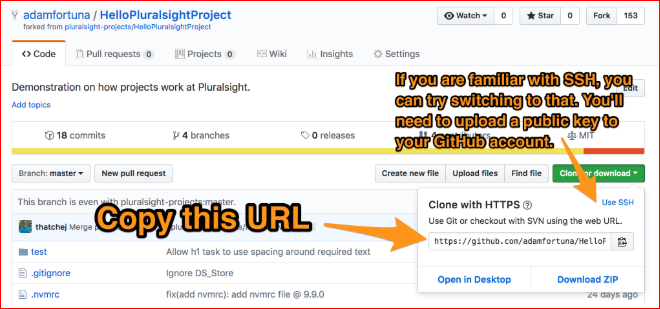
After clicking "Clone", GitHub Desktop will download all the code for this project to your local machine. At that point you're good to go! If you right click on the repository, you can choose to open it in Atom (if installed), the command line, or just the folder with the code.

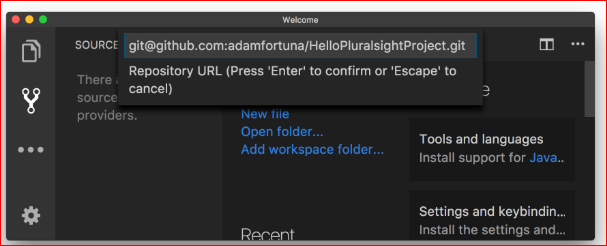
Option 3: Cloning a Repository Using Visual Studio Code

[Visual Studio Code](https://code.visualstudio.com/) is amazing editor that has Git integration built right in.

Open up the Command Palette by going to "View > Command Palette" or entering the key command (cmd+shift+P) for Mac, or (cmd+shift+P) for Windows. Type "clone" into the Command Palatte and hit enter.



Like with the command line approach, you'll need to know the GitHub repository URL. You can grab that from the Git Repository Page. 

Enter that into the Clone field in Visual Studio Code and hit enter. 

VS Code will prompt you to select a location to store this repository locally, then start downloading it to that location.

Open The Code In Your Editor

Now that you have all of the code locally, you can begin working on this project! Open up the entire folder for this project in your editor of choice.

Every Pluralsight Project is different and requires it's own setup. Follow the instructions to setup this project locally and install it's dependencies. Once it's installed and confirmed to be working, you're ready to start coding!

Pushing Code Changes to GitHub

Once you've completed some number of tasks within a project, you can upload those changes to GitHub. This process is called "pushing code to GitHub". Whenever you see "pushing code", think about it as "uploading code" or even "syncing code".

This process is a little different for each way you can interact with Git. Here are three of the most common ways to push code to GitHub.

Option 1: Pushing to GitHub Using the Command Line

The first step to pushing/uploading your code to GitHub is to take a snapshot of this version of your code and add it to the repository. You know how Time Machine needs to take a snapshot of all your files to back them up? Git is similar to that.

Because of that, pushing your code to GitHub is a two step process:

1. "Commit" your changes to your local repository.
2. "Push" all changes in your local repository to GitHub.

In your local terminal try running these two commands to perform these actions:

1

2

git commit -am "Completed the project"

git push

bash

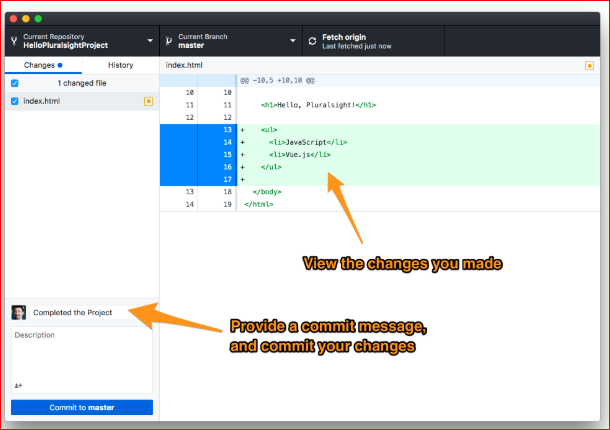
Common Problems Pushing to GitHub

If GitHub asks for your username and password but then claims they are incorrect, then you likely have 2-factor authentication setup. When you authenticate, you'll need to use a a GitHub Personal Access Token rather than your password. Try [creating a new access token](https://github.com/settings/tokens)(with repo permissions), and using that token instead of your password.

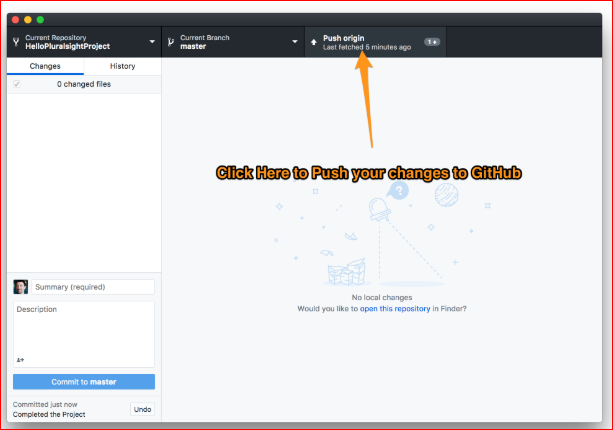
If GitHub claims you don't have access to this repository, then it's likely that your SSH settings aren't correct in some way. Try following the GitHub guide on using [SSH Keys](https://help.github.com/articles/connecting-to-github-with-ssh/) and adding your own to your [GitHub Account](https://github.com/settings/keys).

Option 2: Pushing to GitHub Using GitHub Desktop

Once you've made any updates to the code in your project, you can use GitHub Desktop to send those changes to GitHub.

Once you open up GitHub Desktop, you'll see something like the screenshot below. From this view, provide a commit message and click commit. A commit message is like a change log - something to look back it to understand what change you made to the code. 

Once your code is committed locally, you're ready to push your code to GitHub! Click the "Push origin" button which will sync your code with GitHub.

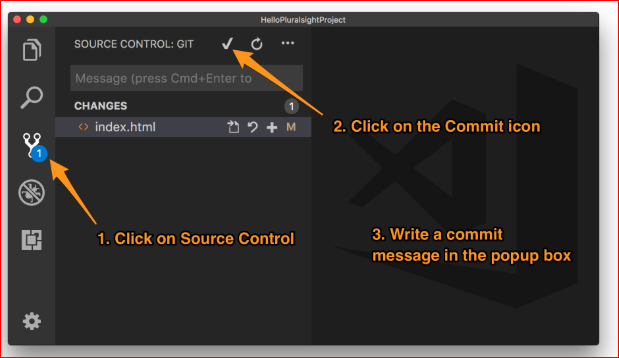


Option 3: Pushing to GitHub Using Visual Studio Code

VS Code makes integration with GitHub shockingly easy - if you know how to use it. I had to Google around and hunt for what to do, but once I found it, I appreciated how simple it is.

1. Click on the "Version Control" logo on the side of the editor.
2. Commit your changes to your local repository.
3. Provide a commit message explaining what changes you made to the code.
4. Push your changes to GitHub.

These three tasks are tough to describe with only words because they are all icon based on the interface. Follow these two screenshots to see where you can click.



Once your code is committed locally, you're ready to push your code to GitHub! 