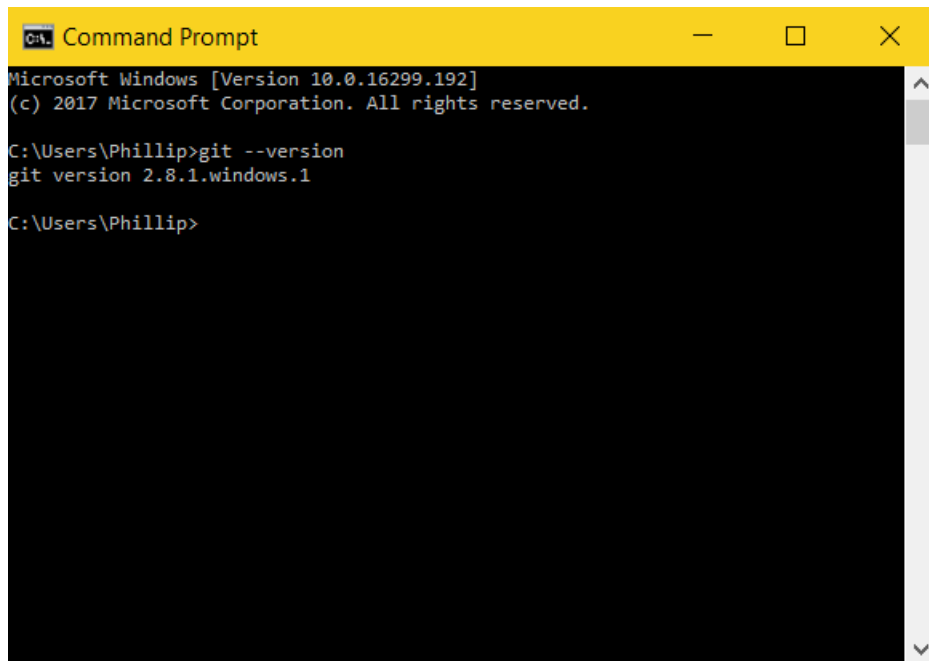


Setting up GitHub Version Control with Qt Creator*

*This tutorial is assuming you already have an account on GitHub. If you don't, go to www.github.com and set up an account using your buckeyemail account. It will also require you to install Git for your OS. You can check to see if you already have it installed by entering “git – version” into your command prompt, as shown below. You can find the install for your OS through a simple search if you need it.



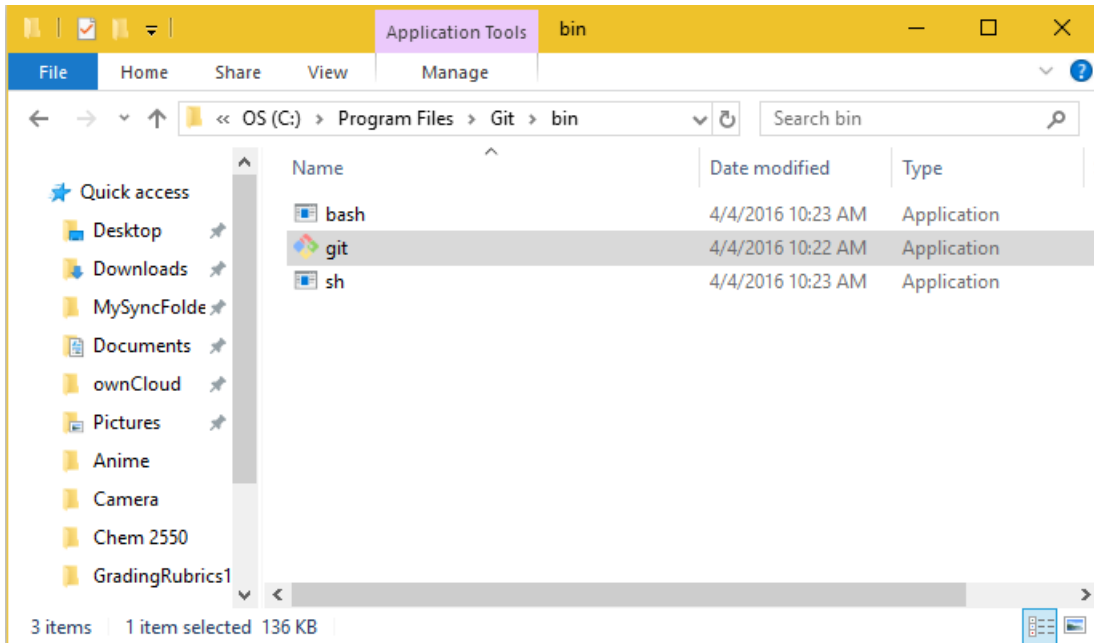
```
Command Prompt
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Phillip>git --version
git version 2.8.1.windows.1

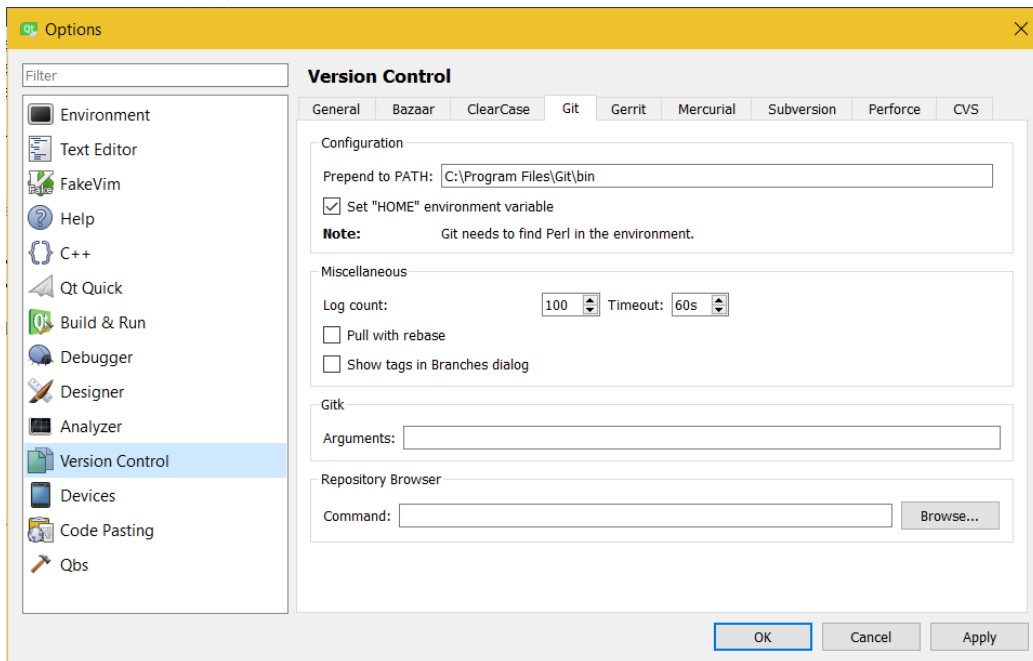
C:\Users\Phillip>
```

Setting up Qt Creator for Git

You'll first need to point Qt to the location of the Git commands. This should be located at “C:\Program Files\Git\bin”, where you'll find the application entitled “git” along with 2 other files as shown on the next page.



Once you've located the file, copy the file location. Open Qt and navigate to Options under the Tools menus. As shown below, navigate to Version Control and select Git. Paste the file location from earlier into the text box labeled "Prepend to PATH:". Once this is done, you'll be able to use Git in Qt Creator.



Starting a New Project with Git in Qt Creator

Now that Qt Creator is configured for Git, it's time to start a new project. You'll first want to create your remote repository on GitHub. Once you've logged in, click "New repository". Make sure to initialize it with a README and to add a Qt .gitignore, as shown below BEFORE creating the repository. To add the .gitignore, click "Add .gitignore" and type in "Qt".

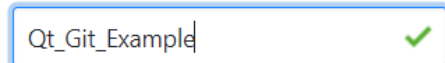
Create a new repository

A repository contains all the files for your project, including the revision history.

Owner





Repository name



Great repository names are short and memorable. Need inspiration? How about [special-telegram](#).

Description (optional)

-  **Public**
Anyone can see this repository. You choose who can commit.
-  **Private**
You choose who can see and commit to this repository.

Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: Qt ▾

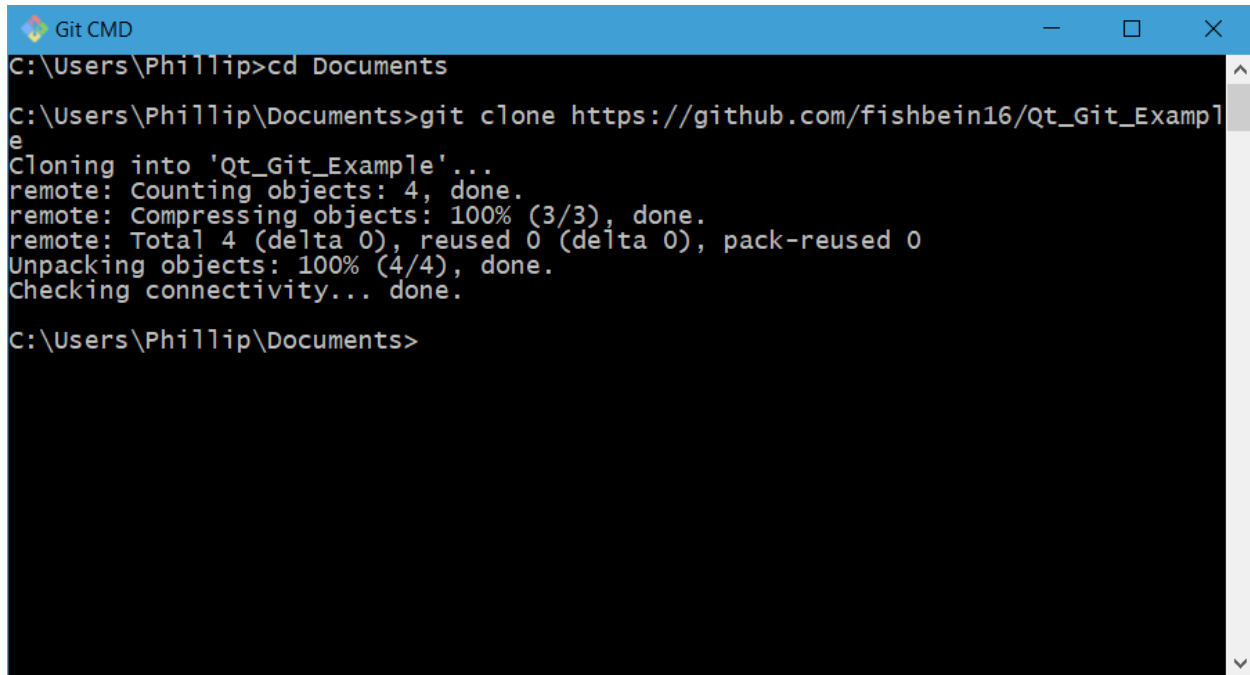
Add a license: None ▾



Create repository

Once you've created your repository, you'll need to clone it to your computer. Open the command line. For Mac users, you can open the normal terminal. For Windows users, you'll

need to open “Git CMD”. Make sure your working directory is where you intend to put the project. To clone the repository, type “git clone *repository_url*” as shown on the next page.



```
Git CMD
C:\Users\Phillip>cd Documents
C:\Users\Phillip\Documents>git clone https://github.com/fishbein16/Qt_Git_Example
Cloning into 'Qt_Git_Example'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
Checking connectivity... done.
C:\Users\Phillip\Documents>
```

If you navigate to the working directory, you should now see a folder matching the repository name. The final step is to create the project in Qt Creator. Create the project as you’ve done previously, selecting “FEH Proteus SD”. Make sure you create it in the same directory that you cloned the repository to. Name the project the exact same as the repository. Don’t worry about the warning of the project existing. Under Summary, make sure the option “Add to version control:” is set to “Git”. You’ll get a warning about the Makefile not being added to the version control; open the project anyways.

Using Version Control in Qt Creator

Your project is now set up and ready to go. You can even build it to be sure. As with other files, it is important to constantly save them. With version control, you’ll be committing your changes locally and pushing those changes to the remote repository. There will be a brief overview of this here. You can find more thorough “best practices” elsewhere online.

To commit your changes, or add files to the repository, go to Tools >> Git >> Local Repository >> Commit. You'll see a list of files with changes or those yet to be added to the remote repository, as well as a description text box. Be sure to have descriptive comments about the changes being made to help you if you need to revert to older code. Commit Information is optional but useful if multiple people are working on your code. Once you've selected your files to commit and added a description, click "Commit". If you've done it correctly, you'll see "Committed x file(s)." in the bottom of your screen.

To push your changes, go to Tool >> Git >> Remote Repository >> Push. If you made your repository Private, you'll get a prompt for your username and password. You'll see the push command at the bottom of the screen. You can confirm the push by reloading the repository in a browser. If it didn't push and you get a timeout error after a minute, you pointed Qt to the wrong place for the Git commands.

If multiple people are working on your code or you're working on multiple computers, it is a good idea to pull the current repository down. To do so, go to Tools >> Git >> Remote Repository >> Pull. If you're up-to-date, you'll be notified. Otherwise, your workspace will be updated with the changes. If a new file didn't appear to be added, right-click on the project and select "Add Existing Files".