Tutorial 12 – Eclipse GitHub Workflow 2

Contents

[1 Introduction 1](#_Toc80006957)

[2 Person 1 – Establish Initial GitHub Repo 2](#_Toc80006958)

[3 Person 1 – Setup Branch 6](#_Toc80006959)

[4 Person 2 – Setup Branch 13](#_Toc80006960)

[5 Submission 15](#_Toc80006961)

# Introduction

This lab serves two purposes.

1. A reference for how your team can use GitHub.
2. A tutorial for you as an individual illustrating how your team can use GitHub.

This tutorial will introduce you to working with Eclipse and GitHub to do Java development. The most recent versions of Eclipse come with a plugin, EGit which integrates Git into Eclipse.

Tutorials 11 & 12 are very similar. I believe the major differences are:

|  |  |  |
| --- | --- | --- |
| **Activity** | **Tutorial 11** | **Tutorial 12** |
| Create local version-controlled project | Configure git in Eclipse and then use Team, Share Project | Only uses: Team, Share Project. Not sure why the configuration for 11 was not needed. |
| Location of local repo | *users/youID/git/myRepo* | Inside the project folder (not recommended, but OK) |
| Initial Pull | Cloning | Download zip file |
| Pulling & Pushing | master branch (not the approach we should use) | Branch off of master |

Note: Both Tutorials 11 & 12 define a workflow that your team can use for the group project. Elements of either can be interchanged. The point is to show you the different possibilities and then you can choose and/or create the best workflow for you and your team.

One thing I know these tutorials do not address is how to roll back (revert) changes in Eclipse.

This tutorial starts from scratch.

# Person 1 – Establish Initial GitHub Repo

The steps in this section demonstrate what *Person 1* will do to initialize the repo. **These steps will only be done once, and by one person.**

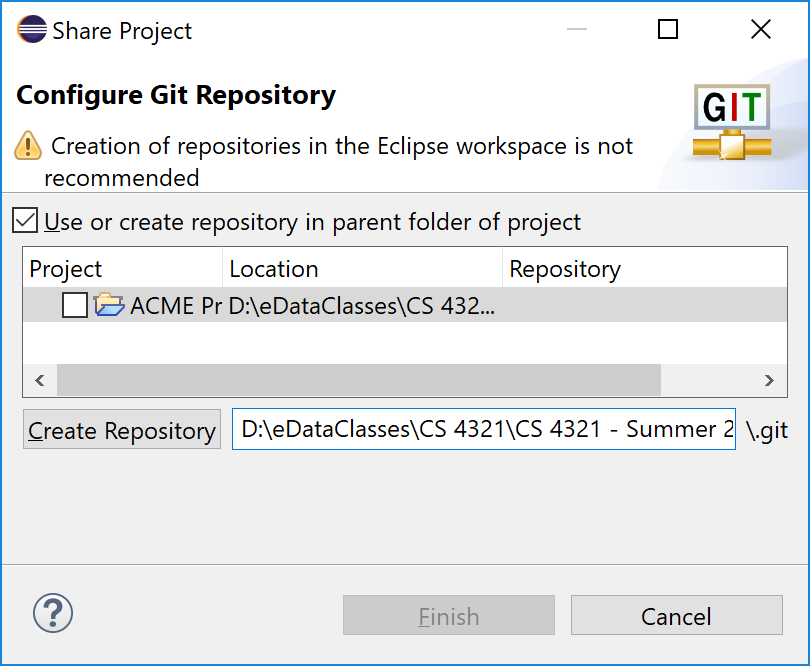
1. Create workspace (File Explorer) named: *ACME Workspace*
2. Open Eclipse in that workspace
3. Create a Java Project name: *ACME Project* (If asked for a module name, dismiss the dialog)
4. Create a package named: *utils*
5. Crate a class named (checkbox for generate a *main*): *Foo*
6. Replace *main* with:

**public** **static** **void** main(String[] args) {

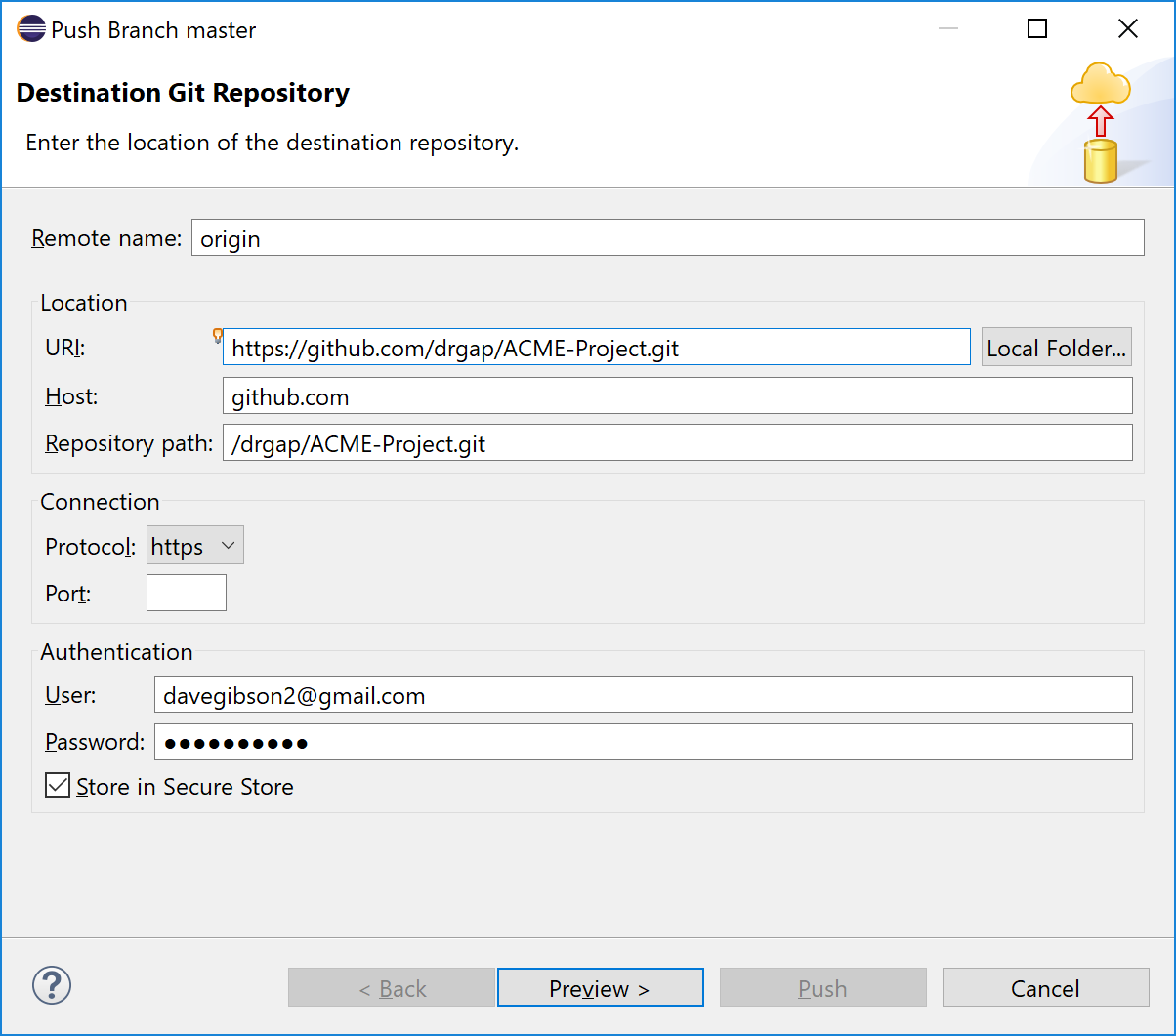
System.***out***.println("howdy");

}

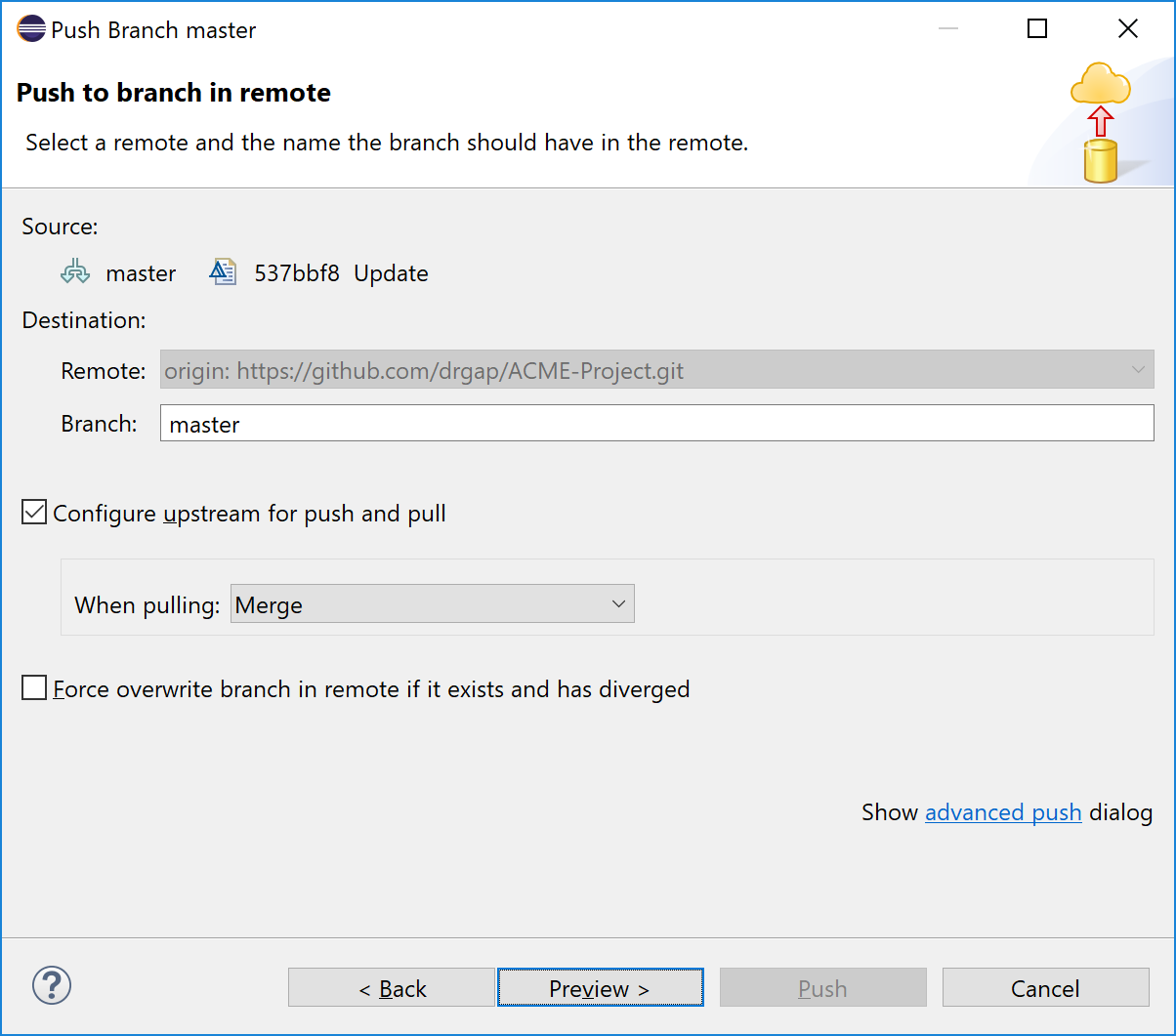
1. Create a repo in GitHub named: *ACME-Project*
2. Copy the HTTPS URI. For example: https://github.com/drgap/ACME-Project.git
3. In Eclipse, right-click the project and choose: Team, Share Project…
4. Do these things:
5. Check “Use or create repo…”
6. Select the project where the checkbox is. When you do this, the Create Repo… Button is enabled (however, it doesn’t select the checkbox, which is fine)
7. Choose: Create Repository
8. Finish



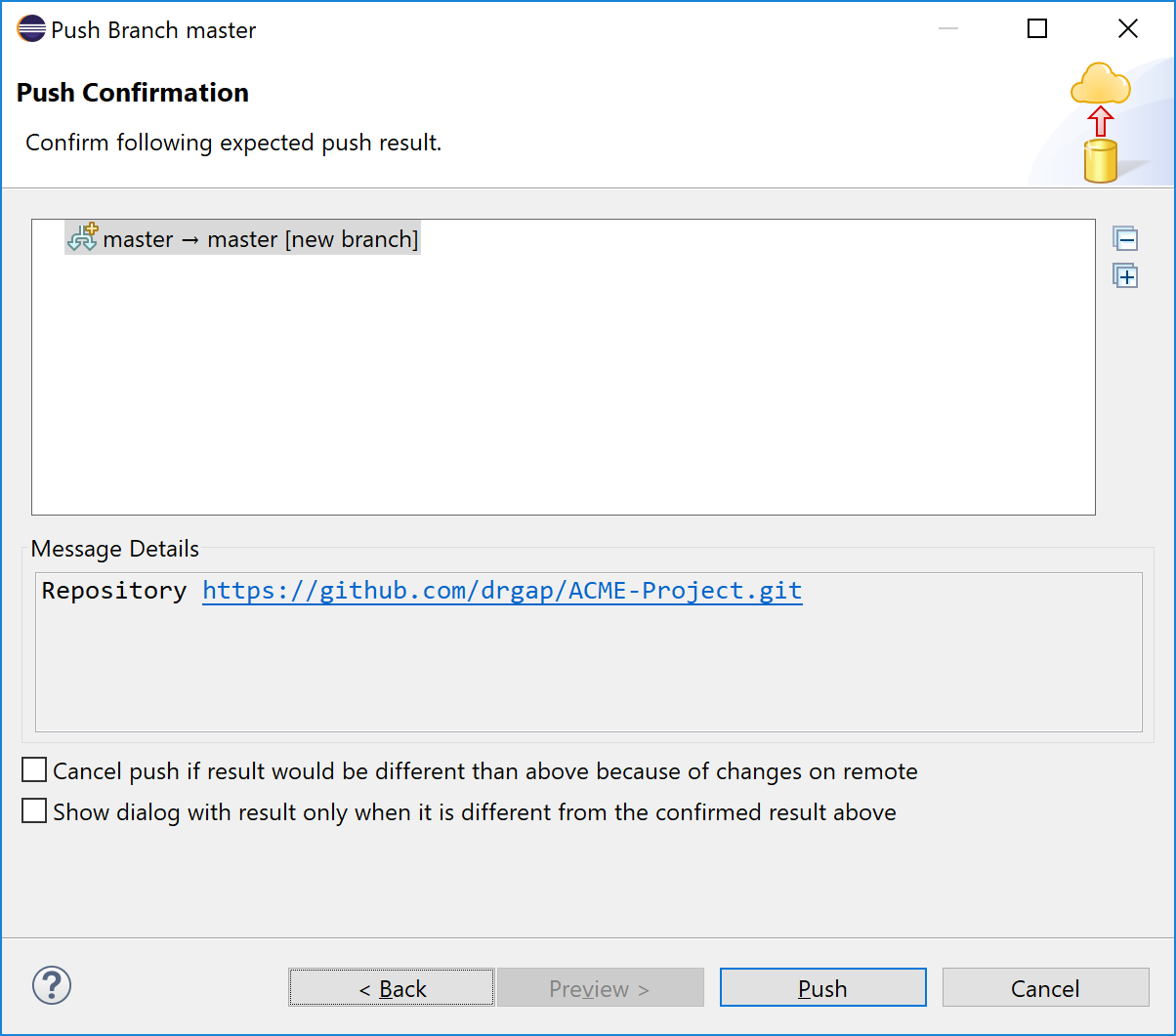
1. Right-click *Foo* and choose: Team, Add to index
2. Right-click *Foo* and choose: Team, Commit, type a commit message, press: Commit
3. Right-click project and choose: Team, Push Branch ‘master’…
4. Paste the URI, specify your ID and password and choose: Preview



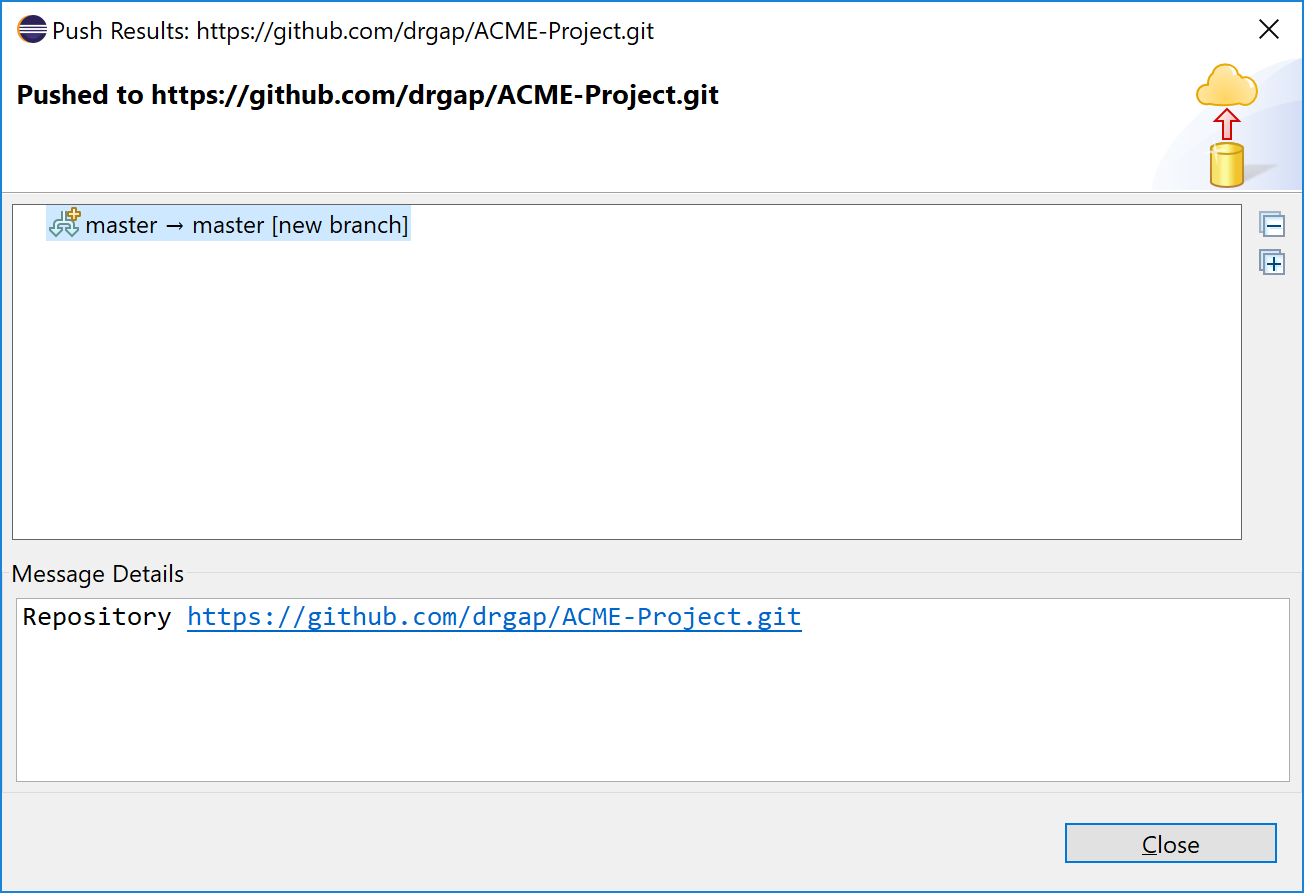
1. Result is shown below, choose: Preview



1. Result is shown below, choose: Push

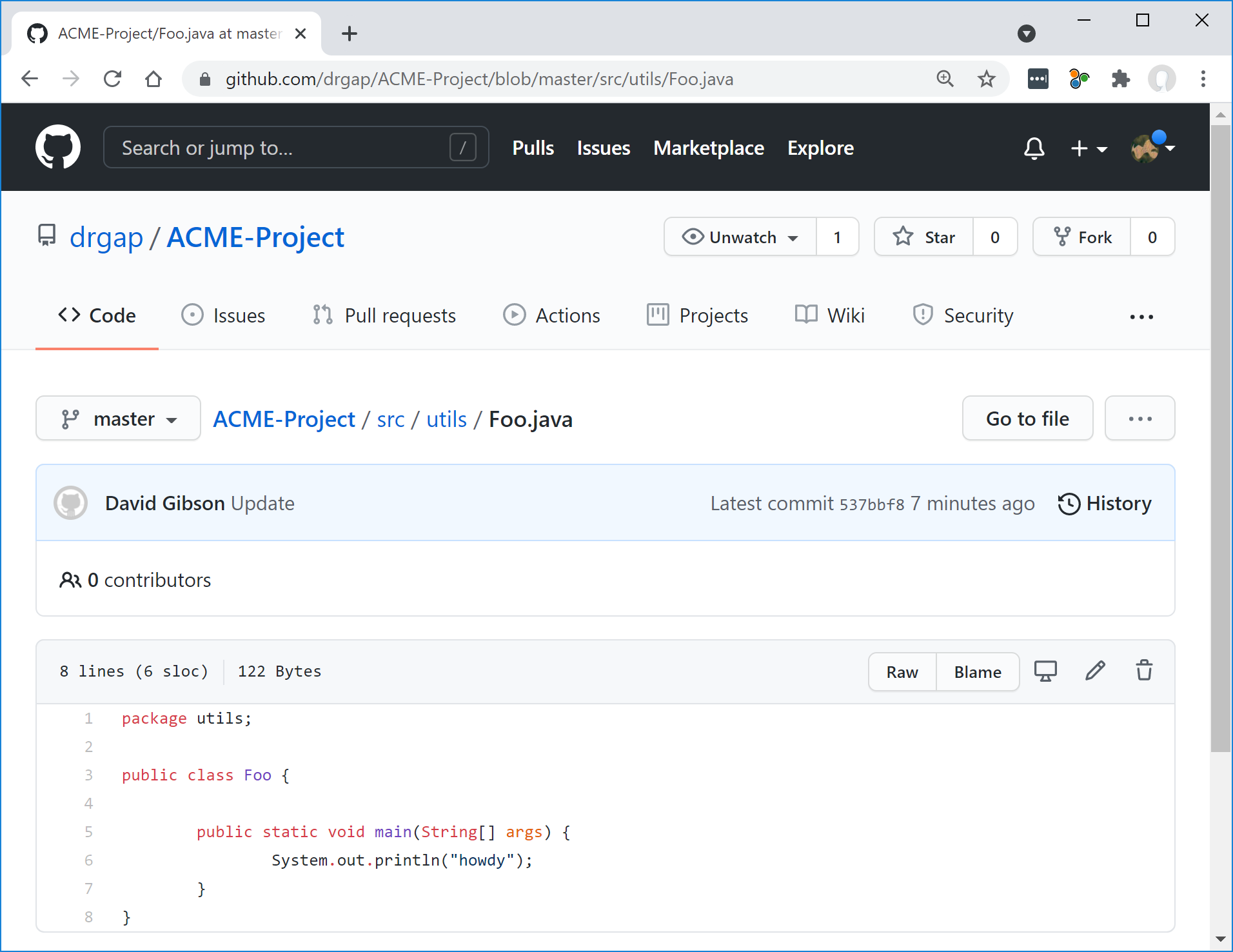


1. Result is shown below, choose: Close. IF yours was “rejected” see below this image for directions.



|  |
| --- |
| If push was not successful, the following will work, but not the best approach. Go back through the dialogs again and choose “Force overwrite…” as show below. |

1. Go to your *ACME-Project* repo in GitHub, display the Code, and verify that *Foo* is present.



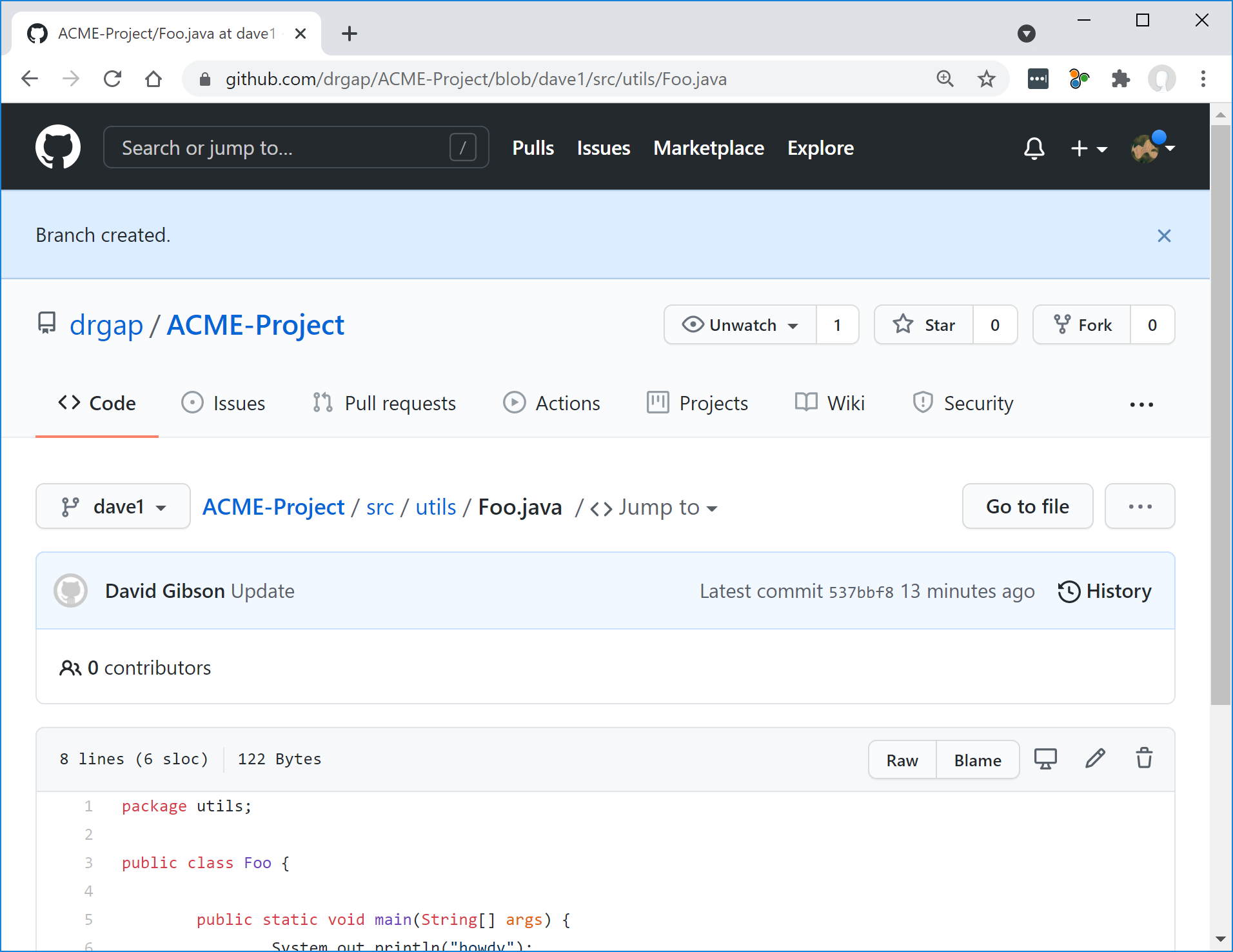
1. Edit the file in GitHub and add this line to the file and commit.

System.out.println("hi from GitHub");

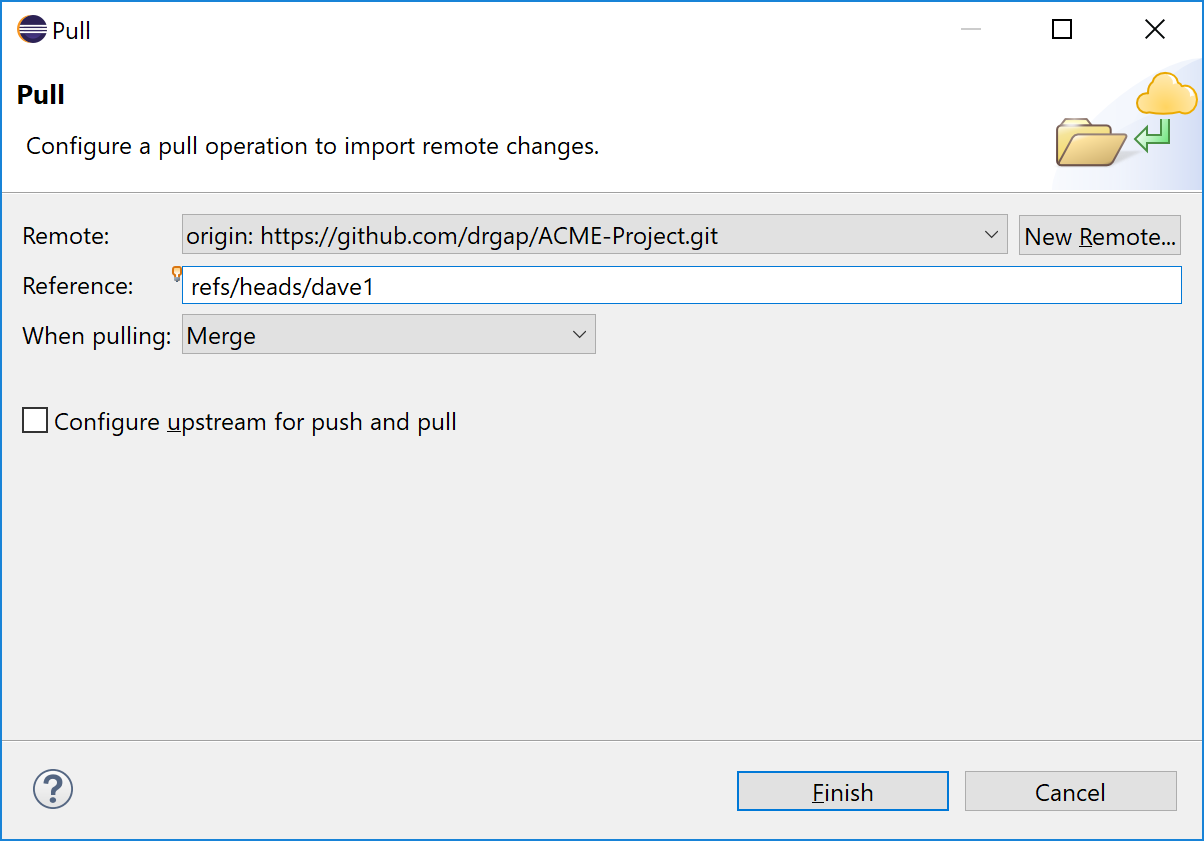
# Person 1 – Setup Branch

Now, suppose Person 1 (who did the initial setup in Section 1 above) is ready to do some coding. Person 1 should follow the steps below.

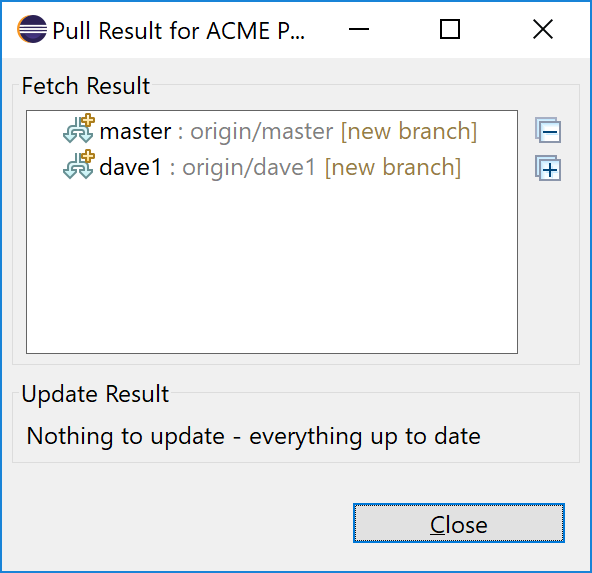
1. Go to your *ACME-Project* repo on GitHub.
2. Create a branch. I named mine *dave1*



1. In Eclipse, right-click project and choose: Team, Pull…
2. Type in the name of your branch beside “Reference” and then choose: Finish. Just time the name and the path stuff will be auto-populated.



1. The result is shown below, choose: Close



1. Open *Foo* and verify that the new line is present:

**public** **static** **void** main(String[] args) {

System.***out***.println("howdy");

System.***out***.println("hi from GitHub");

}

1. Add this line (substitute your branch name) to *Foo* and save.

System.***out***.println("hi 'dave1' branch, from Eclipse");

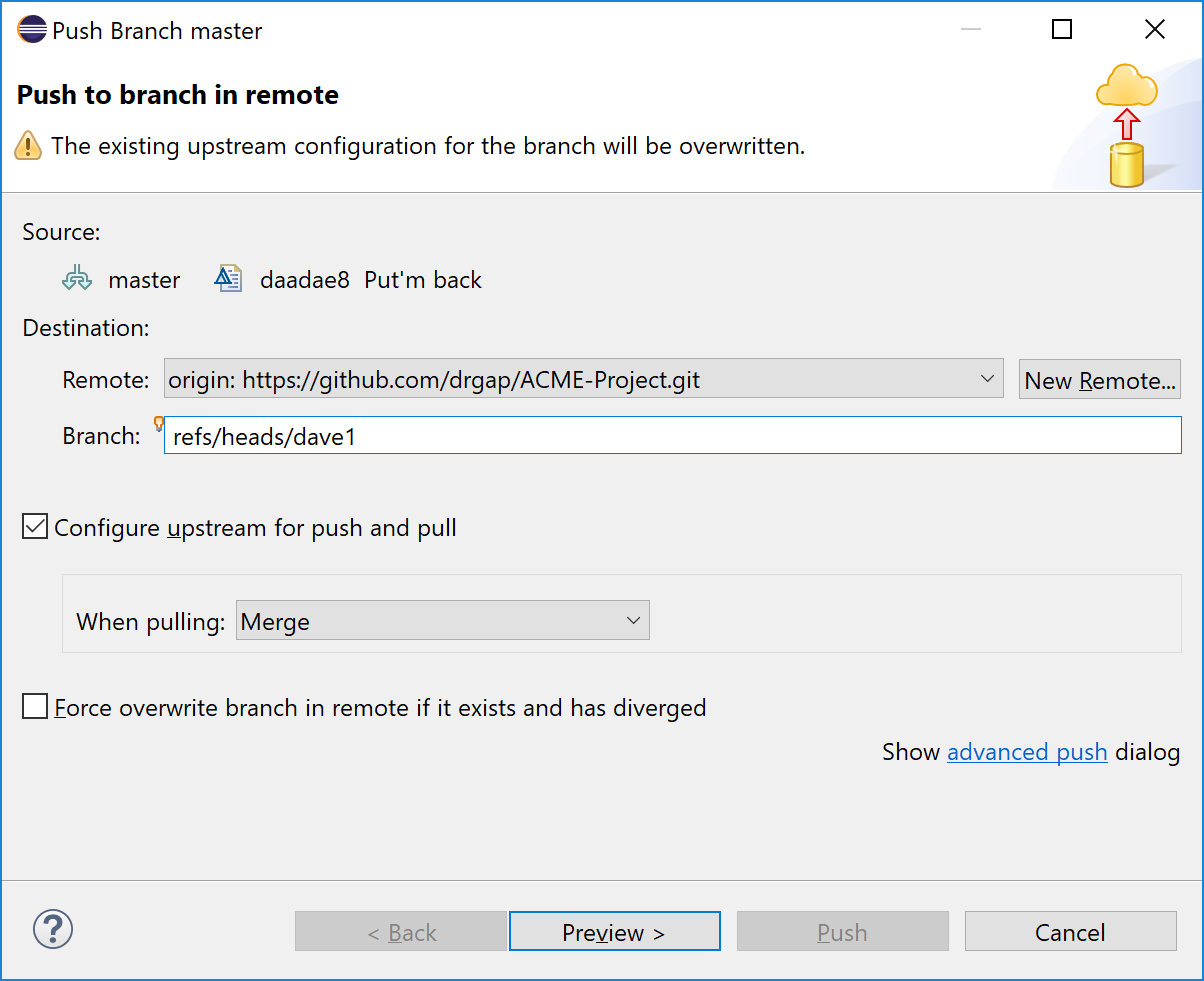
1. Right-click *Foo* and choose: Team, Commit, type a commit message, press: Commit

Note: sometimes I “lose” the Commit button, and the only option is Push Head. Do this: in the upper-right of the “Commit Message” dialog, choose the tiny icon that says, “Amend (edit previous commit)”

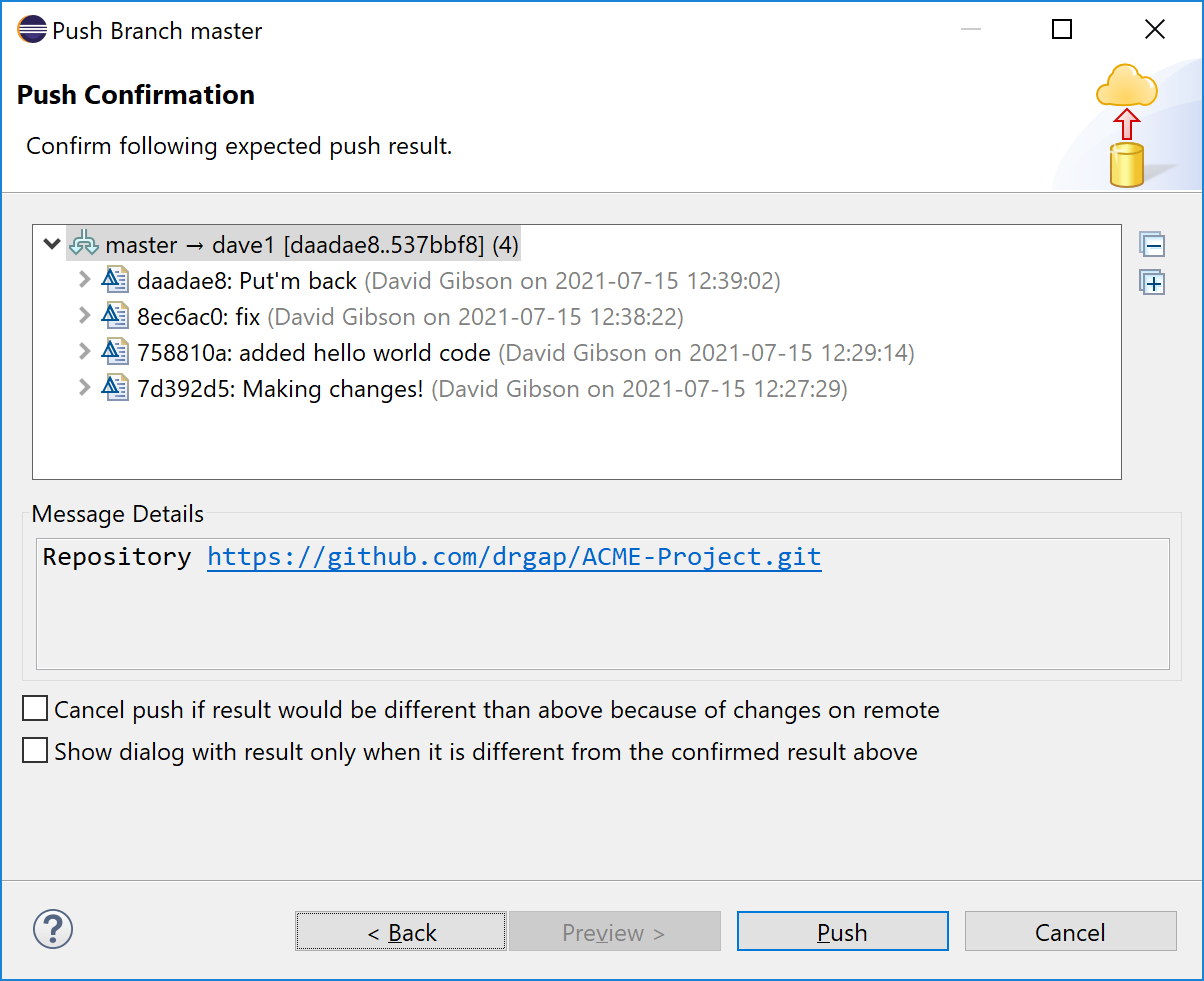
1. Add another line (we want several commits for this demo) to *Foo* and save.

System.***out***.println("hello worlds");

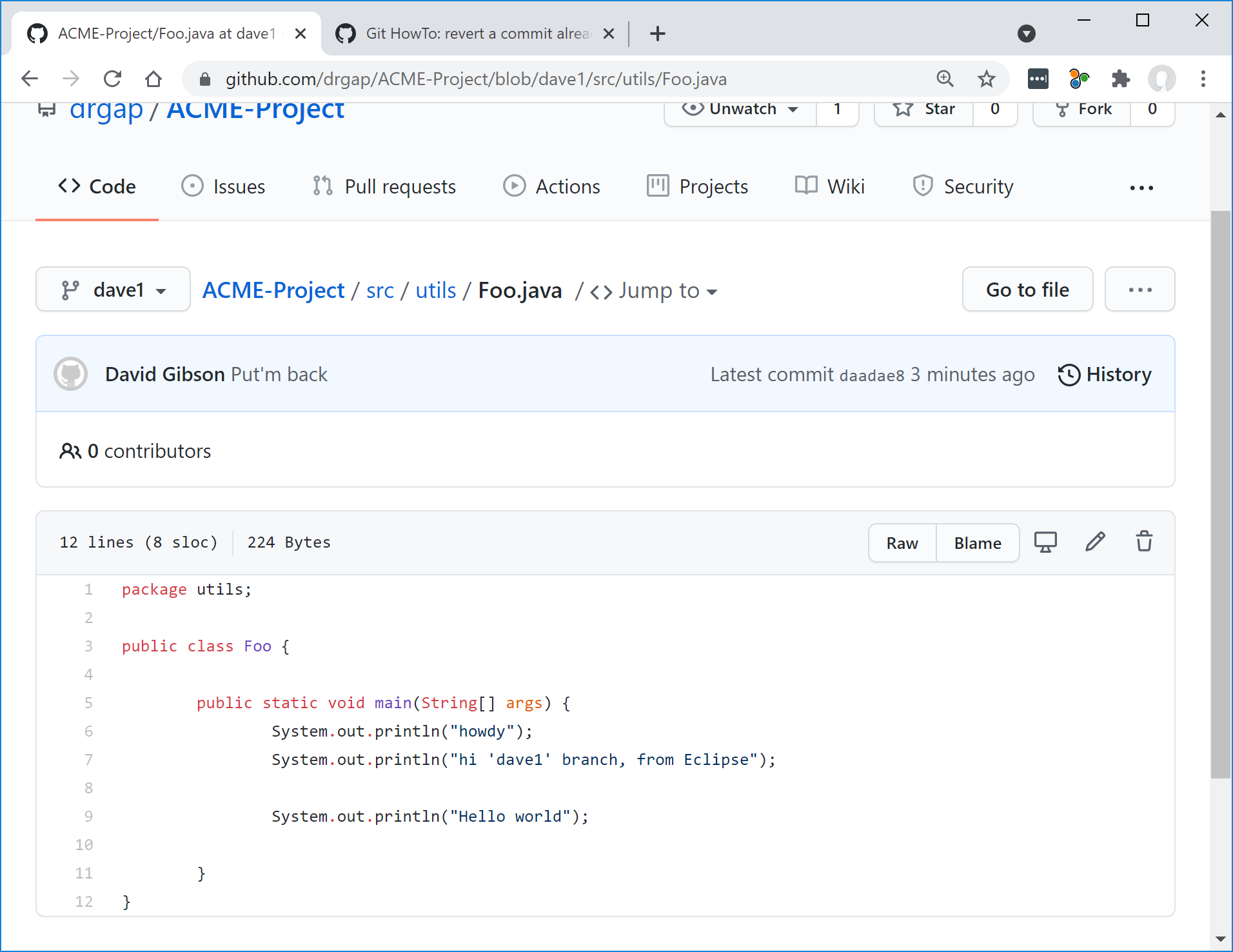
1. Right-click *Foo* and choose: Team, Commit, type a commit message, press: Commit
2. Right-click project and choose: Team, Push Branch ‘master’…
3. At the “Branch” prompt, type the name of your branch, then choose: Preview.



1. Result is shown below. Mine shows I’m pushing 4 commits. Yours will problem have 2. Choose: Push

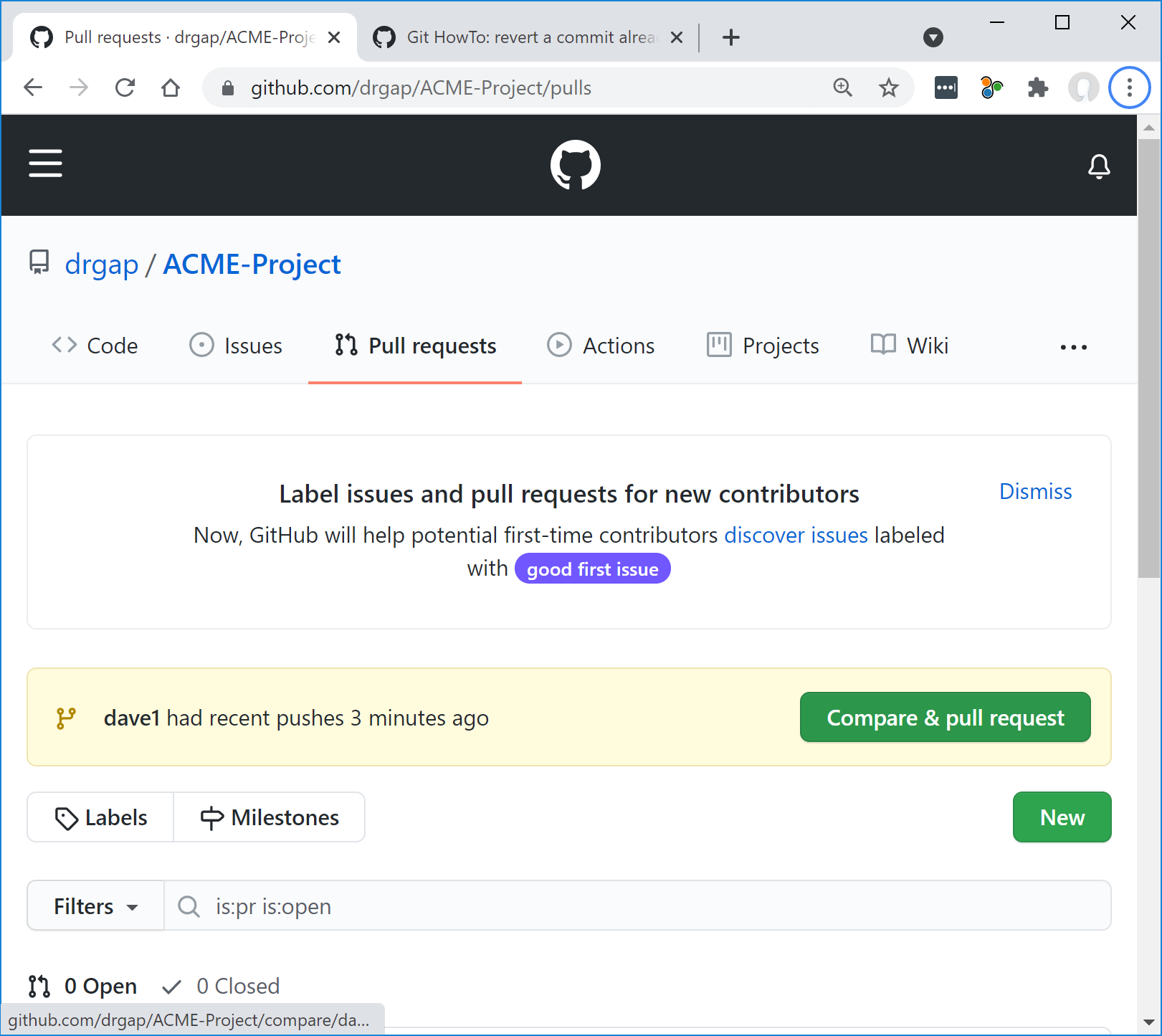


1. Go to GitHub, repo and branch and verify the changes are present (figure below may have different code than yours)

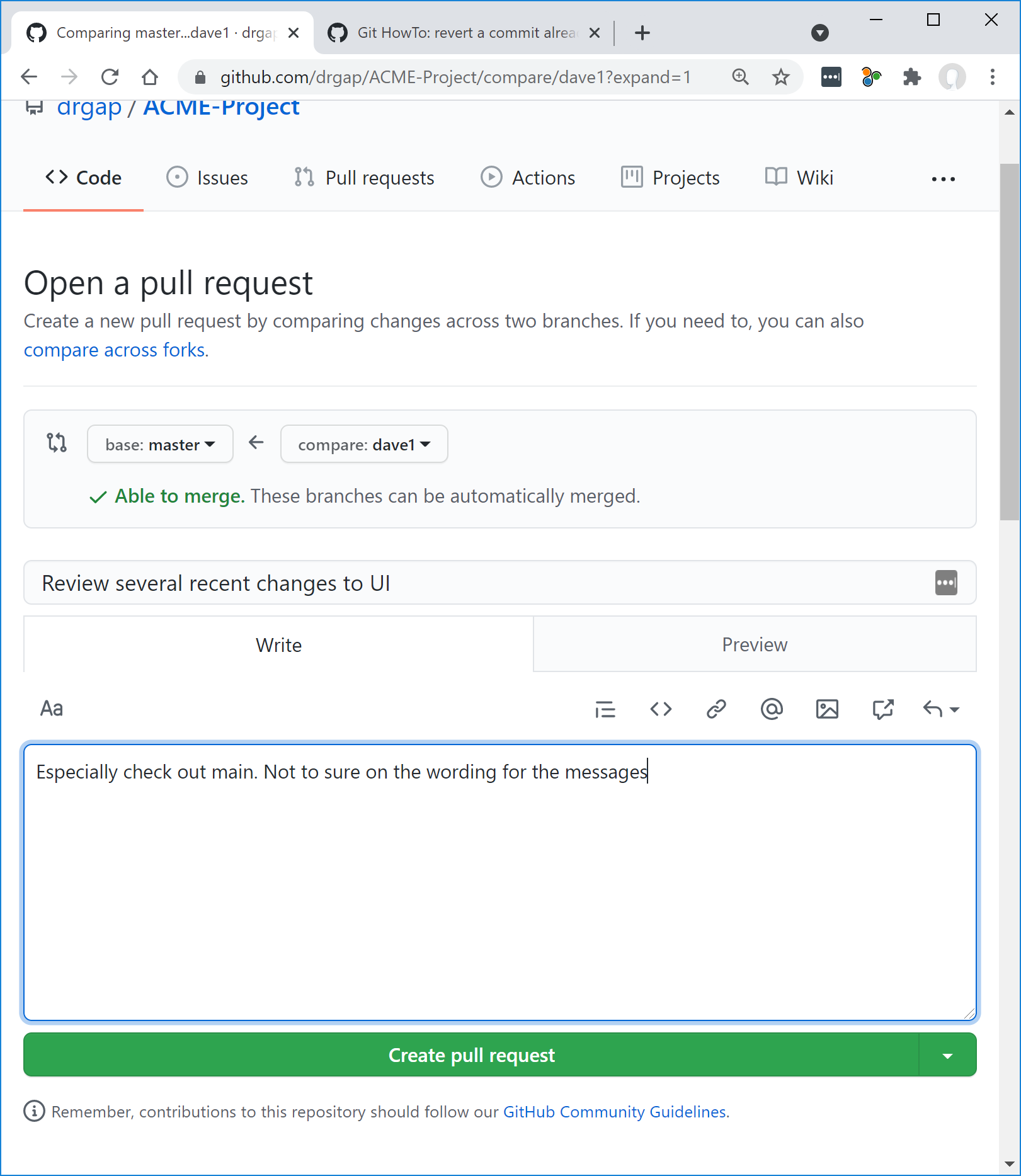


1. Now, you could continue working in this fashion. At some point, you will want to open a Pull Request so that you can merge your branch with master.

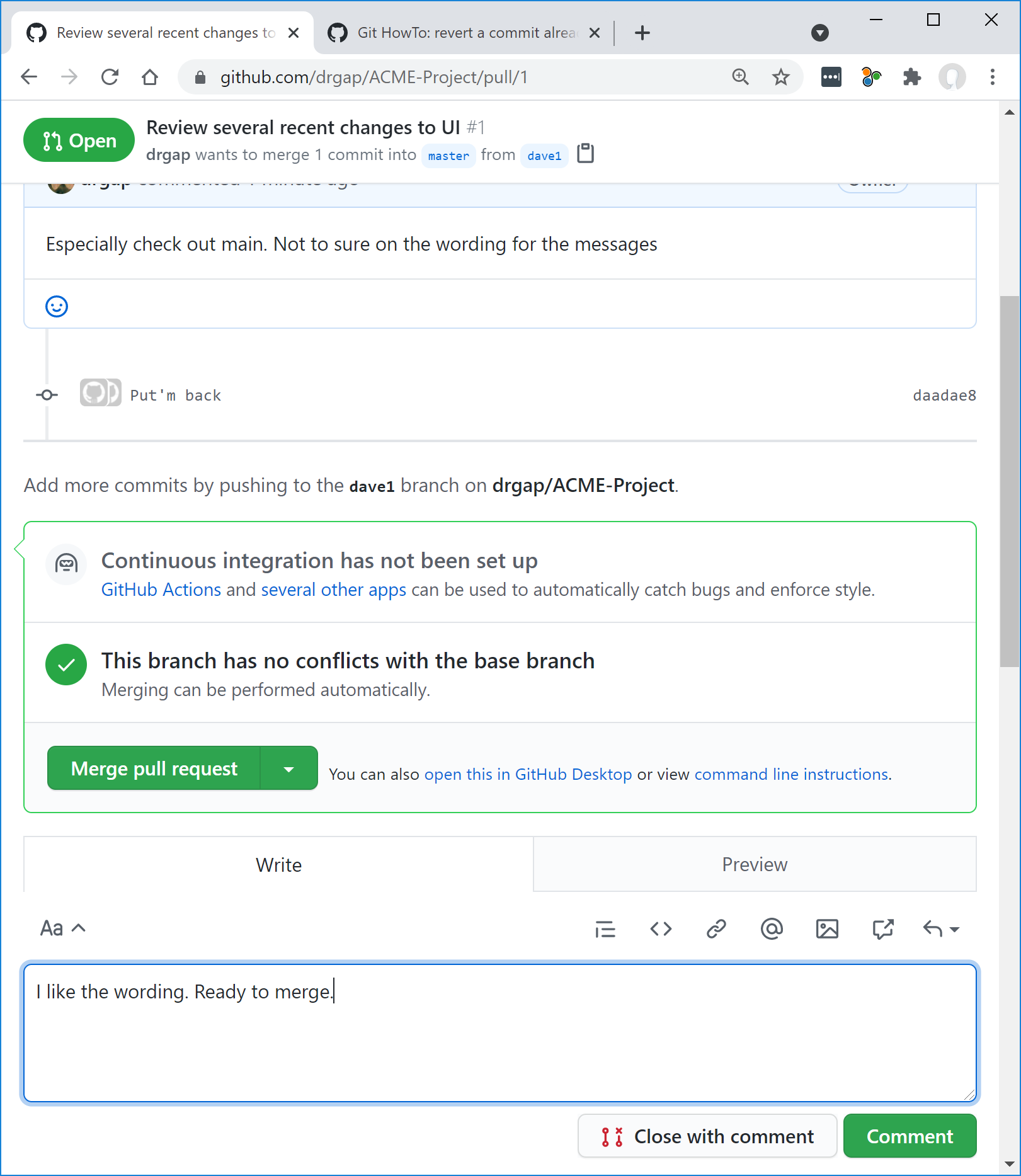
Go to GitHub and choose: Pull requests. You will probably see that one is “ready” (the yellow highlight). Choose: Compare & pull request



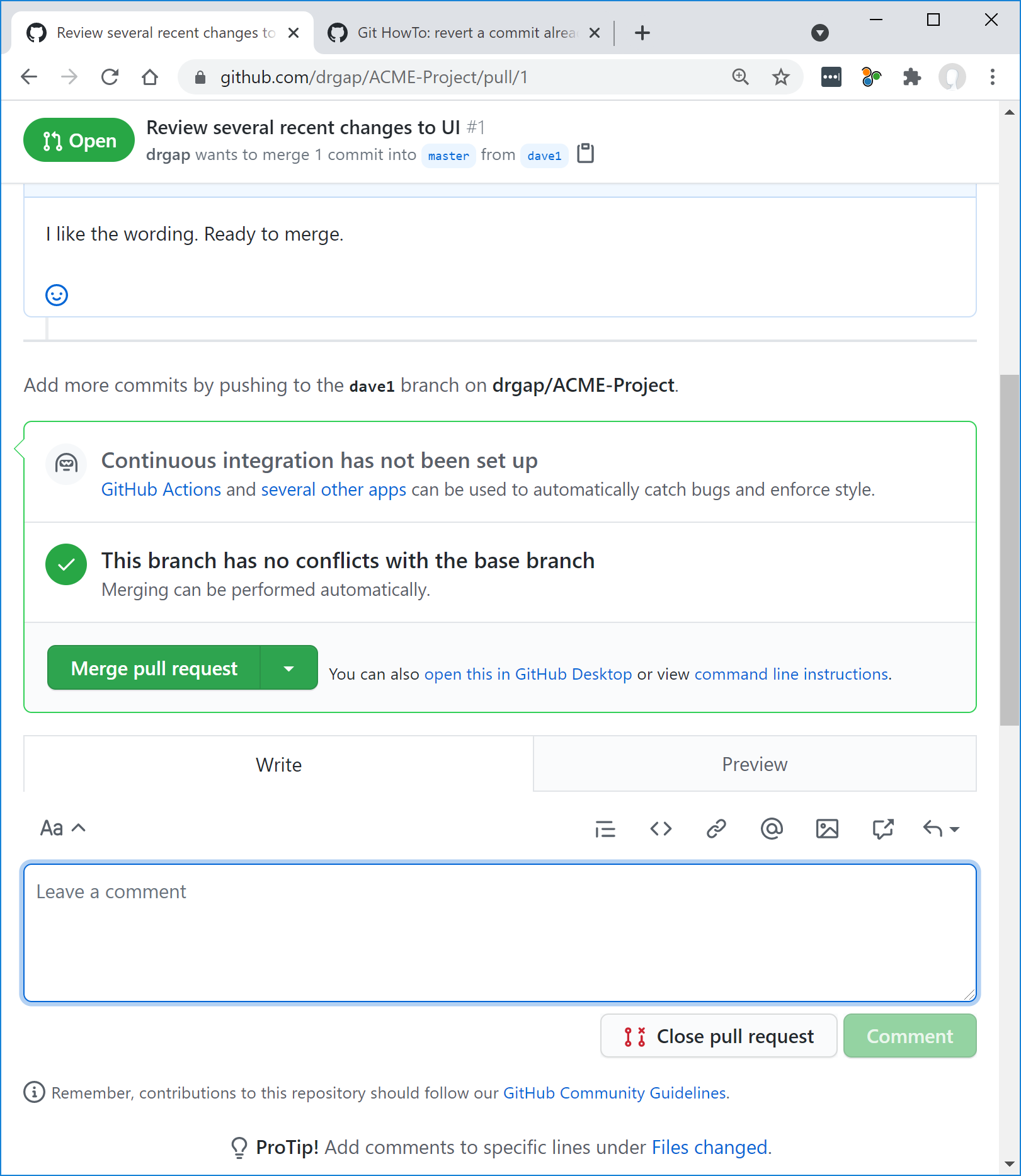
1. Type a comment and choose: Create Pull Request.



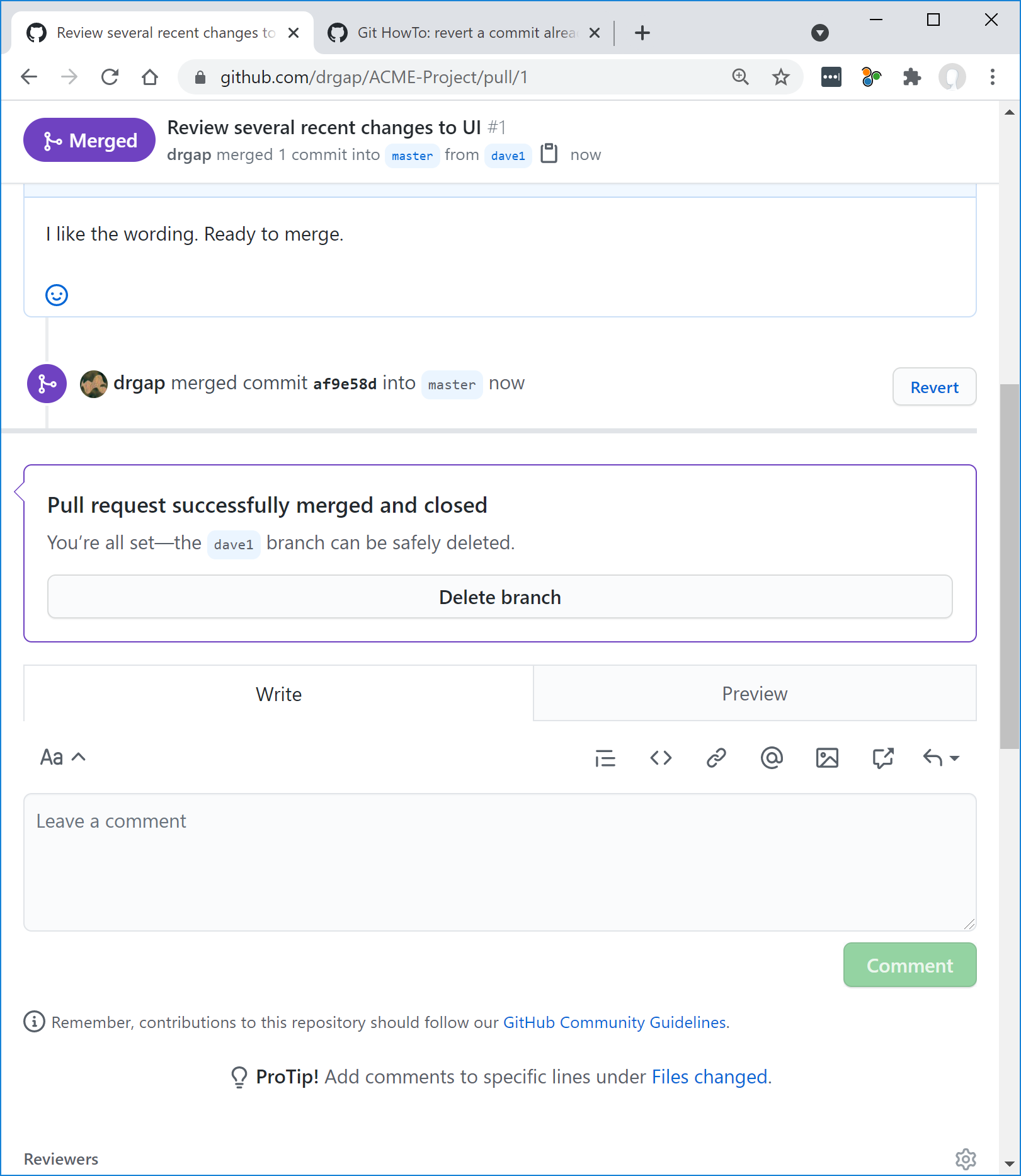
1. Result is shown below. Presumably, someone else would review your changes as requested in the pull request. However, you will do it yourself. Type in a comment at the bottom as shown below, and then press: Comment



1. Result of review is shown below. No, ready to merge. Choose: Merge pull request and then: Confirm merge.



1. Result is shown below

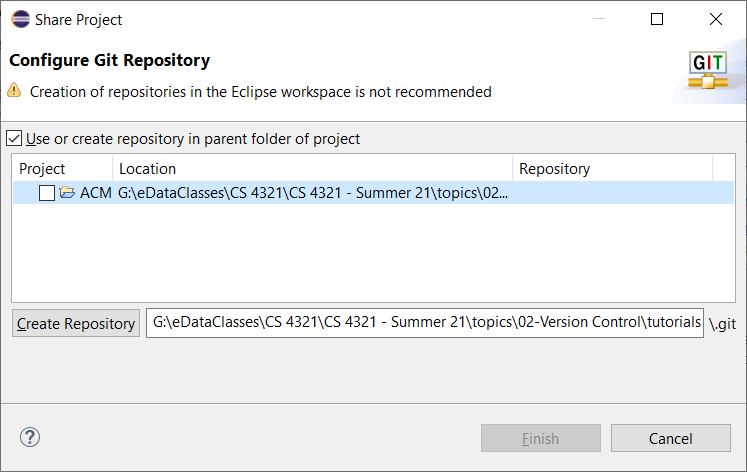


1. Now, person 1 can just keep on working in this fashion.

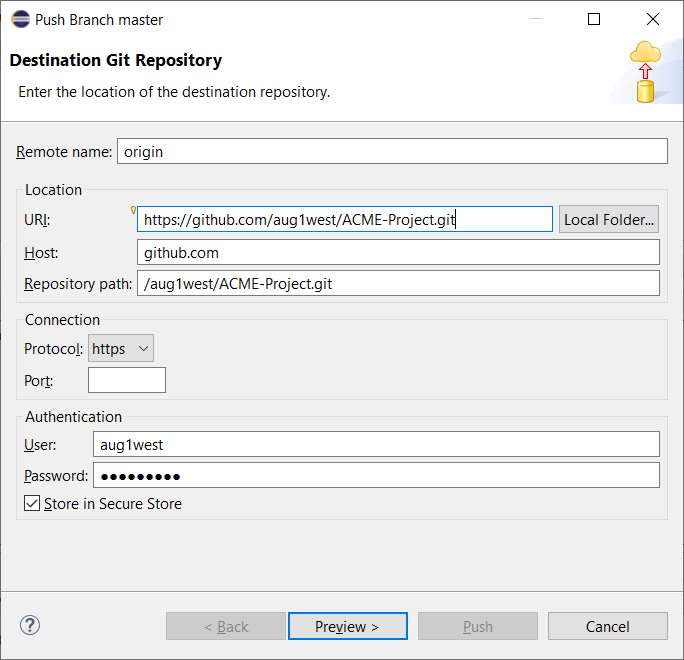
# Person 2 – Setup Branch

This procedure will be followed by Person 2 (and other team members) when they are ready to code. For this tutorial, you will simulate the actions of Person 2.

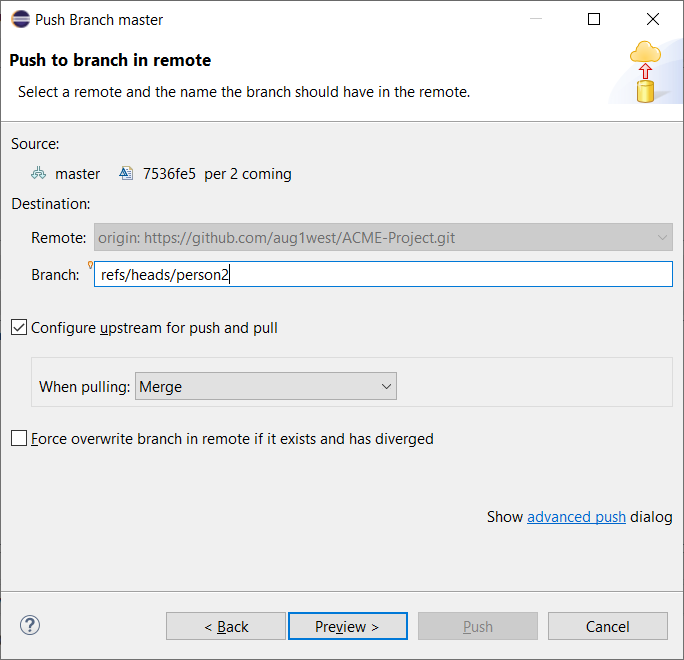
1. Create workspace (File Explorer) named: *ACME Workspace2*
2. Open Eclipse in that workspace
3. Create a Java Project name: *ACME Project* (If asked for a module name, dismiss the dialog)
4. In GitHub, choose the *ACME-Project,* and create a branch named: “person2”.
5. From the “Code” drop down, select, Download ZIP.
6. Unzip somewhere, but NOT in the *ACME Workspace2* folder.
7. In the unzipped folders, drill down to the *utils* folder, and then drag it to *src* node in Eclipse (be sure, “Copy files and folders” is selected).
8. Select the project node in Eclipse, and choose: Team, Share
9. Do these things:
10. Check “Use or create repo…”
11. Select the project where the checkbox is. When you do this, the Create Repo… Button is enabled (however, it doesn’t select the checkbox, which is fine)
12. Choose: Create Repository
13. Finish



1. Do the following:
2. Add a line of code to *Foo*
3. Add it to the index (i.e. stage changes) – Select *Foo,* choose: Team, Add to index
4. Commit it – Select *Foo*, choose: Team, Commit
5. Type message, choose: Commit
6. Select the project node and choose: Team, Push Branch ‘master’…
7. Copy the URI from GitHub and paste into dialog as shown below. Type your Id and password if needed. Then, Preview.



1. Type the name of the Branch: *person2* and choose: Preview, then Push, then Close



1. Verify the changes on GitHub. Refresh the page if necessary. If you don’t see the changes, then likely the push was rejected and you didn’t notice that. If so, a solution was provided in the early part of Section 1 of this tutorial.

# Submission

1. **Do the following:**
2. Make a screen shot similar to 13 above.
3. Place the image in the *HW VCS* document in the appropriate place.
4. The image should easily readable without zooming in or out.