ID\_\_\_\_\_\_\_6388014\_\_ Name \_\_\_\_\_\_\_\_\_\_\_Waris Damkham\_\_\_\_\_\_\_\_\_\_Group \_\_\_\_\_\_Samoyed\_\_\_\_\_\_\_\_

## Lab Exercise #1

## Version Control System (Git) and Code Review

|  |
| --- |
| **Create a copy of this exercise sheet and fill in your answers. After finishing, save as a PDF and submit on MyCourses.** |

This lab exercise is both individual and team work. Complete the exercise by following the steps below [[1]](#footnote-1)\*.

*Git is a distributed version control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows. — Wikipedia*

**0. Preparation (individual):**

1. Installing Git on your operating system by downloading the installation file here: <https://git-scm.com/download>
2. Getting yourself familiar with basic command line terminal and tools.  
     
   **Windows**
3. Open Command Prompt
4. Try the following commands

|  |  |
| --- | --- |
| **Command** | **Description** |
| dir | Show items at the current directory |
| cd | Show the current directory |
| cd <location> | Change directory to <location> |
| mkdir <name> | Create a directory |

1. Create a new directory called “test” for your software project

|  |
| --- |
| $ cd C:\<Your Preferred Location>  $ mkdir test  $ cd test |

1. Open Vim (text editor) and learn basic functions by following the steps below.

|  |
| --- |
| $ notepad se.txt |

1. Type “Hello ITCS371”.
2. Save the file
3. You’ll be back at the prompt again. Check the file(s) in the current directory.

|  |
| --- |
| $ dir |

1. Save a screenshot of your screen with the command prompt at step 7 and paste below.

|  |
| --- |
|  |

**1. Git Setup (individual)**

1. Check Git version

|  |
| --- |
| $ git --version |

1. See all Git commands

|  |
| --- |
| $ git help |

1. Configure your Git username and email using the following commands, **replacing Emma's name with your own.** These details will be associated with any commits that you create:

|  |
| --- |
| $ git config --global user.name "Emma Paris"  $ git config --global user.email "eparis@gmail.com" |

**2. Setting up a local repository (individual)**

1. Initialize Git in the test project directory (i.e., repository) that you created previously.

|  |
| --- |
| $ cd <location>/<to>/<your>/test  $ git init |

1. Check the status of files in your Git repository.

|  |
| --- |
| $ git status |

1. Add the se.txt file to staging mode.

|  |
| --- |
| $ git add se.txt |

1. Check the status of files in your Git repository.

|  |
| --- |
| $ git status |

1. Commit the change.

|  |
| --- |
| $ git commit -m “Added se.txt” |

1. Check the status of files in your Git repository.

|  |
| --- |
| $ git status |

1. See the list of commits

|  |
| --- |
| $ git log  $ git log --name-status |

1. Modify the file se.txt further by adding one more line

se.txt

|  |
| --- |
| Hello ITCS371  Introduction to Software Engineering |

1. Check the status

|  |
| --- |
| $ git status |

1. See the new changes

|  |
| --- |
| $ git diff se.txt |

1. Stage the file and commit the change

|  |
| --- |
| $ git add se.txt  $ git commit -m “Added a modification to se.txt” |

1. See all the commits again

|  |
| --- |
| $ git log  $ git log --name-status |

1. Compare the file a.txt between two commits

|  |
| --- |
| $ git diff <commitID#1 (older)> <commitID#2> se.txt |

1. Save a screenshot of your screen with the command prompt at step 13 and paste below.

|  |
| --- |
|  |

**3. Work on a remote repository on GitHub (Team)**

Next, we will use Git to work in a team using the team project repository that you already have on GitHub.

1. If you are using Mahidol’s Wi-Fi, add the proxy authentication into your machine git configuration (skip this if you’re using your machine at home).

|  |
| --- |
| $ git config --global http.proxy http://<Mahidol Username>:<Mahidol Password>@proxy-sa.mahidol:8080 |

1. Clone the group project’s repository from GitHub by going to github.com, login and navigate to your project, and copy the repo’s clone URL.
2. Fill in the URL of your project’s repository here:

|  |
| --- |
| https://github.com/ICT-Mahidol/2022-ITCS371-1-Samoyed.git |

1. Back to your Terminal, clone the repository

|  |
| --- |
| $ cd <your>/<preferred>/<location>  $ git clone <repo’s URL> |

1. Go into your repo’s directory and check the status of the repo.

|  |
| --- |
| $ cd <repo name>  $ git status |

1. Create a new folder for this exercise. Name it “lab-exercise-1”.

|  |
| --- |
| $ mkdir lab-exercise-1  $ cd lab-exercise-1 |

1. Create a new Java code file based on your first name called <Yourname>.java using any text editor.

Add the following content by replacing <Yourname> with your first name.

|  |
| --- |
| public class <Yourname> {  public static void main(String[] args) {  System.out.println("Hello <Yourname>");  }  } |

1. Stage the file and commit

|  |
| --- |
| $ git add .  $ git commit -m “Added <Yourname> class file” |

1. Push the commit to a remote repository on GitHub

|  |
| --- |
| $ git push |

1. Check the status again

|  |
| --- |
| $ git status |

1. Wait until all the team members push their changes to GitHub, then compare the status of your local repository to the remote repository

|  |
| --- |
| $ git fetch |

1. Pull the new changes and merge them into your local repository.

|  |
| --- |
| $ git pull |

1. Check what is added into the repository and investigate the commit logs.

|  |
| --- |
| $ ls (Linux) or cd (Windows)  $ git log |

1. Save a screenshot of your screen with the command prompt at step 11 and paste below.

|  |
| --- |
|  |

**4. Branching**

1. See the current branches

|  |
| --- |
| $ git branch |

1. Create a new branch and list all the branches again

|  |
| --- |
| $ git branch dev\_<Yourname>  $ git branch |

1. Switch to the new branch to make modifications (e.g., adding a new feature). Check that you have already moved to the new branch.

|  |
| --- |
| $ git checkout dev\_<Yourname>  $ git branch |

1. Create a new file for the newly developed feature using any editor and call it <YournameFeature>.java. Put the following code in.

|  |
| --- |
| public class <Yourname>Feature {  public void feature() {  // this is a new feature  }  } |

1. Stage the file and commit.

|  |
| --- |
| $ git add <YournameFeature>.java  $ git commit -m “Added a new feature” |

1. See the current files in the repository (under the dev\_<Yourname> branch)

|  |
| --- |
| $ ls (Linux) or cd (Windows) |

1. Switch back to the master branch and list the files again. Compared to the previous result.

|  |
| --- |
| $ git checkout master  $ ls (Linux) or cd (Windows) |

1. Push the newly created branch and the changes to GitHub

|  |
| --- |
| $ git push --set-upstream origin dev\_<Yourname> |

1. Go to GitHub and see the branches.
2. Pull new branches created by your teammates into your local repository

|  |
| --- |
| $ git fetch  $ git branch -a |

1. Switch to the new branch

|  |
| --- |
| $ git checkout -b <branchname> origin/branchname |

1. Check the files in the branch and compare them to the master branch.

|  |
| --- |
| ls (Linux) or cd (Windows) |

1. Show all the branches

|  |
| --- |
| git branch -a |

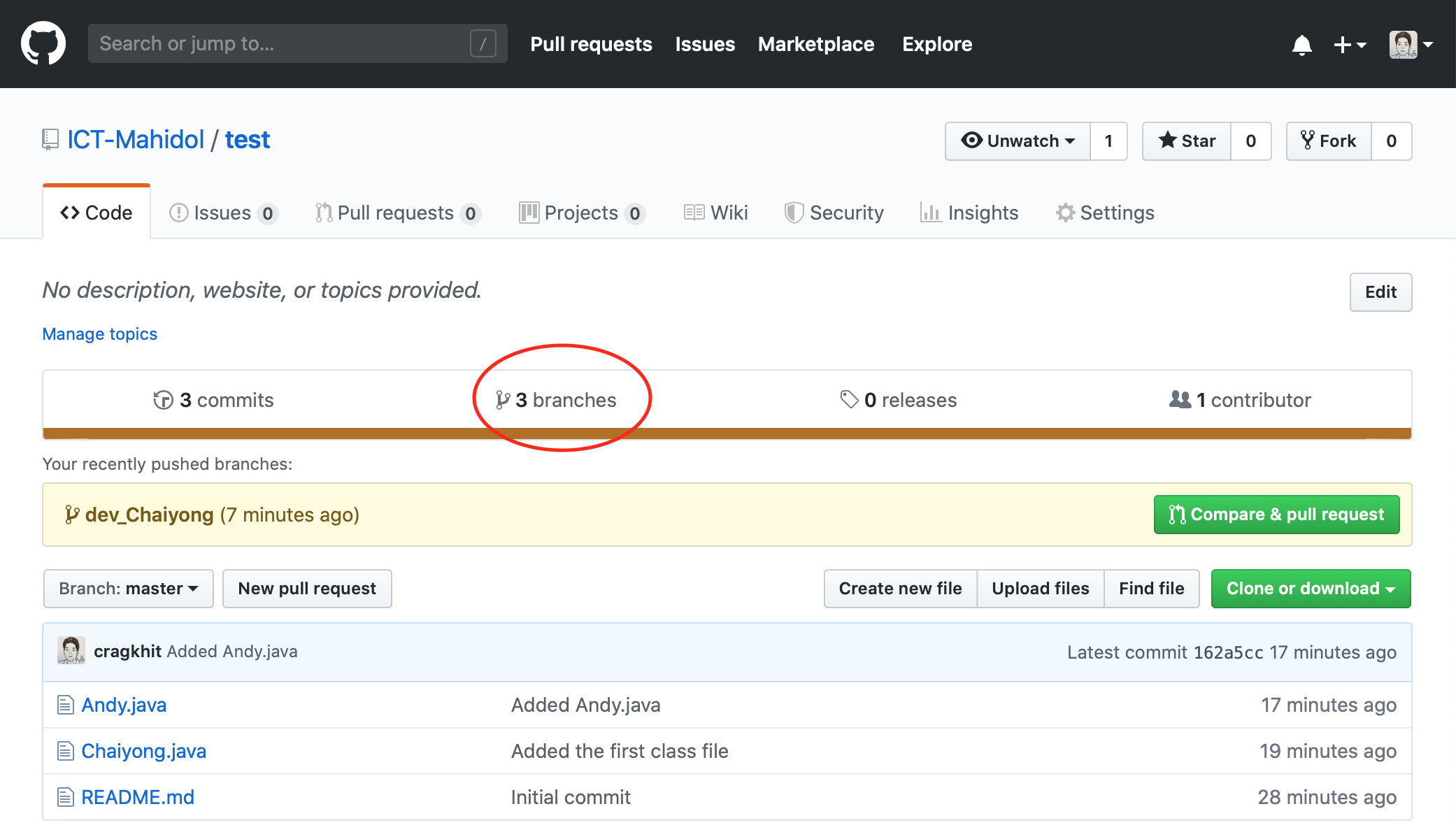
1. Save a screenshot of your screen with the command prompt at step 13 and paste below.

|  |
| --- |
|  |

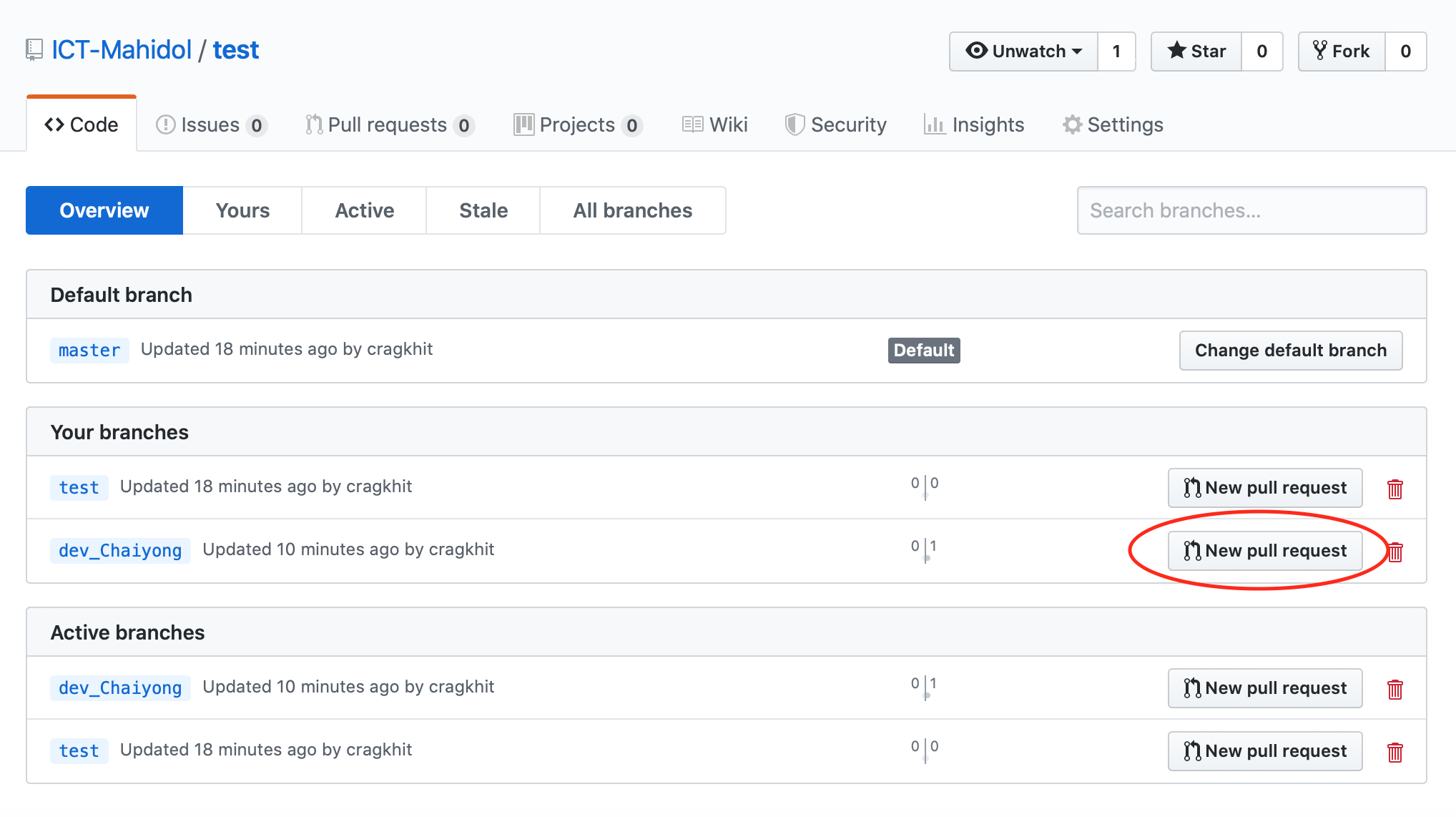
**5. Merging and Code Review (Pull Request)**

Now we are ready to merge the new feature made by each teammate into the master branch. Before merging the change, we will perform a code review first.

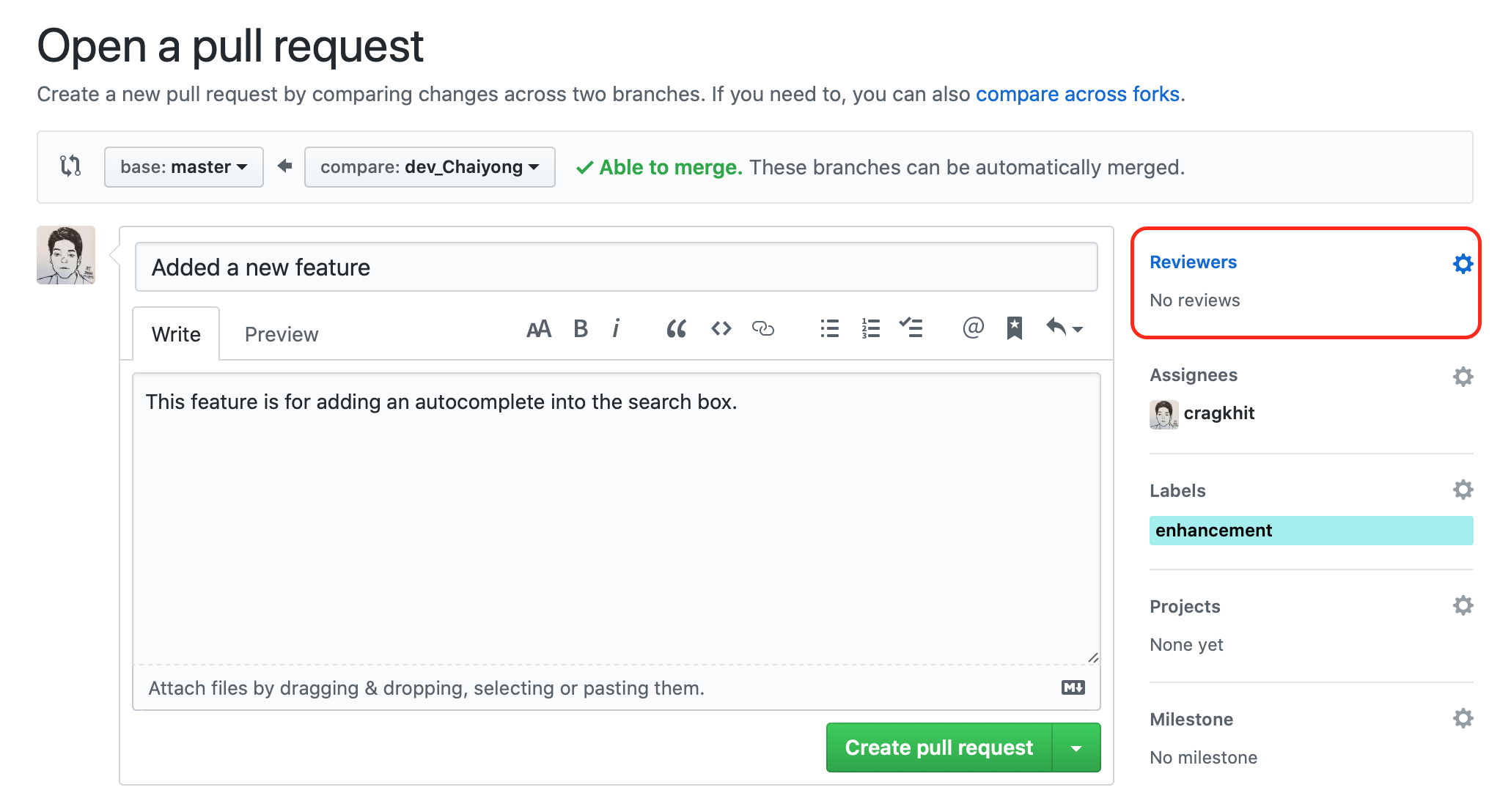
1. Go to your repository on GitHub website. Select “Branches”



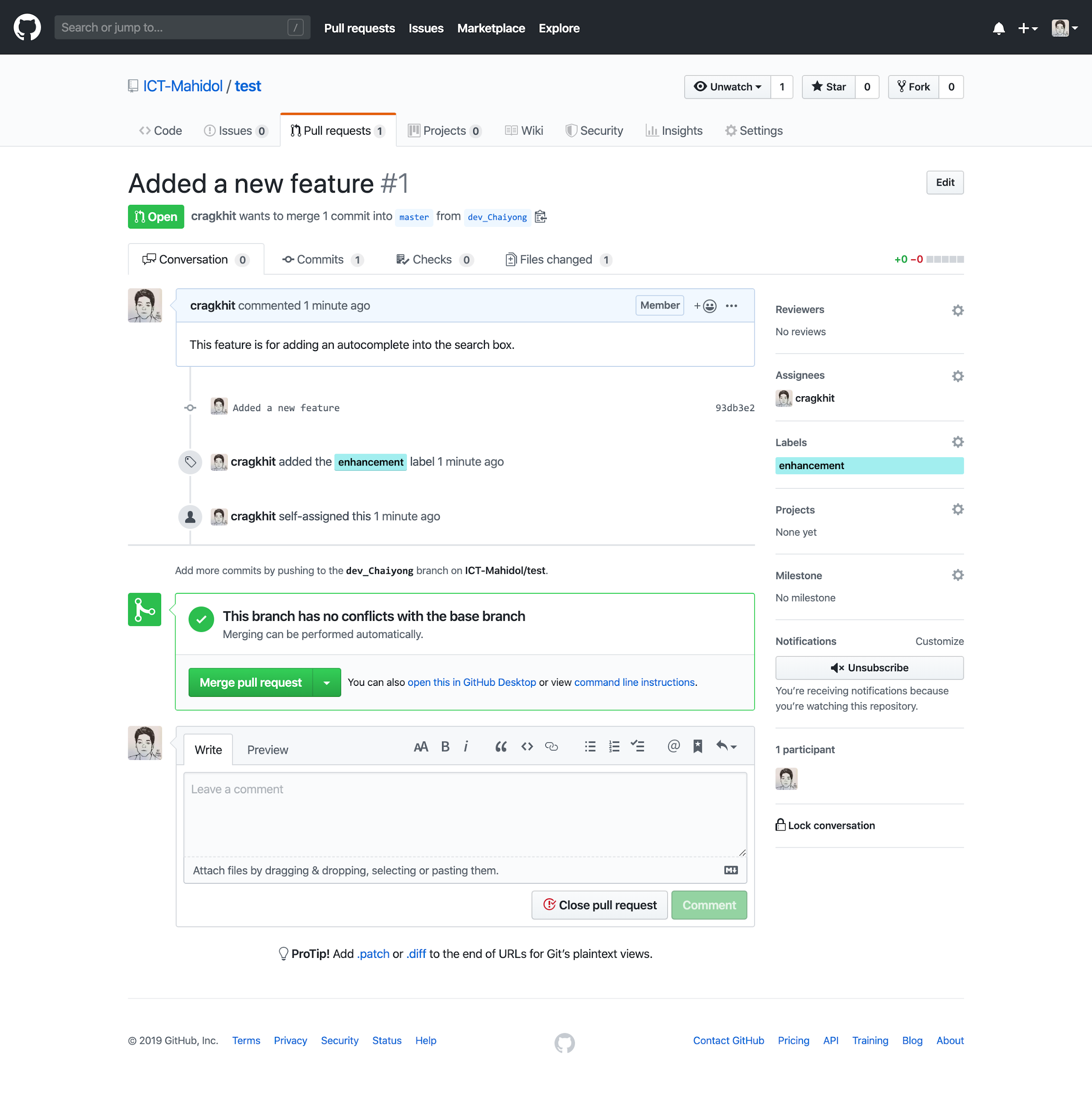
1. Select your branch and select “New pull request”. Every team member must do this for their own branch.



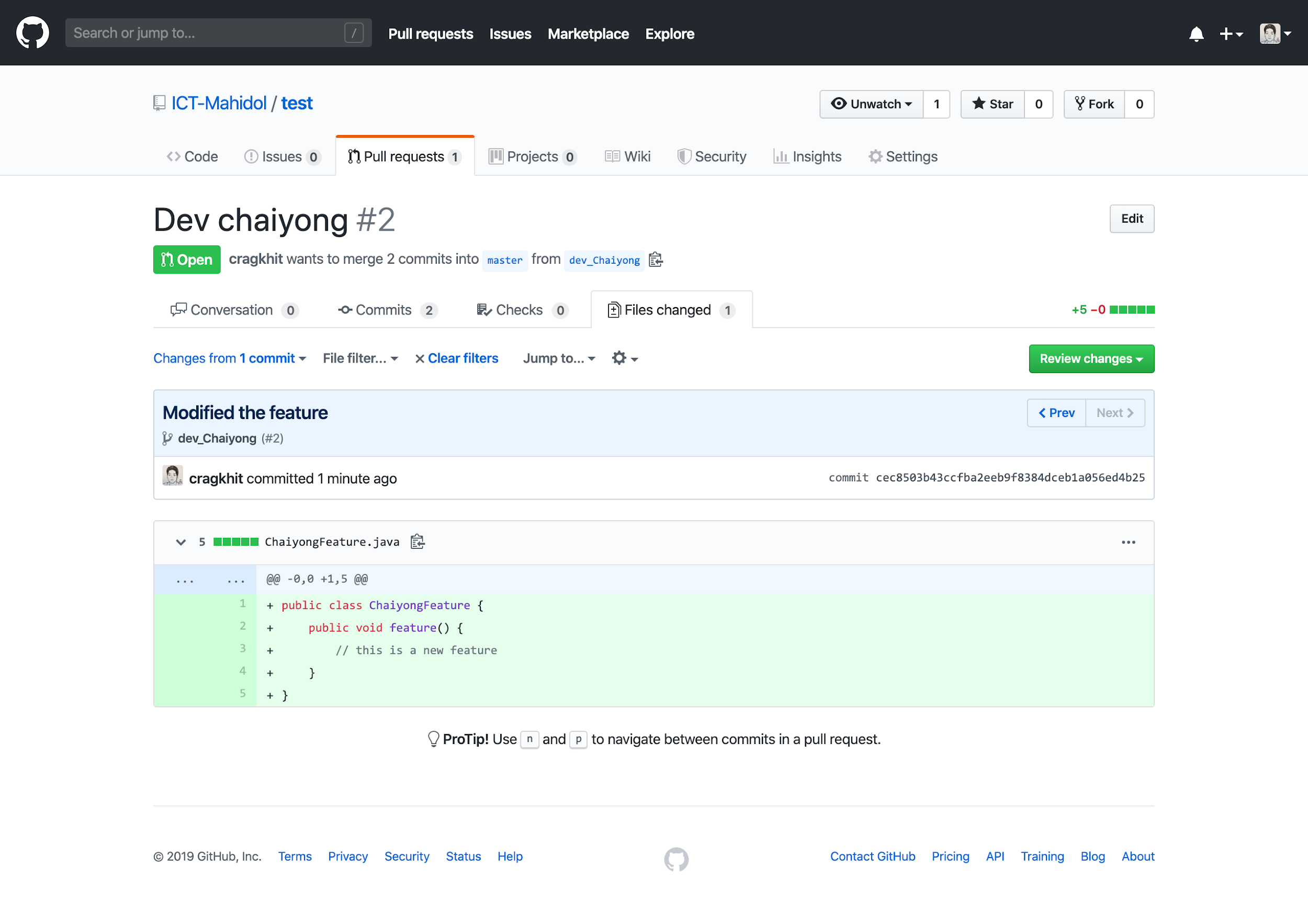
1. In the Pull Request page, fill in the description of the pull request in order to help the reviewer during code review. Then, assign the pull request to all the team members and select other properties for the pull request. Finally, press the “Create pull request” button.



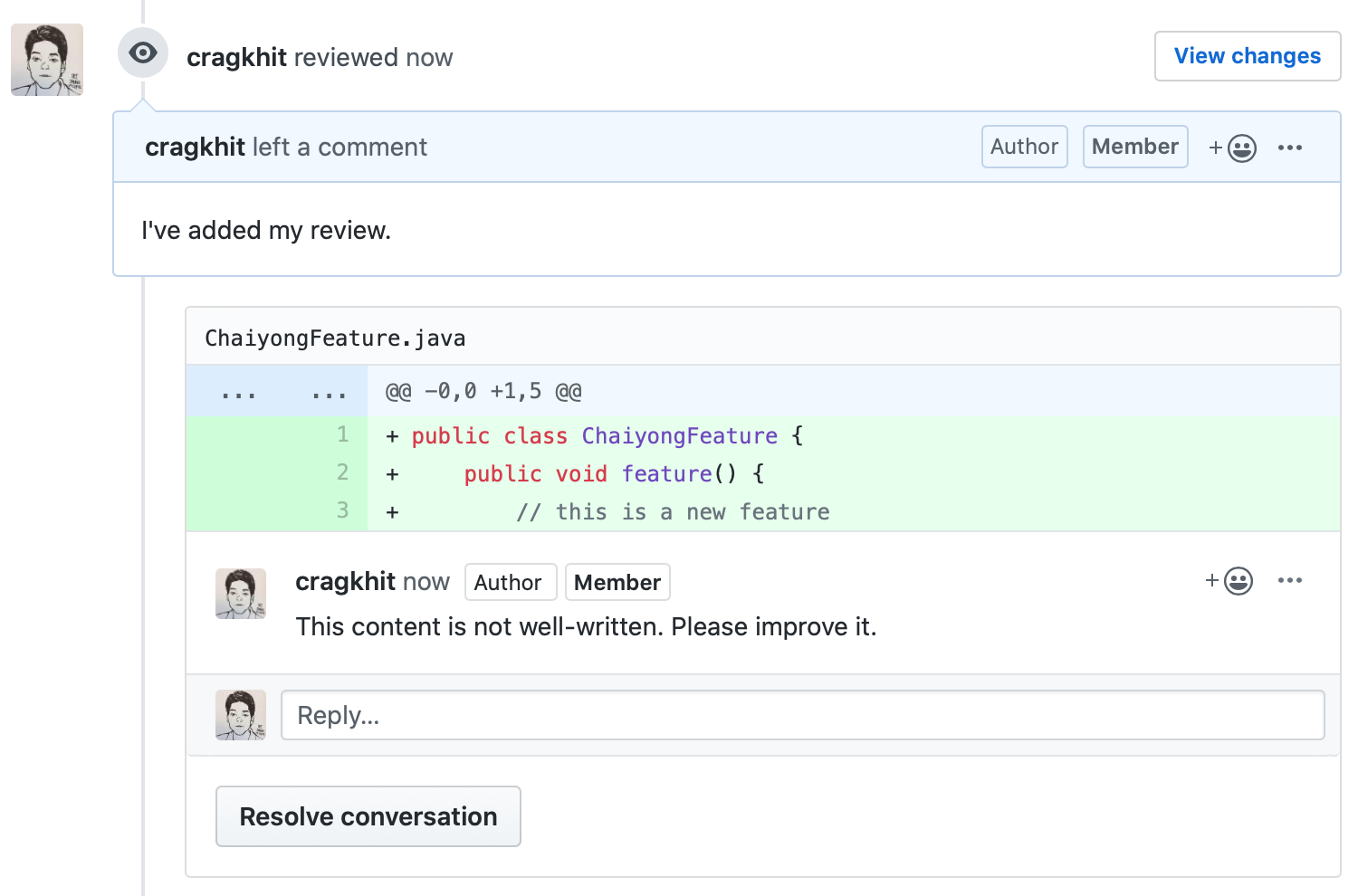
1. The new pull request (i.e., code review) will be created.



1. Click on a commit ID to see the changes and make a review.



1. Click on the ‘+’ sign in front of any line to add the review comment. Then, submit the review.



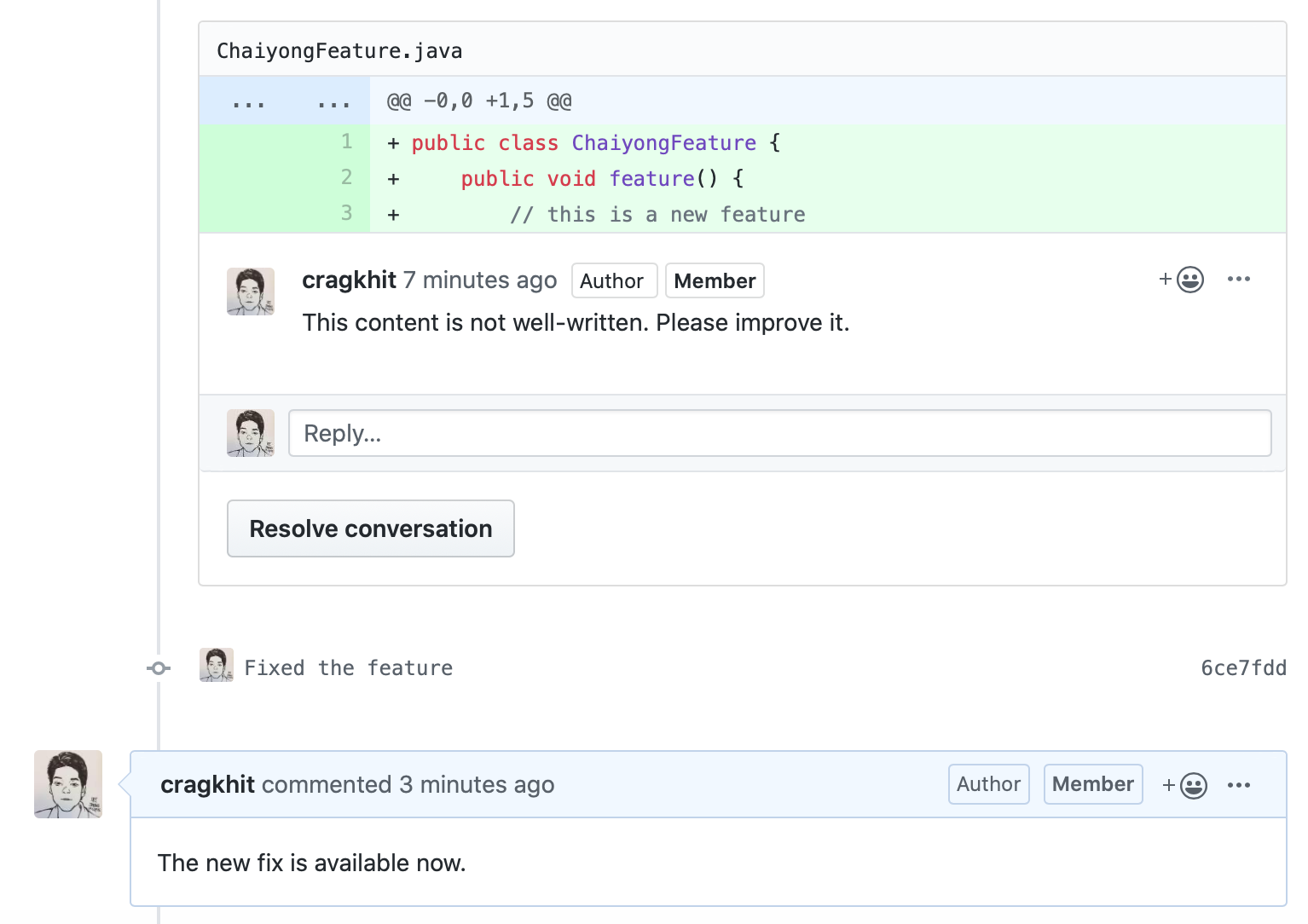
1. \*\*\* Every team member must give one review comment in every pull request. If not, you won’t get a score for this exercise. \*\*\*
2. The owner of the branch comes back to the local repository and modifies the file as follows.

|  |
| --- |
| public class <Yourname>Feature {  public void feature() {  // this is a new feature  int x = 0;  x = x+1;  }  } |

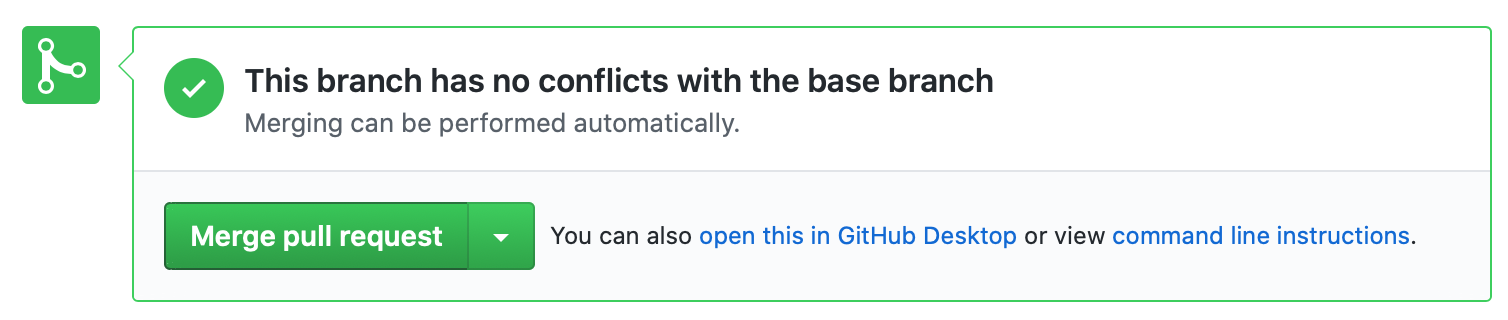
1. Commit and push the change to GitHub.

|  |
| --- |
| $ git add <Yourname>Feature.java  $ git commit -m “Fixed the feature”  $ git push |

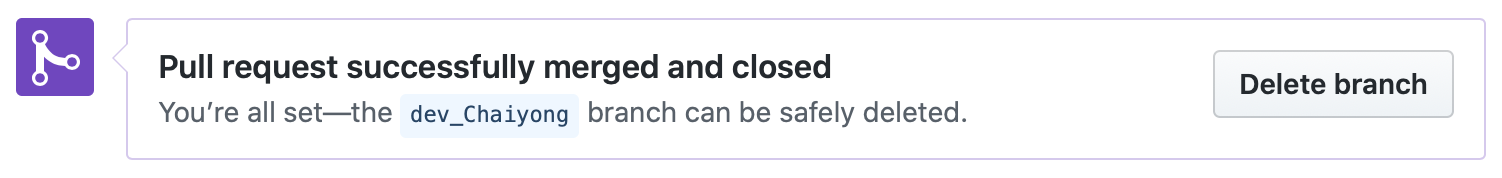
1. The owner of the branch comes back to GitHub. You’ll see the newly added commit in the pull request. Add a comment to let the reviewer know that the new fix is already available.



1. Everyone in the team reviews and comments again. If all the reviewers are satisfied with the change, click “Merge pull request” to integrate the new feature into the master branch.



1. Delete the development branch after merging



1. Each team member comes back to their local repository and fetches the updates.

|  |
| --- |
| $ git fetch -p  $ git branch -vv |

1. You can delete the branch that has already been merged from your local repository.
2. You do not need to upload screenshots for this part. We will check from your GitHub repository.

1. Some of the steps are copied from <https://www.atlassian.com/git/tutorials/install-git> [↑](#footnote-ref-1)