

# CS5010, Fall 2019

## Lab 3: Working with branches in GitHub

Abi Evans

[ab.evans@northeastern.edu](mailto:ab.evans@northeastern.edu)

### 1. Summary

So far this semester, you have likely been working in the “master branch” of your Github repo. In industry, work is never carried out directly in the master branch. Instead, engineers will create a “topic branch” to work on their assigned tasks. When work is complete and thoroughly tested, the topic branch can be merged into master. For the rest of this semester, you will adopt this workflow, writing all your code in topic branches then merging into master when you’re ready to submit your work.

In today’s lab, we will practice:

- Creating and working in branches in Github.
- Submitting pull requests.
- Reviewing pull requests.
- Merging approved pull requests.

There is nothing to submit for this lab although we strongly recommend that you complete this lab before starting work on HW 4, especially if you’re new to working with branches.

### 2. Working with branches

#### 2.1. Setup

You may have noticed a repo called “sandbox” on the course GitHub organization. Everyone involved in this class has write access to this repo. Today, you will use this repo to get some practice working with branches.

**Clone the repo to your machine** by running:

```
git clone https://github.ccs.neu.edu/cs5010seaF19/sandbox.git
```

#### 2.2. Create a branch

**Change into the directory created by the previous step:**

```
cd sandbox
```

Create a branch in your local repo by running the following command, replacing `your_username` with your Khoury GitHub username for all occurrences below:

```
git checkout -b your_username
```

You will see a message along the lines of:

```
Switched to a new branch 'your_username'
```

Creating a new branch effectively creates a copy of the code in the branch you are in when you run the command. For your homework assignments, you will name your branches after the assignments e.g. assignment4. You always want to be in the master branch when you create a new branch.

Run the command `git branch` to see a list of branches available in your local repo, as well as the currently selected branch. You will see output along the lines of:

```
*your_username  
master
```

The asterisk (\*) indicates the branch you are currently in.

The branch you created currently only exists in your local repo. **Push your branch to the remote repo** by running:

```
git push -u origin your_username
```

Open the [sandbox repo](#) in your browser. By default, the code listing shows the code contained in the master branch. Look for the dropdown menu that says “Branch: master”, shown in the bottom left of the following screenshot.



Use the dropdown menu to switch to your branch. You haven’t made any edits on your branch yet so it will contain the same code as the master branch.

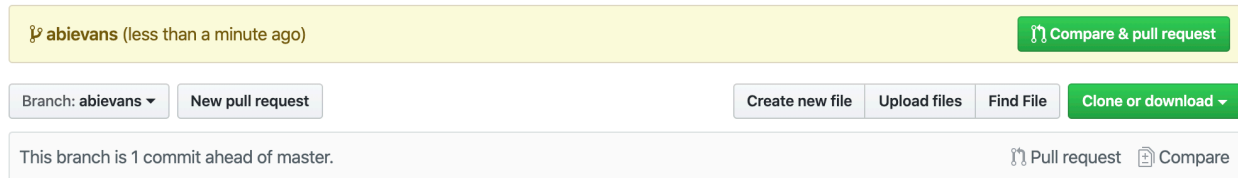
## 2.3. Add code to the branch

**In your local copy of the sandbox repo, create a folder named after your branch name. Write some Java code in the folder you created.** It doesn’t matter what you write for this lab—it doesn’t have to serve any particular purpose—but please remember that anything you post will be visible to everyone involved in the course. Do not post code related to assignments or other labs.

**Commit and push your changes as you would do normally.**

**Open the sandbox repo in your browser and switch to your branch.** If your branch is already selected, you may need to refresh the page to see your changes.

You should see something like the following:



Note the new button, “Compare & pull request”, and the message, “This branch is 1 commit ahead of mater.” The message means that your branch contains new code that is not in the master branch.

With your branch selected, you will see the code you pushed. However, if you switch to the master branch, note that the code you just pushed is not there. Code written on one branch will not exist in another branch until the two branches are merged.

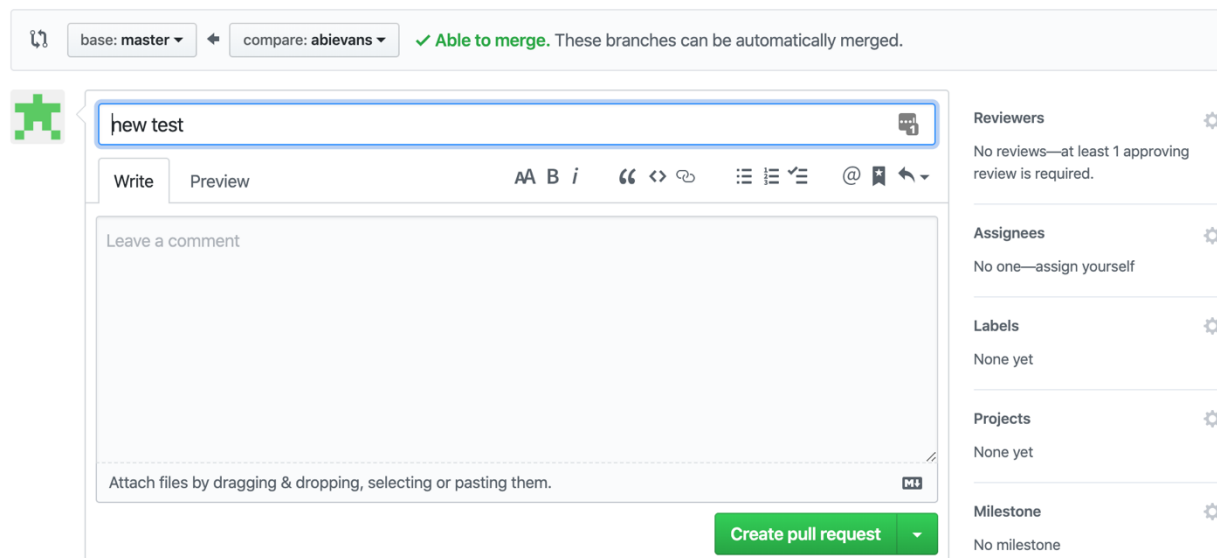
## 2.4. Create a pull request

To merge the code on your branch to master, you will initiate a “pull request”. This has to be done from the browser. **Click the green “Compare & pull request” button.**

This will take you to a new page that looks something like this:

### Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



By default, the title of the pull request will be whatever message you provided with your most recent commit. You do not need to change the title or add anything in the message box.

**Assign a reviewer by clicking the settings icon next to “Reviewers” in the right sidebar.** Search for a reviewer by their username then click on their username to assign them to your pull request. For assignments, you will tag your codewalk TA. For this lab, you may tag a friend in the course if you prefer.

Click **“Create pull request”**. The assigned reviewer will be notified.

**If you receive a review request from someone else, open up the review link, approve the merge and optionally leave them a comment.**

## 2.5. Merge the branch into master

**You will be notified when your pull request has been reviewed.** Assuming the request was approved, you can click the **“Merge pull request”** button, followed by the **“Confirm merge”** button.

At this stage, you can delete the `your_username` branch if you wish. Note that the button **“delete branch”** on the browser UI for Github only deletes the remote version of the branch. To delete your local branch, switch to the master branch by running `git checkout master`, then run `git branch -d your_username`.

**With the repo open in your browser, switch to the master branch.** You should now see all your code in the master branch.

## 2.6. Get the latest version of the master branch

**In your local repo, switch to the master branch:**

```
git checkout master
```

**...then use the pull command to get the latest version of the branch:**

```
git pull
```

# 3. Avoiding merge conflicts

If you experience a merge conflict, consult p17 of the GITrefresher PDF posted under “labs” on Piazza.

The best way to avoid merge conflicts is to ensure that you never work on code that belongs to another branch. Always check you are working on the correct branch before you begin a work session or after you have taken a break by running `git branch`. If you need to switch branches, you can do so by running `git checkout branch_name`, replacing `branch_name` with the name of the branch you want to switch to.

Before creating a new branch, ensure that any outstanding local changes are pushed to the remote repo. Then, run `git pull` from the master branch. This ensures that your local repo is up to date with the remote repo.

