GIT: From Beginner To Fearless

Basic Branching and merging with a pull request

Brian Gorman, Author/Instructor/Trainer

Introduction

We've made some strides to this point, learning about branches and seeing how we can then merge the changes from one local branch to another. However, as mentioned in the basic merging activity, it's dangerous to merge directly to master on the local repository and then push to the remote master branch. If there are two-or-more users working against the repository, each of the users will be dependent on a common history tree for master. Once any user changes this history tree, the other users would have to resolve any conflicts that could arise from changes to that history. In general, simple merging won't cause too many problems, but other advanced operations could.

To help protect our code and prevent history issues, GitHub and other online repositories have the ability to create 'pull requests,' which allow for changes to be merged at REMOTE from one branch to another, including to and from the master branch. Merging via a pull request (PR) is a much safer way to make sure that changes are not lost and that public history is not corrupted for other users of the repository.

In this activity, we'll take a look at making local changes and pushing the branch to GitHub, then merging with a pull request. After the pull request is completed, a merge commit will be created at master. In order to keep our local repository in sync, we'll finish the activity by performing the fetch and pull operations to update our local repository from the remote repository.

Let's get started!

Step 1: Make sure your local repository is up to date

a) Get the latest version of the remote master

Perform the operations that follow to make sure our current local repository is in sync.

[**git checkout master**]

[**git fetch origin**]

[**git pull origin master**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git checkout master
Already on 'master'
Your branch is up-to-date with 'origin/master'.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git fetch origin

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git pull origin master
From https://github.com/majorguidancesolutions/defaultweb_activity
 * branch            master     -> FETCH_HEAD
Already up-to-date.
```

b) Create a local branch and push the branch to GitHub.

[**git checkout -b feature-branch-one**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git checkout -b feature-branch-one
Switched to a new branch 'feature-branch-one'
```

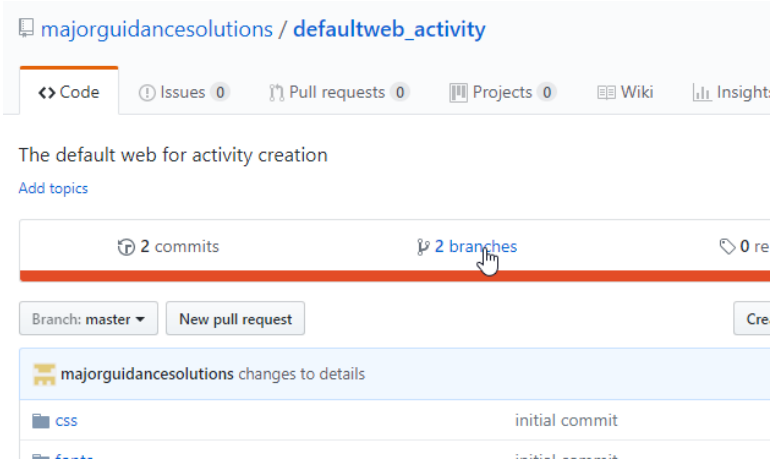
Now push the branch using the -u flag since it is unpublished:

[**git push origin -u feature-branch-one**]

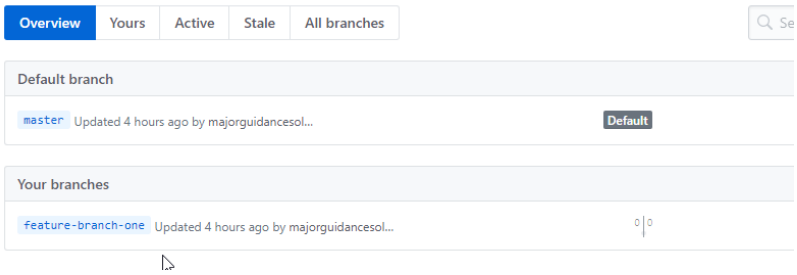
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (feature-branch-one)
$ git push origin -u feature-branch-one
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/majorguidancesolutions/defaultweb_activity.git
 * [new branch]      feature-branch-one -> feature-branch-one
Branch feature-branch-one set up to track remote branch feature-branch-one from
origin.
```

Notes

Validate the branch was pushed. At GitHub, select the 'branches' tab:



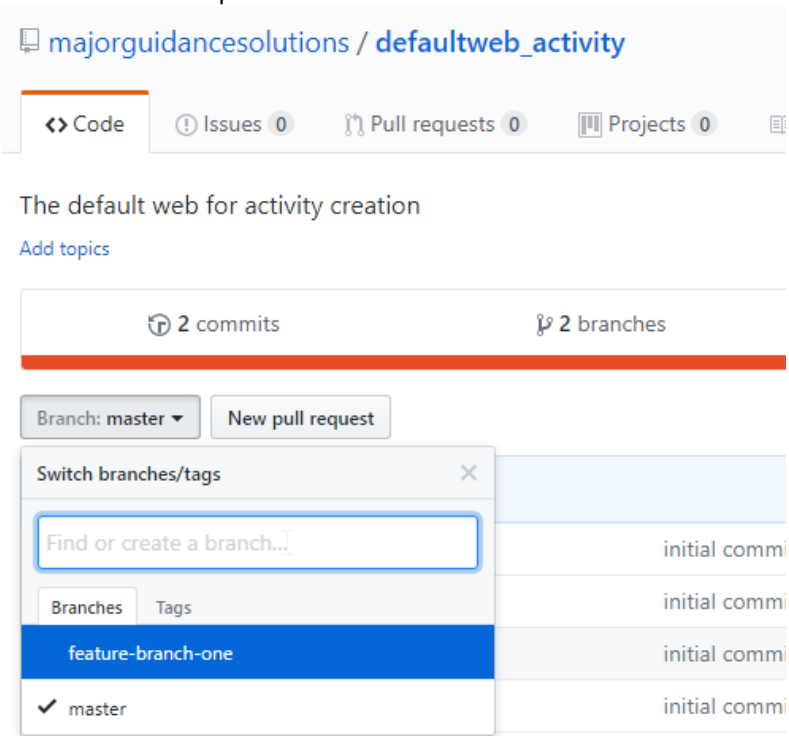
Validate the new branch is there:



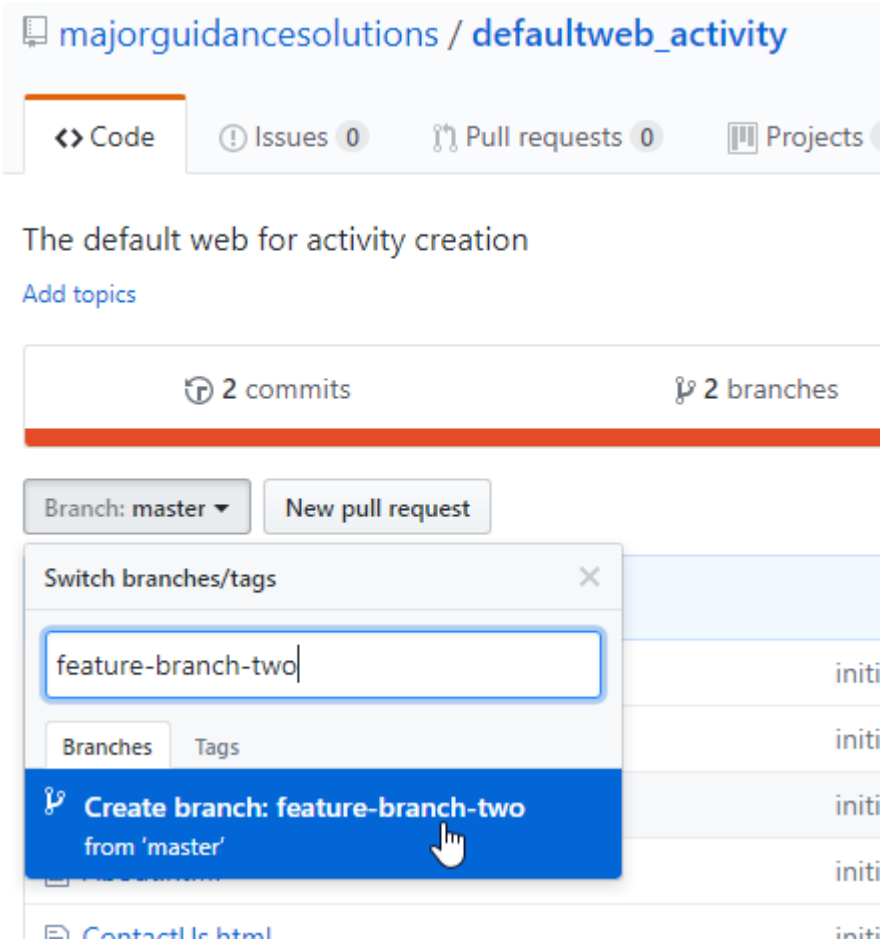
Step 2: Create a feature branch at GitHub

a) Log in to GitHub and browse to your repository

After getting to the correct repository, locate the "Branch:" dropdown (which should be pointing to master. Select this dropdown and enter a new branch name in the empty box. Also note, we could switch to our other branch(es) that are listed in the dropdown if we would like:

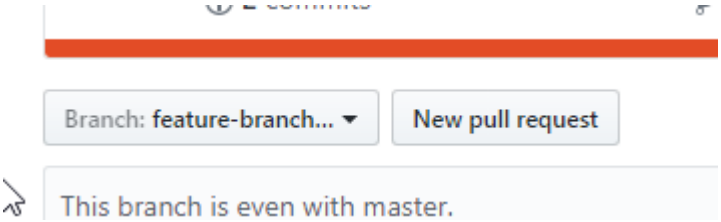


b) Enter the branch name into the dropdown

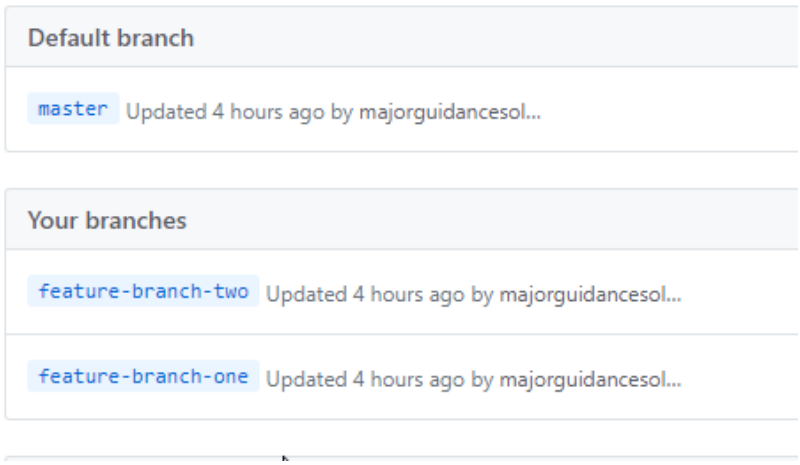


Then select the "Create Branch" button

Note that the new branch is selected in the dropdown:



And validate the branch is listed:



Step 3: Bring the latest changes back to our local repo

a) Now that we created a new branch at GitHub, we need to get it locally

[git checkout master]

[git fetch origin]

[git pull origin]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (feature-bran
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git fetch origin
From https://github.com/majorguidancesolutions/defaultweb_activity
 * [new branch]      feature-branch-two -> origin/feature-branch-two

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git pull origin
Already up-to-date.
```

[git branch -a]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git branch -a
  My-Feature-Branch
  feature-branch-one
* master
  remotes/origin/feature-branch-one
  remotes/origin/feature-branch-two
  remotes/origin/master
```

b) Checkout the feature branch to create a local reference to it

[git checkout feature-branch-two]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git checkout feature-branch-two
Switched to a new branch 'feature-branch-two'
Branch feature-branch-two set up to track remote branch feature-branch-two from
origin.
```

Note that we didn't need to use the -b flag to create the branch since a branch ref to remote already existed. The new branch is already setup and tracks to remote as expected.

Step 4: Make some changes on either feature branch, but only use one of them [for now], then push and merge at GitHub

a) Choose a branch to work on. For example, feature-branch-one

Make some small changes in the details.html file.

[**git checkout feature-branch-one**]

[**code details.html**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (feature-branch-two)
$ git checkout feature-branch-one
Switched to branch 'feature-branch-one'
Your branch is up-to-date with 'origin/feature-branch-one'.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (feature-branch-one)
$ code details.html
```

[you can make any changes you would like]

```
    <h2> I'm making some changes! </h2>
    <h3> Changes made on feature-branch-one for merge at GitHub</h3>
    <hr />
    <footer>
```

Add and commit your changes

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (feature-branch-one)
$ git commit -am 'Changes on feature-branch-one'
[feature-branch-one 7c4f044] Changes on feature-branch-one
 1 file changed, 1 insertion(+)
```

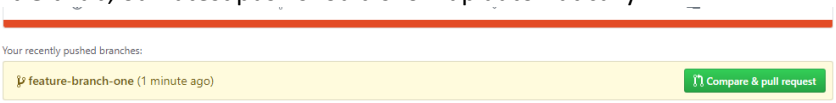
b) Push the changes out to GitHub [do not merge to the local master!]

[**git push origin feature-branch-one**]

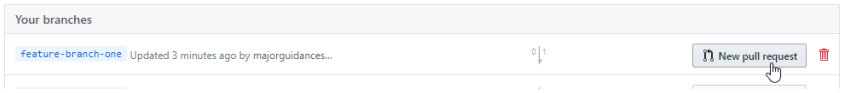
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (feature-branch-one)
$ git push origin feature-branch-one
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 378 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/majorguidancesolutions/defaultweb_activity.git
   7b6a932..7c4f044  feature-branch-one -> feature-branch-one
```

c) Create a pull request at GitHub

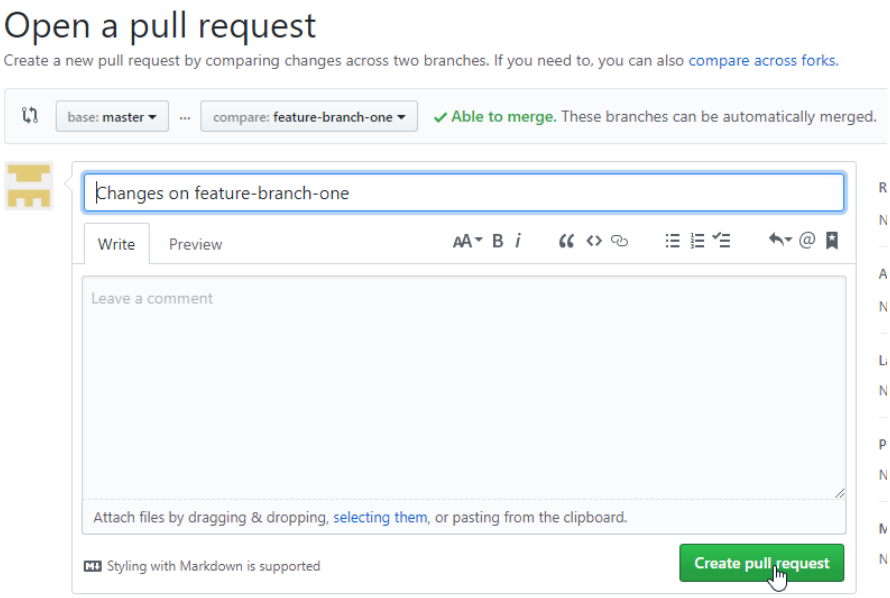
At GitHub, our latest push should show up automatically:



We can click on 'compare & pull request' here. We can also browse to the branch and select 'new pull request'. Also note that on the branch view, we can see that this branch is 1 commit ahead of master and 0 commits behind

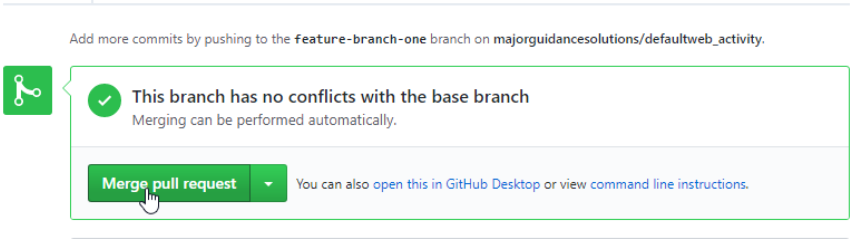


Click on either of the two buttons, and then click “Create Pull Request”



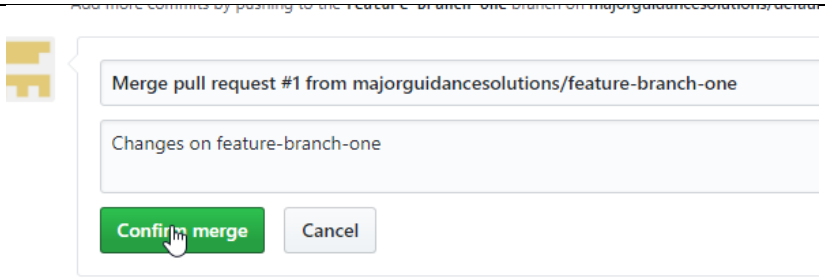
d) Merge the pull request

Click ‘Merge pull request’



And “Confirm Merge”

Also note, you could put details in here and change the commit message. We are creating a new commit in history for the actual merge, so that message will show up in the history (i.e. Merge pull request #1…)



Unless there are specific reasons, it's generally a good idea to delete the branch once it's been merged.

Step 5: Sync the changes locally to make sure our history is up to date

[git checkout master]

[git log --oneline]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity
$ git log --oneline
7b6a932 (HEAD -> master, origin/master, origin/feature-b
h-two, My-Feature-Branch) changes to details
b7edba6 initial commit
```

Even though it says we are up to day, we are not!

[git fetch origin]

[git pull origin master]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git fetch origin
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
From https://github.com/majorguidancesolutions/defaultweb_activity
   7b6a932..69f03ea  master     -> origin/master

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git pull origin master
From https://github.com/majorguidancesolutions/defaultweb_activity
 * branch            master     -> FETCH_HEAD
Updating 7b6a932..69f03ea
Fast-forward
 details.html | 1 +
 1 file changed, 1 insertion(+)
```

[git log --oneline]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git log --oneline
69f03ea (HEAD -> master, origin/master) Merge pull request #1 from majorguidance
solutions/feature-branch-one
7c4f044 (origin/feature-branch-one, feature-branch-one) Changes on feature-branc
h-one
7b6a932 (origin/feature-branch-two, feature-branch-two, My-Feature-Branch) chang
es to details
b7edba6 initial commit
```

Now we are up to date!

Step 6: Cleanup

a) Optional cleanup

At this point, we could do some cleanup. For example, we could delete the feature branch locally and at GitHub. Or, if we wanted, we could keep working on the feature branch, but the best step would be then to merge or rebase origin/master into the feature branch first, in order to keep the history up to date.

Feel free to practice cleaning up the branches that have been merged. We'll continue to learn best practices and strategies for working with our remote and local repositories as we continue to work with GIT.

This concludes our Branching and Merging with a Pull Request Activity

Closing Thoughts

During this activity, we have seen how we can create our changes locally on a feature branch, and then push those changes at the branch level to GitHub. We then saw how we can perform a basic merge with a pull request at GitHub. Once that was done, we needed to pull the changes back to our local to make sure we are in sync going forward.

Ordinarily, there would be some cleanup after this, such as deleting the branches at both remote and local. If you would like to cleanup the feature-branch-one now, please feel free to do so. We know it is incredibly easy to create branches locally, so we can always create a new version of the branch.

Take a few minutes to make some notes about the various commands we've learned about in this activity, and practice using them.

Notes