GIT: From Beginner To Fearless

GIT Merging Activity:
A simple multiple-developer merging exercise

Brian Gorman, Author/Instructor/Trainer

Introduction

We've created a feature branch, and now we want to make those changes permanent. How do we do this? With a MERGE!.

Merging is important because this allows us to get our repository master history updated with the changes that we want to keep which we've created on a feature branch. Eventually, we'll learn that with multiple people working on the repository, conflicts will arise that we'll need to resolve. However, once we have merging under our belt, conflict resolution is just another easy step towards mastering our fears and working with GIT.

In this activity, we rely on changes from the branching activity. If you haven't done that activity, please complete it now. We're going to pick up where we left off on that activity, and merge our changes to our master locally, and then push them out to GitHub.

Let's gets started!

Step 1: Make sure we have changes to merge

a) Check to see that we have changes to merge.

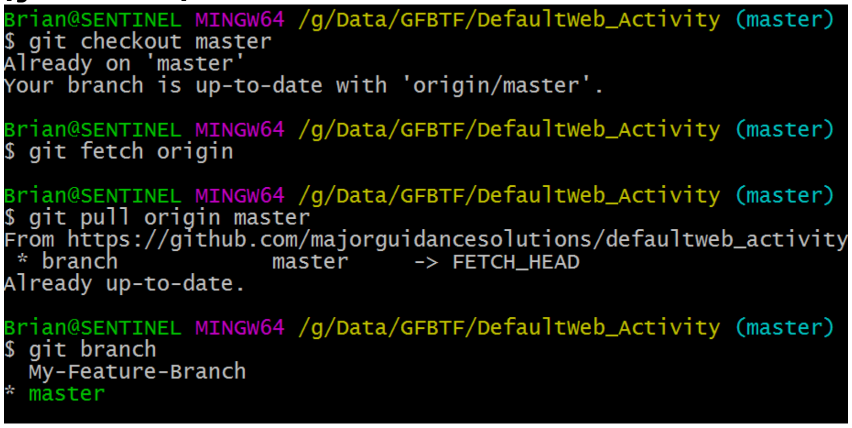
Open our GIT BASH terminal, and review branches. We should have at least two, master, and MY-Feature-Branch. If we don't have these, make sure to complete the branching activity before continuing:

[**`git checkout master`**]

[**`git fetch origin`**]

[**`git pull origin master`**]

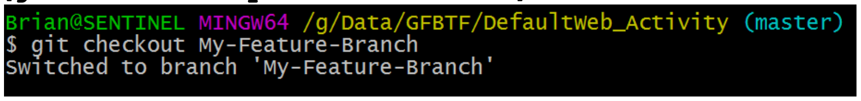
[**`git branch`**]



```
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.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git branch
  My-Feature-Branch
* master
```

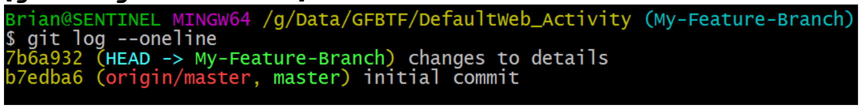
b) Switch to the branch that has changes

[**`git checkout My-Feature-Branch`**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git checkout My-Feature-Branch
Switched to branch 'My-Feature-Branch'
```

[**`git log --oneline`**]



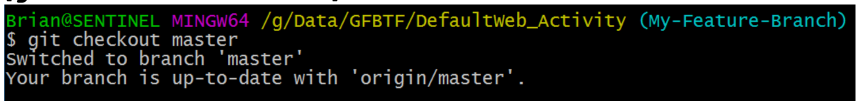
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (My-Feature-Branch)
$ git log --oneline
7b6a932 (HEAD -> My-Feature-Branch) changes to details
b7edba6 (origin/master, master) initial commit
```

We see that we have at least one commit to merge.

Step 2: Merge the changes

a) Switch to master.

[**`git checkout master`**]



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

Notes

Now, before we do this, I want to say something. Ordinarily you don't want to do things this way. We should ***~~never~~ almost never*** modify master directly – instead, use feature branches, push to a remote like GitHub or BitBucket, and then utilize pull requests to merge code.

However, for this demo, we're going to change master locally, and push it to directly to GitHub.

Again, if you are working on a team, this would be very dangerous, especially if someone else merged changes into master first. If you are by yourself, there is less risk involved in working with this workflow.

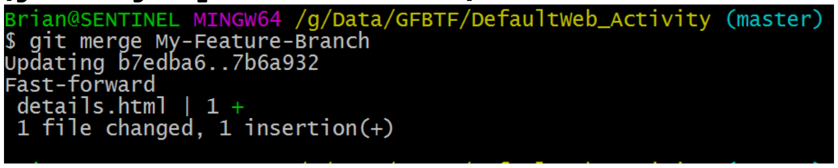
b) Merge the changes locally

Make sure you are on master.

[**`git checkout master`**]

Then merge your feature branch

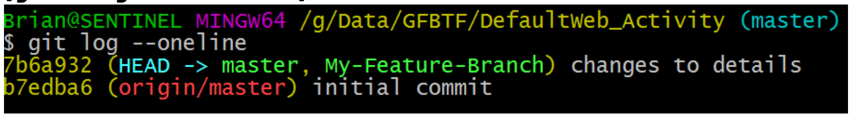
[**`git merge My-Feature-Branch`**]



```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git merge My-Feature-Branch
Updating b7edba6..7b6a932
Fast-forward
 details.html | 1 +
 1 file changed, 1 insertion(+)
```

So now my local master is one commit ahead of origin, has the same commit as the feature branch did:

[**`git log --oneline`**]



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

Note how HEAD -> Master is one ahead of origin/master and is even with My-Feature-Branch

[we can now safely delete My-Feature-Branch locally]

Step 3: Push your changes to GitHub

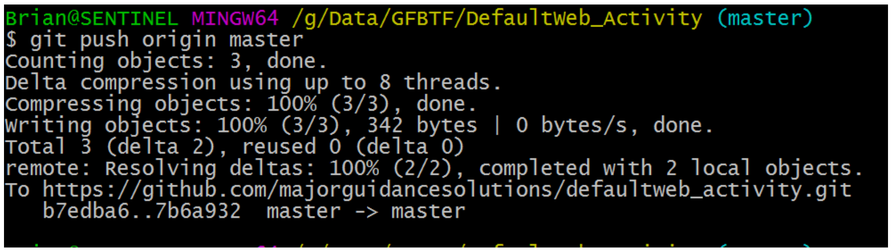
a) Pushing the changes

Now that we have our changes committed locally, we need to get the changes out to GitHub. This will make sure that our REMOTE repository has the changes and we will never have to fear losing them.

To push master, we don't need to use the -u flag, because the branch is already present at REMOTE. The -u flag is only needed when a branch is untracked.

Again: please don't do this on a team project:

[**`git push origin master`**]

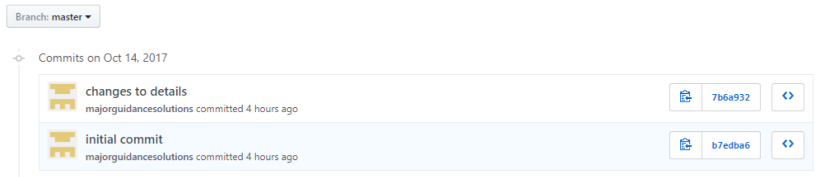


```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git push origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 342 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/majorguidancesolutions/defaultweb_activity.git
   b7edba6..7b6a932  master -> master
```

b) Verify changes at GitHub

Browse to the repo and validate that the file we modified has been updated at the REMOTE repository.

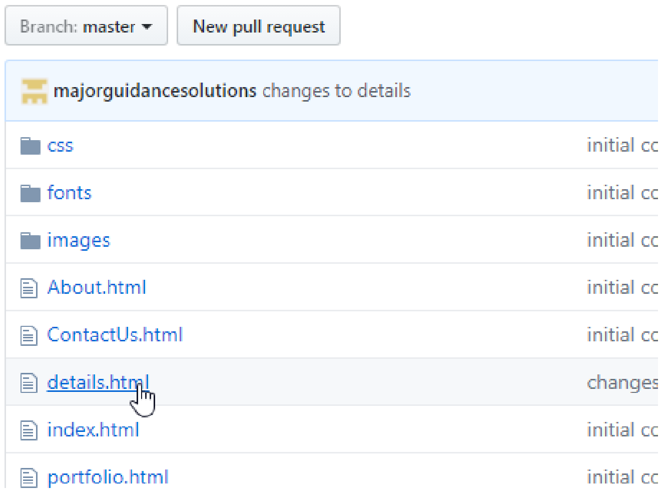
Check the commit history -> the most recent commit should have the message that we entered on our last commit and have the same commit id that we saw a few times in the git log oneline



Commits on Oct 14, 2017

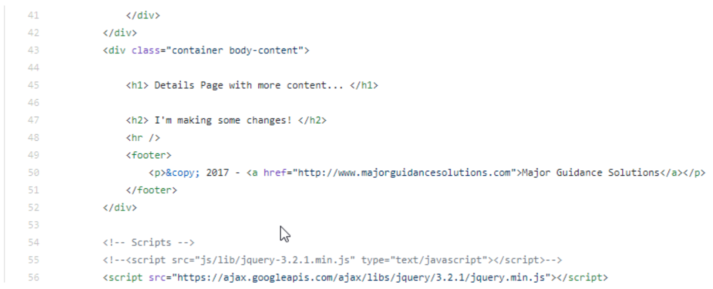
changes to details
majorguidancesolutions committed 4 hours ago 7b6a932

initial commit
majorguidancesolutions committed 4 hours ago b7edba6



majorguidancesolutions changes to details

css	initial cc
fonts	initial cc
images	initial cc
About.html	initial cc
ContactUs.html	initial cc
details.html	changes
index.html	initial cc
portfolio.html	initial cc



```
            </div>
        </div>
        <div class="container body-content">

            <h1> Details Page with more content... </h1>

            <h2> I'm making some changes! </h2>
            <hr />
            <footer>
                <p>&copy; 2017 - <a href="http://www.majorguidancesolutions.com">Major Guidance Solutions</a></p>
            </footer>
        </div>

        <!-- Scripts -->
        <!--<script src="js/lib/jquery-3.2.1.min.js" type="text/javascript"></script>-->
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
```

This concludes our merging activity.

Closing Thoughts

In this initial merging activity, we see how we can move our changes from our local feature branch to the local master branch. After we merged the changes to master, we then pushed our changes out to the remote repository at GitHub.

Since the changes were already in master, no merging was needed at GitHub, and therefore no new commits were generated -> the changes were just updated. This is ok when we are the only person modifying the repository, but if there were multiple people interacting with the repository, this would actually be very dangerous. As we continue to learn through this course, we'll see better ways to go about getting our changes out to the remote repository.

Also, for almost every single case, we should try to avoid directly merging to the master like we did in this activity. Again, this is ok if we are the only person interacting with the repo, but for a team of people, history conflicts could become a very difficult problem to solve.

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

Notes

____

____

____

____

____

____

____

____

____

____

____

____