

# GIT: From Beginner To Fearless

## Team Branching and Merging With a Pull Request and Conflict Resolution

Brian Gorman, Author/Instructor/Trainer

©2019 - MajorGuidanceSolutions



# Introduction

---

If you've followed along to this point, we are in a really good place with understanding how to get changes out to the repository at GitHub (or other REMOTE). However, we've only taken the happy path so far – where only one user is pushing and there have been no conflicts.

In the real world, for most of us who will be using GIT, conflicts will be a regular occurrence, and resolving them while merging will become part of the necessary routine.

In this activity, we're going to simulate how a merge conflict might happen in a team environment. To keep this very simple, we'll be performing all changes ourselves, so we'll have to pretend some of the changes come from another developer. If you really want to take it to the next level, you could go to the level of setting up an organization, creating a second account and using a team repository for your two accounts. However, to simulate the team for this activity, we aren't going to go that deep.

The general flow of what we'll do is to get the current version of the repository to our local machine. We'll then make some changes on a feature branch on our machine, and push them up to GitHub. In the meantime, we'll create a branch at GitHub and make a change there which will conflict with our changes. We'll merge the other developer's changes into master first, and then see what it would take to resolve the conflict on our local machine for our feature branch.

Let's get started!

# Team branching and merging - pull request and merge conflict

## 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 developer-one-branch]
```

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

Now push the branch using the -u flag since it is unpublished:

```
[git push origin -u developer-one-branch]
```

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

### Notes

## Step 2: Make some changes on your local branch and push

### a) Create a simple change

[code details.html]

Enter a change similar to the following:

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

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

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

    <h1>These are developer one's changes, created during team simulation</h1>

    <hr />
    <footer>
    |   <p>&copy; 2017 - <a href="http://www.majorguidancesolutions.com">Major Guidance Solut
    </footer>
</div>

<!-- Scripts -->
```

### b) Make sure you have saved and have changes to commit:

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch)
$ git status -s
M details.html
```

### c) Add, Commit, Push

[git commit -am 'Developer one critical changes']

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch)
$ git commit -am 'Developer one critical changes'
[developer-one-branch 8d80062] Developer one critical changes
1 file changed, 3 insertions(+)
```

[git push origin developer-one-branch]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch)
$ git push origin developer-one-branch
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 399 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
69f03ea..8d80062 developer-one-branch -> developer-one-branch
```

### d) Review current commit history

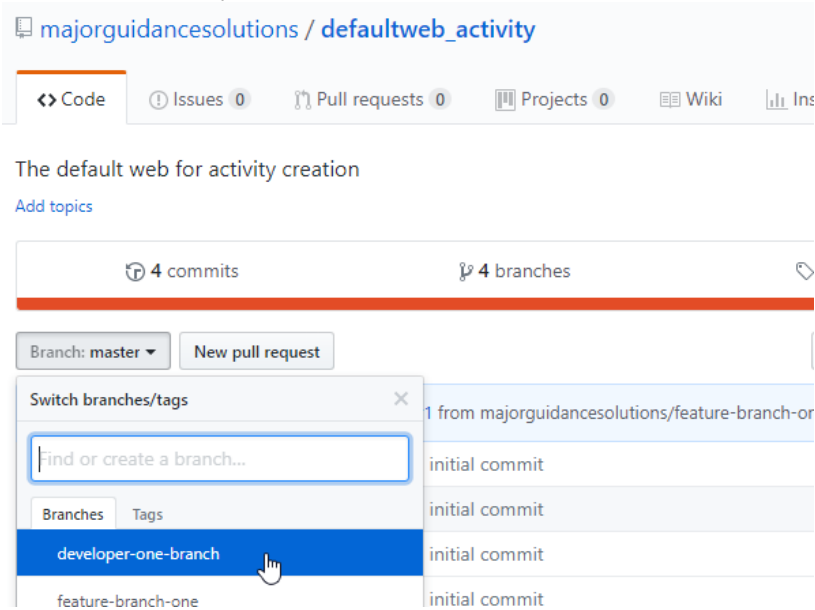
[git log --oneline]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch)
$ git log --oneline
8d80062 (HEAD -> developer-one-branch, origin/developer-one-branch) Developer one critical changes
69f03ea (origin/master, master) Merge pull request #1 from majorguidancesolutions/feature-branch-one
7c4f044 (origin/feature-branch-one, feature-branch-one) changes on feature-branch-one
7b6a932 (origin/feature-branch-two, feature-branch-two, My-Feature-Branch) changes to details
b7edba6 initial commit
```

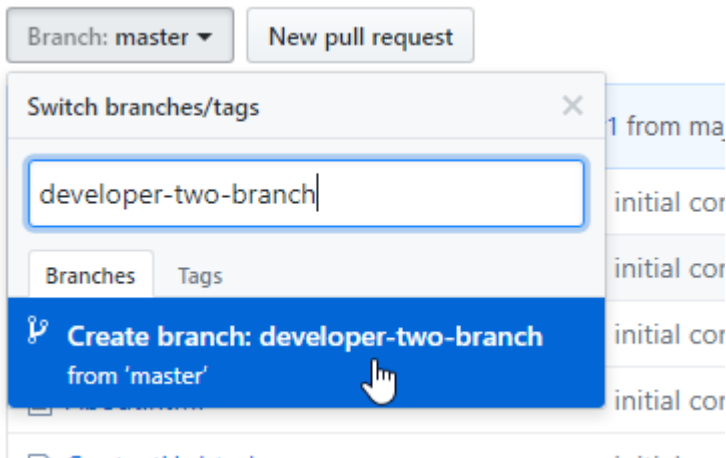
Note your recent commit id [mine is 8d80062].

### Step 3: Create a feature branch at GitHub then Merge a change from it directly 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:

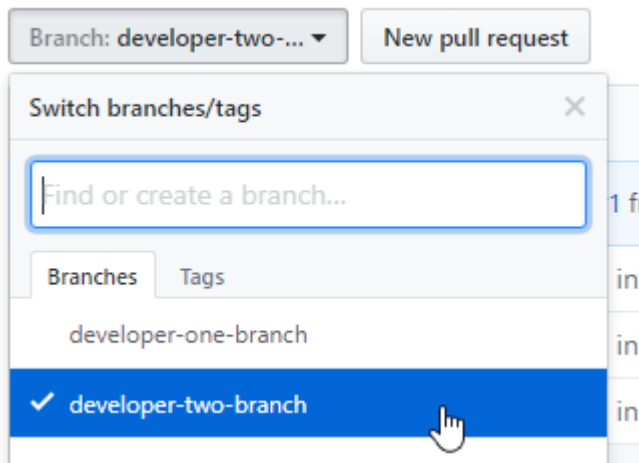


- b) Enter the branch name into the dropdown  
[developer-two-branch]

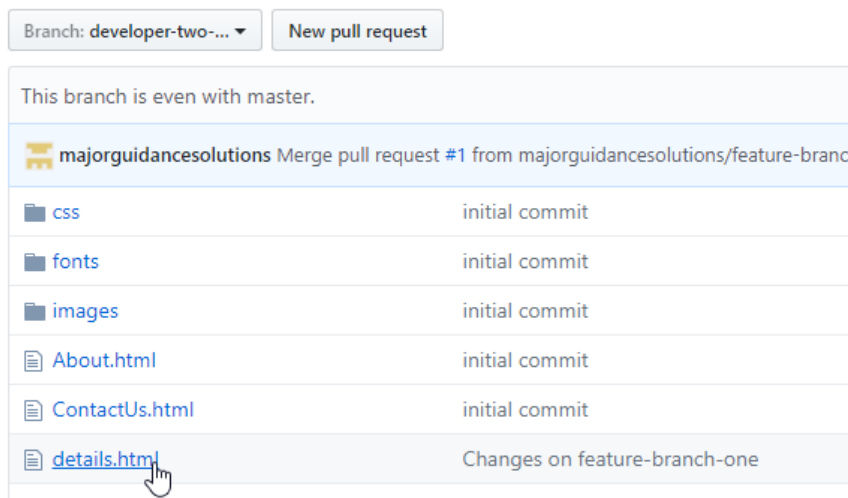


Then select the “Create Branch” button – making sure it says “from ‘master’”

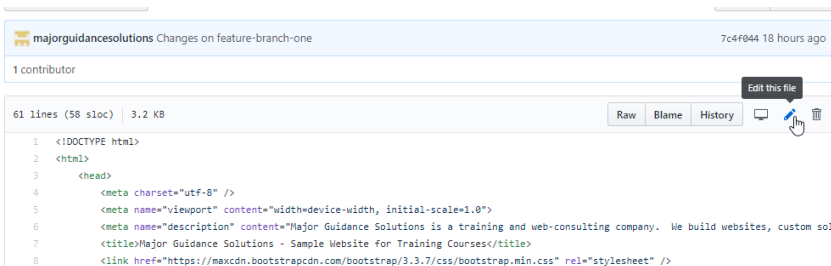
- c) Make sure you are on the developer two branch and make a change  
Make sure you have your dev 2 branch selected in the dropdown



On the main file listing, select the 'details.html' file:



When this opens in GitHub, select the "Edit" pencil on the top-right corner of the file:



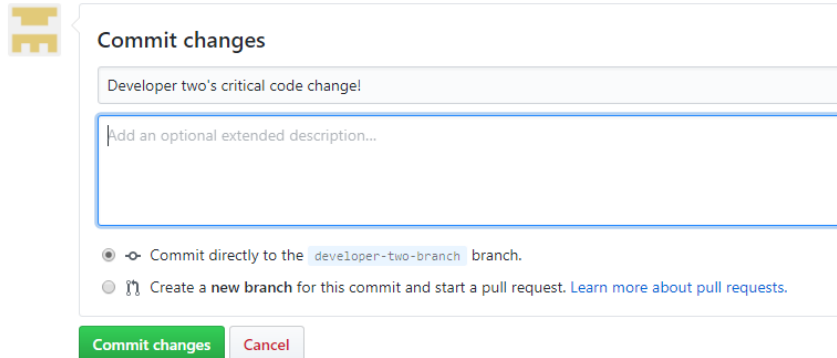
Change anything about the file in generally the same area we made the currently unmerged change for dev 1:

```

46
47     <h2> I'm making some changes! </h2>
48     <h3> Changes made on feature-branch-one for merge at GitHub</h3>
49
50     <h2> I'm developer two, creating a conflict with my critical code change!</h2>
51     <hr />
52     <footer>
53         <p>&copy; 2017 - <a href="http://www.majorguidancesolutions.com">Major Guidance Solutions</a></p>
54     </footer>

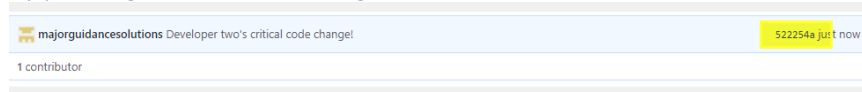
```

Add a commit message and create a commit directly on this branch:



The image shows the GitHub 'Commit changes' dialog. At the top is a commit message input field containing 'Developer two's critical code change!'. Below it is a larger text area for an optional extended description. At the bottom, there are two radio button options: 'Commit directly to the developer-two-branch branch.' (which is selected) and 'Create a new branch for this commit and start a pull request.' Below the options are two buttons: 'Commit changes' (green) and 'Cancel' (grey).

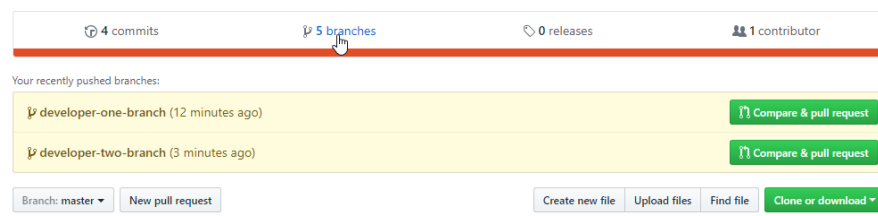
By pressing the “Commit Changes” button. Note the commit id:



#### d) Create and merge a pull request

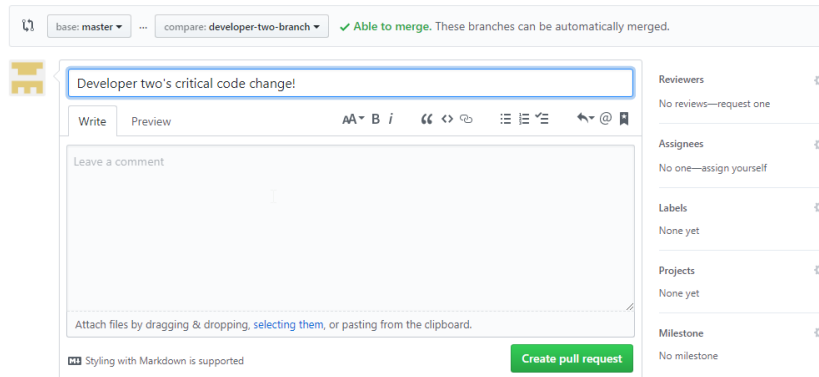
As we are pretending to be developer two, imagine that another developer had created a local feature branch and had made this change, pushed it up, and is now asking for a pull request. You know this will eventually conflict with your changes, but their changes are ready and there is nothing wrong with the code, so you are going to go ahead and merge their commits, then you’ll resolve on your branch eventually before merging your code to master.

Browse to the branch under the branches tab and select “New Pull Request”  
Alternatively, just click on the developer-two-branch “Compare & Pull Request”

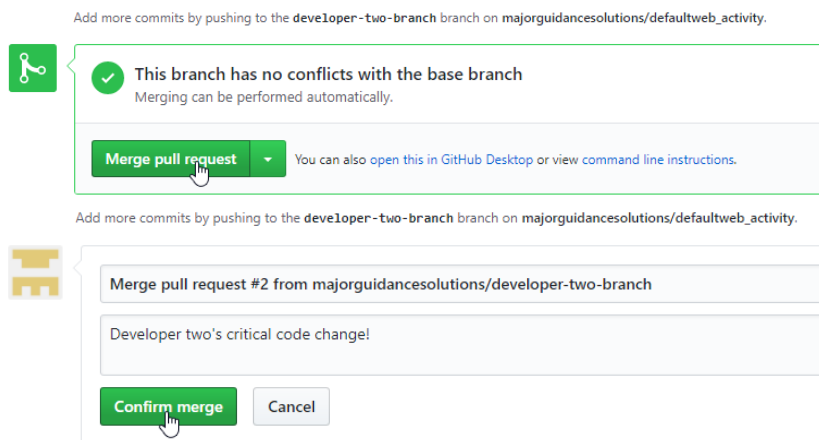


## Note that their changes are able to be merged:

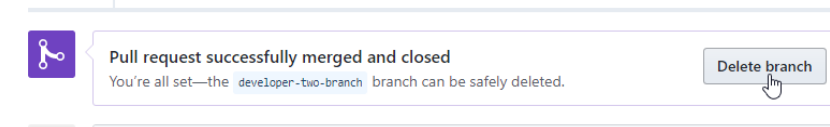
Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



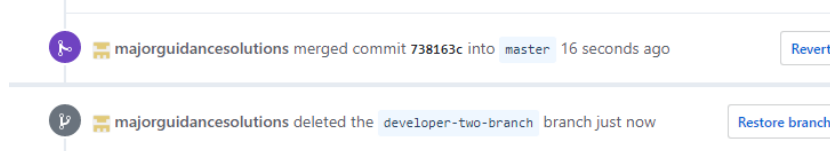
Which is indicated by the green “Able to merge” next to the checkmark. Create and Merge the pull request, pretending to be the code reviewer that allowed the request for dev 2’s changes to be merged:



After the merge, go ahead and delete the branch:



Note the merge commit ID:

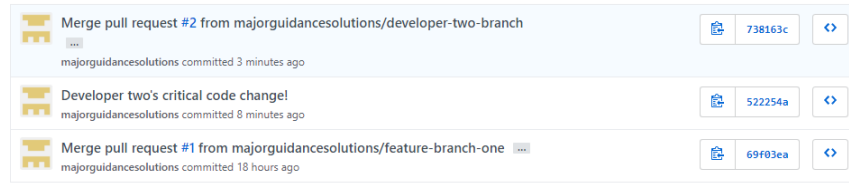


Also note that GitHub has an ‘auto-revert’ button. If you needed to rollback, you could click Revert. You can also restore the branch if you want. We don’t need to do either right now, but note your options.

The merge commit id for me was 738163c.



So here is my current history at GitHub (click on 'Commits' from the main screen to see yours):

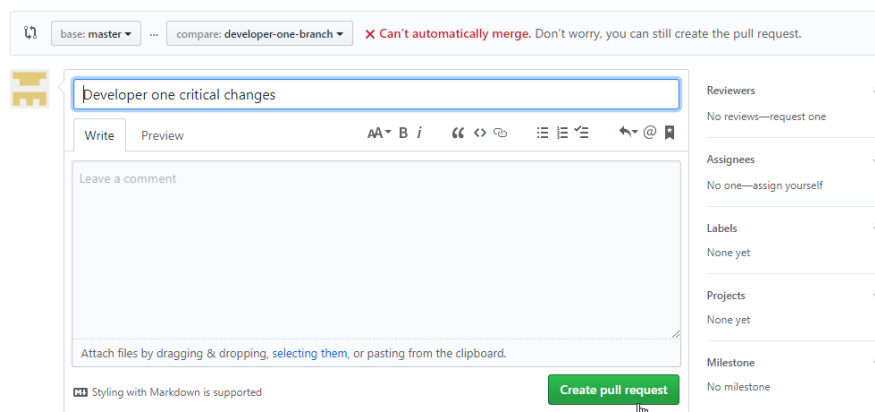


## Step 4: Create the pull request for developer one

### a) Create the Pull Request

While still at GitHub, create a new pull request for the dev 1 changes:  
Open a pull request

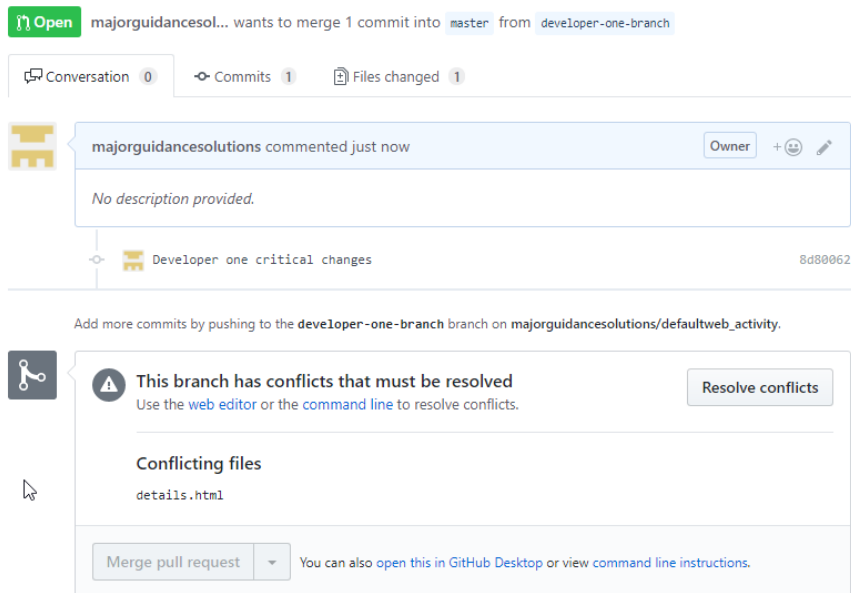
Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



Note the message: "Can't automatically merge"

This is expected. We knew there would be a conflict. Good news: We can still create the PR. Even better news, for something this simple we could easily resolve right at GitHub (if you want to do that, you can, and then you could re-simulate another conflict and continue this again to resolve at Local first, which is the recommended way to do this to keep from making mistakes).

Go ahead and create the pull request, even with the conflict still not resolved:



We are going to resolve at Local. This is the way I would recommend resolving a merge conflict, because you won't have any syntax help at GitHub. It would be very easy to create a "fix" at GitHub for this, but what if I mistype something? I may not know. Therefore, just get the changes pulled locally and fix the conflict with our mergetool, and then we'll be easily able to merge. Remember – real world changes are rarely going to be this easy to fix.

## Step 5: Bring the latest changes back to our local repo, merge and resolve the conflict, then push

- a) Now that we have a couple of commits in the chain on master and a conflict to resolve, we first want to just get everything up to date on master locally:

```
[git checkout master]
[git fetch origin]
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch)
$ 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
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 3), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/majorguidancesolutions/defaultweb_activity
69f03ea..738163c master -> origin/master
```

Note: It is critical for us to get our local master even with the remote master to ensure we have all of the bits for the merge commit.

```
[git pull 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 69f03ea..738163c
Fast-forward
 details.html | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

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

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git log --oneline
738163c (HEAD -> master, origin/master) Merge pull request #2 from majorguidance
solutions/developer-two-branch
522254a Developer two's critical code change!
69f03ea Merge pull request #1 from majorguidancesolutions/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
```

As we expected, we have the latest now for master, including dev two's merged changes

#### b) Merge the changes into our local feature branch via master

Our goal here is to make sure that we can put our own changes into the repository. What we need to do is get the changes that are in conflict and already at GitHub, and then we merge to our feature branch where we are working. In this way, we don't lose the other developer's changes. We may need to modify their change slightly if it affects our code, or we might be able to just manually merge our change within their changes. The end goal is to keep all of their functional changes as well as adding ours. As the developer, we have to discern how to resolve this so no functionality is lost.

*Switch to our dev branch:*

```
[git checkout developer-one-branch]
```

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

*Merge their changes from master into ours:*

```
[git merge master]
```

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch)
$ git merge master
Auto-merging details.html
CONFLICT (content): Merge conflict in details.html
Automatic merge failed; fix conflicts and then commit the result.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch)
$
```

Note, we are now in conflict and the branch is appended with | MERGING on it. IF we didn't want to continue, we could just abort `[git merge --abort]`. We are going to continue with the merge, however.

c) Resolve the conflict with our mergetool

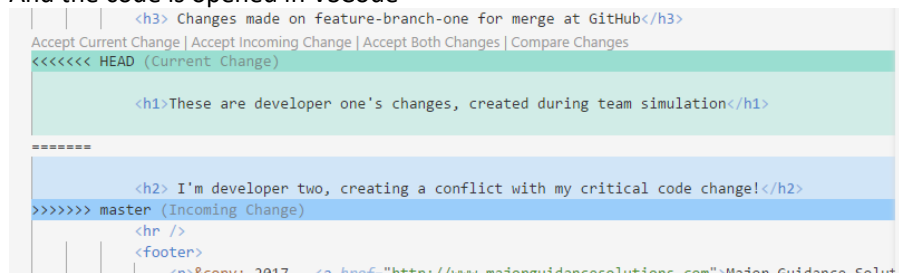
NOTE: If you haven't setup a mergetool, you will be taken back to 1979 in the VIM merge editor. Therefore, make sure you have first setup your default mergetool using the activity "setting VSCode to be our default mergetool"]

```
[git mergetool]
```

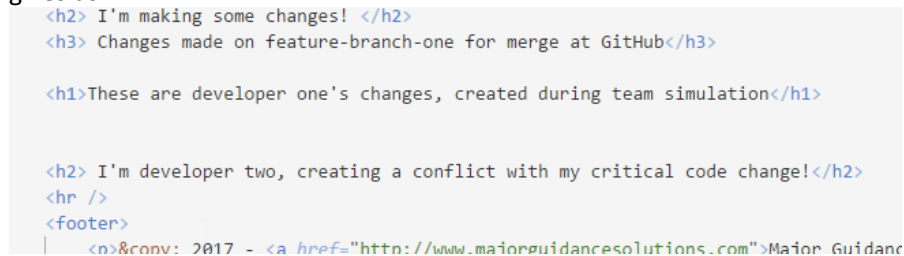
```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch|M
ERGING)
$ git mergetool
Merging:
details.html

Normal merge conflict for 'details.html':
{local}: modified file
{remote}: modified file
```

And the code is opened in VSCode



We're going to click on the item that says "Accept Both Changes" which then gives us:



We could clean this up further if we would wish. It will be your job as the developer to review the conflict and make sure that your changes and the other changes play nice and all intended functionality is still intact after the merge resolution.

*Save the changes and close VSCode to continue the merge operation*

Once you save and close you get taken back to the terminal:

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch|M
ERGING)
$
```

Now you have two choices: **commit** or **continue**. If you want to commit, just type the command `[git commit -m 'merge resolved']`. I'm going

to go the continue route here, and if you have multiple conflicts you would want to do this as well, if you weren't already automatically taken to the next conflict.

*Continue the merge with --continue*

**[git merge --continue]**

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch|MERGING)
$ git merge --continue
```

Since there are no more conflicts, this brings up the editor to add a commit message, which is already set if I want to keep the default message:

```
COMMIT_EDITMSG x
1 Merge branch 'master' into developer-one-branch
2
3 # Conflicts:
4 #   details.html
5 #
```

I'm going to use this message as-is, save it, and close. Saving will then create the commit in the terminal:

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch|MERGING)
$ git merge --continue
[developer-one-branch 42ed51b] Merge branch 'master' into developer-one-branch
```

*Now look at our history to make sure it looks correct:*

**[git log --oneline]**

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch)
$ git log --oneline
42ed51b (HEAD -> developer-one-branch) Merge branch 'master' into developer-one-branch
738163c (origin/master, master) Merge pull request #2 from majorguidancesolutions/developer-two-branch
522254a Developer two's critical code change!
8d80062 (origin/developer-one-branch) Developer one critical changes
69f03ea Merge pull request #1 from majorguidancesolutions/feature-branch-one
7c4f044 (origin/feature-branch-one, feature-branch-one) Changes on feature-branch-one
7b6a932 (origin/feature-branch-two, feature-branch-two, My-Feature-Branch) changes to details
b7edba6 initial commit
```


And that shows us being in place, including the commits from both developers in our history, so we know we are now safe to push and merge at GitHub:

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


```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch)
$ git push origin developer-one-branch
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 437 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
8d80062..42ed51b developer-one-branch -> developer-one-branch
```

d) Now we can merge the pull request at GitHub



## Developer one critical changes #3

 **Open** majorguidancesol... wants to merge 2 commits into `master` from `developer-one-branch`


Conversation 0 Commits 1 Files changed 1

 majorguidancesolutions commented 25 minutes ago Owner + 👤 ✎

No description provided.

  Developer one critical changes 8d8062


Add more commits by pushing to the `developer-one-branch` branch on majorguidancesolutions/defaultweb\_activity.

 **✓** This branch has no conflicts with the base branch  
Merging can be performed automatically.

**Merge pull request** You can also [open this in GitHub Desktop](#) or view [command line instructions](#).


Go ahead and merge and confirm. This will get everything up to date. Then delete the branch at REMOTE.


Add more commits by pushing to the `developer-one-branch` branch on majorguidancesolutions/defaultweb\_activity.


 Merge pull request #3 from majorguidancesolutions/developer-one-branch


Developer one critical changes

**Confirm merge** Cancel



















 **Pull request successfully merged and closed**  
You're all set—the `developer-one-branch` branch can be safely deleted. Delete branch

 Developer one critical changes 8d8062

 majorguidancesolutions merged commit b78feb3 into `master` 22 seconds ago Revert

 majorguidancesolutions deleted the `developer-one-branch` branch just now Restore branch

And review the commits on the repository:

 Merge pull request #3 from majorguidancesolutions/developer-one-branch majorguidancesolutions committed a minute ago	 b78Feb3	
 Merge branch 'master' into developer-one-branch majorguidancesolutions committed 8 minutes ago	 42ed51b	
 Merge pull request #2 from majorguidancesolutions/developer-two-branch majorguidancesolutions committed 36 minutes ago	 738163c	
 Developer two's critical code change! majorguidancesolutions committed 41 minutes ago	 522254a	
 Developer one critical changes majorguidancesolutions committed an hour ago	 8d8062	
 Merge pull request #1 from majorguidancesolutions/feature-branch-one majorguidancesolutions committed 18 hours ago	 69f03ea	

Does that lineup with what we have locally on master?

`[git checkout master]`

`[git log --oneline]`

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (developer-one-branch)
$ 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 log --oneline
738163c (HEAD -> master, origin/master) Merge pull request #2 from majorguidance
solutions/developer-two-branch
522254a Developer two's critical code change!
69f03ea Merge pull request #1 from majorguidancesolutions/feature-branch-one
7c4f044 (origin/feature-branch-one, feature-branch-one) Changes on feature-branch-one
7b6a932 (origin/feature-branch-two, feature-branch-two, My-Feature-Branch) changes to details
b7edba6 initial commit
```

NO! we are missing the final merge commit!

## Step 6: Cleanup

### a) Optional cleanup

We already see that we don't have the merge commit – so let's get that:

`[git fetch origin --prune]`

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git fetch origin --prune
From https://github.com/majorguidancesolutions/defaultweb_activity
- [deleted] (none) -> origin/developer-one-branch
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
738163c..b78feb3 master -> origin/master
```

Wait! What's prune? Oh yeah, "prune" will delete any references [not the actual branch] on my local that no longer exist on remote. In the message you see that it has deleted the origin/developer-one-branch, so we should no longer see 'origin/developer-one-branch' in our branch list after this, right?

`[git branch -a]`

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

Hey, it's still there locally! -- Yes, we have to also delete our branch locally to make it go away for good. This is for our protection – in case someone deletes our remote branch and we didn't want that to happen. Note, however, there is no longer a branch: remotes/origin/developer-one-branch – as our local repository did prune its refs so that it knows there is no such branch at upstream remote.

**[git pull 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 738163c..b78feb3
Fast-forward
 details.html | 3 +++
1 file changed, 3 insertions(+)
```

**[git log --oneline]**

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git log --oneline
b78feb3 (HEAD -> master, origin/master) Merge pull request #3 from majorguidance
solutions/developer-one-branch
42ed51b (developer-one-branch) Merge branch 'master' into developer-one-branch
738163c Merge pull request #2 from majorguidancesolutions/developer-two-branch
522254a Developer two's critical code change!
8d80062 Developer one critical changes
69f03ea Merge pull request #1 from majorguidancesolutions/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
```

And we are up-to-date!

Let's delete the local copy of the dev 1 branch, and any other branches we no longer want around:

**[git branch -d developer-one-branch]**

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb_Activity (master)
$ git branch -d developer-one-branch
Deleted branch developer-one-branch (was 42ed51b).
```

NOTE: Since ALL commits are fully-merged, we should not need to force delete with a capital "D" here. If that is the case, something went wrong with your merge and/or the branch has been further modified since pushing to remote.

This concludes our team merge conflict resolution activity.





everything you would need to know to work in a general manner with your team to have successful source control in place on your valuable code.

