Create Repository

Add .gitignore

Add basic info into the Readme

Create project file structure

Go to settings and Manage Access and login. Add github members. Use github alias and each collaborator will need to accept invite from their email.

Go to GitBash, and find the desired folder to keep your repository

Select the Clone Repo url and go to Gitbash and use command ‘git clone {url}’

Add project structure

Git add .

Git commit -m “first update”

Git push origin master

Folders won’t be added to repository unless they have a file in them

Git Status

\*\*Now we will require approvals\*\*

Settings > Branches > Branch Protection Rules: Add Rule button

Branch Name Pattern : ‘master’

Require pull request reviews before merging

Required approving reviews: 1

Dismiss stale pull request approvals when a new commits are pushed

Include administrators

Select Create then Save Changes

Other options:

* Go to Settings > Notifications: Git Master should have
* To protect environmental variables: Settings > Secrets > Add New Secret

Now the administrator will not be able to push their own changes up. Git push origin master will not work. It will give an error of ‘[remote rejected] master -> master (protected branch hook declined)

This protects the master branch – a git push would write directly to the project and potentially overwrite it. To have your own ‘space’ to store your ongoing changes then you need to create a branch.

Git checkout -b ‘test-branch’

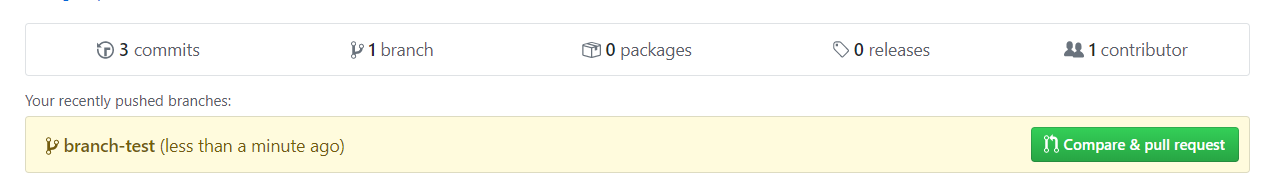
Git Status to see if changes are seen

Git add . then git commit -m ‘added branched instructions’ then git push\*

For this git push, the remote (online) repository doesn’t know about this branch so git bash provides additional instructions about how to push to a new branch (notify remote)

Git push –set-upstream origin test-branch

This will change the github interface to reflect that a branch is available and that it can be merged

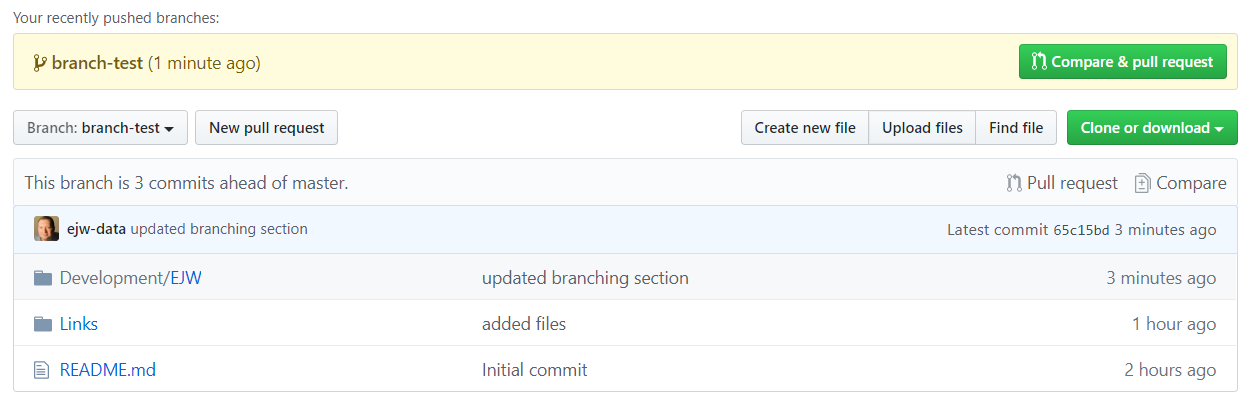


I use the term merge but it is actually a pull into the master branch

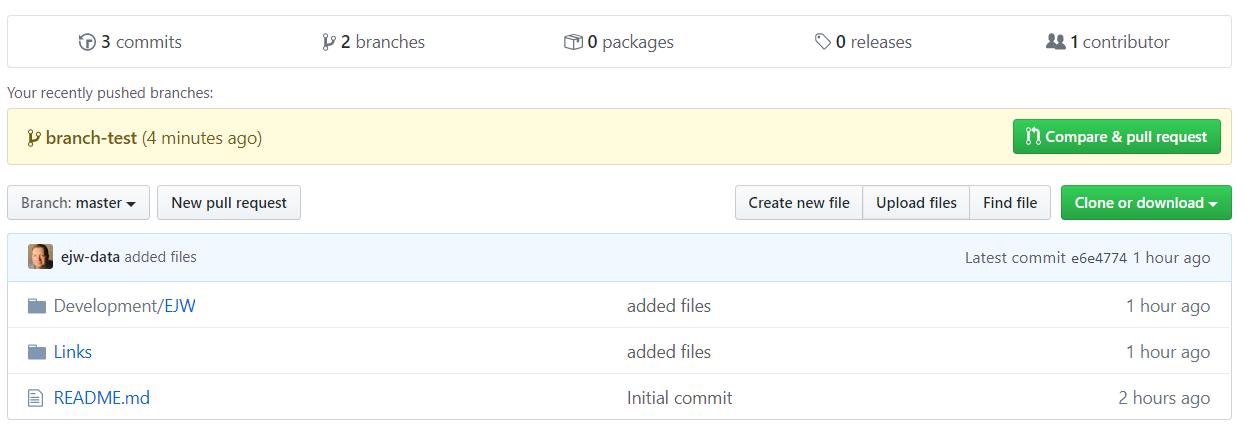
If I make more changes to the local documents then I can do a ‘git add .’ then a ‘git commit -m “….”’ And then a ‘git push origin test-branch’

If I click on the branch-test in github (see image above) then I can go to the branch files and see what is different. These changes will not be seen on the master branch.

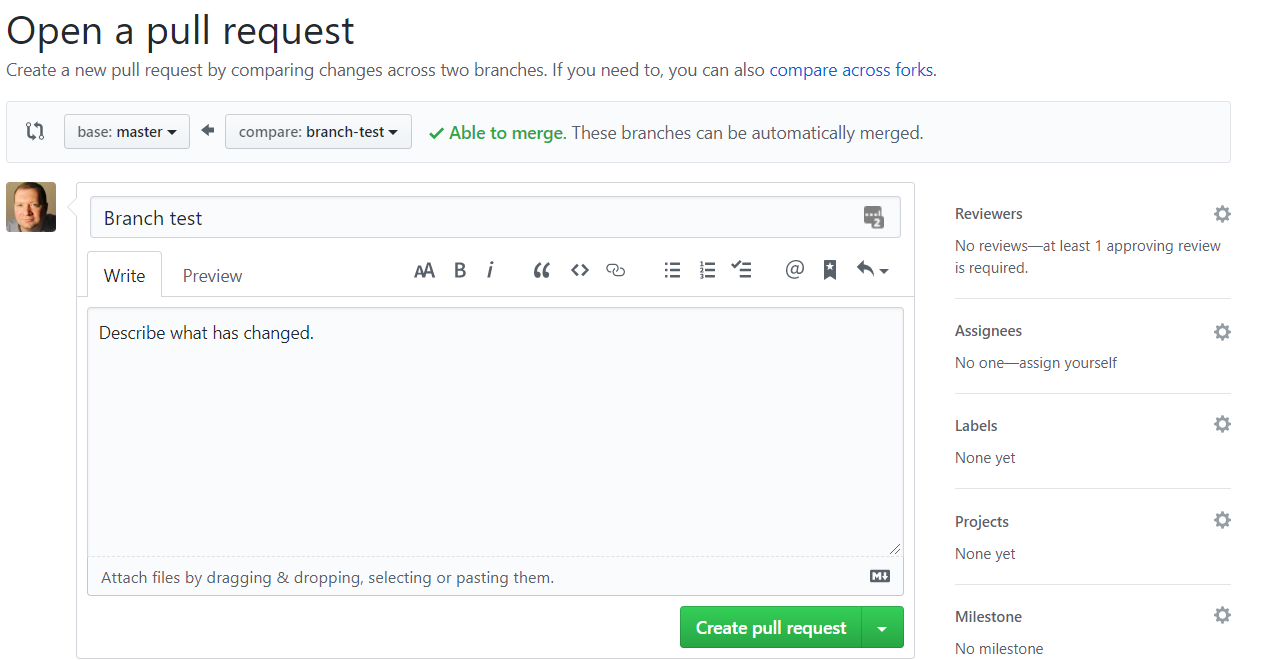
Below are the test branch:



Here is the master branch – notice the commit messages and time stamp differences:

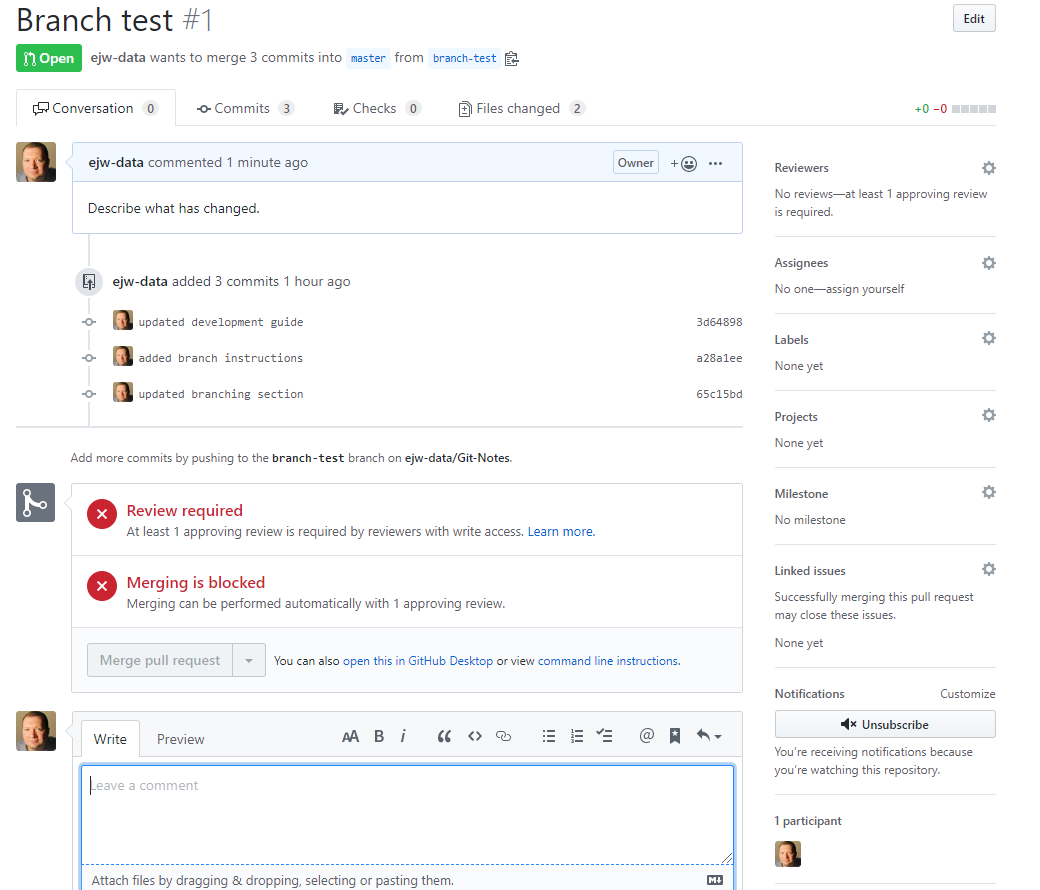


Let’s say the changes should be added to the master so we can now compare and pull (see button above).



Note: it says able to merge but under Reviewers in the right side pane it shows that an approval is needed.

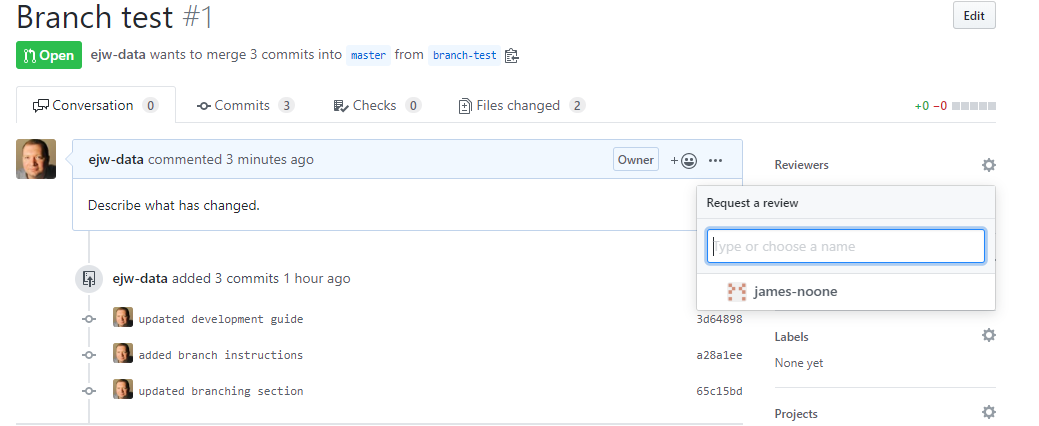
After submitting your pull request, you get something like this:



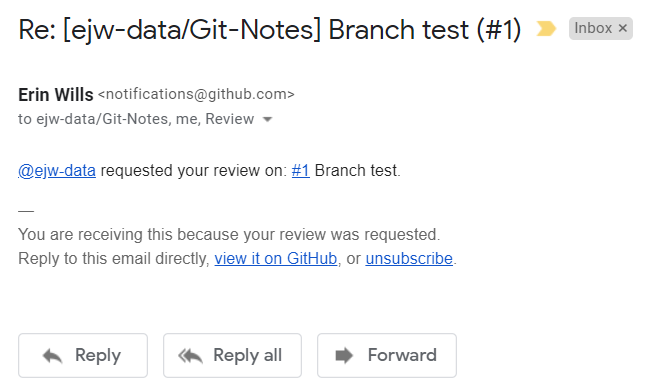
Note: Notifications are on but I don’t receive emails about the requested pull

To speed things up, you can assign a reviewer or any collaborator can come into the repository and review.

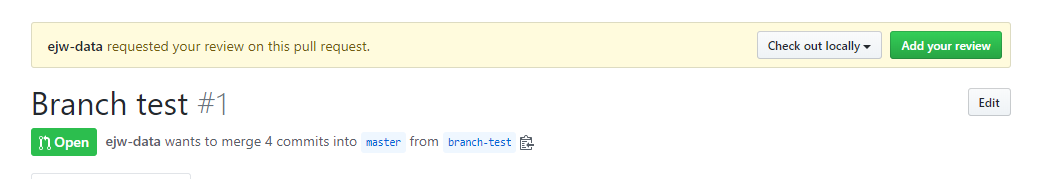
To assign a reviewer, use the right side pane.



The requested reviewer will get a message like this in their email:

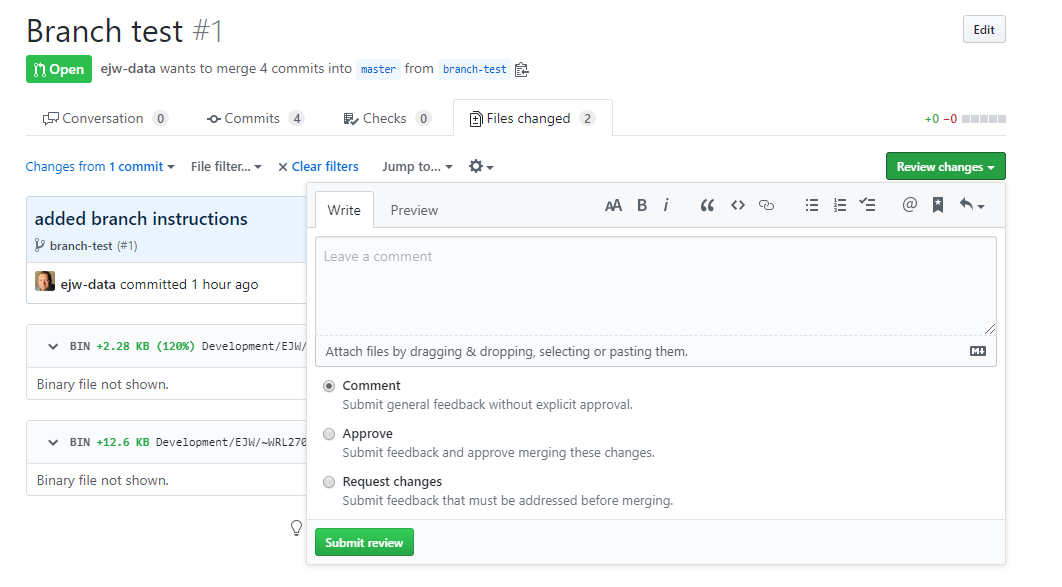


The reviewer also gets this message in github in their account only:

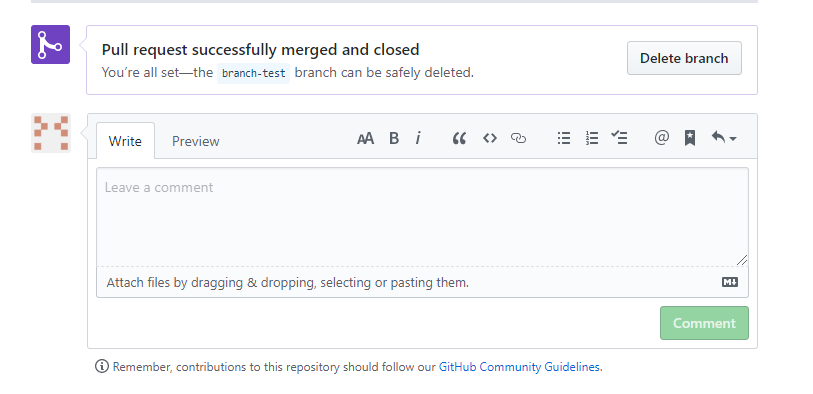


As shown above, the commits that I am submitting are all found in a word document. If I would have stored my notes in a text document then I would be able to review the changes in github.

Each commit can be reviewed separately and does not have to be accepted. When clicking on the commit that is under review the following can be done:



I’m going to approve all these changes and move on. This just requires selecting approve.



After accepting the following message will be available. The branch can be deleted but that will not delete the local branch on the computer. That branch can be continued to be used locally. Also, the branch can be deleted by selecting the branch above or on the frontpage of the repository, the branches link can be selected and the branch can be deleted from that summary page. After deleting the branch-test it is probably a best practice to closeout the local branch. Here is a good procedure:

Git branch to see what branch you are on.

Switch to the master – git checkout master

Read gitbash instructions, probably will ask to push the local copy, so …

Git pull, git add ., git commit -m “…”, git merge branch-test - the merge will cause VS Code to show up asking for a message to tie to the merge.

Delete by Git branch -d branch-test

The branch should no longer be available.

Check git branch

Note: Something interesting – when on a branch, the local directory containing the repository also changes. So if files have been added to a branch then those files will not be visible when the master is checked out.

Check to see if any branches need removed from github.

Here is an alternate project setup:

Verify that the master branch exists and consider this to be the production branch.

Create a new branch called sandbox in github and setup permissions

In gitbash, git checkout -b ‘sandbox’

Git add ., commit -m, then

Either git push –{instructions} or git pull origin sandbox

Then git checkout -b ‘sandbox-testing’

git push origin sandbox-testing

Now use git branch and will find 3 branches – master, sandbox, sandbox-testing

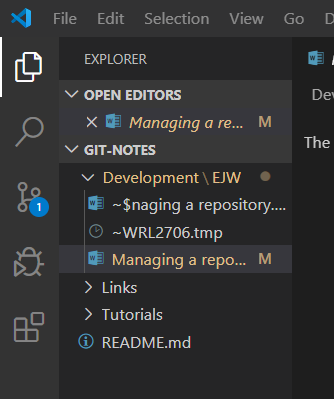
To push changes, it is very similar:

git add ./git commit -m “…” / git push origin sandbox-testing

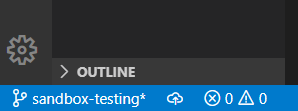
With this setup we now have a master (production code) and a testbed code (sandbox) and branches each member can create modifications.

Describe how to integrate VS Code

VS Code Explorer show what has been modified (M).



At the bottom of VS Code shows what branch and repository I am in.



Exercises:

<https://help.github.com/en/github/administering-a-repository/managing-teams-and-people-with-access-to-your-repository>