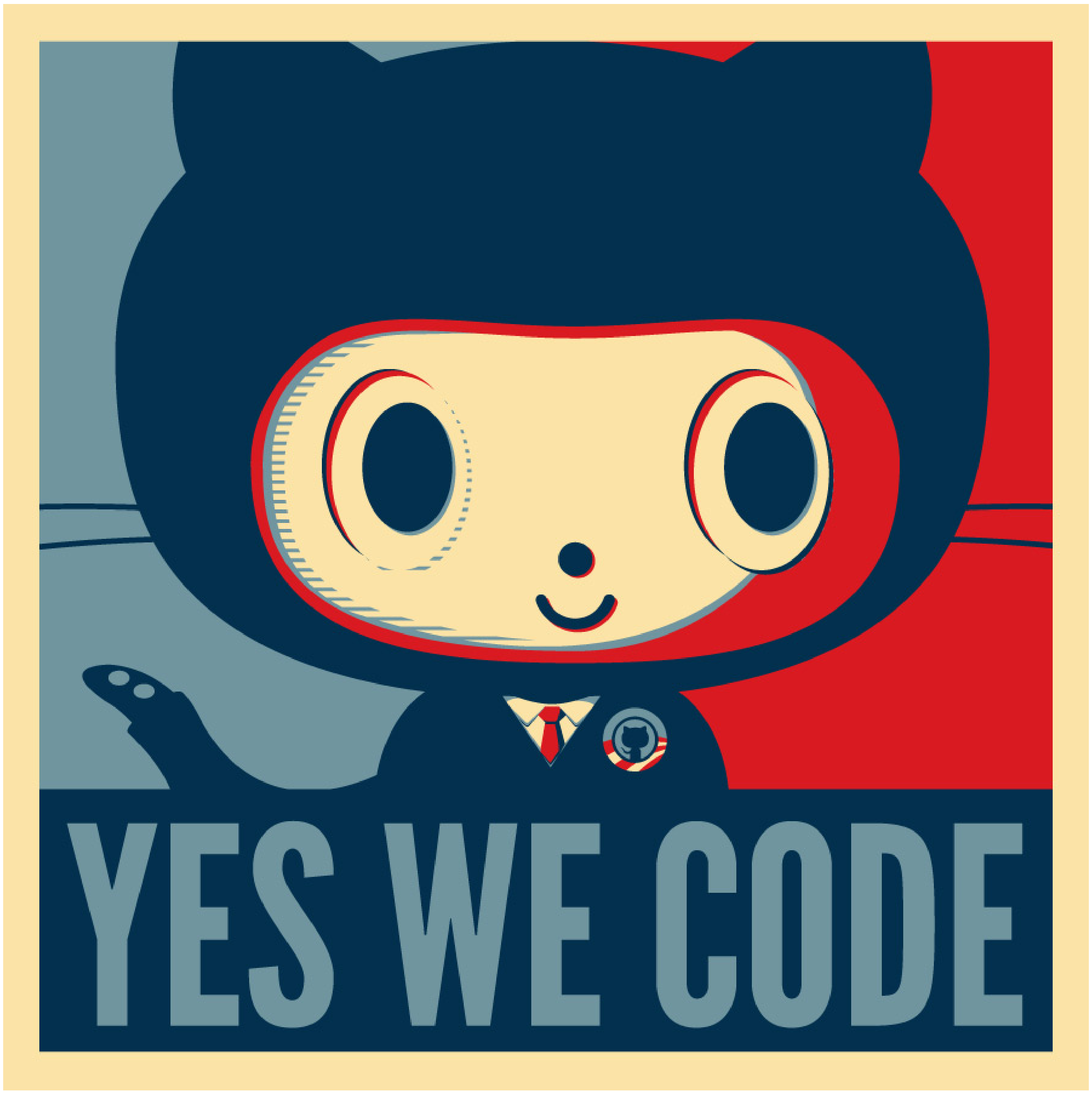
16 Developer Tools: GitHub

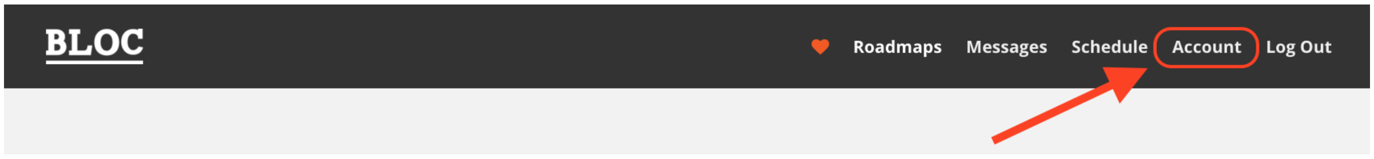


GitHub

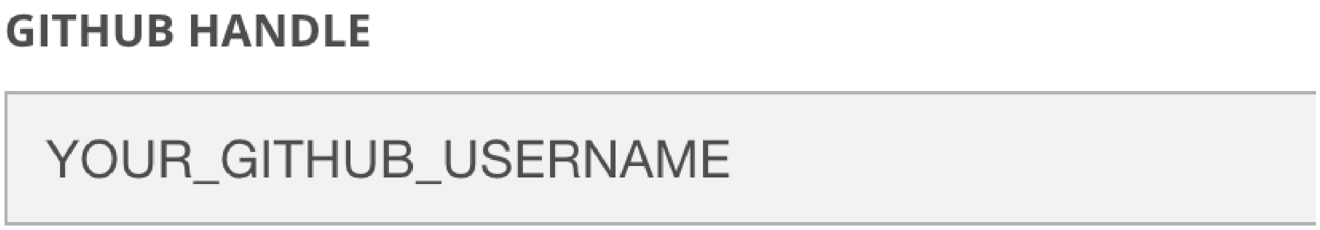
GitHub is a web service which maintains a copy of a repository in the cloud. If you haven't already, take a few moments to **create a GitHub account**. Think of GitHub as a pantry (or refrigerator) in the grocery store example we used from the Git checkpoint. We bought our groceries (created our repository) and want to share them with others. We can put our groceries in our pantry (GitHub) at home for others to consume.

Connect GitHub to Bloc

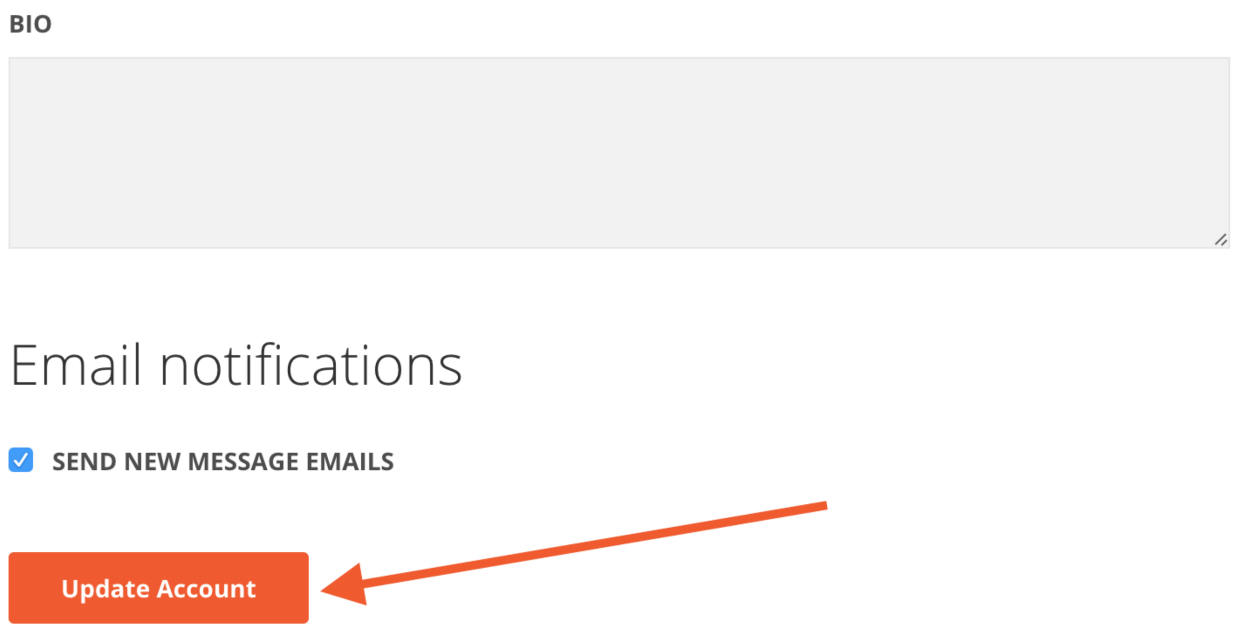
Let's connect your GitHub account to Bloc. Go to your account page in Bloc:



Scroll down to the text box that says "GITHUB HANDLE". Put your GitHub username in:



Continue to scroll down until you see the "Update Account" button. Press the button and your GitHub account should now be connected to Bloc:


BIO

Email notifications

☑ SEND NEW MESSAGE EMAILS

GitHub Authentication

There are two ways to authenticate with GitHub when interacting with repositories from the command line. If you're using SSH, then you can set up SSH keys. If you're using HTTPS, then you can use **password caching**. In this guide, we have used SSH as our means of interacting with GitHub. If you don't have a reason to use HTTPS, use SSH.

SSH Keys

Generate an SSH key for your machine by following **these instructions**.

> If you're using Cloud9 your SSH Key is already generated and can be retrieved from your account settings. Copy the key and start with step 4.

Once the key is generated, follow the remaining steps in GitHub's guide to add them to your account. After the keys are set up, we can actually push the data to GitHub.

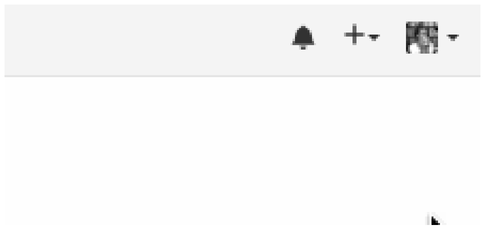
HTTPS Password Caching

If you choose to use HTTPS, then use **this guide** to set up password caching.

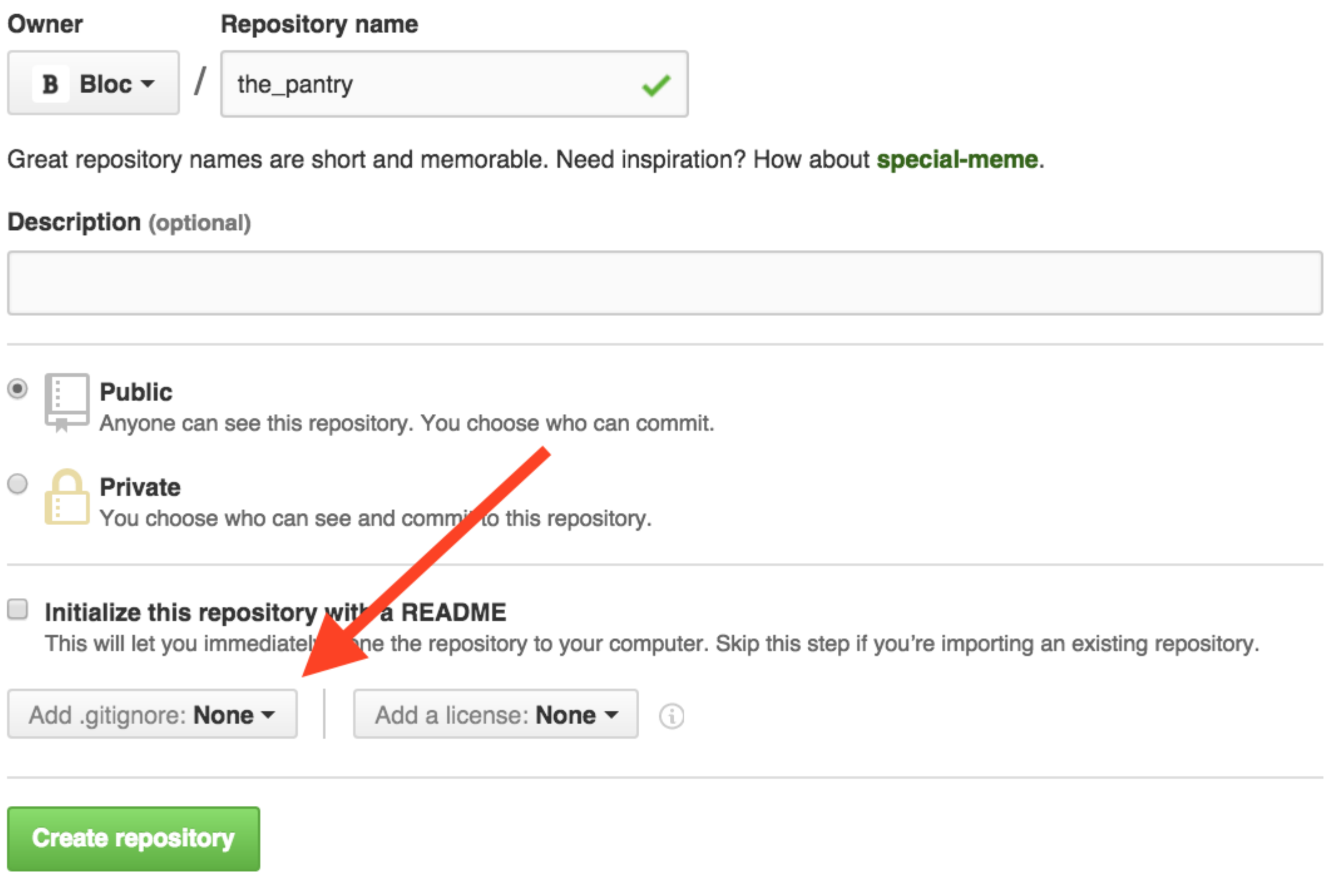
Setting a Remote Repository

How do we get our groceries to the pantry, or rather how do we get the items in a repository to GitHub? Git has a built in command called `push` that will take care of this for us.

But where are we pushing our data to? First, we need to set up the repository on GitHub itself. Go to GitHub, log in, find the "+" mark on the top right of the screen and select "New repository".



Name the repository `the_pantry`. Type any description. We can optionally use GitHub's convenient prebuilt `.gitignore` file feature. Since our repository already has a `.gitignore` we created manually, let's leave it as `None`:



Owner Bloc / **Repository name** the_pantry ✓

Great repository names are short and memorable. Need inspiration? How about **special-meme**.

Description (optional)

[✓ **Public**] Anyone can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

> The selections in the `Add .gitignore` dropdown populate `.gitignore` with a list of known files per project type that don't need to be version controlled.

We can now view the repository on GitHub at this URL:

```
https://github.com/YOUR_GITHUB_USERNAME/the_pantry
```

After we create the repository on GitHub, we need to create our local repository and point it at GitHub.

Create a directory named `bloc` if you don't already have one and a sub-directory in `bloc` entitled `the_pantry`:

~

```
$ cd
$ mkdir -p bloc/the_pantry
```

Change directories to the `the_pantry` directory:

~

```
$ cd bloc/the_pantry
$ pwd
/Users/<YOUR_USERNAME>/bloc/the_pantry
```

Run the following commands, replacing `YOUR_GITHUB_USERNAME` with your GitHub username:

~/bloc/the_pantry

```
$ git init .
$ git remote add origin git@github.com:YOUR_GITHUB_USERNAME/the_pantry.git
```

Verify that the remote server is set correctly by running the following command:

~/bloc/the_pantry

```
$ git remote -v
origin  git@github.com:YOUR_GITHUB_USERNAME/the_pantry.git (fetch)
origin  git@github.com:YOUR_GITHUB_USERNAME/the_pantry.git (push)
```

We can also verify that the remote is set properly by examining Git's configuration file:

~/bloc/the_pantry

```
$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
        ignorecase = true
        precomposeunicode = true
[remote "origin"]
        url = git@github.com:YOUR_GITHUB_USERNAME/the_pantry.git
        fetch = +refs/heads/*:refs/remotes/origin/*
```

The configuration file shows a remote server that we have named "origin". This is the default name used for remote repositories. Its URL points to our repository on GitHub. We want to push the data to this remote repository, but before we do that we need a way to authenticate with GitHub.

Git Push

Let's add a file to `the_pantry`:

~/bloc/the_pantry

```
$ echo "# the_pantry" >> README.md
$ git add README.md
$ git commit -m "first commit"
```

Now that we have pointed our local repository to GitHub and have authentication set properly, push the data to the remote repository:

~/bloc/the_pantry

```
$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 226 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:YOUR_GITHUB_USERNAME/the_pantry.git
 * [new branch]      master -> master
```

We told Git to push our local `master` branch to a remote branch named `master` towards "origin" which we set up previously. We have successfully stowed our groceries in the fridge and pantry.

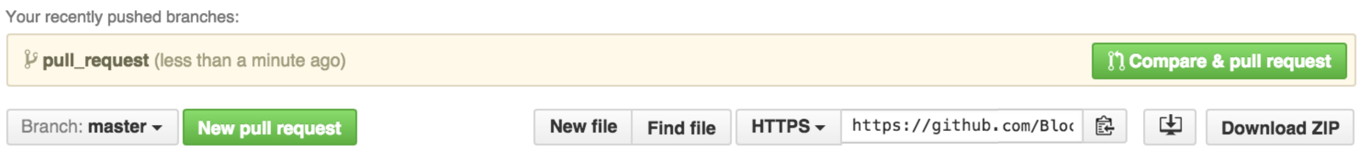
Pull Requests

One of the easiest ways to share a commit on a branch is by creating a **pull request**. Pull requests can only exist on branches that aren't the main branch. Create a new branch off of master called `my_new_branch`, make a trivial change and then push it to GitHub:

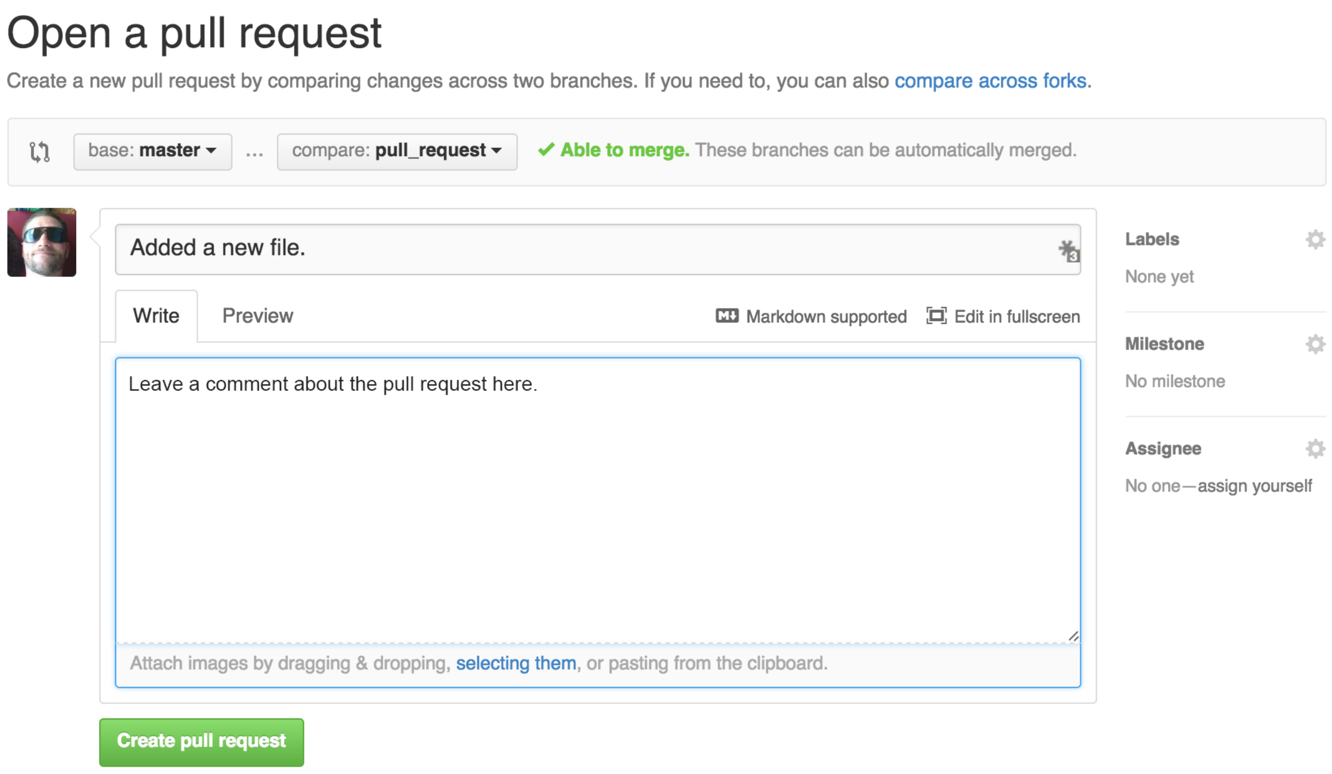
~/bloc/the_pantry

```
$ git checkout master
$ git checkout -b my_new_branch
$ touch my_new_branch_file.txt
$ git add my_new_branch_file.txt
$ git commit -m "Add a new file"
$ git push origin my_new_branch
```

Once you've pushed the branch, visit the repository page on GitHub and there should be a green button that says "Compare & pull request".



Click it "Compare & pull request". We are presented with a the "Open a pull request screen".


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.

base: master ... compare: pull_request ✓ Able to merge. These branches can be automatically merged.

Added a new file.

Leave a comment about the pull request here.

Attach images by dragging & dropping, selecting them, or pasting from the clipboard.

We aren't actually going to submit this pull request. If we did, we would want base to be set to the branch we want to merge to. For example if we wanted to merge `my_new_branch` to `master` we would set base to `master`.

Since this pull request was only meant to be a lesson, you should delete the branch rather than merging the pull request:

~/bloc/the_pantry

```
$ git checkout master
$ git branch -D my_new_branch
$ git push origin --delete my_new_branch
```

`git branch -D my_new_branch` will delete your local copy of the branch while `git push origin --delete my_new_branch` will delete the remote copy.

Merging

Merging combines two branches into one. Merge conflicts happen when the same file or part of a file is changed on two different branches and then said branches are merged back together. Git doesn't know which changes it should take as authoritative. We will illustrate merge conflicts using our grocery store example. First, make sure you are on the `new_grocery_list` branch and add some content to `orange.txt`, then commit it:

~/bloc/the_pantry

```
$ git checkout -b new_grocery_list
$ echo "I'm in new_grocery_list" > orange.txt
$ git add orange.txt
$ git commit -m "Merge conflict"
```

Next, checkout the `master` branch in the `the_pantry` repository:

~/bloc/the_pantry

```
$ git checkout master
```

Create the `orange.txt` file on `master` with the content below using the `echo` command, then commit your changes to `orange.txt`:

~/bloc/the_pantry

```
$ touch orange.txt
$ echo "I'm in master" > orange.txt
$ git add orange.txt
$ git commit -m "Merge conflict"
[master 93261a2] Merge conflict
 2 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100755 orange.txt
```

The next step is to merge the `new_grocery_list` branch into `master`:

~/bloc/the_pantry

```
$ git merge new_grocery_list
Auto-merging orange.txt
CONFLICT (add/add): Merge conflict in orange.txt
Automatic merge failed; fix conflicts and then commit the result.
```

There is a conflict and Git doesn't know which content in the file is authoritative. Open the `orange.txt` file in a text editor. It will look like this:

```
<<<<<<< HEAD
I'm in master.
=======
I'm in new_grocery_list
>>>>>>> new_grocery_list
```

The content from the `master` branch is above the `=======` and the content from `new_grocery_list` is below the `=======`. "HEAD" is a paradigm Git uses to refer to the topmost commit on the current operating branch. We have to make the decision as to what content we want in the file by deleting what we don't want and then saving the file. We can keep all of the content by just removing the lines:`<<<<<<< HEAD`, `=======`, and `>>>>>>> new_grocery_list`.

Remove the content in `master` and leave the content from `new_grocery_list`. Thus the file should look as such:

```
I'm in new_grocery_list
```

After the file is edited, we need to stage it then commit to mark the conflict as resolved:

~/bloc/the_pantry

```
$ git add orange.txt
$ git commit -m "Fix merge conflict"
[master 538d044] Fix merge conflict
```

Push the merged `master` branch, and the `new_grocery_list` branch to GitHub:

~/bloc/the_pantry

```
$ git push origin master
$ git push origin new_grocery_list:new_grocery_list
```

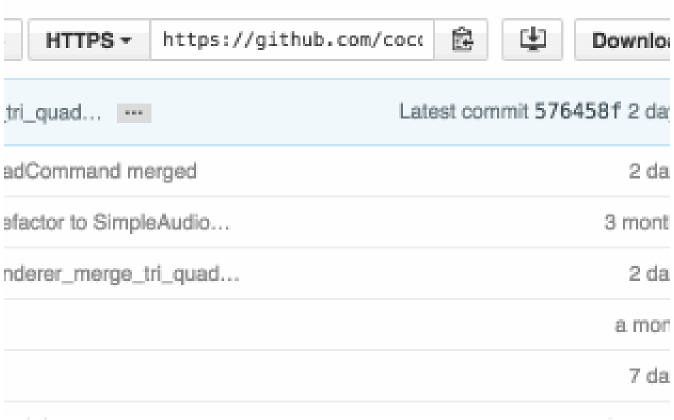
Git Clone

We have successfully pushed content to GitHub. This allows other developers on our team to see changes or *clone* them to their local computer so they can collaborate. Let's simulate what the perspective of a potential team member would look like. First, change to the directory above `the_pantry` and create a directory named `the_pantry_clone`:

~/bloc

```
$ cd ..
$ mkdir the_pantry_clone
$ cd the_pantry_clone
```

Next, copy the SSH link from `the_pantry` by navigating to it on GitHub and selecting "SSH clone URL" on the lower right hand part of the browser as seen below:



With the link copied, go back to the command line and execute the `git clone` command below. Don't forget the period `.` at the end. It tells Git to copy the contents of the repository directly into the current working directory.

~/bloc/the_pantry_clone

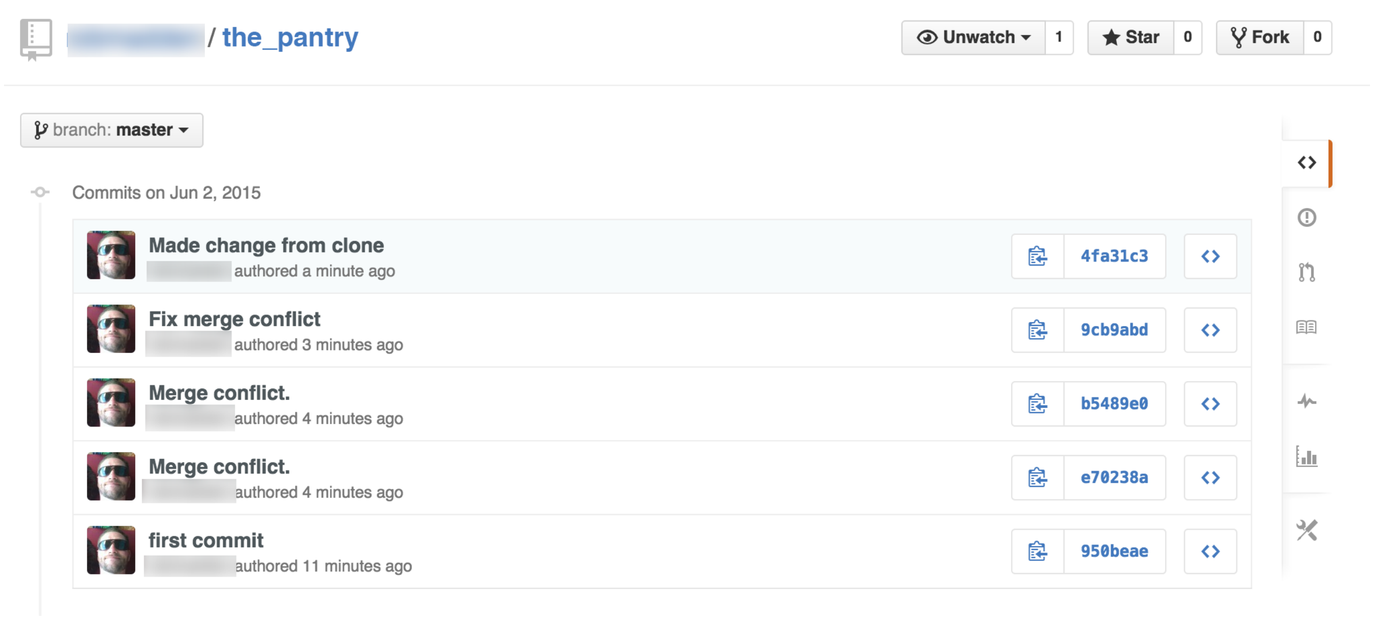
```
 $ git clone git@github.com:YOUR_GITHUB_USERNAME/the_pantry.git .
```

We cloned the initial repository into `the_pantry_clone`. Make a change to `apple.txt`, push the change and view it on GitHub:

~/bloc/the_pantry_clone

```
$ git checkout master
$ echo "Change from clone" >> apple.txt
$ git add apple.txt
$ git commit -m "Change from clone"
$ git push origin master
```

GitHub shows the new commit in the `master` branch. This is a basic example of the type of workflow that Git and GitHub provide.



Commits on Jun 2, 2015

```
Made change from clone
[redacted] authored a minute ago        4fa31c3

Fix merge conflict
[redacted] authored 3 minutes ago       9cb9abd

Merge conflict.
[redacted] authored 4 minutes ago       b5489e0

Merge conflict.
[redacted] authored 4 minutes ago       e70238a

first commit
[redacted] authored 11 minutes ago      950beae
```

Help

- **GitHub has extensive resources**.

16. Developer Tools: GitHub

Assignment	**Discussion**	**Submission**

- Create a new branch named `the_pantry_assignment` from `master` in the `the_pantry_clone` repository.
- Add the text "GitHub Checkpoint Assignment" to `README.md`.
- Stage and commit `README.md`.
- Push `the_pantry_assignment` to GitHub.
- Submit a pull request on GitHub from `the_pantry_assignment` to `master`.
- Merge the pull request to `master` on GitHub.
- Switch back to Bloc.
- In the submission section, select `the_pantry` and "Merge pull request #1" as your commit.

- Submit this commit to your mentor for review.

assignment completed

?