

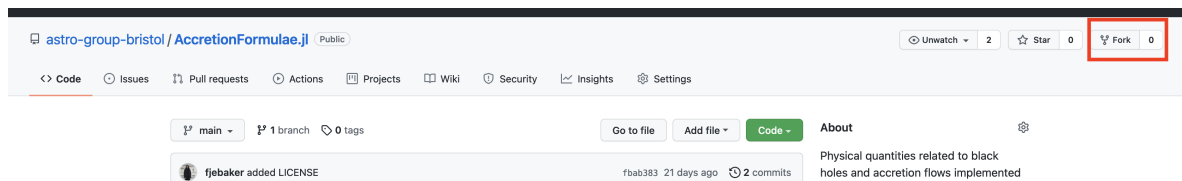
Triangular Workflow with AccretionFormulae.jl

To familiarize ourselves with the *triangular workflow*, we will begin by making a small change to the README file in AccretionFormulae.jl.

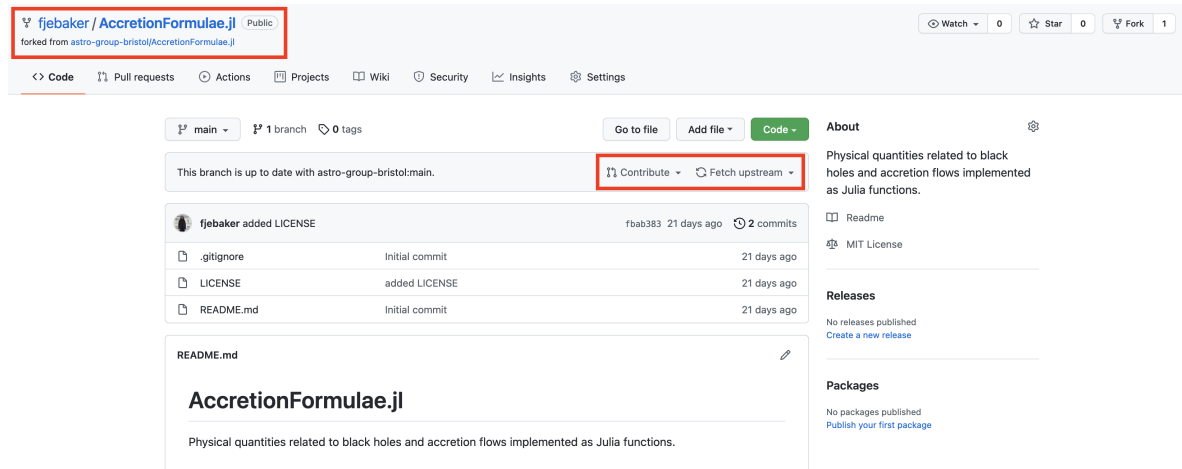
To do this, we will create a *fork*, clone a local copy of this fork, configure the *remote* sources with an upstream, use the `git` command line tool add and push our code changes, and then open a pull request on the upstream repository.

Creating a fork

To create a fork of a repository, visit the GitHub page – in our case, AccretionFormulae.jl. If you are logged-in to GitHub, then in the top right corner you will see a “Fork” button:



Click this to create a Fork of AccretionFormulae.jl for your personal GitHub account. This may take a short moment, but once it is complete, you should see something similar to this:



Notice that the repository name has changed from `astro-group-bristol/AccretionFormulae.jl` to `[your-username]/AccretionFormulae.jl`, stating that it is a fork of the upstream.

There will also be two new buttons available to you, *Contribute*, and *Fetch upstream*. Click on both to see what they are used for – at the moment they will be greyed out, but we may use them when we wish to open a Pull Request (PR) later.

Cloning the fork

Next, we want to clone the git history and source code onto our machines so we can start working on our changes. For this, it is easiest to use the `git` command line tool.

Note that `git` is already available on most operating systems, but there are simple tutorials online to follow for installing it if you don't have it.

Open a terminal and type

```
1 git clone "[URL]"
```

to clone a repository from its URL. In my case, I run

```
1 git clone "https://github.com/fjebaker/AccretionFormulae.jl"
```

but the username in the URL will be different for you. The " are needed to escape any special characters that may be present in the URL.

The output of this command will look something like

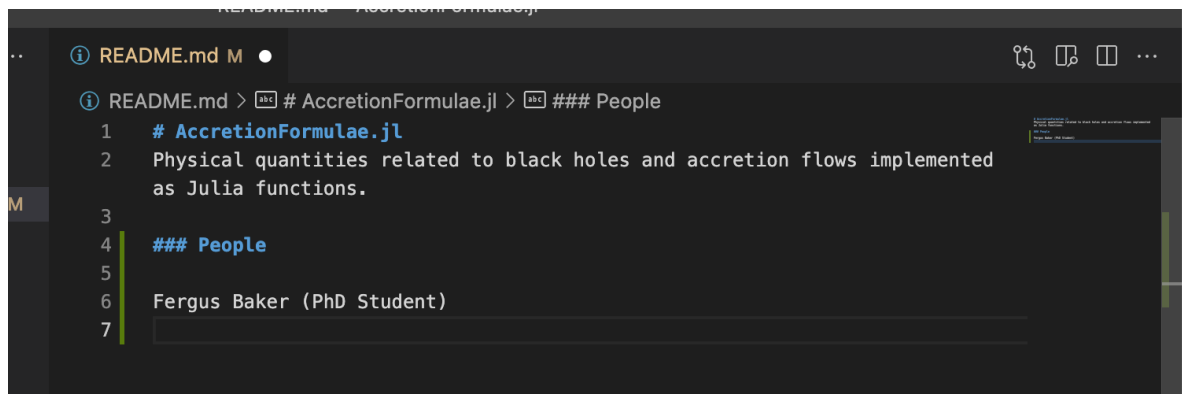
```
1 $ git clone "https://github.com/fjebaker/AccretionFormulae.jl"
2 Cloning into 'AccretionFormulae.jl'...
3 remote: Enumerating objects: 7, done.
4 remote: Counting objects: 100% (7/7), done.
5 remote: Compressing objects: 100% (7/7), done.
6 remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
7 Receiving objects: 100% (7/7), done.
8
9 $
```

We now switch to the newly created directory, and open the project in a text-editor of choice – I personally use VSCode with the Julia extensions, as I believe all of you do too.

```
1 cd AccretionFormulae.jl && code .
```

Making changes

Open the README.md file in your editor, and modify a line – for example, include your name and a little bit of info:



VSCode has a git-gutter extension installed by default, where it will highlight next to the line numbers what you have changed since the last time you added files to the git history. This can be quite handy when you're trying to remember what you modified since your last commits. You can also see that files you have modified appear yellow in the explorer, and newly created files appear green.

For this to really be useful, it's often best to commit small and incremental changes, but this is something you gain a feel for with use.

We can further check to see what status the git history is in with

```
1 git status
```

This will output something similar to

```
1 $ git status
2 On branch main
3 Your branch is up to date with 'origin/main'.
4
5 Changes not staged for commit:
6   (use "git add <file>..." to update what will be committed)
7   (use "git restore <file>..." to discard changes in working directory)
8     modified:   README.md
9
10 no changes added to commit (use "git add" and/or "git commit -a")
11
12 $
```

Status allows us to quickly see what we have changed since our last commit.

Making a commit and pushing changes

To make a commit, we first add all of the files that are relevant to our commit. This can be multiple, but in our case it's just the README.md file:

```
1 git add README.md
```

Running status now:

```
1 $ git status
2 On branch main
3 Your branch is up to date with 'origin/main'.
4
5 Changes to be committed:
6   (use "git restore --staged <file>..." to unstage)
7     modified:   README.md
8
9 $
```

If you accidentally add a file you didn't mean to, you can either use

```
1 git reset
```

to unstage all files, or

```
1 git restore --staged <file>
```

to unstage a specific file.

First time committing

If this is the first time you are committing on your machine, you will just need to tell `git` who you are in the config file. This is not a login, and does not have to correspond to any actual email or username, but it is used by GitHub to link profiles, so it's best to use the email and username corresponding with your GitHub account:

```
1 git config --global user.email "your@email.com"
2 git config --global user.name "github_username"
```

Using git commit

We then commit all the staged files with a short, descriptive commit message

```
1 git commit -m "credited Fergus in People"
```

If you need a longer commit message for whatever reason, you can also just write

```
1 git commit
```

to open the interactive commit prompt.

We can check the commit history now with

```
1 git log
```

and ensure our commit is there:

```
1 $ git log
2 commit 386457c1162702281f80149d6ec28ce49b3926d6 (HEAD -> main)
3 Author: fjebaker <fergusbkr@gmail.com>
4 Date: Thu Dec 2 14:04:02 2021 +0000
5
6     credited Fergus in People
7
8 commit fbab3834cbf786af65dae3c79919750f4b026dde (origin/main, origin/HEAD)
9 Author: Fergus Baker <fergusbkr@gmail.com>
10 Date: Thu Nov 11 13:20:53 2021 +0000
11
12     added LICENSE
13
14 commit b6c25135f93758f048c8a3e429d49aa9e2799232
15 Author: Fergus Baker <fergusbkr@gmail.com>
16 Date: Thu Nov 11 12:44:47 2021 +0000
17
18     Initial commit
19
20 $
```

Note, this also shows you where the different *HEADs* are, i.e. where different repositories are in the commit history. We see here that *origin/main* and *origin/HEAD* are both one commit behind *HEAD*. In this context, *origin* is our forked repository on GitHub, and *HEAD* is locally on our machine.

Using git push

Next, we can push our changes to our fork

```
1 git push origin main
```

Here we specify that we want to push our commits to the repository located at *origin*, we want to push the *main* (default) branch.

```
1 $ git push origin main
2 Enumerating objects: 5, done.
3 Counting objects: 100% (5/5), done.
4 Delta compression using up to 12 threads
```

```

5 Compressing objects: 100% (3/3), done.
6 Writing objects: 100% (3/3), 324 bytes | 324.00 KiB/s, done.
7 Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
8 remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
9 To github.com:fjebaker/AccretionFormulae.jl
10    fbab383..386457c  main -> main
11
12 $

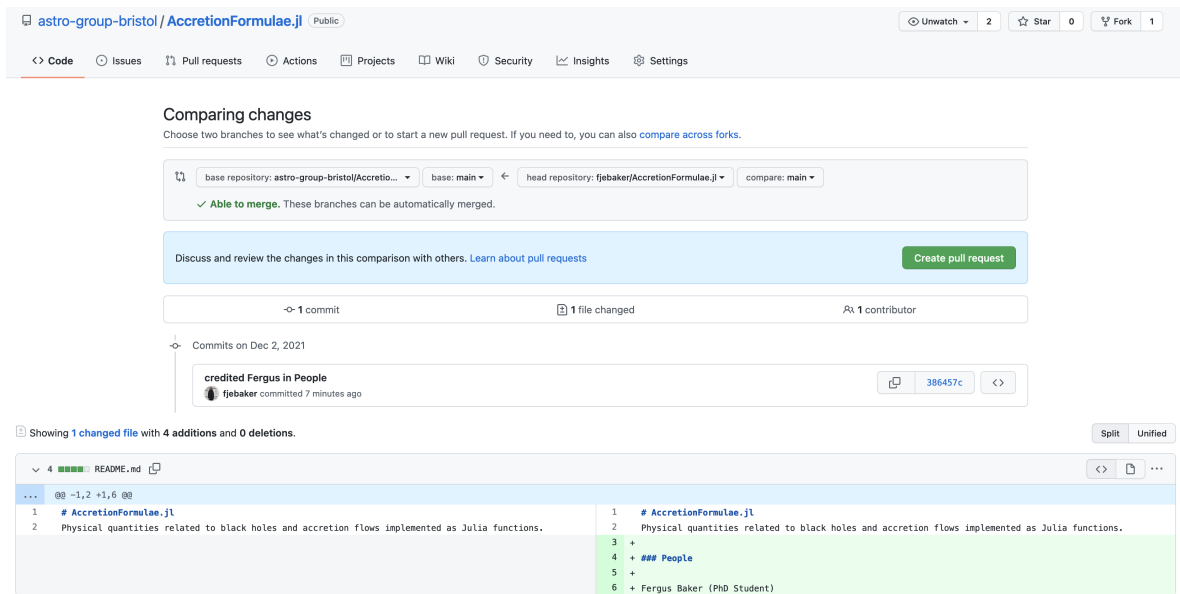
```

If we now navigate back to the GitHub website to view our fork repository, we see that our commits are there:

The screenshot shows the GitHub interface for a repository named 'AccretionFormulae.jl'. At the top, it indicates the current branch is 'main', which is 1 commit ahead of the upstream 'astro-group-bristol:main'. Below this, a commit by 'fjebaker' is highlighted, titled 'credited Fergus in People', with commit hash '386457c' and timestamp '5 minutes ago'. The commit message is 'credited Fergus in People'. The commit history table shows three commits: '.gitignore' (Initial commit, 21 days ago), 'LICENSE' (added LICENSE, 21 days ago), and 'README.md' (credited Fergus in People, 5 minutes ago). The 'README.md' file content is displayed below, showing the title 'AccretionFormulae.jl' and a description: 'Physical quantities related to black holes and accretion flows implemented as Julia functions.' Under the 'People' section, 'Fergus Baker (PhD Student)' is listed.

Opening a pull request

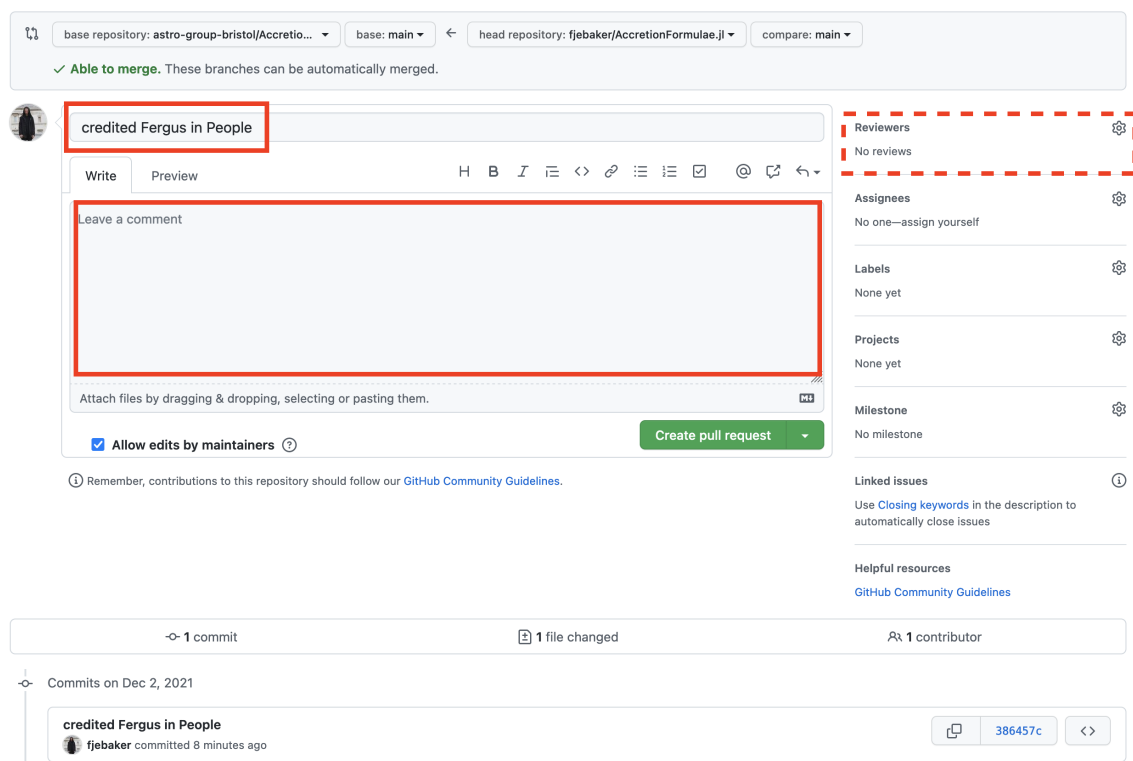
To open a pull request, we can now click the *Contribute* button, and select “Open pull request”. We will then be navigated to a comparison window, allowing us to see what changes we have in our fork compared to the upstream repository:



Here, we just select “Create pull request” again, and then modify a few different sections:

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](#).



We change the title to something descriptive, briefly document our changes by describing what we added, what we changes, and any bugs or issues we note.

Optionally, we can also request for another member of the repository to review the pull request – this

is useful if there is something you are unsure about, or are opening a large pull request.

Once we have filled these two sections in, we click “Create pull request” and wait for someone to check our work, discuss anything that may need changing, and eventually either reject or merge our changes:

The screenshot shows a GitHub pull request interface. At the top, the title is "Added credits to README #1". Below the title, it says "fjebaker wants to merge 1 commit into astro-group-bristol:main from fjebaker:main". The interface includes tabs for Conversation, Commits, Checks, and Files changed. A comment from fjebaker explains the changes: "As part of an introduction to the triangular workflow, I am adding a little 'People' section in the README for everyone to commit their name to." The changes listed are: "added People section in README" and "added Fergus Baker to People". A diff view shows the change in the README file. On the right, there are sections for Reviewers, Assignees, Labels, Projects, Milestone, Linked issues, and Notifications. At the bottom, there is a "Merge pull request" button and a "Leave a comment" section.

And we are done!

Syncing with upstream

Once the pull request has been merged, we want to sync our remote with the upstream repository again:

The screenshot shows a GitHub repository interface. At the top, it says "main", "1 branch", and "0 tags". Below this, a red box highlights a warning: "This branch is 1 commit ahead, 1 commit behind astro-group-bristol:main." To the right of this warning are buttons for "Contribute" and "Fetch upstream". Below the warning, there is a table of files: ".gitignore" (Initial commit), "LICENSE" (added LICENSE), and "README.md" (credited Fergus in People). To the right of the table, a red box highlights a button labeled "Fetch and merge".

This will synchronize changes, if simple, back into your remote, however it does so by creating a new merge commit.

If you just want to restore the git history of the upstream, we have to go back to the terminal and use git:

Configuring an upstream

We add a new *remote* called *upstream* pointing to the Bristol Astro Group repository:

```
1 git remote add upstream  
   "https://github.com/astro-group-bristol/AccretionFormulae.jl"
```

We can then fetch all of the branches from this remote using

```
1 git fetch upstream  
  
1 $ git fetch upstream  
2 remote: Enumerating objects: 5, done.  
3 remote: Counting objects: 100% (5/5), done.  
4 remote: Compressing objects: 100% (1/1), done.  
5 remote: Total 3 (delta 2), reused 2 (delta 2), pack-reused 0  
6 Unpacking objects: 100% (3/3), 699 bytes | 349.00 KiB/s, done.  
7 From https://github.com/astro-group-bristol/AccretionFormulae.jl  
8 * [new branch]      main      -> upstream/main  
9  
10 $
```

We then rebase our git-history to the upstream repository

```
1 git rebase upstream/main  
  
1 $ git rebase upstream/main  
2 Successfully rebased and updated refs/heads/main.  
3  
4 $
```

and force-push the branch back to our remote:

```
1 git push -f origin main  
  
1 $ git push -f origin main  
2 Total 0 (delta 0), reused 0 (delta 0), pack-reused 0  
3 To github.com:fjebaker/AccretionFormulae.jl  
4 + ae42dc0...c641871 main -> main (forced update)  
5  
6 $
```

Now our remote is even and everything is properly synced together:

main

1 branch

0 tags

Go to file


Add file




Code

This branch is up to date with astro-group-bristol:main.


Contribute

Fetch upstream

 fjbaker Added People to README (astro-group-bristol#1) c641871 8 minutes ago 3 commits

 .gitignore	Initial commit	21 days ago
 LICENSE	added LICENSE	21 days ago
 README.md	Added People to README (astro-group-bristol#1)	8 minutes ago

☰ README.md



AccretionFormulae.jl

Physical quantities related to black holes and accretion flows implemented as Julia functions.

People

Fergus Baker (PhD Student)