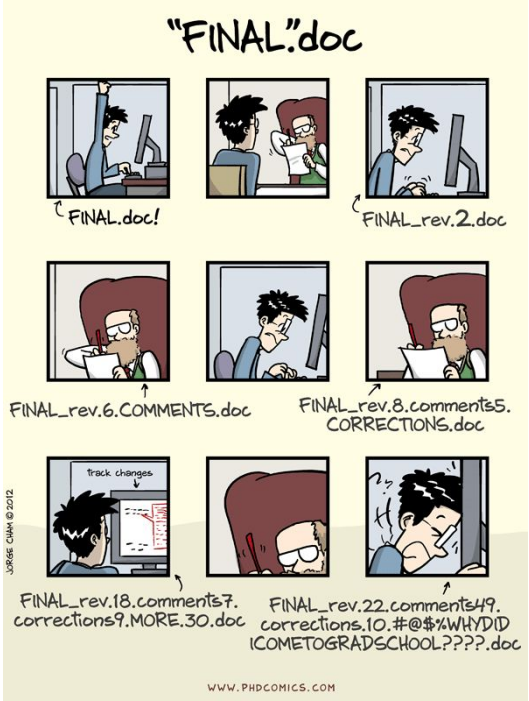
Introduction to git and GitHub

Andrew J. Stewart

E: drandrewjstewart@gmail.com
T: @ajstewart_lang
G: ajstewartlang

The Problem



Every had multiple versions of almost identical documents? It can be hard to keep track of the differences between versions and often even harder to know which is the 'final' version. Makes simultaneous collaboration tricky!

Wouldn't it be better if there was just one 'base' document and a record of the subsequent changes made to that document?

You could then easily see the most recent version, plus easily 'roll back' changes that had been made by reversing the changes (a little like using `undo`).

"Piled Higher and Deeper" by Jorge Cham, http://www.phdcomics.com

The Solution – Version Control

Version control software tracks changes that are made to files in a special kind of folder (called a repository or repo).

By using version control you are able to roll back to previous versions of your files, and for larger projects you can have multiple people collaborate on the project simultaneously.

It means you only ever have one file saved - all the changes made to that file are stored separately (think of this as a diary of changes).

What is git?

Git was created in 2005 by Linus Torvalds (creator of the Linux system) and is one of the most commonly used version control tools. It's free and open source.

https://git-scm.com/

What is git?

Git is used widely by scientists, developers, and researchers both inside and outside academia. Git is also an important tool for reproducibility.

What is GitHub?

GitHub is a git repository hosting service - basically, it allows you to store your git repositories in the cloud.

GitHub super useful as it gives you an easy backup for all your analysis code, data, documents etc. in your git repo and allows you to share all of these things with others.

Crucially, GitHub is one of the easiest ways to make your analysis code and data open - you can even give them a persistent doi (i.e., to make them citeable) via Zenodo.

GitHub as an individual vs. GitHub as part of a research team

You can use git and GitHub just for your own purposes as a way of helping your workflow using version control and as a way of keeping a backup of your important repositories.

You can also use git and GitHub collaboratively, where you will be forking other people repositories, creating a new branch, modifying content in that brand and then submitting a pull request for your changes to be incorporated back into the main repository via merging.

Wait, what? Forking?? Branching??? Pull requests???? Merging????? What is all this?

Don’t worry, let’s go through it one step at a time...

First we need to install git on our computers...

Installing on macOS...

Step 1 - open a Terminal window - the application looks like this
and it's likely to be located in your Utilities folder.

Step 2 - in the Terminal window, copy and paste the following - it will take a few minutes to install:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install.sh )"
```

Step 3 - in the Terminal window, now type the following:

```
brew install git
```

To check that everything is running, in the Terminal type the following:

```
git --version
```

If git has installed successfully, it should display which version of the software is installed.

Installing on Windows...

If you go to the following site, git will download automatically:

https://git-scm.com/download/win

Installing on Linux...

For Debian based systems such as Ubuntu, just type the following in the Terminal:

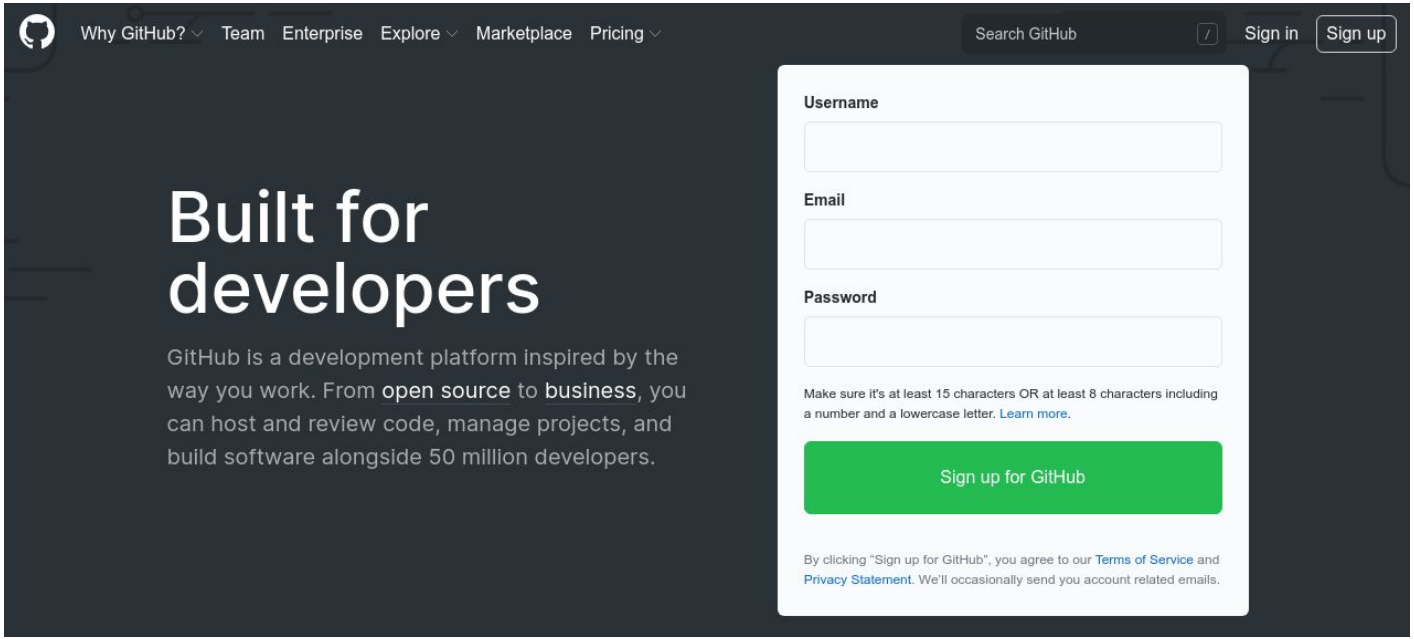
```
sudo apt install git-all
```

For other Unix distributions, follow the instructions here:

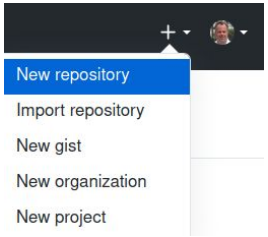
https://git-scm.com/download/linux

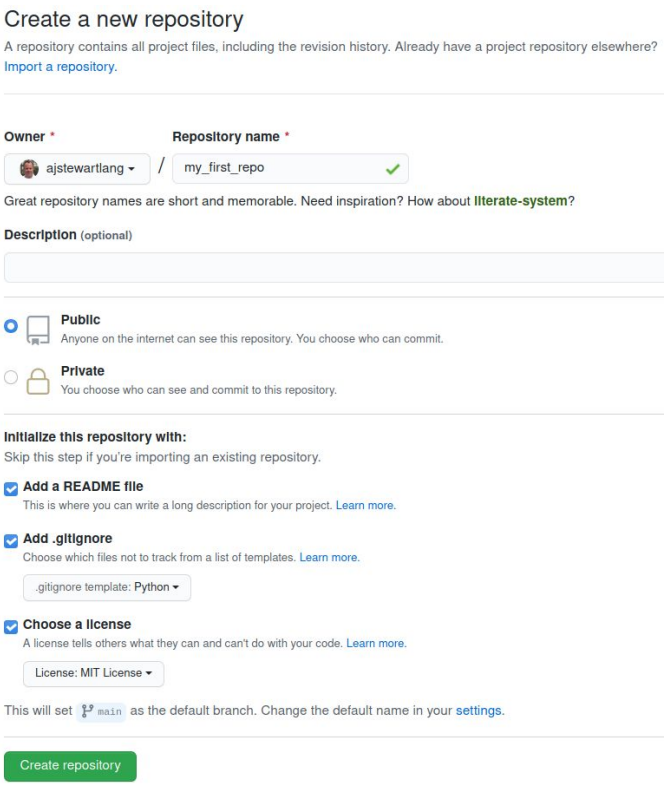
Create a GitHub account

Go to `github.com` and then create a username, add your email address, and create a password.



Now create a new repository...





Come up with a name for your repo - I've called mine `my_first_repo`

Tick the Add a README file box.

Tick the Add .gitignore box. You can use a default such as R or Python (or can also add something more specific later). This ensures that some files that might contain passwords etc. are not version controlled and stored on GitHub.

Tick the Choose a License box - the most permissive license is the MIT License so I tend to use that.

Click the green Create Repository box at the bottom.

main | 1 branch | 0 tags | Go to file | Add file | Code

ajstewartlang Initial commit `7f617fb` now 1 commits

.gitignore	Initial commit	now
LICENSE	Initial commit	now
README.md	Initial commit	now

README.md

my_first_repo

About

No description, website, or topics provided.

Readme

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

We now have our new repository on GitHub. Let’s get it onto our computer and connect our computer’s version with what’s on GitHub...

main 1 branch 0 tags Go to file Add file Code

ajstewartlang Initial commit

.gitignore	Initial commit
LICENSE	Initial commit
README.md	Initial commit

Clone

HTTPS SSH GitHub CLI

https://github.com/ajstewartlang/my

Use Git or checkout with SVN using the web URL.

Download ZIP

README.md

my_first_repo

About

No description, website, or topics provided.

Readme

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Click on this little clipboard icon to copy the link.

In a Terminal window, paste the address from your clipboard - but don't press Return:

```
https://github.com/ajstewartlang/my_first_repo.git
```

Now, edit it using the left arrow key - you want to replace the `github.com` bit of the address with your username and password - separated by a colon - and we want our password to be followed by `@github.com`. My username is `ajstewartlang` - let's pretend my password is `999-999-999` so for me the edit would be:

```
https://ajstewartlang:999-999-999@github.com/ajstewartlang/my_first_repo.git
```

Now, use the left arrow key and before the address type the words `git clone` as follows at the Terminal prompt where you want to create your repository and then press Return:

```
git clone https://ajstewartlang:999-999-999@github.com/ajstewartlang/my_first_repo.git
```

This will clone your repo that's on GitHub and create a local version on your own machine.

Now you're ready to do some version control with your new repo!

Create a new file in your git repo...

Using a word processor of your choice, create a file which contains a message (e.g., “Hello world!”) and save it as a .txt file in your git repository.

We need to add this file to our git repository, commit it (with a meaningful message), and then push it to the cloud version on GitHub. We’ll do all of the following on the command line using Terminal (or equivalent):

```
$ git add --all
$ git commit -m “this is my first commit”
$ git push
```

This will stage all files that have changed (with `git add --all)`, commit them (using `git commit`), and then push the changes in your local repo to GitHub so that your local repository and GitHub repository are synced with each other.

main 1 branch 0 tags Go to file Add file Code

ajstewartlang this is my first commit dbceb54 17 seconds ago 2 commits

.gitignore	Initial commit	28 minutes ago
LICENSE	Initial commit	28 minutes ago
README.md	Initial commit	28 minutes ago
my_new_file	this is my first commit	17 seconds ago

README.md

my_first_repo

I created a new file called `my_new_file` - we can see it's now on GitHub with the commit message beside it. Let's click on it...

main my_first_repo / my_new_file Go to file ...

ajstewartlang this is my first commit Latest commit dbceb54 2 minutes ago History

1 contributor

1 lines (1 sloc) | 13 Bytes Raw Blame

```
Hello World!
```

We can see that as we save it as a plain text file, the contents are displayed in our browser. If you click on the History icon in the top right, you'll see the history of commits we've made - in this case it's just the one. But if we were to edit the file on our local repo, go through the `git add`, `git commit`, and `git push` cycle again we willsee another commit appear in the history (as below)...If we click on this number, we'll actually be able to see what the changes are between the two versions of the file...

History for my_first_repo / my_new_file

Commits on Oct 30, 2020

this is my second commit
ajstewartlang committed now 4bf5b37

this is my first commit
ajstewartlang committed 5 minutes ago dbceb54

Newer Older

Showing **1 changed file** with **2 additions** and **0 deletions**.

Unified | Split

2 my_new_file

```
@@ -1 +1,3 @@
  Hello World!
+
+ And Hello Universe!
```

You can see the file has had 2 additions in that second commit - I've added a blank line, and then another line of text with the words "And Hello Universe!" If we were to click on the my_new_file link in my GitHub repo, we would see this:

main | **my_first_repo** / **my_new_file** | Go to file

ajstewartlang this is my second commit — Latest commit `4bf5b37` 3 minutes ago — **History**

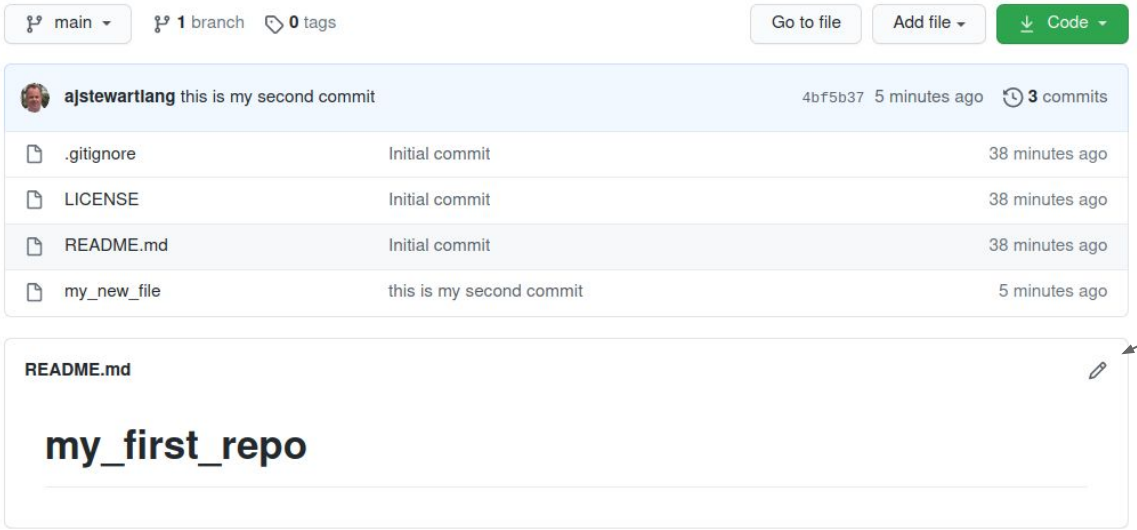
1 contributor

3 lines (2 sloc) | 34 Bytes — Raw | Blame

```
Hello World!

And Hello Universe!
```

Editing our README file



Our README.md file should contain some useful information about our repository.

Let's edit it by clicking on this little pencil icon.

We can then write the README.md file using the Markdown language.

my_first_repo / README.md Cancel

Edit file | Preview changes | Spaces | 2 | Soft wrap

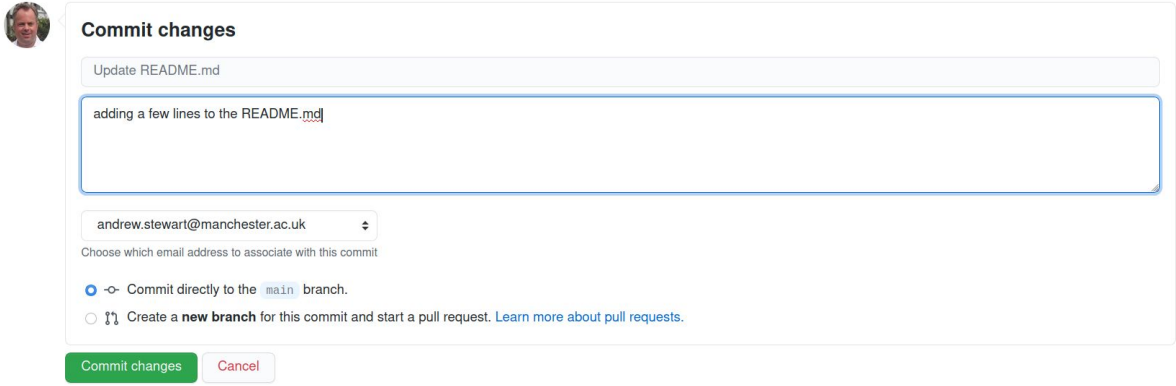
```
# This is my first repo

You can write text here using Markdown - or should I write in bold as **Markdown**?

You can also add links here to webpages such as my own website using Markdown syntax such as [here is my website](https://ajstewartlang.netlify.app/).

```

Once you've made the changes you want to, add a commit message (scroll to the bottom of the page) and then make the commit...



Your repo now looks like this...

main | 1 branch | 0 tags | Go to file | Add file | Code

ajstewartlang Update README.md `b208fdb` 24 seconds ago | **5** commits

.gitignore	Initial commit	1 hour ago
LICENSE	Initial commit	1 hour ago
README.md	Update README.md	24 seconds ago
my_new_file	this is my second commit	12 minutes ago

README.md

This is my first repo

You can write text here using Markdown - or should I write in bold as **Markdown**?

You can also add links here to webpages such as my own website using Markdown syntax such as here is my website.

Our GitHub repo is now one commit ahead of our local repo...

We need now to re-sync our two repositories so our local repo is up to date and reflects this new commit to the repo on GitHub.

We can use the `git pull` command in a Terminal window to pull the new commits on GitHub so they are also present in our local repo:

```
$ git pull
Updating 4bf5b37..b208fdb
Fast-forward
 README.md | 6 +++++-
 1 file changed, 5 insertions(+), 1 deletion(-)
```

So we now have the README.md file in our local repo too!

Recap so far...

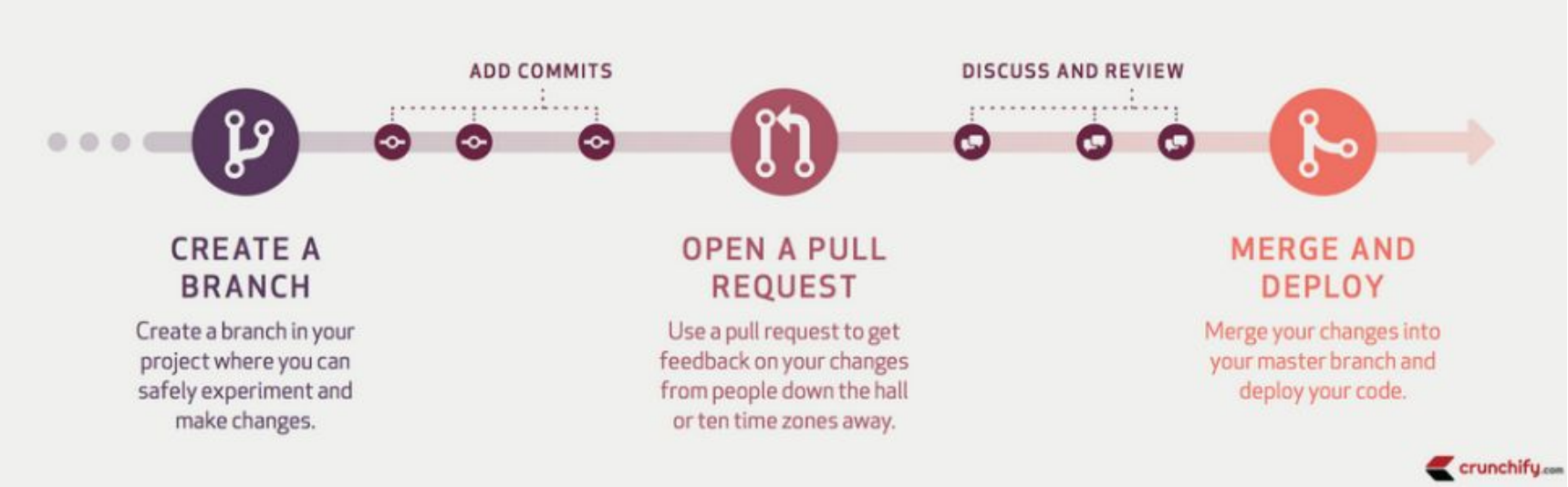
Once you have created a repo on GitHub and cloned it locally to your own machine, you can treat it like any other folder. Just remember, as you edit, add, and delete files follow each of these with a `git add`, `git commit` pair of commands. And then when you want to sync them with GitHub, use `git push`.

Andrew's top tip - commit often and write meaningful commit messages!

If you - or someone else - has made changes to your GitHub repo, you need to do a `git pull` before you start working on your repo locally. This will help avoid merge conflicts - where the local and GitHub repos get out of step with each other.

Using GitHub to work on someone else's repo...

Imagine you want to modify something in the repo that is owned by someone else. You'll fork the original repo to your own account, make your modifications, then ask for your modifications to be incorporated back into the original repo via a pull request and merge.



https://crunchify.com/how-to-fork-github-repository-create-pull-request-and-merge/

metalguru668 / **the_best_bands_in_the_world**

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

main | 1 branch | 0 tags | Go to file | Add file | Code

metalguru668 Create list_of_best_bands `52c4da0` 7 minutes ago | 3 commits

README.md	Update README.md	9 minutes ago
list_of_best_bands	Create list_of_best_bands	7 minutes ago

README.md

the_best_bands_in_the_world

This is the original repo.

The repo “the_best_bands_in_the_world” is owned by the account metalguru668. Let’s look at the file “list_of_best_bands” by clicking on its name...

main

the_best_bands_in_the_world / list_of_best_bands

Go to file

metalguru668 Create list_of_best_bands

Latest commit 52c4da0 7 minutes ago History

1 contributor

7 lines (7 sloc) | 73 Bytes

Raw Blame

```
Opeth
Celtic Frost
Blue Oyster Cult
Jethro Tull
Slayer
Metallica
Anthrax
```

I see there are 7 bands listed here - but no Devin Townsend Project! Let's work on this repo and fix that!

First we need to fork a copy of this repo to our own account...

metalguru668 / **the_best_bands_in_the_world**

Watch 1 | Star 0 | Fork 0

Fork your own copy of metalguru668/the_best_bands_in_the_world to your account

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

main | the_best_bands_in_the_world / **list_of_best_bands** | Go to file

metalguru668 Create list_of_best_bands — Latest commit 52c4da0 10 minutes ago — **History**

1 contributor

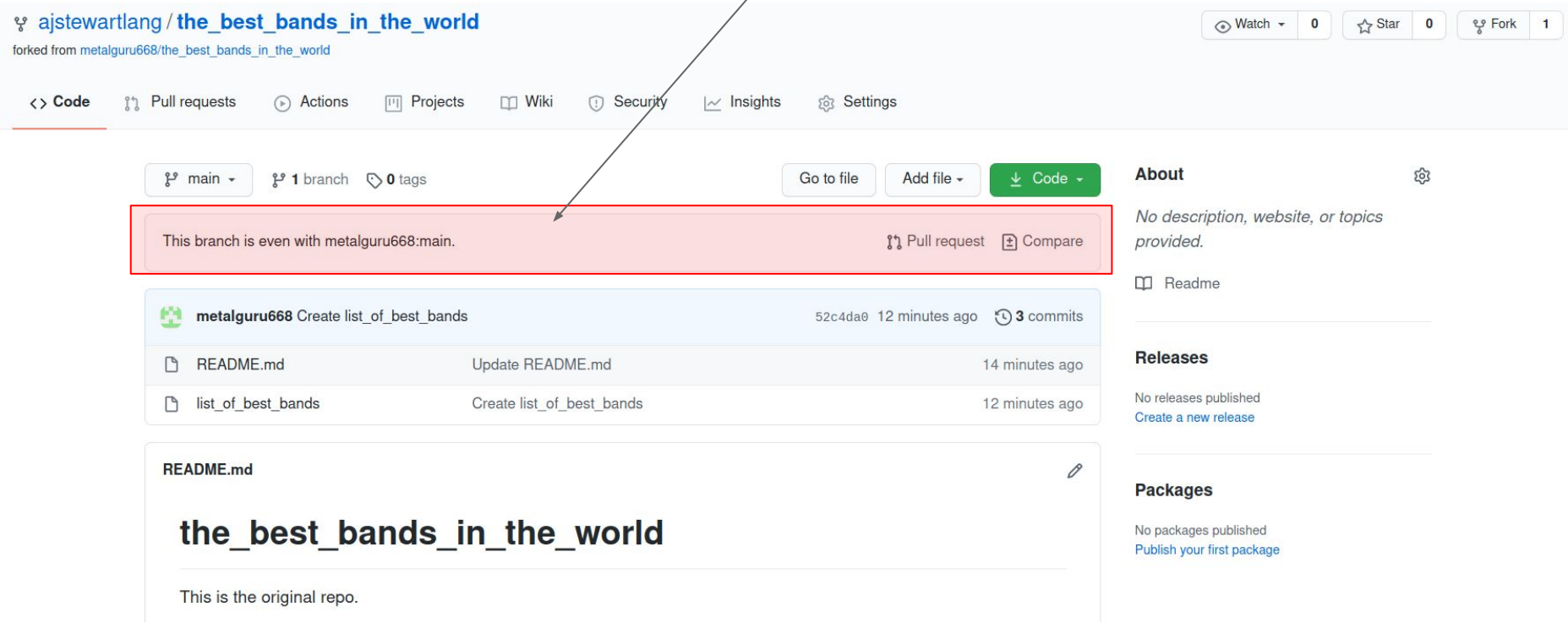
7 lines (7 sloc) | 73 Bytes — Raw | Blame

```
Opeth
Celtic Frost
Blue Oyster Cult
Jethro Tull
Slayer
Metallica
Anthrax
```

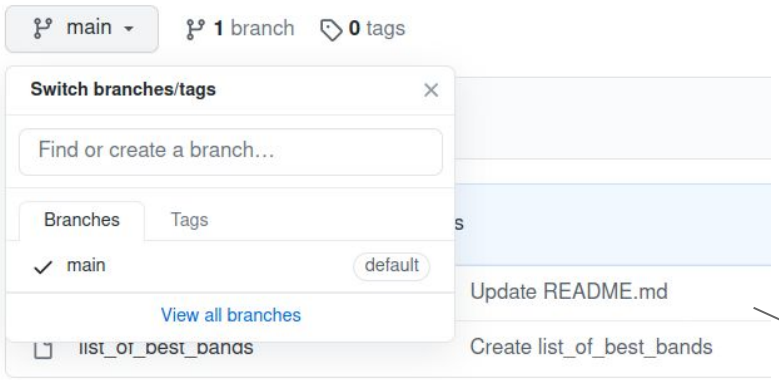
Click here to fork the repo to our own account...

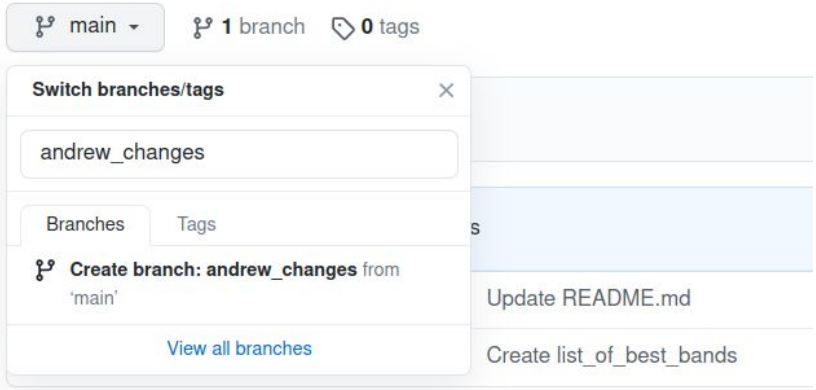
Now we can see we have a copy of the repo “the_best_bands_in_the_world” in our own account.

We see that this version is even with the original - if you click on the `Compare` button it will highlight any changes.



To start editing the file in the repo, let’s create a new ‘branch’ - so that we’re not working in the ‘main’ branch - this is important! Call this branch something meaningful like “andrew_changes” and then press <return>...





As you can see from the below, we're now working in the `andrew_changes` branch...

andrew_changes | 2 branches | 0 tags | Go to file | Add file | Code

This branch is even with metalguru668:main. Pull request | Compare

metalguru668 Create list_of_best_bands 52c4da0 27 minutes ago 3 commits

README.md	Update README.md	29 minutes ago
list_of_best_bands	Create list_of_best_bands	27 minutes ago

Let's now click on the "list_of_best_bands" file so we can edit that...

Then click on the “Edit this file” pencil icon...

andrew_changes

the_best_bands_in_the_world / list_of_best_bands

Go to file

metalguru668 Create list_of_best_bands

Latest commit 52c4da0 30 minutes ago

History

1 contributor

Edit this file

7 lines (7 sloc) | 73 Bytes

Raw Blame

```
Opeth
Celtic Frost
Blue Oyster Cult
Jethro Tull
Slayer
Metallica
Anthrax
```

Edit file | Preview changes

```
1  Opeth
2  Celtic Frost
3  Blue Oyster Cult
4  Jethro Tull
5  Slayer
6  Metallica
7  Anthrax
8  Devin Townsend Project
```

Add the Devin Townsend Project line...

Commit changes

added Devin Townsend Project

Add an optional extended description…

andrew.stewart@manchester.ac.uk

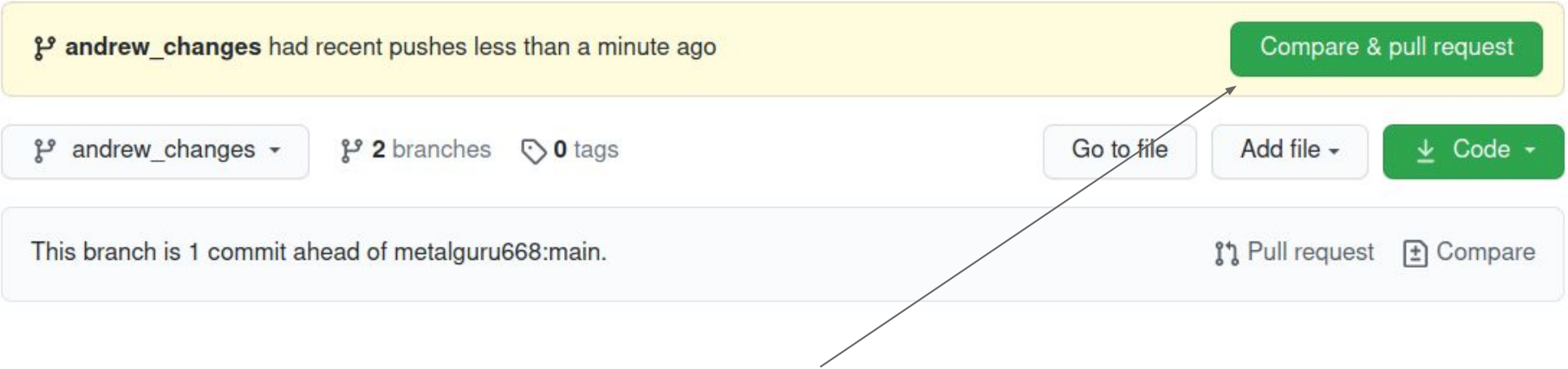
Choose which email address to associate with this commit

Commit directly to the `andrew_changes` branch.

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

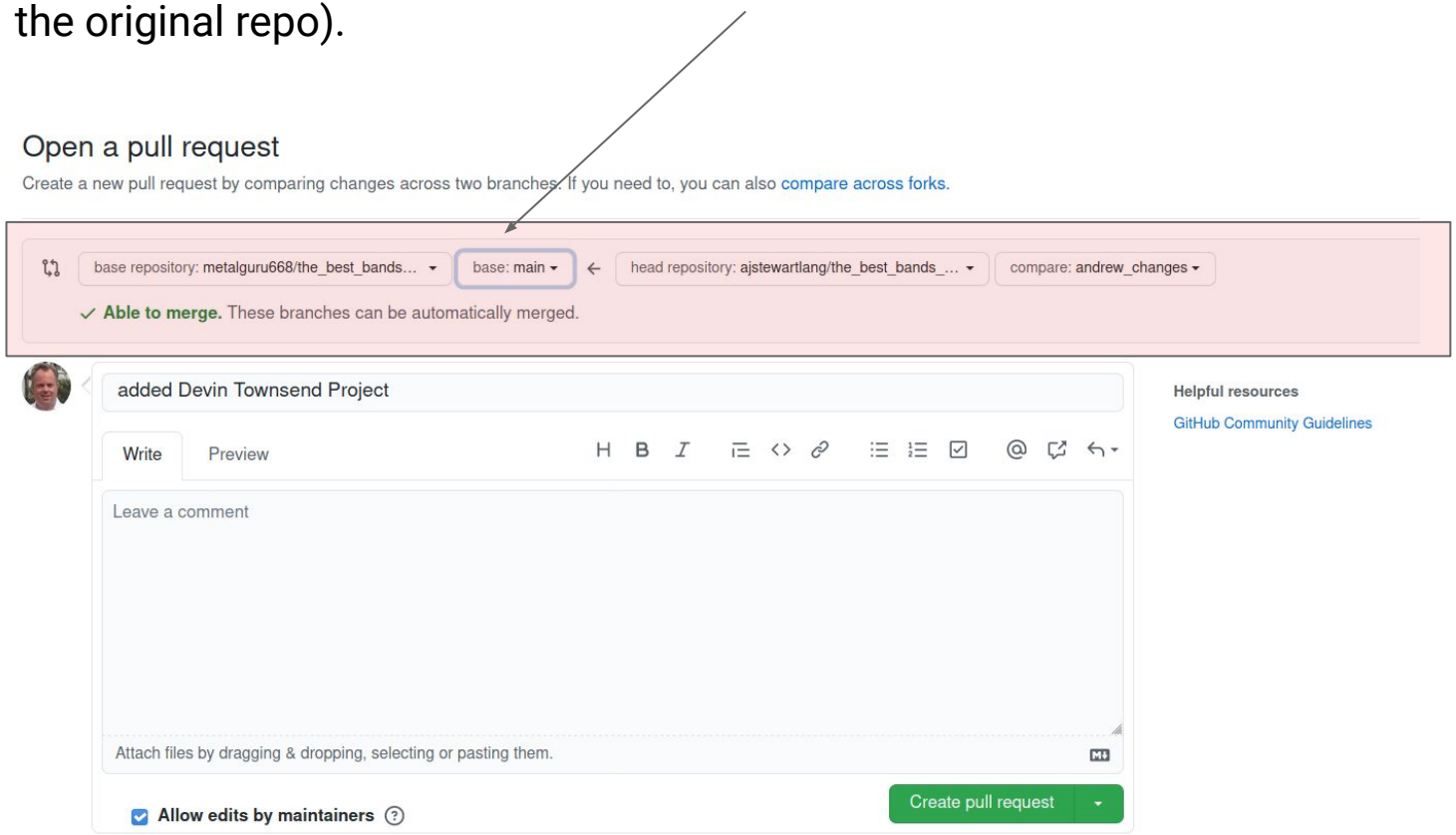
Commit changes | Cancel

Add a meaningful Commit message and then clock on the green Commit changes button...



Let’s click on the green “Compare & pull request” button...

We see a comparison between the main branch of the original repo and the andrew_changes branch of the forked repo in our own account. We also see that the two branches can be merged (in other words, our addition of the Devin Townsend Project can be incorporated into the original repo).



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 repository: metalguru668/the_best_bands… | base: main | head repository: ajstewartlang/the_best_bands_… | compare: andrew_changes

✓ **Able to merge.** These branches can be automatically merged.

added Devin Townsend Project

Write | Preview

I have added the Devin Townsend Project as one of the best bands - it was missing from the original list.

Attach files by dragging & dropping, selecting or pasting them.

Allow edits by maintainers

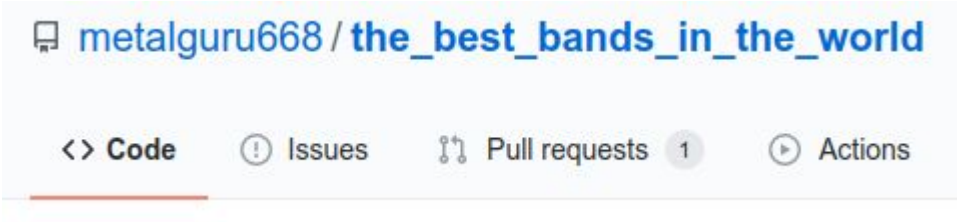
Create pull request

Helpful resources

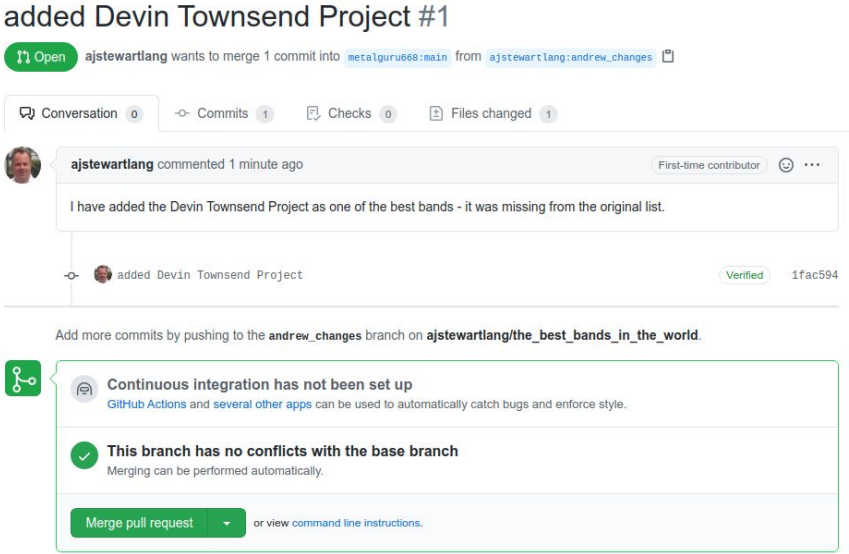
GitHub Community Guidelines

Add a comment explaining what your pull request (PR) is for and then click on the green “Create pull request” button...

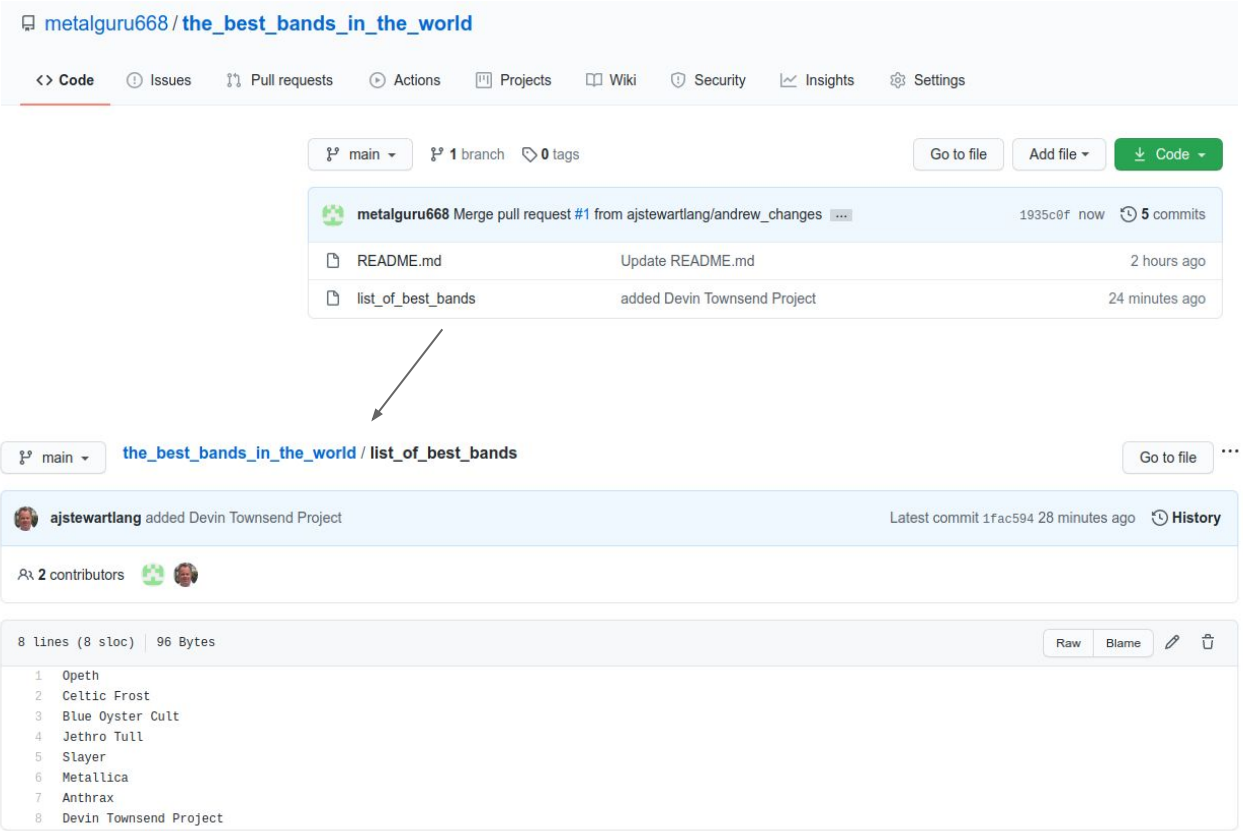
If metalguru668 now logs into their GitHub account they'll see this pending pull request...

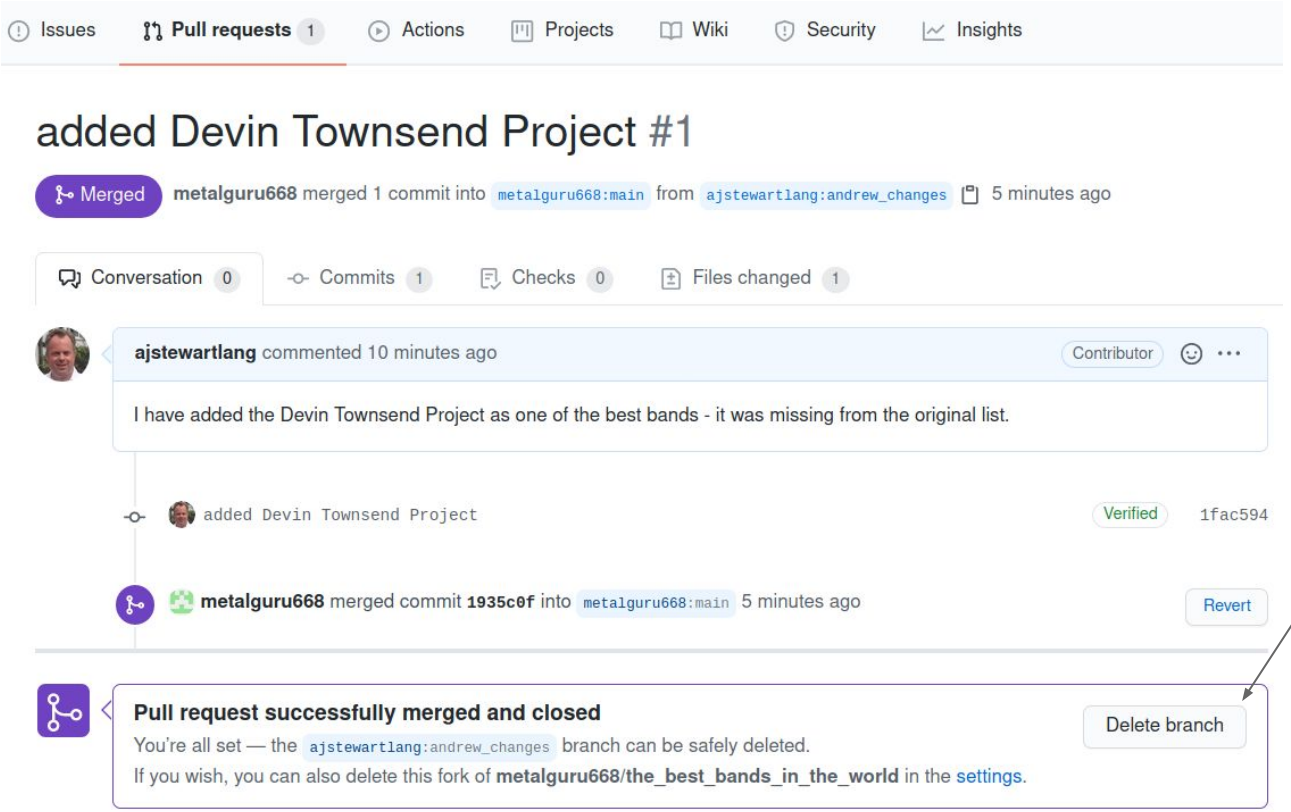


And if they click on it, they can see the modifications can be merged into the original repo...



Once the merge has been confirmed by metalguru668, you'll see the original repo updated with our commit message, and the file list_of_best_bands in metalguru668's repo now includes our Devin Townsend Project addition...





If you look at the Pull requests tab in your own forked version of the repo, you'll see confirmation that your branch has been merged with the original in metalguru668's repo.

You should now delete your branch as it has served its purpose.

You can also delete the forked repo via Settings - this way you don't have to worry about your fork going out of sync with the original repo.

Your task...

Fork this repository:

https://github.com/metalguru668/the_best_bands_in_the_world

Add your own favourite band (doesn't have to be rock/metal!) and do a pull request (PR). I'll then incorporate your changes back into the main list. After everyone has done a PR, this list will contain the list of everyone's favourite bands...

Summary

Git is an incredibly powerful tool for version control - never again will you struggle to remember which was the ‘final’ version of your script/analysis/paper.

Combined with GitHub, git offers the perfect environment for collaborating with others both inside and outside your organisation. Being able to use git and GitHub will allow you to work on large-scale global projects and be part of a broad and diverse community of researchers, coders, and data scientists.

We’ve only just scratched the surface of what git and GitHub and do for you! You can read more about git (incl. the full set of git commands) and GitHub here:

https://git-scm.com/

https://guides.github.com/activities/hello-world/