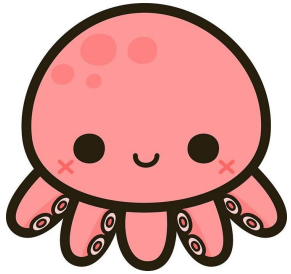


OCTOPUS

Onboarding & **C**ollaborative **T**utorial for
Open-source **P**rojects **U**tilised in **S**cience

What we'll cover here

What and how **git** and **github** works.

What is the **nu-ZOO**? And why should you care.

How to interact with **big collaborative repositories** (at least some of them).

- Making branches,
- adding changes,
- resolving issues,
- rebasing.

General style guide & tests.

A fun game to end the day (better have your laptops!)

What is git & github?

Git is “the information manager from hell”.

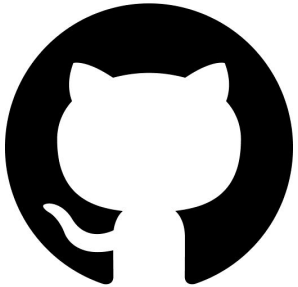
Source control software that allows:

- For the tracking and managing of changes to code over time,
- easy collaboration with fast branching and merging of many peoples work into one repository,
- the ability to explore project histories and revert to earlier versions.

Github is (generally) where these repositories are stored, and comes with lots of tools:

- ‘organisations’ and publicly available repositories,
- automatic testing suites,
- ‘forums’ for discussing issues, discussing changes, etc
- even more, Github and git have lots of features.

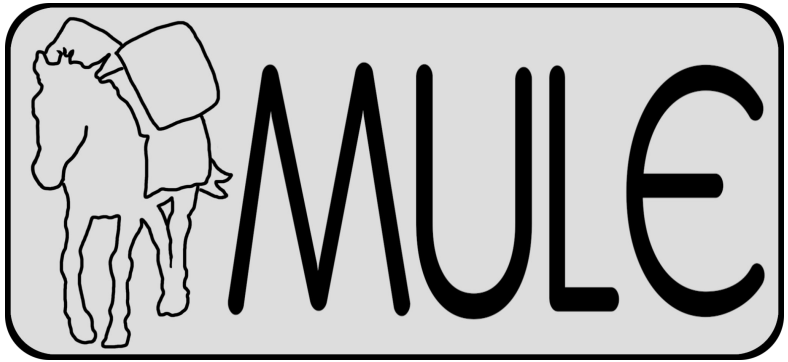
In practice, you’ll be working with both!



What is nu-ZOO?

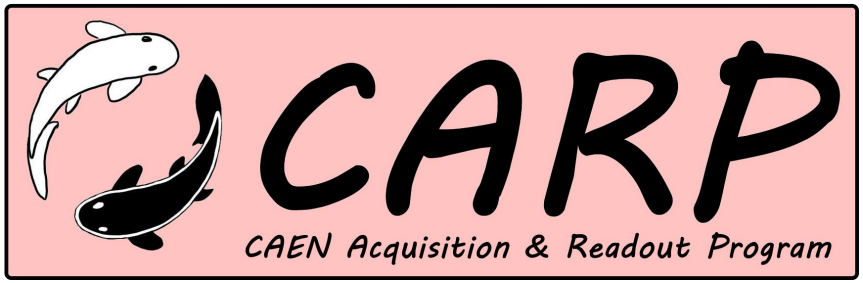
Organisation on github made by me and Brais to host any relevant working software repositories:

- MULE - **M**easurement and **U**tilisation of **L**ight **E**xperiments → decoding and analysis framework



- MARE - **M**ULE **A**rena for **R**ecursive **E**nhancement → Messy storage of work related to MULE

- CARP - **C**aen **A**cquisition and **R**eadout **P**rogram → WIP GUI-based software for working with CAEN digitisers.



- OCTOPUS - **O**nboarding & **C**ollaborative **T**utorial for **O**pen-source **P**rojects **U**sed in **S**cience.

Why work within nu-ZOO?

Developing good software development practices will make your life **easier** in the long term.

(It's also a very marketable skill).

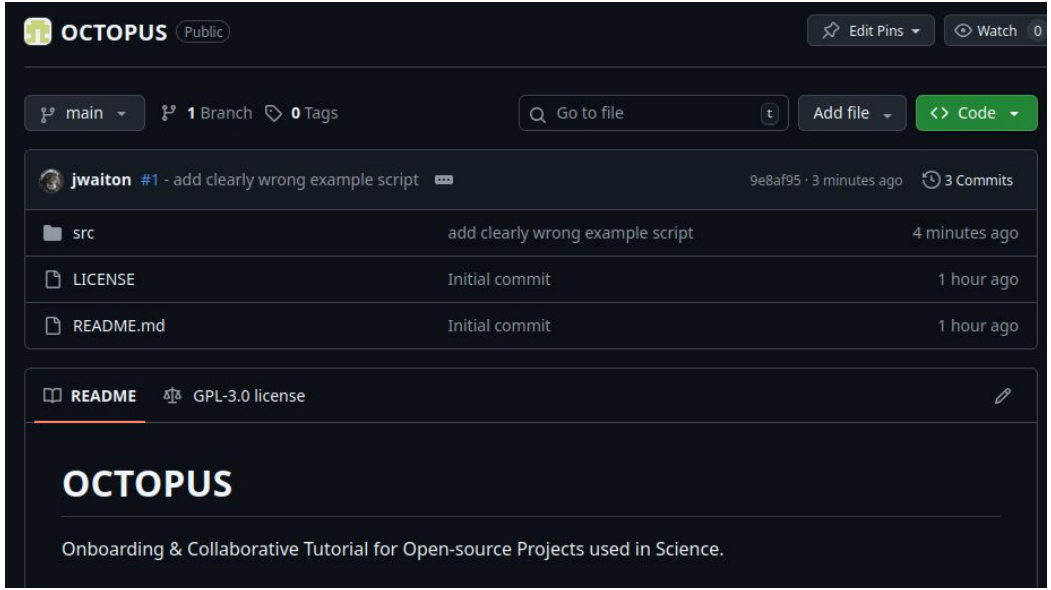
Stops duplication of the **near identical code**. If someone has written a baseline subtraction algorithm already, why waste two days writing a new one?*

Collaborating in shared frameworks allows for code to be more easily adapted and used for differing tasks (MULE).

Work through an example: OCTOPUS

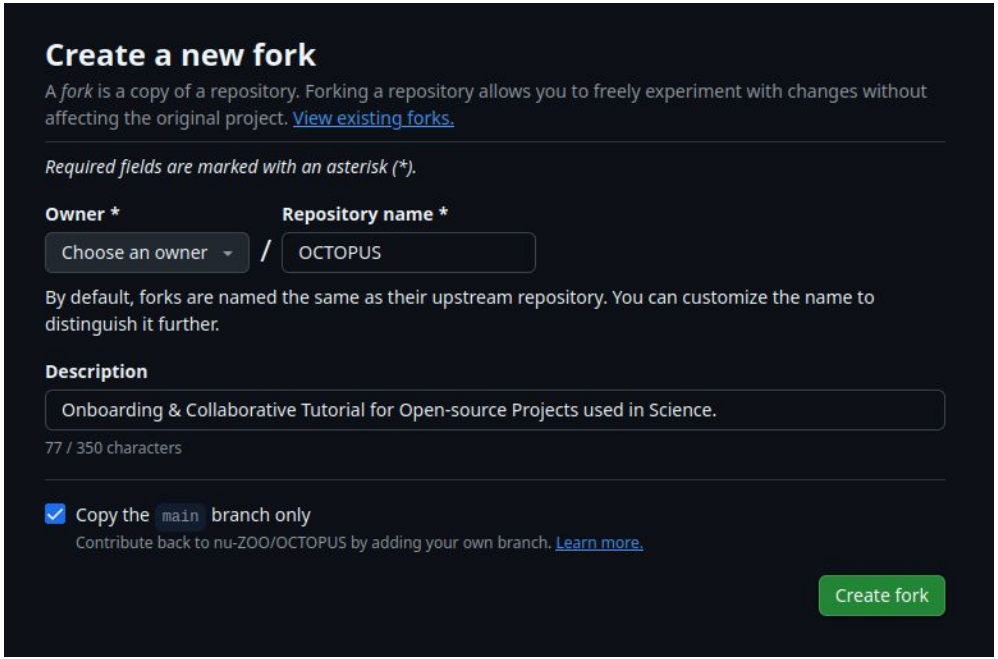
OCTOPUS - Onboarding & Collaborative Tutorial for Open-source Projects used in Science.

We'll work through a simple example that will go through all the steps here.



Work through an example: OCTOPUS

First, fork the repo:



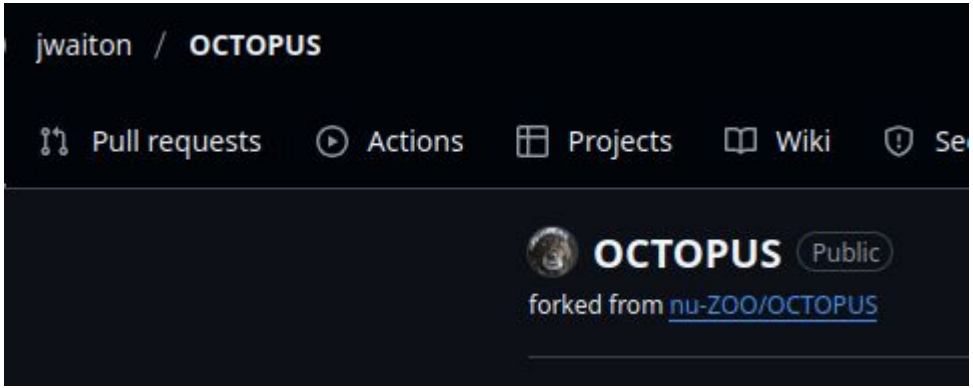
Work through an example: OCTOPUS

Once you have a fork (should look like this), you can pull this locally to work on it!

```
git clone https://github.com/jwaiton/OCTOPUS.git
```

This will be your ‘origin’

Whats an origin?



SLIDE OF UPSTREAM, REMOTE, LOCAL

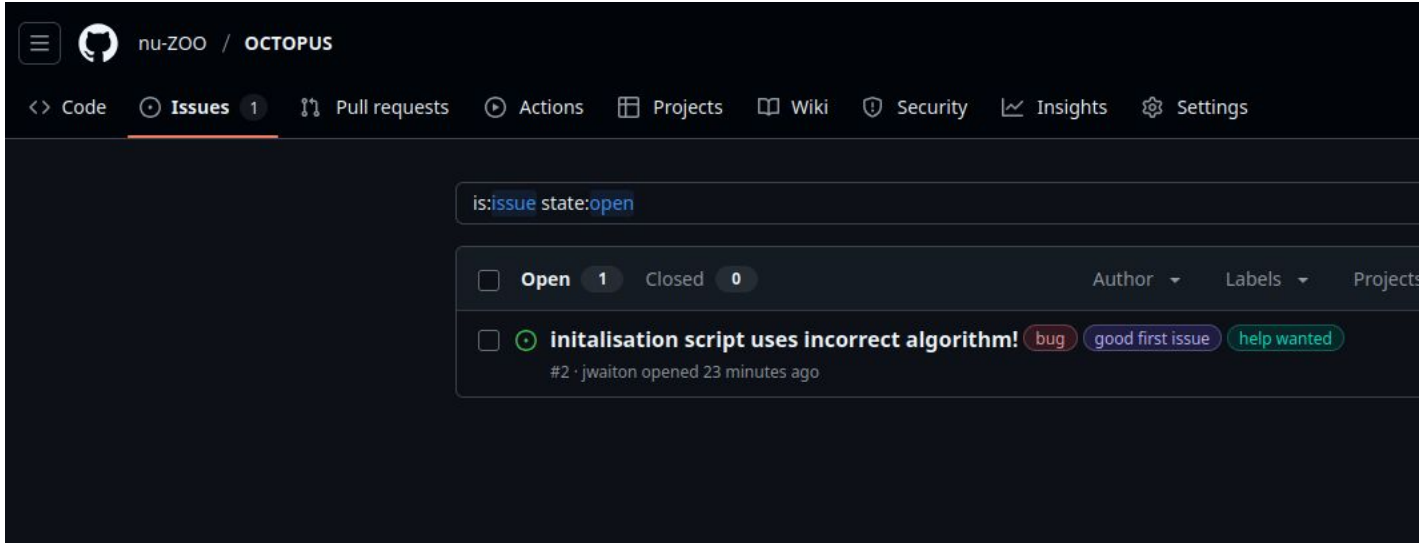
VISUAL OF THIS HERE, BRANCHES TOO

Work through an example: OCTOPUS

Okay, you have your fork. Let's get to writing some code!

You can either create **new features**, or work on **issues**.

There already exists an issue!



Work through an example: OCTOPUS

Lets resolve the issue, create a new branch to work in:

```
git checkout -b fix-incorrect-acronym
```

To look at the history (and where you are within it), use `git log`

```
e78368jw@e-10lux3072wzz:OCTOPUS$ git log
commit a8d74dc5e15cebaa392b3829ffa9db3b35054595 (HEAD -> main, origin/main, origin/HEAD)
Author: jwaiton <john.waiton@postgrad.manchester.ac.uk>
Date:   Mon Oct 6 15:24:13 2025 +0100

    add clearly wrong example script

commit 8b2ab96ce66fda11027c9a7bfb678e5f135c3a11 (upstream/main)
Author: John Waiton <john.waiton@postgrad.manchester.ac.uk>
Date:   Mon Oct 6 13:53:13 2025 +0100

    Initial commit
e78368jw@e-10lux3072wzz:OCTOPUS$
```

Work through an example: OCTOPUS

After making the changes, you can see what you've done with:

```
git status
```

and in more detail:

```
git diff FILE_NAME_HERE
```

```
e78368jw@e-10lux3072wzz:OCTOPUS$ git status
On branch main
Your branch is up-to-date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/example_code.py

no changes added to commit (use "git add" and/or "git commit -a")
e78368jw@e-10lux3072wzz:OCTOPUS$
```

Work through an example: OCTOPUS

To apply your changes:

```
git add FILE_AND_PATH_HERE
```

This ‘stages’ your file, prepared to be ‘committed’ like so

```
git commit -m ‘MESSAGE EXPLAINING WHAT YOUR COMMIT DOES’
```

You can commit multiple files at once.

Make your commits **imperative.** (for style)

Make your commits **small.** (for ease of reversion)

```
nothing to commit, working tree clean
e78368jw@e-10lux3072wzz:OCTOPUS$ git status
On branch main
Your branch is up-to-date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/example_code.py

no changes added to commit (use "git add" and/or "git commit -a")
e78368jw@e-10lux3072wzz:OCTOPUS$
```

Work through an example: OCTOPUS

To apply your changes:

```
git push
```

You may need to assign a ‘target’, but we get there when we get there.

```
nothing to commit, working tree clean
e78368jw@e-10lux3072wzz:OCTOPUS$ git status
On branch main
Your branch is up-to-date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/example_code.py

no changes added to commit (use "git add" and/or "git commit -a")
e78368jw@e-10lux3072wzz:OCTOPUS$
```

Work through an example: OCTOPUS

PRs

```
nothing to commit, working tree clean
e78368jw@e-10lux3072wzz:OCTOPUS$ git status
On branch main
Your branch is up-to-date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/example_code.py

no changes added to commit (use "git add" and/or "git commit -a")
e78368jw@e-10lux3072wzz:OCTOPUS$
```

Work through an example: OCTOPUS

Rebasing! SET IT UP with multiple wasteful commits to demonstrate first rebasing, then interactive rebasing

```
e78368jw@e-10lux3072wzz:OCTOPUS$ git status
On branch main
Your branch is up-to-date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/example_code.py

no changes added to commit (use "git add" and/or "git commit -a")
e78368jw@e-10lux3072wzz:OCTOPUS$
```