FOXSI packages Github instructions

Rev 1

Github tutorial

To help keep track of all the software and code being created for FOXSI, there have been Github repositories (or repos) created for each detector system.

- CdTe (https://github.com/foxsi/cdte-tools)
- CMOS (https://github.com/foxsi/cmos-tools)
- Timepix (https://github.com/foxsi/timepix-tools)

The following tutorial assumes

- you have a Github account.
- Your computer you will work on has `git` installed

Why Github and why this way?

Instead of reading any justification below, here is the TLDR:

- Your code contributions **do not** need to be in their amazing, final forms
- We do not have to worry about tracking old versions of the code
- We only have to be aware of the one location where the most up-to-date code lives
- Loads of small code contributions are better than a few very large ones

First of all, it is very important that any contributed code ***does not*** need to be in a beautiful and pristine "final" form. The main goal here is to try and help individuals, as well as collaborators, keep track of the code they have written for the project.

If changes are made to the code that Github is keeping track of it will log those changes. This means that any previous work is never truly lost, we can always look at or revert back to previous versions. This also means that we will all effectively be working one version of the code and so removes the chance of multiple different versions of the analysis code being spread across several people; everyone can then just focus on the main FOXSI package code and be aware this is where the latest version of the code always lives.

Further on this, when it comes to contributing code, making loads of small contributions might seem silly if you are working towards adding a larger tool into the package but it is actually far easier to manage loads of small contributions than a few very large ones.

Most importantly, this is supposed to make the code collaboration easier, not more difficult, so if there is any doubt on what to do then do not spend loads of time on trying to work it out and please get in touch (email, slack, etc.) with someone able to help.

Please, do not spend loads of time on the Git/Github side of things. Please contact someone to help (Kris, Thanasi, etc.). We will be more than happy to help.

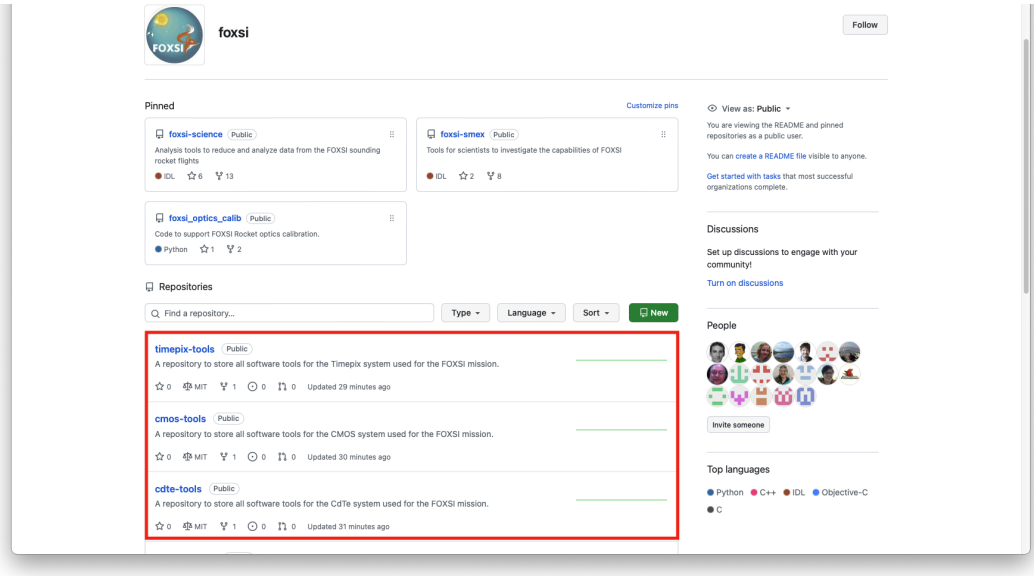
Getting your own copy of the code to work on

In order for multiple people to contribute to a repository at the same time, it is good practice to have your own copy of the software you are working on. Changes can be made to your own version which can then be proposed to the main FOXSI repository.

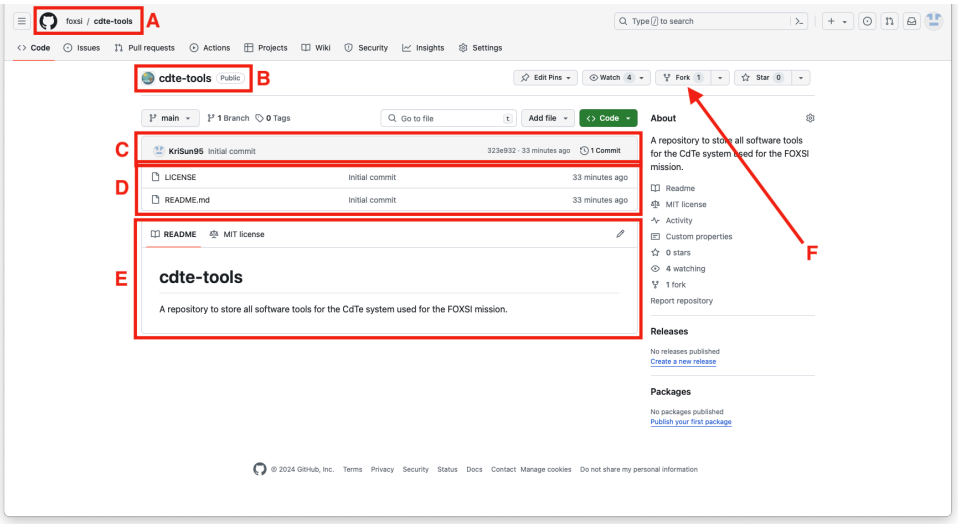
Don't worry if this process doesn't make too much sense at this stage, it will likely only become clearer when you step through the process a number of times for real.

To get your own version of the code

1. Go to the FOXSI Github page (https://github.com/foxsi) and find the system repo you want

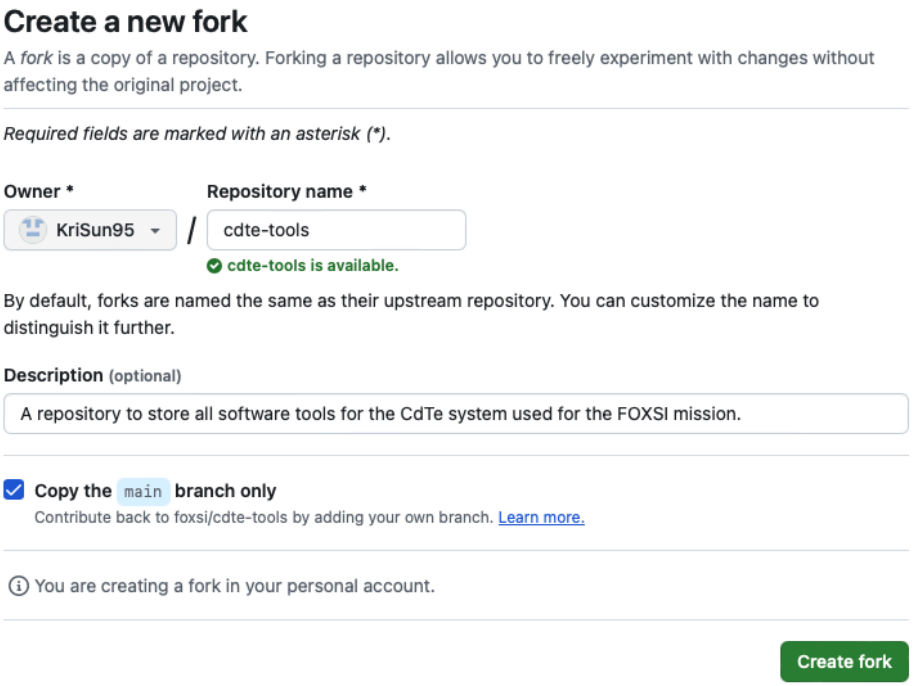


   a. Or use the individual links at the top of this document
2. Picking `cdte-tools` as an example, we go to the repository for the CdTe code
   a. At the minute, the repo is a bit empty but there is plenty of information



   b. A: owner/repo name, B: repo name, C: latest activity in the repo, D: files and folders in the repo, E: short package description, F: the "Fork" button

3. To get your own version of the `cdte-tools` repo, just click the “Fork” button
    a. Press “Create fork” on the following page



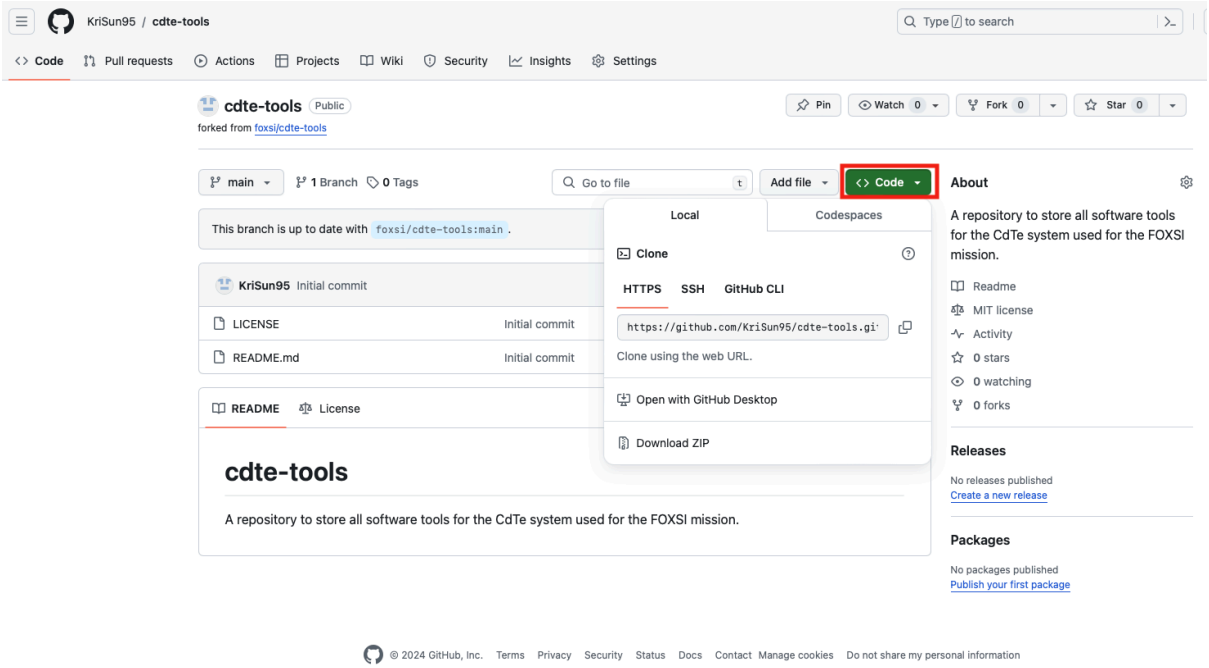
    b. After this, you will have your very own copy of the `cdte-tools` repo on your Github account

Being able to work on your repo on your own machine

Now you have your own repo of the code (your own Fork), it just has to be downloaded onto your own machine. Here you can make changes on your machine and easily back them up to your own fork. When you want your changes in the main FOXSI repo you can move your changes from your Fork.

Code from Github to your machine

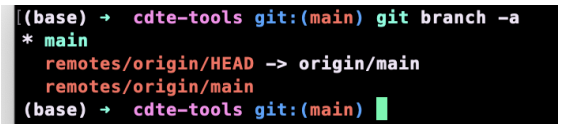
1. Go to your Github page and locate your Fork of the appropriate repo (I will know I'm on my Fork because my username `KriSun95` is at the top left of the page)
    a. Again, I will use `cdte-tools` as an example
    b. Locate the green "Code" button and click it



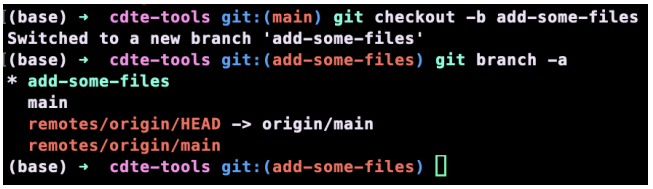
    c. This shows a few options but the one we was is to copy the HTTPS URL that is displayed
        i. Copy by clicking on the two small overlapping boxes to the right of the URL
2. Next, on your machine you will be working on, navigate to the folder you want to keep the code to work on (using the terminal)
    a. I am saving mine in a folder called `cdte-tools-main`
3. Type `git clone <COPIED_URL>` into your terminal
    a. For example *(your terminal display may vary but the outputs should be consistent)*

```
(base) ➜  cdte-tools-main pwd
/Users/kris/Documents/umnPostdoc/projects/both/foxsi4/foxsi4-analysis/cdte-tools-main
(base) ➜  cdte-tools-main git clone https://github.com/KriSun95/cdte-tools.git
Cloning into 'cdte-tools'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
(base) ➜  cdte-tools-main
```

4. My computer now has my Forked code in the `cdte-tools-main` folder
    a. You can now change directory into the `cdte-tools` folder
    b. As we saw earlier, this folder at the minute is pretty empty and only has the LICENSE and README.md file in it
        i. This will change now as I will add some files and folders we need
        ii. This should be a good example of how you incorporate your changes into the package
5. Making any change in the `cdte-tools` folder will be tracked by `git`
6. Similar to the idea that we should work on our own Fork of the code rather than directly with the main FOXSI code, we also do not want to make changes directly to the only version of the code you have on you machine
    a. To avoid this, we make use of Branches
        i. Branches let you have multiple versions of the code that can be edited separately without having to change the original code
        ii. Then, once the changes being made are deemed to be safe, they can be applied to the other code
7. The default branch your code will be on is called `main` and it is good practice to never directly interact with main on you machine, leaving it untouched
    a. By typing `git branch -a` we see the branches available in this project



    b. We see there is some information here but the important thing is that `main` is there and we know we're on the `main` branch because of the "*" next to the name
8. We will create a new branch to work on and *leave `main` alone*
    a. I will make a new branch, and switch to it, called `add-some-files`
        i. This was done using the command `git checkout -b add-some-files`
        ii. Of course the name `add-some-files` is not very descriptive and is only used as an example, please use a name more appropriate for the feature/fix/etc you are coding
    b. The `git branch -a` command can be used again to make sure the new branch exists and that the "*" has moved



9. The branches can be swapped between using `git checkout <branch-name>` command
    a. E.g., to change back to `main` then `git checkout main`
    b. E.g., to change to the `add-some-files` branch then `git checkout add-some-files`

Working on your your repo on your own machine

10. Let's now add some files while we're in the `add-some-files` branch
    a. I'll add a folder for all the Python code we want to add
        i. This will now be `cdte-tools/cdte-tools-py/`
        ii. If a folder has been added but is empty then `git` will ignore it
    b. Let's add some files to the folder for Python code
    c. Now checking the status of the `git` controlled folder with the `git status` command we see there is a folder that is untracked

```
(base) ➜  cdte-tools git:(add-some-files) ✗ git status
On branch add-some-files
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        cdte-tools-py/

nothing added to commit but untracked files present (use "git add" to track)
```

    d. To track the file we want to add it to the tracked files
        i. Use the command `git add`
        ii. Using the status command again we can see `git` is now tracking the new files that have just been added

```
(base) ➜  cdte-tools git:(add-some-files) ✗ git add cdte-tools-py/
(base) ➜  cdte-tools git:(add-some-files) ✗ git status
On branch add-some-files
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   cdte-tools-py/cdte_tools_py/__init__.py
        new file:   cdte-tools-py/setup.py

(base) ➜  cdte-tools git:(add-some-files) ✗
```

        iii. Adding these files puts them in a staging area and we could continue adding files as we make more changes
    e. Once all the files have been added that we require then we want to "commit" the changes
        i. This tells git that we are finished making this lot of changes and it gives us a chance to write a short description
        ii. This is done using the `git commit -m "Added some initial files."` command where "Added some initial files." is the short description

```
(base) ➜  cdte-tools git:(add-some-files) ✗ git commit -m "Added some initial files."
```

    f. Running git status again shows us nothing indicating everything was added correctly

```
(base) ➜  cdte-tools git:(add-some-files) git status
On branch add-some-files
nothing to commit, working tree clean
(base) ➜  cdte-tools git:(add-some-files)
```

    g. This *adding* and *committing* process can be performed as many times as you like before moving on
        i. E.g., you could *add*, *commit* (with description) a few files at a time

Putting your local changes back on your Github

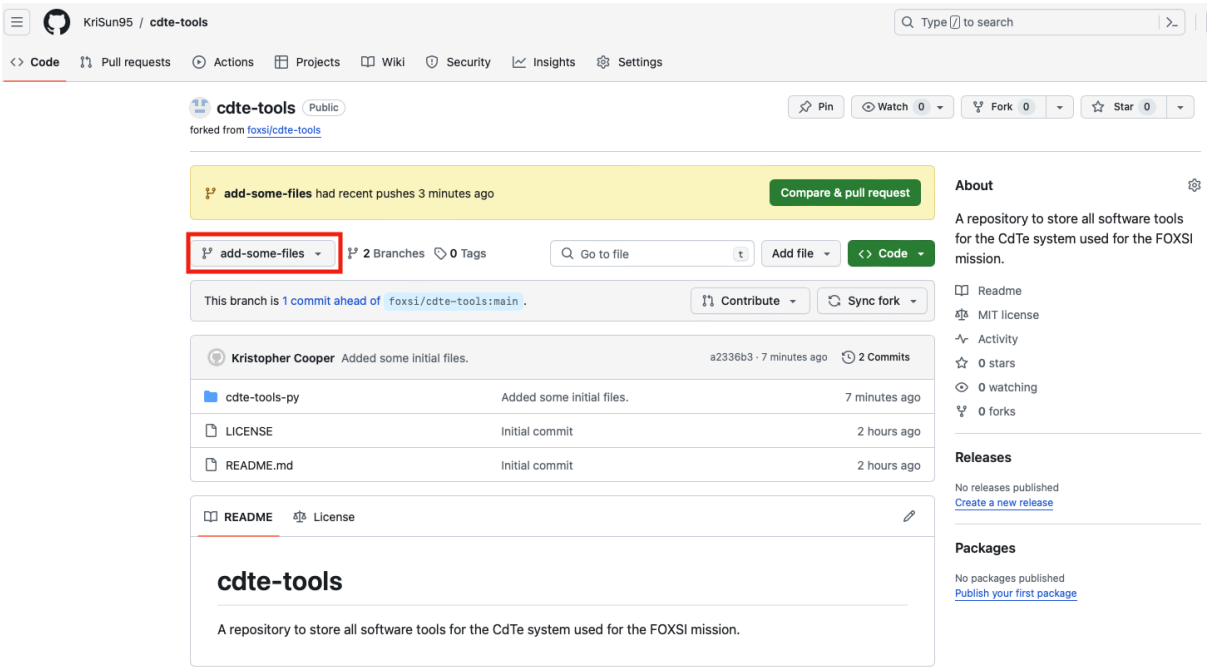
We have now made changes locally and made sure `git` is aware of them, now we want to make sure those changes are backed up on your Github account.

Making sure all changes are committed

1. We can push the commits to your Github repo

```
(base) ➜  cdte-tools git:(add-some-files) git push origin add-some-files
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 10 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 700 bytes | 700.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'add-some-files' on GitHub by visiting:
remote:      https://github.com/KriSun95/cdte-tools/pull/new/add-some-files
remote:
To https://github.com/KriSun95/cdte-tools.git
 * [new branch]      add-some-files -> add-some-files
(base) ➜  cdte-tools git:(add-some-files)
```

2. Going back onto my Github repo page again, we see the new files are now here

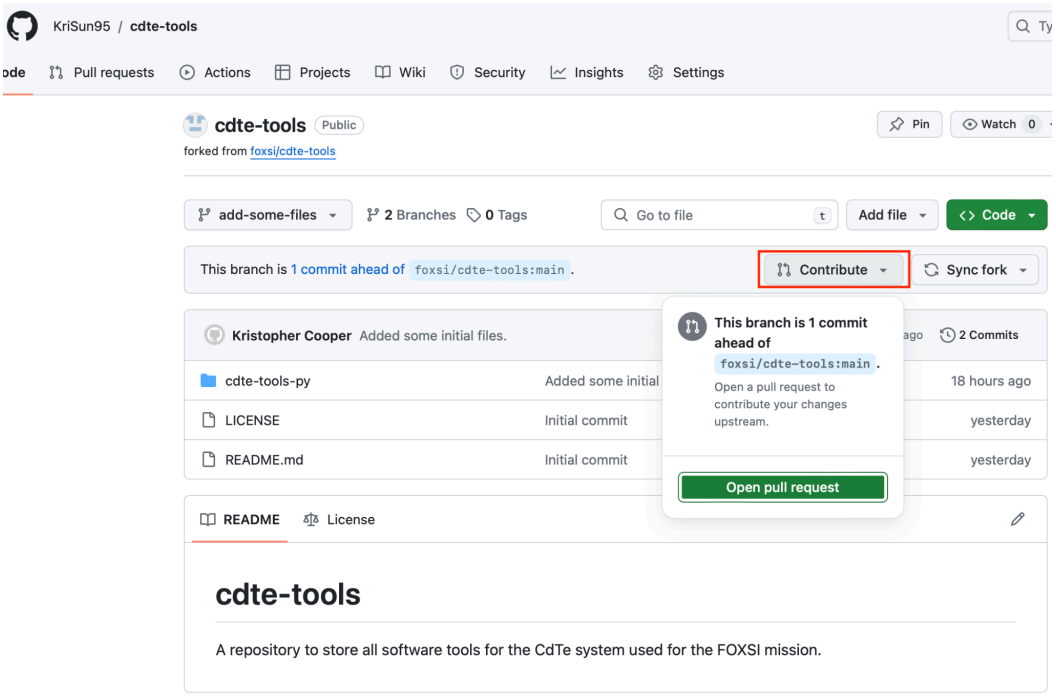


   a. Note that I have changed the branch being looked at on my repo indicated by the red box
   b. Just below the red box, we see that this branch is “1 commit ahead” of the main `cdte-tools` package in the FOXSI account
      i. This commit contains the newly tracked files we have just uploaded
   c. After this point, we could go back and make more local changes on your machine (go back to #heading=h.169539ld602c and follow to here again)
      i. This would increase the number of commits your branch is ahead of the FOXSI account

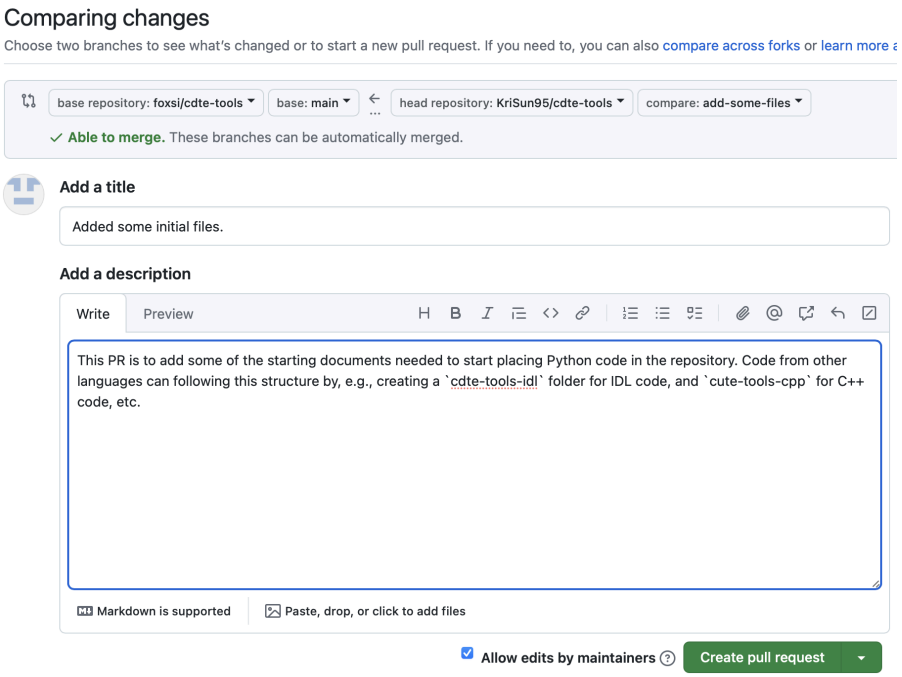
Updating the main FOXSI code

Now that all of your changes have been backed up to your Github account and you have added in the code/features you would like, it is time to consider applying these changes to the `foxsi:main`.

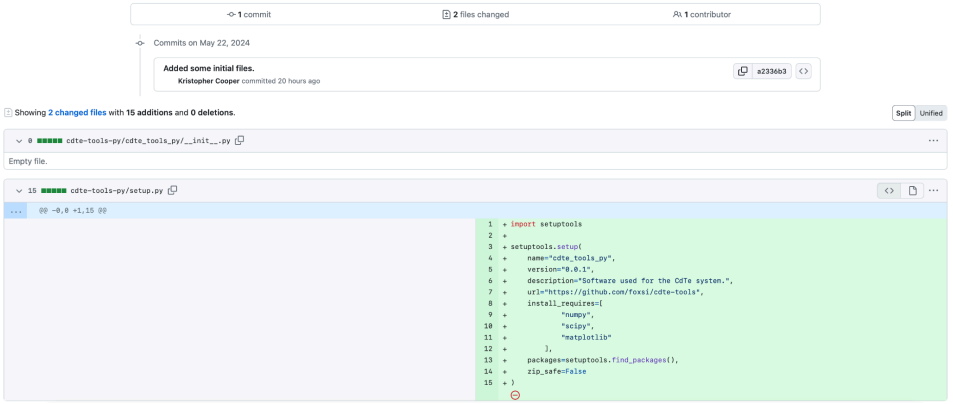
1. Contributing to FOXSI by clicking on the contribute button on your repository's branch page



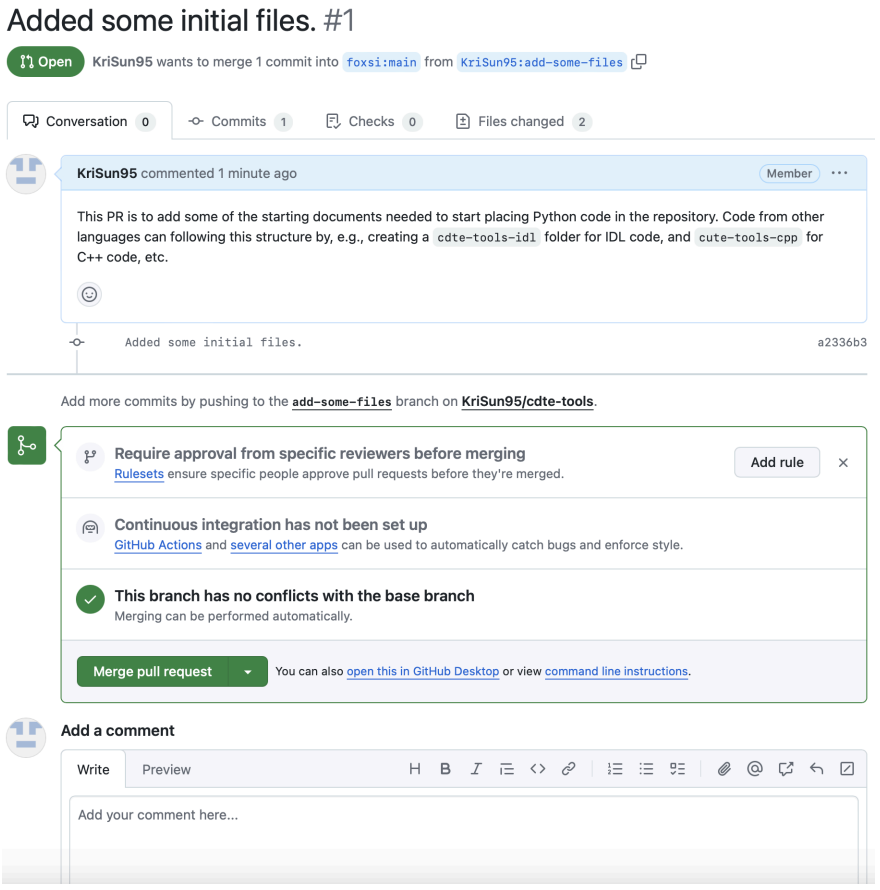
2. Click "Open pull request"
    a. A Pull Request is often just referred to as a *PR*
    b. This will take you to the following page



i. Here I can see the changes I have made in my Fork's branch of `cdte-tools` called `add-some-files` are being proposed to be included in the `main` branch of `foxsi/cdte-tools`
ii. I am able to leave a description of what I am adding
iii. additionally, if I scroll down the webpage, I can also easily see the changes that have been made to the code (new code in green, old and replaced code in red)

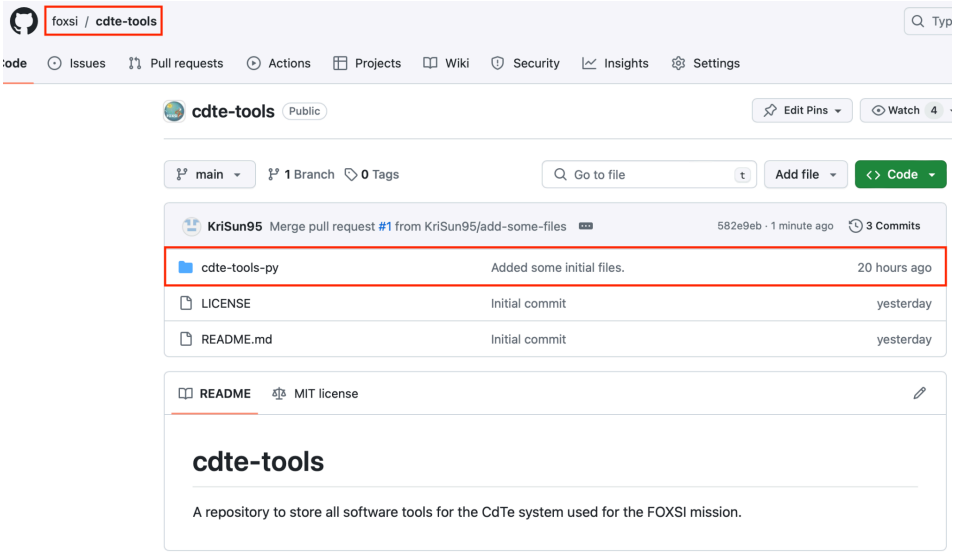


3. Once you are happy with the PR, go ahead and click "Create pull request"
4. You will then be taken to the following page where you and others can look through your changes and chat about it if need

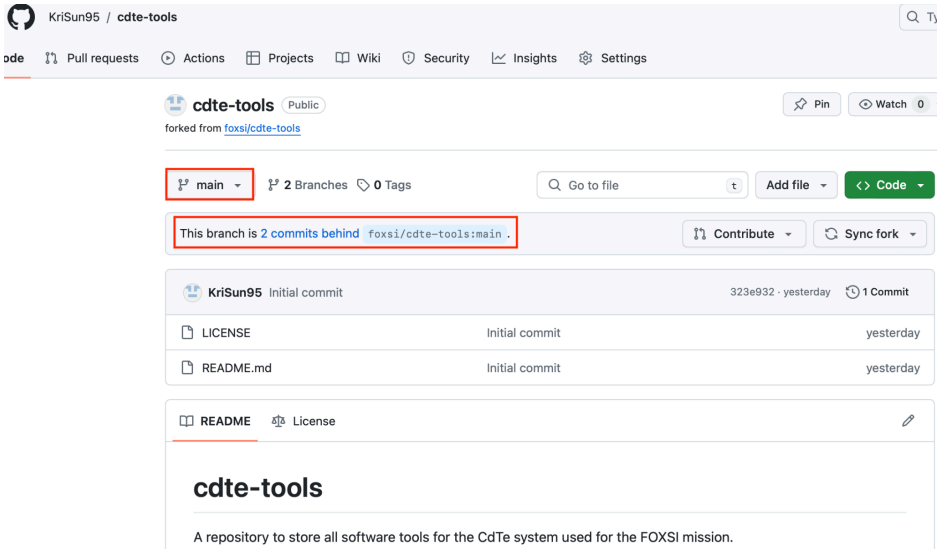


a. The PR page is also a great place if you want to ask any questions about your code

5. For this example I will just go and merge the PR by clicking on “Merge pull request”
    a. I get another option to add an additional description before I click “Confirm merge”
6. Once the changes have been made we can see that the code I have added is now available on the main FOXSI repository



7. **IMPORTANT**: now the `main` branch on the FOXSI page is updated to the latest version of the code, we now need to look at our `main` branches that we have ignored so far
8. Going back to your Github page for the `cdte-tools` repository, we can see it is now behind the code available in the `main` FOXSI repository

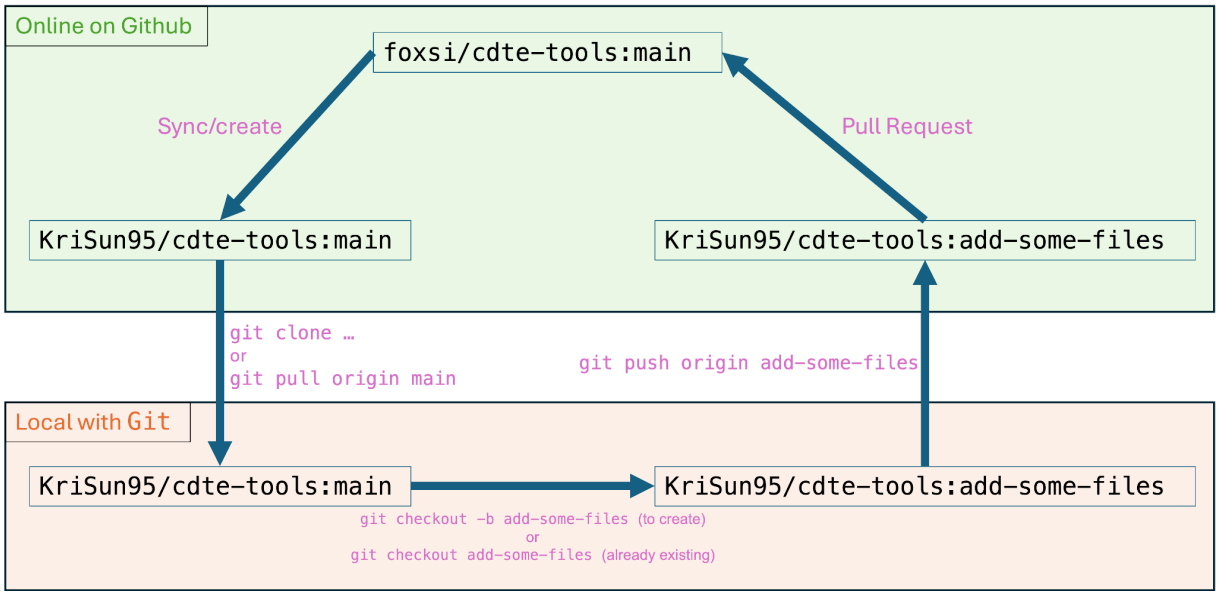


9. We simply need to click the “Sync fork” button and confirm to update our `main` branch to match that of the FOXSI `main` branch
    a. If we want to keep the `add-some-files` branch around for longer (maybe we want to use it for further additions instead of deleting it) then we can also change to that branch and click the “Sync fork” button and confirm
    b. Now your online Github page will be up-to-date with the FOXSI repository

10. We now need to make sure that everything on you own machine is up-to-date with the changes so let's go back to the terminal
    a. make sure we are on the `main` branch with `git checkout main`
11. To pull down the updates now on your own Github page we type
    a. Command: `git pull origin main`
    b. This pulls the changes from `origin` (your online code) to you local `main` branch

```
(base) ➜  cdte-tools git:(main) git pull origin main
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 910 bytes | 455.00 KiB/s, done.
From https://github.com/KriSun95/cdte-tools
 * branch            main       -> FETCH_HEAD
   323e932..582e9eb  main       -> origin/main
Updating 323e932..582e9eb
Fast-forward
 cdte-tools-py/cdte_tools_py/__init__.py |  0
 cdte-tools-py/setup.py                  | 15 +++++++++++++++
 2 files changed, 15 insertions(+)
 create mode 100644 cdte-tools-py/cdte_tools_py/__init__.py
 create mode 100644 cdte-tools-py/setup.py
(base) ➜  cdte-tools git:(main)
```

    c. If we are keeping the `add-some-files` branch around for future work then we would also move to that branch with `git checkout add-some-files` the type `git pull origin add-some-files` to update the local code
12. The crucial nature of the last few steps is to make sure any of you main branches (online or local) should be the same as the latest version on the FOXSI repository
    a. Therefore, all work is done on another branch, merged into FOXSI, then your main branches get updated
    b. This helps avoid conflicts in the code development process
    c. The cycle we have performed for creating the repositories is like the following



    i. This might look complicated but once you do it a few times it will seem much simpler
    ii. Notice the your `main` branches never go towards FOXSI `main`

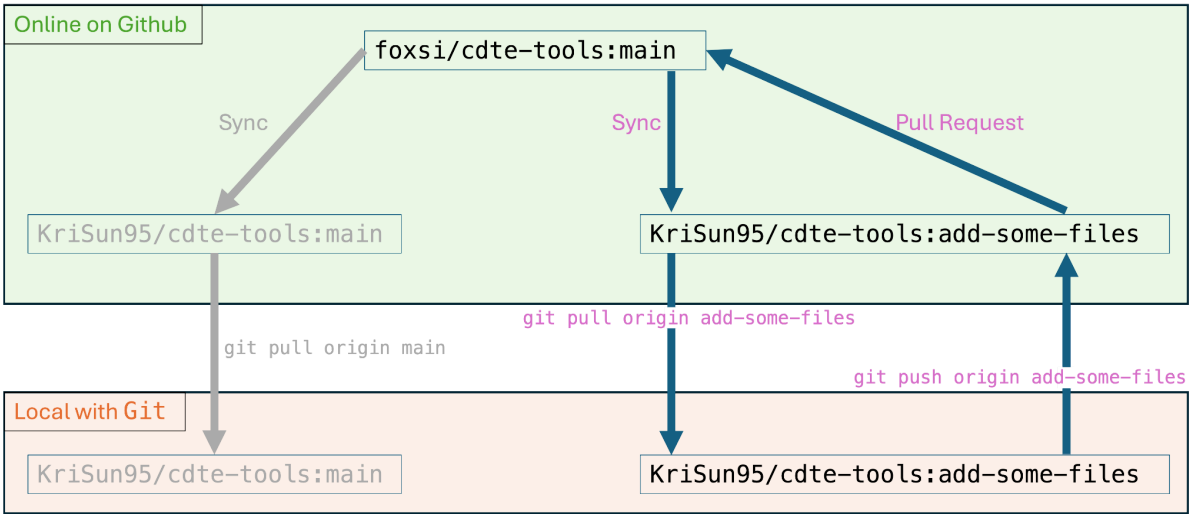
When things are set-up and other useful tips

Remember, do not spend loads of time on the Git/Github side of things. Please contact someone to help (Kris, Thanasi, etc.). We will be more than happy to help.

Once you have been able to create your own Fork of the code you want and can make branches then all of these concepts will become easier.

Re-using a branch after a PR

Another cycle you might find yourself in is:



This would be similar to what you might want to do if you wanted to reuse a branch you previously opened a PR with. Remember to keep your `main` branches up to date.

Before and after a work session

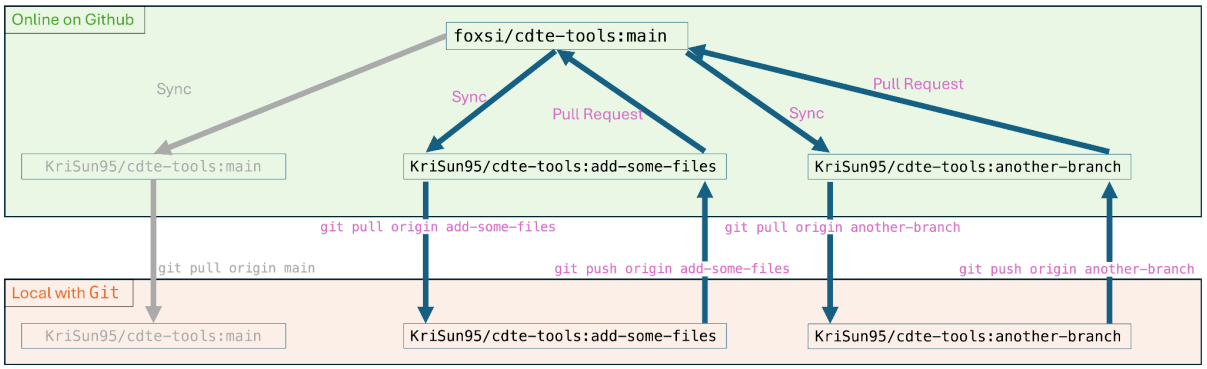
It is good practice to always change back to your local `main` branch when you're done for the day (using `git checkout main`) after *adding* and *commiting* all the changes you have made on the branch you are working on. Then at the start of the next session, make sure your online account is synced up to the main FOXSI code then perform `git pull` on all of your local machine branches to make sure you're working on the most recent code that everyone else sees.

Adding files with `git add`

When it comes to *adding* files using `git add <file-names>`, try to avoid using `git add .` or `git add -A` which will add ALL files that have changed. Sometimes this is what you want (which is fine) but it is easy to forget about a changed file somewhere you did not want to alter. If in doubt, typing `git status` will show you what files have changed and what files have and have not been *added*.

Working with multiple branches

A fantastic thing about using `git` to control your code workflow is that you can also have multiple branches being worked on at the same time. The image below shows another branch being worked on.



This means you could work on adding a new feature in one branch while fixing a bug in another and the two branches won't interfere with each other. When one of the branches undergoes a successful PR then all that is needed is to "Sync" then `git pull` each of the branches to make sure they are all back up-to-date and consistent with each other.

Github Issues

You may have noticed that along the top banner of the FOXSI Github repository page (as well as in other places) there is a tab for "Issues". The Issues are there to keep track of known bugs, features we might want to add, and other broader conversations. Issues can be made by anyone and are highly encouraged.

Glossary

A rough guide.

- Fork: A copy of a central Github repository
- Branch: A copy of a repository that someone is working on.
    - You might make several branches from your Fork to work on different features
- Commit: A logged change in the branch someone is working on
- Pull Request: A proposal to merge the code from one branch into another