Contributing to a Github repository

This guide is based on the guide for OGGM rules of contribution. But it can be apply to any project as I have done below for the C3S work.

Before you contribute, you will need to learn how to work with GitHub: Version control, Git, and GitHub

For example:

The code is hosted on Github under this link. To contribute you will need to sign up for a free GitHub account.

We use Git for version control to allow many people to work together on the project.

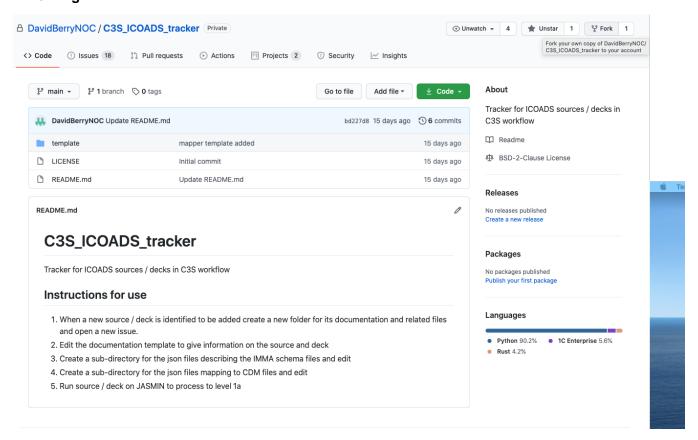
Some great resources for learning Git:

- · the GitHub help pages.
- the NumPy's documentation.
- Matthew Brett's Pydagogue.
- Getting started with Git

GitHub has instructions for installing git, setting up your SSH key, and configuring git. All these steps need to be completed before you can work seamlessly between your local repository and GitHub.

Main steps

1. Forking



You will need your own fork to work on the code. Go to the **Repository project** and hit the **Fork** button. You will want to clone your fork to your machine:

git clone git@github.com:your-repository-url cd your-project-copy git remote add upstream git://github.com/Repository_main_url

For example for the C3s repository and my user name:

```
git clone git@github.com:bearecinos/C3S_ICOADS_tracker.git cd C3S_ICOADS_tracker
git remote add upstream git://github.com/DavidBerryNOC/C3S_ICOADS_tracker.git
```

This creates the directory /your_user_name/C3S_ICOADS_tracker and connects your repository to the upstream (main project) /main_user_nam e/C3S_ICOADS_tracker. In the case of the C3s work David is the owner of the main repository.

2. Creating a branch (optional)

This is ideal for large projects but you could skip this step for the C3S work

You want your main branch to reflect only production-ready code, so create a feature branch for making your changes.

For example:

```
git branch shiny-new-feature
git checkout shiny-new-feature
```

The above can be simplified to:

```
git checkout -b shiny-new-feature
```

This changes your working directory to the shiny-new-feature branch. Try to keep any changes in this branch specific to one bug or feature. You can have many shiny-new-features and switch in between them using

```
git checkout branch-name
```

To update this branch, you need to retrieve the changes from the **main branch**:

```
git fetch upstream
git rebase upstream/main
```

This will replay your commits on top of the latest C3S_ICOADS_tracker git main.

If this leads to merge conflicts, you must resolve these before submitting your pull request. If you have uncommitted changes, you will need to **stash** them prior to updating. This will effectively store your changes and they can be reapplied after updating.

3. Contributing your changes

After your are done with your changes to your local pc repository. You need to commit your changes.

Keep style fixes to a separate commit to make your pull request more readable. Once you've made changes, you can see them by typing in your directory:

```
git status
```

If you have created a new file, it is not being tracked by git. Add it by typing:

```
git add path/to/file-to-be-added.py
```

Doing git status again should give something like:

```
# On branch shiny-new-feature
#
# modified: /relative/path/to/file-you-added.py
#
```

Or if you are in the main branch

```
# On branch main
#
# modified: /relative/path/to/file-you-added.py
#
```

Finally, commit your changes to your local repository with an explanatory message:

```
git commit -a -m 'added shiny feature'
```

If you do git status again you will see a list of files that you will be committing to the main branch.

You can make as many commits as you want before submitting your changes to the main repository, but it is a good idea to keep your commits organised. Also in the -m (message) of your commit you can mention issues by number, so this issues can be close automatically after you commit changes that are associated to such issues. For example

```
git commit -a -m "added docs, schema, cdm for deck704 Issue#8"
```

4. Pushing your changes

When you want your changes to appear publicly on your GitHub page, push your forked feature branch's commits:

```
git push origin shiny-new-feature
```

If you want to push to the main branch and you skipped step 2 you need to do:

```
git push origin main
```

Here **origin** is the default name given to your remote repository on GitHub.

You can see the remote repositories by doing:

```
git remote -v
```

If you added the upstream repository as described above you will see something like:

```
origin git@github.com:bearecinos/C3S_ICOADS_tracker.git (fetch) origin git@github.com:bearecinos/C3S_ICOADS_tracker.git (push)

upstream git://github.com/DavidBerryNOC/C3S_ICOADS_tracker.git (fetch) upstream git://github.com/DavidBerryNOC/C3S_ICOADS_tracker.git (push)
```

Now your code is on GitHub, but it is not yet a part of the main project repository C3S_ICOADS_tracker.git .

For that to happen, a pull request needs to be submitted on GitHub

5. Review your changes

When you're ready to ask for a code review, file a pull request. Before you do, once again make sure that you have followed all the guidelines of the repository. You should also double check your branch changes against the branch it was based on:

```
#. Navigate to your repository on GitHub -- https://github.com/your-user-name/oggm
#. Click on ``Branches``
#. Click on the ``Compare`` button for your feature branch
#. Select the ``base`` and ``compare`` branches, if necessary. This will be ``master`` and
``shiny-new-feature``, respectively.
```

6. Finally, make the pull request

If everything looks good, you are ready to make a pull request. A pull request is how code from a local repository becomes available to the GitHub community and can be looked at and eventually merged into the master version. This pull request and its associated changes will eventually be committed to the master branch and available in the next release.

To submit a pull request:

```
#. Navigate to your repository on GitHub
#. Click on the ``Pull Request`` button
#. You can then click on ``Commits`` and ``Files Changed`` to make sure everything looks okay one last time
#. Write a description of your changes in the ``Preview Discussion`` tab
#. Click ``Send Pull Request``.
```

This request then goes to the repository maintainers, and they will review the code. If you need to make more changes, you can make them in your branch, push them to GitHub, and the pull request will be automatically updated. Pushing them to GitHub again is done by:

```
git push -f origin shiny-new-feature
```

Or if you did not created have a branch

```
git push -f origin main
```

This will automatically update your pull request with the latest version.

7. In case you created a branch for your feature. You can delete it

Once your feature branch is accepted into upstream, you'll probably want to get rid of the branch. First, merge upstream master into your branch so git knows it is safe to delete your branch:

```
git fetch upstream
git checkout main
git merge upstream/main
```

Then you can just do:

```
git branch -d shiny-new-feature
```

Make sure you use a lower-case -d, or else git won't warn you if your feature branch has not actually been merged. The branch will still exist on GitHub, so to delete it there do:

```
git push origin --delete shiny-new-feature
```

Note that the workflow above is the ideal way of working. If you find making branches too complicated you can stick to the following summary.

Be aware that some repositories have rules for contributions and most of them follow the workflow above.

- 1. Fork repository
- 2. Clone your fork repository:

git clone git@github.com:your-repository-url cd your-project-copy

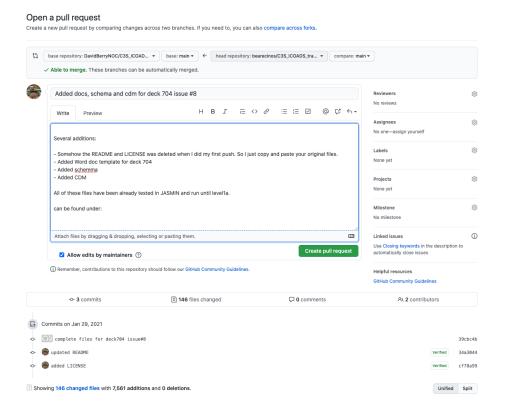
- 3. Make your changes
- 4. Check the status of the files changed git status
- 5. Add the files to a commit

git add /relative/path/to/file-you-added.py

Or alternative to add all the files

- git add .
 6. Commit changes and add a message
 - git commit -m "added docs, schema, cdm for deck704 Issue#8"
- You can combine steps 5 and 6 in a single command by using git commit -a (add all files) and -m (message): git commit -a -m "added docs, schema, cdm for deck704 Issue#8"
- 8. Push changes to your Fork repository
 - git push origin main
- 9. Make a pull Request, following 6.

The screen looks like this, and you can expand the message and make sure you tag the issues with the # symbol! You can also mentioned users with @



Once you succeed it will look like this:

