Development

The aim of this section will be to explain how to contribute to the development of the DRM4G.

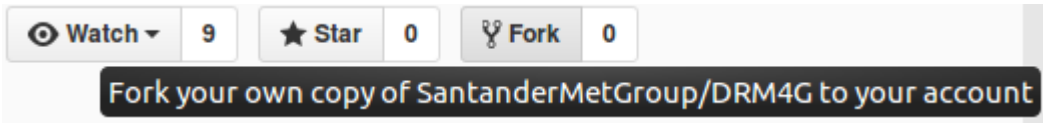
To make it easier and more accessible for anyone to play around with the DRM4G, we have decided to host our source code on **GitHub**.

- You can find the project ?here.

Necessary Steps

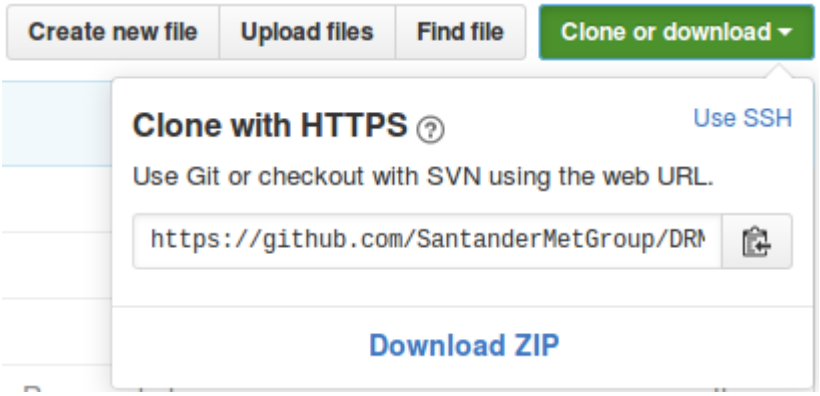
For those of you who wish to help but don't know how, the first thing you need is a ?GitHub account.

In our project's page hit the "***Fork***" button at the top right corner:



This will create a copy of our repository in your account where you'll develop your own feature or implement a bugfix that you may believe is necessary. You'll be submitting changes to this one until you are certain everything works properly, at which point you can request to have your changes integrated into the DRM4G's repository.

To continue, you'll need to setup a local repository where you'll be changing the code and doing your testing. To do this you'll need your repository URL, that can be obtained adding **.git** to your project page or by clicking on the "***Clone or download***" button:



In Linux operating systems:

- Open a terminal on the folder in which you wish your local copy of the repository to be stored (it's recommended to use an empty folder) and run the following commands:

```
git init  #to initialize an empty Git repository
git remote add origin <your_repository_url> #to make your local repository point to your remote repository in GitHub
git fetch
git checkout develop #to create a local copy of the develop branch
```

To be consistent with our gitflow, all you'll be able to do is create **feature or bugfix branches**, and you'll have to follow our naming conventions to do so.

  Our GitHub's default branch is "**develop**", and this is done intentionally, since the "**master**" will only be updated when publishing a new release
  - This means that all pull requests to the "**master**" branch will be rejected
- The naming convention will just be to create branches in lower case letters separated by underscores ("_") that describe what you're trying to accomplish with the branch.

With this, you will now have DRM4G's source code at your disposal.

From here you could create branches for every new feature you'd like to include to the DRM4G, for a more in depth tutorial on how to do that, click ?here.

Testing the DRM4G

Once you've made the changes you wanted to, you'll want to install your version to be sure that your new feature is working properly.

Just in case you would like to try out different versions, we recommend you use a virtual environment to test it.

- ?Here you can find a tutorial on how to install a virtual environment, or you can look for one on your own.

Before you can install and try out your own version, you'll have to build your own package:

- Open a terminal in the folder where your repository is located.
- Run the command `python setup.py sdist`

This will create a a distribution package under a folder called **dist**.

Installing your version in a virtual environment

Go to wherever you have your virtual environment, open a terminal and execute the following commands:

```
source bin/activate
export DRM4G_DIR = $PWD/conf
pip install path/to/drm4g/package
```

DRM4G_DIR is where the configuration files will be installed. More information here.

And that's it. Now you can use and test your own version of DRM4G.

For other ways to install the DRM4G, you can check here.

Commiting changes

After you've tested that everything is in working order, it's time to update your GitHub fork.

```
git add .
git commit -m "Description of the changes you've made"
git push origin develop
```

From here you'll have to create a **Pull request**.

Creating a Pull Request

As time passes, chances are that the main DRM4G repository has had other contributors merge changes into it, so you should first update your local repository and check that there are no conflicts.

```
git checkout develop
git remote add github https://github.com/SantanderMetGroup/DRM4G.git #just do run this command the first time to add the m
git pull github develop
```

If there have been any changes, but there are no conflicts, you can just *push* your changes into your remote repository.

```
git push origin develop
```

In the case that there are conflicts, resolve them and commit the changes, after do the *push* to update your remote repository.

After, you just have to go to your repositories web page and click on the "***New pull request***" button.

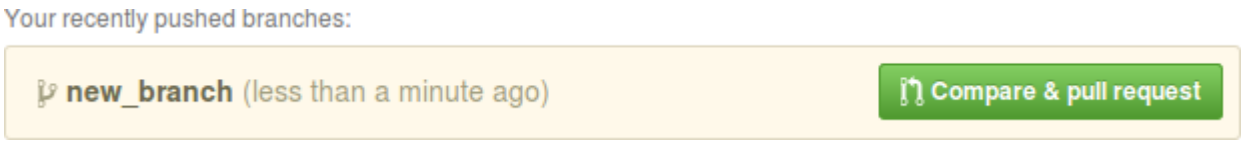
A message will tell you if there are any conflicts that need to be solved or if an automatic merge is possible. Then you just have to click on the "***Create pull request***", give this merge a title (which will serve as the commit message if the pull request is accepted), a comment if you want (can be useful to explain in more detail why this should be integrated into the DRM4G), and click a second time on the "***Create pull request***" button.

Now you'll have to wait and see if the administrator of the project accepts the changes you're proposing.

Another option would be to create a new branch from the "**develop**" branch and then try to make a pull request from that.

```
git checkout develop
git checkout -b new_branch
#make some changes
git add .
git commit -m "Description of the changes you've made"
git push origin new_branch
```

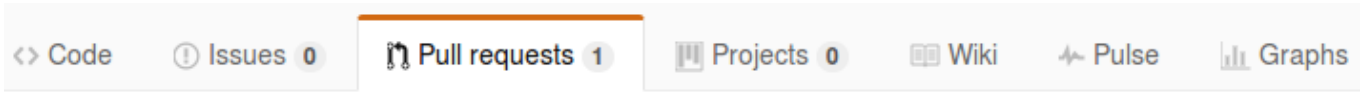
The next time you enter your repository's web page, you'll see a new button:



Click on "***Compare & pull request***" to perform a pull request on to the main repository. The same will happen as when clicking on "New pull request" when on a previously existing branch.

For the administrator of the DRM4G's GitHub repository:

When you see that there's a new pull request, either because you saw it on the projects page or because you received a message in your mail informing you about it, you can click on the "***Pull requests***" button.



If you agree with the changes made, and there are no conflicts, you can just click on the "***Merge pull request***" button.

- You'll automatically see a commit message that will look something like this: "Merge pull request #X from <contributor>/<branch_name>"
- Bellow that, you can write an "optional extended description".
  - When running `git log` from the command line, you'll just see both lines shown as the commit message.
- When done just click on the "***Confirm merge***" button to finish.

If don't agree or there are conflicts, you can click right next to the "Merge pull request" button, where it says "***command line instructions***" to perform a manual merge.

- Just follow the instructions shown there.
- Optionally, you could also try to solve the conflicts wiht the web editor, but this isn't recommended since you can't test what you're changing, unless the conflict is in a comment or a text message, not in the code.