How to Git

April 16, 2019

1 How to get started

1. Make sure you have working command-line versions of `git` and `conda`
2. Go to `https://github.com/BCCN-Prog/2019_elegANT` if you aren't there yet
3. Click on 'Fork'
4. In your own copy of the repository click on 'Clone or download'
5. Make sure that you are cloning using SSH
6. Copy the link presented in the pop-up window
7. In the terminal navigate to the folder where the project should be created
8. Type `git clone` and insert the url you copied (and execute the command).
9. Switch into the directory that was created!
10. Run the following commands
    (a) `git remote add upstream git@github.com:BCCN-Prog/2019_elegANT.git`
    (b) `git config branch.master.remote upstream`
    (c) `git config branch.master.merge refs/heads/master`
    (d) `git fetch upstream`
    (e) `git branch --set-upstream-to=upstream/master master`
11. Run `while read requirement; do conda install --yes $requirement; done < requirements.txt`
12. Run `conda install pytest flake8` and make sure both commands work
13. Do not start coding yet! - view the next section

2 How to get started on a new feature

Identify who you are going to work with for the feature you chose (or were assigned). (One such group will be referred to as a 'workgroup')

1. Amongst yourselves designate a 'Lieutenant' (you will collaborate in his/her public repository and she/he will make the pull request in the end)
2. This Lieutenant should do the following
   (a) Pick a name for the feature
   (b) `git checkout -b <branch name>`
   (c) `git config branch.<branch name>.remote origin`
   (d) `git config branch.<branch name>.merge refs/heads/<branch name>`
   (e) `git push`
   (f) Restrict access to master branch (see section 6)
   (g) Invite all members of the workgroup as collaborators (see section 7)
3. Now, the non-Lieutenant collaborators should
   (a) Accept the invitation they recieved via e-mail
   (b) `git remote add lieutenant git@github.com:<lieutenant username>/2019_elegANT.git`
   (c) `git fetch lieutenant`
   (d) `git checkout -b <branch name> lieutenant/<branch name>`
4. Now you can start coding, but please follow the steps of the following section

3 How to collaborate in this setup

Even though responsibilities within a workgroup are not defined by the repository structure, always make sure you don't create alternative (incompatible) versions of the same piece of code - otherwise you will have merge conflicts. This is not the end of the words, but avoid it if you can.

To further minimize the probability of hard to resolve merges make sure to commit and push frequently. Nevertheless do not push broken code (i.e. a program version that fails where it shouldn't) to a shared repository/branch.

1. Before you start coding on a new feature run `git pull` while you are in the feature branch
2. Now you can code!
3. Once you think you made good progress
   (a) See if your code passes all tests by running `pytest` and then `flake8`

(b) If errors exist, fix them before going further and return to (a)

(c) Add tests for all functions, classes and other functionalities you wrote.

(d) Identify which file you changed meaningfully (`<file1> ... <fileN>`)

(e) `git add <file1> ... <fileN>`

(f) `git commit`

(g) Always give a description of what you achieved in this commit in the commit message

(h) `git pull`

(i) Run `while read requirement; do conda install --yes $requirement; done < requirements.txt` again

(j) Should git inform you that a merge conflict occurred - resolve it (see section **??**)

(k) `git push`

4. Start again at step 2 until the feature is completed.

4 How to close a feature branch

Once you all agree that this feature is completed the lieutenant should follow the steps below - the rest of the group can do something enjoyable (after step 1 and once the lieutenant explicitly agrees that the feature is ready)

1. If you haven't already, extend the already existing test by extensive tests for your feature.
2. In the feature branch: `git pull`
3. Go to your Travis-CI (see section 8) account and see if any recent pushes failed and if so, get the teammate to fix the problem.
4. `git checkout master`
5. `git pull`
6. Run `while read requirement; do conda install --yes $requirement; done < requirements.txt` again
7. `git checkout <branch name>`
8. `git merge master`
9. Resolve any merge conflict (with the help of your team)
10. Add, commit and push.
11. Go to `https://github.com/BCCN-Prog/2019_elegANT` again. Your fork and the time of last change should be shown above the files of the project.

12. Click the ‘Compare & pull request button’
13. Make sure the ‘Able to merge’ comment appears
14. Describe in how far you completed the feature
15. Click on ‘Create pull request’
16. Restrict the feature branch like you did with your master branch
17. Wait until an integration master comments on it, accepts it or rejects it.
18. In case of rejection or a comment requesting changes
    (a) Remove the restriction on the feature branch
    (b) Get your teammates back
    (c) To fix the problem start again at step 1 in section 3.
    (d) If your pull request was not rejected then you do not need to make another pull request - simply push your changes onto the remote feature branch

5 How to resolve a merge conflict

- When git detects changes that are not compatible (mostly when the same line was changed) it nevertheless creates a merged file version on your local branch
- However this file now contains some additions automatically created by git. A file with these looks like

```
<<<<<<< HEAD
Version 1 of the code that causes a conflict
=======
Version 2 of the code that causes a conflict
>>>>>>> [<the branch your are trying to merge with>]
```

- Now it is up to you and possibly the other author of the conflicting code to manually edit the file so that the problematic versions are merged.
- This can mean that
  - ... you simply delete one version and the lines added by git.
  - ... keep both versions sequentially (for example when the same class was created by different people but they added wrote different methods with different purpose)
  - ... really create ‘merged’ version with some lines from both versions
  - ... ...

- Once you have handled every part that was marked by git, check is all the test still complete successfully
- Now you should be able to add, commit and push your changes without git returning unresolved-conflict-error.

6 How to restrict a branch

1. Go to your fork of the project on GitHub
2. Go to 'Settings'
3. Choose category 'branches'
4. Under 'Branch protection rules' choose 'Add rule'
5. Enter the name of the branch you want to restrict
6. Check 'Require pull request reviews before merging' and 'Require status checks before merging'
7. Click 'Create'

7 How to add a collaborator

1. Go to your fork of the project on GitHub
2. Go to 'Settings'
3. Choose category 'Collaborators'
4. Enter the GitHub username of the person you want to add

8 How to use Travis-CI

1. Go to `https://travis-ci.org/`
2. On the top-right click 'Sign in with GitHub' if you haven't already and log in with your GitHub account
3. Again at the right-top click your GitHub profile picture
4. Locate the 2019_elegANT repository in the list and click it
5. Click the 'Activate repository' button
6. Now Travis-CI can show the result of automatically triggered builds
7. If you want to manually start the testing go to the 'Dashboard'-tab (on the top left)

8. On the entry for the 2019_elegANT repository click the 'Trigger build' or if that button is not present then press the three lines and it should appear
9. Now wait for a long time until the result appears
10. For more details click the name of the repository
11. If the master branch does not pass the test run the following commands in your local fork of the project:
    (a) `git checkout master`
    (b) `git pull`
    (c) `git push origin master`

9 Some tips for the Ubuntu-Subsystem of windows

- In the command prompt you can access all the files on your Windows systems, but you nee to run the following command
  1. `cd ../../mnt/c/Windows/Users/<your windows username>/Desktop`
  2. If you now run `ls` you should see all the files that are actually on your Windows desktop
- Copy and pasting both works by right-clicking into the command prompt. If you right-click onto a marked text-section this section will be copied into your windows clipboard and if you right-click into the vicinity of the line where you would enter a new command your windows clipboard is inserted there.

10 What your git-config file should look like

You can check whether yout git is set up correctly by running `cat .git/config`. If you are the Lieutenant of the feature 'test_feature' and a non-Lieutenant collaborator on 'test_feature_2' (where Dorothy is the Lieutenant) then your config file should look as follows:

```
        filemode = true
        bare = false
        logallrefupdates = true
[remote "origin"]
        url = git@github.com:REZ-K/2019_elegANT.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
        remote = upstream
        merge = refs/heads/master
```

```
[remote "upstream"]
        url = git@github.com:BCCN-Prog/2019_elegANT.git
        fetch = +refs/heads/*:refs/remotes/upstream/*
[branch "test_feature"]
        remote = origin
        merge = refs/heads/test_feature
[remote "lieutenant"]
        url = git@github.com:DorotheaMueller/2019_elegANT.git
        fetch = +refs/heads/*:refs/remotes/lieutenant/*
[branch "test_feature_2"]
        remote = lieutenant
        merge = refs/heads/test_feature_2
```