Gestion de versions collaborative avec Git et GitHub

Auteur: Ludovic Blanc, Gymnase du Bugnon, 3MOCINFO
Date: Mai 2018

Gestion de versions avec Git

Sources et références:

- https://fr.wikipedia.org/wiki/Git (https://fr.wikipedia.org/wiki/Git)
- https://cours-web.ch/git/ (https://cours-web.ch/git/)
- https://gist.github.com/aquelito/8596717 (https://gist.github.com/aquelito/8596717)

Git est un **logiciel de gestion de versions décentralisé**. C'est un logiciel **libre** créé en **2005** par Linus Torvalds, auteur du noyau Linux. En 2016, il s'agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes. C'est un outil en **ligne de commande**. Pour du meilleur confort d'utilisation, il existe plusieurs logiciel et services en ligne offrant une interface graphique.

Il est codé en **c, Shell Unix, Perl, Tcl et GNU Bash**.

Il utilise un système de connexion **pair à pair**. C'est-à-dire que le code informatique développé est stocké non seulement sur l'ordinateur de chaque contributeur du projet, mais il peut également l'être sur un serveur dédié.

Voici quelques-uns des avantages que Git apporte:

- Permet à un grand nombre de collaborateurs de travailler sur un même projet, en offrant une bonne gestion des modifications.
- Permet de synchroniser un projet entre plusieurs collaborateurs, en ayant l'assurance que tous les fichiers soient à jour.
- Permet d'avoir un historique précis de tous les changements et modifications d'un projet. Cela permet de clarifier les questions récurrentes: *"Où est la dernière version du fichier X?"* et *"Qu'est-ce qui a été changé entre les révisions 41 et 42"?*.

Annecdote selon magazine PC World: « quand on lui a demandé pourquoi il avait appelé son logiciel "git", qui est à peu près l'équivalent de "connard" en argot britannique, Linus Torvalds a répondu "je ne suis qu'un sale égocentrique, donc j'appelle tous mes projets d'après ma propre personne. D'abord Linux, puis Git."»

Commandes

Git dispose notamment des commandes suivantes:

- `git status`
- `git init`
- `git clone`
- `git add`
- `git commit`
- `git push`
- `git pull`
- `git branch`
- `git checkout`

Je vais expliquer leur fonctionnement de "base". La meilleur documentation reste l'aide en ligne de commande:

```
git help config
git help push
git help pull
git help branch
```

Configuration lors de la première utilisation

Pour savoir qui modifie les fichiers et fait des opérations sur Git, il faut préciser son pseudo et son email.

- Identity Name

`git config --global user.name "yourname"` Il faut mettre son pseudo entre les guillemets.

- Identity Email

`git config --global user.email "your@email.com"` Il faut mettre son email entre les guillemets.

On peut executer la commande suivante pour connaître ses réglages personnels:

In [6]:

```
#!git config --list

# (Je ne l'execute pas pour des raisons de confidentialité.)
```

Exemple d'application

Nous allons montrer dans ce Jupyter notebook, comment utiliser Git. Dabord nous allons créer un nouveau dossier dans lequel nous allons créer un nouveau répositoire Git.

Ce notebook jupyter se trouve (pendant que je travail dessus) dans le dossier **gitdemo**. Vérifions cela avec `pwd` .

In [1]:

```
!pwd
```

```
/Users/ludovicblanc/Desktop/info/gitdemo
```

En ce moment ce dossier ne contient pas de sous-dossiers cachés. Vérifions cela avec `ls -la` .

In [2]:

```
!ls -la
```

```
total 48
drwxr-xr-x@  4 ludovicblanc  staff    128 28 mai 11:26 .
drwxr-xr-x  11 ludovicblanc  staff    352 28 mai 11:18 ..
-rw-r--r--@  1 ludovicblanc  staff   6148 28 mai 11:26 .DS_Sto
re
-rw-r--r--@  1 ludovicblanc  staff  14348 28 mai 11:25 gitdem
o.ipynb
```

Maintenant nous allons créer un nouveau répertoire dans ce dossier. Nous utilisons `git init`

In [3]:

```
!git init
```

```
Initialized empty Git repository in /Users/ludovicblanc/Deskto
p/info/gitdemo/.git/
```

Maintenant nous devrions voir un dossier cachés `.git`

In [4]:

```
!ls -la
```

```
total 48
drwxr-xr-x@  5 ludovicblanc  staff    160 28 mai 11:27 .
drwxr-xr-x  11 ludovicblanc  staff    352 28 mai 11:18 ..
-rw-r--r--@  1 ludovicblanc  staff   6148 28 mai 11:26 .DS_Sto
re
drwxr-xr-x   9 ludovicblanc  staff    288 28 mai 11:27 .git
-rw-r--r--@  1 ludovicblanc  staff  14348 28 mai 11:25 gitdem
o.ipynb
```

Que se trouve dans ce dossier `.git` ? Allons voir avec `ls -la .git`

In [5]:

```
!ls -la .git
```

```
total 24
drwxr-xr-x   9 ludovicblanc  staff  288 28 mai 11:27 .
drwxr-xr-x@  5 ludovicblanc  staff  160 28 mai 11:27 ..
-rw-r--r--   1 ludovicblanc  staff   23 28 mai 11:27 HEAD
-rw-r--r--   1 ludovicblanc  staff  137 28 mai 11:27 config
-rw-r--r--   1 ludovicblanc  staff   73 28 mai 11:27 descripti
on
drwxr-xr-x  12 ludovicblanc  staff  384 28 mai 11:27 hooks
drwxr-xr-x   3 ludovicblanc  staff   96 28 mai 11:27 info
drwxr-xr-x   4 ludovicblanc  staff  128 28 mai 11:27 objects
drwxr-xr-x   4 ludovicblanc  staff  128 28 mai 11:27 refs
```

Ce sont les fichiers de configuration, de description, et de gestion des données. Affichons un de ces fichiers avec `cat .git/config` . Regardons maintenant le status de ce nouveau répertoire git avec la commande `git status`

In [6]:

```
cat .git/config
```

```
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
        ignorecase = true
        precomposeunicode = true
```

In [7]:

```
!git status
```

```
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committe
d)

        .DS_Store
        gitdemo.ipynb

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

Nous avons des fichier non-suivi par Git. Ajoutons le avec `git add *` et vérifions de nouveau le status. Le ficher (ce notebook Jupyter) apparait maintenant en vert.

In [8]:

```
!git add *
```

In [9]:

```
!git status
```

```
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   gitdemo.ipynb

Untracked files:
  (use "git add <file>..." to include in what will be committe
d)

        .DS_Store
```

Ajoutons le ficher avec `git commit -m 'message'`

In [10]:

```
!git commit -m 'initial commit of Jupyter notebook'
```

```
[master (root-commit) f76942f] initial commit of Jupyter noteb
ook
 1 file changed, 533 insertions(+)
 create mode 100644 gitdemo.ipynb
```

In [11]:

```
!git status
```

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committe
d)

        .DS_Store

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

Le notebook Jupyter est constamment sauvegardé. Essayons de créer un nouveau ficher. Ecrivons le calendrier de ce mois dans un fichier `cal.txt` .

In [12]:

```
!cal
```

```
      Mai 2018
Di Lu Ma Me Je Ve Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27    29 30 31
```

In [13]:

```
!cal > cal.txt
!git status
```

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in worki
ng directory)

        modified:   gitdemo.ipynb

Untracked files:
  (use "git add <file>..." to include in what will be committe
d)

        .DS_Store
        .ipynb_checkpoints/
        cal.txt

no changes added to commit (use "git add" and/or "git commit -
a")
```

Maintenant nous pouvons ajouter et intégrer `cal.txt` à Git.

In [14]:

```
!git add cal.txt
```

In [15]:

```
!git commit -m 'added calender text file'
```

```
[master 9361f0d] added calender text file
 1 file changed, 8 insertions(+)
 create mode 100644 cal.txt
```

In [16]:

```
!git status
```

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in worki
ng directory)

        modified:   gitdemo.ipynb

Untracked files:
  (use "git add <file>..." to include in what will be committe
d)

        .DS_Store
        .ipynb_checkpoints/

no changes added to commit (use "git add" and/or "git commit -
a")
```

Cloner un dossier existant

Si on veut cloner un dossier provenant par exemple de GitHub, on utilise la fonction `$git clone` . Par exemple, le répertoire `oc-2017` où on a stocké les exercices durant l'année, peut être copié de cette manière:

In [1]:

```
!git clone https://github.com/Bugnon/oc-2017
```

```
Cloning into 'oc-2017'...
remote: Counting objects: 1088, done.
remote: Compressing objects: 100% (105/105), done.
remote: Total 1088 (delta 76), reused 57 (delta 26), pack-reus
ed 956
Receiving objects: 100% (1088/1088), 27.25 MiB | 4.09 MiB/s, d
one.
Resolving deltas: 100% (486/486), done.
```

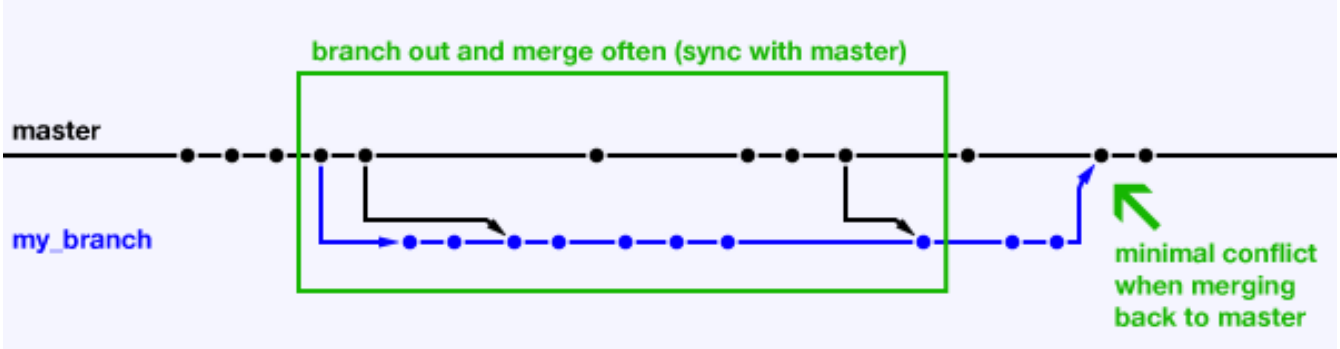
Maintenant tout le répertoire est cloné.

Git push et Git pull

`$git push` permet d'uploader les nouvelles révisions et `$git pull` permet de downloader les dernières modifications distantes du projet

Développer avec Git

Comme dit précédemment, Git à l'avantage d'être pratique à utiliser pour les projets nécessitant beaucoup de collaborateur. Pourquoi? C'est simple à comprendre: Il y a un système de **branche**.

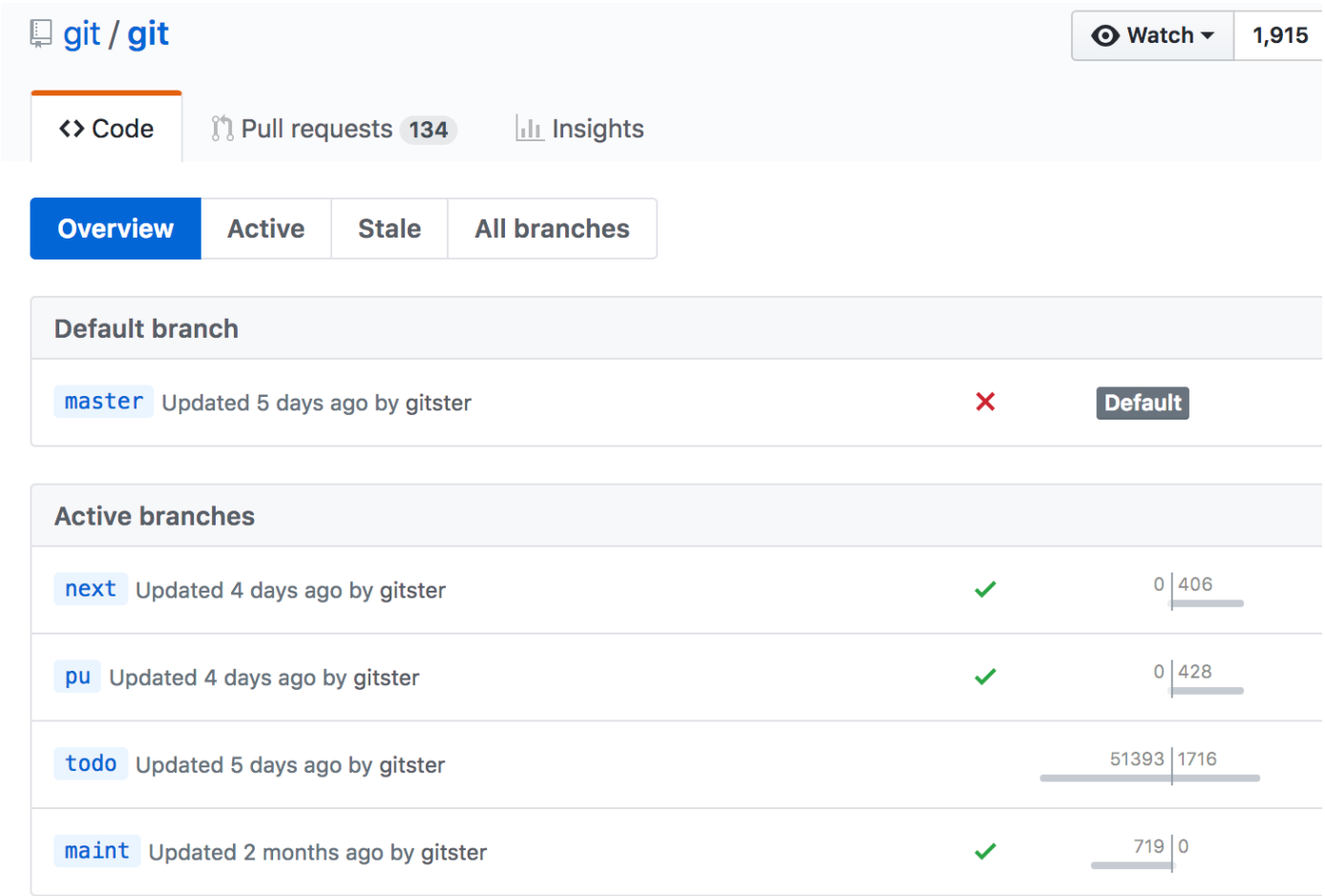


Source de l'image: [kentnguyen.com (http://kentnguyen.com/development/visualized-git-practices-for-team/)](http://kentnguyen.com/development/visualized-git-practices-for-team/)

Si je veux ajouter une nouvelle fonctionnalité (ou corriger un beug), je crée une nouvelle **branche** (par exemple: my_branch) afin de ne pas modifier le projet "final" situé sur la branche principale (**master**).

Après, dès que ces fonctionnalités sont fonctionnelles, on peut fusionner (**merge**) la nouvelle branche sur la principale.

Par exemple sur le code de dévellopement de Git lui même, il y a 5 branches (voici un aperçu graphique sur GitHub):



L'OpenSource est mis en avant avec Git (et GitHub). Même si on est pas administrateur d'un projet, mais qu'on a quand même envie de contribuer en enlevant des erreurs ou en ajoutant des fonctionnalité, on a déjà accès au code. De plus, on peut faire un **Pull request**. C'est un peu comme une branche mais n'importe qui peut en proposer. En faisant le pull request, on donne accès au "supplément de code" et le(s) administrateur(s) du projet on le choix d'intégrer ou non ce nouveau code à la branche master. *Sur la capture décran ci-dessus, on voit que pour "améliorer" Git, il y a 134 Pull requests. (certaines ont été acceptées, d'autres non)*

Mettons cela en application:

imaginons qu'en plus du calendrier du mois de mai, on veut aujouter tout le calendrier de 2018. Mais on aimerait pas modifier tout de suite la branche master.

In [3]:

```
!cal 2018
```

```
                                2018

      Janvier                 Février                   Mars

Di Lu Ma Me Je Ve Sa  Di Lu Ma Me Je Ve Sa  Di Lu Ma Me Je Ve
Sa
    1  2  3  4  5  6                1  2  3               1  2
3
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   4  5  6  7  8  9
10
14 15 16 17 18 19 20  11 12 13 14 15 16 17  11 12 13 14 15 16
17
21 22 23 24 25 26 27  18 19 20 21 22 23 24  18 19 20 21 22 23
24
28 29 30 31           25 26 27 28           25 26 27 28 29 30
31


       Avril                   Mai                     Juin

Di Lu Ma Me Je Ve Sa  Di Lu Ma Me Je Ve Sa  Di Lu Ma Me Je Ve
Sa
 1  2  3  4  5  6  7         1  2  3  4  5                  1
2
 8  9 10 11 12 13 14   6  7  8  9 10 11 12   3  4  5  6  7  8
9
15 16 17 18 19 20 21  13 14 15 16 17 18 19  10 11 12 13 14 15
16
22 23 24 25 26 27 28  20 21 22 23 24 25 26  17 18 19 20 21 22
23
29 30                 27    29 30 31        24 25 26 27 28 29
 30


      Juillet                  Août                  Septembre

Di Lu Ma Me Je Ve Sa  Di Lu Ma Me Je Ve Sa  Di Lu Ma Me Je Ve
 Sa
 1  2  3  4  5  6  7            1  2  3  4
 1
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   2  3  4  5  6  7
 8
15 16 17 18 19 20 21  12 13 14 15 16 17 18   9 10 11 12 13 14
 15
22 23 24 25 26 27 28  19 20 21 22 23 24 25  16 17 18 19 20 21
 22
29 30 31              26 27 28 29 30 31     23 24 25 26 27 28
 29
                                            30
```

```
      Octobre                Novembre               Décembre

Di Lu Ma Me Je Ve Sa  Di Lu Ma Me Je Ve Sa  Di Lu Ma Me Je Ve
 Sa
    1  2  3  4  5  6                1  2  3
 1
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   2  3  4  5  6  7
 8
14 15 16 17 18 19 20  11 12 13 14 15 16 17   9 10 11 12 13 14
 15
21 22 23 24 25 26 27  18 19 20 21 22 23 24  16 17 18 19 20 21
 22
28 29 30 31           25 26 27 28 29 30     23 24 25 26 27 28
 29
                                            30 31
```

Pour cela, on crée une nouvelle branche.

In [5]:

```
!git branch cal_2018
```

Pour connaître **toutes les branches** du projet, on peut utiliser `git branch -a`

Maintenant il faut se déplacer de master à cal_2018.

In [7]:

```
!git checkout cal_2018
```

```
M       gitdemo.ipynb
Switched to branch 'cal_2018'
```

Ensuite on ajoute le calendrier dans un document texte et ajoute (**add**) et **commit**.

In [20]:

```
!cal 2018 > cal_2018.txt
!git status
```

```
On branch cal_2018
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in worki
ng directory)

        modified:   cal_2018.txt
        modified:   gitdemo.ipynb

Untracked files:
  (use "git add <file>..." to include in what will be committe
d)

        .DS_Store
        .ipynb_checkpoints/
        oc-2017/

no changes added to commit (use "git add" and/or "git commit -
a")
```

In [22]:

```
!git add cal_2018.txt
!git commit -m 'added calender 2018 text file in branch "cal_2018"'
```

```
[cal_2018 f34a202] added calender 2018 text file in branch "ca
l_2018"
 1 file changed, 36 insertions(+), 36 deletions(-)
 rewrite cal_2018.txt (63%)
```

Maintenant que notre "fonctionnalité" fonctionne, nous pouvons l'ajouter sur **master**.

In [23]:

```
!git checkout master
!git merge cal_2018
```

```
M       gitdemo.ipynb
Switched to branch 'master'
Updating 9361f0d..f34a202
Fast-forward
 cal_2018.txt | 36 ++++++++++++++++++++++++++++++++++++
 1 file changed, 36 insertions(+)
 create mode 100644 cal_2018.txt
```

Pour aller plus loin, on pourrait **supprimer la branche**. Pour cela, il faudrait utiliser `git branch -d cal_2018` , et pour **supprimer un document** de Git, il faut utiliser `git rm nom_du_fichier`

Pour voir les dernières modifications entre le dernier commit et l'espace actuel de travail:

In [7]:

```
!git diff HEAD
```

```
ces en ligne offrant une interface graphique.\n",
@@ -36,7 +37,7 @@
     "- Permet d’avoir un historique précis de tous les change
ments et modifications d’un projet. Cela permet de clarifier l
es questions récurrentes: _“Où est la dernière version du fich
ier X?”_ et _“Qu’est-ce qui a été changé entre les révisions 4
1 et 42”?_.\n",
     "\n",
     "\n",
-    "Annecdote selon magazine PC World: _« quand on lui a dem
andé pourquoi il avait appelé son logiciel “git”, qui est à pe
u près l'équivalent de “connard” en argot britannique, Linus T
orvalds a répondu “je ne suis qu'un sale égocentrique, donc
 j'appelle tous mes projets d'après ma propre personne. D'abor
d Linux, puis Git.”»_"
+    "Annecdote selon magazine PC World: « quand on lui a dema
ndé pourquoi il avait appelé son logiciel “git”, qui est à peu
 près l'équivalent de “connard” en argot britannique, Linus To
rvalds a répondu “je ne suis qu'un sale égocentrique, donc j'a
ppelle tous mes projets d'après ma propre personne. D'abord Li
```

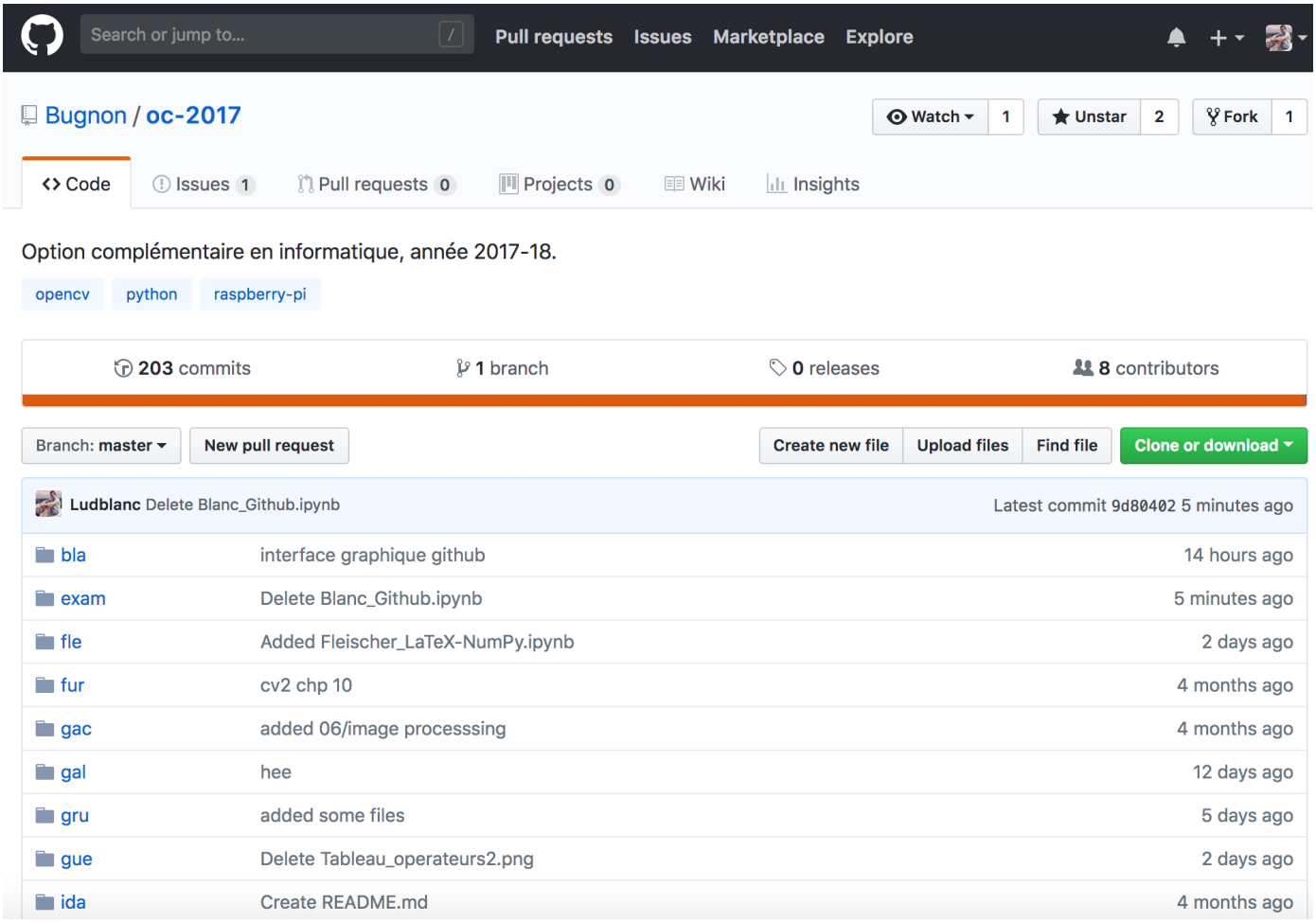
Collaboration avec GitHub

GitHub est un service web offrant l’hébergement de projets utilisant Git, lancé en 2008, GitHub offre de nombreuses fonctionnalités facilitant la communication et la collaboration. Cette plateforme est très pratique car c'est une interface graphique, simplifiant grandement l'accèsibilité. Elle est aussi comme un réseau social:

- On peut suivre des personnes
- On peut suivre l'évolution d'un projet

De plus, le service est gratuit pour les privés. Le code de tout les projets y est visible ce qui, encore une fois, permet le côté OpenSource. On peut aussi voir les pull requests et des **issues**. Les issues sont une sorte de chat ou on peut discuter des problèmes et de leurs soulutions d'un projet.

Voici l'illustration de l'interface graphique avec le dossier de classe.



La plupart des logiciels et modules python qu'on a utilisé durant l'année sont en développement sur GitHub. Ils sont donc OpenSource.

Je vous recommande d'aller jeter un coup d'œil aux liens suivants:

- https://github.com/jupyter/notebook (https://github.com/jupyter/notebook)
- https://github.com/opencv/opencv (https://github.com/opencv/opencv)
- https://github.com/numpy/numpy (https://github.com/numpy/numpy)
- https://github.com/python/cpython (https://github.com/python/cpython)
- https://github.com/git/git (https://github.com/git/git)
- https://github.com/martinohanlon/mcpi (https://github.com/martinohanlon/mcpi)
- https://github.com/RPi-Distro/thonny (https://github.com/RPi-Distro/thonny)
- https://github.com/RPi-Distro (https://github.com/RPi-Distro)

On voit qu'il y a plein de Pull requests et d'issues.