Git et Github

Ce document est là pour vous aider à comprendre le fonctionnement de Git et Github

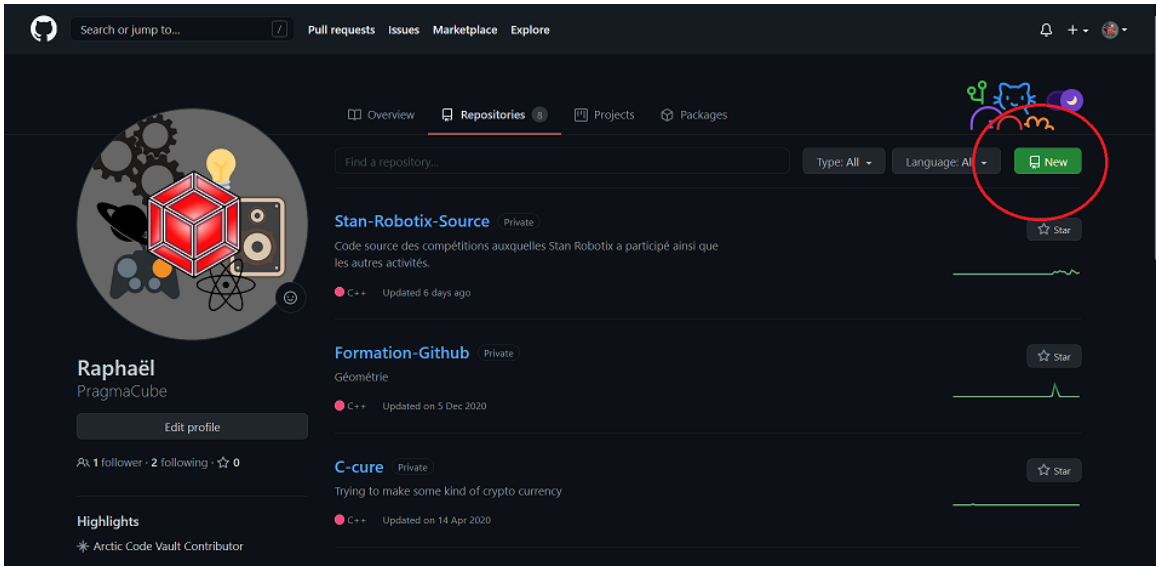
Rédaction : Raphaël
Révision : Cyril

I – Notions de base

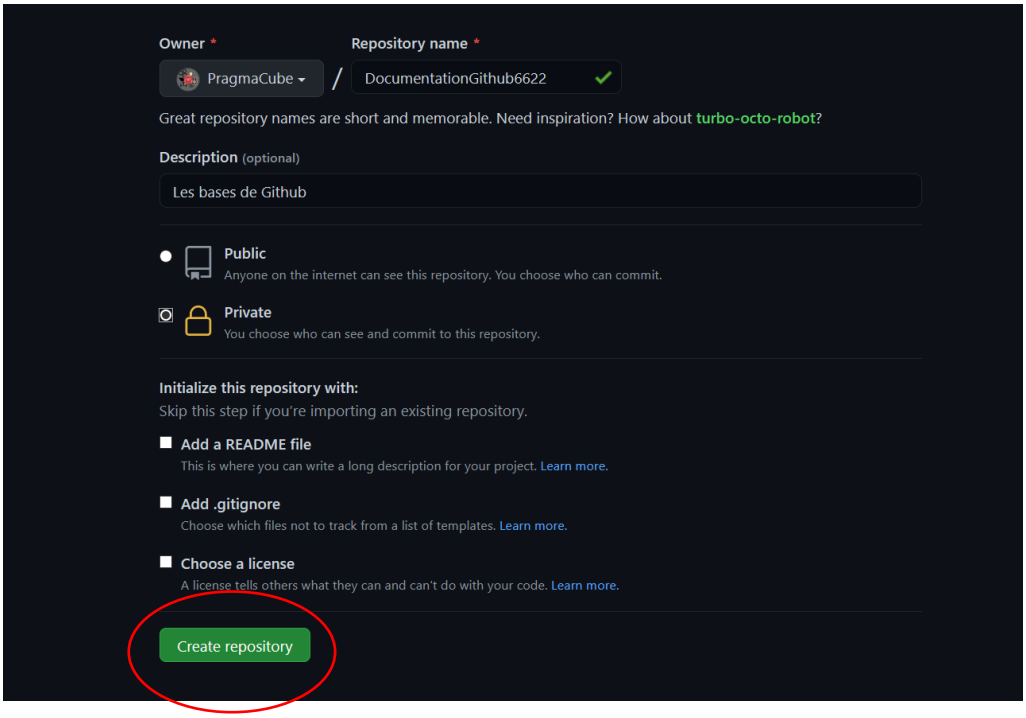
- Une repository est un l'espace de stockage sur un serveur (e.g. les serveurs de Github pour notre cas).
- Git est un système de gestion de version (de code). Il fonctionne par branches dans lesquelles les contributeurs peuvent rajouter une pierre à l'édifice (ils font un commit).
- Github est la plateforme que nous utilisons pour héberger le code. Il en existe d'autres comme Gitlab.

II – Créer / cloner une repository

Pour en créer, il suffit d'aller sur votre compte Github et de cliquer sur « New » dans la section « Repositories ».



Vous devrez ensuite choisir son nom, sa description, sa disponibilité (public/privée) et d'autres options comme inclure un Readme (description plus complète pour l'utilisation) ou une license. Appuyez ensuite sur « Create repository ».



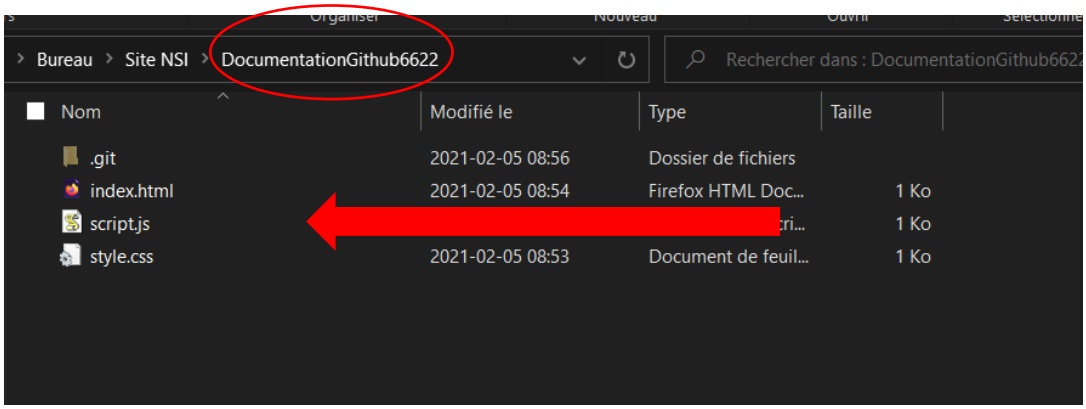
Si vous voulez travailler sur une repository déjà existante, vous devrez la cloner. Assurez-vous d'abord d'avoir les droits d'accès à cette repository, puis allez sur sa page principale pour récupérer son lien.

Sur votre Bureau, créer un nouveau répertoire, ouvrez-le et faites clic-droit. Appuyez sur « Git bash here » et entrez la commande « git clone [le lien] » comme suit (puis Enter) :

```
pragm@LAPTOP-JOH63IEO MINGW64 ~/OneDrive/Bureau/Site NSI
$ git clone https://github.com/PragmaCube/DocumentationGithub6622.git
```

III – Rajouter du code à votre repository

Maintenant que vous avez une repository dans laquelle travailler, vous pouvez rajouter du code. Pour l'exemple j'ai fait un mini site (très petit, moins de 40 lignes de code). Il faut placer les fichiers dans votre répertoire :



La prochaine étape est de faire un commit puis de push les changements.

D'abord, ajouter vos fichier avec « git add [fichier.extension] » ou « git add * » si vous voulez ajouter tous les fichiers du répertoire. Ensuite, « git commit -m "`message qui a du sens`" » pour faire un commit et enfin « git push » pour mettre à jour la branche actuelle (ici master, donc la branche principale).

```
pragm@LAPTOP-JOH63IEO MINGW64 ~/OneDrive/Bureau/Site NSI/DocumentationGithub6622
 (master)
$ git add index.html

pragm@LAPTOP-JOH63IEO MINGW64 ~/OneDrive/Bureau/Site NSI/DocumentationGithub6622
 (master)
$ git add script.js

pragm@LAPTOP-JOH63IEO MINGW64 ~/OneDrive/Bureau/Site NSI/DocumentationGithub6622
 (master)
$ git add style.css

pragm@LAPTOP-JOH63IEO MINGW64 ~/OneDrive/Bureau/Site NSI/DocumentationGithub6622
 (master)
$ git commit -m "Ajout des fichiers cote client"
[master (root-commit) cc75454] Ajout des fichiers cote client
 3 files changed, 33 insertions(+)
 create mode 100644 index.html
 create mode 100644 script.js
 create mode 100644 style.css

pragm@LAPTOP-JOH63IEO MINGW64 ~/OneDrive/Bureau/Site NSI/DocumentationGithub6622
 (master)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 750 bytes | 375.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/PragmaCube/DocumentationGithub6622.git
 * [new branch]      master -> master
```

Pour mettre à jour votre répertoire local, utilisez « git pull ».

IV – Les branches

Comment faire pour ajouter un code mais qu'il ne soit pas sur master (dans l'optique d'avoir une révision)?

Réponse : utiliser une / des branche(s).

Pour créer une branche, utilisez la commande « git checkout -b NomDelaBranche » et pour changer de branche enlevez simplement « -b ».

Pour l'exemple j'ai légèrement changé le fichier style.css en changeant la couleur et la taille du titre.
Maintenant que j'ai modifié un fichier, si je fais « git status », je vais voir un fichier en rouge :

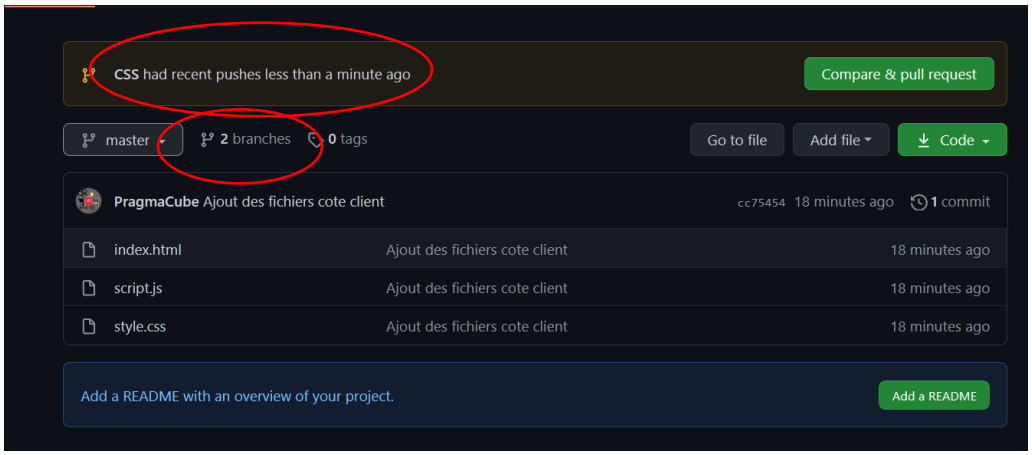
```
pragm@LAPTOP-JOH63IEO MINGW64 ~/OneDrive/Bureau/Site NSI/DocumentationGithub6622
 (CSS)
$ git status
On branch CSS
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   style.css

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

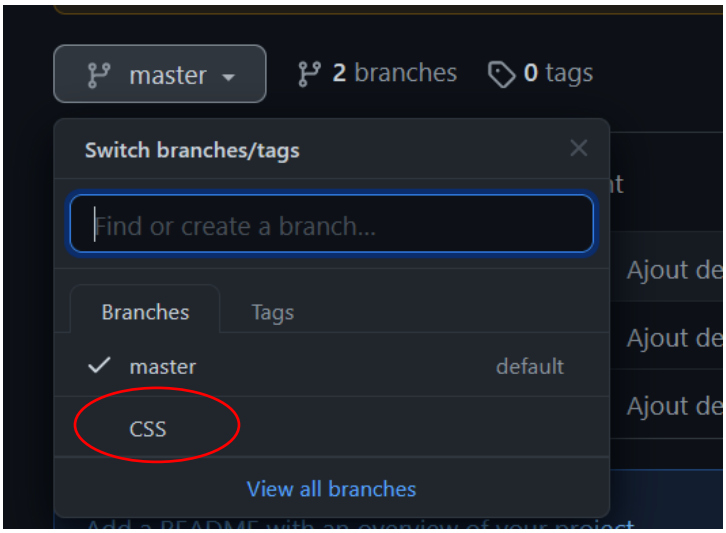
Je dois donc rajouter avec « git add » mon fichier avant de faire un commit (même procédure que pour master). Cependant, lorsque vous allez faire « git push », faites « git push –set-upstream origin NomDeLaBranche ».

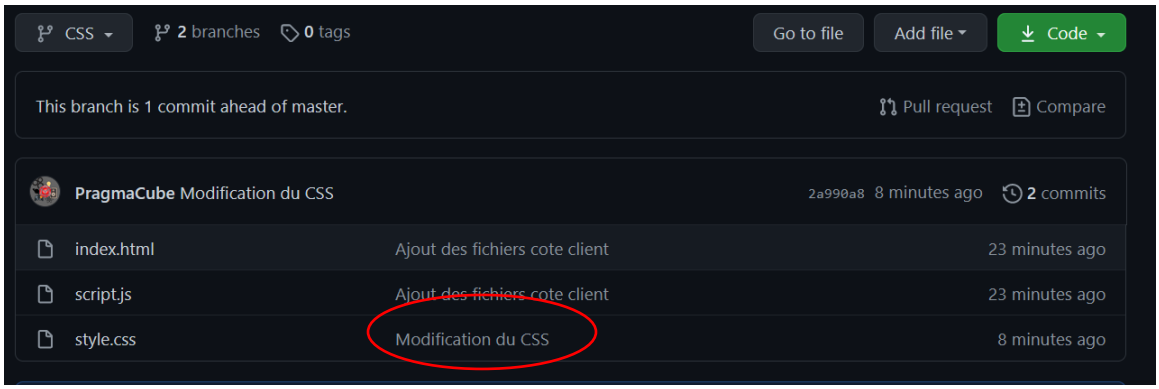
```
pragm@LAPTOP-JOH63IEO MINGW64 ~/OneDrive/Bureau/Site NSI/DocumentationGithub6622
 (CSS)
$ git add style.css

pragm@LAPTOP-JOH63IEO MINGW64 ~/OneDrive/Bureau/Site NSI/DocumentationGithub6622
 (CSS)
$ git commit -m "Modification du CSS"
[CSS 2a990a8] Modification du CSS
 1 file changed, 2 insertions(+)
```

```
pragm@LAPTOP-JOH63IEO MINGW64 ~/OneDrive/Bureau/Site NSI/DocumentationGithub6622
 (CSS)
$ git push --set-upstream origin CSS
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 357 bytes | 357.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'CSS' on GitHub by visiting:
remote:      https://github.com/PragmaCube/DocumentationGithub6622/pull/new/CSS
remote:
To https://github.com/PragmaCube/DocumentationGithub6622.git
 * [new branch]      CSS -> CSS
Branch 'CSS' set up to track remote branch 'CSS' from 'origin'.
```

Finalement, si je reviens sur Github, je constate ceci :







Les changements ont bien été pris en compte et vous n'avez pas besoin de vous souciez de « pull request », le capitaine d'équipe s'en chargera.