

Première partie – Démonstration Git

Dans cette première partie, vous allez individuellement avec votre formateur noter l'évolution d'un suivi de version git réaliser entre le serveur et votre machine (ou différentes machines).

NB : pour ne pas rentrer dans le détail de l'édition de fichier, on invente une commande `write (Address file, int line, String msg)` qui écrit dans le fichier `file`, à la ligne `line` le contenu `msg`. S'il y a déjà quelque chose d'écrit à la ligne `line`, le contenu est écrasé et remplacé par `msg`.

Int → Nombre Entier, ex : 5

String → Chaîne de caractère, ex : « Coucou »

Exercice 1-1 – SANS UTILISER D'OUTIL GIT MAIS EN PRENANT DES NOTES :

Question 1-2 - Expliquez ce qu'il se passe en donnant le résultat de chacune des commandes suivantes, sur la machine de Leia, sur la machine de Luke, ainsi que sur le serveur.

Créer un fichier de type .txt avec le résultat de chacune des commandes.

Pour cela utilisez la documentation disponible sur : <https://git-scm.com/docs>

1. Leia@linux : git clone
<https://github.com/Michael16b/PlatoonVehicule/episode4.git>
2. Leia@linux : write (plan.txt, 1, Etoile Noire)
3. Leia@linux : git status
4. Leia@linux : git add .
5. Leia@linux : git status
6. Leia@linux : git commit -m "creation du plan"
7. Leia@linux : git status
8. Leia@linux : write(plan.txt, 2, Plan technique) 9. Leia@linux : git commit -m "Plan"
10. Leia@linux : git add .
11. Leia@linux : git commit
12. Leia@linux : write(plan.txt, 3, Diamètre : 120 kilomètres)
13. Leia@linux : git commit -a -m "diametre"
14. Leia@linux : git rm plan.txt -f
15. Leia@linux : ls
16. Leia@linux : git status
17. Leia@linux : git checkout -f
18. Leia@linux : git fetch
19. Leia@linux : git status
20. Leia@linux : git push
21. Luke@macosx : git clone
<https://github.com/Michael16b/PlatoonVehicule/episode4.git>

Deuxième partie – Application de git sur PC en ligne de commande

Exercice 2 –

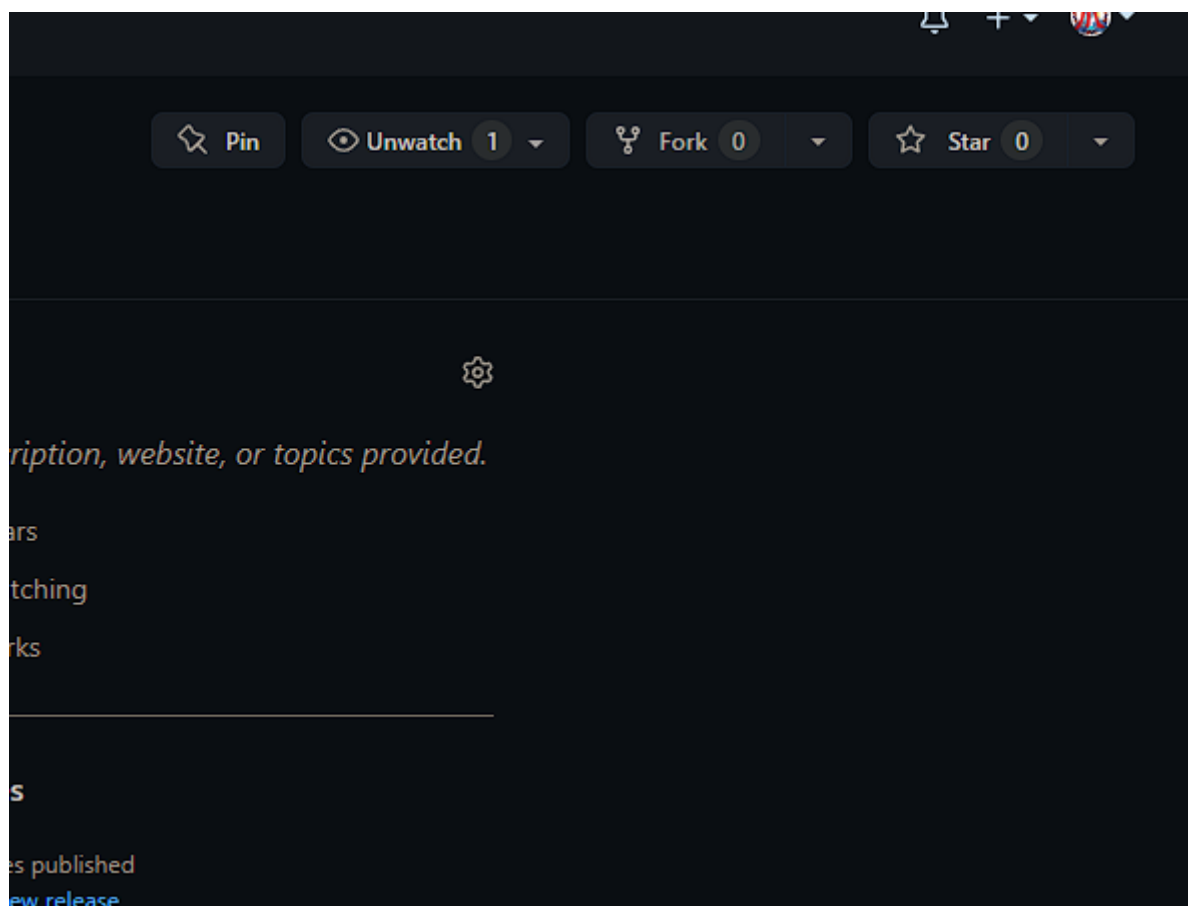
Super vous avez réussi à comprendre le fonctionnement des différentes notions de base d'un git. Désormais nous allons passer par du pratique (ce que vous attendiez depuis le début non? 😊).

Question 2-1 – Rendez-vous sur : <https://github.com/Michael16b/PlatoonVehicule>

ATTENTION : Vous ne pouvez pas modifier le projet avec le git clone : vous n'en avez d'ailleurs pas les droits.

En revanche, vous avez le droit de faire un « fork », au sens littéral c'est une bifurcation : cela crée un nouveau projet. Vous aurez le droit de travailler dans ce nouveau projet. De plus, il reste connecté au projet original, permettant des échanges entre les deux. Classiquement, cela permet de travailler de son côté, à l'extérieur de l'équipe du projet initial pour ensuite proposer ses contributions. Ou bien même du clone vu d'ailleurs en haut mais là n'est pas notre but.

En haut à droite du projet de Michael16b, vous voyez un logo fork. Cliquez dessus :




À l'écran suivant ça vous affichez un truc de ce style (Attention, si vous avez déjà créé un répertoire qui a le même nom que « PlatoonVehicule » une erreur apparaîtra, c'est normal car sur GitHub, chaque répertoire sont uniques :

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Owner *

Repository name *

 Michael16b ▾

 /


PlatoonVehicules ✓

By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.

Description (optional)

☒ Copy the `compose` branch only

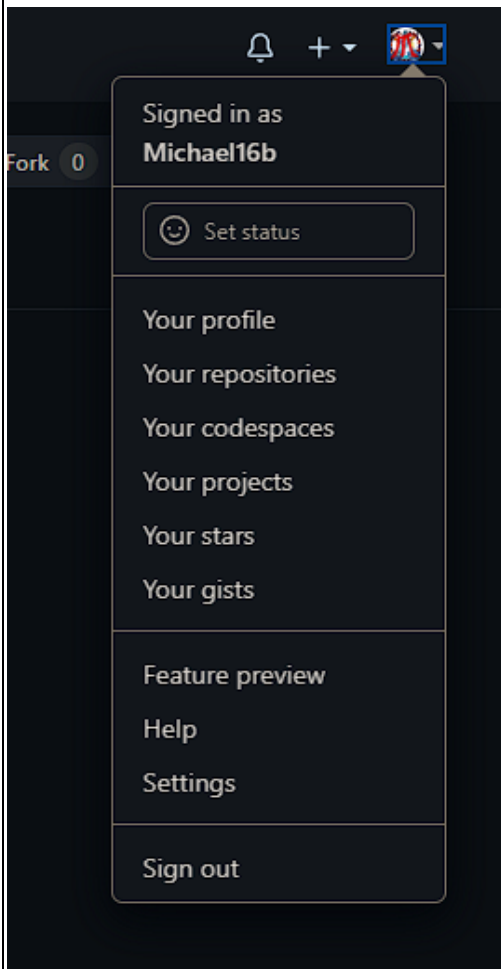
Contribute back to revanced/revanced-manager by adding your own branch. [Learn more.](#)

 You are creating a fork in your personal account.

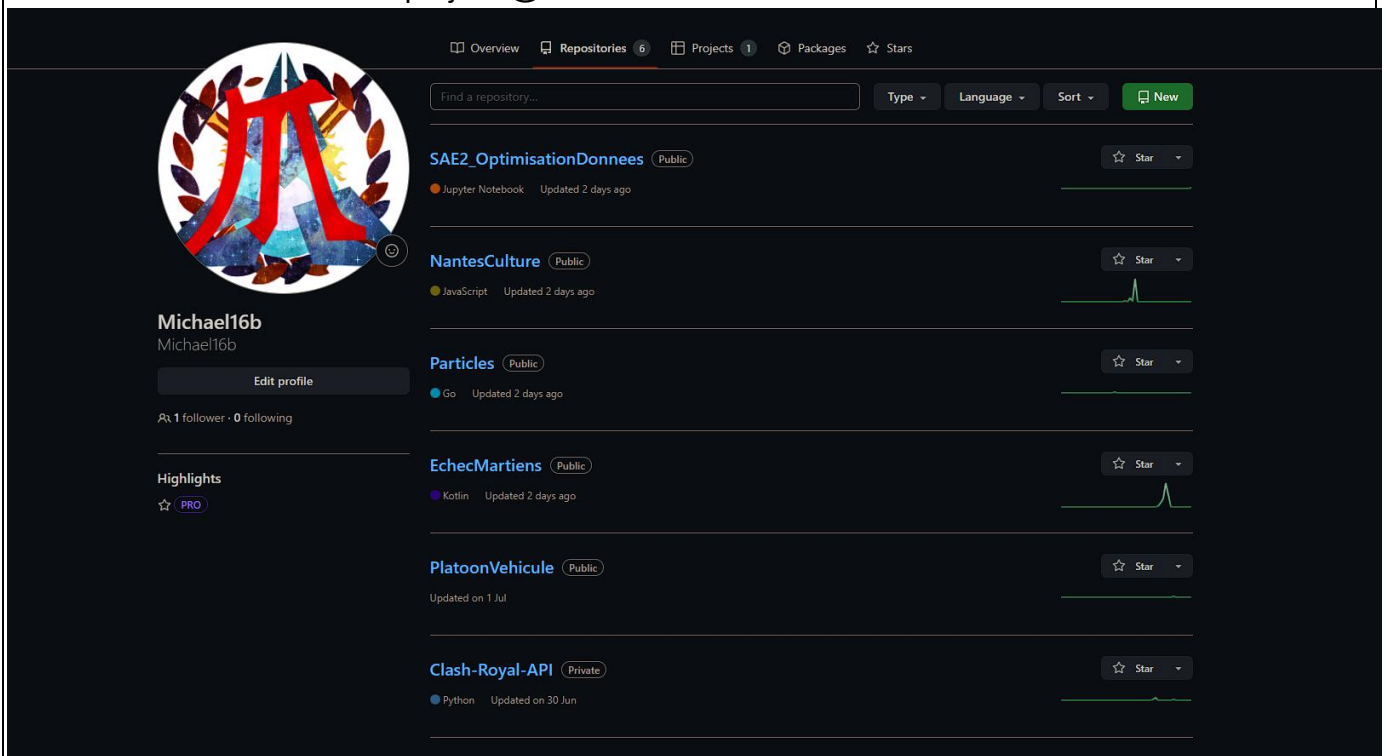
Create fork

Parfait, après avoir créé votre propre projet, vous pouvez y entrer à l'adresse suivante :
<https://github.com/votrePseudo/nomDuRepertoire>

Vous avez la flemme ? Pas d'inquiétude, vous pouvez y entrer via le menu GitHub :
En haut à droite, juste à côté de votre logo, cliquez sur la flèche, puis « Your repositories ».



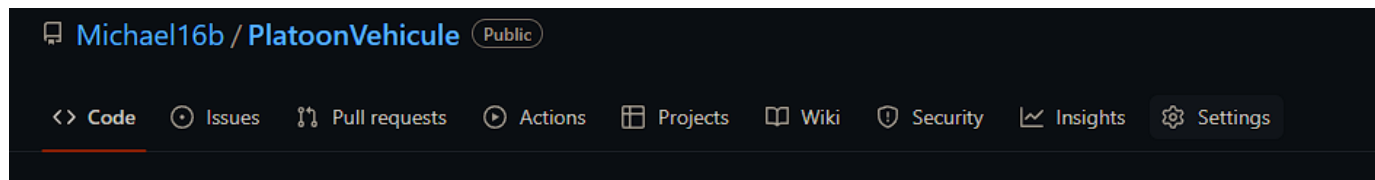
Et vous aurez accès à tous vos projets 😊



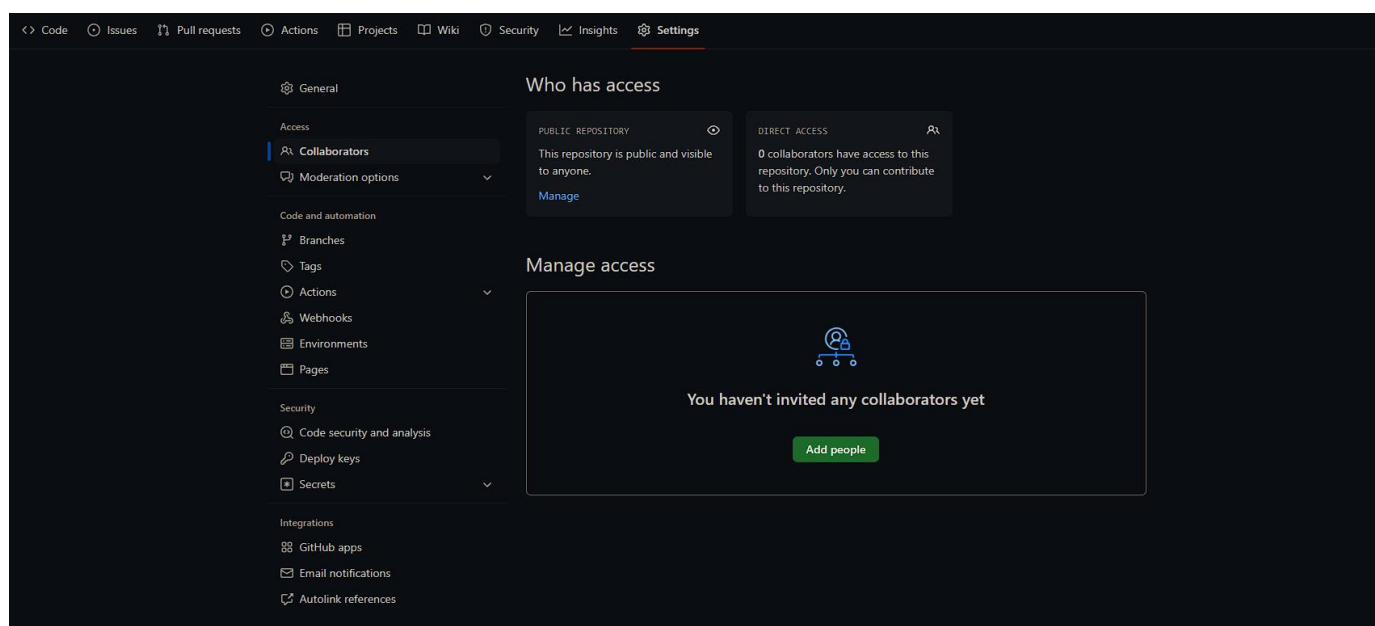
Bon c'est cool tout ça mais imaginons que vous créer un projet qui doit rester secret, que vous ne voulez pas partager votre code, travail à n'importe qui mais uniquement à des personnes spécifiques ?

Cela est possible !!

En haut de votre projet, cliquez sur Settings :



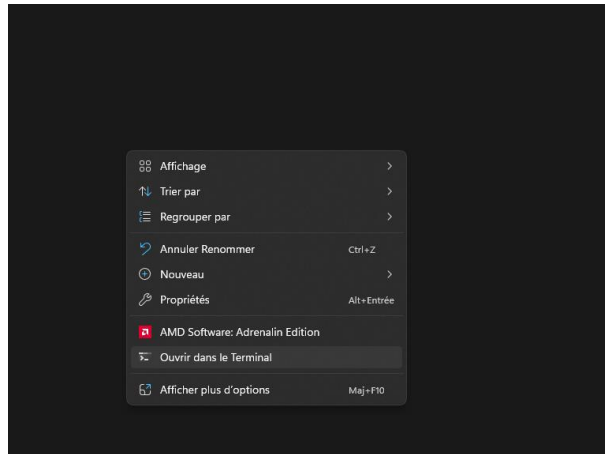
Puis collaborators et « Add people » puis ajoutez le pseudo mail des personnes à qui vous souhaitez rendre disponible votre projet.



PS : Si vous êtes des personnes qui sont curieux vous avez peut-être remarqué que le répertoire est public, c'est-à-dire tout le monde peut y entrer, vous pouvez le passer priver pour que personne autre que vos membres peuvent y entrer (d'où l'utilité de ça).

Question 2-2 – Bon c'est cool tout ça mais on a enfin notre projet sur notre git et on aimerait pouvoir y accéder depuis notre PC !

Ok, ok, on va commencer la partie la plus compliqué !! Ouvrez un terminal, déplacez-vous dans un répertoire de travail créé pour l'occasion. Si vous êtes sur Windows 11 vous n'avez juste qu'à faire clic droit sur le dossier correspondant puis « Ouvrir depuis le terminal » :



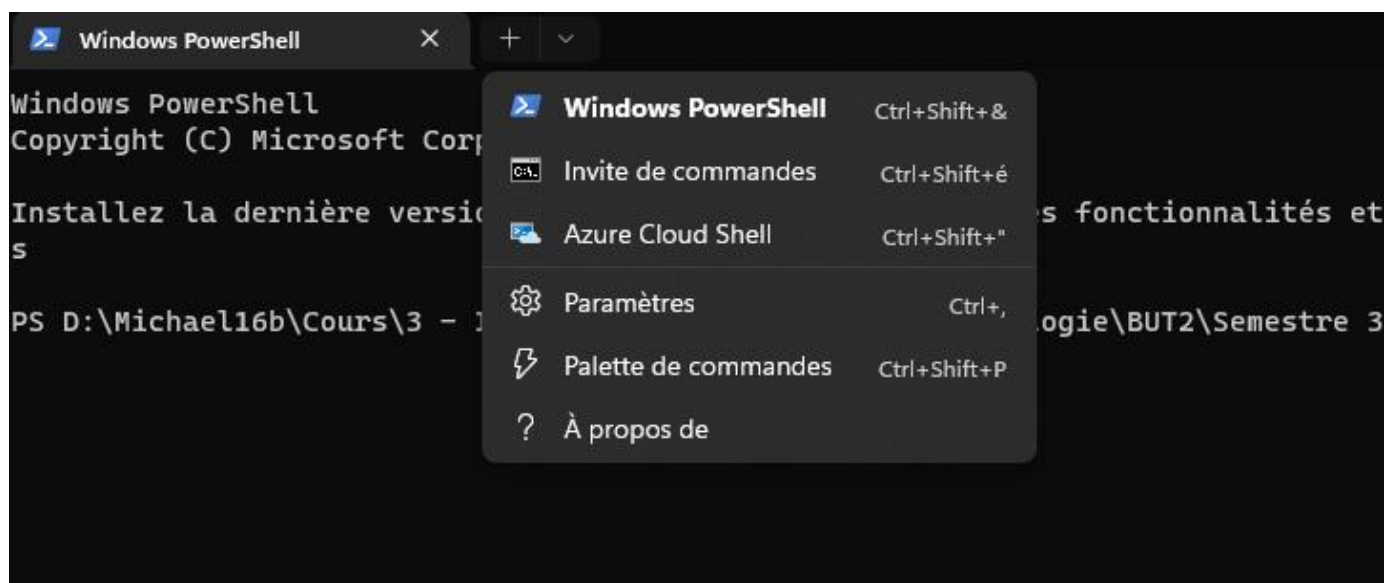
Et hop vous avez ça :

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

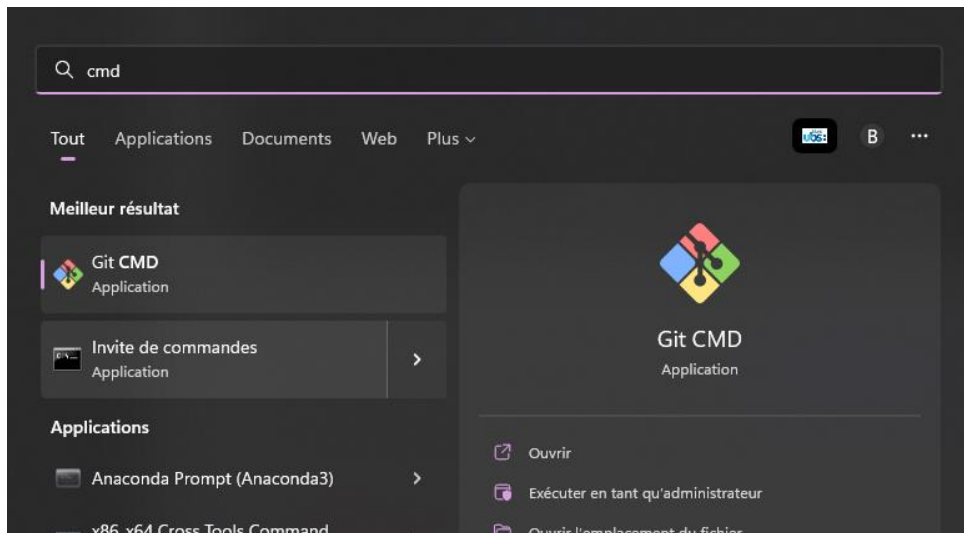
Installez la dernière version de PowerShell pour de nouvelles fonctionnalités et améliorations ! https://aka.ms/PSWindows

PS D:\Michael16b\Cours\3 - Institut Universitaire de Technologie\BUT2\Semestre 3\Préparation\Cours\Git\monPremierGit> |
```

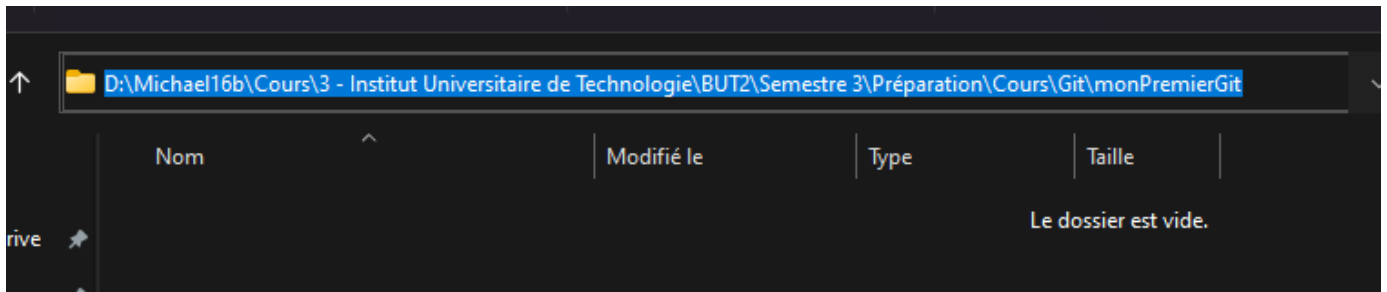
Je vous conseille d'ailleurs de passer en Invite de Commande(cmd) c'est beaucoup plus pratique et moins puissant que PowerShell. Pour se faire, cliquez sur la petite flèche en haut à gauche du terminal puis « Invite de commande » :



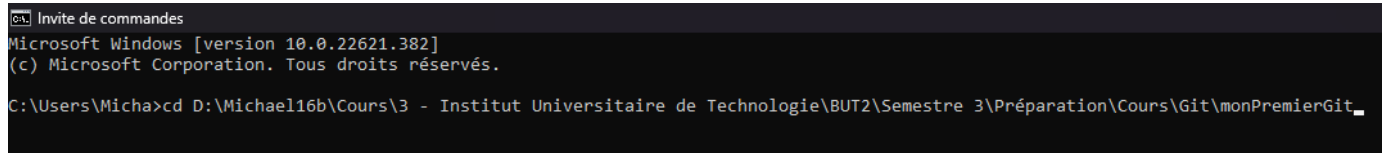
Pour ceux qui ne possède pas Windows 11, cliquez sur le logo Windows puis sur la barre de recherche, tapez « cmd » puis cliquez sur « Invite de commande ».



Désormais il faut retrouver le répertoire de votre projet, pour ce faire depuis votre répertoire courant de votre projet, double cliquez sur la barre d'adresse et copiez-le :



Ensuite, faites « cd votreRepertoire » :



Pour ceux qui sont sur Linux et Mac, je n'ai pas eu de formation sur ça du coup je vous conseille de faire manuellement (Ex : `cd monRepertoire/unAutreRepertoire/monRepertoireDeProjet`) et de créer un script exécutable (.sh sur Linux).

Alternative (toutes appareils) : Installez [VSCode](https://code.visualstudio.com/), ouvrez le dossier de votre projet puis ouvrez un Terminal (disponible en haut au milieu de votre application).

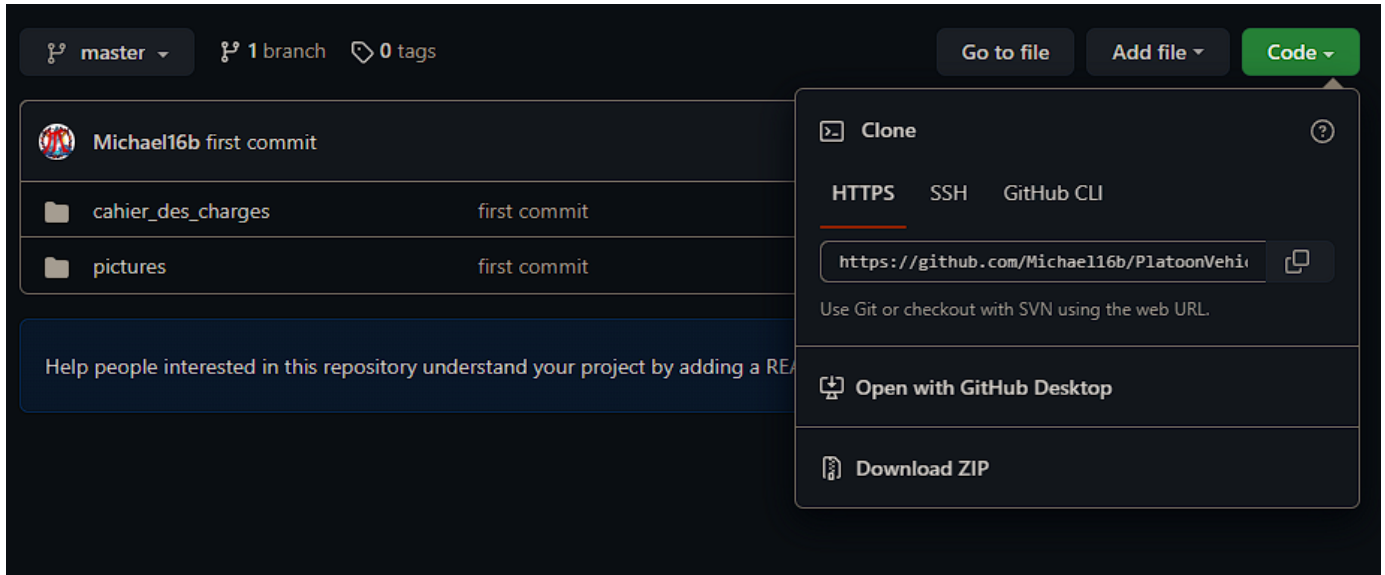
PS : Sur vos machines personnelles, il faut installer git au préalable : <https://git-scm.com/>

Sur la page d'accueil du projet (avant qu'il y ait déjà eu du travail effectué), quelques commandes devraient être données pour vous guider. Il faut d'abord vous identifier (si ce n'est pas le cas tant mieux !) :

```
git config --global user.name "<identifiant>" git
git config --global user.email "<mail>"
```

Puis, effectuez le clone dans ce répertoire (il y a plusieurs manières de s'authentifier, comme la suivante) :

Cliquez sur « Code » sur GitHub puis copier le lien HTTPS :



Question 2-3 – Essayer désormais de télécharger votre travail sur votre PC.

Indice : `git clone` + le lien que vous avez copié précédemment

Après cela essayer de modifier des choses sur le fichier écrit en mark down (.md)

Après avoir fait cela, puis enregistré faites les commandes dîtes classiques :

1. On ajoute à l'index (`git add .`), on versionne (`git commit -m « votre message »`)
2. **IMPORTANT !! On vérifie qu'il n'y a rien de nouveau (`git fetch`, `git status` et s'il y a du nouveau `merge`, `git merge`).**
3. POURQUOI ? Car il peut y exister des conflits si vous travaillez à plusieurs où sur plusieurs PCs et pour éviter tout problèmes on fait ces commandes pour vérifier que les fichiers sont bien mis à jour par rapport à nos fichiers sur notre PC
4. On pousse (`push`)

Faites cela, en observant l'évolution du dépôt en ligne (il faut actualiser la page) : Insights > Commits

Et en faisant des commandes `git status` pour l'évolution en local.

Troisième partie – Un travail collaboratif (Optionnel)

Exercice 3 – Passez votre projet GitHub en privé comme vu en haut et ajoutez-y des personnes en collaboration avec vous. Donnez-le des droits pour permettre d'écrire, s'il n'y arrive pas.

Plus d'information ici : <https://docs.github.com/en/organizations/organizing-members-into-teams/assigning-the-team-maintainer-role-to-a-team-member>

On ne donne pas forcément les mêmes droits d'écriture sur le référentiel public à tous les collaborateurs. Typiquement, ils ne sont pas « maintenir » (donne des permissions admins) mais seulement « développeur », ils ont le droit de récupérer la branche principale (origin/master) mais pas d'y faire des pushes. Néanmoins ils ont le droit de créer d'autres branches sur le référentiel public pour travailler. Finalement quand ils pensent que leur travail mérite d'être intégré, ils demanderont aux maintainers de récupérer leur travail : c'est un « *merge request* » (dénomination de gitlab, « *pull request* » pour github).

Propriétaire du projet Git : Descendez le niveau des droits à votre groupe en tant que « Member ».

Question 3-1 - Travaillez cette série d'action :

Essayez d'ajouter du texte dans `cahierdescharges.md`

« L'entreprise ElectroIUT souhaite produire des véhicules qui peuvent rouler en train en se suivant les uns les autres. »

Il essaie de partager cela sur le dépôt partagé : *add, commit, push*.

Que se passe-t-il ?

Question 3-2 – Travail sur une branche

1. `Personne2 : git branch collaboration`
2. `Personne2: git checkout collaboration`
3. `Personne2: ls`
4. `Personne2: Vérifier le contenu de cahierdescharges.md`
5. `Personne2: git push origin collaboration`

Il crée une branche (*branch*), il bascule sa copie de travail sur cette branche pour y travailler (*checkout*), il travaille (versionne), il pousse sur le dépôt partagé (origin) la branche (collaboration).

Question 8 – Une fois cette contribution faite sur sa branche, il va falloir l'intégrer à la branche principale. Cela se passe en ligne sur GitHub.

Trinôme2 demande la création d'un « `pull request` » qu'on documente pour convaincre les maintainers(=admin) du référentiel public.

Les autres trinômes peuvent aller sur la page « `pull requests` » pour trouver la demande. Cela peut ouvrir des discussions, on peut aller comparer les branches, analyser le graphe, etc.

Si les maintainers acceptent, l'un d'eux fait « *merge* », ce que tout le monde peut vérifier en ligne et en local :

1. `git pull`
2. `git checkout master` (pour le Trinôme2)

Un *pull* récupère l'ensemble du dépôt avec toutes ses branches, un *checkout* bascule la version de travail dans la branche voulue pour ouvrir et vérifier son contenu.

INFO SUP : Tout cela peut être fait avec des outils plus graphiques. Soit directement dans un IDE (un environnement de développement logiciel), par exemple utiliser Eclipse et le plugin eGit, soit avec un outil dédié à git (sourcetree par exemple sous windows, gitkraken sous linux) pour effectuer un travail similaire.

Vous pouvez vous perfectionner sur l'utilisation du Git gratuitement via :

<https://openclassrooms.com/fr/courses/7162856-gerez-du-code-avec-git-et-github>