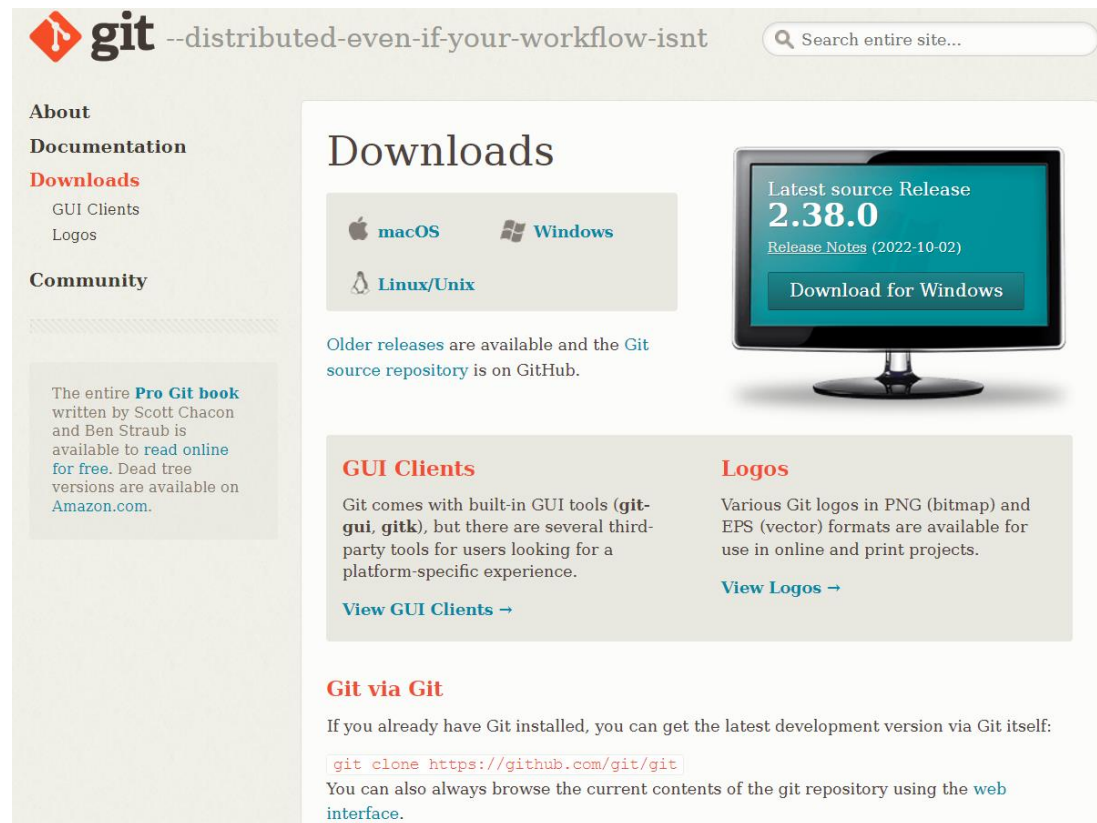


Git : est un logiciel (un outil) permettant de contrôler les versions d'un projet informatique.

Pour se faire :

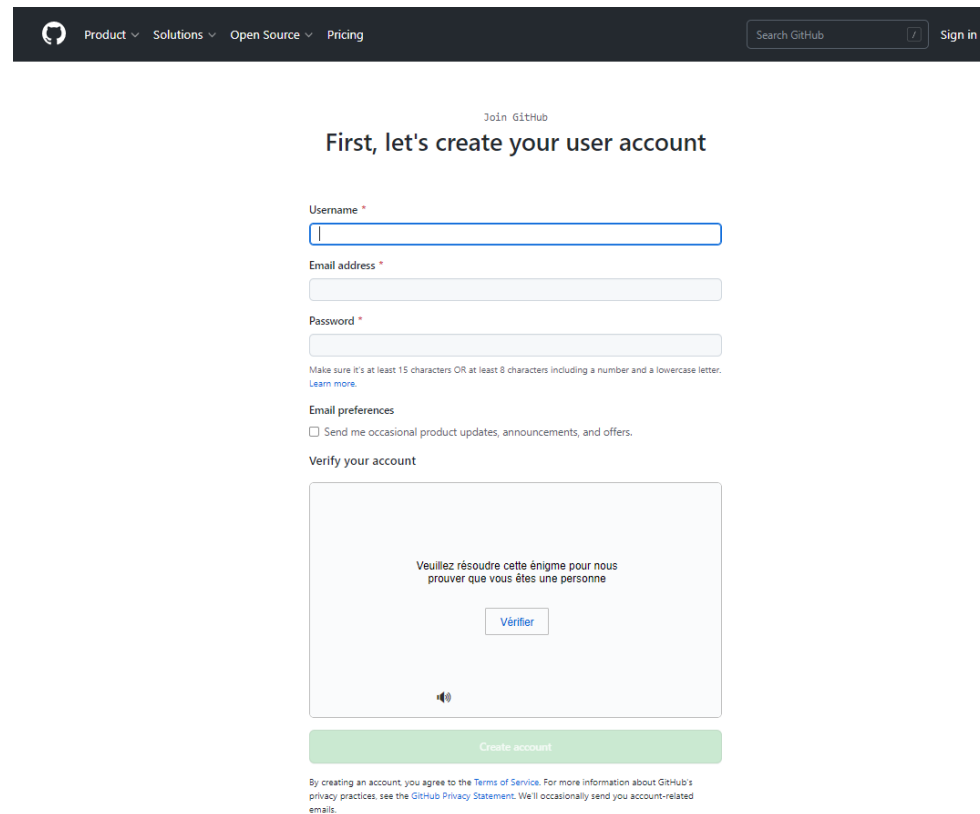
Etape1: On commence par télécharger la dernière version de git (<https://git-scm.com/downloads>),
(Téléchargement et installation classique)



Etape2: Créer un espace qui contiendra l'ensemble de fichiers de projet pour le partager à distance avec un collaborateur.

Cet espace hébergé à distance s'appelle un **Repository** (dépôt), il existe plusieurs sites pour en créer un facilement et gratuitement (ex. JFrog, kbroman...), le plus connu est GitHub.

En ligne donc, on crée un compte sur github (on peut aussi télécharger la version desktop).

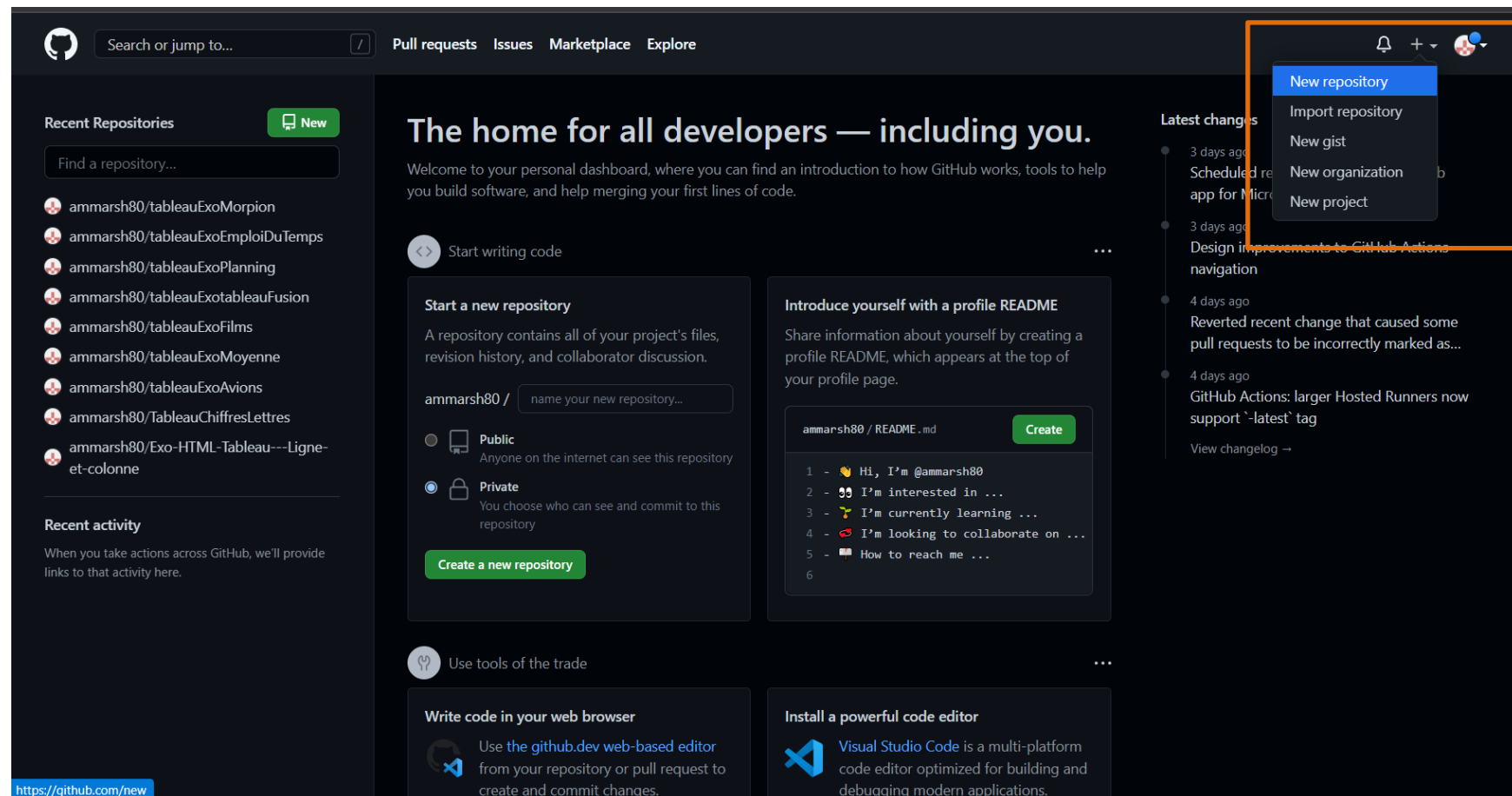


The screenshot shows the GitHub website's account creation page. At the top is a dark navigation bar with the GitHub logo, links for Product, Solutions, Open Source, and Pricing, a search bar, and a Sign in button. The main heading is 'Join GitHub' followed by 'First, let's create your user account'. The form includes fields for Username, Email address, and Password, with a note about password requirements. Below these is an 'Email preferences' section with a checkbox for product updates. A 'Verify your account' section contains a large box with a CAPTCHA puzzle and a 'Vérifier' button. At the bottom is a green 'Create account' button. A footer note states that creating an account implies agreement to the Terms of Service and Privacy Statement.

À partir de ce moment on peut créer autant de repositories/dépôts qu'on souhaite.

Pour se faire : on clique sur new repository.

Et là, il faudra spécifier plusieurs informations comme :



- le nom de dépôt : **Repository name** (pour nous le nom de l'exercice),
- une petite **Description** (pas obligatoire, ex. première page web).

Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * **Repository name ***
ammarsh80 /
Great repository names are short and memorable. Need inspiration? How about [supreme-succotash?](#)

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)
.gitignore template: **None**

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)
License: **None**

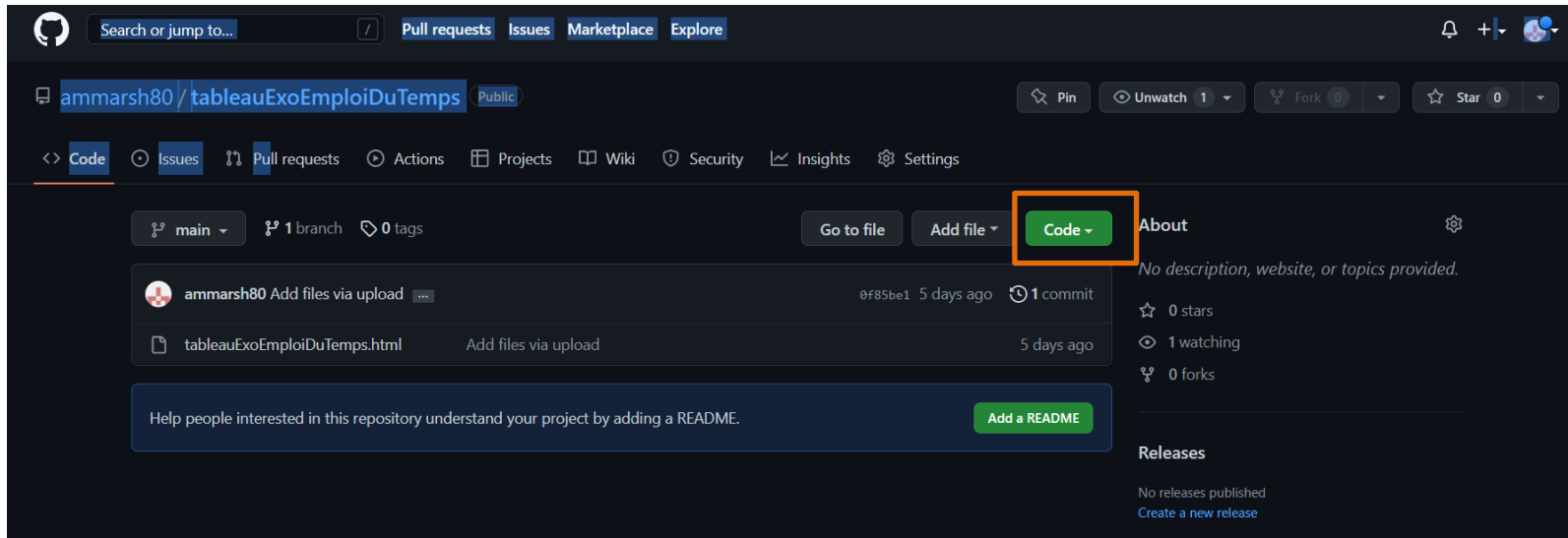
📘 You are creating a public repository in your personal account.

Create repository

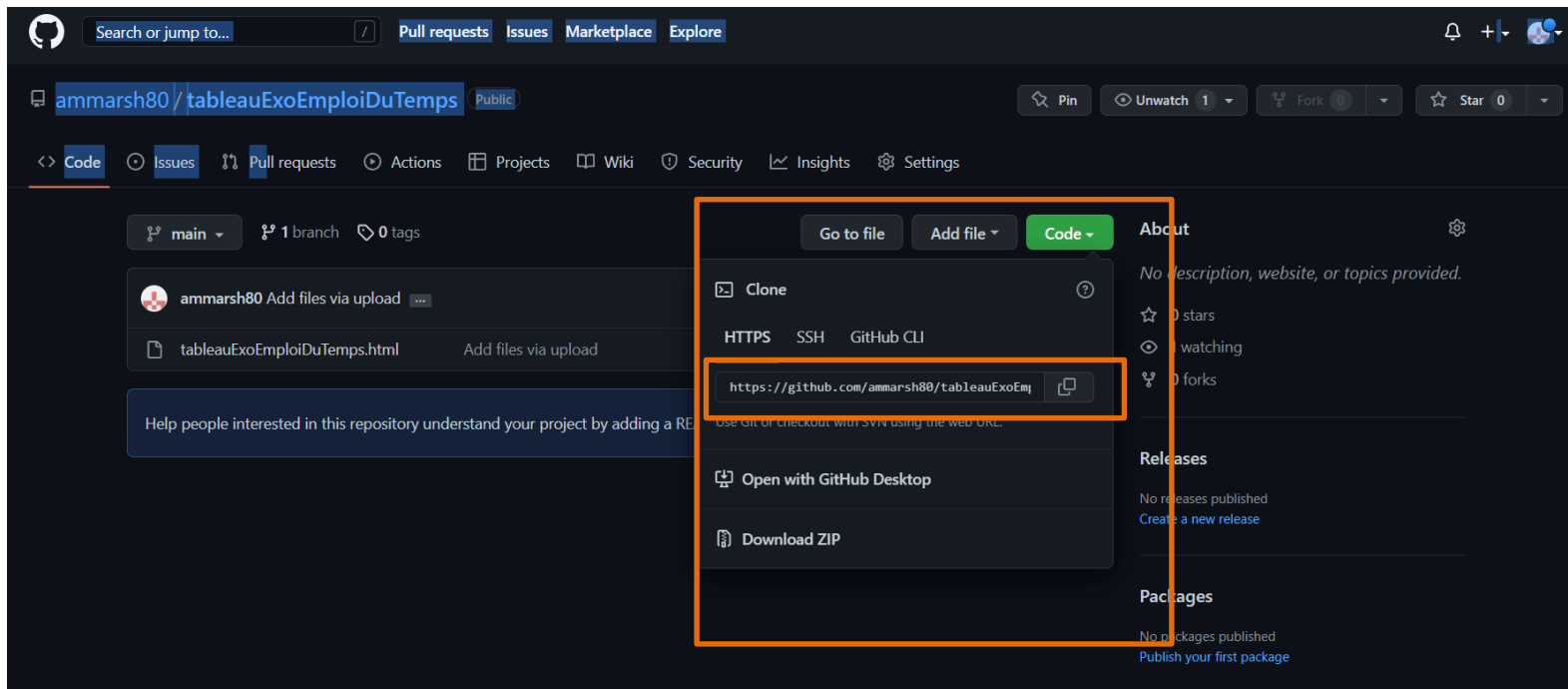
Puis, (obligatoire) si on veut que le code soit privé ou publique (attention il est publique par défaut).

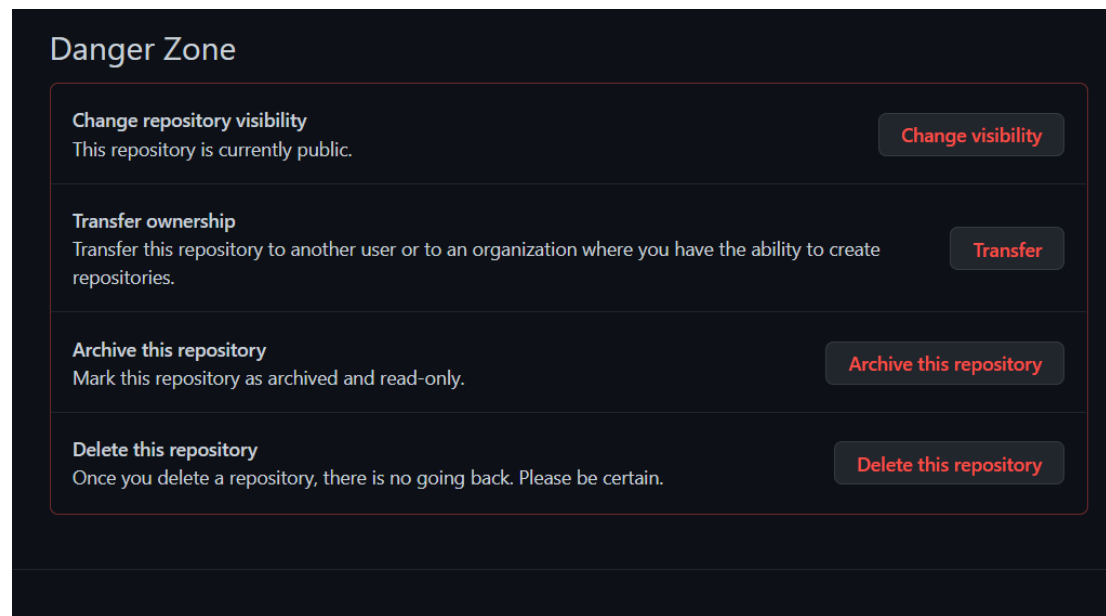
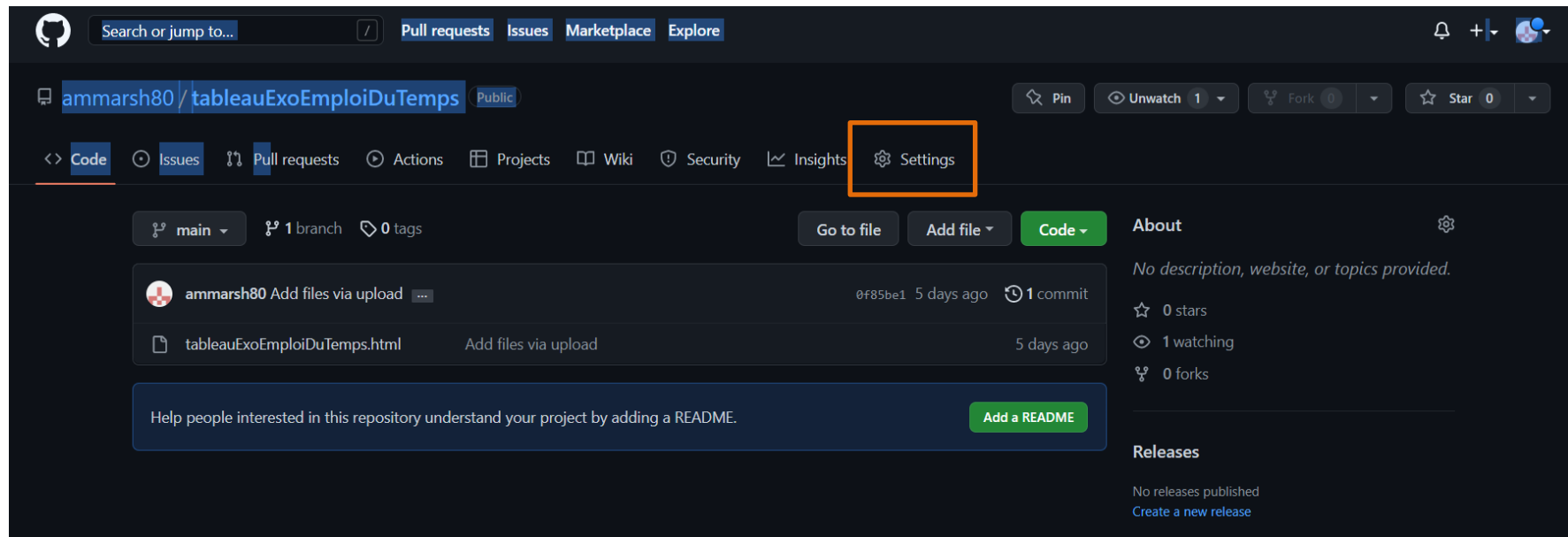
Au même encore l'option **README** (pas obligatoire, et on peut le faire plus tard à n'importe quel moment) qui est un petit fichier contenant la documentation de base (c'est-à-dire : comment récupérer le projet, comment il fonctionne et comment il est structuré). Une fois que toutes ces infos seront remplies, on clique **Create repository** et le dépôt apparaîtra dans (Your repositories).

The screenshot shows the GitHub interface for user Ammar SHIHAN (ammash80). The browser address bar is highlighted with an orange box, showing the URL 'github.com/ammash80?tab=repositories'. The user's profile is on the left, and the 'Repositories' tab is selected, showing three public repositories: 'tableauExoEmploiDuTemps', 'tableauExoPlanning', and 'tableauExotableauFusion'. A dropdown menu is open on the right, showing options like 'Your repositories', 'Your codespaces', 'Your projects', etc.



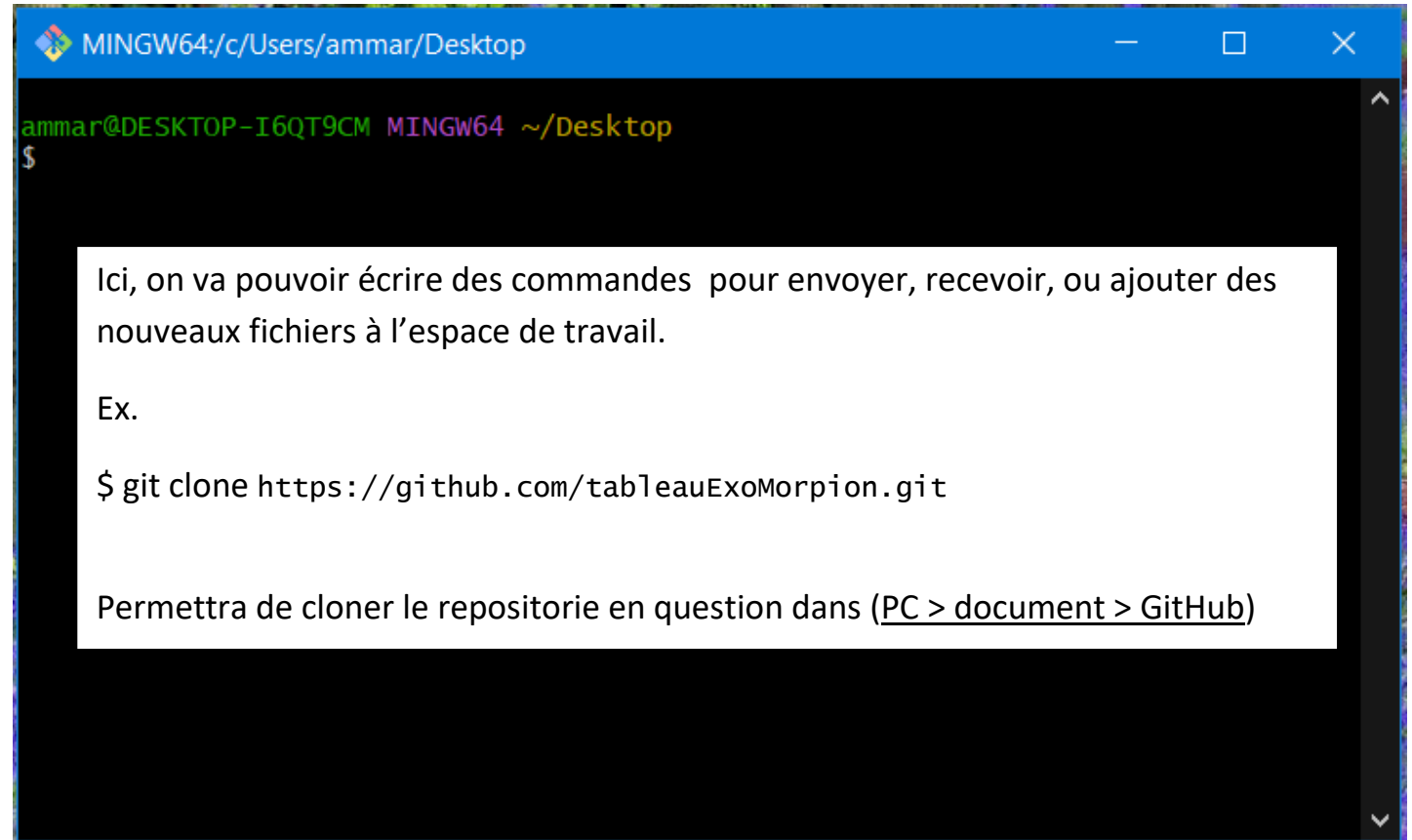
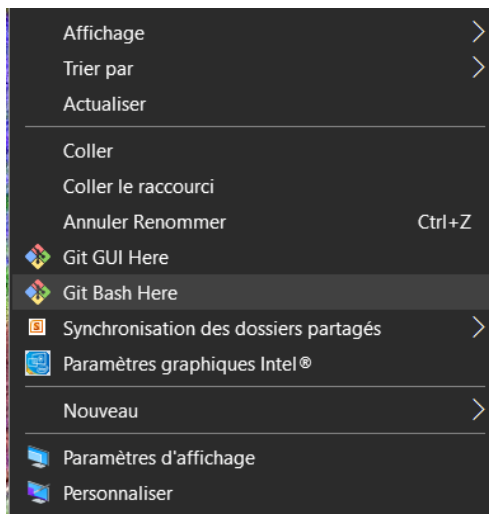
Pour partager un seul repository : copier partager ce lien (dans code).





Le dépôt est en ligne pour le moment, il faut donc en créer en copie en local (sur son ordinateur).

Pour cela (comme nous avons déjà installer Git 😊),



Dans le dossier local, chaque un pourra travailler, créer de nouveaux fichiers etc...

MAIS, pour sauvegarder et partager les changements, il faut retourner dans le terminal :

Se rendre dans le dossier (commande : `cd` qui permet de changer de répertoire et ensuite spécifier le nom de dépôt :

```
cd nomDeRepository
```

Ensuite avec la commande : (pour envoyer le/les fichier ajouté)

```
git add nomDeFichier.html
```

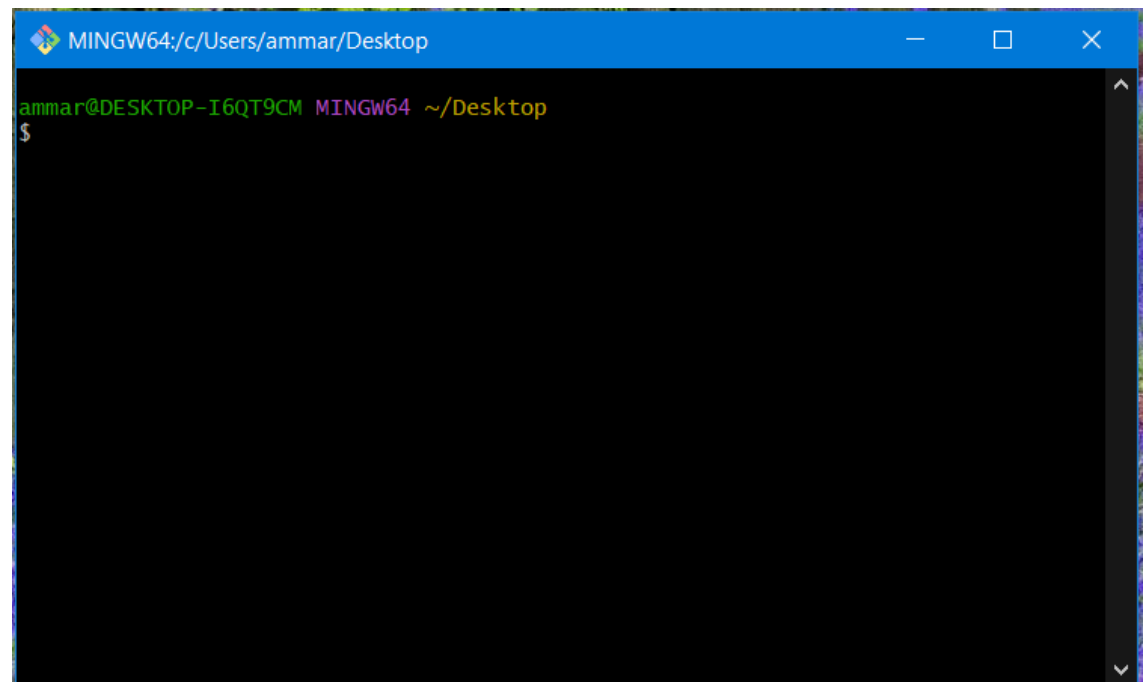
Encore mieux : (pour envoyer tous les changements 😊)

```
git add *
```

Attention tous cela permet de créer une liste temporaire en attente d'envoi !! (et ce n'est pas suffisant).

Avec

```
add commit
```



On va pouvoir devoir impacter tous les changements dans un (**commit**) : une sorte de boite qui va contenir tous les changements correspondant à une **version** (ex. ajout d'un nouveau code, suppression, modification d'une ligne, etc...). Ce pack (commit) on va pouvoir l'envoyer en ligne grâce à la commande :

```
git push
```

qui permet l'envoi de commit sur le dépôt en ligne cette voici !!!!

à partir de là, on a une copie en local + en copie sur le dépôt et on peut se déconnecter car tout est sauvegardés en ligne.

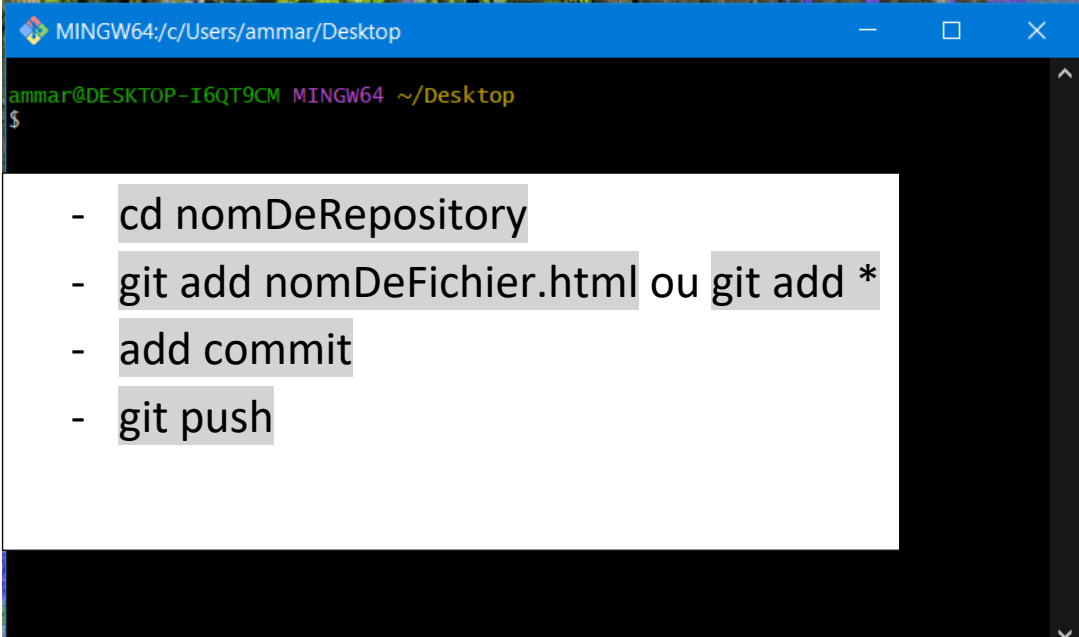
A chaque envoie, GitHub crée une nouvelle version actualisée

Pour récupérer les nouveautés, le collaborateur devra faire la commande :

```
git pull
```

et à son tour il recommence la procédure....

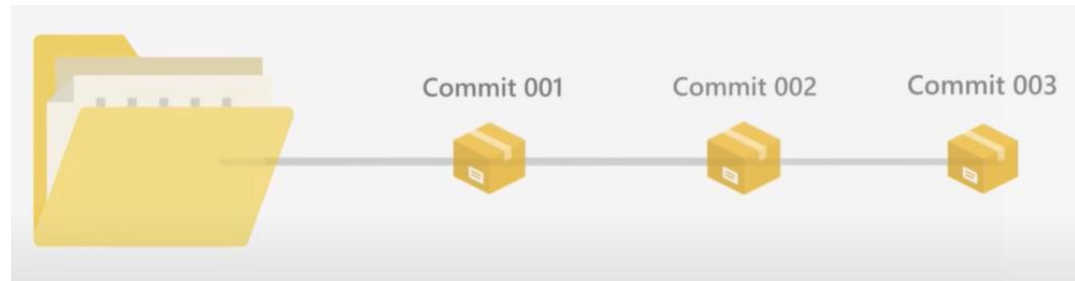
- Travailler, apporter des modifications....etc.
- Ensuite écrire les commandes dans le terminal →



The screenshot shows a terminal window titled 'MINGW64:/c/Users/ammam/Desktop'. The prompt is 'ammam@DESKTOP-I6QT9CM MINGW64 ~/Desktop \$'. A list of commands is displayed in a white box with a black background, each preceded by a hyphen and the command text is highlighted in grey:

- `cd nomDeRepository`
- `git add nomDeFichier.html` ou `git add *`
- `add commit`
- `git push`

Une Branche

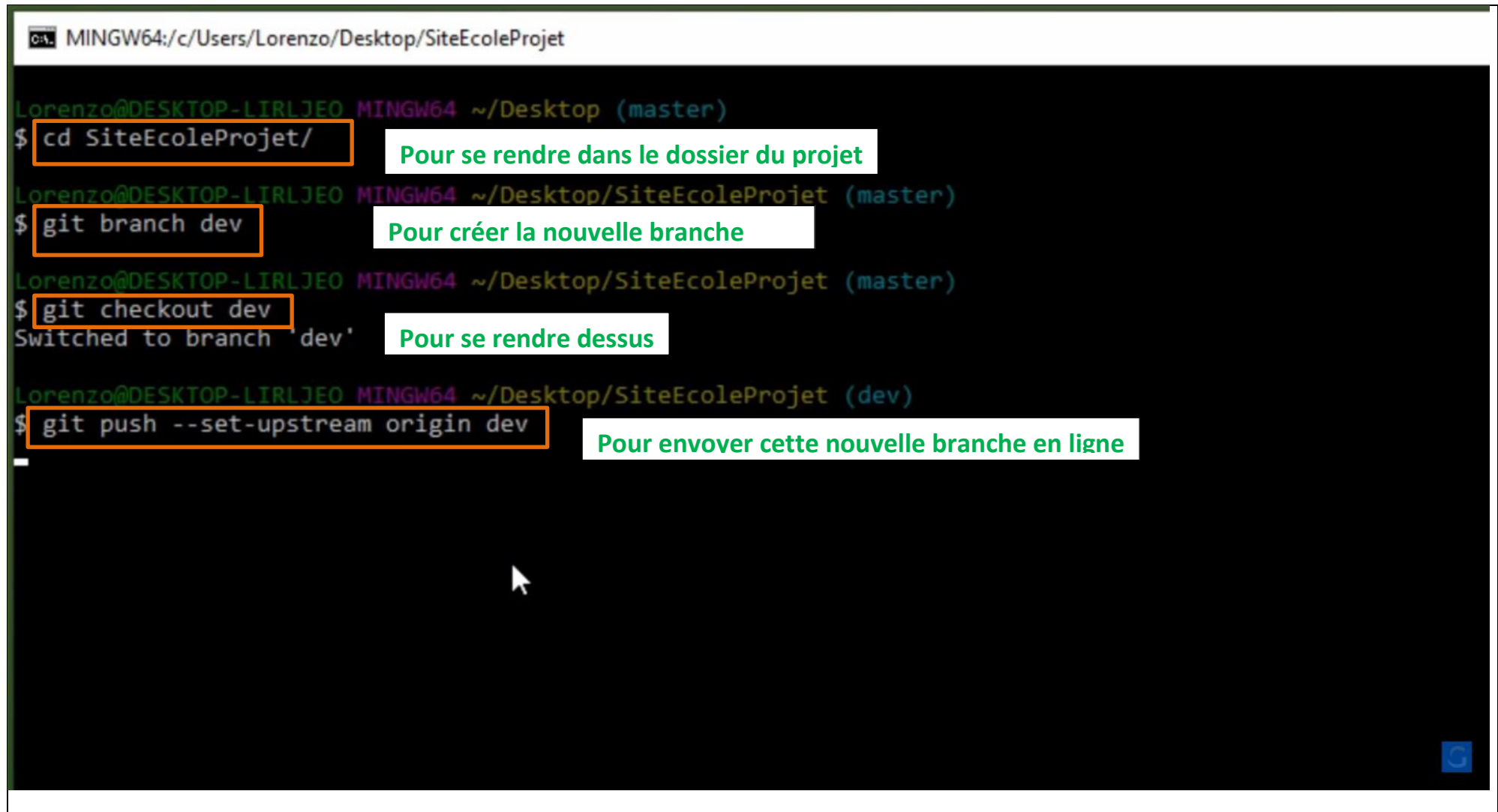


La branche générée à la création du projet : **Branche master**
Et surtout, elle serait la branche finale de projet !!!
Au début donc, il ne faut rien mettre dessus tant que le projet n'est pas fini



Il faudra donc s'organiser pour créer une autre branche pour chaque partie :

Ex. Dans le projet (SiteEcoleProjet) on va créer une branche (dev) pour la partie développement :



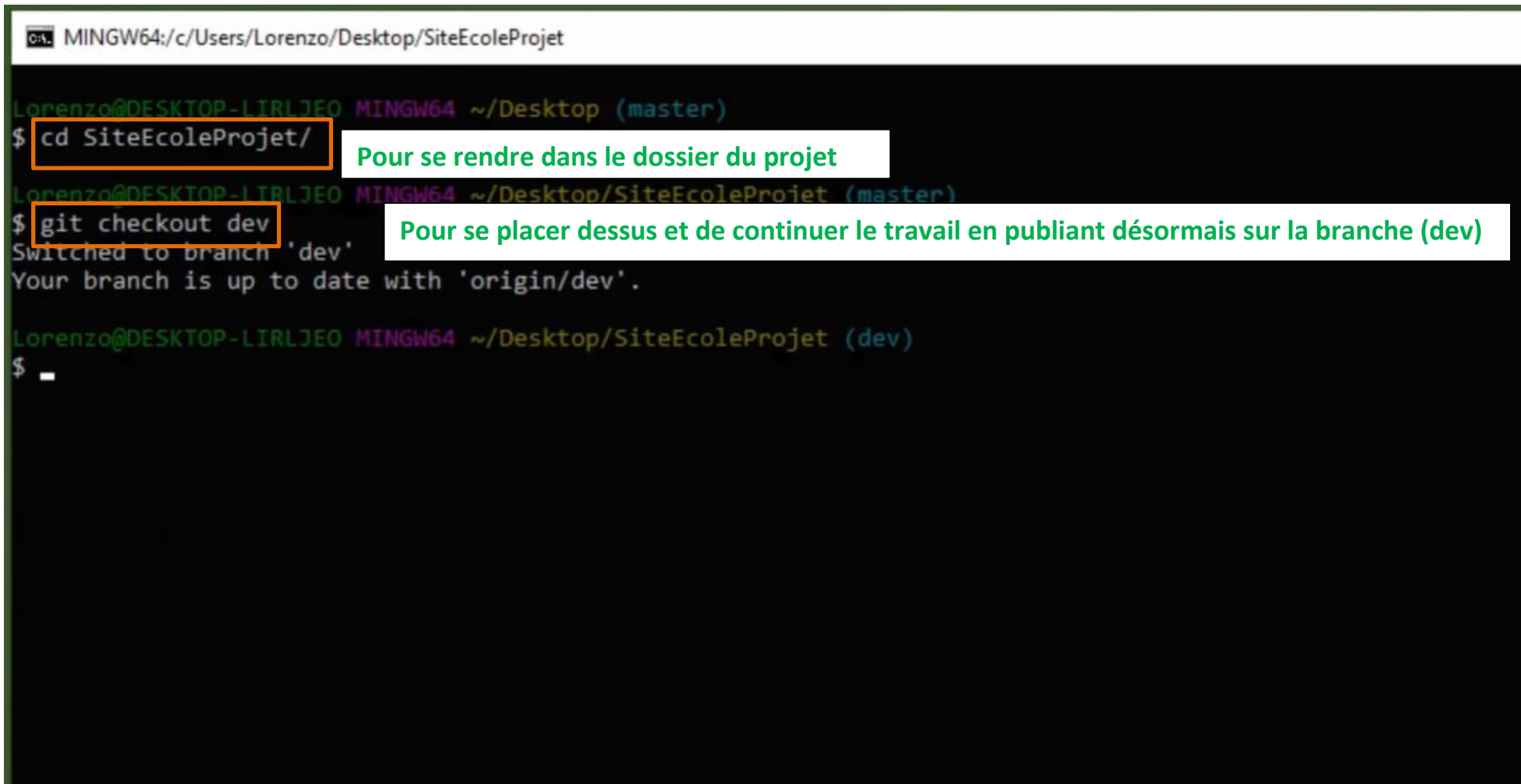
A terminal window titled 'MINGW64:/c/Users/Lorenzo/Desktop/SiteEcoleProjet' showing a series of Git commands and their outputs. The commands are: 'cd SiteEcoleProjet/', 'git branch dev', 'git checkout dev', and 'git push --set-upstream origin dev'. Each command is highlighted with an orange box, and a green callout box explains its purpose: 'Pour se rendre dans le dossier du projet', 'Pour créer la nouvelle branche', 'Pour se rendre dessus', and 'Pour envoyer cette nouvelle branche en ligne' respectively. The terminal output shows the user switching to the 'dev' branch and successfully pushing it to the remote repository.

```
MINGW64:/c/Users/Lorenzo/Desktop/SiteEcoleProjet

Lorenzo@DESKTOP-LIRLJEO MINGW64 ~/Desktop (master)
$ cd SiteEcoleProjet/
Lorenzo@DESKTOP-LIRLJEO MINGW64 ~/Desktop/SiteEcoleProjet (master)
$ git branch dev
Lorenzo@DESKTOP-LIRLJEO MINGW64 ~/Desktop/SiteEcoleProjet (master)
$ git checkout dev
Switched to branch 'dev'
Lorenzo@DESKTOP-LIRLJEO MINGW64 ~/Desktop/SiteEcoleProjet (dev)
$ git push --set-upstream origin dev
```

La branche est créée.

Pour l'absorber (par un collaborateur) :



A terminal window titled 'MINGW64:/c/Users/Lorenzo/Desktop/SiteEcoleProjet' showing a series of Git commands and their outputs. The user is in the 'master' branch. The first command is 'cd SiteEcoleProjet/' which is highlighted with an orange box. A green callout box points to it with the text 'Pour se rendre dans le dossier du projet'. The second command is 'git checkout dev' which is also highlighted with an orange box. A green callout box points to it with the text 'Pour se placer dessus et de continuer le travail en publiant désormais sur la branche (dev)'. The output shows the branch has been switched to 'dev' and is up to date with 'origin/dev'. The prompt then changes to indicate the current branch is 'dev'.

```
MINGW64:/c/Users/Lorenzo/Desktop/SiteEcoleProjet

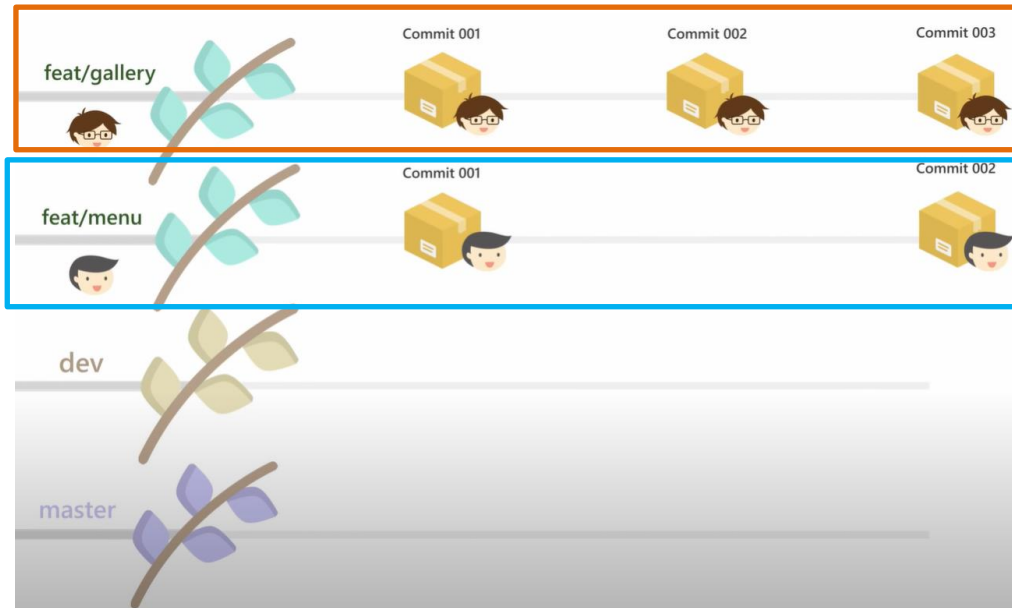
Lorenzo@DESKTOP-LIRLJEO MINGW64 ~/Desktop (master)
$ cd SiteEcoleProjet/
Lorenzo@DESKTOP-LIRLJEO MINGW64 ~/Desktop/SiteEcoleProjet (master)
$ git checkout dev
Switched to branch 'dev'
Your branch is up to date with 'origin/dev'.

Lorenzo@DESKTOP-LIRLJEO MINGW64 ~/Desktop/SiteEcoleProjet (dev)
$ _
```

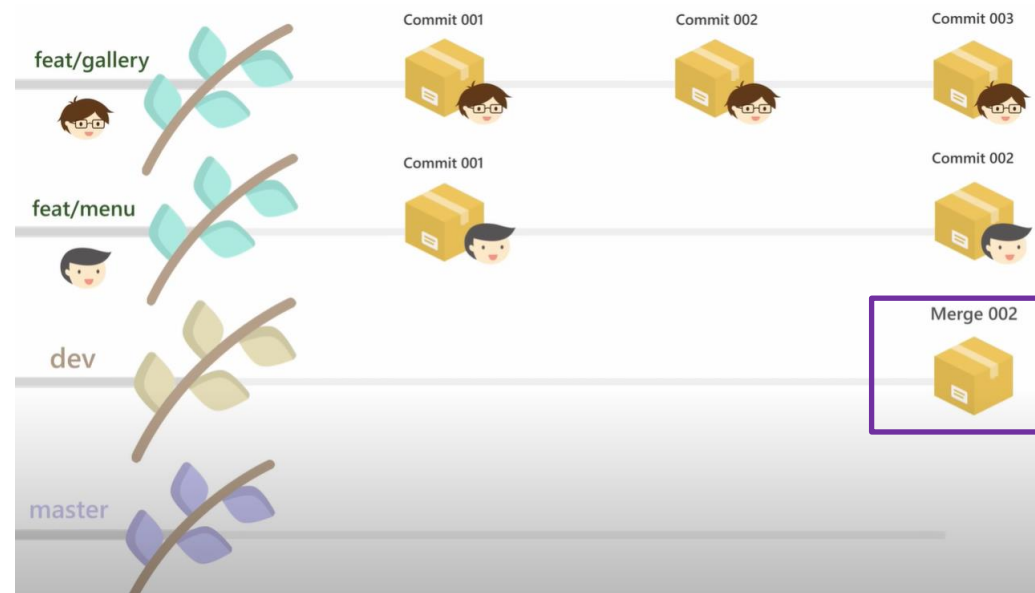
Il va falloir diviser la branche par fonctionnalité :

- 1) Galerie photos..,
- 2) menu de navigation.

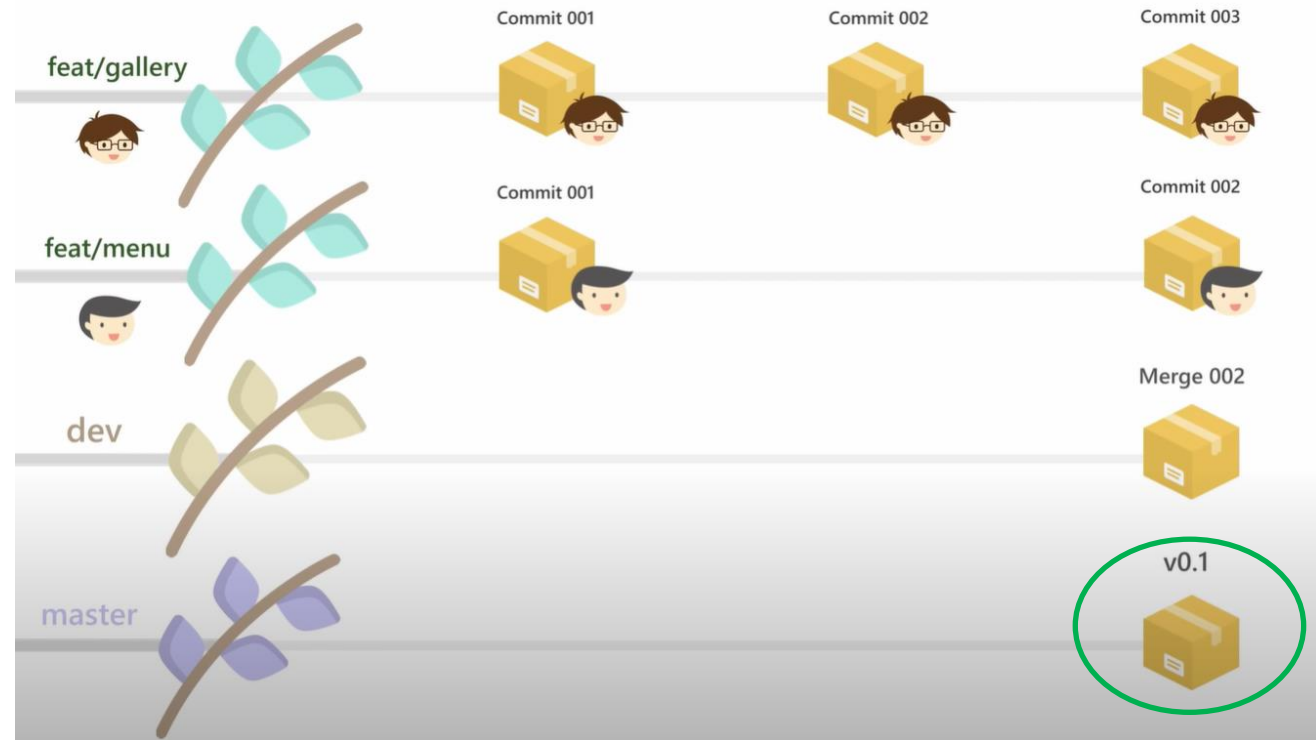
Chaque 'un donc travail sur sa propre branche de fonctionnalité



Avec la commande `git merge`
On va fusionner les deux branches (commit) de fonctionnalité.



Avec encore, la commande
`git merge`
cette fois pour envoyer le
projet de la branche dev à la
branche master



Fin du projet