

Tutoriel Git / GitHub

Versionner Son Travail

Table des matières

Naviguer dans Git Bash :	2
pwd : savoir dans quel dossier on se trouve.....	2
cd : aller dans le dossier (cd : Change Directory).....	2
mkdir « dossier » : créer un dossier (mkdir : Make Directory).	2
touch : créer un fichier.....	3
ls : liste le dossier courant	3
cd .. : remonter d'un dossier.....	3
Versionner en local :	4
git init : initialise le dépôt (ATTENTION : se mettre sur le bon dossier), mieux à faire depuis GitHub.com	4
git add . : ajoute toutes les modifications (le symbole « . » symbolise tout)	5
git commit -m « explication » : crée un nouveau commit.....	6
Gérer les commits :	7
git log : liste les commits.....	7
git show sha-1 : voir un commit spécifique en récupérant le sha-1.....	8
git checkout sha-1 : remettre la version d'un commit précédent à l'aide du sha-1.....	9
git checkout master : remettre la version la plus récente de notre commit.....	10
Partager les productions en ligne :	11
Versionner sur un dépôt distant :	12
git clone lien-github.com : récupérer notre travail depuis un dépôt distant.	12
git push -u origin main : pousse les modifications réalisées vers le serveur.	13
Annexe : GitHub Cheat Sheet	16

Naviguer dans Git Bash :

pwd : savoir dans quel dossier on se trouve.

```
romai@DESKTOP-8Q08HON MINGW64 ~  
$ pwd  
/c/Users/romai
```

Dans cet exemple, nous nous trouvons au niveau du disque « c » de l'ordinateur, dans la partie « Users » (« Utilisateurs ») et le dossier « romai ».

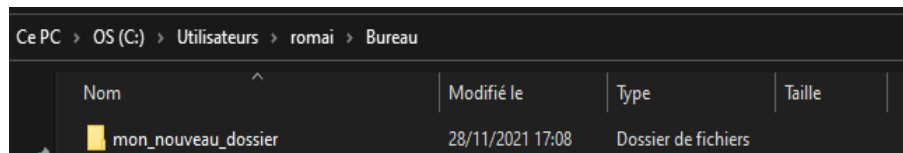
cd : aller dans le dossier (cd : Change Directory).

```
romai@DESKTOP-8Q08HON MINGW64 ~  
$ cd desktop  
  
romai@DESKTOP-8Q08HON MINGW64 ~/desktop  
$ |
```

En tapant la commande « cd desktop » (« desktop » correspond au « Bureau » sur votre ordinateur) dans Git Bash, on se déplace dans le dossier « desktop ». A partir de ce moment, toutes les lignes de commande réalisées se feront dans le dossier « desktop » (comme l'indique le texte en jaune sur la capture d'écran au-dessus).

mkdir « dossier » : créer un dossier (mkdir : Make Directory).

```
romai@DESKTOP-8Q08HON MINGW64 ~/desktop  
$ mkdir mon_nouveau_dossier
```



Dans un premier temps, on tape la ligne de commande « mkdir » suivi d'un espace et du nom du fichier que l'on veut créer. Ici le dossier créé s'appelle « mon_nouveau_dossier ». On appuie sur Entrée.


Dans un second temps, grâce à cette ligne de commande, un nouveau dossier du nom de « mon_nouveau_dossier » sera créé sur mon Bureau (voir l'arborescence sur la photo précédente).

Pour entrer dans ce nouveau dossier, on utilisera la commande « cd mon_nouveau_dossier ». Pour vérifier que nous nous trouvons bien dans notre dossier « mon_nouveau_dossier », il suffit de regarder le texte affiché en jaune sur Git Bash.

```
romai@DESKTOP-8Q08HON MINGW64 ~/desktop  
$ cd mon_nouveau_dossier  
  
romai@DESKTOP-8Q08HON MINGW64 ~/desktop/mon_nouveau_dossier  
$ ..
```

touch : créer un fichier

```
romai@DESKTOP-8Q08H0N MINGW64 ~/desktop/mon_nouveau_dossier
$ touch test.txt
```

Ce PC > OS (C:) > Utilisateurs > romai > Bureau > mon_nouveau_dossier		
Nom	Modifié le	Type
 test.txt	28/11/2021 18:15	Document texte

En inscrivant la commande « touch test.txt », nous créons dans le dossier « mon_nouveau_dossier » un fichier nommé « test.txt ». Nous pouvons vérifier sur notre ordinateur que ce fichier « test.txt » a bien été créé dans le dossier « mon_nouveau_dossier ». Un autre moyen de vérifier est l'utilisation de la commande « ls ».

ls : liste le dossier courant

```
romai@DESKTOP-8Q08H0N MINGW64 ~/desktop/mon_nouveau_dossier
$ ls
test.txt
```

En tapant la commande « ls », on voit s'afficher à l'écran l'ensemble des fichiers que contient le dossier dans lequel nous nous trouvons. Dans notre exemple, nous n'avons que le fichier « test.txt » pour le moment dans le dossier « mon_nouveau_dossier ».

cd .. : remonter d'un dossier

```
romai@DESKTOP-8Q08H0N MINGW64 ~/desktop/mon_nouveau_dossier
$ cd ..

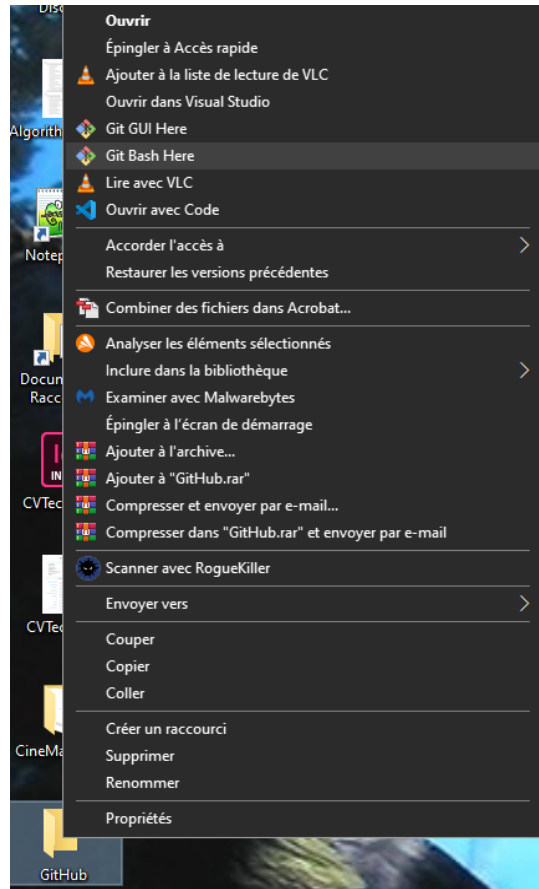
romai@DESKTOP-8Q08H0N MINGW64 ~/desktop
$
```

En utilisant la commande « cd .. », nous revenons au dossier précédent dans l'arborescence des dossiers. Ainsi, dans notre exemple, le fait de taper la commande « cd .. » nous ramène au niveau de notre « Desktop » (« Bureau »).

Versionner en local :

Petite Astuce : Pour lancer Git Bash au niveau du dossier souhaité, il suffit de faire un clic droit de la souris sur le dossier choisi et de choisir dans le panneau de commande l'option « Git Bash Here ». Vous ouvrirez ainsi le dossier souhaité avec Git Bash.

Dans notre exemple, on souhaite ouvrir le dossier « GitHub » avec Git Bash. On réalise un clic droit sur ce dossier et choisissons l'option « Git Bash Here ». Git Bash se lance et on se trouve directement dans notre dossier « GitHub », comme le montre les images suivantes.



```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub
$
```

git init : initialise le dépôt (ATTENTION : se mettre sur le bon dossier), mieux à faire depuis GitHub.com

On tape dans Git Bash la commande « git init ». A la suite de cette commande, dans le dossier dans lequel nous nous trouvons (à savoir dans notre exemple « GitHub »), un dossier « .git » va s'initialiser.

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub
$ git init
Initialized empty Git repository in C:/Users/romai/OneDrive/Bureau/GitHub/.git/
```

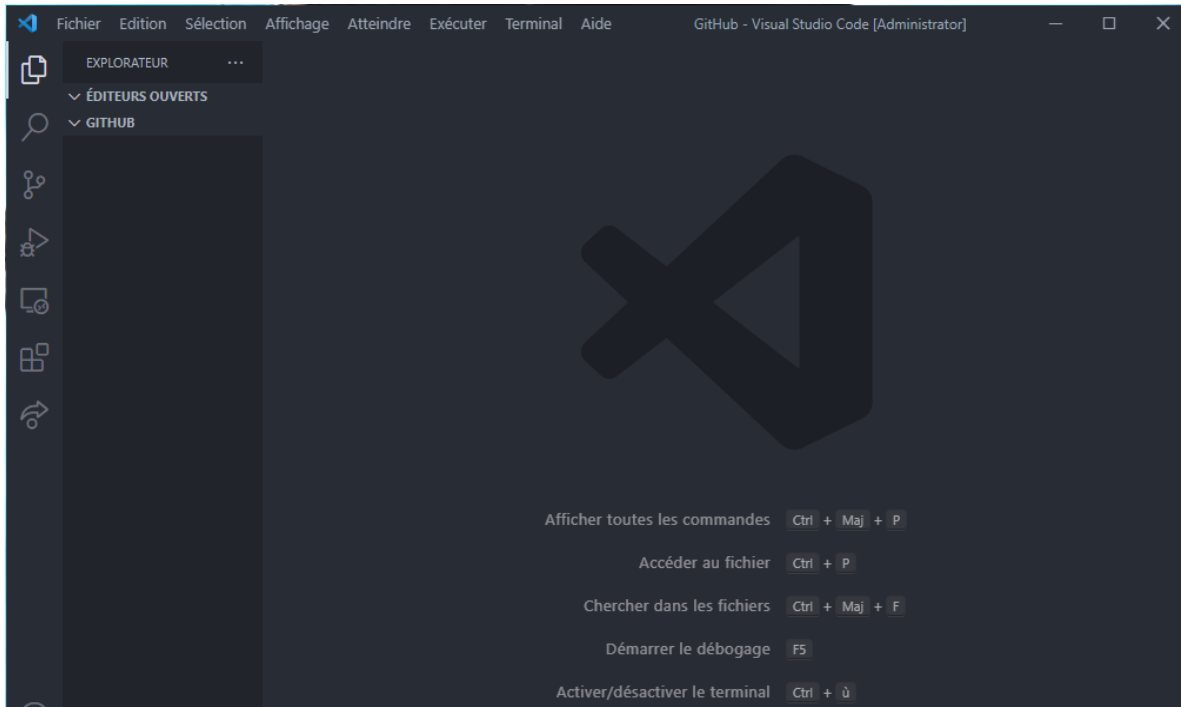
GitHub				
	Nom	Statut	Modifié le	Type
	.git		28/11/2021 21:22	Dossier de fichiers

git add . : ajoute toutes les modifications (le symbole « . » symbolise tout)

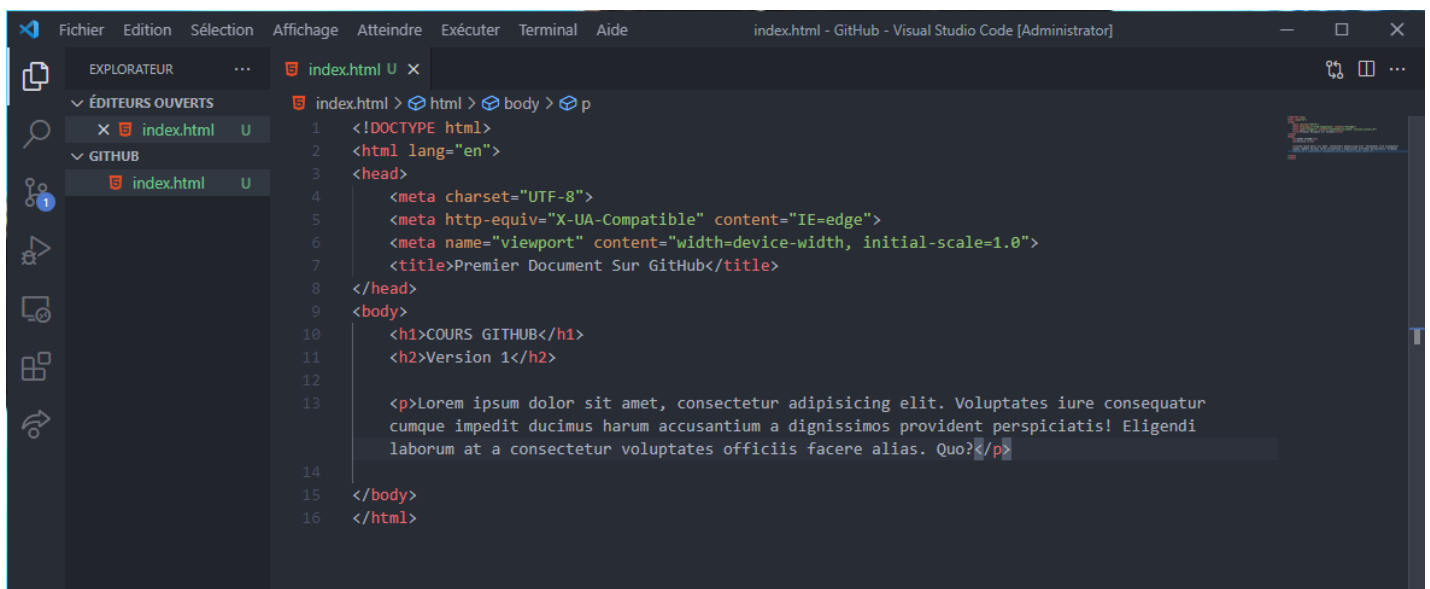
Après avoir utilisée la commande « git init », nous pouvons taper la commande « code . » pour ouvrir notre logiciel d'éditeur de code (ici, notre éditeur de code est défini sur Visual Studio Code). Nous nous retrouvons ainsi à cette étape :

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub
$ git init
Initialized empty Git repository in C:/Users/romai/OneDrive/Bureau/GitHub/.git/

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub (master)
$ code .
```



Nous allons maintenant créer un fichier HTML simple que nous allons nommer « index.html » dans notre dossier « GitHub ». Dans le fichier « index.html » on va écrire une petite portion de code en HTML comme dans l'exemple ci-dessous :



On veut maintenant sauvegarder tout notre travail (à savoir le code HTML5 compris dans le fichier « index.html »). Pour cela, nous allons taper dans Git Bash la commande « git add . » (ATTENTION : il y a un espace entre « add » et « . »). La commande « git add . » met dans une zone d'index la sauvegarde de notre travail sur le fichier « index.html ». Mais nous n'avons toujours pas commité notre travail. Ainsi, pour vérifier cette affirmation, nous pouvons taper la commande « git status » pour en avoir le cœur net. Nous nous retrouvons avec l'image suivante :

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub (master)
$ git add .

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub (master)
$ git status
On branch master

No commits yet

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

Nous nous trouvons face à plusieurs informations ici :

1. Nous nous situons sur la branche « master » ou « main » (« On branch master »)
2. Aucun commit n'a encore été réalisé (« No commits yet»). Pour rappel, notre travail se trouve en zone d'index : il est sauvegardé sur notre ordinateur mais pas encore commité.
3. C'est à partir de ce moment là qu'il faut réaliser ou pas un commit. On voit sur l'illustration précédente que notre sauvegarde de travail sur le fichier « index.html » est bien pris en compte (« Changes to be committed : new file : index.html »). La commande « git status » nous montre dans quel état se trouve notre fichier « index.html ». Dans notre exemple, ce fichier est en attente de commit.

git commit -m « explication » : crée un nouveau commit

Nous avons vu que la commande « git add . » pousse notre fichier « index.html » en zone d'index. C'est à ce moment que la commande « git commit » entre en jeu pour sauvegarder réellement notre travail dans un nouveau commit. Dans Git Bash, nous allons donc taper la commande « git commit -m « Première version du document index.html » » pour sauvegarder notre travail. Le commentaire que nous écrivons entre guillemets nous permettra de préciser le commit que nous allons réaliser mais aussi à retrouver par la suite cette sauvegarde parmi plusieurs qui pourront être réalisées par la suite.

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub (master)
$ git commit -m "Première version du document index.html"
[master (root-commit) 2fc9d00] Première version du document index.html
1 file changed, 16 insertions(+)
create mode 100644 index.html
```

Mais comment pouvons-nous à ce stade vérifier que notre travail a bien été sauvegardé lors du commit ? Grâce à la commande « git log ».

Gérer les commits :

git log : liste les commits

En tapant cette commande, nous pourrions voir tous les commits réalisés. Dans notre exemple, en tapant « git log », nous voyons plusieurs informations s'afficher dans Git Bash :

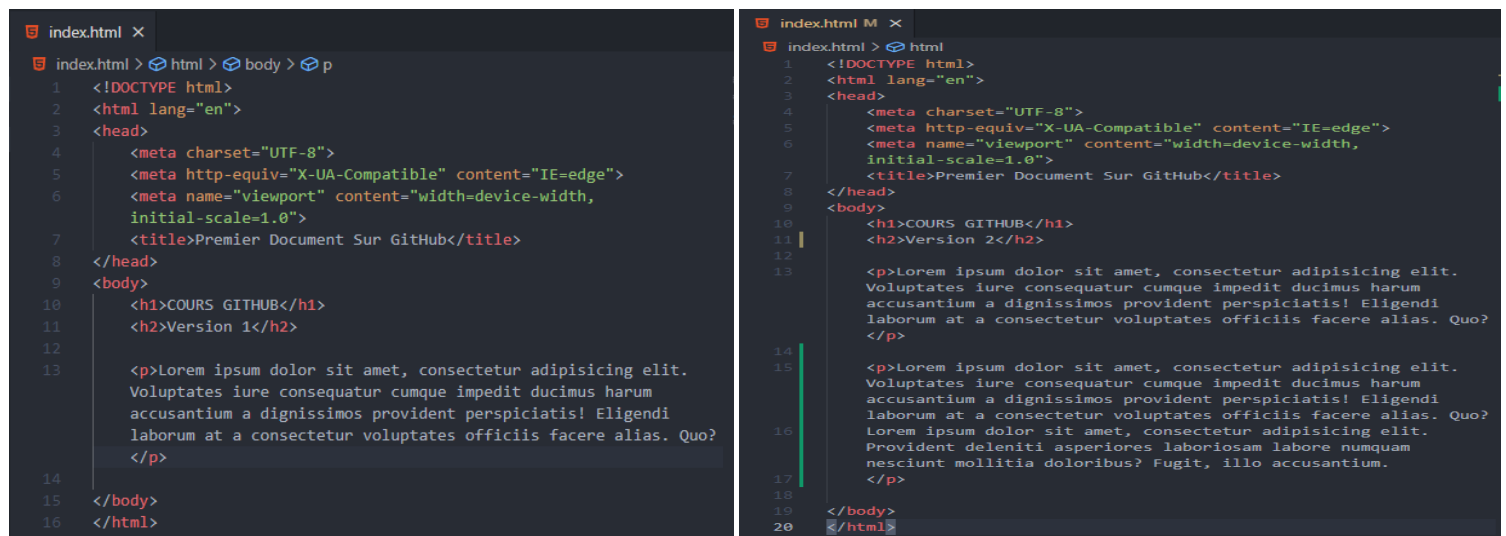
```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub (master)
$ git log
commit 2fc9d002b839446cc0d967eb991130079e5e48f4 (HEAD -> master)
Author: RmNLCN06 <63708684+RmNLCN06@users.noreply.github.com>
Date: Sun Nov 28 22:12:58 2021 +0100

Première version du document index.html
```

1. Nous avons tout d'abord la clé d'identification de notre commit : c'est le SHA-1 (pour Secure Hash Algorithm : de manière simplifiée, c'est une fonction de hachage cryptographique produisant un résultat appelé un hash, de 160 bits (20 octets), habituellement représenté par un nombre hexadécimal de 40 caractères).
2. On retrouve ensuite l'auteur du commit.
3. La date et l'heure apparaissent ensuite en dessous de l'auteur du commit.
4. Enfin nous retrouvons le commentaire que nous avons écrit.

Si un jour, après de multiples commits nous avons besoin de revenir au tout premier commit réalisé, il suffit de récupérer la clé d'identification SHA-1 (à savoir : « 2fc9d002b839446cc0d967eb991130079e5e48f4 »).

Pour démontrer l'exemple précédent, nous allons réaliser des modifications dans notre fichier « index.html ». Nous passons donc de notre « Version 1 » du fichier « index.html » à une « Version 2 » avec des modifications :



```
index.html x
index.html > html > body > p
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width,
7     initial-scale=1.0">
8   <title>Premier Document Sur GitHub</title>
9 </head>
10 <body>
11   <h1>COURS GITHUB</h1>
12   <h2>Version 1</h2>
13   <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.
14     Voluptates iure consequatur cumque impedit ducimus harum
15     accusantium a dignissimos provident perspiciatis! Eligendi
16     laborum at a consectetur voluptates officiis facere alias. Quo?
17   </p>
18 </body>
19 </html>
```

```
index.html M x
index.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width,
7     initial-scale=1.0">
8   <title>Premier Document Sur GitHub</title>
9 </head>
10 <body>
11   <h1>COURS GITHUB</h1>
12   <h2>Version 2</h2>
13   <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.
14     Voluptates iure consequatur cumque impedit ducimus harum
15     accusantium a dignissimos provident perspiciatis! Eligendi
16     laborum at a consectetur voluptates officiis facere alias. Quo?
17   </p>
18   <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.
19     Voluptates iure consequatur cumque impedit ducimus harum
20     accusantium a dignissimos provident perspiciatis! Eligendi
21     laborum at a consectetur voluptates officiis facere alias. Quo?
22   </p>
23 </body>
24 </html>
```

Nous allons créer une sauvegarde de nos modifications sur le fichier « index.html ».

Nous n'avons pas besoin de taper de nouveau la commande « git init » car notre dépôt est déjà initialisé. Pour créer une sauvegarde, nous tapons la commande « git add ». Ainsi, toutes les modifications réalisées sur notre fichier « index.html » sont poussées en zone d'index. Nous pouvons vérifier si l'étape précédente a bien été réalisée en tapant « git status » :

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub (master)
$ git add .

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html
```

On voit ainsi sur la capture précédente que notre fichier « index.html » a été modifié (« modified : index.html »). Nous avons donc cette modification en zone d'index pour le moment. Pour sauvegarder réellement dans un nouveau commit, on tape « git commit -m « Version 2 du document index.html avec ajout d'un paragraphe » ».

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub (master)
$ git commit -m "Version 2 du document index.html avec ajout d'un paragraphe"
[master b922e6a] Version 2 du document index.html avec ajout d'un paragraphe
1 file changed, 5 insertions(+), 1 deletion(-)
```

Plusieurs informations apparaissent :

1. « 1 file changed » : cela correspond aux modifications apportées sur notre fichier « index.html ». Si par exemple nous avons modifié deux fichiers, nous aurions eu le texte « 2 file changed »,
2. « 5 insertions » : cela correspond aux nombres d'ajouts réalisés sur notre fichier « index.html ». Dans notre cas, c'est l'ajout du paragraphe supplémentaire.
3. « 1 deletion » : cela correspond aux nombres de suppressions dans le fichier « index.html ». Dans notre cas, c'est le fait d'avoir supprimé le « 1 » dans notre titre « h2 » et de l'avoir remplacé par « 2 ».

Si nous faisons un « git log », nous allons voir tous les commits réalisés sur ce fichier « index.html » jusqu'à maintenant :

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub (master)
$ git log
commit b922e6a88bb8e6f06414eaf7b3de16bb8b12f4be (HEAD -> master)
Author: RmNLcN06 <63708684+RmNLcN06@users.noreply.github.com>
Date: Fri Dec 3 17:28:27 2021 +0100

    Version 2 du document index.html avec ajout d'un paragraphe

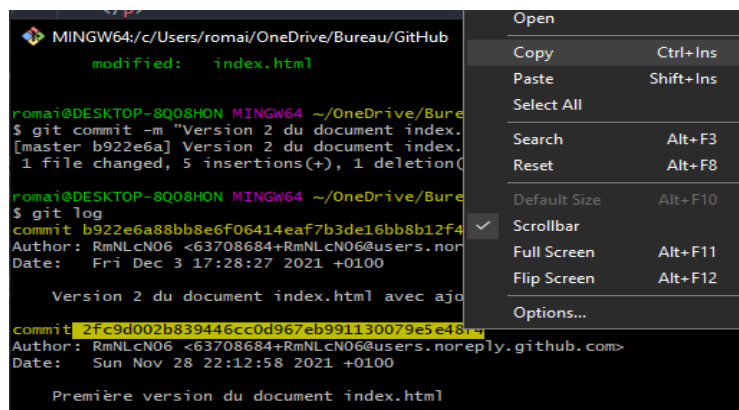
commit 2fc9d002b839446cc0d967eb991130079e5e48f4
Author: RmNLcN06 <63708684+RmNLcN06@users.noreply.github.com>
Date: Sun Nov 28 22:12:58 2021 +0100

    Première version du document index.html
```

Nous avons donc finalement deux commits : la « Version 2 » avec plus de texte et la « Version 1 ». Notre branche « master » (ou « main ») se trouve sur la « Version 2 » car on se trouve toujours sur le commit le plus récemment réalisé.

git show sha-1 : voir un commit spécifique en récupérant le sha-1

Imaginons maintenant que nous voulons voir ce que contenait notre premier commit. Pour cela, nous allons sélectionner la clé d'identification sha-1 du premier commit, faire un clic droit et la copier :



Ensuite, nous tapons la commande « git show » et on appuie sur la molette de la souris pour coller la clé d'identification sha-1 copiée précédemment :


```

commit 2fc9d002b839446cc0d967eb991130079e5e48f4
Author: RmNLcN06 <63708684+RmNLcN06@users.noreply.github.com>
Date: Sun Nov 28 22:12:58 2021 +0100

    Première version du document index.html

diff --git a/index.html b/index.html
new file mode 100644
index 0000000..2d97f7e
--- /dev/null
+++ b/index.html
@@ -0,0 +1,16 @@
+<!DOCTYPE html>
+<html lang="en">
+<head>
+  <meta charset="UTF-8">
+  <meta http-equiv="X-UA-Compatible" content="IE=edge">
+  <meta name="viewport" content="width=device-width, initial-scale=1.0">
+  <title>Premier Document Sur GitHub</title>
+</head>
+<body>
+  <h1>COURS GITHUB</h1>
+  <h2>Version 1</h2>
+  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptates iure consequatur cumque impedit ducimus harum accusantium a dignissimos provident perspiciatis! Eligendi laborum at a consectetur voluptates officiis facere alias. Quo?</p>
+</body>
+</html>

```

Nous retrouvons ainsi le code que nous avions tapé lors de la « Version 1 » de notre fichier « index.html ».

git checkout sha-1 : remettre la version d'un commit précédent à l'aide du sha-1

Pour remettre une version antérieure d'un commit sur un fichier, on tape « git checkout » et on colle ensuite le sha-1 du commit que l'on veut récupérer. Dans notre exemple, on veut remettre la « Version 1 » de notre fichier « index.html ». Pour cela, on copie le sha-1 de notre « Version 1 », on tape « git checkout » et on colle le sha-1 en appuyant sur la molette de la souris. Nous avons le résultat suivant :

```

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub (master)
$ git checkout 2fc9d002b839446cc0d967eb991130079e5e48f4
Note: switching to '2fc9d002b839446cc0d967eb991130079e5e48f4'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 2fc9d00 Première version du document index.html
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub ((2fc9d00...))
$

```

En tapant cette commande, nous sommes revenus sur notre « Version 1 » du fichier « index.html ». De plus, nous remarquons que nous ne nous situons plus sur la branche « master » (ou « main ») mais sur une branche en lien avec notre « Version 1 ». Si nous jetons un œil maintenant à notre éditeur de texte ... :

```
index.html X
index.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width,
7     initial-scale=1.0">
8   <title>Premier Document Sur GitHub</title>
9 </head>
10 <body>
11   <h1>COURS GITHUB</h1>
12   <h2>Version 1</h2>
13   <p>Lorem ipsum dolor sit amet, consectetur adipisicing
14     elit. Voluptates iure consequatur cumque impedit ducimus
15     harum accusantium a dignissimos provident perspiciatis!
16     Eligendi laborum at a consectetur voluptates officiis
17     facere alias. Quo?</p>
18 </body>
19 </html>
```

... nous sommes revenus également à notre « Version 1 » du fichier « index.html » !

git checkout master : remettre la version la plus récente de notre commit

Si nous voulons maintenant récupérer notre « Version 2 » du fichier « index.html », il nous suffit de taper la commande « git checkout master » pour revenir à notre commit le plus récent :

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub ((2fc9d00...))
$ git checkout master
Previous HEAD position was 2fc9d00 Premier version du document index.html
Switched to branch 'master'

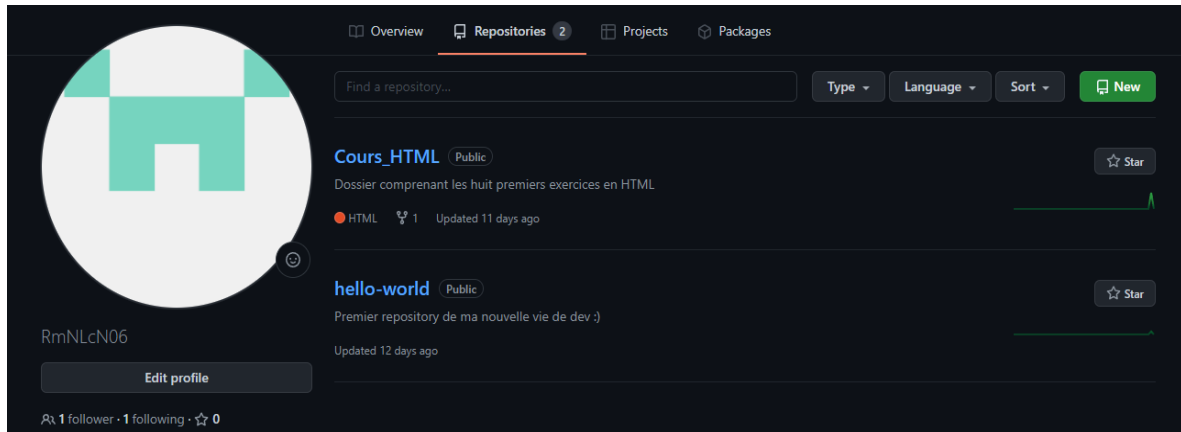
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/GitHub (master)
$ |
```

Nous voyons également que nous nous repositionnons au niveau de la branche « master » (ou « main »). Au niveau de notre éditeur de texte, nous avons également de nouveau notre « Version 2 » du fichier « index.html » :

```
index.html X
index.html > html > body > p
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width,
7     initial-scale=1.0">
8   <title>Premier Document Sur GitHub</title>
9 </head>
10 <body>
11   <h1>COURS GITHUB</h1>
12   <h2>Version 2</h2>
13   <p>Lorem ipsum dolor sit amet, consectetur adipisicing
14     elit. Voluptates iure consequatur cumque impedit ducimus
15     harum accusantium a dignissimos provident perspiciatis!
16     Eligendi laborum at a consectetur voluptates officiis
17     facere alias. Quo?</p>
18   <p>Lorem ipsum dolor sit amet, consectetur adipisicing
19     elit. Voluptates iure consequatur cumque impedit ducimus
20     harum accusantium a dignissimos provident perspiciatis!
21     Eligendi laborum at a consectetur voluptates officiis
22     facere alias. Quo?
23     Lorem ipsum dolor sit amet, consectetur adipisicing elit.
24     Provident deleniti asperiores laboriosam labore numquam
25     nesciunt mollitia doloribus? Fugit, illo accusantium.
26   </p>
27 </body>
28 </html>
```

Partager les productions en ligne :

Pour pouvoir partager ses productions, il faut tout d'abord sur votre espace GitHub créer un nouveau repository. Cliquez sur l'onglet « Repositories » pour à droite sur l'onglet « New » :



En arrivant sur la page suivante, il vous suffit d'écrire dans l'espace « Repository name » le nom que vous voulez donner à votre repository. Ici, on va appeler ce repository « github-tuto ». N'oubliez pas d'écrire une brève description de votre repository et de cocher la case « Add a README file » pour y ajouter un fichier README. Une fois ces différentes étapes réalisées, vous pouvez cliquer sur « Create repository » :

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

Owner * RmNLcN06 / **Repository name *** github-tuto ✓

Great repository names are short and memorable. Need inspiration? How about [super-duper-journey?](#)

Description (optional)
Repository de tutoriel sur la pratique de GitHub.

☒ **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.](#)

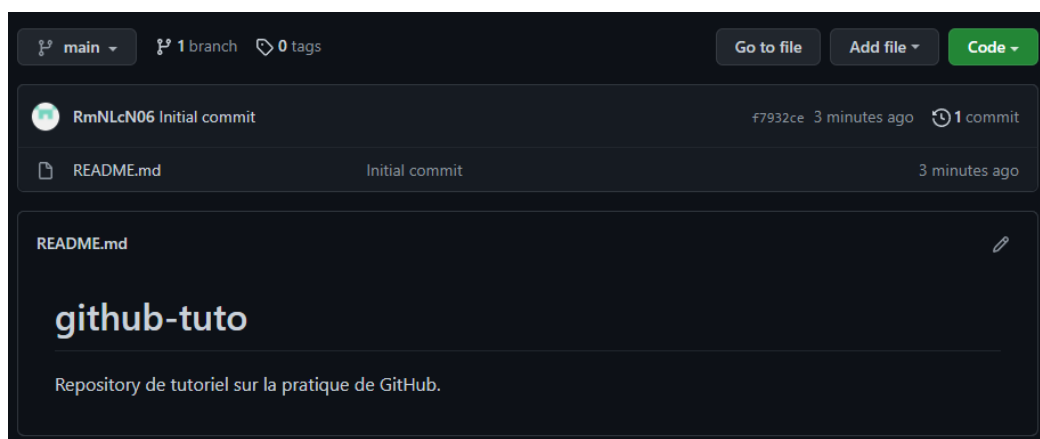
☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

Create repository

Le repository « github-tuto » est ainsi créé :

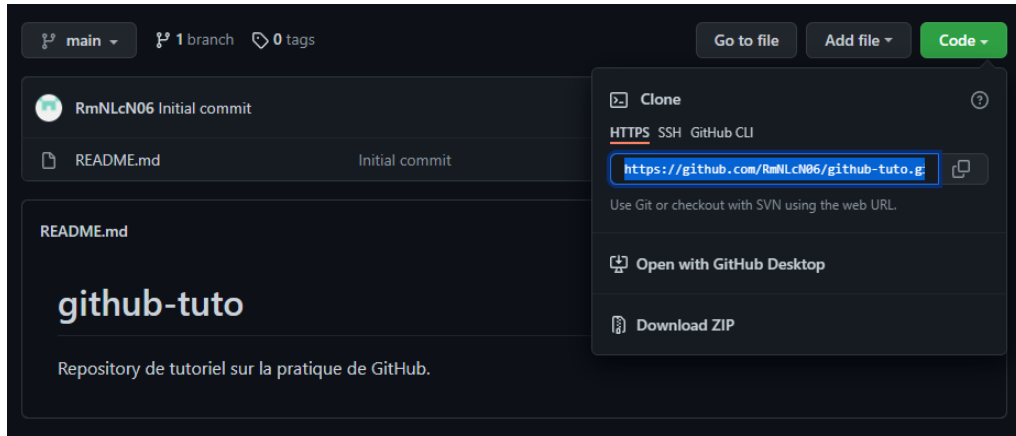


Nous allons voir maintenant comment versionner sur un dépôt distant.

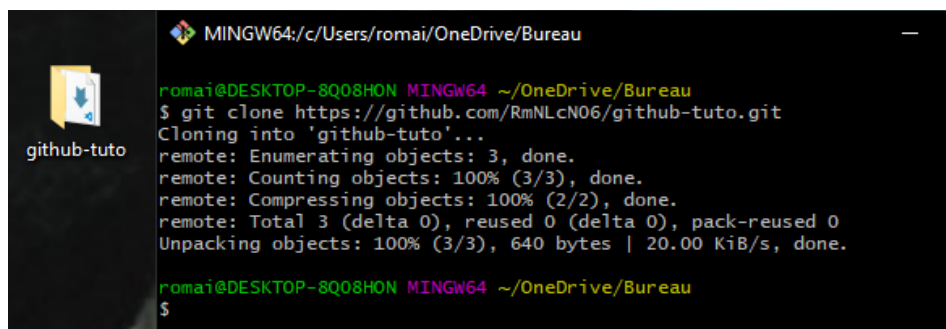
Versionner sur un dépôt distant :

git clone lien-github.com : récupérer notre travail depuis un dépôt distant.

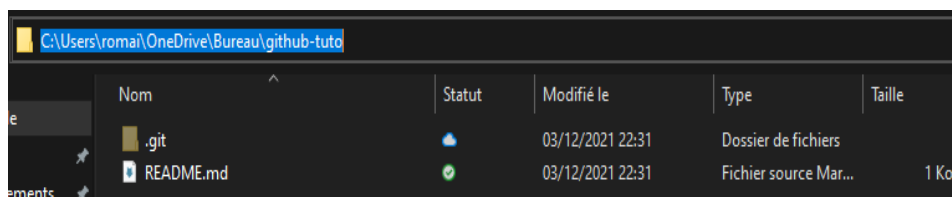
Pour travailler sur notre repository fraîchement créé, il va falloir le récupérer depuis le dépôt distant. Pour ce faire, sur notre exemple, on va aller cliquer sur l'onglet « Code » puis copier l'adresse HTTPS qui nous est donné :



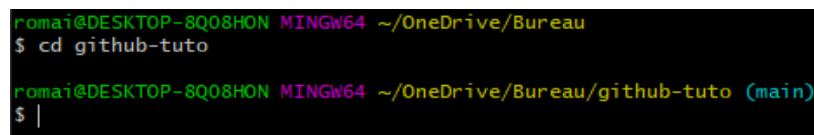
Une fois copiée, faites un clic droit sur votre Bureau ou dans l'emplacement où vous voulez stocker votre repository et choisissez l'onglet « Git Bash Here ». Une fenêtre de Git Bash devrait s'ouvrir. Taper ensuite la ligne de commande suivante : « git clone » et coller en appuyant la molette de la souris l'adresse HTTPS récupérée précédemment sur GitHub.com. A la suite de cette étape vous devriez voir sur votre Bureau que le repository « github-tuto » est apparu :



Pour le moment, ce dossier « github-tuto » ne comprend que le dossier masqué nommé « .git » et le fichier README.md. Si le dossier masqué n'apparaît pas, cliquez sur l'onglet « Affichage » puis cochez la case « Éléments masqués » :



Pour pouvoir travailler sur notre dossier, il va falloir y accéder. On utilise la commande « cd github-tuto » pour entrer dans notre dossier « github-tuto ». On se trouve ainsi sur la branche « main » (ou « master ») de notre dossier repository « github-tuto » :



A partir de ce point, nous allons créer un fichier « index.html » grâce à la commande « touch index.html ». Nous allons ouvrir ce fichier dans notre éditeur de texte et taper quelques lignes basiques de code en HTML comme le montre la photo suivante :

```
index.html U X
index.html > html > body > p
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Tuto GitHub ... et attention ça peut faire mal !</title>
8 </head>
9 <body>
10 <h1>Voilà un peu de texte pour remplir notre fichier</h1>
11 <p>
12   Lorem ipsum dolor sit, amet consectetur adipisicing elit. Vel rerum vero voluptas. Aut
    laboriosam quidem tempora, labore eaque explicabo nobis repellat, asperiores ea nostrum
    quaerat soluta, aliquam corrupti? Corporis temporibus saepe minima illo. Beatae voluptate
    consequuntur, odit quia quod temporibus error consequatur veritatis hic ab aut perferendis
    architecto illo voluptates adipisci amet nam, tempore officia excepturi eius. Impedit
    tempore, ut dolor saepe ipsa neque quos magni labore doloribus delectus laboriosam sapiente
    optio. Quia consequuntur facilis ad quibusdam ducimus quis, veritatis dolorum perspiciatis
    nemo unde animi, assumenda iusto quisquam laborum? Minus eligendi autem labore quisquam
    expedita vero nisi, consectetur, assumenda, corrupti dolor iure tempore tenetur fugit ea
    totam.
13 </p>
14 <p>
15   Praesentium voluptatem aliquam facilis quisquam quas recusandae beatae! Quos ducimus
    voluptate expedita dignissimos natus voluptates culpa temporibus quis error magnam aliquid
    ut, doloremque ad iusto quae praesentium voluptatibus veniam delectus! Quia aut modi
    molestiae, debitis harum itaque explicabo enim facilis totam alias provident, excepturi
    repellat eaque delectus, suscipit dolores. Explicabo suscipit sed qui aperiam ipsa eligendi
    expedita autem placeat mollitia adipisci. Sapiente at, ad voluptatum blanditiis itaque
    mollitia quisquam doloribus impedit, libero eos, molestias tenetur nobis nulla! Quasi ullam,
    quo dicta eum nobis sint quod ratione, inventore repudiandae enim minima voluptatum itaque
    expedita!
16 </p>
17 </body>
18 </html>
```

Une fois que nous avons fini de remplir avec quelques balises notre fichier « index.html », nous allons ajouter toutes les modifications réalisées en utilisant « git add . ». Comme d’habitude, rien ne nous empêche de vérifier si l’opération s’est bien déroulée en utilisant la commande « git status » :

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/github-tuto (main)
$ git add .

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/github-tuto (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   index.html
```

On observe que le fichier « index.html » a bien été poussé en zone d’index (« new file : index.html»). Nous allons maintenant utiliser la commande « git commit -m « Premier envoi index.html sur repository github-tuto » » pour sauvegarder réellement dans un nouveau commit. Pour rappel, à ce niveau, nous sommes toujours en local :

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/github-tuto (main)
$ git commit -m "Premier envoi index.html sur repository github-tuto"
[main e7df9a3] Premier envoi index.html sur repository github-tuto
1 file changed, 18 insertions(+)
create mode 100644 index.html
```

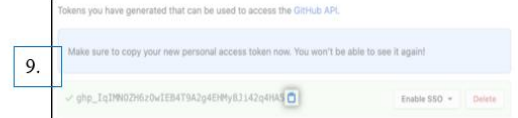
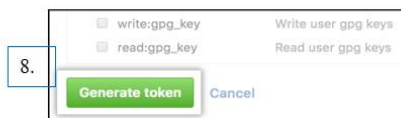
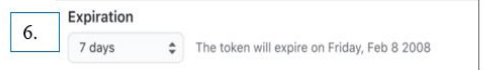
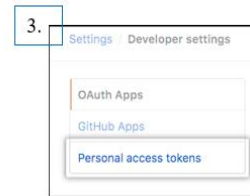
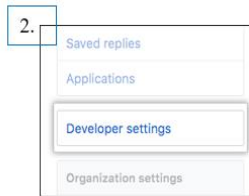
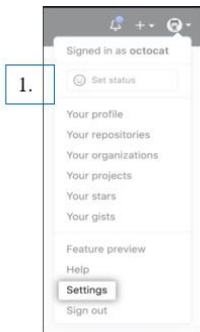
Il nous faut maintenant versionner sur un dépôt distant.

git push -u origin main : pousse les modifications réalisées vers le serveur.

Avant de réaliser votre push, il va falloir vous créer un jeton d’accès personnel.

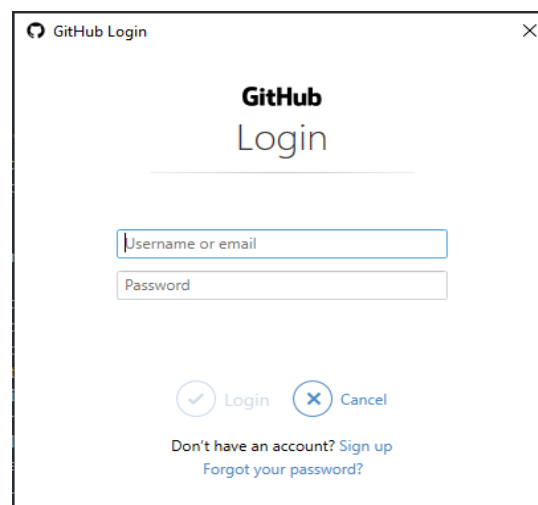
1. Pour cela, allez sur votre page GitHub.com et cliquez sur votre photo de profil en haut à droite de la page puis sur « Settings ».
2. Dans le menu de gauche, sélectionnez « Developer settings » ...
3. ... puis cliquez sur « Personal access tokens ».
4. Cliquez sur le bouton « Generate new token ».
5. Donnez-lui un nom informatif dans le champ « Note ».
6. Vous pouvez donner une date d’expiration à votre jeton d’accès personnel dans le menu « Expiration » ...

7. ... et gérer la portée et les permissions de votre jeton d'accès personnel (ici, pour accéder aux repositories par les lignes de commandes, sélectionnez « repo »).
8. Enfin, cliquez sur le bouton « Generate token ».
9. Vous avez ainsi accès à votre jeton d'accès personnel.



!!\ ATTENTION /\ : Considérez que vos jetons d'accès personnel sont comme des mots de passe : veuillez les noter sur un support de votre choix et ne les transmettez à personne. **!!\ ATTENTION /**

Pour versionner sur un dépôt distant, nous allons utiliser la commande « git push -u origin main ». En utilisant cette commande, rien de spécifique ne va se passer sur Git Bash. En revanche, une fenêtre d'identification va s'ouvrir pour faire le lien avec GitHub.com, comme le montre l'image suivante :



Entrez votre email ou votre nom d'utilisateur ainsi que votre mot de passe. Une nouvelle fenêtre devrait s'ouvrir en vous demandant de nouveau dans un premier temps votre nom d'utilisateur puis votre mot de passe. Dans le champ « mot de passe », entrez la clé du jeton d'accès personnel que vous avez généré précédemment. Si tout se passe bien, vous devriez voir dans Git Bash une confirmation que votre push s'est déroulé sans encombre, comme le montre l'image suivante :

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/github-tuto (main)
$ git push -u origin main
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), 347 bytes | 347.00 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.
To https://github.com/RmNLcN06/github-tuto.git
   e7df9a3..f76e8c6  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

De plus, si vous vous dirigez maintenant sur la page du repository « github-tuto », vous devriez voir que le fichier « index.html » a bien été ajouté :

The screenshot shows the GitHub interface for the repository 'github-tuto' by user 'RmNLcN06'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows two commits: 'Initial commit' for 'README.md' (yesterday) and 'Nouveau commit avec un paragraphe de plus' for 'index.html' (8 minutes ago). The repository description is 'Repository de tutoriel sur la pratique de GitHub.'

RmNLcN06 / **github-tuto** Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights

main 1 branch 0 tags Go to file Add file Code

RmNLcN06 Nouveau commit avec un paragraphe de plus f76e8c6 8 minutes ago 3 commits

File	Commit	Time
README.md	Initial commit	yesterday
index.html	Nouveau commit avec un paragraphe de plus	8 minutes ago

README.md

github-tuto

Repository de tutoriel sur la pratique de GitHub.



GitHub Cheat Sheet

Versionner son travail

Versionner en local

git init	initialise le dépôt (se mettre sur le bon dossier), mieux à faire depuis Github.com
git add .	ajoute toutes les modifications (le . symbolise tout)
git commit -m "explication"	créer un nouveau commit. git add pousse les fichiers en zone d'index, git commit les sauvegarde réellement dans un nouveau commit

Gérer les commits

git log	liste des commits
git log -n2	affiche les 2 derniers commits
git show sha-1	voir commit spécifique (cliquer molette souris pour coller)
git checkout sha-1	remettre la version du sha-1
git checkout main	remettre version la plus récente

Versionner sur un dépôt distant

git clone lien-github.com	recupérer travail depuis dépôt distant
git push -u origin main	pousse les modifications vers serveur
git push -f origin main	pousse de force des modifications (à manipuler avec précaution)

Naviguer dans Git Bash

pwd	savoir dans quel dossier je suis
mkdir "dossier"	créer un dossier (Make Directory)
touch fichier.txt	créer fichier
ls	liste le dossier courant
ls -la	liste tout plus précisément que ls
cd dossier	aller dans le dossier (Change Directory)
cd ..	Remonter d'un dossier

Initialisation de Git

git config --global user.name	"Mon Nom"
git config --global user.email	mon@mail.com
git config --global --list	Affiche nom et mail

Autres commandes

git status	état du fichier
git diff	affiche les modifs avant commit

Commit son projet sur Github

git add .
git commit -m "message"
git push -u origin main

