

Cours Inf3523 – Architecture et Génie des Logiciels

Lab_3 : Travailler avec GIT de manière distante avec Github

Ce lab a pour but de vous initier à travailler de manière distribuée, en utilisant des serveurs distants comme point de contact pour différents développeurs.

Gérer les dépôts distants

Git est un outil de versionnage de fichiers mais il a été conçu en ayant la collaboration à l'esprit. Essentiellement, un dépôt distant Git est un autre "endroit" qui possède le même dépôt que celui que vous avez sur votre ordinateur. Comme le montre l'image suivante, vous pouvez le considérer comme différentes copies du même dépôt qui peuvent échanger des données entre elles.



Cloner un dépôt distant

Créez un nouveau dossier sur votre disque pour cloner notre dépôt "grocery", puis, clonez le dépôt "grocery" en utilisant la commande "git clone" :

```
MINGW64/C:/Repos/grocery-cloned
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos
$ mkdir grocery-cloned

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos
$ cd grocery-cloned

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned
$ git clone ../grocery .
Cloning into '.'...
done.

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$

MINGW64/C:/Repos/grocery-cloned
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$ git log --oneline --graph --decorate --all
* 691f414 (HEAD -> master, origin/master, origin/HEAD) Merged melons branch into master
| \
| * ca5b2c0 (origin/melons, origin/berries) Add a watermelon
| * 1f11487 Add a blackberry
* | ac4433e Add a peach
| /
* 99bf1c3 Add an orange
* e906701 Add an apple
* 17e882a Add a banana to the shopping list

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$
```

L'origine

Git utilise "origin" comme nom par défaut pour un dépôt distant.

Git, grâce à l'option --all dans la commande "git log", nous montre qu'il y a d'autres branches dans le dépôt distant, mais comme vous pouvez le voir, elles n'apparaissent pas dans le dépôt cloné localement. Dans le dépôt cloné, il n'y a que la branche "master".

Une branche locale sur laquelle travailler peut être créée en la vérifiant simplement :

```
MINGW64/C/Repos/grocery-cloned
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/grocery-cloned (master)
$ git checkout berries
Switched to a new branch 'berries'
branch 'berries' set up to track 'origin/berries'.
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/grocery-cloned (berries)
$
```

Git indique qu'une branche locale a été configurée pour suivre la branche distante ; cela signifie que, à partir de maintenant, Git suivra activement les différences entre la branche locale et la branche distante, en vous notifiant les différences tout en vous fournissant des messages de sortie (par exemple, lorsque vous utilisez la commande "git status").

Donc si vous effectuez un commit dans cette branche, vous pouvez l'envoyer au dépôt distant et il fera partie de la branche distante "origin/berries".

Cela semble évident, mais dans Git, vous pouvez associer des branches comme vous le souhaitez ; par exemple, vous pouvez suivre une branche distante "origin/foo" avec une branche locale "bar", si vous le souhaitez. Alternativement, vous pouvez avoir des branches locales qui n'existent tout simplement pas sur le dépôt distant. Plus tard, nous verrons comment travailler avec les branches distantes.

```
MINGW64/C/Repos/grocery-cloned
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/grocery-cloned (berries)
$ git log --oneline --graph --decorate --all
* 691f414 (origin/master, origin/HEAD, master) Merged melons branch into master
| \
| * ca5b2c0 (HEAD -> berries, origin/melons, origin/berries) Add a watermelon
| * 1f11487 Add a blackberry
* | ac4433e Add a peach
| /
* 99bf1c3 Add an orange
* e906701 Add an apple
* 17e882a Add a banana to the shopping list
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/grocery-cloned (berries)
$
```

Maintenant, une branche verte "berries" apparaît juste à côté de la branche rouge "origin/berries". Cela nous indique que la branche locale "berries" et la branche distante "origin/berries" pointent vers le même commit.

Maintenant modifions la branche locale berries, et commitons le :

```
MINGW64/C/Repos/grocery-cloned
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/grocery-cloned (berries)
$ echo "blueberry" >> shoppingList.txt

DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/grocery-cloned (berries)
$ git commit -am "Add a blueberry"
warning: in the working copy of 'shoppingList.txt', LF will be replaced by CRLF
[berries 63a325c] Add a blueberry
1 file changed, 1 insertion(+)
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/grocery-cloned (berries)
$
```

```
MINGW64/C:/Repos/grocery-cloned

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (berries)
$ git log --oneline --graph --decorate --all
* 63a325c (HEAD -> berries) Add a blueberry
| * 691f414 (origin/master, origin/HEAD, master) Merged melons branch into master
| | \
| | /
| | /
| | /
* | ca5b2c0 (origin/melons, origin/berries) Add a watermelon
* | 1f11487 Add a blackberry
| * ac4433e Add a peach
| /
* 99bf1c3 Add an orange
* e906701 Add an apple
* 17e882a Add a banana to the shopping list

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (berries)
$
```

Partager des commits locaux avec **git push**

Lorsque vous effectuez un commit, il est disponible uniquement localement ; si vous souhaitez le partager avec une branche distante, vous devez l'envoyer en faisant un "push" (pousser).

Maintenant, nous allons essayer de pousser les modifications de la branche "berries" vers "origin" ; la commande est "git push", suivie du nom du dépôt distant et de la branche cible :

```
MINGW64/C:/Repos/grocery-cloned

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (berries)
$ git push origin berries
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 334 bytes | 167.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To C:/Repos/grocery-cloned/..grocery
    ca5b2c0..63a325c  berries -> berries

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (berries)
$
```

Git nous indique où il envoie les objets, la destination, qui dans ce cas est simplement un autre dossier sur mon ordinateur : vers C:/Repos/grocery. Il nous indique ensuite le hash du commit distant où se trouvait origin/berries à l'origine, ainsi que le hash du nouveau commit, qui est le même que celui de la branche locale berries, ca5b2c0..63a325c. Enfin, il donne le nom des deux branches, la branche locale et la branche distante, berries -> berries.

Maintenant, nous voulons évidemment vérifier s'il y a un nouveau commit dans la branche berries du dépôt distant ; donc, ouvrez le dossier "grocery" dans une nouvelle console et faites "git log" :

```
MINGW64/C:/Repos/grocery

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery (master|CHERRY-PICKING)
$ git log --oneline --graph --decorate --all
* 63a325c (berries) Add a blueberry
| * 691f414 (HEAD -> master) Merged melons branch into master
| | \
| | /
| | /
| | /
* | ca5b2c0 (melons) Add a watermelon
* | 1f11487 Add a blackberry
| * ac4433e Add a peach
| /
* 99bf1c3 Add an orange
* e906701 Add an apple
* 17e882a Add a banana to the shopping list

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery (master|CHERRY-PICKING)
$
```

Récupérer les commits distants avec **git pull**

Nous allons faire l'expérience inverse : récupérer les mises à jour du dépôt distant et les appliquer à notre copie locale.

Donc, effectuez un nouveau commit dans le dépôt "grocery", puis nous le téléchargerons dans la copie "grocery-cloned" :

```
MINGW64/C:/Repos/grocery
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery (master|CHERRY-PICKING)
$ echo "apricot" >> shoppingList.txt

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery (master|CHERRY-PICKING)
$ git commit -am "Add an apricot"
warning: in the working copy of 'shoppingList.txt', LF will be replaced by CRLF
[master 9f64a48] Add an apricot
Author: Bass Toure <bassirou.toure@esp.sn>
Date: Thu May 11 17:55:50 2023 +0000
1 file changed, 2 insertions(+)

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery (master)
```

D'accord, maintenant revenons au dépôt "grocery-cloned".

Nous pouvons récupérer des objets à partir d'un dépôt distant avec la commande "git pull".

En réalité, "git pull" est une commande composée ; en fait, c'est essentiellement la combinaison de deux autres commandes Git, "git fetch" et "git merge".

Essayons d'abord "git pull", puis nous essaierons d'utiliser "git fetch" et "git merge" séparément.

Revenez au dépôt "grocery-cloned", passez à la branche "master" et effectuez un "git pull".

```
MINGW64/C:/Repos/grocery-cloned
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery (master)
$ cd ..

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos
$ cd grocery-cloned/

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (berries)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$
```

Git dit que notre branche est à jour avec 'origin/master', mais ce n'est pas le cas. Cela est dû au fait que, pour Git, la seule façon de savoir si nous sommes à jour par rapport à un dépôt distant est d'effectuer un "git fetch".

Pour l'instant, utilisons "git pull" : la commande vous demande de spécifier le nom du dépôt distant à partir duquel vous souhaitez effectuer la récupération, qui est "origin" dans ce cas, puis la branche que vous souhaitez fusionner avec votre branche locale, qui est "master" :

```

MINGW64/C:/Repos/grocery-cloned
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 343 bytes | 34.00 KiB/s, done.
From C:/Repos/grocery-cloned/../../grocery
 * branch            master      -> FETCH_HEAD
    691f414..9f64a48  master      -> origin/master
Updating 691f414..9f64a48
Fast-forward
 shoppingList.txt | 2 ++
 1 file changed, 2 insertions(+)

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$

```

Git nous indique qu'il y a trois nouveaux objets à récupérer ; après les avoir obtenus, il effectue une fusion (merge) sur la branche locale "master", et dans ce cas, il effectue une fusion (merge) en avance rapide (fast-forward merge).

Maintenant, essayons de faire ces étapes séparément ; créez un nouveau commit dans le dépôt "grocery", sur la branche "master".

```

MINGW64/C:/Repos/grocery
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery (master)
$ echo "plum" >> shoppingList.txt

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery (master)
$ git commit -am "Add a plum"
warning: in the working copy of 'shoppingList.txt', LF will be replaced by CRLF
[master f4939e1] Add a plum
 1 file changed, 1 insertion(+)

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery (master)
$

```

Maintenant, effectuez un "git fetch" sur le dépôt "grocery-cloned" :

```

MINGW64/C:/Repos/grocery-cloned
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 296 bytes | 29.00 KiB/s, done.
From C:/Repos/grocery-cloned/../../grocery
    9f64a48..f4939e1  master      -> origin/master

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$

```

Git a trouvé de nouveaux objets sur le dépôt distant et les a téléchargés.

NB : Vous pouvez effectuer un "git fetch" dans n'importe quelle branche, car cela télécharge simplement les objets distants ; il ne les fusionnera pas. En revanche, lors d'un "git pull", vous devez vous assurer d'être dans la bonne branche cible locale.

```
MINGW64/C:/Repos/grocery-cloned

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$ git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$
```

Lorsque vous avez un dépôt distant, la commande "git status" vous informe également de l'état de la synchronisation entre votre dépôt local et le dépôt distant. Ici, elle nous indique que nous sommes en retard par rapport au dépôt distant car il a un commit de plus que nous dans la branche "master", et ce commit peut être fusionné en avance rapide (fast-forward).

Maintenant, synchronisons-nous avec un "git merge" ; pour fusionner une branche distante, nous devons spécifier, en plus du nom de la branche, le nom du dépôt distant, comme nous l'avons fait dans la commande "git pull" précédente :

```
MINGW64/C:/Repos/grocery-cloned

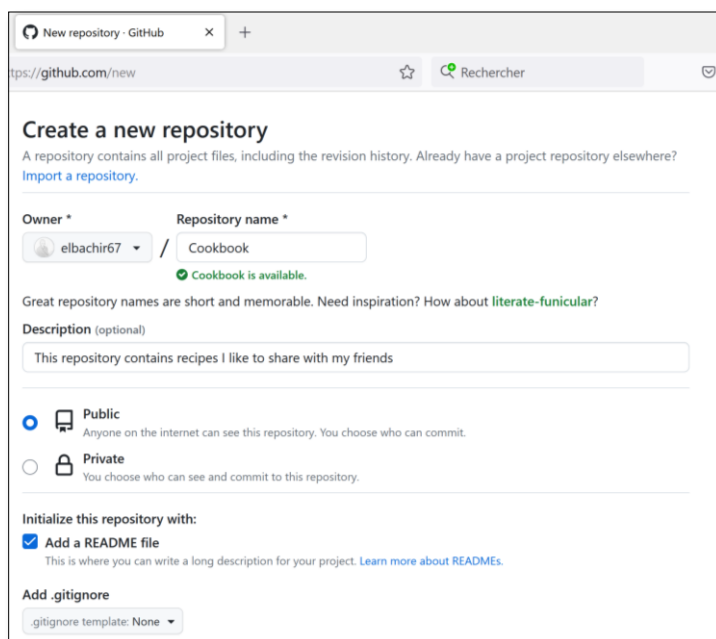
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$ git merge origin master
Updating 9f64a48..f4939e1
Fast-forward
 shoppingList.txt | 1 +
 1 file changed, 1 insertion(+)

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/grocery-cloned (master)
$
```

Travailler avec un serveur public sur GitHub

Nous allons utiliser Github. Commencez par créer un nouveau compte Github : www.github.com.

Allez dans l'onglet "Repositories", cliquez sur le bouton vert "New" et choisissez un nom pour votre dépôt, comme vous pouvez le voir dans la capture d'écran suivante :

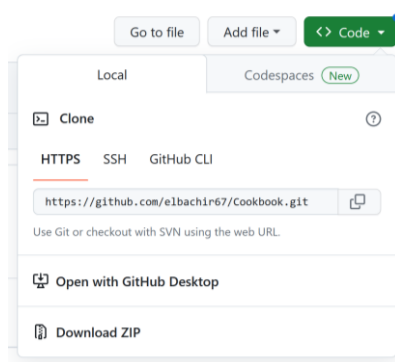


Ensuite, vous pouvez rédiger une description pour votre dépôt ; cela est utile pour permettre aux visiteurs de votre profil de mieux comprendre l'objectif de votre projet. Nous créons notre dépôt en tant que dépôt public car les dépôts privés ont un coût, puis nous l'initialisons avec un fichier README ; en choisissant cette option, GitHub effectue un premier commit pour nous, initialisant le dépôt qui est désormais prêt à être cloné.

Cloner le dépôt

Maintenant que nous avons un dépôt distant, il est temps de le relier localement. Pour cela, Git fournit la commande `git clone`, comme nous l'avons déjà vu.

L'utilisation de cette commande est assez simple ; dans ce cas, tout ce que nous devons savoir est l'URL du dépôt à cloner. L'URL est fournie par GitHub dans la partie inférieure droite de la page d'accueil du dépôt :



Essayons de suivre ces étapes:

1. Rendez-vous dans un dossier racine local pour les dépôts.
2. Ouvrez un terminal Bash à l'intérieur.
3. Tapez `git clone` suivi du lien copié ci-haut :

```
MINGW64/C:/Repos/Cookbook
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos
$ git clone https://github.com/elbachir67/Cookbook.git
Cloning into 'Cookbook'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos
```

Upload des modifications vers les dépôts distants

Alors, essayons de modifier le fichier README.md et d'uploader les modifications vers GitHub :

1. Modifiez le fichier README.md en utilisant votre éditeur préféré.
2. Ajoutez-le à l'index, avec `git add` . puis effectuez un commit.
3. Envoyez votre commit vers le dépôt distant en utilisant la commande `git push`.

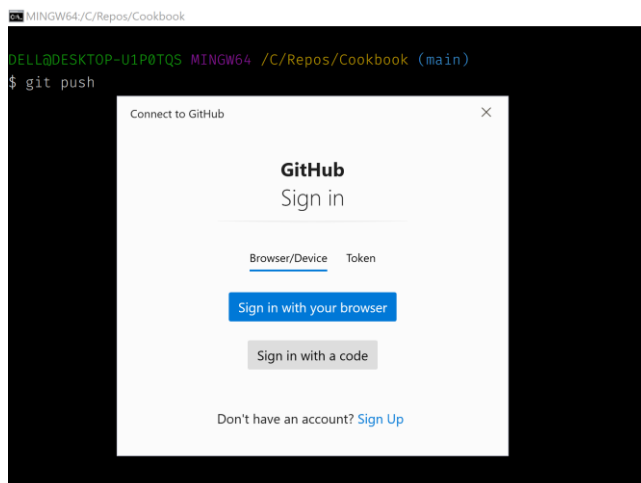
```
MINGW64/C/Repos/Cookbook
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (main)
$ vim README.md

DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (main)
$ git add README.md

DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (main)
$ git commit -m "Add a sentence to readme"
[main 487c97a] Add a sentence to readme
1 file changed, 1 insertion(+)

DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (main)
$
```

Maintenant, essayez de taper "git push" et appuyez sur ENTRÉE sans spécifier autre chose :



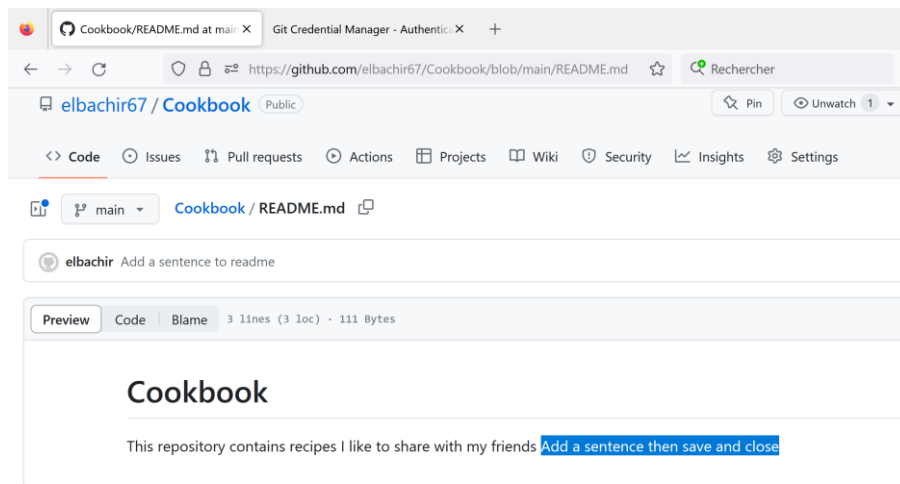
Il s'agit du Gestionnaire d'informations d'identification Git ; il vous permet de définir des informations d'identification sur votre machine Windows. Si vous êtes sur Linux ou macOS, la situation peut être différente, mais le concept est le même : nous devons fournir à Git les informations d'identification afin d'accéder au dépôt distant GitHub ; elles seront ensuite stockées sur notre système.

Saisissez vos informations d'identification, puis appuyez sur le bouton Connexion ; après cela, Git continue avec :

```
MINGW64/C/Repos/Cookbook
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (main)
$ git push
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 348 bytes | 348.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/elbachir67/Cookbook.git
  4320a75..487c97a  main -> main

DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (main)
$
```

La commande "git push" vous permet d'uploader le travail local vers un emplacement distant configuré ; dans ce cas, un dépôt distant GitHub :



Lorsque vous effectuez un git push sans spécifier autre chose, Git envoie tous les nouveaux commits et tous les objets associés que vous avez réalisés localement dans votre branche actuelle vers le dépôt distant ; pour les nouveaux commits, cela signifie que nous enverrons uniquement les commits locaux qui n'ont pas encore été uploadés.

Pusher ou pousser une nouvelle branche vers le dépôt distant

Évidemment, nous pouvons créer et pousser une nouvelle branche vers le dépôt distant pour rendre notre travail public et visible aux autres collaborateurs ; par exemple, je vais créer une nouvelle branche pour une nouvelle recette, puis je la pousserai vers le serveur distant GitHub. Suivez ces étapes simples :

Créez une nouvelle branche, par exemple Mafe. Ajoutez-y un nouveau fichier, par exemple mafe-senegal.md, et effectuez un commit.

```
MINGW64/C/Repos/Cookbook
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (main)
$ git branch Mafe

DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (main)
$ git checkout Mafe
Switched to branch 'Mafe'

DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (Mafe)
$ code mafe-senegal.md

DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (Mafe)
$ git add mafe-senegal.md

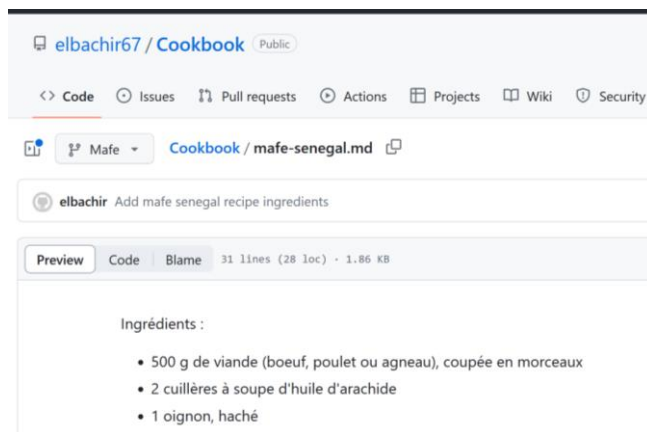
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (Mafe)
$ git commit -m "Add mafe senegal recipe ingredients"
[Mafe aa526c9] Add mafe senegal recipe ingredients
1 file changed, 31 insertions(+)
create mode 100644 mafe-senegal.md

DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Cookbook (Mafe)
```

Effectuez un push de la branche vers le dépôt distant en utilisant la commande `git push -u origin Mafe`.

```
MINGW64/C:/Repos/Cookbook
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/Cookbook (Mafe)
$ git push -u origin Mafe
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.15 KiB | 1.15 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'Mafe' on GitHub by visiting:
remote:   https://github.com/elbachir67/Cookbook/pull/new/Mafe
remote:
To https://github.com/elbachir67/Cookbook.git
 * [new branch]      Mafe -> Mafe
branch 'Mafe' set up to track 'origin/Mafe'.

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/Cookbook (Mafe)
$
```



Avec la commande "git push -u origin Mafe", nous avons indiqué à Git d'envoyer notre branche Mafe (et les commits qui s'y trouvent) vers l'origine (remote); avec l'option "-u", nous avons configuré les branches locales pour suivre la branche distante.

Nous savons que "origin" est le nom par défaut du remote d'un dépôt, tout comme "master" ou "main" est le nom par défaut de la branche. Lorsque vous clonez un dépôt à partir d'un remote, ce remote devient votre alias "origin". Lorsque vous demandez à Git d'envoyer ou de récupérer quelque chose, vous devez souvent lui indiquer le remote que vous souhaitez utiliser. En utilisant l'alias "origin", vous indiquez à Git que vous souhaitez utiliser votre remote par défaut.

Si vous souhaitez voir les remotes réellement configurés dans votre dépôt, vous pouvez taper simplement la commande "git remote", suivie de "-v" ("--verbose") pour obtenir plus de détails.

```
MINGW64/C:/Repos/Cookbook
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/Cookbook (Mafe)
$ git remote -v
origin https://github.com/elbachir67/Cookbook.git (fetch)
origin https://github.com/elbachir67/Cookbook.git (push)

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/Cookbook (Mafe)
$
```

Dans les détails, vous verrez l'URL complète du remote et vous découvrirez que Git stocke deux URL différentes :

- L'URL de récupération (Fetch URL), d'où nous obtenons les mises à jour des autres.
- L'URL de publication (Push URL), où nous envoyons nos mises à jour aux autres.

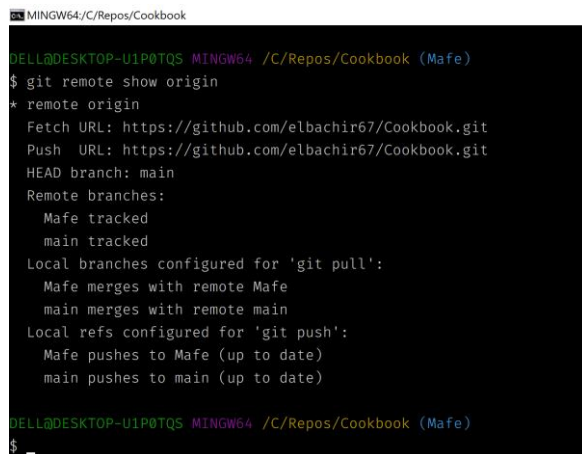
Cela nous permet de pusher et de tirer des modifications à partir de différents remotes, si vous le souhaitez, et souligne comment Git peut être considéré comme un système de gestion de versions pair à pair.

Vous pouvez ajouter, mettre à jour et supprimer des remotes en utilisant la commande "git remote".

Suivi des branches

En utilisant l'option -u, nous avons indiqué à Git de suivre la branche distante. Le suivi d'une branche distante permet de lier votre branche locale avec la branche distante ; veuillez noter que ce comportement n'est pas automatique, vous devez le configurer si vous le souhaitez. Lorsqu'une branche locale suit une branche distante, vous avez en réalité une branche locale et une branche distante qui peuvent être facilement synchronisées (veuillez noter qu'une branche locale ne peut suivre qu'une seule branche distante). C'est très utile lorsque vous devez collaborer avec des collègues distants sur la même branche, permettant à chacun de maintenir son travail synchronisé avec les modifications des autres.

Pour mieux comprendre la configuration actuelle de notre dépôt, essayez de saisir "git remote show origin".



```
MINGW64/C:/Repos/Cookbook
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/Cookbook (Mafe)
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/elbachir67/Cookbook.git
  Push URL: https://github.com/elbachir67/Cookbook.git
  HEAD branch: main
  Remote branches:
    Mafe tracked
    main tracked
  Local branches configured for 'git pull':
    Mafe merges with remote Mafe
    main merges with remote main
  Local refs configured for 'git push':
    Mafe pushes to Mafe (up to date)
    main pushes to main (up to date)
DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/Cookbook (Mafe)
$
```

Comme vous pouvez le voir, les branches Mafe et main sont toutes suivies. Vous voyez également que vos branches locales sont configurées pour pusher et récupérer les branches distantes portant le même nom. Cependant, il n'est pas obligatoire d'avoir des branches locales et distantes avec le même nom. Une branche locale "foo" peut suivre la branche distante "bar", et vice versa, sans aucune restriction.

Associer un remote à un dépôt local / Publier un dépôt local sur GitHub

Il peut arriver que vous ayez besoin de mettre votre dépôt local dans un endroit partagé où d'autres personnes pourront consulter votre travail. Nous allons essayer de mettre en œuvre ce mécanisme.

Créez un nouveau dépôt local à publier en suivant ces étapes simples :

1. Rendez-vous dans le dossier de vos dépôts locaux.
2. Créez un nouveau dossier HelloWorld.
3. À l'intérieur, créez un nouveau dépôt.
4. Ajoutez un nouveau fichier README.md et effectuez un commit :

```

MINGW64/C:/Repos/HelloWorld

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos
$ mkdir HelloWorld

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos
$ cd HelloWorld/

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/HelloWorld
$ git init
Initialized empty Git repository in C:/Repos/HelloWorld/.git/

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/HelloWorld (master)
$ echo "Hello World!" >> README.md

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/HelloWorld (master)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF
$ git commit -m "First commit"
[master (root-commit) 3721e38] First commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/HelloWorld (master)
$

```

Maintenant, créez un dépôt GitHub vide, nommé HelloWorld. Ne l'initialisez pas avec un fichier README ; nous en avons déjà un dans notre dépôt local.

Associer un remote à un dépôt local

Pour publier notre dépôt HelloWorld, il nous suffit d'ajouter son premier remote. Ajouter un remote est assez simple : `git remote add origin <URL-du-dépôt-distant>`.

```

MINGW64/C:/Repos/HelloWorld

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/HelloWorld (master)
$ git remote add origin https://github.com/elbachir67/HelloWorld.git

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/HelloWorld (master)
$

```

L'ajout du remote "origin" permet à votre dépôt local de communiquer avec le dépôt distant spécifié, facilitant ainsi le partage et la synchronisation des commits et des branches entre les deux. Vous pouvez ensuite utiliser des commandes telles que git push pour envoyer vos modifications vers le remote "origin" et git pull pour récupérer les dernières mises à jour du dépôt distant.

Publier un dépôt local sur GitHub

Après cela, poussez vos modifications locales vers le dépôt distant en utilisant la commande `git push -u origin master`.

```

MINGW64/C:/Repos/HelloWorld

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/HelloWorld (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 229 bytes | 229.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/elbachir67/HelloWorld.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

DELL@DESKTOP-U1P0TQS MINGW64 /C:/Repos/HelloWorld (master)
$

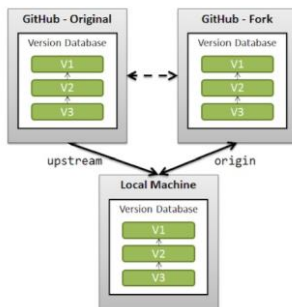
```

Collaboration utilisant GitHub

GitHub a facilité le partage de code, le suivi du travail d'autres personnes et la collaboration en utilisant deux concepts fondamentaux : les forks (dérivations) et les pull requests (demandes de fusion).

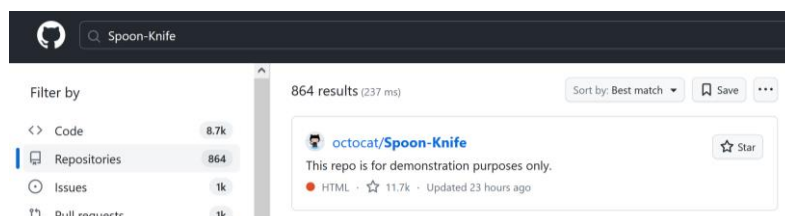
Dériver un fork

Lorsque vous effectuez un fork sur GitHub, vous obtenez essentiellement un clone côté serveur du dépôt sur votre compte GitHub ; si vous clonez votre dépôt forké localement, dans la liste des remotes, vous trouverez un "origin" qui pointe vers votre dépôt de compte, tandis que le dépôt d'origine prendra l'alias "upstream" (vous devez l'ajouter manuellement) :



Allez sur votre compte GitHub et essayez de fork un dépôt GitHub courant appelé Spoon-Knife, créé par l'utilisateur mascotte de GitHub, octocat. Voici les étapes à suivre :

1. Connectez-vous à votre compte GitHub.
2. Recherchez "spoon-knife" à l'aide de la barre de recherche située en haut à gauche de la page.
3. Cliquez sur le premier résultat, le dépôt octocat/Spoon-Knife.



4. Fork le dépôt en utilisant le bouton "Fork" situé à droite de la page.

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 *

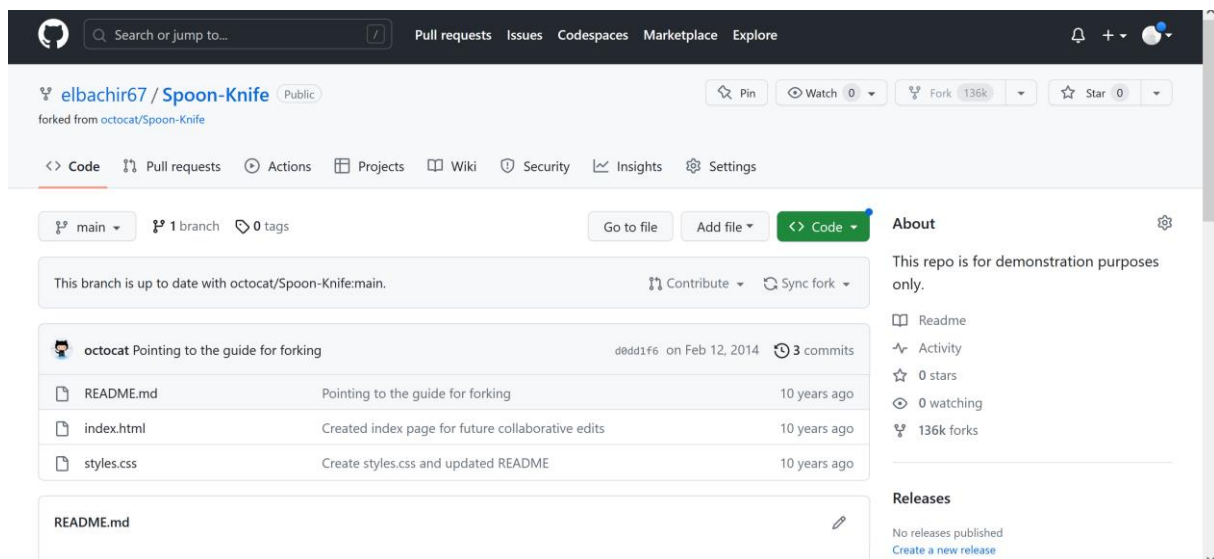
By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

☒ Copy the `main` branch only
Contribute back to octocat/Spoon-Knife by adding your own branch. [Learn more.](#)

📘 You are creating a fork in your personal account.

5. Vous obtiendrez un tout nouveau dépôt Spoon-Knife dans votre compte GitHub.



Maintenant nous pouvons cloner ce répertoire en local :

```
MINGW64/C/Repos
DELL@DESKTOP-UIP0TQS MINGW64 /C/Repos
$ git clone https://github.com/elbachir67/Spoon-Knife.git
Cloning into 'Spoon-Knife'...
remote: Enumerating objects: 10, done.
remote: Total 10 (delta 0), reused 0 (delta 0), pack-reused 10
Receiving objects: 100% (10/10), done.
Resolving deltas: 100% (1/1), done.

DELL@DESKTOP-UIP0TQS MINGW64 /C/Repos
$
```

```
MINGW64/C/Repos/Spoon-Knife
DELL@DESKTOP-UIP0TQS MINGW64 /C/Repos/Spoon-Knife (main)
$ git remote -v
origin https://github.com/elbachir67/Spoon-Knife.git (fetch)
origin https://github.com/elbachir67/Spoon-Knife.git (push)

DELL@DESKTOP-UIP0TQS MINGW64 /C/Repos/Spoon-Knife (main)
$
```

Comme vous pouvez le voir, le remote "upstream" n'est pas présent, vous devez l'ajouter manuellement ; pour l'ajouter, utilisez la commande `git remote add` :

```
MINGW64/C/Repos/Spoon-Knife
DELL@DESKTOP-UIP0TQS MINGW64 /C/Repos/Spoon-Knife (main)
$ git remote add upstream https://github.com/elbachir67/Spoon-Knife.git

DELL@DESKTOP-UIP0TQS MINGW64 /C/Repos/Spoon-Knife (main)
$ git remote -v
origin https://github.com/elbachir67/Spoon-Knife.git (fetch)
origin https://github.com/elbachir67/Spoon-Knife.git (push)
upstream https://github.com/elbachir67/Spoon-Knife.git (fetch)
upstream https://github.com/elbachir67/Spoon-Knife.git (push)

DELL@DESKTOP-UIP0TQS MINGW64 /C/Repos/Spoon-Knife (main)
$
```

Maintenant, vous pouvez maintenir votre dépôt local synchronisé avec les modifications de votre remote, "origin", et vous pouvez également récupérer celles provenant du remote "upstream", le dépôt original que vous avez forké. À ce stade, vous vous demandez probablement comment gérer deux remotes différents ; eh bien, c'est facile : il suffit de récupérer les modifications du remote

"upstream" et de les fusionner dans votre dépôt local, puis de les pousser vers votre remote "origin" en les regroupant avec vos propres modifications. Si quelqu'un d'autre clone votre dépôt, il recevra votre travail fusionné avec le travail effectué par quelqu'un d'autre sur le dépôt original.

Soumettre des pull requests (demande de fusion)

Si vous avez créé un fork d'un dépôt, c'est parce que vous n'êtes pas directement contributeur du projet original, ou simplement parce que vous ne voulez pas perturber le travail des autres avant de vous familiariser avec le code.

Cependant, à un certain moment, vous réalisez que votre travail peut être utile même pour le projet original : vous avez réalisé une meilleure implémentation d'une partie de code existante, ajouté une fonctionnalité manquante, etc.

Ainsi, vous vous retrouvez dans la situation où vous devez informer l'auteur original que vous avez réalisé quelque chose d'intéressant, lui demandant s'il souhaite jeter un coup d'œil et éventuellement intégrer votre travail. C'est à ce moment-là que les pull requests sont utiles.

Une pull request est une manière de dire à l'auteur original : "J'ai réalisé quelque chose d'intéressant en utilisant votre code original, voulez-vous jeter un coup d'œil et intégrer mon travail ?". Cela ne représente pas seulement une manière technique d'intégrer le travail, mais c'est également une pratique puissante pour favoriser les révisions de code comme le recommandent les adeptes de l'XP.

Une autre raison d'utiliser une pull request est que vous ne pouvez pas pusher directement vers le dépôt "upstream" si vous n'êtes pas contributeur du projet original : les pull requests sont le seul moyen. Dans de petits scénarios (comme une équipe de deux ou trois développeurs travaillant dans la même pièce), le modèle de fork et de fusion peut représenter une surcharge, il est donc plus courant de partager directement le dépôt original avec tous les contributeurs.

Création d'une pull request

Pour créer une demande de fusion, vous devez vous rendre sur votre compte GitHub et la créer directement à partir de votre compte fork ; mais avant cela, vous devez savoir que les demandes de fusion ne peuvent être créées que à partir de branches distinctes.

Pour faire une tentative, créons une nouvelle branche locale "TeaSpoon" dans notre dépôt, ajoutons un nouveau fichier, et poussons-le vers notre compte GitHub :



```
MINGW64/C:/Repos/Spoon-Knife
DELL@DESKTOP-UIP0TQS MINGW64 /C:/Repos/Spoon-Knife (main)
$ git branch TeaSpoon

DELL@DESKTOP-UIP0TQS MINGW64 /C:/Repos/Spoon-Knife (main)
$ git checkout TeaSpoon
Switched to branch 'TeaSpoon'

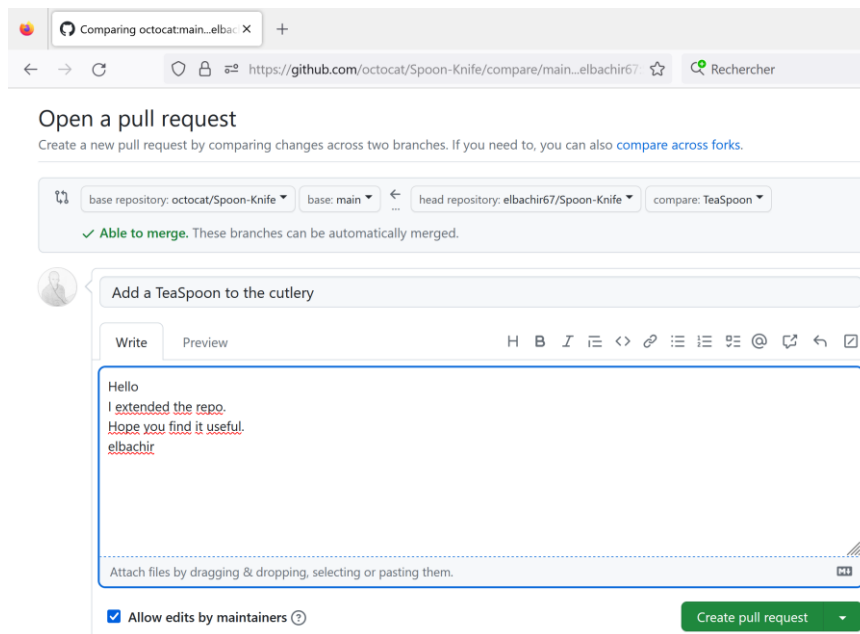
DELL@DESKTOP-UIP0TQS MINGW64 /C:/Repos/Spoon-Knife (TeaSpoon)
$ code teaSpoon.md
```

```
MINGW64:/C/Repos/Spoon-Knife
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Spoon-Knife (TeaSpoon)
$ git commit -m "Add a TeaSpoon to the cutlery"
[TeaSpoon e680401] Add a TeaSpoon to the cutlery
1 file changed, 1 insertion(+)
create mode 100644 teaSpoon.md

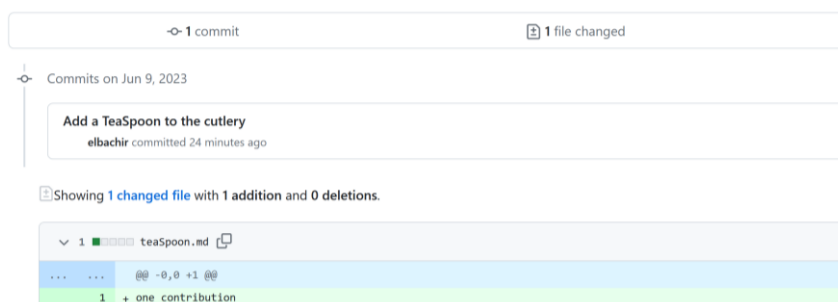
DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Spoon-Knife (TeaSpoon)
$ git push origin TeaSpoon
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 312 bytes | 312.00 KiB/s, done.
Total 3 (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 'TeaSpoon' on GitHub by visiting:
remote:   https://github.com/elbachir67/Spoon-Knife/pull/new/TeaSpoon
remote:
To https://github.com/elbachir67/Spoon-Knife.git
 * [new branch]      TeaSpoon -> TeaSpoon

DELL@DESKTOP-U1P0TQS MINGW64 /C/Repos/Spoon-Knife (TeaSpoon)
$
```

Si nous visitons le lien remote, nous verrons la page ci-dessous :



En bas de la page, vous verrez :



Et maintenant la dernière étape : cliquez sur le bouton "Créer une pull request" pour envoyer votre demande à l'auteur original, lui permettant ainsi de recevoir votre travail et de l'analyser avant d'accepter la demande d'extraction.

Avec les collaborateurs du projet, pouvez commencer à discuter de votre travail. Vous pouvez modifier le code, jusqu'à ce qu'un collaborateur du référentiel d'origine décide d'accepter votre demande ou de la rejeter, en fermant la pull request.

Résumé

Dans ce lab, nous avons exploré la capacité de Git à gérer plusieurs copies distantes de référentiels. Cela vous offre une large gamme de possibilités pour mieux organiser votre flux de travail collaboratif au sein de votre équipe.