

Tutoriel Git / GitHub

Travailler En Equipe

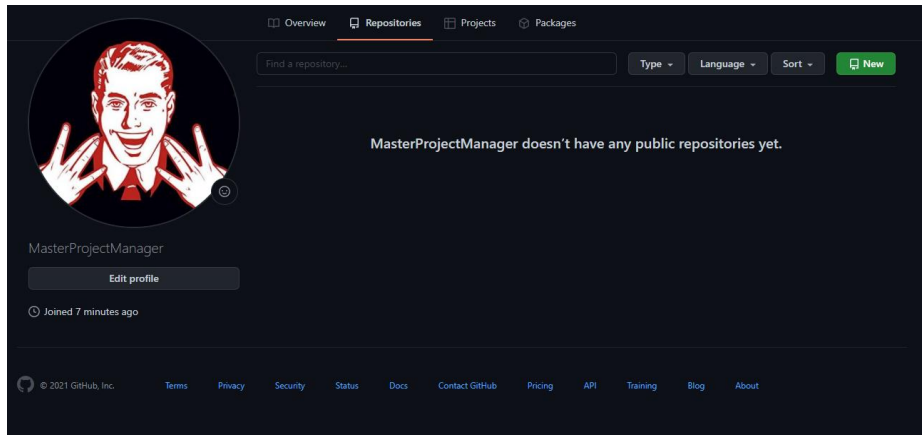
Table des matières

L'interface GitHub.....	2
Travailler avec un dépôt distant	6
git pull : récupérer les mises à jour sur un dépôt distant.....	6
Gérer les branches.....	7
git branch nom_branche : créer une branche.....	7
git checkout nom_branche : se positionner sur la branche à modifier.....	7
Gérer les branches sur un dépôt distant	8
git push --set-upstream origin nom_branche : créer la branche en dépôt distant.....	8
git merge nom_branche : fusionner une branche.....	9
Annexe : GitHub Cheat Sheet	11

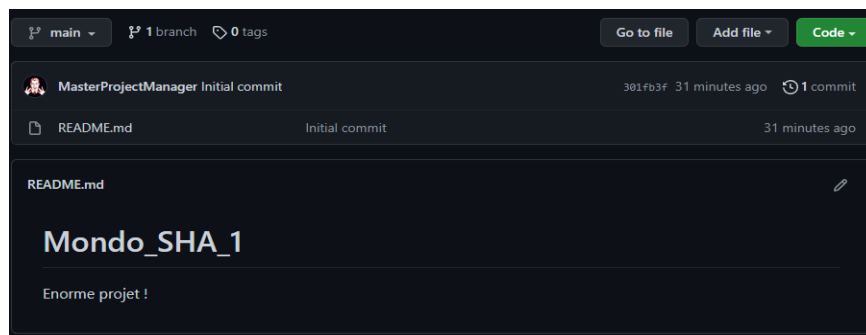
L'interface GitHub

Note : Pour les besoins de cette partie, nous allons créer un nouveau profil sur GitHub.com pour simuler un travail en équipe. Le nouveau profil que nous allons nommer « MasterProjectManager » représentera le dépôt du chef de projet (ce n'est qu'un exemple : cela peut tout aussi bien être un autre développeur ou autres).

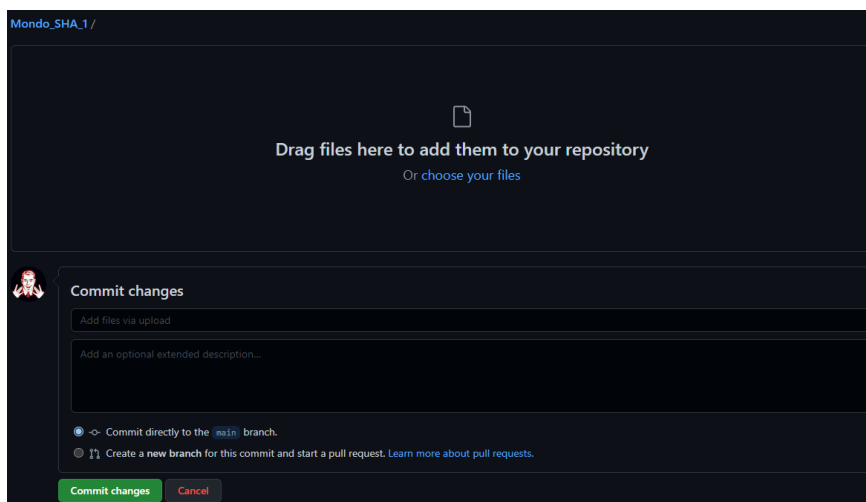
Voici le profil de « MasterProjectManager » :



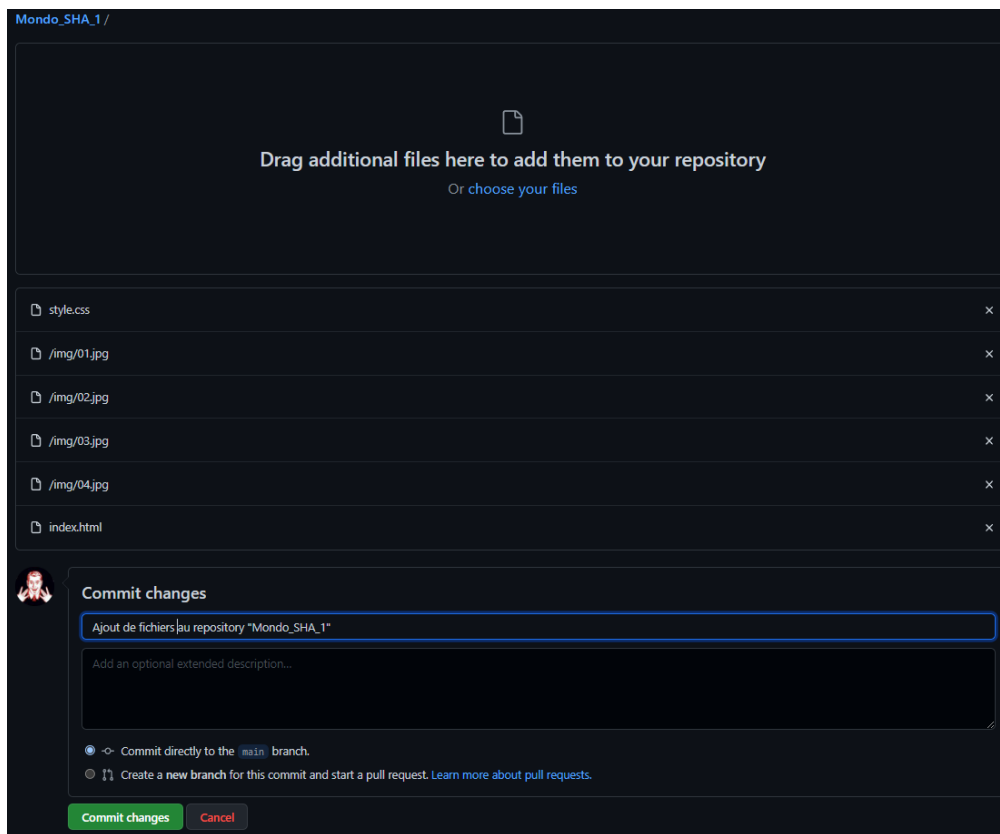
Il ne connaît absolument pas les lignes de commande et l'utilisation de Git Bash. Il réalisera toutes ses opérations à partir de l'interface de GitHub.com. Il n'y a pour le moment aucun repository sur son profil. Nous allons en créer un en cliquant sur le bouton vert « New » en haut à droite et nous le nommerons « Mondo_SHA_1 » :



Ce repository ne contient pour le moment que le fichier README. Nous allons lui ajouter des fichiers. Pour cela, nous cliquons sur l'onglet « Add file » puis sur « Upload files ». Nous arrivons à l'écran suivant :



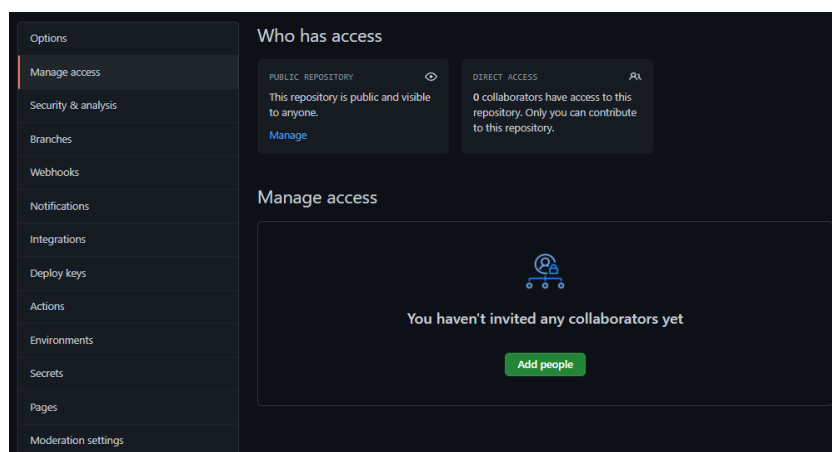
A partir de cette étape, nous avons deux possibilités qui nous sont proposées : soit réaliser un drag pour mettre directement nos fichiers dans notre repository soit cliquez sur « choose your files ». Nous choisissons la première option en transférant des fichiers « index.html », « style.css » et un dossier « img » contenant quatre photos (**je précise que j'ai créé ces fichiers et dossier pour le bien de cette présentation**). Nous avons bien nos fichiers transférés. Vous pouvez également ajouter un petit commentaire dans le champ « Add files via upload » pour expliquer de manière synthétique et précise votre commit, comme vous pouvez le voir dans l'image suivante :



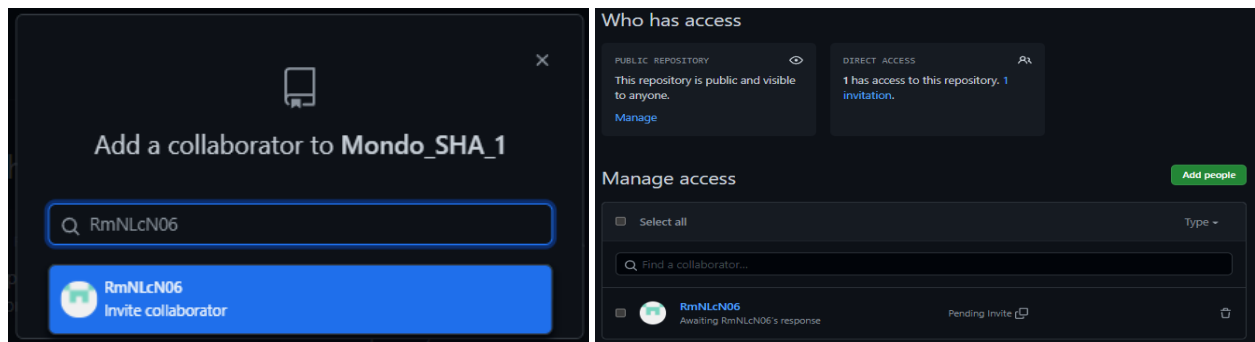
Enfin nous cliquons sur le bouton vert « Commit changes » pour commiter nos changements. Si vous rencontrez aucun problème durant cette opération, vous devriez arriver sur la page suivante (voir photo suivante) : vos fichiers ont bien été commités au repository « Mondo_SHA_1 ». Félicitations ! :



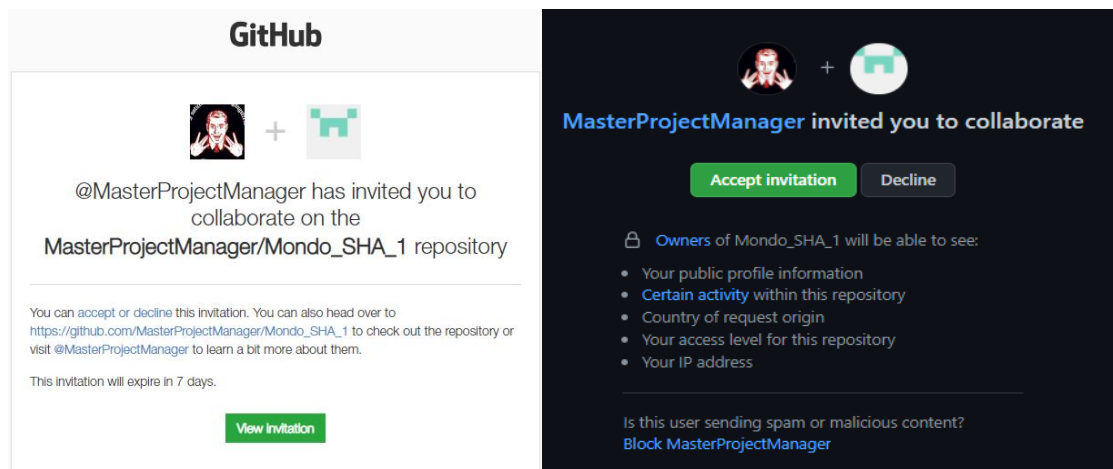
Il faut maintenant accorder les droits au développeur pour qu'il puisse pousser les modifications sur le repository. Pour se faire, allez dans l'onglet « Settings » et cliquez sur « Manage access » dans le menu de gauche. On vous demandera de taper votre mot de passe à ce moment-là. Vous devriez arriver sur cette page :



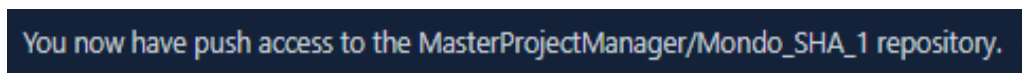
Cliquez sur le bouton vert « Add people » pour ajouter un collaborateur. Tapez ensuite l'identifiant du collaborateur que vous souhaitez ajouter au projet. Dans notre exemple, MasterProjectManager va autoriser RmNLcN06 à travailler sur le repository « Mondo_SHA_1 » :



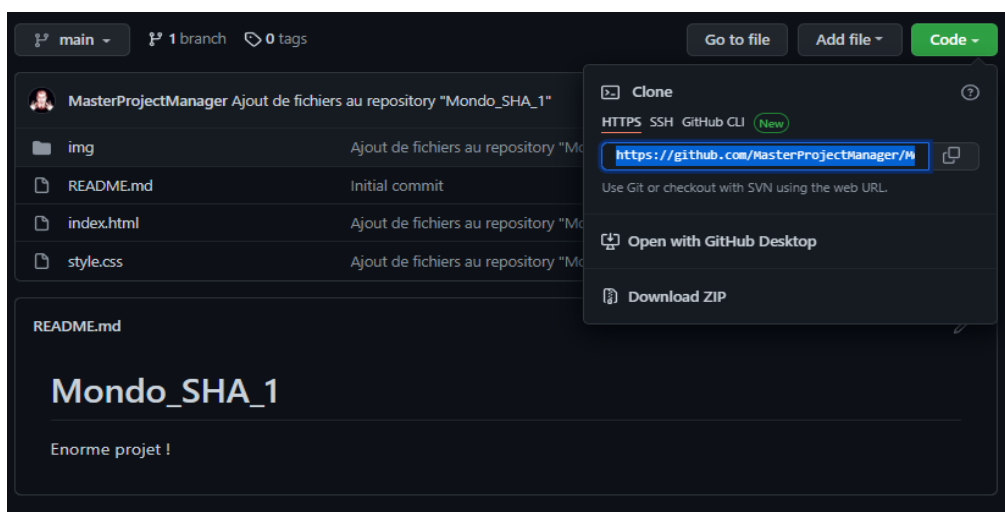
MasterProjectManager attend maintenant la réponse de la part de RmNLcN06 pour valider cette participation. Du côté de RmNLcN06, une invitation à travailler sur ce repository a été envoyée par mail. Un lien amène ensuite sur GitHub.com pour accepter cette invitation :



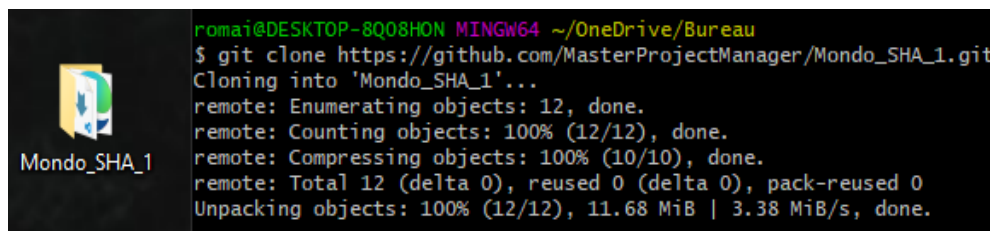
De cette manière, RmNLcN06 peut maintenant pousser les modifications réalisées sur le repository « Mondo_SHA_1 ».



A partir de maintenant, nous nous plaçons du point de vue d'un développeur (RmNLcN06) qui veut récupérer ce repository sur son ordinateur. Nous allons d'abord cloner ce repository. Cliquez sur le bouton vert « Code » et copiez l'adresse HTTPS :



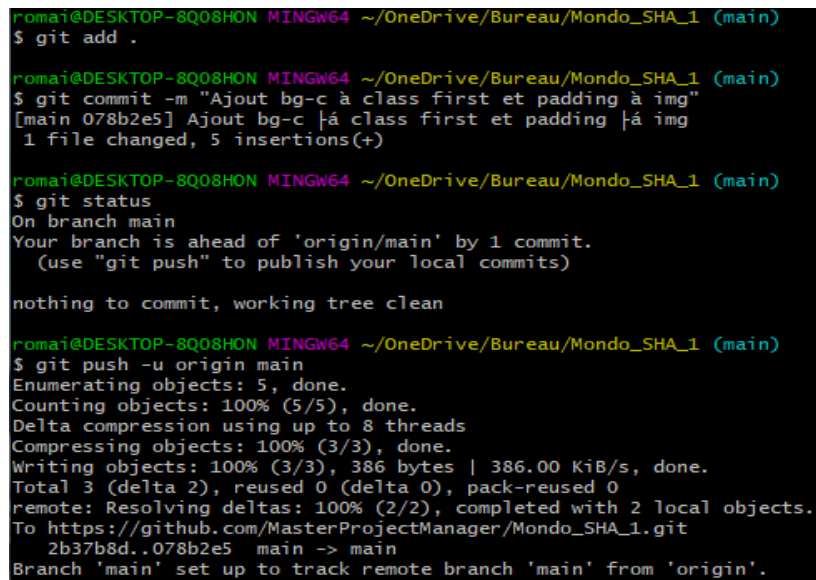
Faites un clic droit sur votre Bureau et choisissez l'onglet « Git Bash Here ». Une fenêtre Git Bash s'ouvre. Tapez ensuite la commande « git clone » et collez l'adresse HTTPS à l'aide de la molette de la souris. Vous devriez arriver au résultat suivant sur Git Bash et le repository « Mondo_SHA_1 » devrait se trouver sur votre Bureau :



```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau
$ git clone https://github.com/MasterProjectManager/Mondo_SHA_1.git
Cloning into 'Mondo_SHA_1'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 12 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (12/12), 11.68 MiB | 3.38 MiB/s, done.

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1
$ git checkout -b main
```

Nous ouvrons maintenant le dossier « Mondo_SHA_1 » avec notre éditeur de code (dans cet exemple, j'utilise Visual Studio Code). Nous allons modifier un peu notre fichier « style.css ». Dans ce fichier, nous ajoutons à la classe « .first » un « background-color : lightgrey ; » et au sélecteur « img » un « padding : 40px 0px ; ». Après avoir enregistré ces modifications, nous revenons à Git Bash et tapons « git add . » pour pousser toutes les modifications en zone d'index, puis « git commit -m « Ajout bg-c à class first et padding à img » » pour les sauvegarder réellement dans un nouveau commit et enfin « git push -u origin main » pour pousser les modifications vers le serveur :



```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git add .

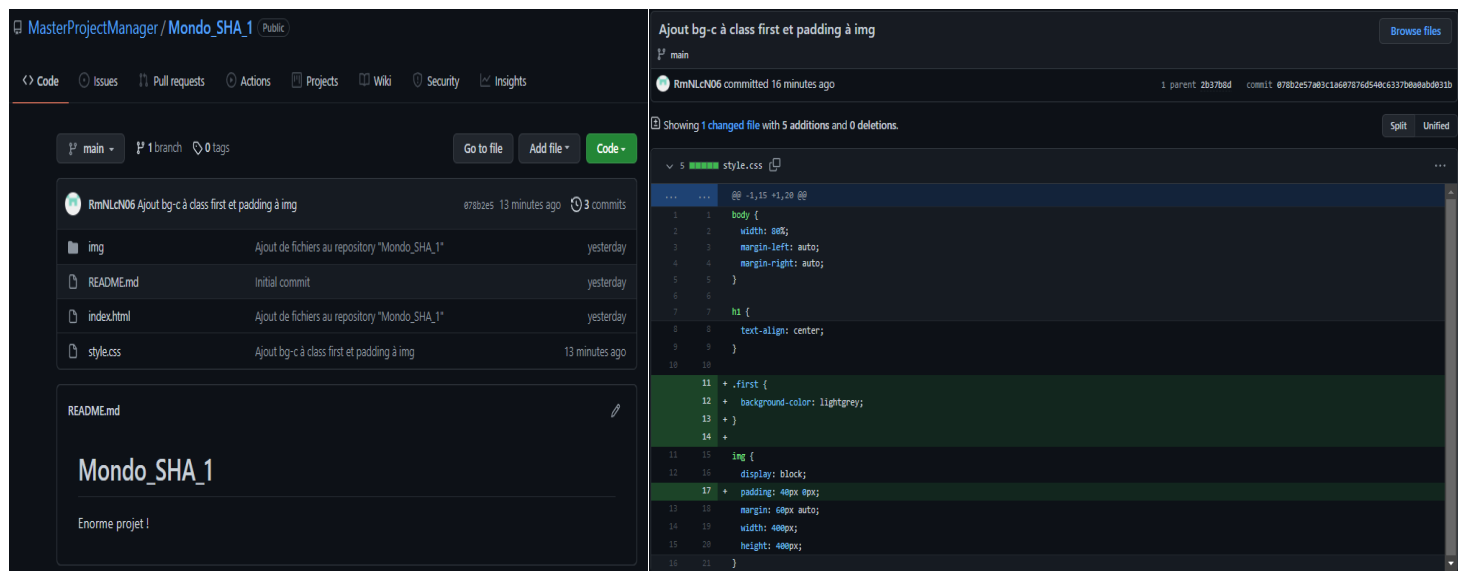
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git commit -m "Ajout bg-c à class first et padding à img"
[main 078b2e5] Ajout bg-c à class first et padding à img
1 file changed, 5 insertions(+)

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (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), 386 bytes | 386.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/MasterProjectManager/Mondo_SHA_1.git
2b37b8d..078b2e5 main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Nous pouvons vérifier maintenant si les modifications ont été ajoutées au repository « Mondo_SHA_1 » sur le profil de MasterProjectManager. Si tout s'est bien déroulé, vous devriez avoir un résultat du même ordre que sur les images suivantes :



The screenshot shows the GitHub repository page for 'MasterProjectManager / Mondo_SHA_1'. The left sidebar displays the repository overview with a list of files: 'img', 'README.md', 'index.html', and 'style.css'. The main content area shows the commit history, with the latest commit 'Ajout bg-c à class first et padding à img' by 'RmNLcN06' highlighted. The right sidebar shows the diff for the selected commit, highlighting the changes in 'style.css'.

```
@@ -1,15 +1,20 @@
1 1 body {
2 2 width: 80%;
3 3 margin-left: auto;
4 4 margin-right: auto;
5 5 }
6 6
7 7 h1 {
8 8 text-align: center;
9 9 }
10 10
11 + .first {
12 + background-color: lightgrey;
13 + }
14 +
15 15 img {
16 16 display: block;
17 + padding: 40px 0px;
18 18 margin: 60px auto;
19 19 width: 400px;
20 20 height: 400px;
21 21 }
```

Travailler avec un dépôt distant

git pull : récupérer les mises à jour sur un dépôt distant.

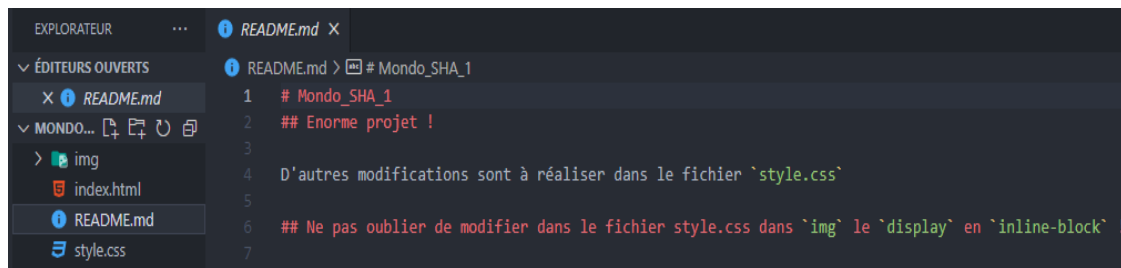
Imaginons que MasterProjectManager a modifié le fichier « README.md » dans la nuit pour nous donner des pistes de modifications sur notre repository commun « Mondo_SHA_1 » :



Le lendemain, RmNLcN06 doit vérifier si des mises à jour ont été réalisées sur ce repository. Pour ce faire, nous allons taper dans Git Bash la commande « git pull » qui va nous permettre de récupérer les mises à jour sur le dépôt distant :

```
romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git pull
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), 1.55 KiB | 28.00 KiB/s, done.
From https://github.com/MasterProjectManager/Mondo_SHA_1
 078b2e5..e33f3a2  main    -> origin/main
Updating 078b2e5..e33f3a2
Fast-forward
 README.md | 6 +++++-
 1 file changed, 5 insertions(+), 1 deletion(-)
```

En ouvrant notre repository « Mondo_SHA_1 » avec notre éditeur de texte et si nous cliquons sur le fichier « README.md », nous voyons apparaître à l'écran les modifications réalisées par MasterProjectManager :



Gérer les branches

En général, il est déconseillé lors du travail en équipe de travailler sur des pages ou des composants similaires car cela peut entraîner des conflits lors du commit. C'est pourquoi, dans le but d'éviter les conflits de commit, il est conseillé de créer des branches pour le travail en équipe.

git branch nom_branche : créer une branche.

Imaginons que nous devons travailler sur le développement d'un système de navigation pour l'une de nos pages. Pour ne pas entrer en conflit avec l'un de nos collaborateurs qui travaillerait sur le même sujet, nous allons nous créer une branche avec la commande « git branch navbar » dans Git Bash. Si on retape la commande « git branch » sans un nom de branche, nous voyons que dorénavant nous avons deux branches qui apparaissent : une branche « main » sur laquelle nous nous trouvons actuellement, et une branche « navbar » que nous venons de créer :

```
romai@DESKTOP-8Q08H0N MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git branch navbar

romai@DESKTOP-8Q08H0N MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git branch
* main
  navbar
```

git checkout nom_branche : se positionner sur la branche à modifier.

Pour se déplacer de notre branche « main » à notre branche « navbar » précédemment créée, nous utiliserons la commande « git checkout navbar » :

```
romai@DESKTOP-8Q08H0N MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git checkout navbar
Switched to branch 'navbar'

romai@DESKTOP-8Q08H0N MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (navbar)
$
```

Pour le moment, cette branche « navbar » ne se trouve qu'en local ; à savoir sur l'ordinateur de RmNLcN06. Nous allons envoyer notre branche sur un dépôt distant dans la partie suivante.

Gérer les branches sur un dépôt distant

git push --set-upstream origin nom_branche : créer la branche en dépôt distant

!! ATTENTION !! Il faut au préalable faire un « git branch nom_branche » pour créer une branche en local avant d'utiliser la commande « git push --set-upstream origin nom_branche » **!! ATTENTION !!**

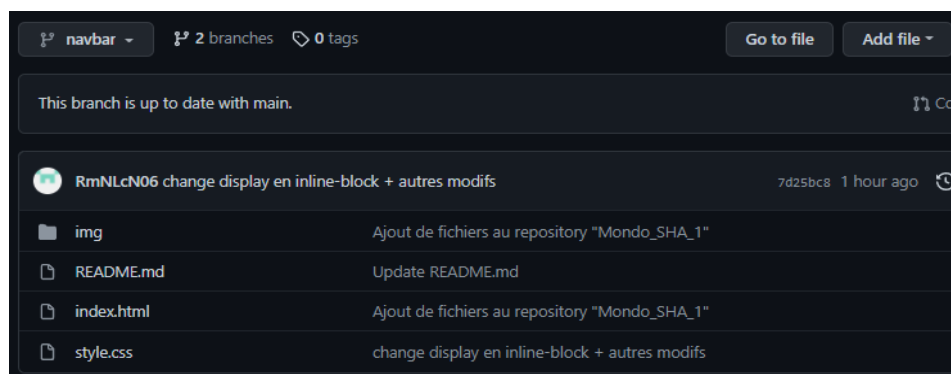
En tapant la commande « git push --set-upstream origin navbar », nous allons envoyer notre branche « navbar » sur le repository « Mondo_SHA_1 » :

```
romai@DESKTOP-8Q08H0N MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (navbar)
$ git push --set-upstream origin navbar
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'navbar' on GitHub by visiting:
remote:   https://github.com/MasterProjectManager/Mondo_SHA_1/pull/new/navbar
remote:
To https://github.com/MasterProjectManager/Mondo_SHA_1.git
 * [new branch]      navbar -> navbar
Branch 'navbar' set up to track remote branch 'navbar' from 'origin'.
```

Après avoir tapé cette commande, la branche « navbar » a été ajoutée au repository :



La branche « navbar » contient tous les fichiers qui existent à l'intérieur de la branche « main ». De cette manière, nous ne modifions pas les fichiers se trouvant dans la branche « main » :

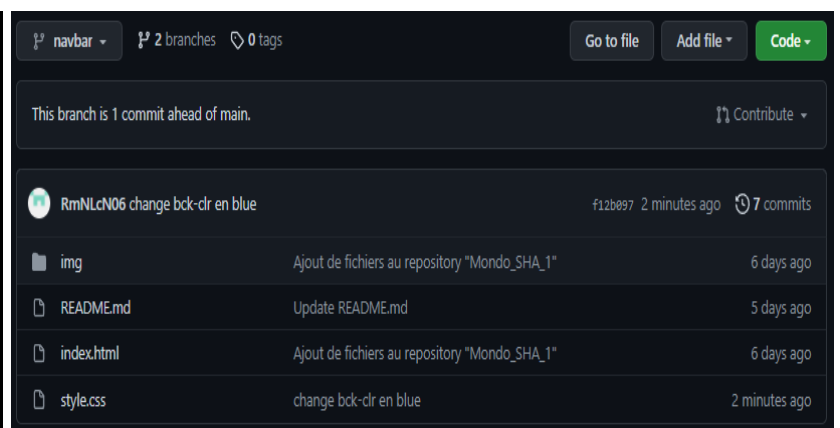


Le fait de réaliser des modifications et des commits avec les fichiers se trouvant sur la branche « navbar » reste sur cette branche. Si nous réalisons des modifications sur le fichier « style.css » en changeant le « background-color » en « blue », et qu'en fin de journée nous voulons réaliser un commit pour enregistrer notre travail seulement sur la branche « navbar », nous taperons les commandes « git add . », puis « git commit -m « change bck-clr en blue » » et enfin « git push -u origin navbar » :

```
romai@DESKTOP-8Q08H0N MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (navbar)
$ git add .

romai@DESKTOP-8Q08H0N MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (navbar)
$ git commit -m "change bck-clr en blue"
[navbar f12b097] change bck-clr en blue
1 file changed, 1 insertion(+), 1 deletion(-)

romai@DESKTOP-8Q08H0N MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (navbar)
$ git push -u origin navbar
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), 311 bytes | 311.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/MasterProjectManager/Mondo_SHA_1.git
 7d25bc8..f12b097 navbar -> navbar
Branch 'navbar' set up to track remote branch 'navbar' from 'origin'.
```



git merge nom_branche : fusionner une branche

Imaginons que nous voulons changer la couleur de notre titre <h1> et le mettre en rouge :

1. Tout d'abord, dans Git Bash, nous allons vérifier si notre repository en local « Mondo_SHA_1 » est bien à jour avec la commande « git pull ».
2. Après cette vérification, nous nous plaçons dans notre branche secondaire ; à savoir « navbar » grâce à la commande « git checkout navbar ».
3. Ensuite, nous modifions le fichier « style.css » à l'aide de notre éditeur de code pour ajouter une couleur rouge à notre titre <h1> (h1 {color : red ;}).
4. Une fois cela fait, nous enregistrons notre travail puis nous retournons sur Git Bash. Nous tapons la commande « git merge main » pour fusionner notre travail avec la branche « main ».
5. Puis on utilise les commandes « git add . » puis « git commit -m « change h1 couleur rouge » » et enfin « git push -u origin navbar » pour réaliser notre commit sur la branche secondaire « navbar ».
6. Une fois cette opération faite, nous nous replaçons sur la branche principale « main » en tapant la commande « git checkout main ».
7. Nous utilisons ensuite la commande « git merge navbar » pour fusionner la branche secondaire (« navbar ») avec la branche principale (« main »).
8. Enfin, on réalise les commandes « git add . », « git commit -m « change h1 couleur rouge » » et pour finir « git push -u origin main » pour réaliser notre commit et finaliser cette fusion :

```
1 romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git pull
Already up to date.

2 romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git checkout navbar
Switched to branch 'navbar'
Your branch is up to date with 'origin/navbar'.

4 romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (navbar)
$ git merge main
Already up to date.

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (navbar)
$ git add .

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (navbar)
$ git commit -m "change h1 couleur rouge"
[navbar c9e520f] change h1 couleur rouge
1 file changed, 1 insertion(+), 1 deletion(-)

5 romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (navbar)
$ git push -u origin navbar
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), 310 bytes | 310.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/MasterProjectManager/Mondo_SHA_1.git
 3584d07..c9e520f navbar -> navbar
Branch 'navbar' set up to track remote branch 'navbar' from 'origin'.

6 romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (navbar)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

7 romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git merge navbar
Updating 3584d07..c9e520f
Fast-forward
 style.css | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git add .

romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git commit -m "change h1 couleur rouge"
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

8 romai@DESKTOP-8Q08HON MINGW64 ~/OneDrive/Bureau/Mondo_SHA_1 (main)
$ git push -u origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MasterProjectManager/Mondo_SHA_1.git
 3584d07..c9e520f main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Ainsi, notre branche secondaire (« navbar ») et notre branche principale (« main ») ont exactement la même modification sur le fichier « style.css » :

main2 branches0 tagsGo to fileAdd fileCode

RmNLcN06 change h1 couleur rougec9e528f 6 minutes ago17 commits

img

Ajout de fichiers au repository "Mondo_SHA_1"

7 days ago

README.md

Update README.md

6 days ago

index.html

Ajout de fichiers au repository "Mondo_SHA_1"

7 days ago

style.css

change h1 couleur rouge

6 minutes ago

navbar2 branches0 tagsGo to fileAdd fileCode

This branch is up to date with main.Contribute

RmNLcN06 change h1 couleur rougec9e528f 6 minutes ago17 commits

img

Ajout de fichiers au repository "Mondo_SHA_1"

7 days ago

README.md

Update README.md

6 days ago

index.html

Ajout de fichiers au repository "Mondo_SHA_1"

7 days ago

style.css

change h1 couleur rouge

6 minutes ago



GitHub Cheat Sheet

Travailler en équipe

Travailler avec un dépôt distant

git pull	recupérer MAJ sur dépôt distant
git pull --rebase	fusionner les branches
git merge nom_branche	fusionner la branche

Gérer les branches

git branch nom_branche	créer branche
git branch	afficher les branches
git checkout nom_branche	positionne sur la branche à modifier
git branch -d nom_branche	supprimer branche en local (ne pas se mettre sur la branche à supprimer)

Gérer les branches sur un dépôt distant

git push --set-upstream origin nom_branche	créer la branche en dépôt distant (il faut d'abord faire un " git branch nom_branche " pour la créer en local)
git branch -a	voir liste des branches sur dépôt distant
git push origin --delete nom_branche	supprimer branche sur dépôt distant

Fichiers indispensables

.gitignore	renseigner le chemin de chaque fichier que l'on ne veut pas commit. ex : dossier/fichier.txt (ignore fichier.txt) ex : dossier/* (ignore tous les fichiers à l'intérieur de dossier)
README.md	Renseigner toutes les informations utiles au projet

Trouver qui a fait quoi

git blame index.html	affiche les modifs du fichier <i>index.html</i> ligne par ligne
git blame -L 10,20 index.html	affiche les modifs de la ligne 10 à 20. (appuyer sur "q" pour quitter l'affichage)

Commit son projet sur Github

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

