

01 /12/2019

Rapport

Individuel du brief projet



ABDELABIR KHOUILID.

Sommaire :

- 1. REMERCIEMENT**
- 2. INTRODUCTION**
- 3. CONTEXTE DU PROJE**
- 4. CONCEPTION DU trois scenarios**

Scenario 1 :

- 1-Première étape : Immersion
- 2-Deuxième étape : La découverte
- 3-Troisième étape : Historique
- 4-Quatrième étape : Excluding files
- 5-Cinquième étape : Branching and Merging
- 6-Sisieme étape : Conflict Resolution
- 7-Septième étape : merge tools
- 8-Huitieme étape : Challenge
- 9-Dictionnaire des commandes git utiliser

Scenario 2 :

- 1-Etape : Tagging
- 2-Etape : stashing and saving work in progress
- 3-Etape : Voyage sur Github, Local Repo to github Repo
- 4-Etape : Mini challenge (optionnel)
- 5-Etape : Création d'une local copy
- 6-Etape : Sending the website
- 7-Etape : Fetch and pull

Scenario 3 :

- 1- Première étape : Changes on GithubImmersion
- 2- Deuxième étape : Branching and merging sur GITHUB
- 3- Troisième étape : compare pull Requests

- 4- Quatrième étape : merging en local
- 5- Cinquième étape : The Cleaning up
- 6- Sixième étape : sixth step: Rebasing
- 7- Septième étape :seventh step GitHub Insights
- 8- Huitième étape : Challenge

5. Conclusion

Introduction :

Dans le cadre de la validation des compétences de la période SAS, le brief projet est un moyen utile pour valider les compétences dans leur niveau respectif (N1, N2, N3).

La gestion des workflows sous GIT/GITHUB sera la base de notre apprentissage en terme de la gestion de projet agile.

Le but est également d'apprendre à gérer un projet professionnel, de la reformulation du cahier des charges jusqu'à sa réalisation.

Vous verrez également dans ce projet une application direct des compétences acquises pendant ces trois premières semaines de formation à youcode notamment en ce qui concerne les outils git, GitHub et Trello.

Le projet s'est déroulé en trois étapes :

La première étape consiste à Créer un tableau Trello pour la planification du Product BACKLOG et le délai de la livraison de du première scenario. Ainsi que la division des tâches entre les membres de groupe.

La deuxième étape est la réalisation du projet, nous évoquerons dans la partie conception le choix des commandes git et les résultats obtenus.

La troisième étape a été consacré pour la rédaction du rapport et la préparation d'un dictionnaire des commandes git.

Dans les ligne qui suivent nous allons détailler le développement de chacune de ces parties pour que le lecteur de ce rapport soit éclairé sur les différents étapes de ce brief projet.

Premiere etape :

```
MINGW64:/c/Users/youcode/Desktop/projects/demo
youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop (master)
$ cd projects/

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects (master)
$ mkdir demo

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects (master)
$ cd demo/

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git init
Initialized empty Git repository in C:/Users/youcode/Desktop/projects/demo/.git/

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ touch README.md

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ code README.md

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    README.md

nothing added to commit but untracked files present (use "git add" to track)

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git add .

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git commit -m "creation redme.md"
[master (root-commit) 61a50be] creation redme.md
1 file changed, 1 insertion(+)
create mode 100644 README.md

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
```

On a créé un répertoire nommée projects, un repo local sous le nom démo. Ensuite on a créé un fichier dans ce repo et on a modifier ce fichier. Et enfin le staging et le committing.

Le staging area est une sorte de zone de transit où se trouvent les fichiers modifiés qui sont pris en compte pour le prochain commit.

deuxième étape : La découverte

MINGW64:/c/Users/youcode/Desktop/projects/demo

```
youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ cd .git

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo/.git (GIT_DIR!)
$ ls -al
total 13
drwxr-xr-x 1 youcode 197121  0 nov. 26 14:35 ./
drwxr-xr-x 1 youcode 197121  0 nov. 26 14:30 ../
-rw-r--r-- 1 youcode 197121 19 nov. 26 14:34 COMMIT_EDITMSG
-rw-r--r-- 1 youcode 197121 130 nov. 26 14:29 config
-rw-r--r-- 1 youcode 197121  73 nov. 26 14:29 description
-rw-r--r-- 1 youcode 197121  23 nov. 26 14:29 HEAD
drwxr-xr-x 1 youcode 197121  0 nov. 26 14:29 hooks/
-rw-r--r-- 1 youcode 197121 137 nov. 26 14:32 index
drwxr-xr-x 1 youcode 197121  0 nov. 26 14:29 info/
drwxr-xr-x 1 youcode 197121  0 nov. 26 14:32 logs/
drwxr-xr-x 1 youcode 197121  0 nov. 26 14:32 objects/
drwxr-xr-x 1 youcode 197121  0 nov. 26 14:29 refs/

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo/.git (GIT_DIR!)
$ cd ..

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ touch Licence.md

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git add ;
Nothing specified, nothing added.
Maybe you wanted to say 'git add .'

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git add .

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git commit -m "création de fichier Licence.md"
[master 8fd3915] création de fichier Licence.md
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Licence.md

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git status
On branch master
```

Dans cette étape on a créé un fichier licence.md puis on a comité. Ensuite on doit afficher les fichiers traqués (les fichiers qui se trouvent dans la staging area).

troisième étape : Historique

```
youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git log --oneline --graph --decorate --all
* 8fd3915 (HEAD -> master) création de fichier Licence.md
* 61a50be creation readme.md

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ alias
.git/      Licence.md  README.md

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ alias
.git/      Licence.md  README.md

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ alias historique="git log --oneline --graph --all"

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ historique
* 8fd3915 (HEAD -> master) création de fichier Licence.md
* 61a50be creation readme.md

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ |
```

Git permet la définition des alias pour chaque commande afin d'éviter de taper l'intégralité de la commande, c'est un moyen qui facilite le travail sous git. Cette technique peut aussi être utile pour créer des commandes qui nous manquent.

Dans cette partie et pour se familiariser avec l'alias, on a créé un alias appelé « historique ». Ensuite on affiche la liste des alias et l'historique de commit du fichier readme.md avec l'alias.

-quatrième étape: Excluding files

```
youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ mv licence.md licence.txt

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    Licence.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        licence.txt

no changes added to commit (use "git add" and/or "git commit -a")

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git add licence.txt

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git commit -m "renommer le fichier licence.md a licence.txt"
[master 2c14640]renommer le fichier licence.md a licence.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 licence.txt

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ |
```



```

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ touch application.log

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo
(master)
$ touch gitignore

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (mas
ter)
$ vim gitignore

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (mas
ter)
$ git add .
warning: LF will be replaced by CRLF in gitignore.
The file will have its original line endings in your working
directory

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (mas
ter)
$ git commit -m " creation application.log et gitignore"
[master c17ee8e] creation application.log et gitignore
2 files changed, 1 insertion(+)
rename Licence.md => application.log (100%)
create mode 100644 gitignore

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (mas
ter)
$ mv gitignore .gitignore

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (mas
ter)
$ vim .gitignore

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (mas
ter)
$ git add .
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working
directory

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (mas
ter)
$ git commit -m "creer nouveaux fichier"
[master 6c03ca8] creer nouveaux fichier
1 file changed, 0 insertions(+), 0 deletions(-)

```

Si on veut exclure certains fichier ou répertoire du suivi par git on doit créer un fichier « .gitignore ».

Lorsqu'un fichier ou un répertoire est ignoré il ne sera pas suivi par git, signale par des commandes telles git status ou git diff ni mis en scene avec des commandes telles que git add.

fifth Step : Branching and Merging

```
youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git branch updates
```

```
youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git checkout updates
Switched to branch 'updates'

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (updates)
$ git log
* b1c1a96 (master) gdhjk
* 6c03ca8 (HEAD -> updates) creer nouveaux fichier
* c17ee8e creation application.log et gitignore
* 2c14640 renommer le fichier licence.md a licence.txt
* 8fd3915 création de fichier Licence.md
* 61a50be creation readme.md

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (updates)
$ git checkout master
Switched to branch 'master'

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git log
* b1c1a96 (HEAD -> master) gdhjk
* 6c03ca8 (updates) creer nouveaux fichier
* c17ee8e creation application.log et gitignore
* 2c14640 renommer le fichier licence.md a licence.txt
* 8fd3915 création de fichier Licence.md
* 61a50be creation readme.md

youcode@DESKTOP-18D2QB2 MINGW64 ~/Desktop/projects/demo (master)
$ git merge updates
Already up to date.
```

Ce qui permet de faire le « versionning » c'est le système de branche. Lorsque on veut essayer de nouvelle future on crée une branche pour diverger de la ligne principale de développement, si les modifications marchent bien ou peut les garder sinon on peut les abandonner

Lorsque on crée une branche, ce n'est qu'une copie du repo master, il contient le même historique de la branche principale.

Le merging permet le fusionnement entre la branche principale et celle qui contient les changements quand veut ajouter au projet.

ETAPE 6 : Conflict Resolution

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ git commit --amend
[master 046ce59] step 6.2
Date: Tue Nov 26 21:17:58 2019 +0100
1 file changed, 1 insertion(+), 2 deletions(-)

Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ historique
046ce59 (HEAD -> master) step 6.2
8be6881 (BAD) step 6.1
4ae4cc6 (updates) step 4.2
9c673ff step 4.1
5ba4fb7 step 3.2
e313edf step 3.1
a77420e second step
0f1011a first Step

Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ |
```

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ git merge BAD
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master|MERGING)
$ cat README.md
<<<<<< HEAD
troubleshooting'
=====
trouble
>>>>>> BAD

Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master|MERGING)
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
git mergetool will now attempt to use one of the following tools:
pdiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffmerge ecmerge p4m
rge araxis bc codecompare smerge emerge vimdiff
merging:
README.md

Normal merge conflict for 'README.md':
{local}: modified file
{remote}: modified file
git return to start merge resolution tool (vimdiff): merge
fichiers à éditer
```

Lorsque on fait deux modification sur le même fichier et sur la même ligne, la première sur la branche principale et la deuxième sur une autre branche (BAD), on constate qu'il est impossible de faire le fusionnement (merge).

```
git config --global --list
user.email=abdelkbirkhouilir32@gmail.com
user.name=khouilid
alias.historique=log --oneline --graph --decorate --all
```

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects (master)
$ git config --global mergetool.path C:\Program Files\Perforce/p4merge.exe
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects (master)
$ git mergetool
No files need merging
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects (master)
$ code .
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects (master)
$ |
```

Après l'installation et la configuration du logiciel « p4merge » ; outil de comparaison des fichiers ; on a pu faire le fusionnement de la branche « BAD » avec la branche master.

8 Step : Challenge

Après avoir résolu les conflits des branches ,e la création du fichier .gitignore On a rejeter les fichiers indésirables et redondants

First Step TAG

```
Youcode@DESKTOP-TTNENQC MINGW64 ~/Desktop/projects/demo (master)
$ git tag -a v1.0 -m "REALESE 1.0"

Youcode@DESKTOP-TTNENQC MINGW64 ~/Desktop/projects/demo (master)
$ git show v1.0
tag v1.0
Tagger: naima najjar <naima.najja4@gmail.com>
Date: Sun Dec 1 11:23:11 2019 +0100

REALESE 1.0

commit 195ef10878978e2926f24db2b6366b15581cd050 (HEAD -> master, tag: v1.0)
Author: naima najjar <naima.najja4@gmail.com>
Date: Sat Nov 30 22:09:06 2019 +0100

    add igno

diff --git a/.gitignore b/.gitignore
index 51d7f60..6b04722 100644
--- a/.gitignore
+++ b/.gitignore
@@ -1,2 @@
```

In this step we Create a Tag , So what is the tags :

Les tags sont des références qui pointent vers des point spécifiques de l'histoire de Git. Le marquage est généralement utilisé pour capturer un point de l'historique utilisé pour une version marquée.

Second Step STASH

Dons c'est Etape nous avons Créés un modifications et après Ça nous avons Tapez la commande git stash , so what is this command ?

Git stash stocke temporairement (ou cache) les modifications que vous avez apportées à votre copie de travail afin que vous puissiez travailler sur autre chose, puis revenir et les réappliquer plus tard.

```
Youcode@DESKTOP-TTNENQC MINGW64 ~/Desktop/projects/demo (master)
$ code .

Youcode@DESKTOP-TTNENQC MINGW64 ~/Desktop/projects/demo (master)
$ git stash
Saved working directory and index state WIP on master: 195ef10 add igno

Youcode@DESKTOP-TTNENQC MINGW64 ~/Desktop/projects/demo (master)
$ git stash list
stash@{0}: WIP on master: 195ef10 add igno

Youcode@DESKTOP-TTNENQC MINGW64 ~/Desktop/projects/demo (master)
$ git status
On branch master
nothing to commit, working tree clean
```

```
Youcode@DESKTOP-TTNENQC MINGW64 ~/Desktop/projects/demo (master)
$ code .

Youcode@DESKTOP-TTNENQC MINGW64 ~/Desktop/projects/demo (master)
$ git commit -am "modifi lincence"
[master 75164e9] modifi lincence
1 file changed, 1 insertion(+)

Youcode@DESKTOP-TTNENQC MINGW64 ~/Desktop/projects/demo (master)
$ git stash pop
Removing README_REMOTE_779.md
Removing README_REMOTE_678.md
Removing README_LOCAL_779.md
Removing README_LOCAL_678.md
Removing README_BASE_779.md
Removing README_BASE_678.md
Removing README_BACKUP_779.md
Removing README_BACKUP_678.md
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md
```

Et après la commande git stash , we need to return our modifications , Dons pour fait ça nous avons tapez la commande Git stash pop

3: third step

Donc pour push les tags nous avons tapé la commande `git push -u origin master -tag`

Step 4 : ssh and http

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo/Monsiteweb  
$ mkdir .ssh
```

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo/Monsiteweb  
$ cd .ssh/
```

En examinant les types d'authentification sur GITHUB, on tombe sur l'authentification HTTPS et SSH, certes HTTPS est beaucoup plus facile à manager tandis que le SSH est plus sécurisé tandis que fiable en ce qui concerne les transferts cryptés. Le but de ce challenge est de créer une authentification SSH entre votre repo local et le repo GITHUB.

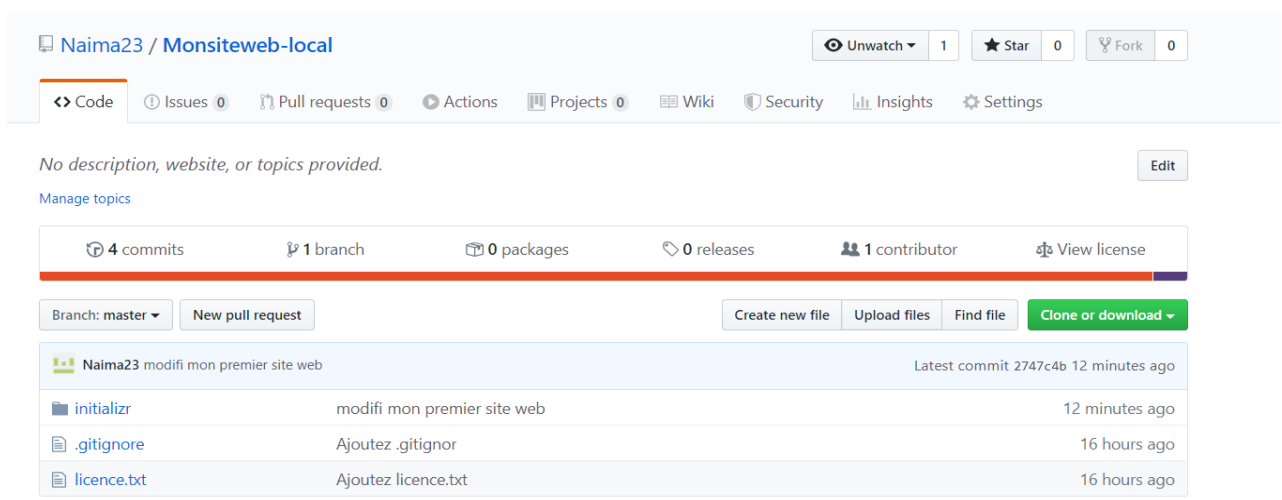
```
Youcode@DESKTOP-TTNNQC MINGW64 ~/Desktop/projects/demo (master)  
$ git remote add origin https://github.com/Naima23/repo-Scen2.git  
  
Youcode@DESKTOP-TTNNQC MINGW64 ~/Desktop/projects/demo (master)  
$ git push -u origin master --tag  
Enumerating objects: 31, done.  
Counting objects: 100% (31/31), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (21/21), done.  
Writing objects: 100% (31/31), 2.91 KiB | 229.00 KiB/s, done.  
Total 31 (delta 5), reused 0 (delta 0)  
remote: Resolving deltas: 100% (5/5), done.  
To https://github.com/Naima23/repo-Scen2.git  
* [new branch]      master -> master  
* [new tag]         v1.0 -> v1.0  
Branch 'master' set up to track remote branch 'master' from 'origin'.
```



The screenshot shows a GitHub repository interface for 'Naima23 / Monsiteweb-local'. The 'Edit file' tab is active, displaying an HTML document. The code includes a doctype, conditional comments for IE, a meta charset of 'utf-8', a meta http-equiv for 'X-UA-Compatible', a title 'mon premier site web', and various meta tags for description, viewport, and apple-touch-icon. It also includes a link to a Bootstrap stylesheet and a CSS style block for the body with padding-top: 50px and padding-bottom: 20px.

5: Fifth step

Sur c'est étape nous avons Créés une nouveau nommes MONSITEWEB-LOCA



The screenshot shows the GitHub repository page for 'Naima23 / Monsiteweb-local'. The repository has 1 commit, 1 branch, 0 packages, 0 releases, and 1 contributor. The 'Code' tab is selected, showing the repository's structure with files like 'initializr', '.gitignore', and 'licence.txt'. The 'Clone or download' button is visible.

And we clone it inton our local repo

```
Youcode@DESKTOP-TTNENQC MINGW64 ~/Desktop/projects/demo (master)
$ git clone https://github.com/Naima23/Monsiteweb-local.git
Cloning into 'Monsiteweb-local'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
```


7: seventh ste

Dons c'est etape nous avons fait quelque modifications dans le fichier README.md ,
Et apres Ça nous avons push tout into our repo github .

```
Youcode@DESKTOP-TTNNQC MINGW64 ~/Desktop/projects/Monsiteweb-local (master)
$ echo "modifi readme" >> README.md

Youcode@DESKTOP-TTNNQC MINGW64 ~/Desktop/projects/Monsiteweb-local (master)
$ git add .
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

Youcode@DESKTOP-TTNNQC MINGW64 ~/Desktop/projects/Monsiteweb-local (master)
$ git add .

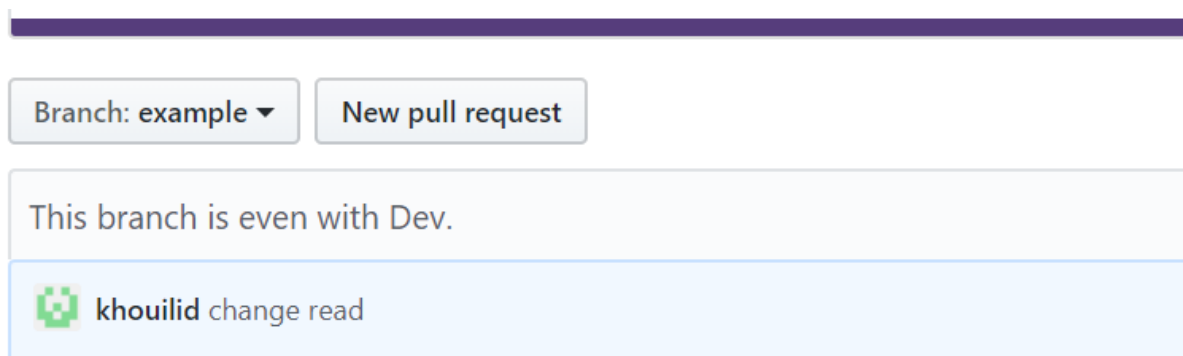
Youcode@DESKTOP-TTNNQC MINGW64 ~/Desktop/projects/Monsiteweb-local (master)
$ git commit -m "modifier readme"
[master ec2bfe4] modifier readme
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

pe 1 : : Changes on GithubImmersion

In first time , nous avons fait la modification dans les fichiers CSS et index.html . On ajoute la ligne `<h1>` modification récente `</h1>` , et on a renommé 404.html to error404.html .

And in our local Repo we do the verification with git status command , Et finalement après git pull on a fait la verification final .

Etape 2 : Branching and merging sur GITHUB



On premier on a créé la branche Exemple , et on a fait la modification dans le fichier README.md et nous avons fait le commit « Toujours sur Github »

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ git branch annulation

Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ git checkout annulation
Switched to branch 'annulation'

Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (annulation)
$ code .
```

- 1- . In our local Repo we créés une branch nommer ANNULATION.
- 2- And we delete la balise H1 dans le fichier index.html .

```

youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (annulation)
$ git show
commit 7b19edfa4575f1ef96ccd3c9e572140c0f08cc01 (HEAD -> annulation)
Author: khouilid <abdelkbirkhouilid32@gmail.com>
Date:   Fri Nov 29 14:37:31 2019 +0100

```

cod change

```


diff --git a/index.html b/index.html
index 403ae82..114cf11 100644
--- a/index.html
+++ b/index.html
@@ -19,8 +19,7 @@
         <div class="header-container">
             <header class="wrapper clearfix">
-                <h1 class="title">modification
- récente</h1>
+
                 <nav>
                     <ul>
                         <li><a href="#">nav ul li a</a></li>


```

- 3- Et finalement we valide thi modification and we show it with git show command .
- 4- In the and we push it to our Repo sur githut .

Etape 3 : compare pull Requests

Your recently pushed branches:

 example (less than a minute ago)

 Compare & pull request

After la pull request

your branches

example

Deleted just now by khouilid

We delete the branch

Etape 4: merging en local

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (ajustement)
$ git checkout ajustement
Already on 'ajustement'
```

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (ajustement)
$ code .
```

Apers la modification dans le fichier main.css ,On fait le commit et le tagging en seule commande ,and we push it into github

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ git merge ajustement
hint: Waiting for your editor to close merge made by the 'recursive' strategy.
css/main.css | 4 ++++
index.html   | 2 +-
2 files changed, 5 insertions(+), 1 deletion(-)
```

After a modification we merged with master branch .

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ git pull
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 33% (1/remote: Compressing objects: 66% (2/remote: Com
remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/khouilid/YOUCODE
   b72fdee..a1ea945  master      -> origin/master
   * [new branch]   Example     -> origin/Example
Already up to date!
hint: Waiting for your editor to close merge made by the 'recursive' strategy.

Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ git merge ajustement
hint: Waiting for your editor to close merge made by the 'recursive' strategy.
css/main.css | 4 ++++
index.html   | 2 +-
2 files changed, 5 insertions(+), 1 deletion(-)
```

A mon avis the best merge is the manual merge .

```

Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ git branch -a
* master
  remotes/origin/Example
  remotes/origin/ajustement
  remotes/origin/annulation
  remotes/origin/master

Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ git pull --all
Fetching origin
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 25% (1/remote: Compressing objects: 5
remote: Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), done.
From https://github.com/khouilid/YOUCODE
   e03689c..82f5e17  master      -> origin/master

```

- 1- Nous avons vérifié notre modification avec Git STATUS and we push .
- 2- After we delete ajustement branch .
- 3- Nous avons tapé la commande GIT BRANCH -A pour show all the branches : local et also sur github .
- 4- Et finalement , we delete all it we just keep master branch .

Etape 5 :The Cleaning up

```

Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ git pull --all
Fetching origin
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 25% (1/remote: Compressing objects:
remote: Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), done.
From https://github.com/khouilid/YOUCODE
   e03689c..82f5e17  master      -> origin/master
   * [new branch]      updatellicence -> origin/updatellicence
Updating e03689c..82f5e17
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
$ git merge updatellicence
merge: updatellicence - not something we can merge

```

Did you mean this?

Pour Show all branches nous avons tapez la commande git pull --all , and push after we merge it

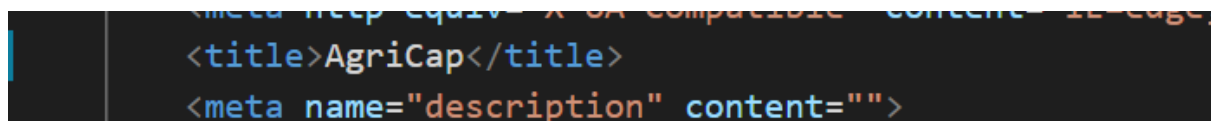
Étape 6 : sixth step: Rebasing

Nous avons fait la modification sur README.md dans github .
Et sur notre REPO local we delete this line :

```
<!--[if lt IE 7]> <html class="no-js lt-ie9 lt-ie8 lt-ie7"
lang=""> <![endif]-->
<!--[if IE 7]> <html class="no-js lt-ie9 lt-ie8"
lang=""> <![endif]-->
<!--[if IE 8]> <html class="no-js lt-ie9" lang=""> <![endif]-->

<!--[if gt IE 8]><!-->
```

And we change our title to AgriCap :

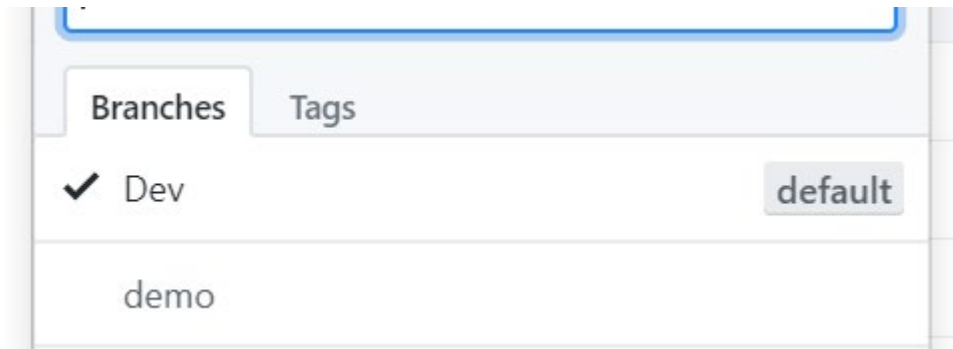


```
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<title>AgriCap</title>
<meta name="description" content="">
```

In this paragraphe we must explain what git rebase do ,

First git rebase command is a commande for merging , Most visibly, rebase differs from merge by rewriting the commit history in order to produce a straight, linear succession of commits.

Etape 7 : GitHub Insights



On premier nous avons créés une branch « DEV », and we make it en mode par default,
Et nous avons créés un autre branche nommes DEMO , et avant de faire un pull request sur
le branche DEMO , we editez README.md fil and we do some modification .

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo (master)
id/YOUCODE.githttps://github.com/khouili
Cloning into 'YOUCODE'...
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (53/53), done.
remote: Total 95 (delta 28), reused 80 (delta 24), pack-reused 0
Unpacking objects: 100% (95/95), done.
```

After we delete all our website fils from local Repo ,
We clone it from github Repo , t nous avant vérifiez le contenu .

```
Youcode@DESKTOP-QB8DP93 MINGW64 ~/Desktop/projects/demo/YOUCODE (Dev)
$ ls
css/  error404.html  index.html  js/  Licence.txt  Monsiteweb/  README.md
```

Et on fin Toutefois la branche par défaut est : DEV

Conclusion :

Ce brief projet n'a pas été aussi difficile du fait que ce dernier a pour objectif la familiarisation avec les commandes git qui est un outil incontournable pour tout développeur.

S'il y'a un point qu'il fallait réussir tout de même c'est le travail en groupe et le respect de délai de livraison du projet.

Nous avons durant la réalisation de ce projet appliqué directement les connaissances acquises pendant ce mois de formation a youcode. Nous avons fait beaucoup de recherche pour atteindre nos objectifs.