

01 /12/2019

Rapport

Indivudial du brief projet



Essaid Yassir.

**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

**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. Sisieme étape : sixth step: Rebasing
7. Septième étape :seventh step GitHub Insights
8. Huitieme étape : Challenge
9. **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 taches entre les membres de groupe.

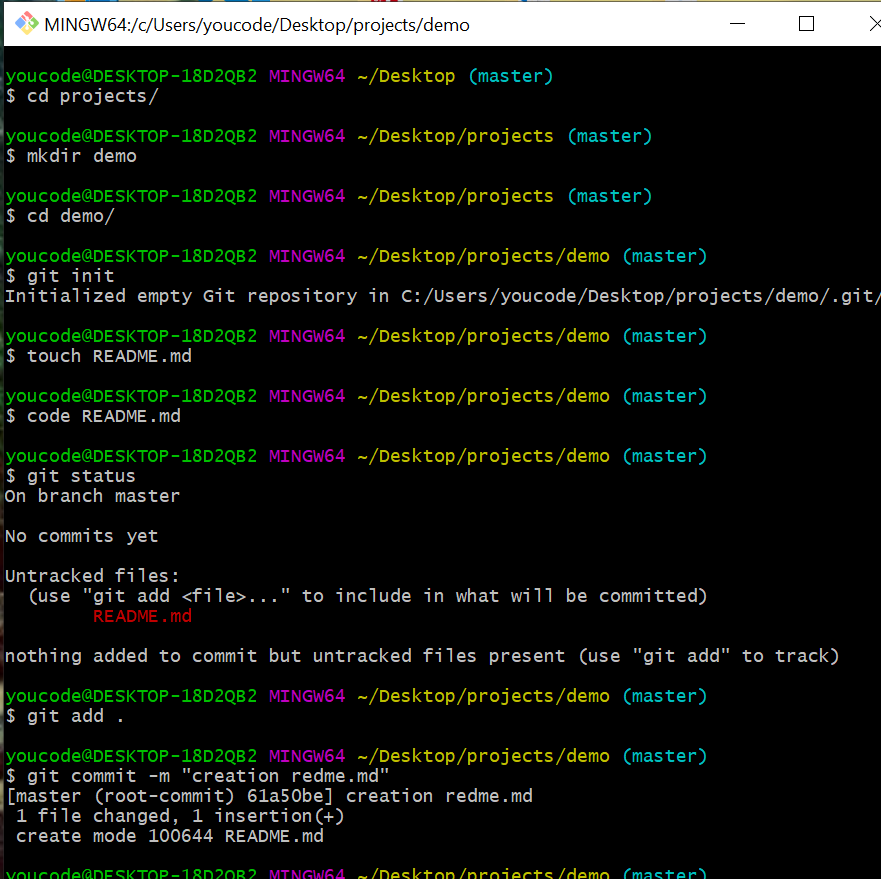
La deuxième étape est la réalisation du projet, nous évoquerons dans la partie conception le choix des commandes git et les résultats obtenus.

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.

**Première étape :**

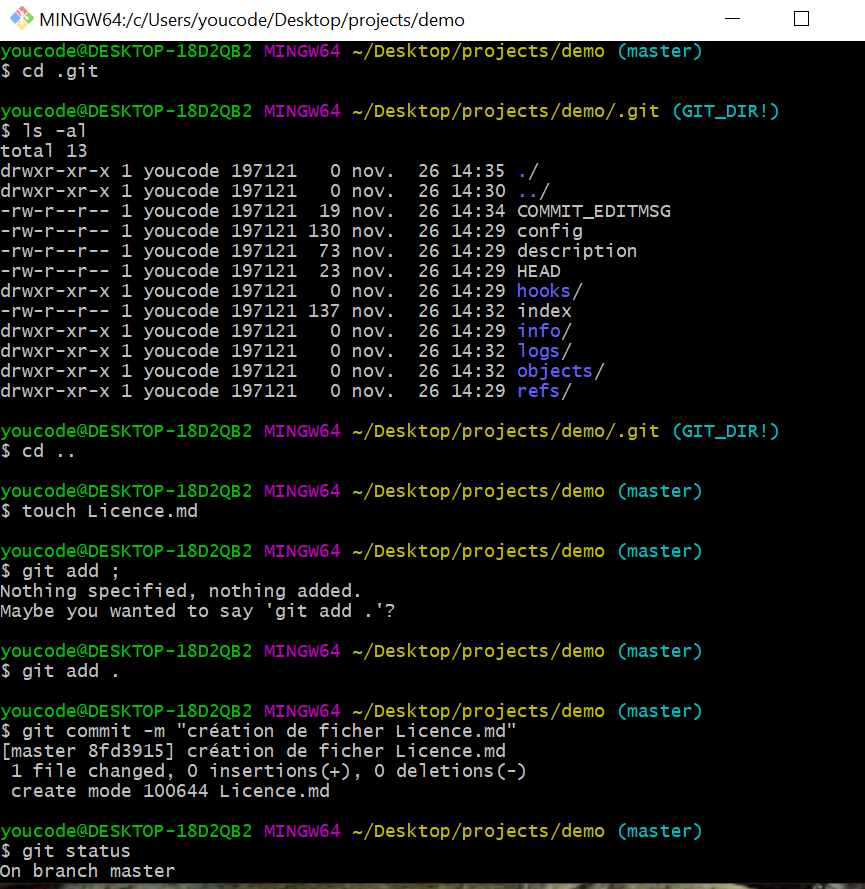
Git est un système de contrôle de version permettant de suivre l'évolution des fichiers informatiques et de coordonner le travail sur ces fichiers entre plusieurs personnes.

 La fonction du fichier caché .git est traque les modifications dans le repo.

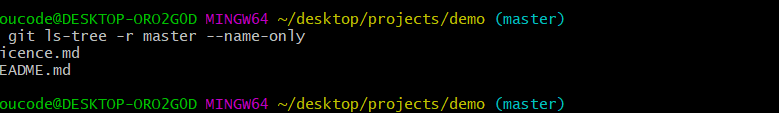
On a créé un répertoire nommée projects, un repo local sous le nom démo. Ensuite on a créé un fichier dans ce repo et on a modifier ce fichier. Et enfin le staging et le commiting.

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

**Deuxième étape : La découverte**

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 trouve dans la staging area).

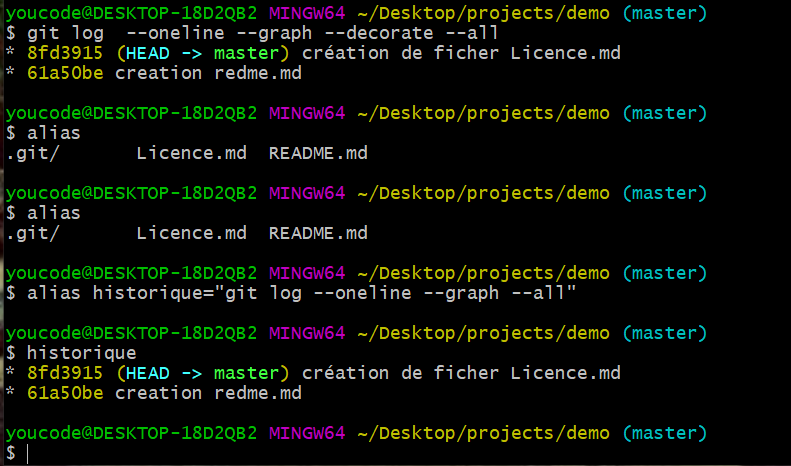
Afficher les fichiers traqués :



Explication des clauses suivantes : HEAD, LOGS, BRANCHES :

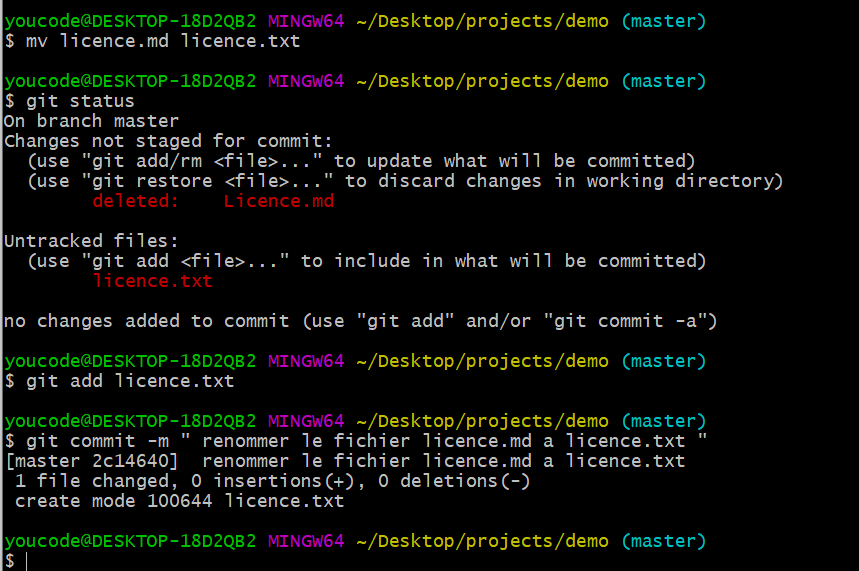
● Logs : endroit contenant les commits effectués et sa commande permet de les afficher.

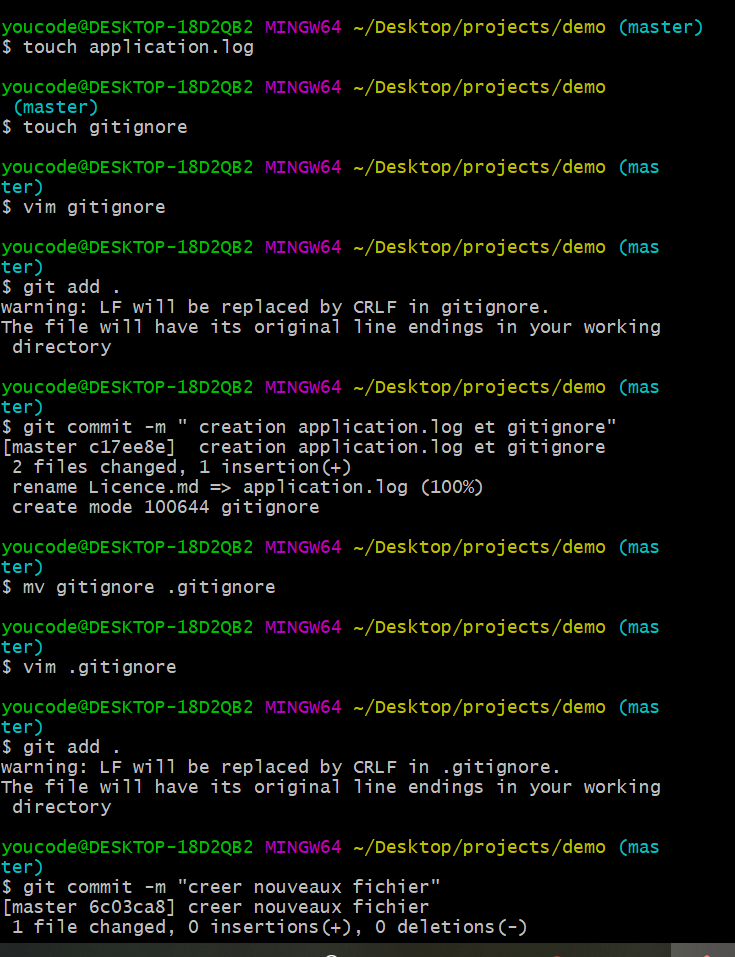
● Branches : endroit contenant les branches effectuées

● Head : le commit en cour où se trouve votre répo. Souvent le HEAD pointe vers le dernier commit dans notre branche en cour. En gros HEAD veut dire: vers quoi mon répo pointe actuellement**Troisième étape : Historique**

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 ». Ensuit on affiche la liste des alias et l’historique de commit du fichier readme.md avec l’alias.

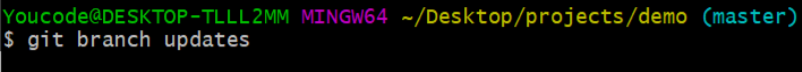
**-Quatrième étape: Excluding files**

****

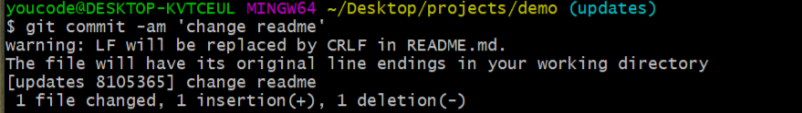
Si on veut exclure certains fichier ou répertoire du suivi par git on doit créer un fichier « .gitignore ».

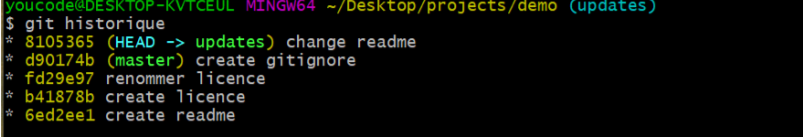
Lorsqu’un fichier ou un répertoire est ignoré il ne sera pas suivi par git, signale par des commandes telles git status ou git diff ni mis en scène avec des commandes telles que git add.

**fifth Step : Branching and Merging**

****

Apres création branche updates on fait le commit

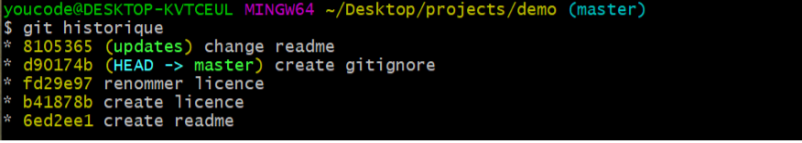




Les commits de la branche master apparaissent tous sur la branche updates en plus du commit effectué sur cette nouvelle branche.

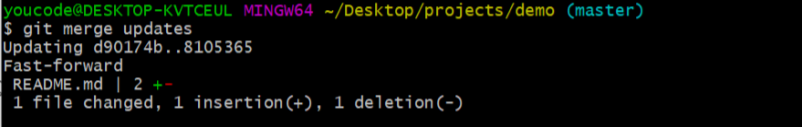
On peut visualiser sur cet historique dans quelle branche le commit a été effectué.

On retourne vers la branche ‘master’ et afficher l’historique

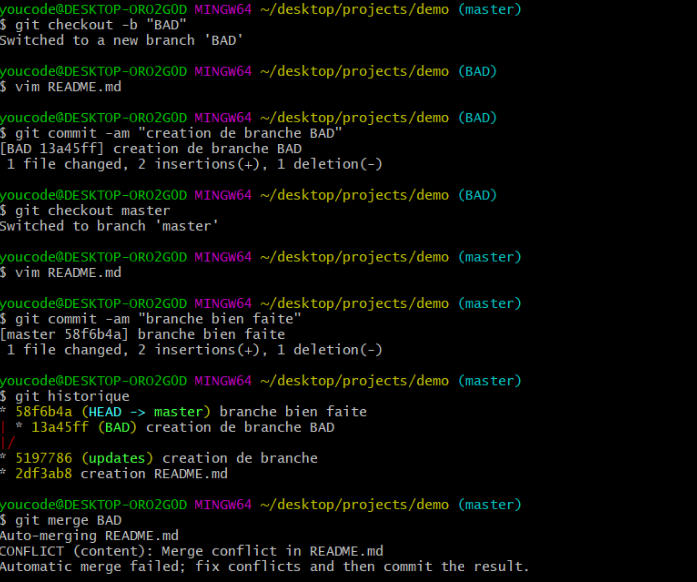


Dans l’historique on nous indique sur quelle branche on est positionné, et la branche où le commit a été créé depuis l’ajout d’une nouvelle branche.

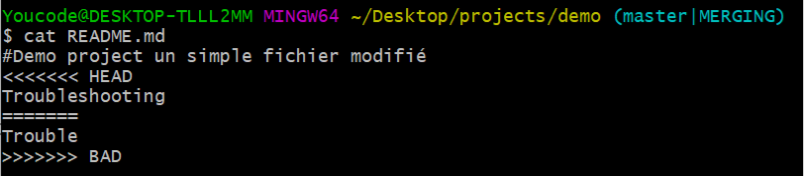
Le merge :



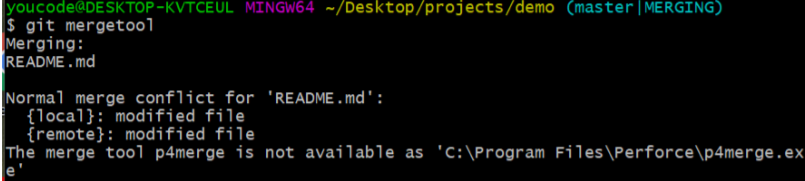
**ETAPE 6: Conflict Resolution**



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).



On remarque que les deux modifications sont séparées par un trait donc on peut conclure que ça confirme que le merge n’est pas vraiment passé.

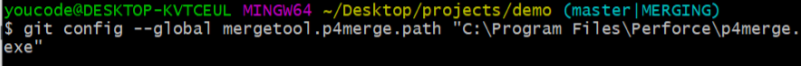


On obtient un conflit de merge avec le message disant que l’outil p4merge n’est pas disponible donc on devrait l'installer pour l’utiliser pour le mege.

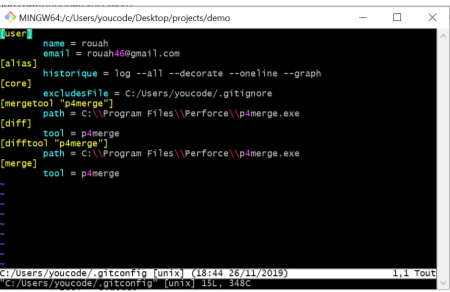
**7 Step : merge tools**

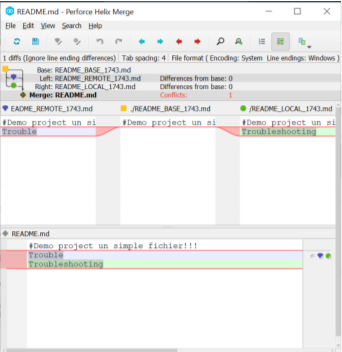
L’installation de p4merge sur notre PC

**C:\Users\youcode\Pictures\git senario\rapport\Annotation 2019-12-04 111927.png**

****

**C:\Users\youcode\Pictures\git senario\rapport\Annotation 2019-12-04 112008.png**

****

****

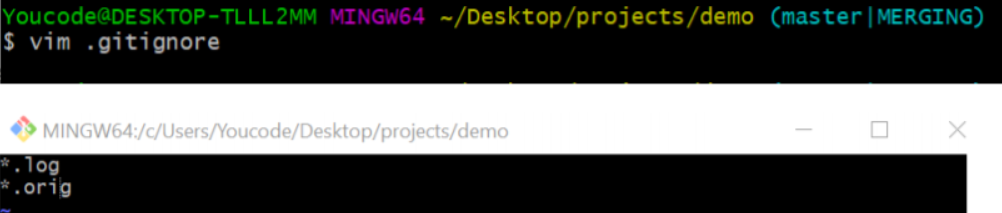
Analyser et expliquer la plateforme ouverte en temps réel :

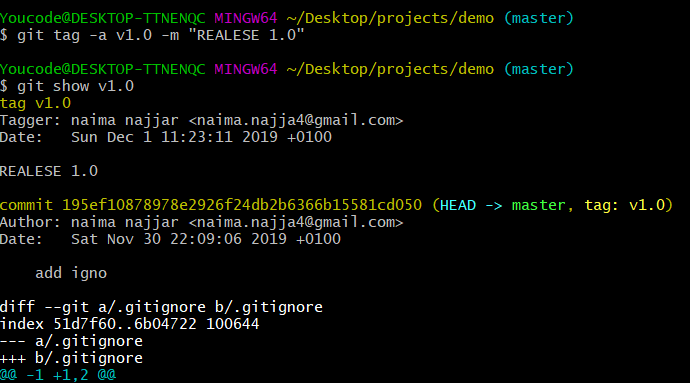
On obtient la plateforme ci-dessus qui affiche en haut à gauche la modification apportée à la branche master, la partie en haut à droite représente la branche BAD. Celle du milieu est celle où devrait se trouver le merge des deux modifications. La partie du bas et celle qu'on peut modifier manuellement pour résoudre le conflit du merge.

**8 Step : Challenge**

Après avoir résolu les conflits des branches et la création du fichier

‘. gitignore’ On a rejeter les fichiers indésirables et redondants.



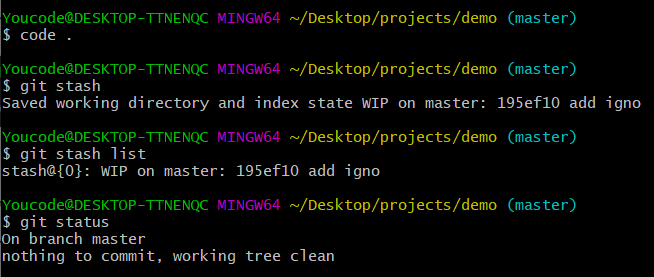
 First Step TAG

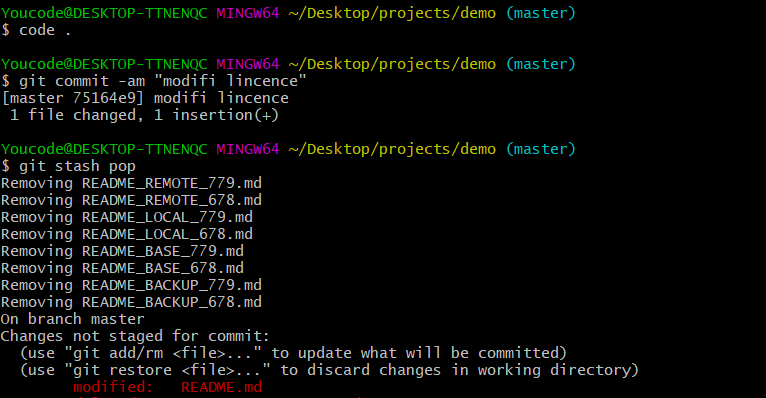
Les tags sont des références qui pointent vers des points spécifiques de l’histoire de Git. Le marquage est généralement utilisé pour capturer un point de l’historique utilisé pour une version marquée.

Second Step STASH

Dons c’est Etape nous avons Crées une modification et après Ça nous avons Tapez la commande git stash.

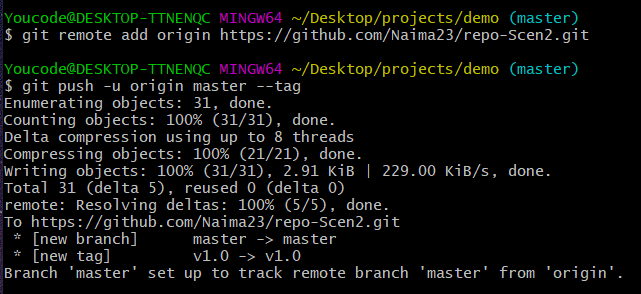
Le fonctionnement de git stash :

 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.

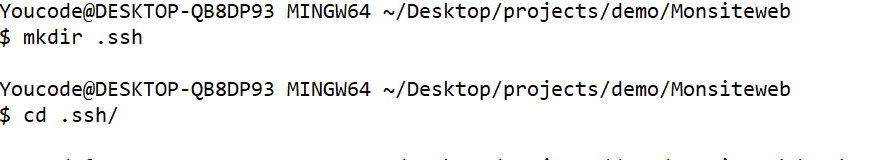


Et après la commande git stash , we need to return our modifications , Dons pour fait ça nous avons tapez la commande Git stash pop

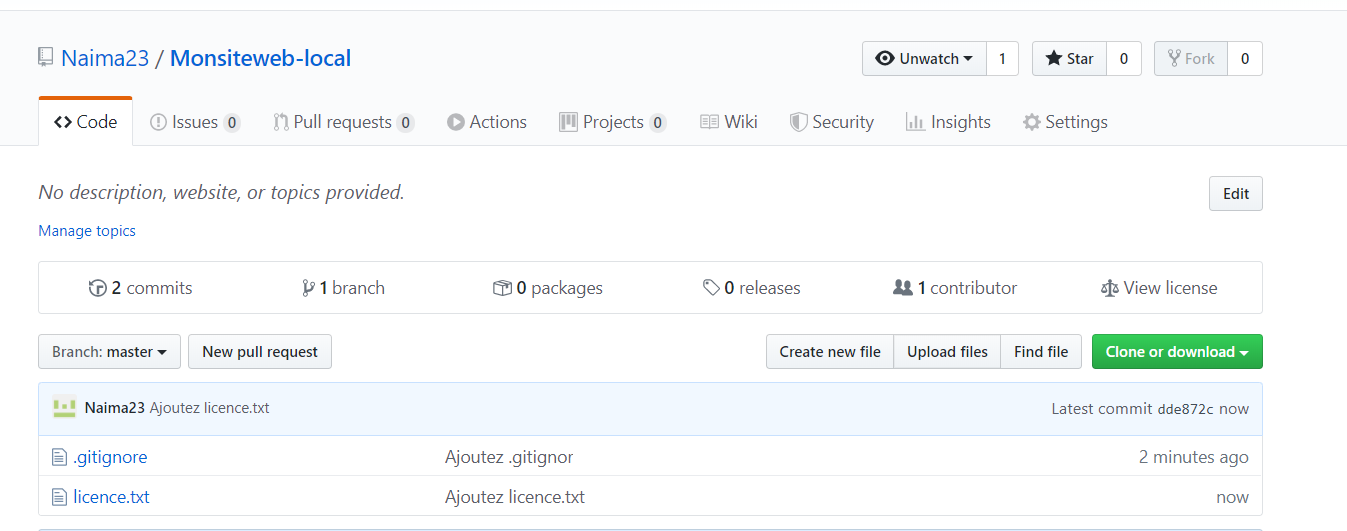
3: third step

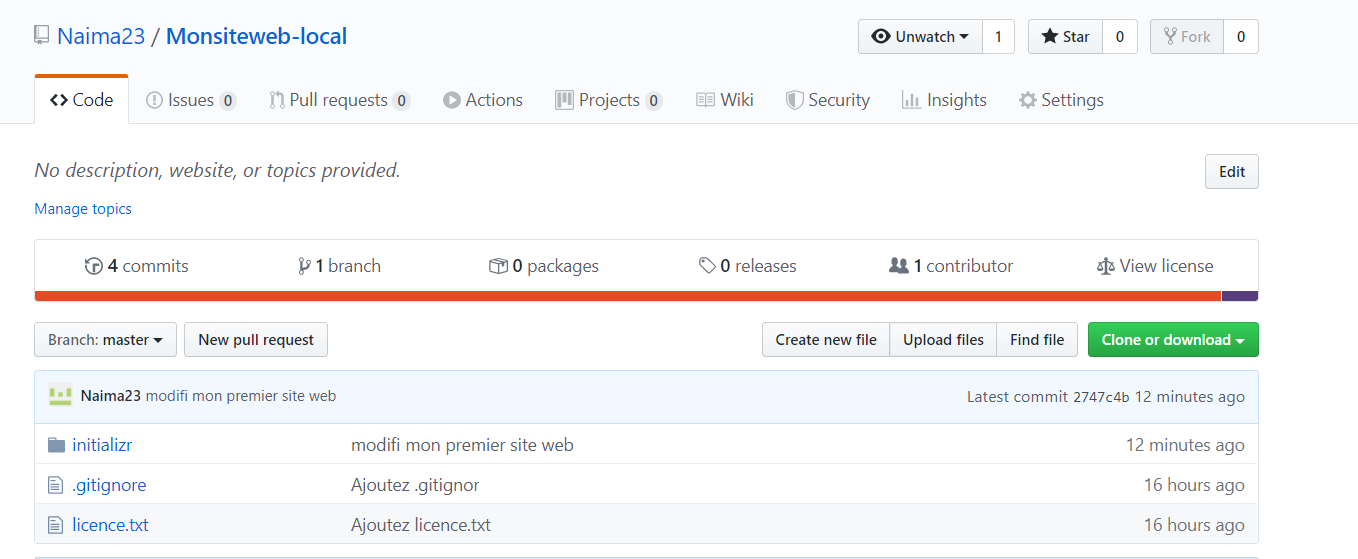


Donc pour push les tags nous avons tapez la commande git push –u origin master –tag

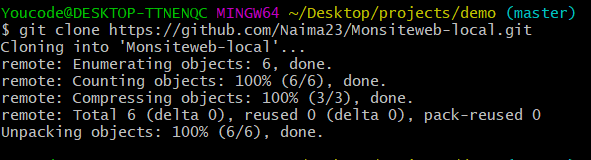
Step 4 : ssh and http

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.

 5: Fifth step

 Sur c’est étape nous avons Crées une nouveau nommes MONSITEWEB-LOCA

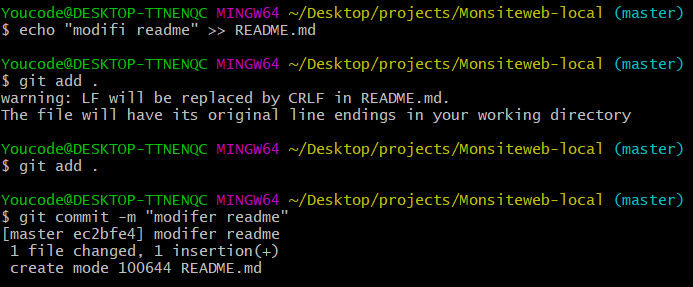
And we clone it inton our local repo

****

7: seventh ste

Dons c’est etape nous avons fait quelque modifications dons le fichier README.md ,

Et apres Ça nous avons push tout into our repo github .



**pe 1** : : Changes on GithubImmersion

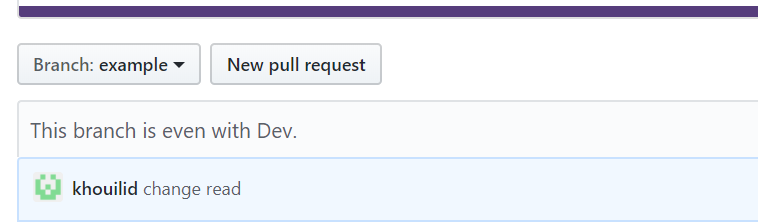
In first time , nous avons fait la modification dons les fichier CSS et index.html . On ajoute la line <h1> modification

récente </h1> , et on a renommer 404.html to error404.html .

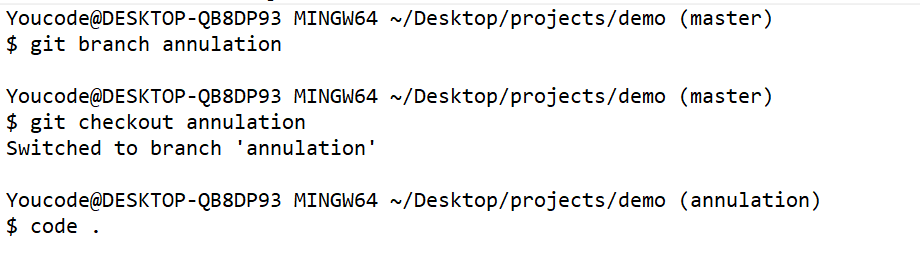
And in our local Repo we do the verification with git status command ,

Et finalemment après git pull on a fait la verification final .

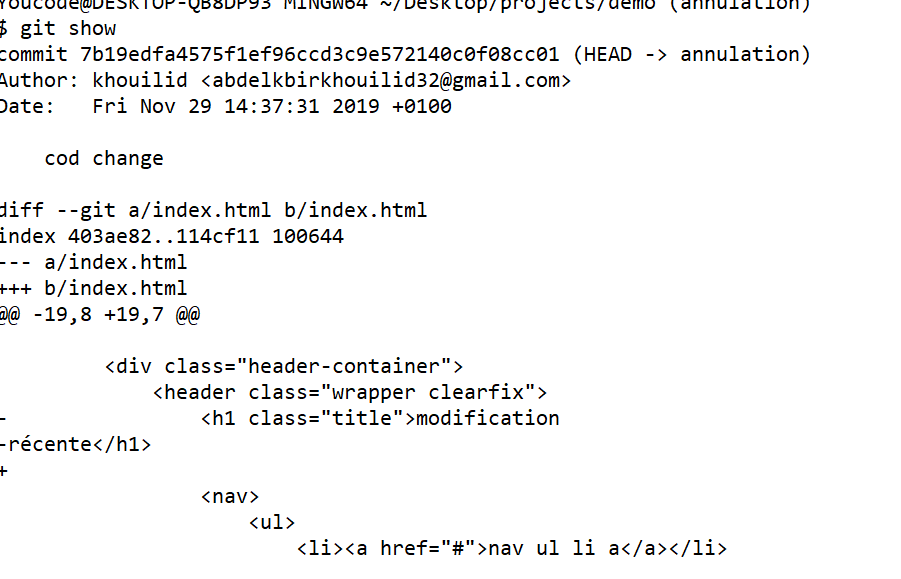
Etape 2 : Branching and merging sur GITHUB



On premair on a crées la branche Exemple , et ona fiat la modification Dans la fichier README.md et nous avans fiar the commit « Toujours sur Github »

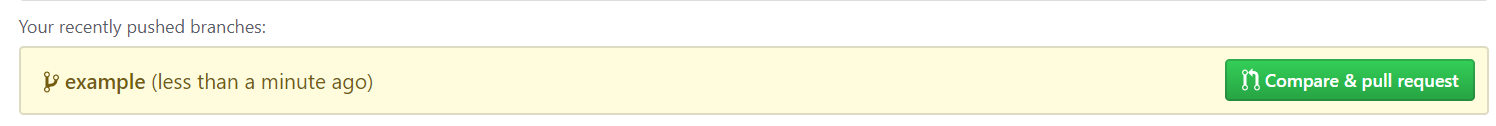


1. . In our local Repo we crées une branch nommer ANNULATION.
2. And we delete la balise H1 dans la fichier index.html .

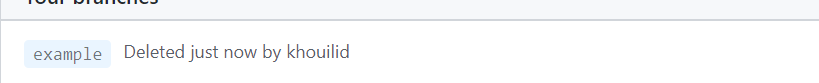


1. Et finalemment we valide thi modification and we show it with git show command .
2. In the and we push it to our Repo sur githut .

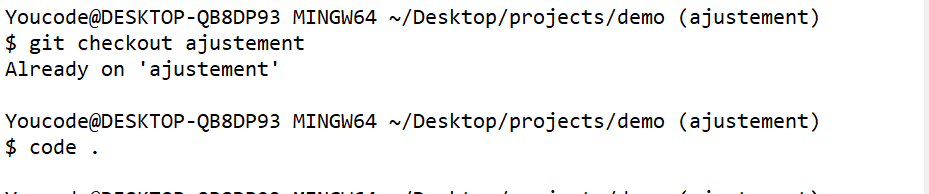
Etape 3 : compare pull Requests



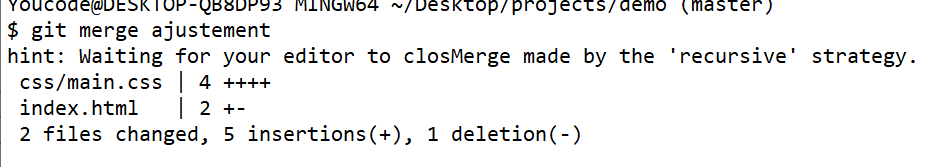
After la pull request



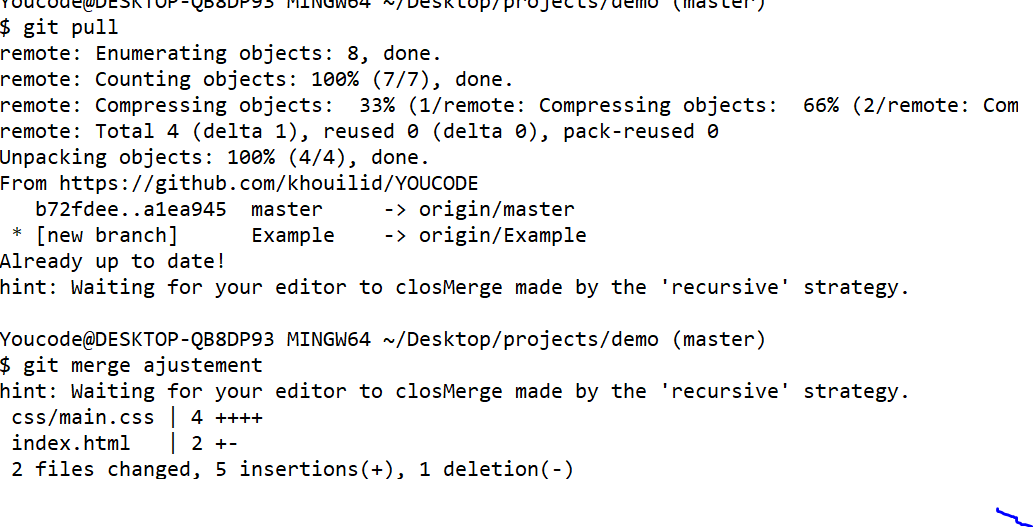
We delete the branch

 Etape 4: merging en local

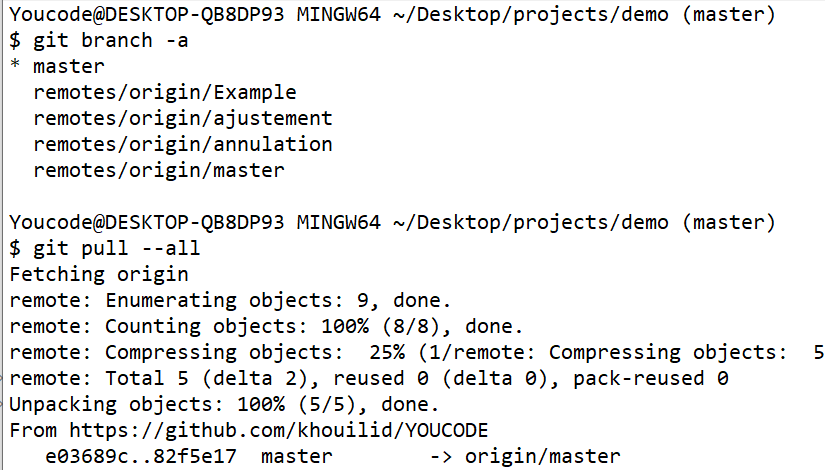
Apers la modification dons le fichier main.css ,On fait le commit et le stagging en seule commande ,and we push it into github



Aftar a modification we merged with master branch .



A mon avis the best merge is the manual mrege .

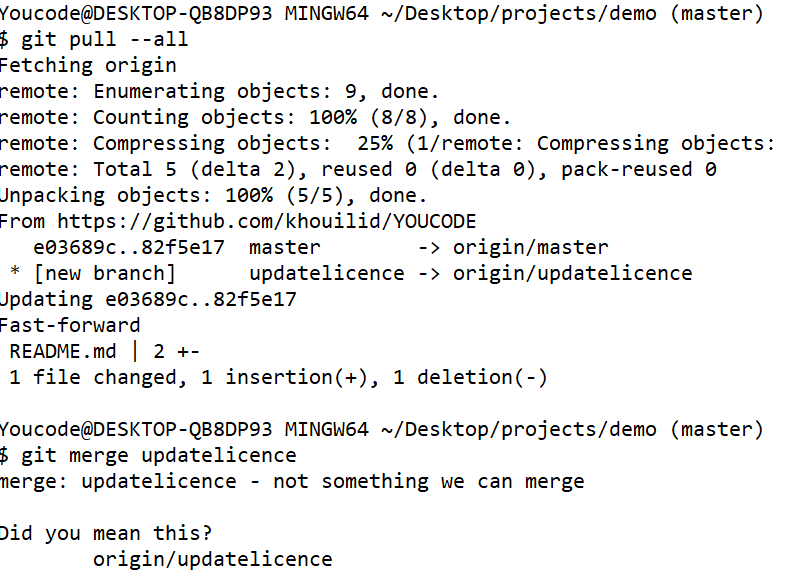


# Nous avons vérifer notre modification avec GiT STATUS and we push .

# After we delete ajustement branch .

# Nous avons tapez la commande GIT BRANCH –A pour show all the branchs : local et also sur gihub .

# Et finalemment , we delete all it we just keep master branch .

Etape 5 :The Cleaning up

# Pour Show all branchs nous avons tapez la commande git pull –all , and push after we merge it

Étape 6 : sixth step: Rebasing

# Nous avons fiar la modification sur README.md dons github .

# Et sur notre REPO local we delete this line :

<!--[if lt IE 7]> <html class="no-js lt-ie9 lt-ie8 ltie7"

lang=""> <![endif]-->

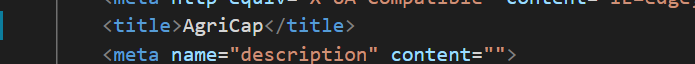
<!--[if IE 7]> <html class="no-js lt-ie9 ltie8"

lang=""> <![endif]-->

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

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

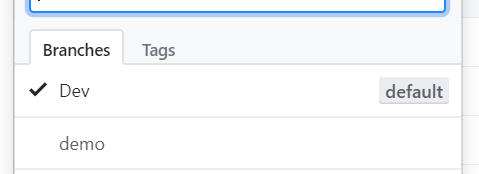
And we change our title to AgriCap :



In this paragraphe we must explaine what git rebase do ,

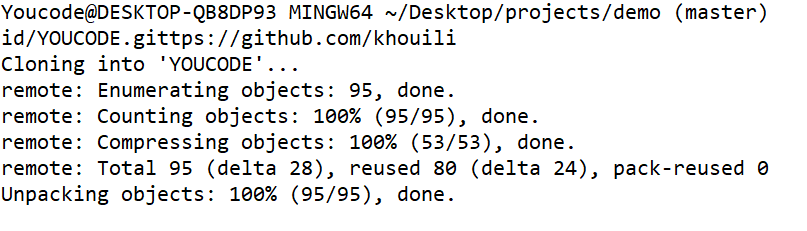
First git rebase command is a commande fir merging , Most visibly, rebase differs from merge by rewriting the commit history in order to produce a straight, linear succession of commits.

Etape 7 : GitHub Insights



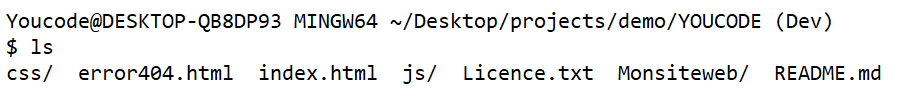
On premier nous avons crées une branch « DEV » , and we make it en mode par defaut,

Et nous avons crées 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 .



After we delete all our website fils from local Repo ,

We clone it from github Repo , t nous avant vérifez le contenu .



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.