

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.

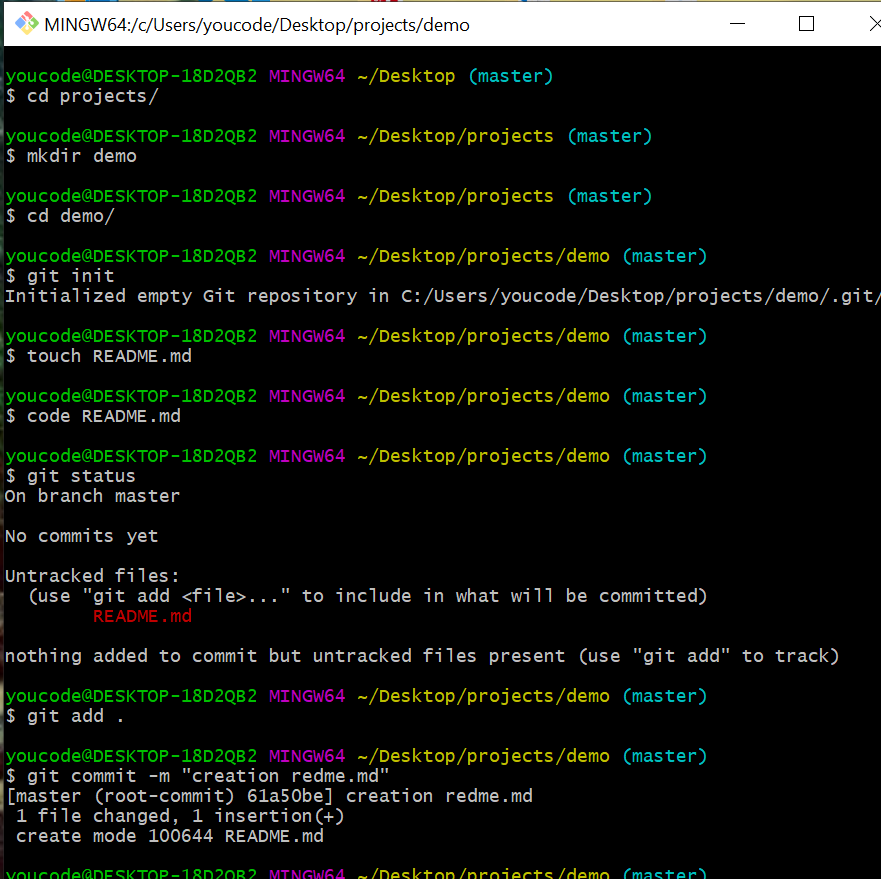
La troisième étape a été consacré pour la rédaction du rapport et la préparation d’un dictionnaire des commandes git.

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.

**Scenario 1**

**Première étape :**

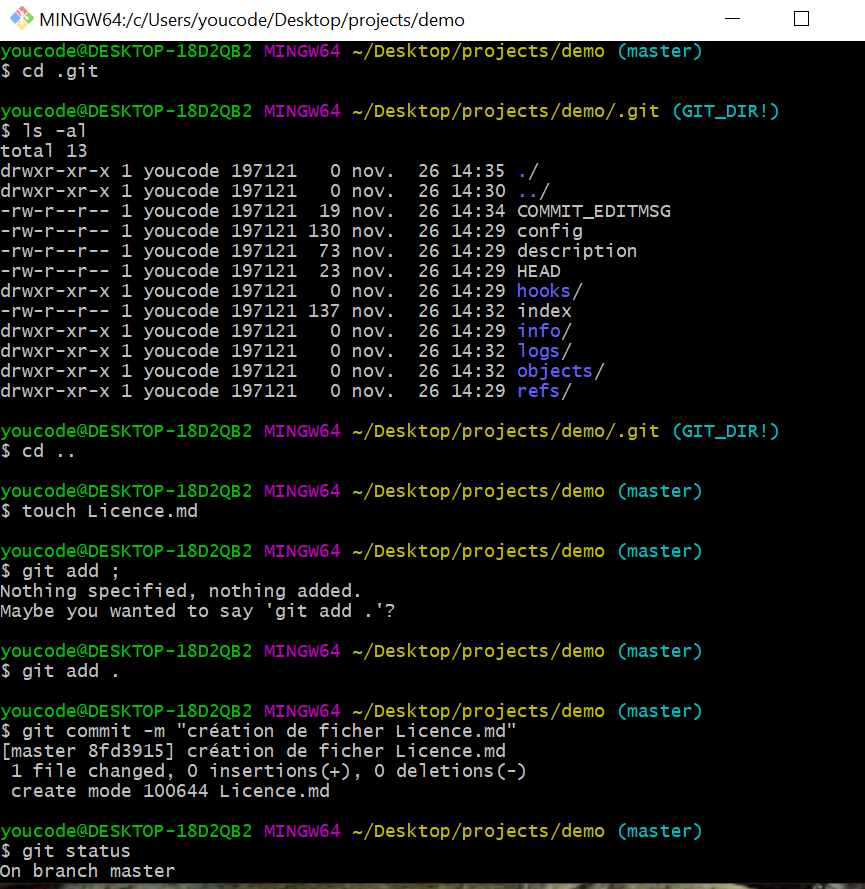
Git est un système de contrôle de version permettant de suivre l'évolution des fichiers informatiques et de coordonner le travail sur ces fichiers entre plusieurs personnes.

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

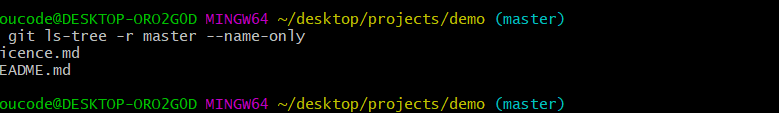
On a créé un répertoire nommée projects, un repo local sous le nom démo. Ensuite on a créé un fichier dans ce repo et on a modifier ce fichier. Et enfin le staging et le commiting.

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).

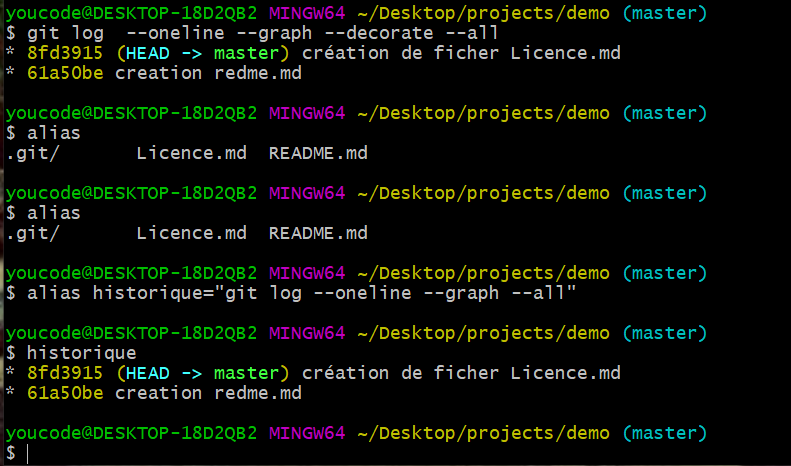
Afficher les fichiers traqués :



Explication des clauses suivantes : HEAD, LOGS, BRANCHES :

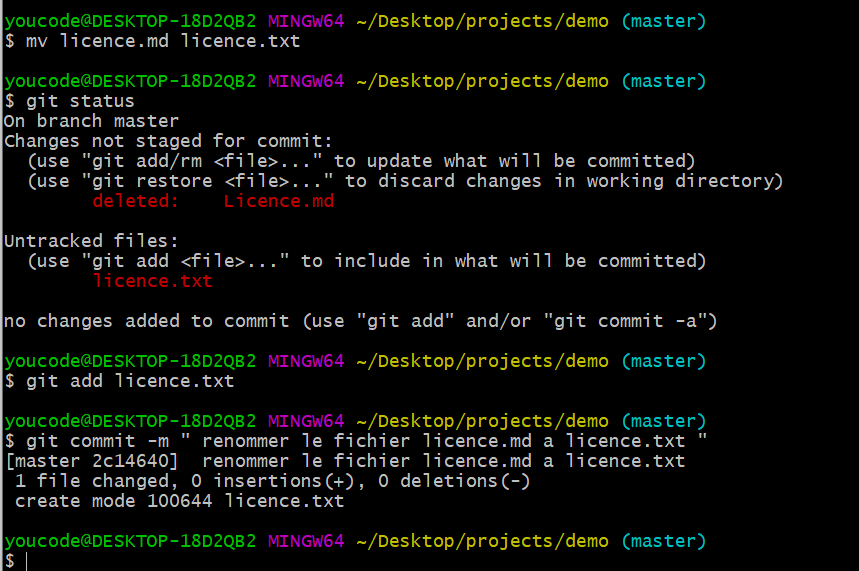
● Logs : endroit contenant les commits effectués et sa commande permet de les afficher.

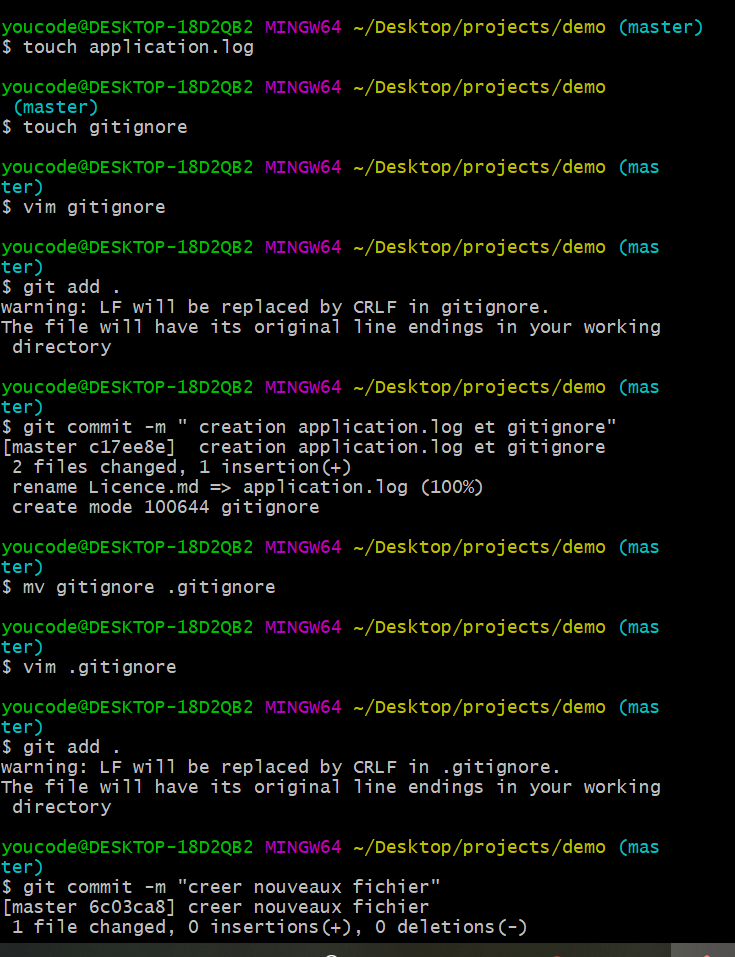
● Branches : endroit contenant les branches effectuées

● Head : le 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.

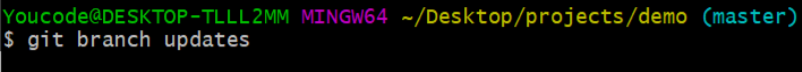
**-Quatrième étape : Excluding files**

****

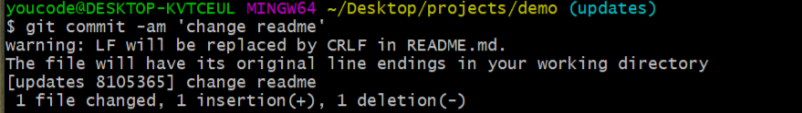
Si on veut exclure certains fichier ou répertoire du suivi par git on doit créer un fichier « .gitignore ».

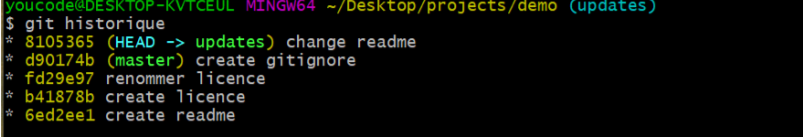
Lorsqu’un fichier ou un répertoire est ignoré il ne sera pas suivi par git, signale par des commandes telles git status ou git diff ni mis en scène avec des commandes telles que git add.

**fifth Step : Branching and Merging**

****

Apres création branche updates on fait le commit

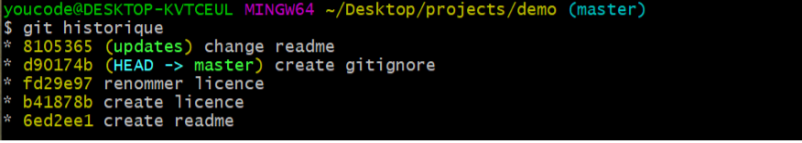




Les commits de la branche master apparaissent tous sur la branche updates en plus du commit effectué sur cette nouvelle branche.

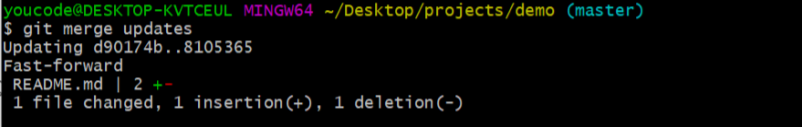
On peut visualiser sur cet historique dans quelle branche le commit a été effectué.

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

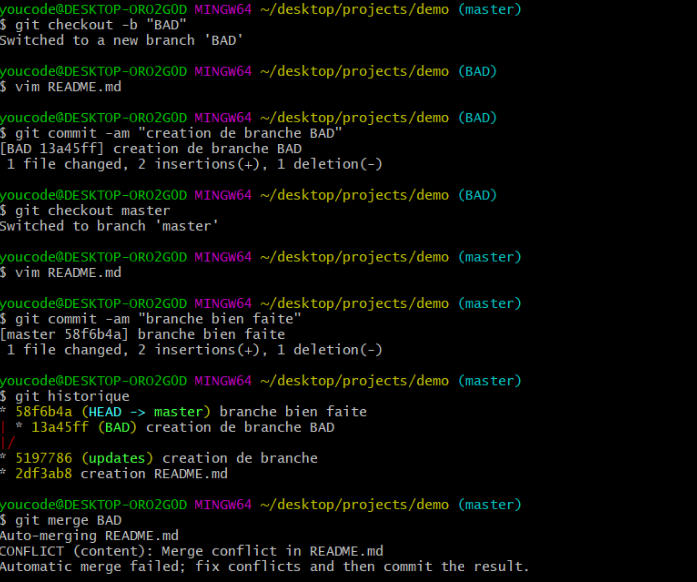


Dans l’historique on nous indique sur quelle branche on est positionné, et la branche où le commit a été créé depuis l’ajout d’une nouvelle branche.

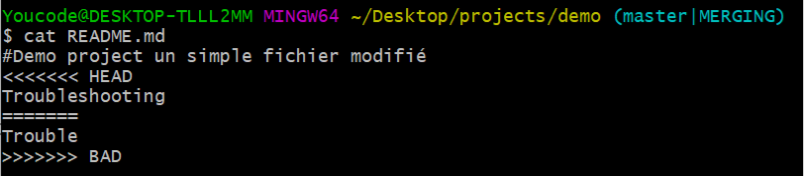
Le merge :



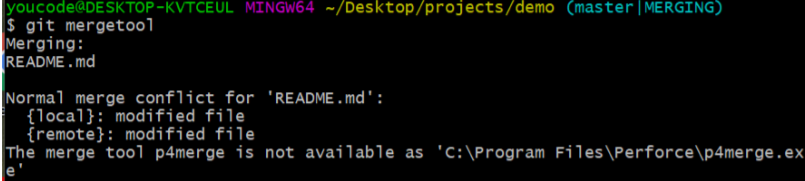
**ETAPE 6: Conflict Resolution**



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.

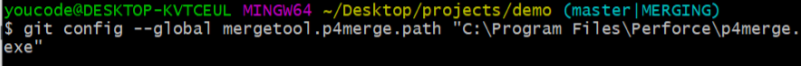


On obtient un conflit de merge avec le message disant que l’outil p4merge n’est pas disponible donc on devrait l'installer pour l’utiliser pour le mege.

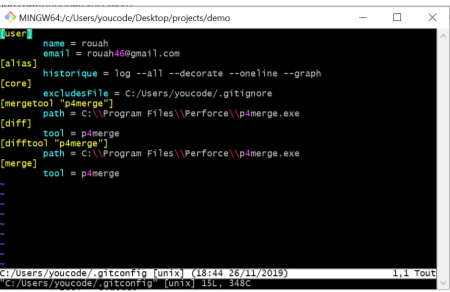
**7 Step : merge tools**

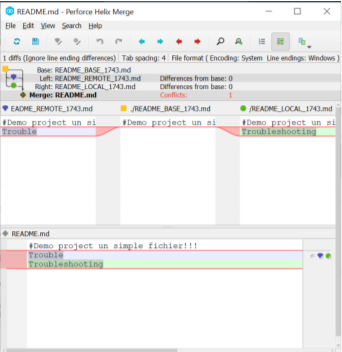
L’installation de p4merge sur notre PC

**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**

****

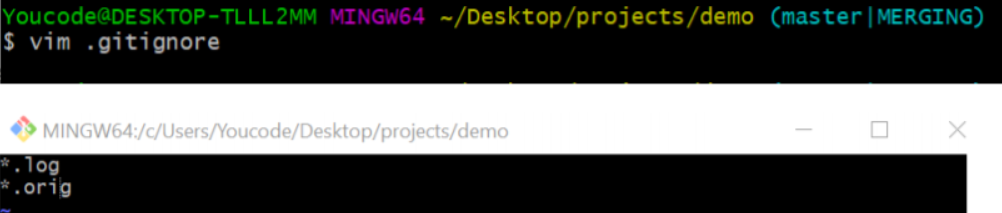
****

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.

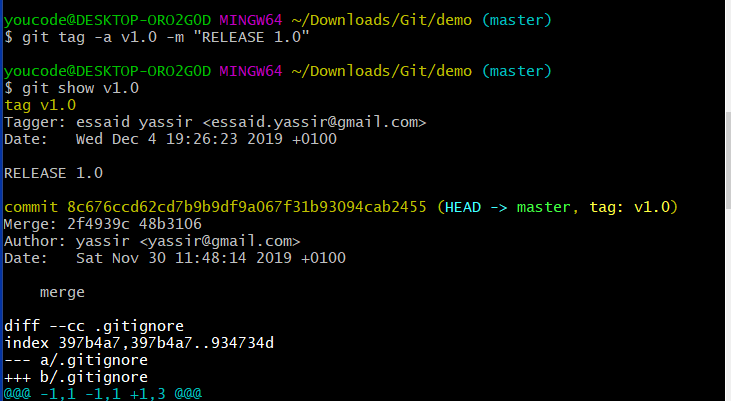
**8 Step : Challenge**

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



**Scenario 2**

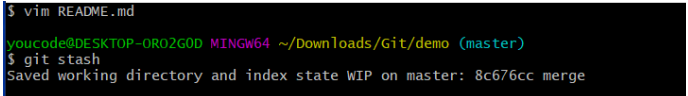
First Step: Tagging



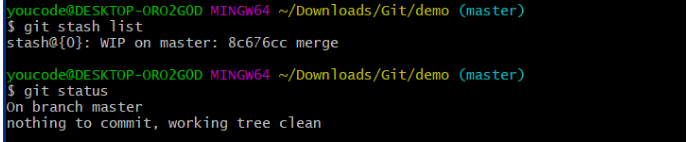
Tag vous permettent d’identifier des versions spécifiques de votre code. Vous pouvez considérer tag comme une branche qui ne change pas. Une fois créé, il perd la possibilité de changer l'historique des commits.

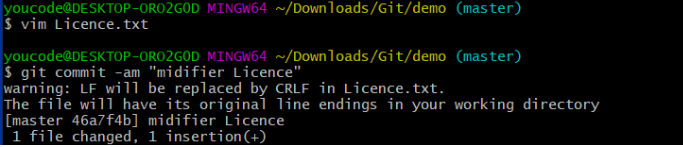
Second Step STASH

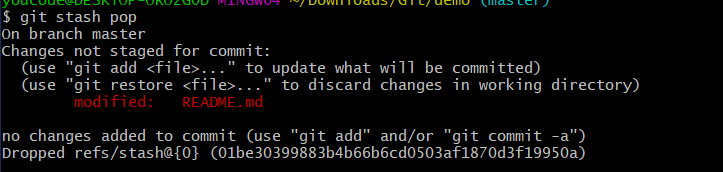
Dans c’est Etape nous avons Crées une modification et après Ça nous avons Tapez la commande git stash.



La commande git stash prend vos modifications non validées (staged and unstaged), les enregistre pour une utilisation ultérieure, puis les revenir de votre copie de travail.





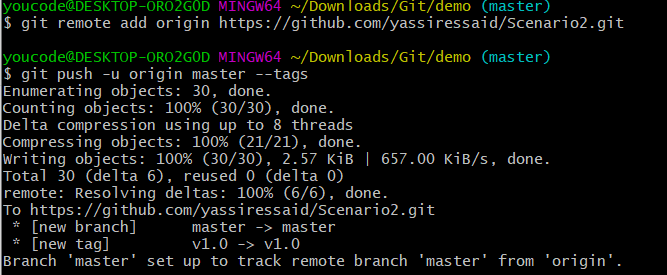


La commande git stash pop supprime les modifications de votre stash et les réapplique à votre copie de travail.

third step: Voyage sur Github, Local Repo to github Repo

Créer un repo GitHub public sans ajouter le fichier README.md

Créer et examiner le remote et pusher :



La commande git push –u origin master - -tags est utilisé pour charger le contenu du repo local vers un repo distant.

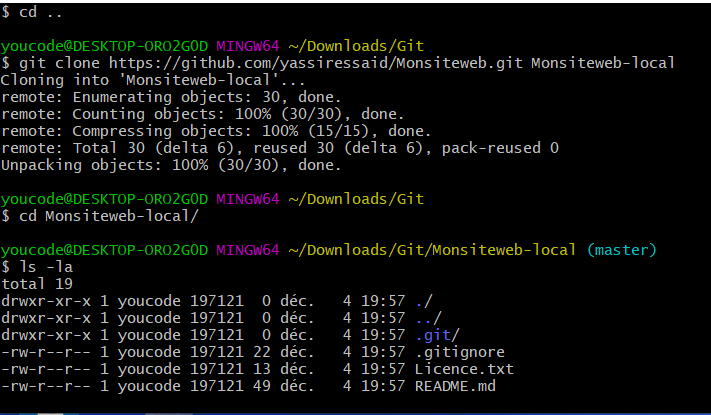
L’indication –u vous permet de tirer un git sans fournir plus d'arguments. Par exemple, une fois que vous avez créé un git push –u origin master, vous pourrez ensuite appeler git pull et il comprendra que vous vouliez bien dire git pull origin master.

L'indicateur --tags envoie toutes vos tags locaux au repo distant

Fifth Step: Création d’une local copy

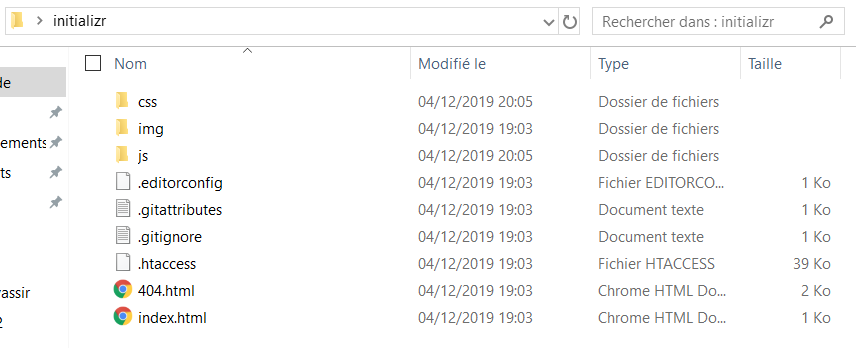
Sur GitHub créer un autre repo nommé (Monsiteweb)

Créer un clone GitHub et vérifier

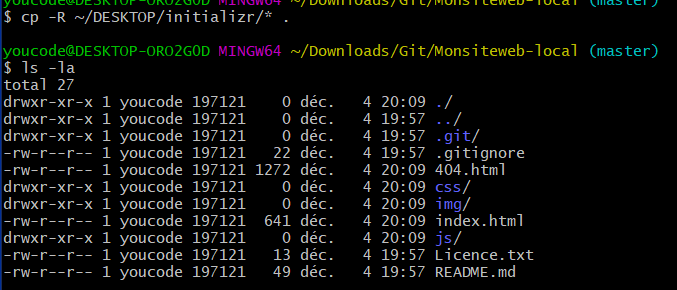


sixth step: Sending the website

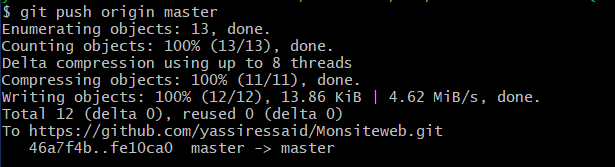
Après télécharger le site et l’analyse de l’arborescence



On copie le site dans repo local

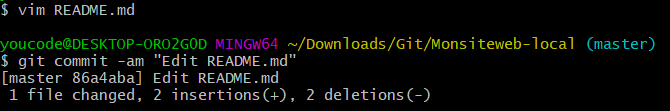


On fait le status, le staging, le commit et push à GitHub

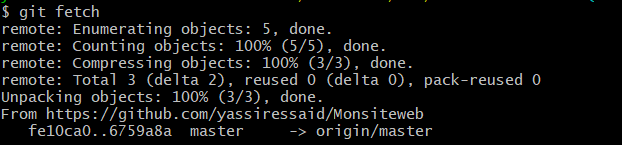


Seventh step: Fetch and pull

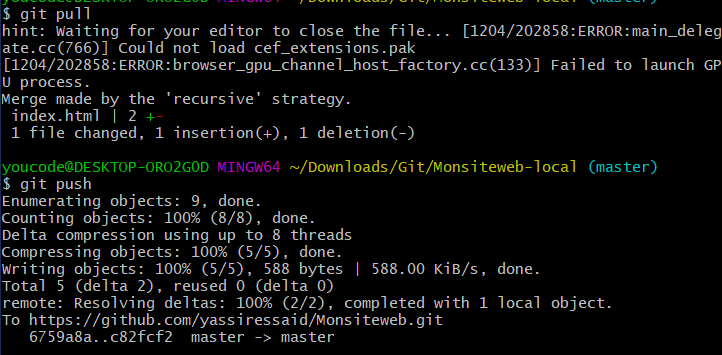
Sur GitHub on édite le fichier **Index.html** et fait le commit, et sur repo local on édite **README.md** et fait le commit



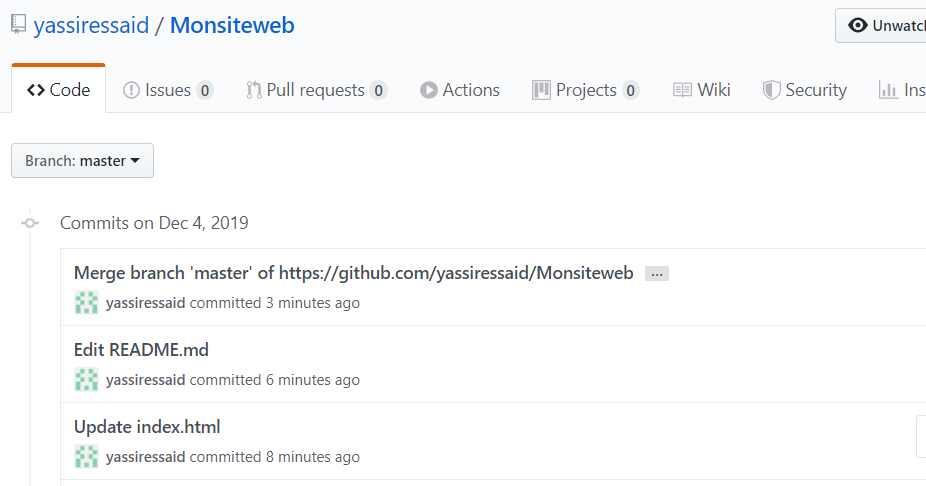
Exécuter git fetch



Git pull et push



Vérifier les commit sur GitHub



**Scenario 3**