Rapport de breif project

par

Hind Lhaimer

**SOMMAIRE**

Introduction

* Objectif
* Scénarion 1
* Scénarion 2
* Scénarion 3

– Introduction

Dans le cadre de la validation des compétences de la période SAS. le brief projet est un moyen utile pour que nous validons les compétences dans notre niveau (N1, N2, N3)

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

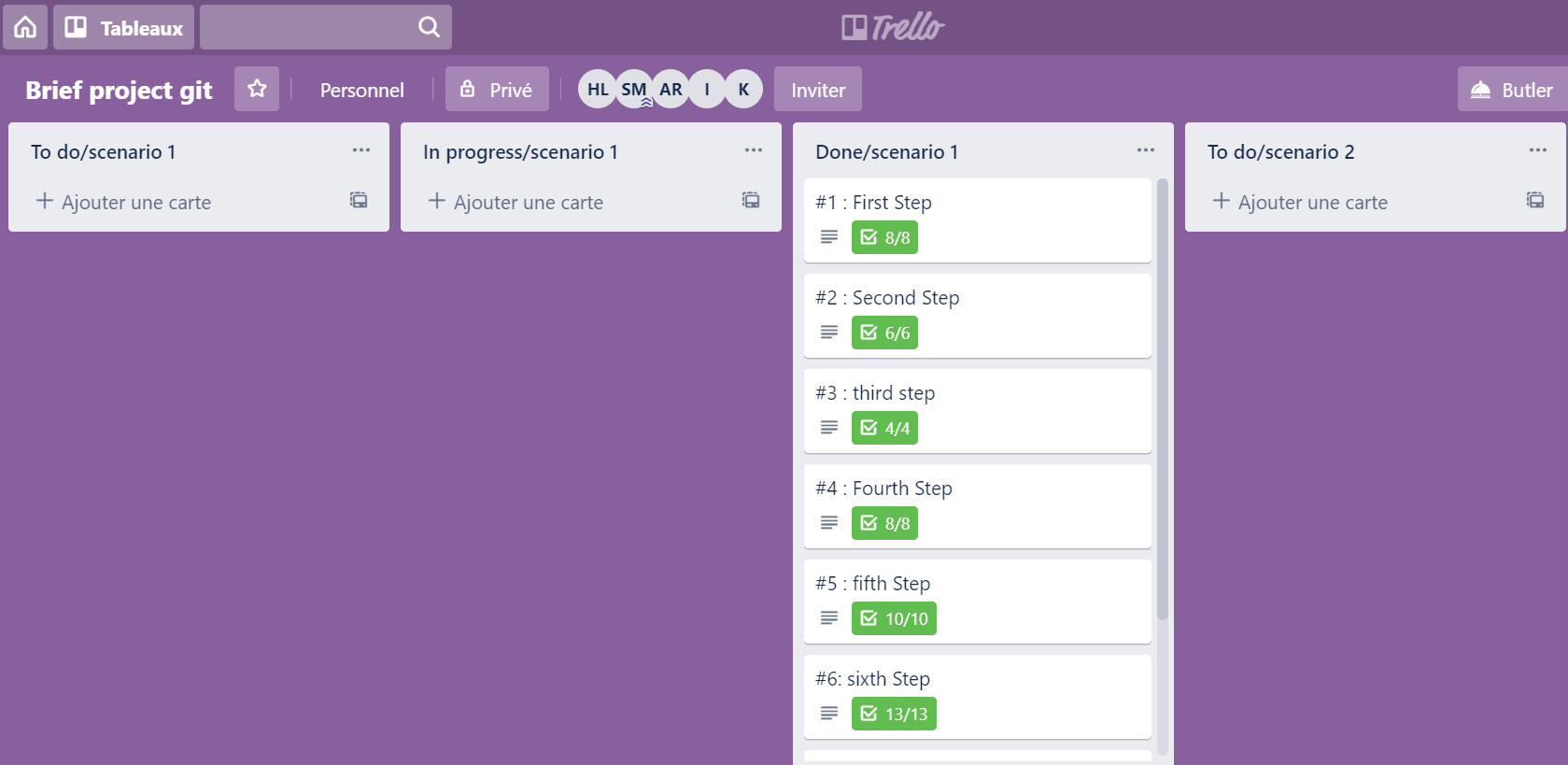
Pour réaliser ce projet nous avons utilisé la méthode Scrum et Trello.

La méthde Scrum :

**Scrum est** un schéma d'organisation de développement de produits complexes. Il est défini par ses créateurs comme un « cadre de travail holistique itératif qui se concentre sur les buts communs en livrant de manière productive et créative des produits de la plus grande valeur possible.

**Trello :**

Est un outil de gestion de projet en ligne, lancé en Septembre 2011 et inspiré par la méthode [Kanban](https://fr.wikipedia.org/wiki/Kanban) de [Toyota](https://fr.wikipedia.org/wiki/Toyota). Il est basé sur une organisation des projets en planches listant des cartes, chacune représentant des tâches. Les cartes sont assignables à des utilisateurs et sont mobiles d'une planche à l'autre, traduisant leur avancement.



**Scénario 1**

Il contient 8 étapes

La première chose qui nous avons vu dans ce scénario c’est la définition du Git et le fonctionnement du fichier caché. Git :

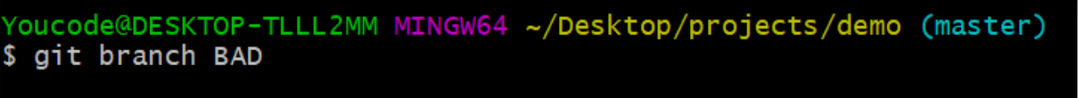
Git est un outil de versioning qui permet de tracer l’évolution de votre projet et d’y apporter des modifications sereinement.

Le git repository (ou history tree) est le fichier. Git habituellement caché, il contient l’historique des commits du projet. Il garde en mémoire tous les changements qui

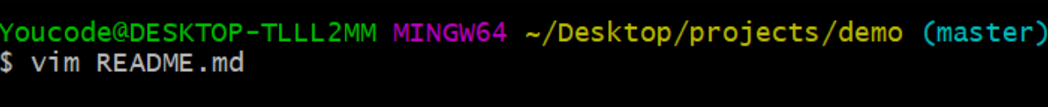
Ont été comités. C'est ce qui est cloné quand on clone la version d'un "serveur".

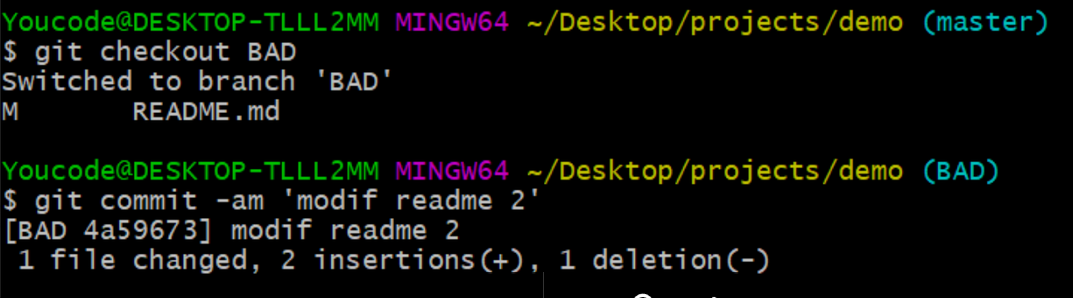
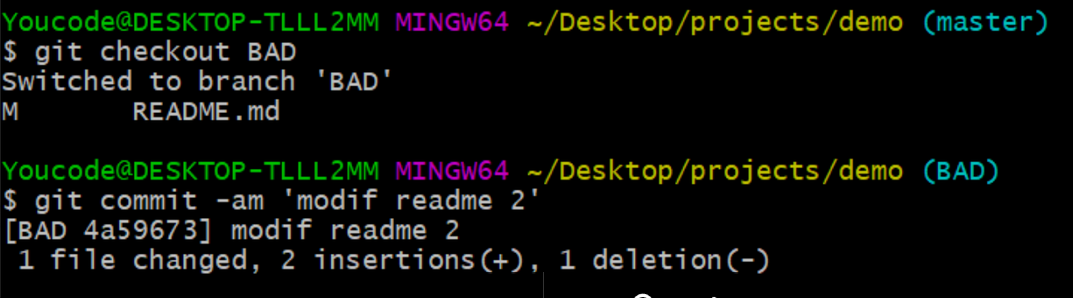
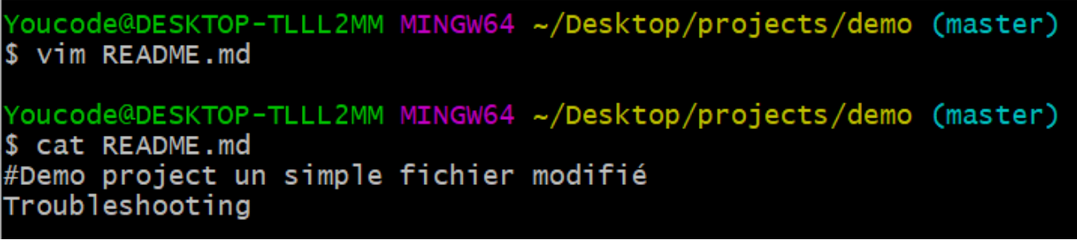
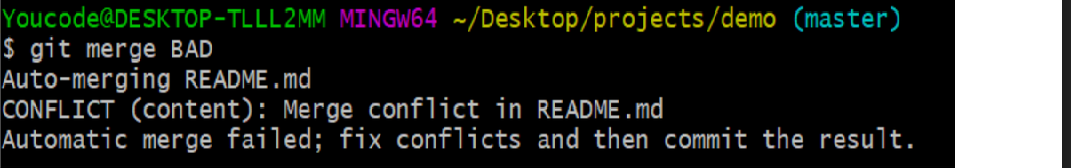
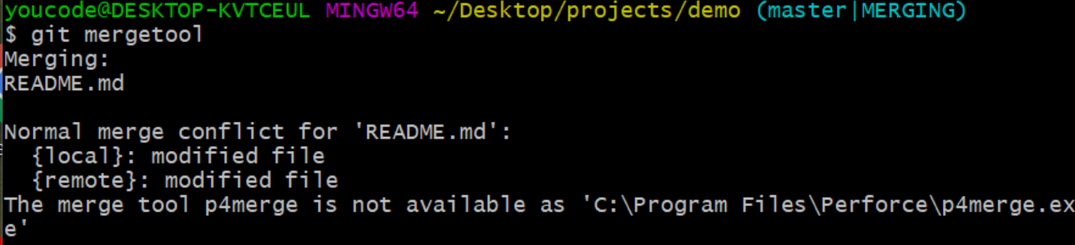
Puis nous avons fait un dossier qui s’appelle Projects et un autre dossier ( demo) .. .

La seul conflit qui nous avons rencontré c’est le conflit de merge de branche BAD car nous avons fait deux modifications le fichier README.md la première dans la branche master et la deuxième dans la branche BAD.

1. Créer une branche avec le nom ‘BAD’ :

2) Modifier le fichier README.md et ajouter la ligne ‘Trouble’:



1. Faire le stagging et le commit en une seule ligne:
2. Switcher faire la branche principale ‘master’
3. Switcher faire la branche principale ‘master’
4. Faire le merge de la branche ‘BAD’
5.  la commande « git mergetool » et conclusion:

**Scénario 2**

Il contient 7étapes

Dans ce scénario nous avons vu le Tag.

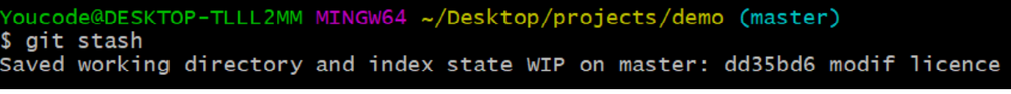
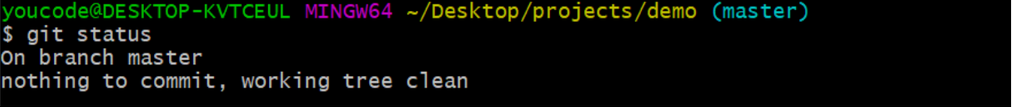
Les tags sont un aspect simple de Git, ils permettent d’identifier des versions spécifiques du code. Ils peuvent être considérés comme une étiquette, comme une branche qui ne change pas. Une fois créé, il perd la possibilité de changer

L’historique des commits.

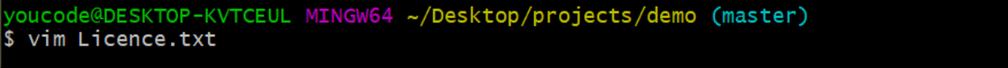
Et quelques nouveaux commande comme :

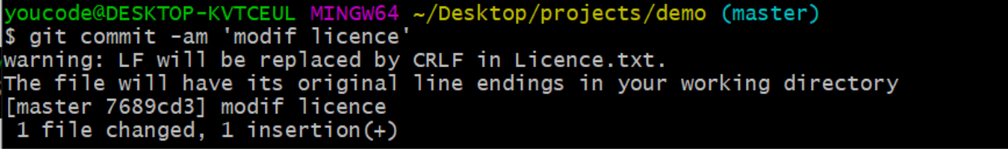
1. Tapez la commande git Stash ?
2. Expliquer le fonctionnement de la commande git Stash

Git Stash permet d’ouvrir une nouvelle fenêtre de travail sans inclure les actions précédentes effectués avant. Dans cette nouvelle fenêtre on peut réaliser des modifications sans se soucier qu’ils n’affectent notre travail principal car une fois lestash fermé aucune trace de ces modifications ne reste. En gros on peut décrire un stash comme une feuille de brouillon.

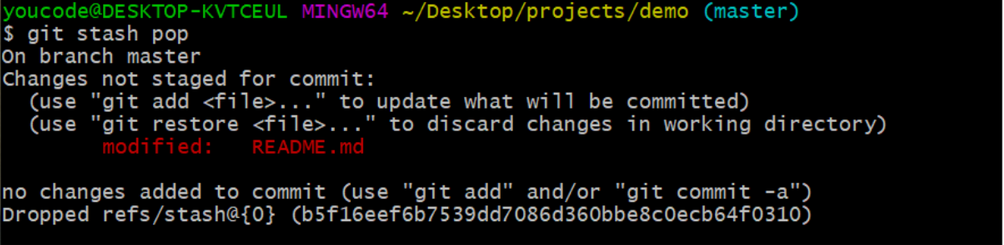
1. Taper la commande git stash list
2. Exécuter la commande git status

On observe que l’utilisation de git stash c’est comme ouvrir une nouvelle page vide où on n’observe rien desmodifications précédentes. Même si on a modifié le fichier README avant sans commiter, il ne le montre pas et nous dit qu’il y a rien à commiter.

1. Modifier le fichier « Licence.txt » et ajouter la ligne « APACHE 2.0 »

 6)Faire le staging et le commit en une seule ligne :

7) Exécuter la commande « git stash pop »



1. Conclusion

On constate que la dernière modification effectuée sur le fichier Licence.txt n’a pas été prise en compte comme si ça n'a jamais été fait et on ne voit que le document modifié avant l’ouverture du Stash, et reste en attente d'être commit.

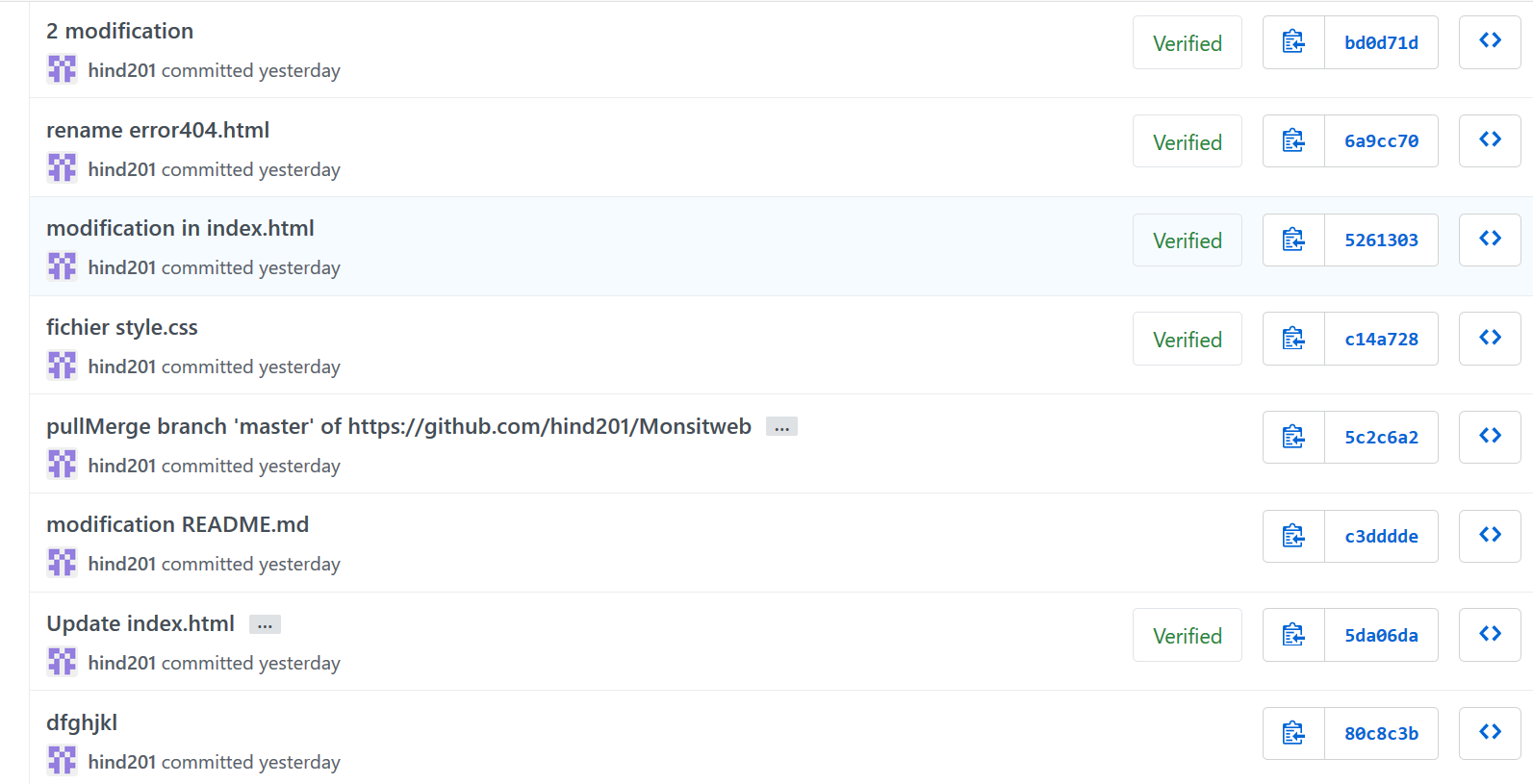
**Scénario 3**

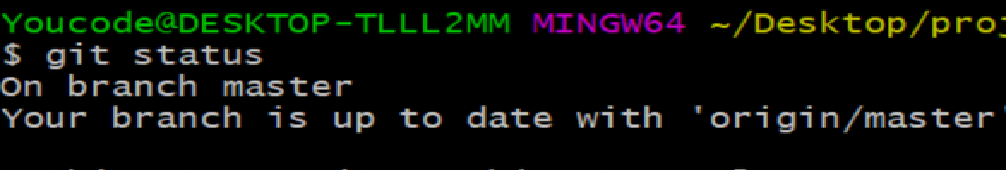
Il y a 8 étapes

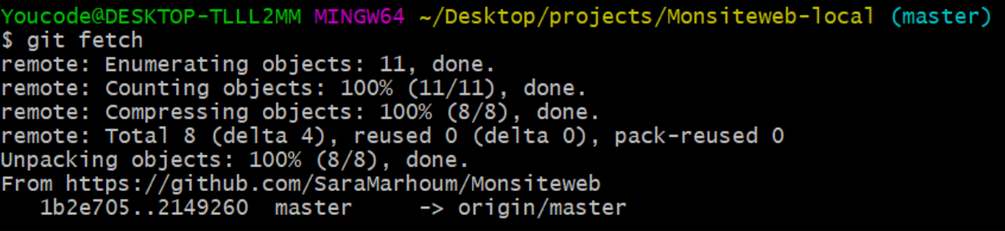
Parmi les choses qui nous avons fait la différence entre git fetch et git pull.

La commande git fetch va récupérer toutes les données des commits effectués sur la branche courante qui n'existent pas encore dans votre version en local. Ces données seront stockées dans le répertoire de travail local mais ne seront pas fusionnées avec votre branche locale.

La commande git pull est en fait la commande qui regroupe les commandes git fetch suivie de git merge. Cette commande télécharge les données des commits qui n'ont pas encore été récupérées dans votre branche locale puis fusionne ensuite ces données.

1. Sur gitHub vérifier la liste des commits .
2. Sur GITBASH, vérifier avec « git status »



1. Visualiser et télécharger les fichiers distants sur GITBASH
2. Maintenant faire le pull et fusionner les changements distants avec le repo local

