

**RAPPORT DE SCENARIO #2**

**Réalisée par : Nouhaila iounousse, MASSIF Abdessamed, Nada ElFadili, Hassan Hmiyed, Zakia Antary et abdelhaq elfakir**

**Encadré par : Mr. Youssef chouni**

**Année universitaire : 2019/2020**

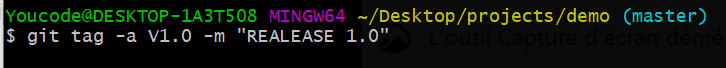
**Scénario 2**

#1 : First Step : Tagging

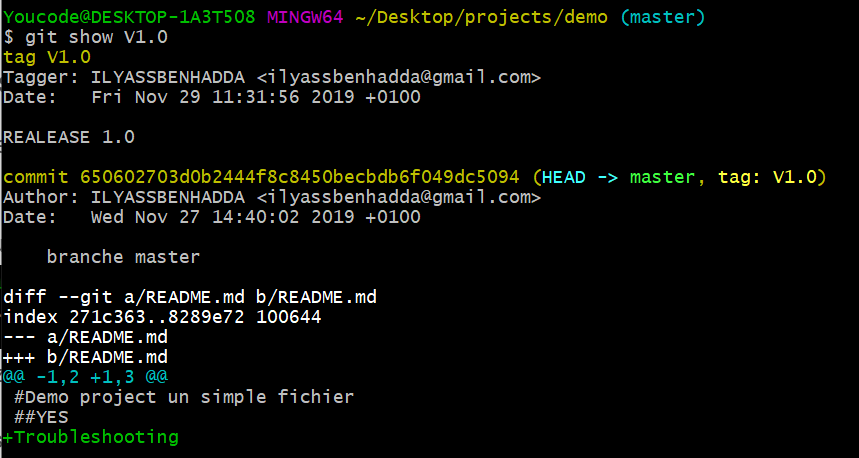
⇨ Déplacez-vous sur la branche Principale

**OK**

⇨ Créez un TAG avec un nom V1.0 et un commentaire ‘ REALEASE 1.0 ‘



⇨ Affichez les informations sur le TAG



⇨ D’après vous un **TAG** il sert à faire quoi au juste ?

**Git tag** sert à marquer des points spécifiques de l'historique d'un référentiel comme étant importants. Généralement, les utilisateurs utilisent cette fonctionnalité pour marquer les points de publication.

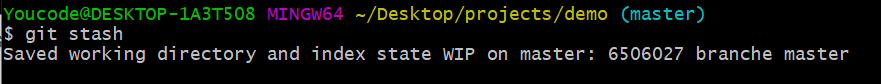
#2 : Second Step : Stashing and Saving work in Progress

⇨ Modifiez le fichier README.md , ajoutez une ligne

**OK**

⇨ A vrai dire , l’équipe a décidé que cette tâche sera en standby et que les modifications progressent avec le timeline du projet

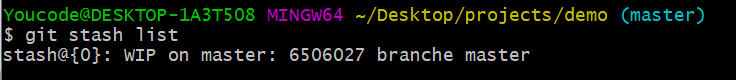
1. Tapez la commande git Stash ?



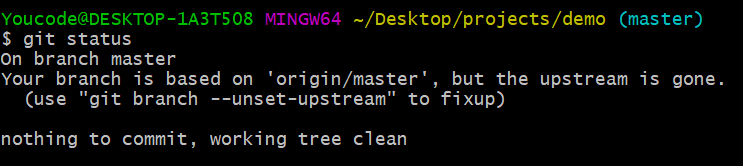
2. Expliquez le fonctionnement de la commande git Stash

Utilisez cette option git stash lorsque vous souhaitez enregistrer l'état actuel du répertoire de travail et de l'index, mais que vous souhaitez revenir à un répertoire de travail vide

3. Tapez la commande git stash list



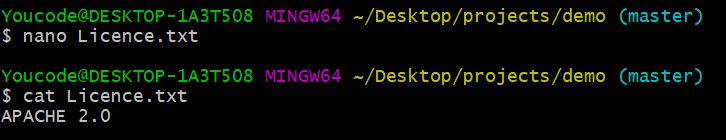
4. Exécutez la commande git status ?



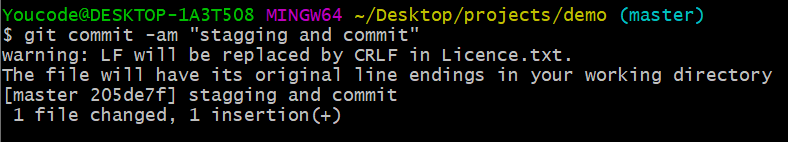
5. Qu’est-ce que vous constatez ?

Dans le branche master aucun commit été enregistré .

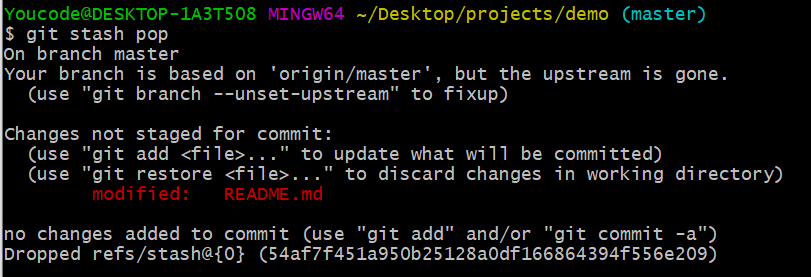
⇨ Modifiez le fichier « Licence.txt », ajoutez la ligne « APACHE 2.0 »



⇨ Faites stagging et le commit en une seule ligne



⇨ Exécutez la commande « git stash pop » ?



⇨ Qu’est-ce que vous constatez ?

Les données de stash seront supprimées de la pile de la liste de stash.git stash pop prend un changement caché, le supprime de la «pile cachée» et l'applique à votre arbre de travail actuel.

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

⇨ Pour cette phase vous devez avoir un compte GITHUB

ok

⇨ Créez un repo github public sans ajouter le fichier README.md

ok

⇨ Créez le remote en https

https://github.com/abdessamed11/scenario2.git

⇨ Examinez le remote

**Youcode@DESKTOP-KRPHUO1 MINGW64 ~/Desktop/projects (master)**

**$ git remote add origin https://github.com/abdessamed11/scenario2.git**

⇨ Pushez le tous à travers la commande : git push –u origin master - -tags

**Youcode@DESKTOP-KRPHUO1 MINGW64 ~/Desktop/projects (master)**

**$ git push -u origin master --tags**

✔ Expliquez la commande ?

**GIT PUSH Utilisation de la ligne de commande pour PUSH to GitHub**

**$ git push -u origin master --tags cette commande Met à jour les références distantes à l'aide de références locales en transfert même les tags créés.**

✔ Expliquez les options –u et - -tags ?

git push -u pour pusher le dossier modifier sur le dossier distant et fait la mise à jour

--tags pour pusher le dossier local avec les tags deja creer.

⇨ Sur Github Vérifiez la liste des commits , les branches , les releases et les tags



#4 : Fourth Step : Mini challenge ( optionnel)

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.

▪ HINTS :

🡺 Créez un fichier sous git nommé **.SSH**

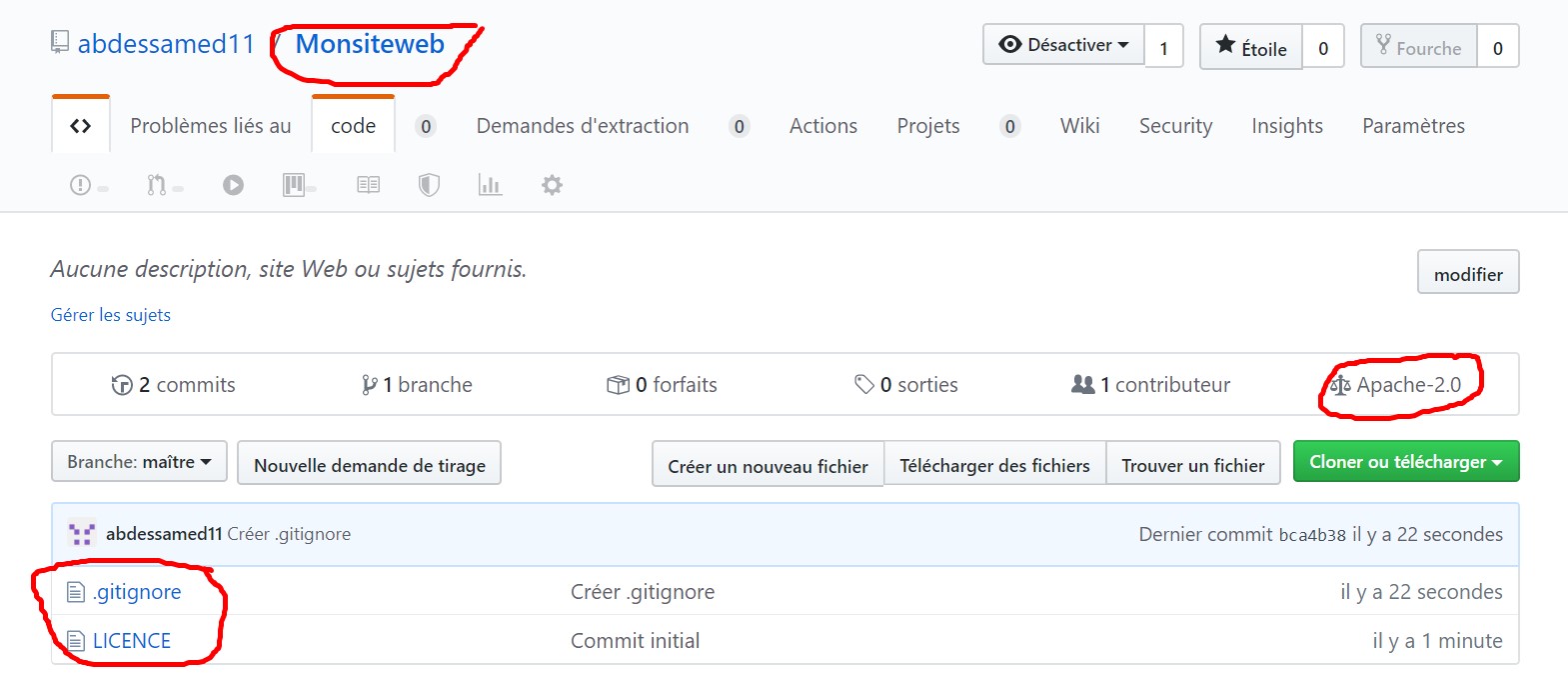
⇨ Déplacez-vous dans le fichier .SSH

⇨ Maintenant à vos mains : Créez une authentification sécurisé SSH entre votre repo local et votre repo distant (aide : Ssh –keygen-t rsa –C ' email’).

#5: Fifth Step: Création d’une local copy :

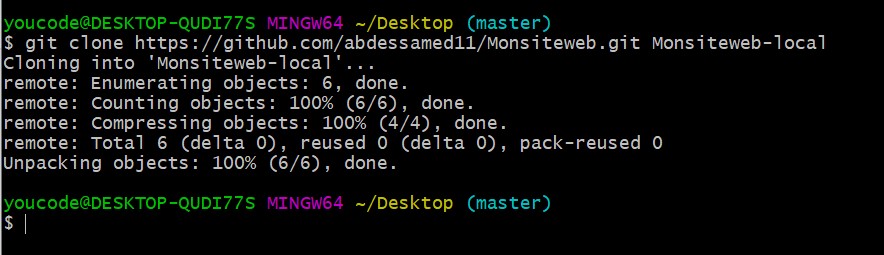
⇨ Sur github créez un autre repo nommé ( Monsiteweb)

⇨ Ajoutez à l’arboresence toujours sur github le fichier **.gitignore** et un fichier **licence.txt ‘APACHE 2.0 ‘**



⇨ Déplacez-vous dans le répertoire projets sous GIT

⇨ Créez un clone github vers le local sous le nom (Monsiteweb-local)



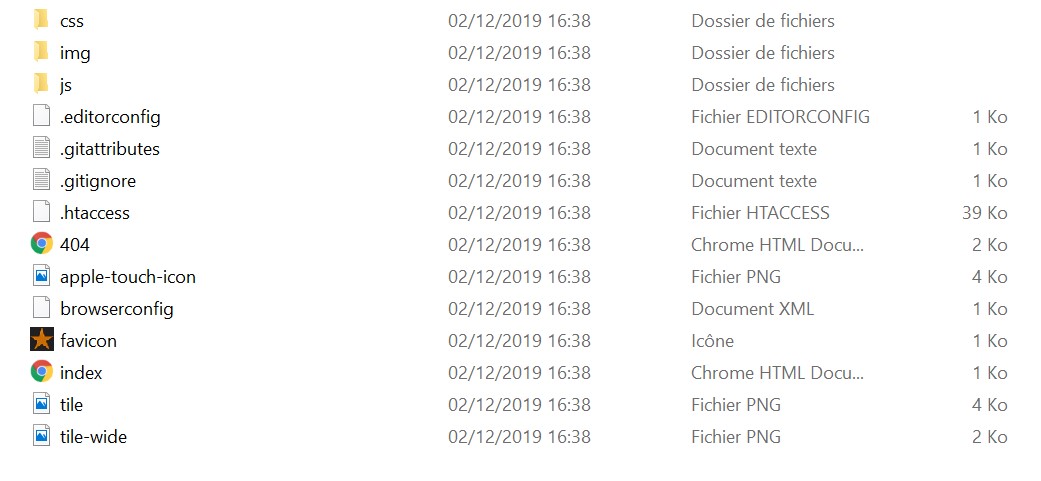
⇨ Vérifiez si le clone

#6: sixth step : Sending the website :

⇨ Télécharger le site web depuis le lien suivant :<http://www.initializr.com/>

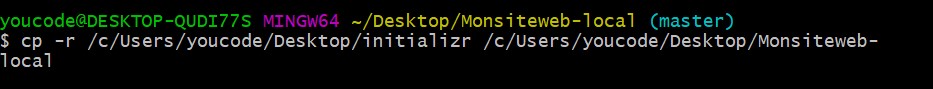
⇨ Sur le site telecharger un site bootstrap avec le fichier **.htaccess** et le fichier **404.html**

⇨ Analysez l’arboresence.

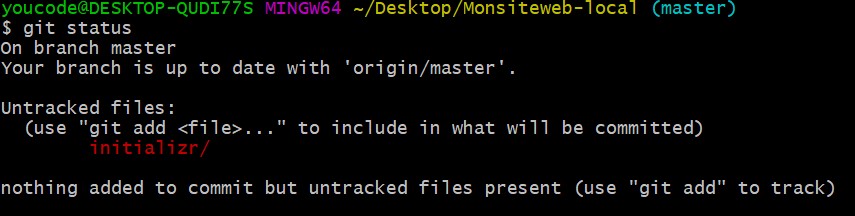
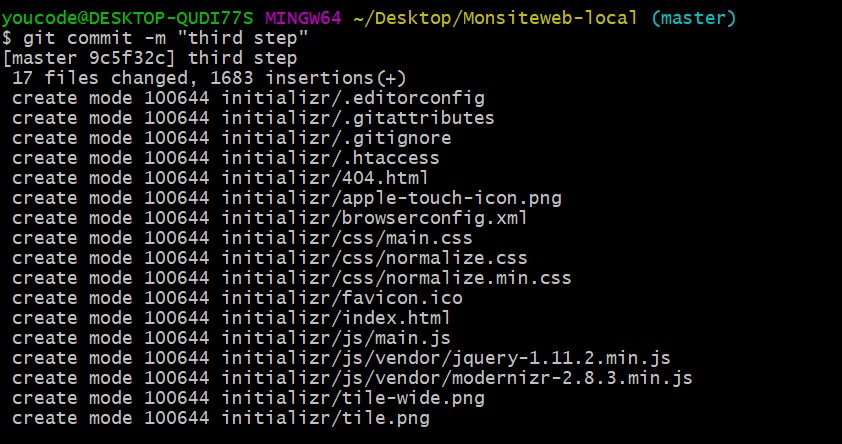


⇨ Expliquez ?

⇨ Copiez le site télécharger dans votre repo local à travers une seule commande :



⇨ Git status



⇨ Faites le stagging et le commit en une seule ligne

⇨ Faite le push à github



⇨ Vérifiez l’existence du site local sur votre repo github

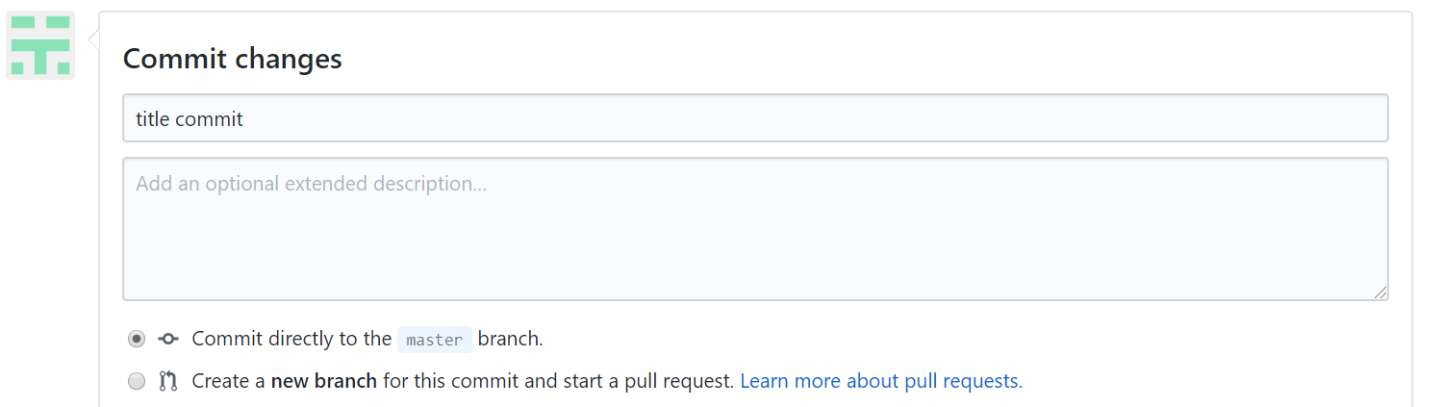
Ok

#7: seventh step : Fetch and pull :

⇨ Sur github éditez le fichier **Index.html**, sur la balise <title> </title> ajoutez le titre , mon premier site web .

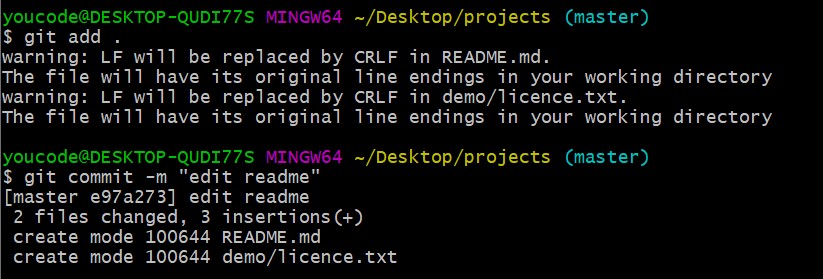


⇨ Faites le commit sur github



⇨ Sur git et sur le repo local , Editez le fichier README.md

⇨ Faites le stagging et le commit en une seule ligne

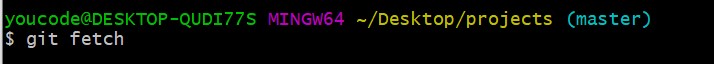


⇨ Qu’est-ce que vous constatez ?

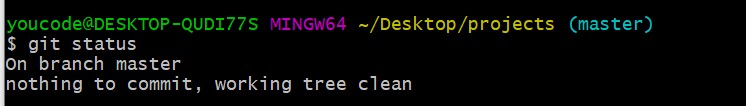
La commit est bien fait

● Trouble shooting :

⇨ Executez la commande « git fetch »



⇨ Git status



⇨ Expliquez ?

⇨ Git pull

**$ git pull origin master**

**From https://github.com/khaoulabendaoud/Monsiteweb**

**\* branch master -> FETCH\_HEAD**

**Merge made by the 'recursive' strategy.**

**initializr/index.html | 2 +-**

**1 file changed, 1 insertion(+), 1 deletion(-)**

⇨ Git push

**$ git push origin master**

**Enumerating objects: 9, done.**

**Counting objects: 100% (8/8), done.**

**Delta compression using up to 8 threads**

**Compressing objects: 100% (4/4), done.**

**Writing objects: 100% (5/5), 541 bytes | 541.00 KiB/s, done.**

**Total 5 (delta 2), reused 0 (delta 0)**

**remote: Resolving deltas: 100% (2/2), completed with 1 local object.**

**To https://github.com/khaoulabendaoud/Monsiteweb.git**

**8a526e4..c3fab08 master -> master**

⇨ Verifiez les commit sur github

