Support d'apprentissage | Utiliser GIT

1] Configurations initiales

Pré-requis : Git installé sur votre système.

- ***Configurer l'environnement de Git en mode console***

A ce stade, la configuration de Git se fait de la même façon quelque soit le système (Windows avec **Git Bash** et Linux avec une **console dans le répertoire de l'utilisateur**).

T1.1 | La première chose à faire est d'entrer votre identité (le « `username` » de GitLab) et votre mail, avec éventuellement la coloration syntaxique :

```
$ git config --global user.name "mettre ici le username"
$ git config --global user.email "mettre ici l'adresse email du compte"
$ git config --global color.ui auto
```

Votre configuration est alors enregistrée dans le fichier caché `gitconfig` :

```
$ cat .gitconfig
```

Ce qui donne en retour, toujours en étant dans votre répertoire utilisateur :

```
[user]
        name = votre_username
        email = votre_adresse_email
[color]
        ui = auto
```

Par défaut (Linux), Git utilise l'éditeur de texte **nano** ou se replie sur l'éditeur que vous avez configuré pour la création et l'édition des messages de validation et d'étiquetage. Pour modifier ce programme par défaut pour un autre, vous pouvez utiliser la commande :

```
$ git config --global core.editor vim
```

Si vous devez indiquer un **proxy** :

```
$ git config --global http.proxy http://url:port
```

Vous pourrez obtenir de l'aide sur la configuration de `git` avec la commande :

```
$ git config --help
```

Vous pourrez obtenir de l'aide sur l'utilisation de `git` avec la commande :

```
$ git help nom_de_la_commande
```

Note : *la complétion des commandes fonctionne avec Git.*

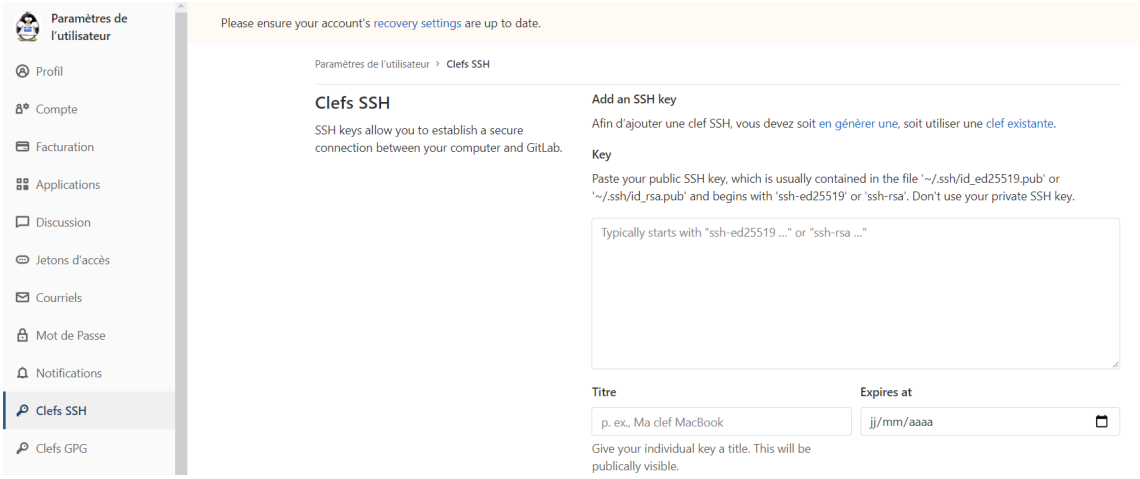
- ***Créer une clé SSH pour pouvoir effectuer des transferts***

L'échange de données cryptées entre votre ordinateur et GitLab nécessite la création d'une clé SSH.

T1.2 | Connectez-vous à votre compte sur GitLab et dans les paramètres de celui-ci, choisissez le menu Clés SSH (à gauche).

T1.3 | Générez une clé et ajoutez-la à votre système et la procédure décrite ici :

https://gitlab.com/help/ssh/README#generating-a-new-ssh-key-pair

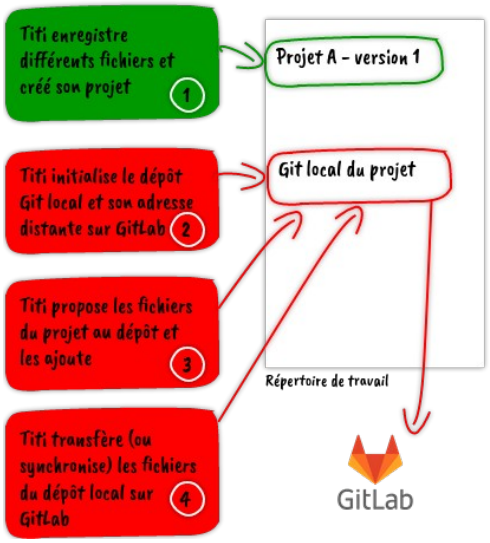


2] Scénario n°1 : un projet personnel privé

Détails : vous désirez mettre en place puis gérer un projet personnel de développement Web afin de disposer d'une sauvegarde et disponible par Internet ; votre **nom de code** : **Titi**.

Titi → T2.1 | Trouvez un partenaire pour ce TP ; il aura comme **nom de code** : **Grosminet**.

- ***Premières opérations***



Titi → T2.2 | Connectez-vous sur GitLab et créez un nouveau projet nommé `ProjectWeb` avec **Privé** comme niveau de visibilité et **sans cocher la case du README.**

Nous nous situons dans le cas d'un répertoire local déjà existant (cas le plus courant) qui se nomme par commodité… `ProjectWeb` qui se situera dans votre répertoire personnel.

Titi → T2.3 | Créez le répertoire `ProjectWeb` dans votre répertoire personnel, et créez le fichier `README.md` contenant la phrase « README du dossier ProjectWeb. ».

Ce fichier est au format **Markdown** (langage léger de balisage) et vous pouvez utiliser le logiciel **Typora** (https://typora.io/) pour le créer (conseillé).

Titi → T2.4 | Ouvrez une console ligne de commande et tapez les instructions ci-dessous :

→ Déplacez-vous dans le répertoire du projet :

```
$ cd ProjectWeb
```

→ Procédez aux initialisations :

```
$\projectweb\ git init
$\projectweb\ git remote add origin git@gitlab.com:votre_username/projectweb.git
```

→ Mettez à jour le dépôt local en proposant les changements :

```
$\projectweb\ git add .
```

→ Validez les changements :

```
$\projectweb\ git commit -m "Création du README"
```

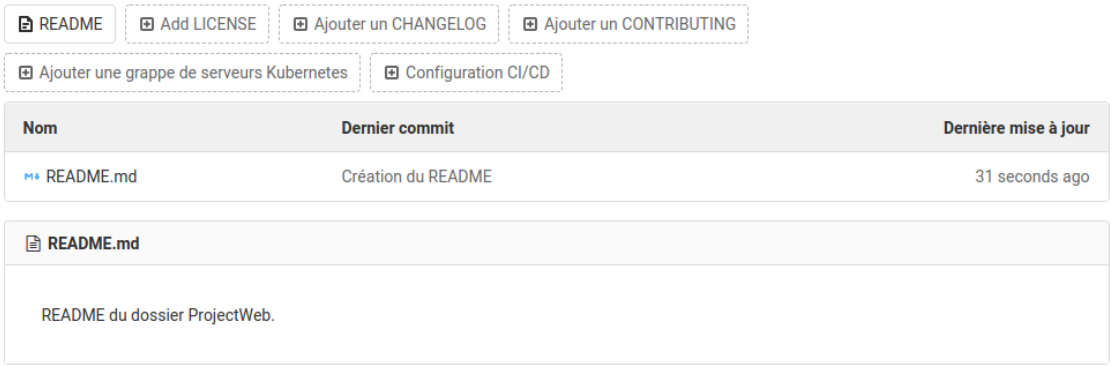
→ Envoyez les changements :

```
$\projectweb\ git push origin master
```

`Master` signifie que les changements auront lieu sur la branche principale du projet.

Note : *la dernière étape, c'est-à-dire le transfert vers GitLab, ne peut s'effectuer que si vous avez correctement paramétré la clef SSH.*

En revenant dans un navigateur sur votre projet, vous devriez voir votre premier `commit` :



- ***Ajout/modification avec le dépôt***

Hypothèse : Vous ajoutez des éléments dans votre code, **au moins une page HTML** `index.html` (contenu libre) et vous désirez mettre à jour le dépôt GitLab.

Titi → T2.5 | Ajoutez un fichier dans votre répertoire `ProjectWeb` et mettez à jour le dépôt local (proposition de changements) :

```
$\Documentation\ git add *
```

L'étoile représente toutes les modifications, on peut la remplacer par le nom d'un seul fichier par exemple.

Titi → T2.6 | Validez les changements :

```
$\projectweb\ git commit -m "Ajout de la page index.html"
```

Titi → T2.7 | Envoyez les changements :

```
$\projectweb\ git push origin master
```

Important :

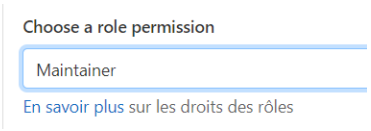
La référence au projet GitLab (`remote`) du projet local en version 1 est la même que la référence placée sur GitLab (même clé de hachage : **SHA=X**). La soumission (`push`) de **Titi** sera donc acceptée, le projet passant en version 2 (changement de HEAD), cette clé de hachage deviendra alors **SHA=Y**.

3] Scénario n°2 : un projet partagé

Détails : vous désirez partager votre projet de développement avec l'autre développeur de **nom de code** : **Grosminet**.

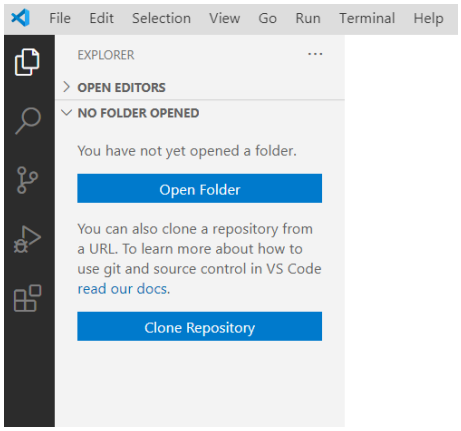
- ***Premières opérations***

Titi → T3.1 | Sur GitLab invitez, en tant que `maintainer` sur le projet (Menu `Membres`), le développeur **Grosminet** (rechercher par son adresse email).

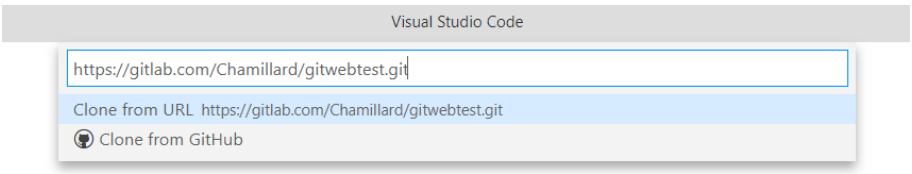


A ce stade, les deux développeurs doivent disposer, puisque nous sommes dans un projet de développement, de l'IDE **Visual Studio Code** (https://code.visualstudio.com) et avec l'**extension GitLab WorkFlow d'installée** (il vous sera demandé le **mot de passe** de votre compte GitLab, attention pas la clé SSH !).

Grosminet → T3.2 | Ouvrez Visual Studio Code et à partir du menu `Explorer`, choisissez le bouton `Clone Repository` :



Grosminet → T3.3 | Retournez sur GitLab et dans le projet en haut à droite choisit `clone`, puis `clone with HTTPS` pour copier l'**URL** du projet, choisit l'endroit du projet et intègre l'URL ; voici mon cas :

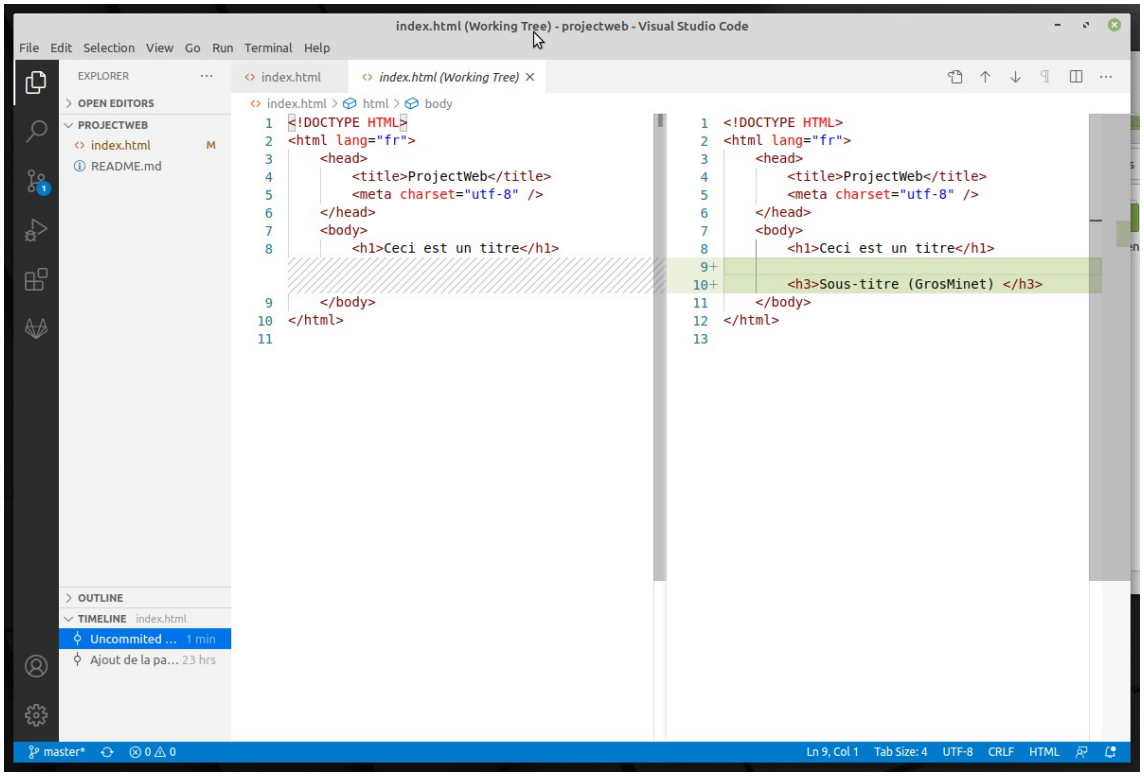


Au final, après avoir renseigné le nom du dossier local et les identifiants pour GitLab, l'importation s'effectue et la page HTML s'affiche.

- ***Modification sur le projet avec Visual Studio Code d'un développeur***

Grosminet va travailler maintenant en effectuant un ajout dans le code HTML de la page. La démarche est la même que précédemment. Il peut être intéressant d'utiliser quelques outils de Visual Studio Code même si préférence reste à la ligne de commande...

Grosminet → T3.4 | Ajoutez du code et cliquez sur `Uncommited changes` dans le l'onglet TIMELINE (bas de page d'Explorer) pour obtenir le visuel des deux arborescences de travail (Git local et répertoire de travail) :



A ce stade, vous pouvez soit utiliser les commandes de Visual Studio Code (que je vous laisse découvrir…) soit utiliser classiquement les lignes de commandes vues plus haut (méthode préférée).

Rien ne vous empêche par contre d'utiliser le **Terminal** de Visual Studio Code !

Grosminet → T3.5 | Effectuez les opérations nécessaires pour synchroniser le projet sur GitLab.

Important :

La référence au projet GitLab (`remote`) du projet local en version 2 (clone) est la même que la référence placée sur GitLab (même clé de hachage : **SHA=Y**). La soumission (`push`) de **Grosminet** sera donc acceptée, le projet passant en version 3 (changement de HEAD), cette clé de hachage deviendra alors **SHA=Z**.

- ***Modification sur un projet déjà modifié***

Détails : Grosminet a soumis son travail en premier. Du coup la référence distante du travail de Titi (**SHA=X**) ne correspond plus à celle actuelle sur GitLab (**SHA=Z** au lieu de **Y**). Conclusion : Titi ne peut donc pas faire de `push`.

Titi devra donc faire un `pull` au préalable, régler d'éventuels conflits, synchroniser son dépôt local et le distant. Cette dernière version devenant la référence pour Grosminet.

Important :

La commande `git fetch` télécharge le contenu distant mais ne change pas l'état du dépôt local. Il faudra effectuer un `git merge` ensuite. La commande `git pull` effectue les deux automatiquement mais de fait est un peu plus « dangereuse »...

Titi → T3.6 | Effectuez un `git fetch`, un `git status` et un `git pull` pour obtenir les modifications de Grosminet.

- ***Suppression d'un élément du dépôt***

Titi → T3.7 | Ajoutez au niveau local puis distant un autre fichier nommé `essai.txt`.

Hypothèse : Vous vous êtes trompé et vous désirez maintenant supprimer le fichier `essai.txt` avec synchronisation des dépôts.

Titi → T3.8 | Tapez les instructions ci-dessous :

```
$\projectweb\ git rm essai.txt
$\projectweb\ git status
$\projectweb\ git commit -m "Suppression de essai.txt"
$\projectweb\ git push origin master
$\projectweb\ git status
```

Remarques :

1. La commande `git rm` réunit à la fois la commande `rm` du système pour le répertoire local et pour le dépôt local Git
2. La commande `git status` permet de voir si tout c'est bien passé, n'hésitez pas à la lancer !
3. La dernière commande `git status` doit vous indiquer que la copie de travail est propre

En résumé :