|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description : defi_rvb_72dpi | Mode opératoire de traitement | | | **Référence :**SI | |
| (selon nomenclature) | |
| Utiliser GitLab et Atlassian Source Tree | | | | | |
| **Nature et objectif du traitement :** | | l’intérêt et utilisation de GitLab et Atlassian Source Tree. | | | |
| **Domaine d’application :** | | Processus de déclôture des dossiers OPTIFORM, s’adressant à toutes personnes. | | | |
| **Document(s) produit(s) :** | | Utiliser GitLab et Atlassian Source Tree | | | |
| **Diffusion du présent document** | | Publique | Restreinte | | Non diffusable |

**Suivi des versions :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versions** | **Date** | **Objet de la modification** | **Rédacteur(s) / Valideur(s)** |
| 0.1 | 24/06/15 | Création | GW Flamant |
|  |  |  |  |
|  |  |  |  |

Table des matières

[1 Qu’est-ce que GitLab et AtlassianSourceTree et pourquoi ? 3](#_Toc430880309)

[2 Installation et configuration d’Atlassian Sourcetree 4](#_Toc430880310)

[3 Se connecter au GitLab 6](#_Toc430880311)

[4 Présentation rapide de GitLab 6](#_Toc430880312)

[5 Cloner un projet 7](#_Toc430880313)

[6 COMMIT et PUSH vers le serveur distant 9](#_Toc430880314)

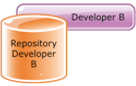
# Qu’est-ce que GitLab et AtlassianSourceTree et pourquoi ?

1. GitLab est un gestionnaire de version.

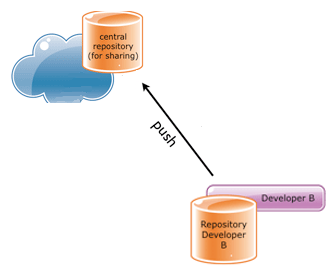
Un gestionnaire de version est un système qui enregistre l'évolution d'un fichier ou d'un ensemble de fichiers au cours du temps de manière à ce qu'on puisse rappeler une version antérieure d'un fichier à tout moment.

1. Pourquoi GitLab ?

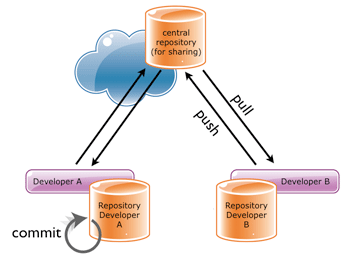
Si vous êtes un développeur par exemple, et que vous voulez conserver toutes les versions de votre code, un système de gestion de version (**VCS** en anglais pour Version Control System) est un outil qu'il est sage d'utiliser. Il vous permet de ramener un fichier à un état précédent, de ramener le projet complet à un état précédent, de visualiser les changements au cours du temps, de voir qui a modifié quelque chose qui pourrait causer un problème, qui a introduit un problème et quand, et plus encore. Utiliser un VCS signifie aussi généralement que si vous vous trompez ou que vous perdez des fichiers, vous pouvez facilement revenir à un état stable. De plus, vous obtenez tous ces avantages avec peu de travail additionnel. Voici comment ça fonctionne :  
  
Il existe un **dépôt de fichier central**, stocké dans un serveur. Ce dépôt sert de base, c’est la « référence » des sources :

Nous avons ensuite des contributeurs, par exemple « Developer B ». Lorsque vous arrivez sur un projet existant, vous devez cloner le projet sur votre poste, en « local ». Vous allez donc « copier » les sources sur votre ordinateur. Vous aurez donc votre propre répertoire.

Cela fait à présent deux répertoires, le **dépôt central** et le **dépôt local**. Quand vous travaillez, vous travaillez sur les sources qui sont en **local**. Après avoir mis à jour les sources locales, il est nécessaire de faire un « **Commit** ». Cela permet de créer une révision (version) en local. Il est ensuite nécessaire de mettre à jour le **dépôt central** en effectuant un « **Push** ». Nous allons « pousser » les sources vers le serveur.



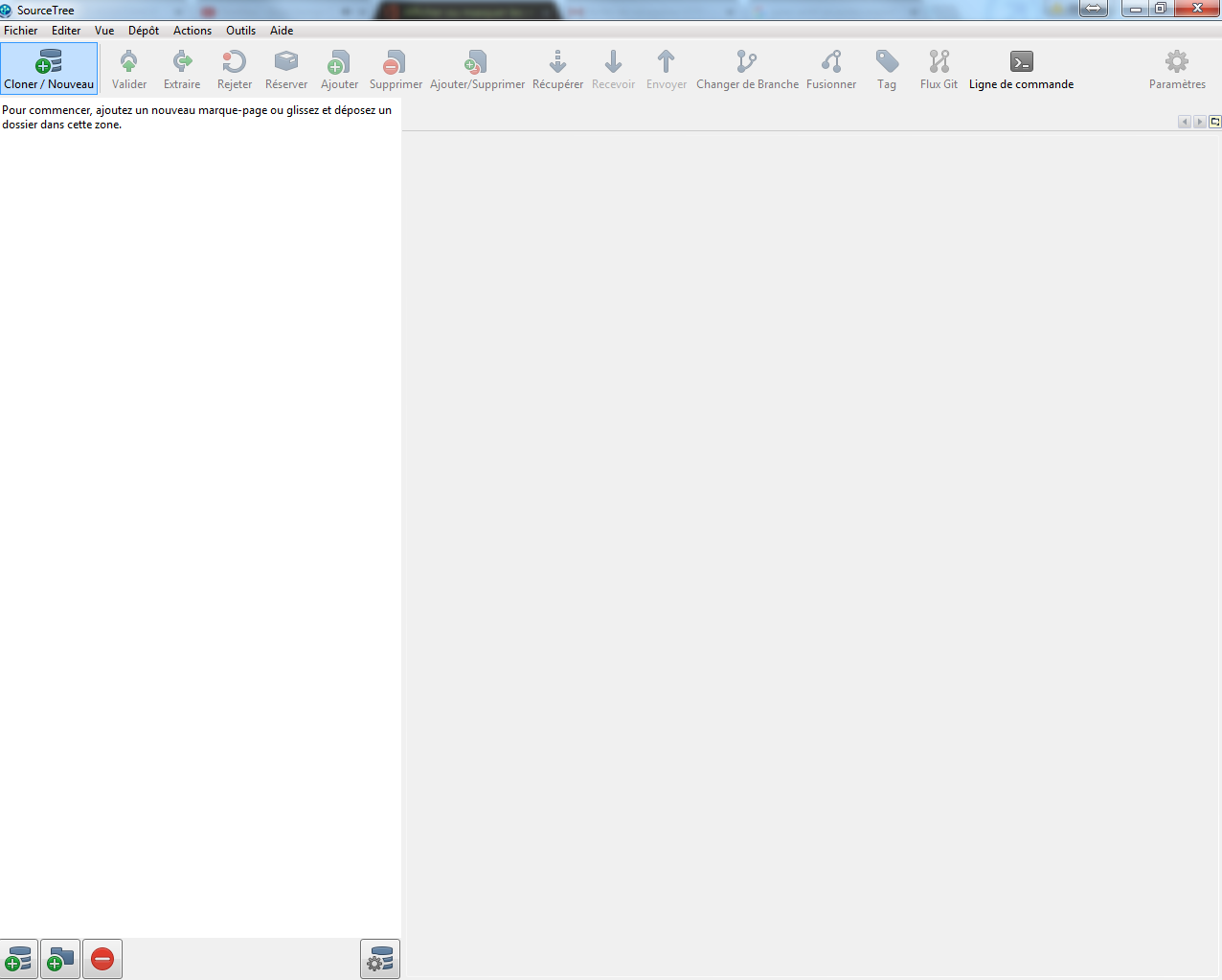
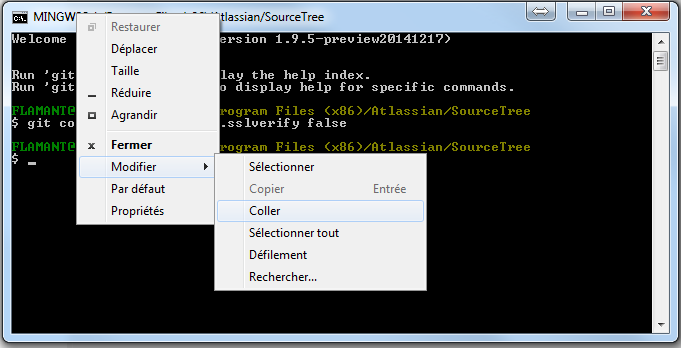
Un autre développeur nommé « Developer A » rejoint le projet. Le développeur fait également une copie du projet. Des commit, des push, des pull etc... Ce qui nous donne au final le schéma suivant :



1. Atlassian Source Tree

Atlassian SourceTree est simplement un utilitaire qui donne une interface utilisateur à GitLab. En effet, GitLab est un outil de VCS utilisable en ligne de commande. Grâce à Atlassian SourceTree, vous n’avez pas besoin d’apprendre par cœur toutes les commandes, une interface graphique simple est mise à votre disposition.

# Installation et configuration d’Atlassian Sourcetree

1. Lancer l’exécutable SourceTreeSetup qui se trouve dans [\\domdefi.fr\Partages\Public\DSPC\Outils\Sourcetree\](file:///\\domdefi.fr\Partages\Public\DSPC\Outils\Sourcetree\)
2. Cliquer sur **Suivant** 🡪 **Install** 🡪 **Finish**.
3. Ouvrir SourceTree. L’écran suivant s’affiche :  
     
    
4. Cliquer sur « **Ligne de commande** » en haut à gauche. Une fenêtre de commande s’ouvre.
5. Copier la ligne « ***git config --global http.sslverify false*** ».
6. Pour coller la ligne dans la commande, faire un clic droit sur la fenêtre puis suivre l’arborescence comme ci-dessous :  
     
   
7. Appuyer sur Entrer pour valider la ligne de commande.



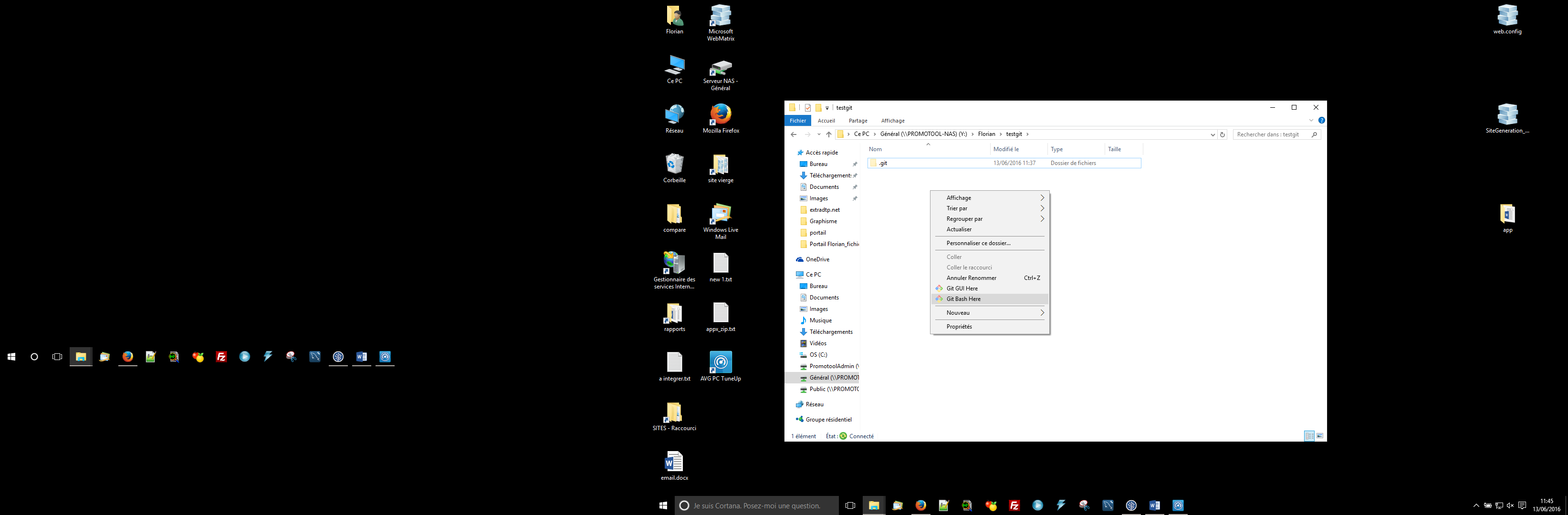
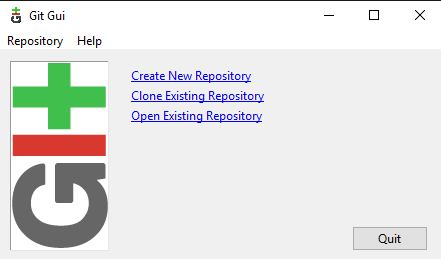
Installer et configurer Atlassian SourceTree

# Create / install a git repository manager on local server

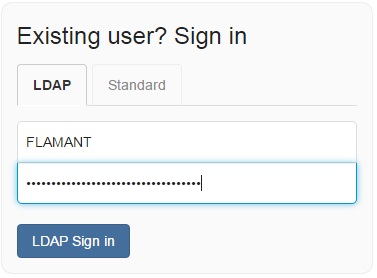
Aller sur le site :

<https://git-scm.com/book/it/v2/Git-on-the-Server-Summary>

Comment créer un repository/dépôts source :

1. créer un dossier qui servira à contenir le répo
2. Dans ce dossier, click droit et lancer un outil GIT (par exemple GUI GIT)
3. 
4.  clicker create new repository
5. Un dossier caché « .git » apparait (ce dossier à transformer un dossier simple en dossier repo git)

# Se connecter au GitLab

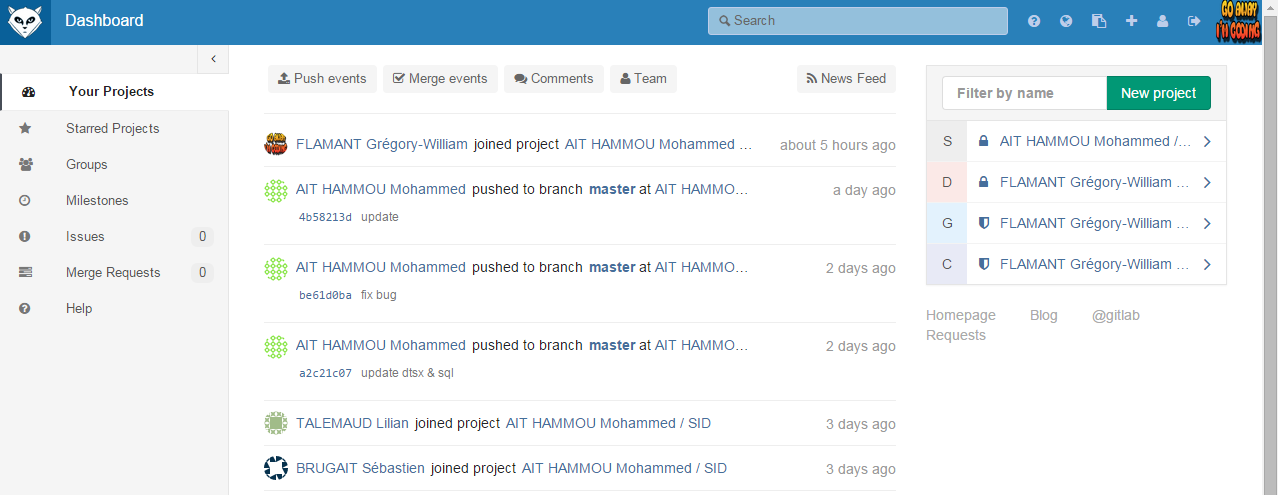
1. Se rendre à l’adresse «<https://gitlab.opcadefi.fr> ».
2. Dans la partie « Sign in », en haut à droite, renseigner ses identifiants Windows et cliquer sur « LDAP Sign In ».  
     
   



Se connecter à GitLab

# Présentation rapide de GitLab

Accueil



Gestion compte, logout, ...

Gestion projets

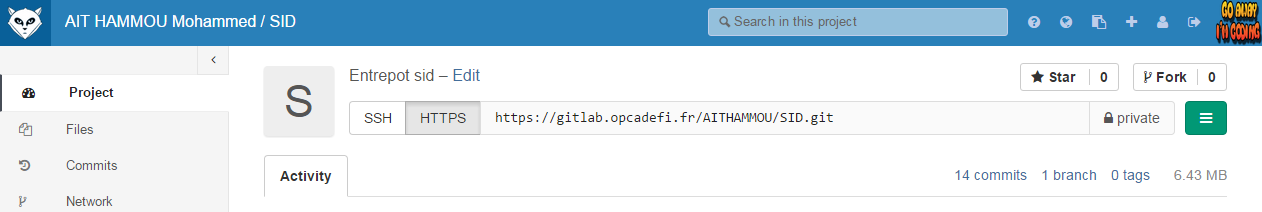
Fil d’actualité

# Cloner un projet

Cloner un projet signifie récupérer les sources du serveur vers votre ordinateur. Autrement dit, récupérer les sources distantes en local.   
  
*Nb : il est nécessaire de faire partie d’un projet avant de cloner un projet.*

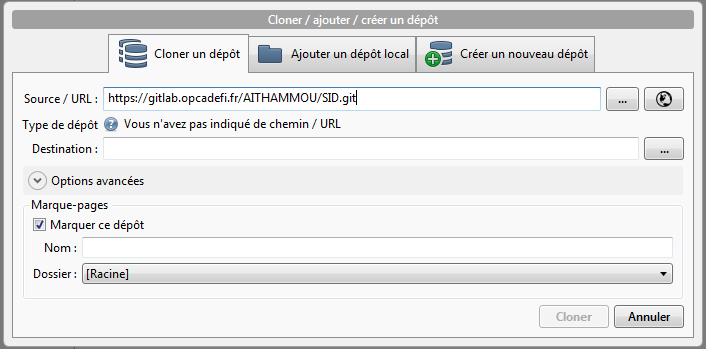
1. Dans l’accueil GitLab, dans la partie Gestion projets, cliquer sur le nom du projet.

2. Cliquer sur « **HTTPS**» puis copier le lien.

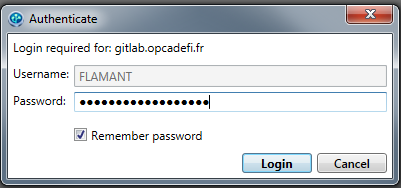


  
3. Dans SourceTree, cliquer sur « Cloner / Nouveau », en haut à gauche :

4. Dans la fenêtre qui apparaît, dans l’onglet « Cloner un dépôt », coller le lien dans « Source / URL » puis cliquer sur le champ « Destination ».



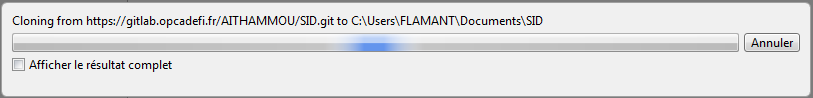
5. Attendre la vérification : 

6. Dans la fenêtre « Authenticate » qui apparaît, saisir les identifiants LDAP et cliquer sur « **Login** ».  
  
 

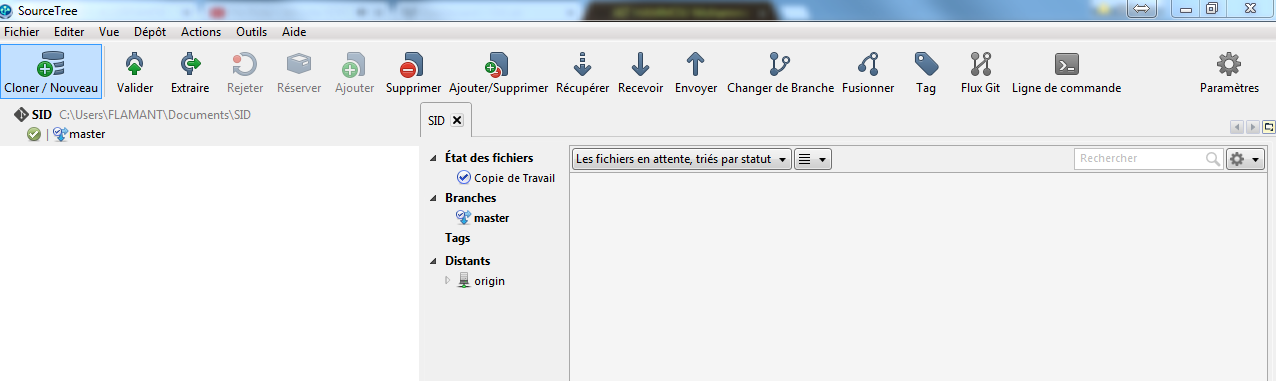
7. Si l’opération s’est bien déroulée, vous verrez le message suivant :

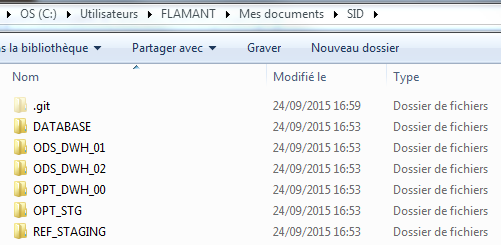


8. Valider en cliquant sur « **Cloner** ».

9. Attendre le clonage...

10. Le projet apparaît bien à l’accueil de SourceTree ainsi que dans l’explorateur de fichier.

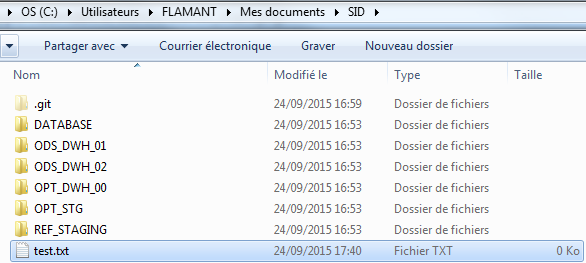


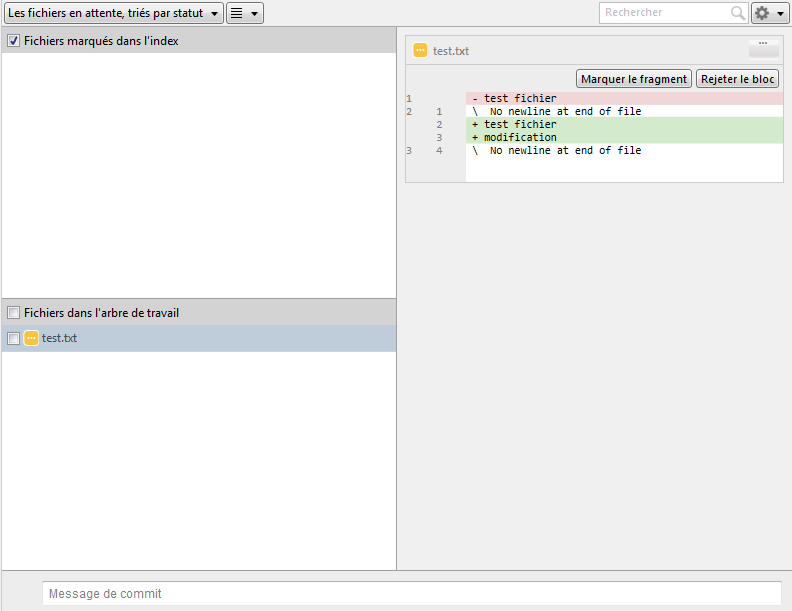




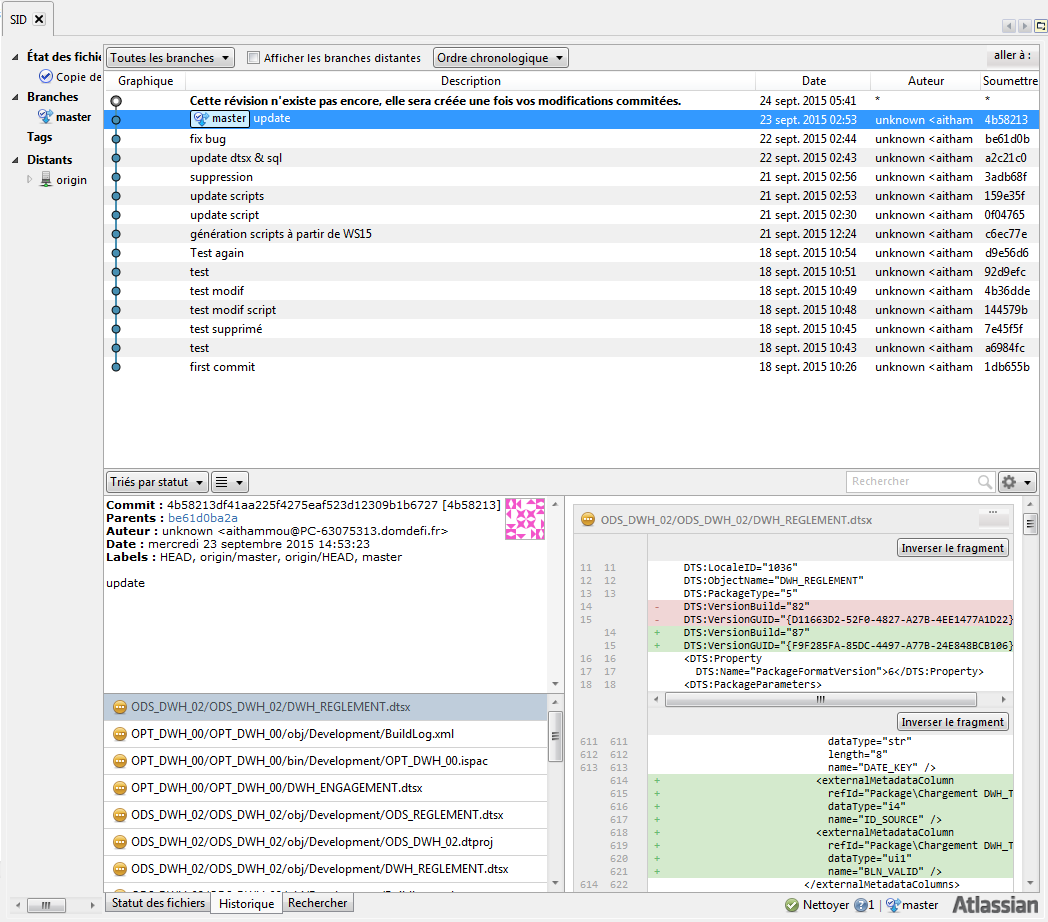
Cloner un projet GitLab

# COMMIT et PUSH vers le serveur distant

1. Ajouter / Modifier / Supprimer des sources dans le dépôt local.  
   Exemple, ajouter un fichier « test.txt ».  
     
   
2. Dans SourceTree, dans l’onglet Statut des fichiers, le fichier modifié/ajouté/supprimé apparait dans la fenêtre de l’arbre de travail, ainsi que les modifications effectuées :

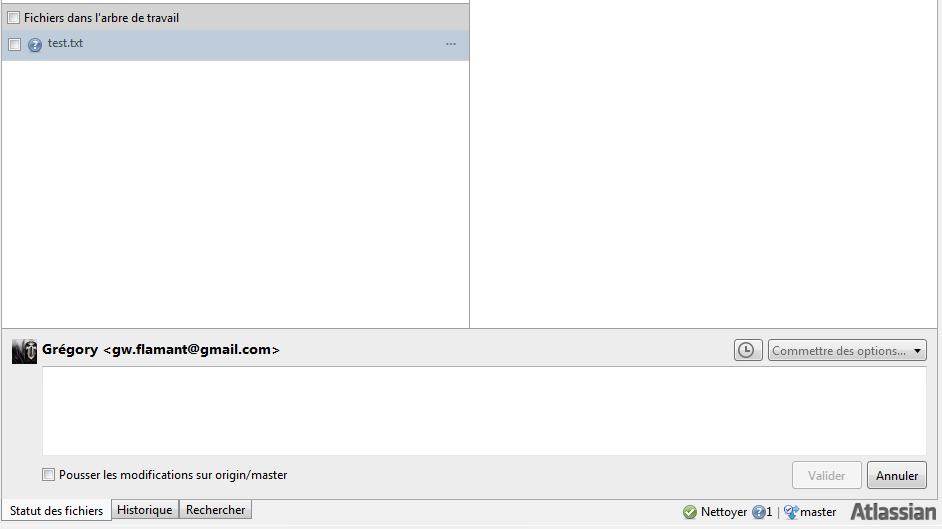
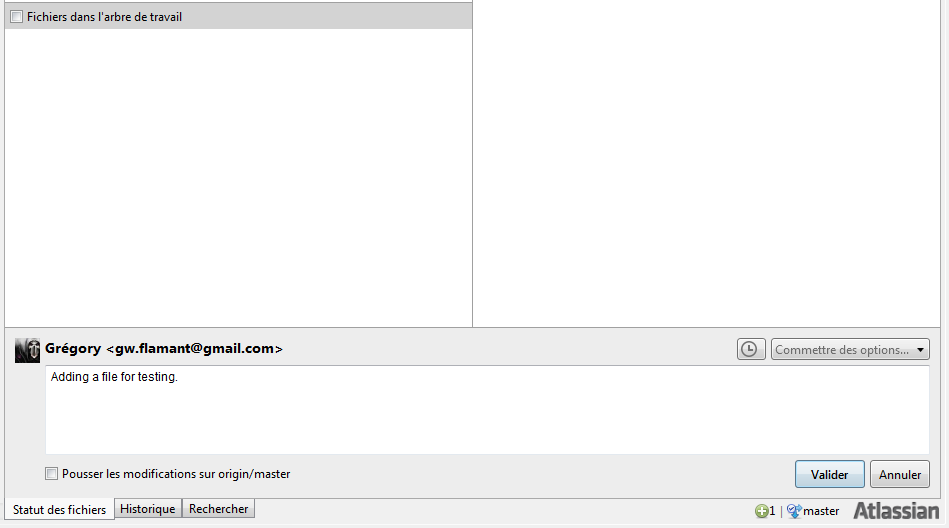
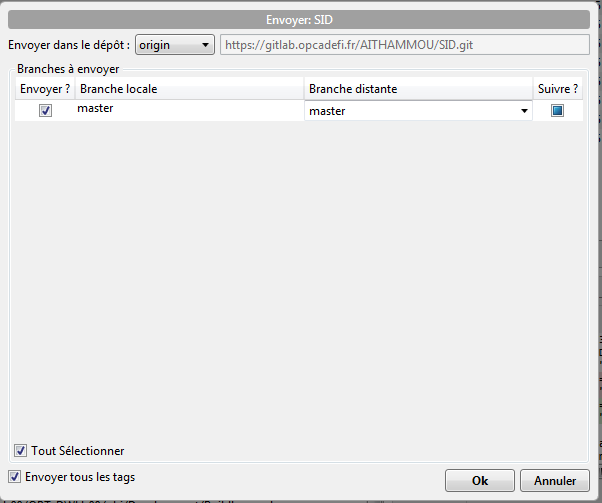


Taper un message de commit, marquer le fichier en le checkant (le fichier passe alors de la fenetre arbre de travail à la fenetre fichiers marqués) puis cliquer sur valider pour commiter LOCALEMENT les modifications éffectuées.

1. Dans SourceTree, dans l’onglet Historique, une nouvelle ligne apparaît :   
     
   

Nouvelle version à ajouter.

Version actuelle du serveur distant.

1. Dans l’onglet « **Statut des fichiers** », sélectionner les fichiers à « **COMMIT** ».  
     
    
2. Écrire la description du commit :  
     
    
3. Pour « **COMMIT**» uniquement, cliquer sur **Valider**.
4. Pour « **COMMIT + PUSH** », cocher la case « *Pousser les modifications sur origin/master* » puis cliquer sur **Valider**.
5. Si vous avez simplement « **COMMIT** », le bouton « Envoyer » a été mis à jour, avec le nombre de modification à pousser sur le serveur :   
     
     
     
     
     
     
     
     
     
     
     
   Cliquer sur le bouton « **Envoyer**» pour « **PUSH**» les modifications. Cliquer ensuite sur « **Ok**».   
     
    



COMMIT + PUSH

# Créer / travailler avec les branches

Avez-vous déjà enregistrer différente version d’un fichier ? Par exemple :

* story.txt
* story-joe-edit.txt
* story-joe-edit-reviewed.txt
* story\_V1.txt

Les branches accomplissent le même travail, avec plusieurs différences importantes :

* Les différentes versions sont « cachées » dans le repository, et le dossier de travail ne contient que la version de la branche en cours. On peut changer de branche dans le menu



* Une fois qu’on est satisfait de ses changements et qu’on veut les intégrer à la branche principale, on dispose d’outils de merging pour résoudre les conflits éventuels.

# Merging et Pull Requests



A travers les pull requests (récupérer), on peut proposer ses changements.

Où sont les outils de merging ?