**PPE JAVA**

**Hébergement WEB**

**Et**

**Versioning**



Table des matières

[Introduction : 3](#_Toc478666278)

[Créer un dépôt local 3](#_Toc478666279)

[Enregistrer des modifications 4](#_Toc478666280)

[Envoyer un dépôt local vers Bitbucket 5](#_Toc478666281)

[Comment envoyer un dépôt local vers bitbucket ? 6](#_Toc478666282)

[Comment éditer un fichier sur Bitbucket ? 7](#_Toc478666283)

[Comment récupérer les modifications des fichiers distants (en ligne) ? 7](#_Toc478666284)

[Récupérer les modifications du dépôt distant 7](#_Toc478666285)

[Intégrer une nouvelle fonctionnalité 9](#_Toc478666286)

[Annuler les modifications d’un fichier de code 10](#_Toc478666287)

[Ignorer une bibliothèque 10](#_Toc478666288)

[Effectuer un commit pour tous les fichiers modifiés/ajoutés 12](#_Toc478666289)

[Envoyer les commits au dépôt distant 12](#_Toc478666290)

[Afficher les différences entre deux commits 12](#_Toc478666291)

[Cloner un dépôt 14](#_Toc478666292)

[Gérer/exploiter des branches 14](#_Toc478666293)

[Pour créer une branche 14](#_Toc478666294)

[Pour fusionner 2 branches 15](#_Toc478666295)

[Lier NetBeans 7.1 à un dépôt privé sur github.com 16](#_Toc478666296)

[Installation de Git 16](#_Toc478666297)

[Configuration de NetBeans 17](#_Toc478666298)

[Création du projet 17](#_Toc478666299)

[Cloner le dépôt 19](#_Toc478666300)

[Créer Branches 21](#_Toc478666301)

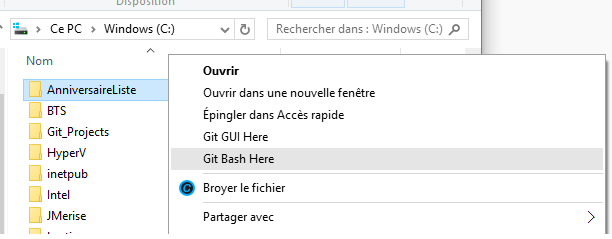
# Introduction :

GitHub est un service web d’hébergement et de gestion de développement de logiciels, en ayant la particularité d’être un outil de versioning.

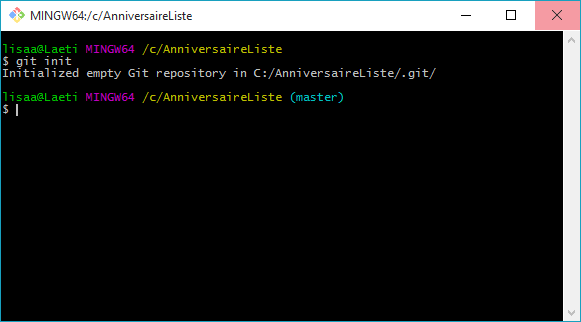
Le versioning permet de conserver une trace des modifications successives apportées à un fichier numérique à travers un logiciel spécialisé (nous utiliserons ici NetBeans).

# Créer un dépôt local

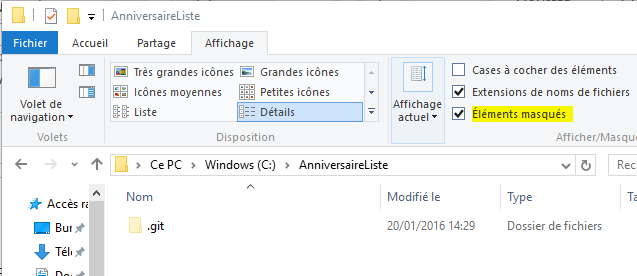
Afin de démarrer le développement d’un projet, il faut commencer par créer un dépôt local sur son poste. C’est dans ce dossier que l’on va avoir tous les fichiers nécessaires pour le projet.  
Pour cela il faut tout d’abord, créer un nouveau dossier sur le disque (C:), le nommer avec le nom du projet (ex : AnniversaireListe). Ensuite il faut faire un clic droit sur le dossier AnniversaireListe et cliquer sur Guit Bash Here :



Ce qui ouvre une interface de ligne commande.  
On tape ensuite ***git init*** dans l’interface de commande puis on appuie sur la touche entrée.



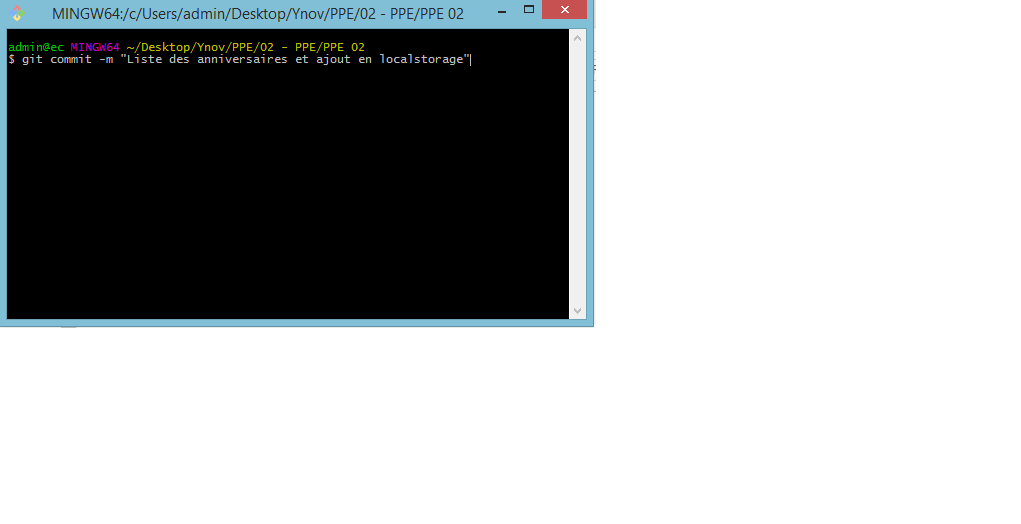
Cette commande va créer un nouveau dépôt Git dans le dossier courant. Ce qui signifie que Git va maintenant pouvoir être utilisé. Git a créé un dossier .git dans le dossier courant, il est caché par défaut. On peut le voir en cochant « Eléments masqués » dans l’onglet Affichage.



Ce dossier caché va contenir tout le contenu du dépôt. Il faut tout de même savoir que c’est uniquement Git qui doit gérer les fichiers de ce dossier. Une simple modification dans ce dossier pourrait corrompre le dépôt et l’empêcher de fonctionner correctement.

# Enregistrer des modifications

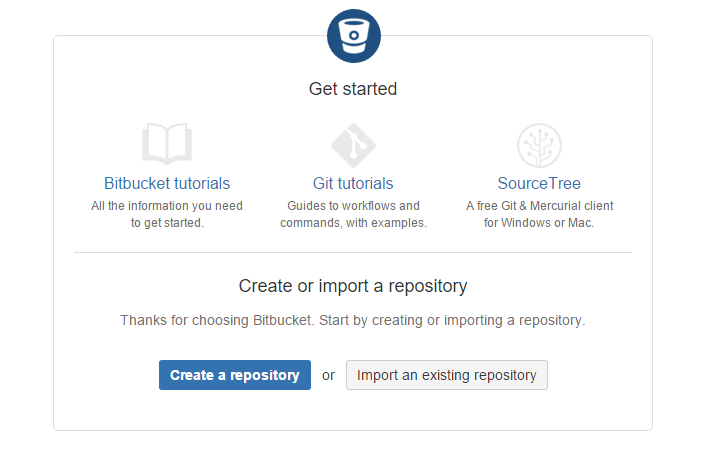
Un commit représente une modification unique servant à combler un besoin précis comme le correctif d’un bug, une modification des fichiers internes au dossier auquel on applique les modifications grâce au GIT (Bash). Commiter les fichiers (ajoutés à l’index) les ajoute à l’historique de Git.   
Un commit stocke les changements de l’Index dans le dépôt et doit contenir un commentaire servant à décrire l’origine de la modification escomptée par exemple : ***git commit –m « Liste des anniversaire et ajout en localstorage »***



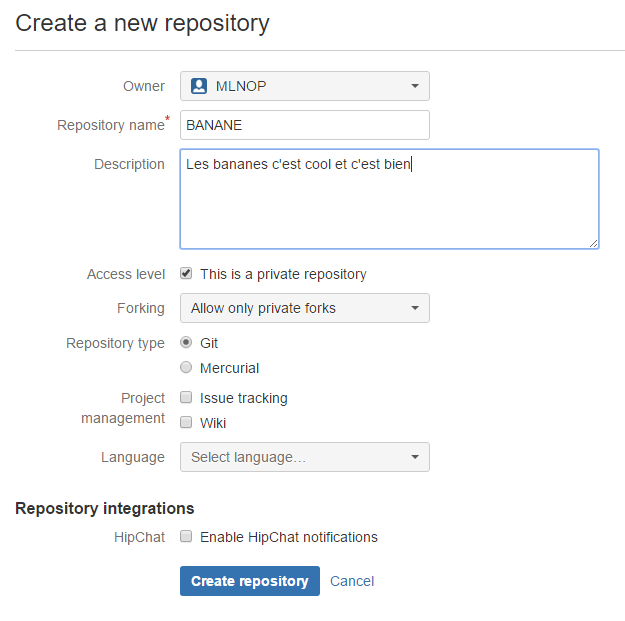
# Envoyer un dépôt local vers Bitbucket

Bitbucket comparé à Github favorise les petites équipes de développeurs, le dépôt de fichiers est illimité mais le nombre de membres est limité à 5.

Après avoir créé un compte et avoir validé le compte avec l’e-mail de confirmation, vous arrivez sur cette page :



Il faut maintenant créer un dépôt, donc cliquer sur le bouton « Create a repository » et ensuite suivre les instructions données par le site.



Le dépôt étant créé vous pouvez y accéder grâce au menu sur la gauche.

## Comment envoyer un dépôt local vers bitbucket ?

Ouvrir le Git Bash dans le dossier voulu et faire les commandes :

***git remote add origin*** [**https://bitbucket.org/MLNOP/BANANE.git**](https://bitbucket.org/MLNOP/BANANE.git)

Fait passer les fichiers sur Bitbucket.

***git push -u origin –all***

Envoie tous les git remote en une seule commande.

***git push -u origin –tags***

Transfert les différents tags sur le serveur.

Les éléments sont maintenant envoyés sur Bitbucket.

## Comment éditer un fichier sur Bitbucket ?

Il faut donc cliquer sur le lien source du menu, ainsi les fichiers présents à la racine du dépôt sont affichés. Afin d’éditer un fichier il suffit de cliquer sur le nom de ce dernier.



3 options s’offrent alors à nous :

\_Blame qui permet d’afficher les informations sur les lignes des fichiers.

\_Raw qui permet de télécharger ou d’afficher le fichier sous sa forme de base sans fioritures.

\_Edit permet bien évidement d’éditer le fichier.

## Comment récupérer les modifications des fichiers distants (en ligne) ?

Afin de connaître les commits effectués en ligne il suffit d’entrer la commande :

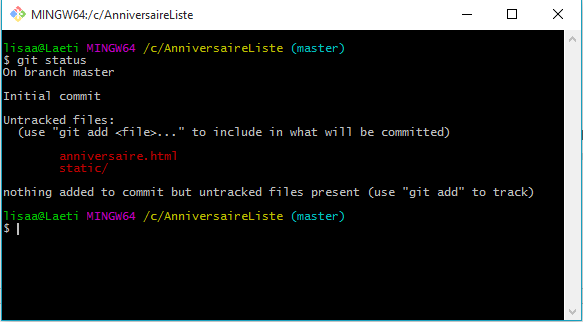
***git pull***

Afin que le fichier distant s’actualise localement.

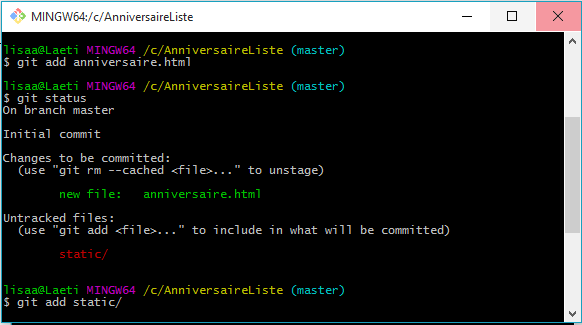
Vérifier avec ***git log*** la mise à jour du fichier.

# Récupérer les modifications du dépôt distant

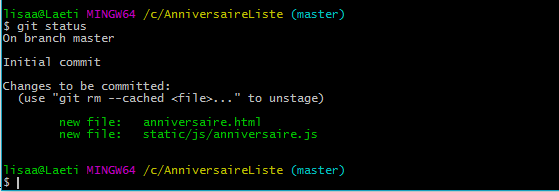
Tout d’abord, ouvrir la fenêtre de commande de Git en faisant un clic droit sur le fichier AnniversaireListe (cf : Créer un dépôt local) et tapez la commande ***git status*** pour pouvoir regarder les modifications entre son répertoire de travail et ce qui est présent dans la dernière version du dépôt.



Les fichiers qui ont été modifiés s’affichent en rouge. Pour les récupérer il faut taper la commande ***git add* *nom-du-fichier-que-l’on-veut-récupérer***ici par exemple on va taper ***git add******anniversaire.html***puis ***git add static/***   
(On peut aussi taper la commande ***git add –all*** qui va ajouter tous les fichiers nouveaux ou modifiés.)



Enfin, faire un ***git status***pour vérifier que la récupération des fichiers a bien été effectuée.



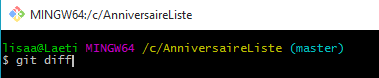
Les fichiers récupérés s’affichent alors en vert.

# Intégrer une nouvelle fonctionnalité

Afin d’ajouter une fonctionnalité il faut modifier le fichier html (ici on va modifier le fichier anniversaire.html). On souhaite ajouter une couleur de son choix pour chaque anniversaire, par conséquent on va devoir ajouter un champ pour définir la couleur et on ne va pas oublier de modifier la fonction (« anniv\_ajout() » et « anniv\_liste() ») du fichier : static/js/anniversaire.js

Afin de relire le code ajouté ou supprimer avant de commiter, on peut taper la commande :

***git diff***



Git donne une réponse sous forme de ligne. Les lignes en rouge avec un – devant sont les lignes qui ont été supprimées. Les lignes en vert avec un + devant sont les lignes qui ont été ajoutées. Faire un ***git diff*** avant d’ajouter les fichiers à l’index fait partie des bonnes pratiques.

Une fois ces modifications vérifiées il faut ajouter les fichiers à l’index Git. On le fait grâce aux commandes :

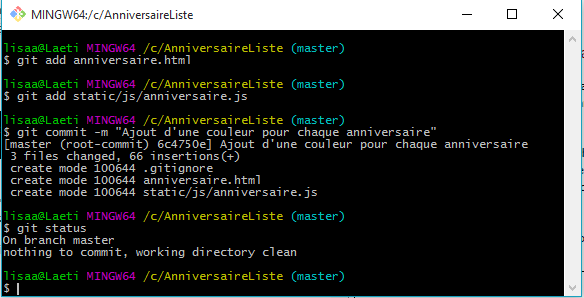
***git add anniversaire.html***

***git add static/js/anniversaire.js***

Puis, il faut commiter :

***git commit -m « Ajout d’une couleur pour chaque anniversaire »***

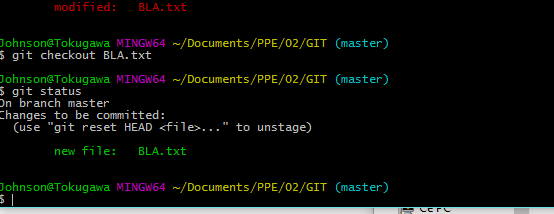
Et enfin vérifier que tout a bien été fait avec un ***git status :***



# Annuler les modifications d’un fichier de code

Pour annuler les modifications d’un fichier, il suffit simplement de rentrer la commande suivante dans Git :

***git checkout anniversaire.html***

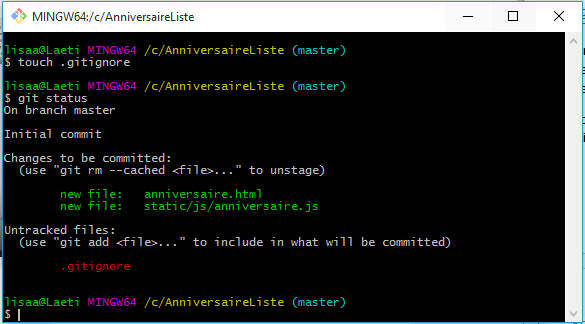


Ici on a annulé les modifications d’un fichier de code mais on peut aussi faire un ***git checkout identifiant-du-dernier-commit*** pour retourner à l’avant-dernière version de l’avant-dernier commit.

# Ignorer une bibliothèque

Ignorer une bibliothèque est utile par exemple pour les fichiers générés ou compilés car le contenu des fichiers ne peut être suivi et est illisible et pour les fichiers de configuration contenant des mots de passe pour ne pas que tout le monde y ait accès.

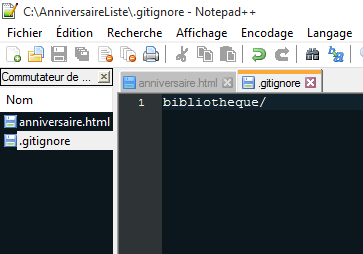
Pour cela il faut créer le fichier .gitignore à la racine du projet (dans le dossier qui contient le dossier .git masqué pour la plupart des systèmes). Pour cela on peut taper la commande ***touch .gitignore***



Puis ajouter la ligne suivante dans le ficher .gitignore :

*nom-du-dossier-que-l’on-veut-ignorer/*

Si tous les fichiers que l’on veut ignorer sont dans le dossier bibliotheque, il faut taper :



Puis enregistrer le fichier. Ce qui signifie que tous les fichiers contenus dans ce dossier ne seront pas pris en compte par Git, donc ils ne seront ni versionnés, ni affichés lors d’un Git status puisque Git va totalement ignorer ces fichiers.

**Astuce** : Les motifs glob fonctionnent dans les lignes de .gitignore, donc on peut utiliser « \* » pour remplacer un ou plusieurs caractères, « ? » pour remplacer un caractère, « [abc] » pour chercher un des trois caractères (soit a, b, ou c) ou « [0-9] » qui indique une valeur parmi toutes les valeurs entre la première et la dernière (soit, entre 0 et 9 ici).

# Effectuer un commit pour tous les fichiers modifiés/ajoutés

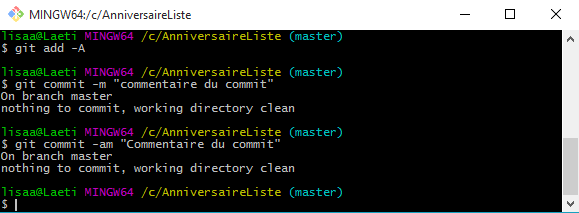
Commiter tous les fichiers un par un peut vite devenir une opération répétitive, c’est pour cela qu’il existe une commande permettant de commiter tous les fichiers simultanément grâce aux commandes :

***git add –A***

***git commit –m « … »***

On pourrait aussi faire :

***git commit -am « message de commit »***

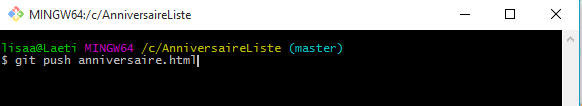


# Envoyer les commits au dépôt distant

Faire la mise à jour des commits sur le dépôt dit distant il suffit de faire la commande :

***git push nom-du-dossier-que-l’on-veut-pousser-sur-le-serveur***

Exemple :



Puis de vérifier avec un ***git status***.

# Afficher les différences entre deux commits

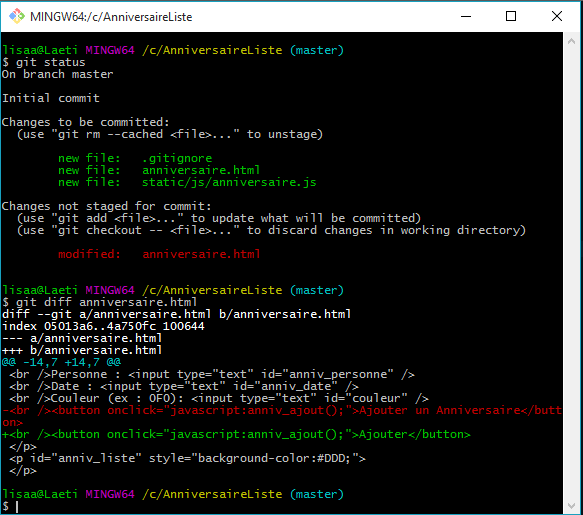
Dans un premier temps, on consulte la liste des commits avec la commande :

***git log***

Puis, pour regarder les modifications entre 2 commits précis, on utilise la commande :

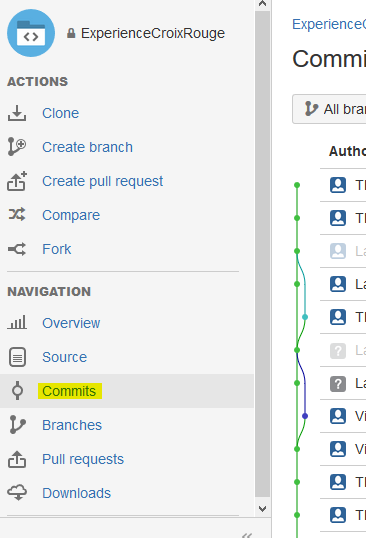
***git diff identifiant-du-commit1 identifiant-du-commit2***

Le commit2 étant le dernier commit, le commit1 étant l’avant-dernier commit. On peut taper que les 5 premiers caractères de l’identifiant des commits pour que ça marche.  
On peut aussi faire un ***git status*** pour voir les fichiers modifiés puis faire un ***git diff nom-du-fichier*** pour afficher les modifications apportées au fichier :



Git donne une réponse sous forme de ligne. Les lignes en rouge avec un – devant sont les lignes qui ont été supprimées. Les lignes en vert avec un + devant sont les lignes qui ont été ajoutées.

Il est également possible de voir les modifications apportés par un commit directement sur Bitbucket en appuyant sur l’onglet commit sur le menu à gauche de l’écran.



# Cloner un dépôt

Pour cloner un dépôt il faut Git sur le pc ainsi que l’URL de clonage sur Bitbucket. Cette URL se trouve sur la page principale de Bitbucket « Clone in SourceTree ». ensuite il faut accéder à l’interface en ligne de commande afin d’être dans le dossier où on veut placer son projet et écrire la commande :

***git clone https://rdauzon@bitbucket.org/rdauzon/***

***anniversaireliste.git anniversaireVacances***

Cette commande va copier le dépôt dans un nouveau dossier « anniversaireVacances ». On aura donc accès à tout le travail effectué précédemment.

# Gérer/exploiter des branches

Comme sur un arbre une branche correspond à une dérivation d’une autre branche. La branche master correspond au tronc du projet.

Un branche permet de modifier une partie du projet sans pour autant le modifier entièrement, c’est en quelque sorte une manière « sure » de faire des modifications.

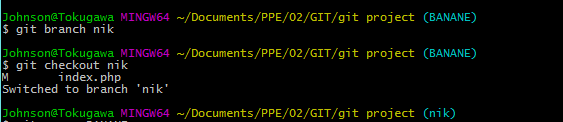
## Pour créer une branche

Il suffit de rentrer la commande :

***git branch …………………….***

Et ensuite pour se déplacer vers cette branche pour travailler dedans, la commande est :

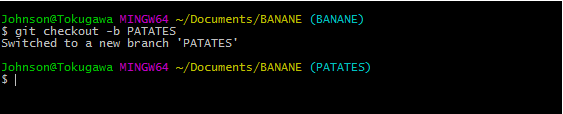
***git checkout …………………..***



Ou bien vous pouvez faire directement :

***Git checkout –b ………………………***

Pour créer et switcher directement à la nouvelle branche.



## Pour fusionner 2 branches

Tout d’abord aller dans la branche à fusionner :

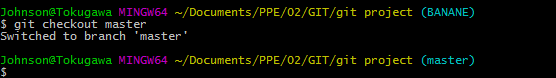
***git checkout ………………***

Puis par exemple pour fusionner avec la branche master :

***git merge master***

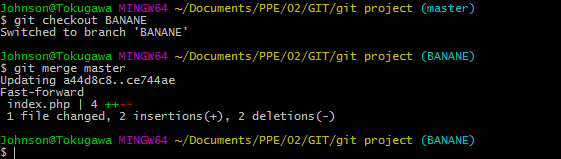
Intégrer les modifications de la première branche a la branche master :

***git checkout master***



Puis enfin fusionner les 2 branches :

***git merge ………………***



# Lier NetBeans 7.1 à un dépôt privé sur github.com

## Installation de Git

Il est nécessaire d'installer Git sur votre poste de développement. Téléchargez-le sur [http://gitscm.com/](http://git-scm.com/)

Une fois l'installation terminée, ouvrez le "Git Bash" et tapez les commandes de configuration suivantes :

git config --global user.name "Votre Nom"

git config –-global user.email votreadressemail@courriel.com

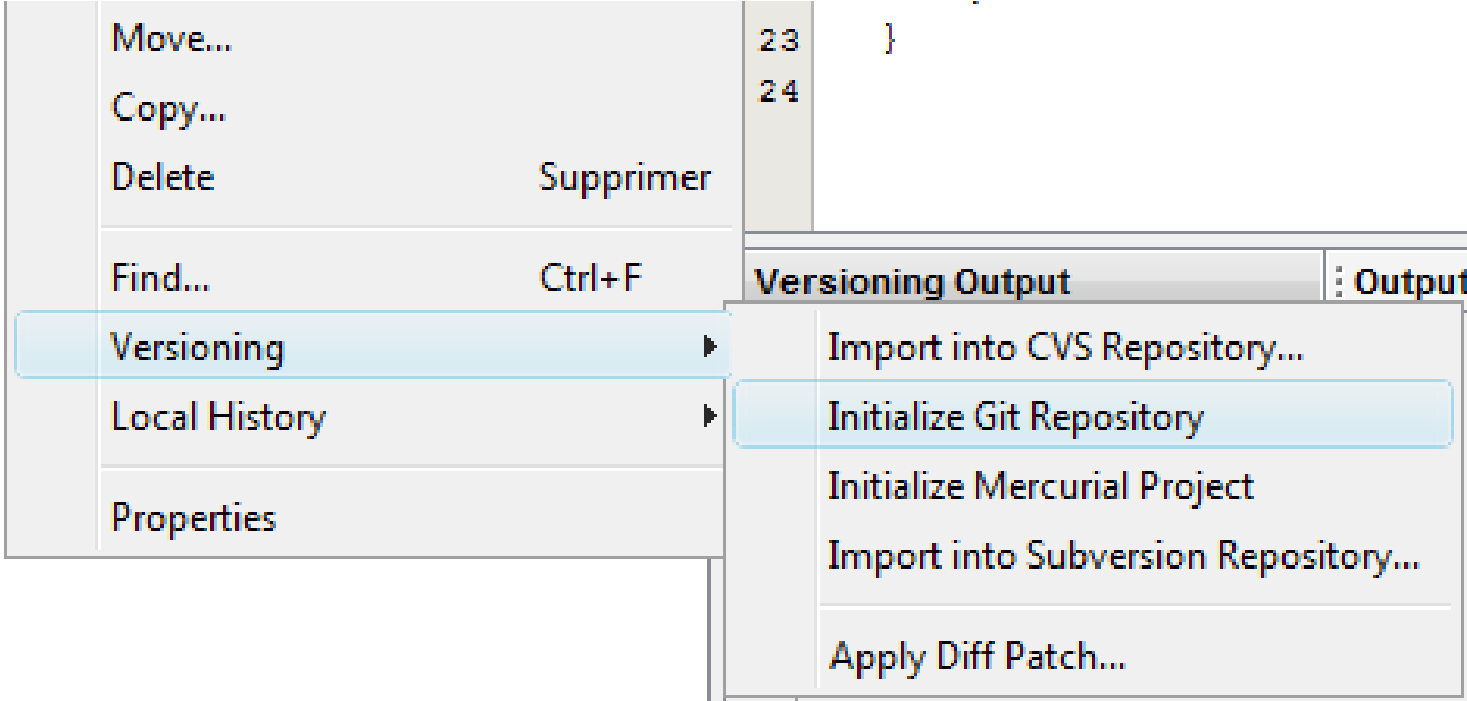
## Configuration de NetBeans

Le plugin Git est déjà inclut dans la version 7.1. Je vous conseille d'utiliser la version 7.1 et non pas une version antérieure car le plugin Git pour les versions antérieures ne fonctionne pas très bien.

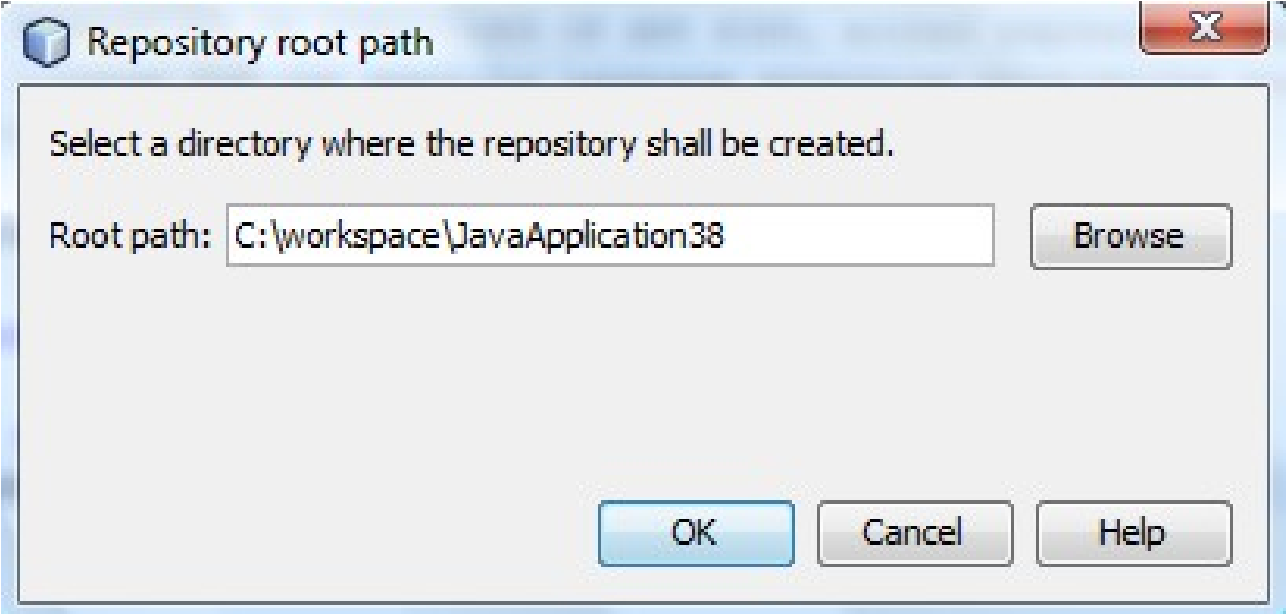
## Création du projet

Créez votre projet Java dans NetBeans. Une fois la création complétée, cliquez avec le bouton droit sur le projet dans l'explorateur de fichiers et sélectionnez :

Versioning > Initialize Git Repository



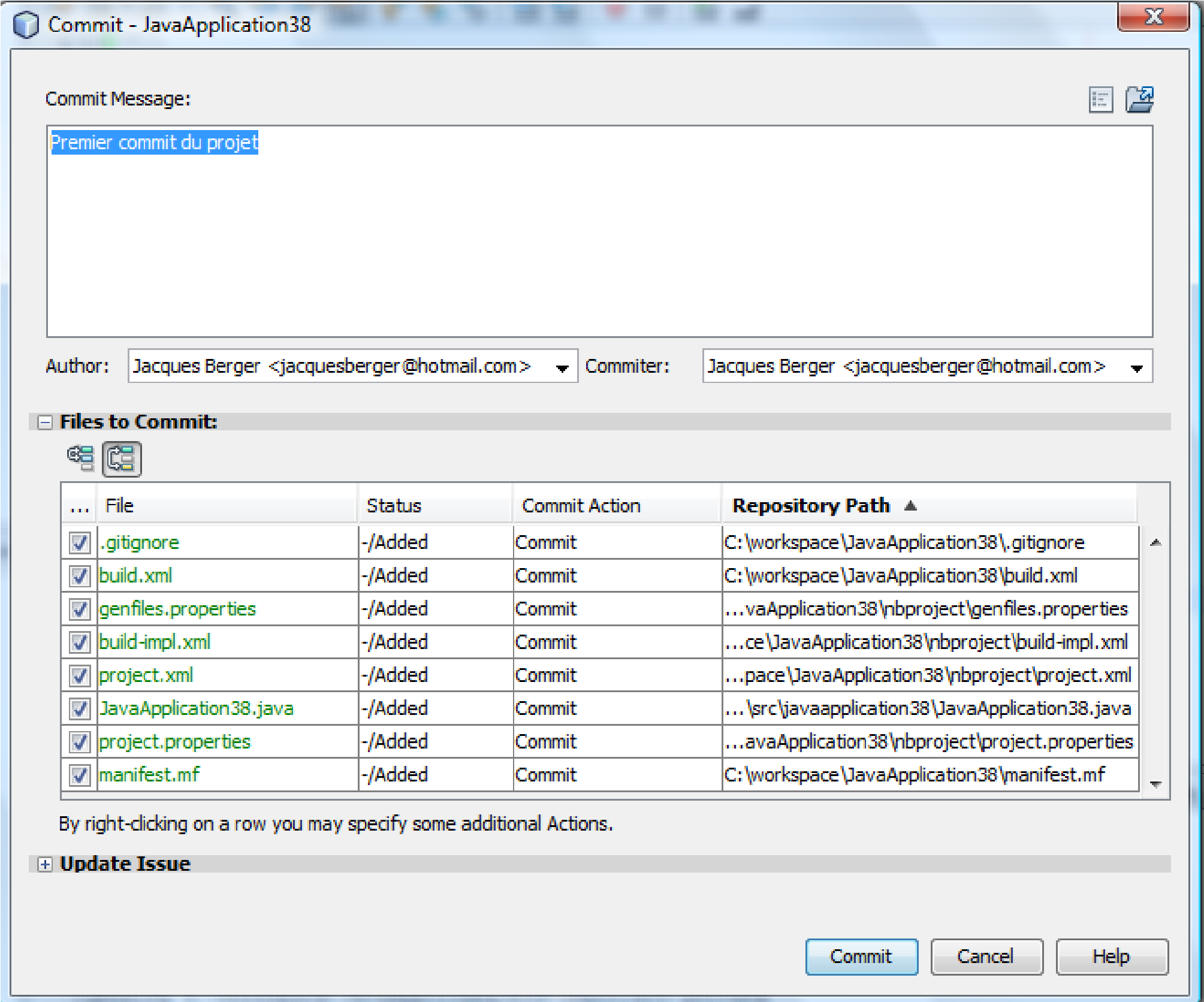
Choisissez le répertoire qu'on vous propose.



Faites votre premier commit sur le projet. On clique à droite sur le projet :

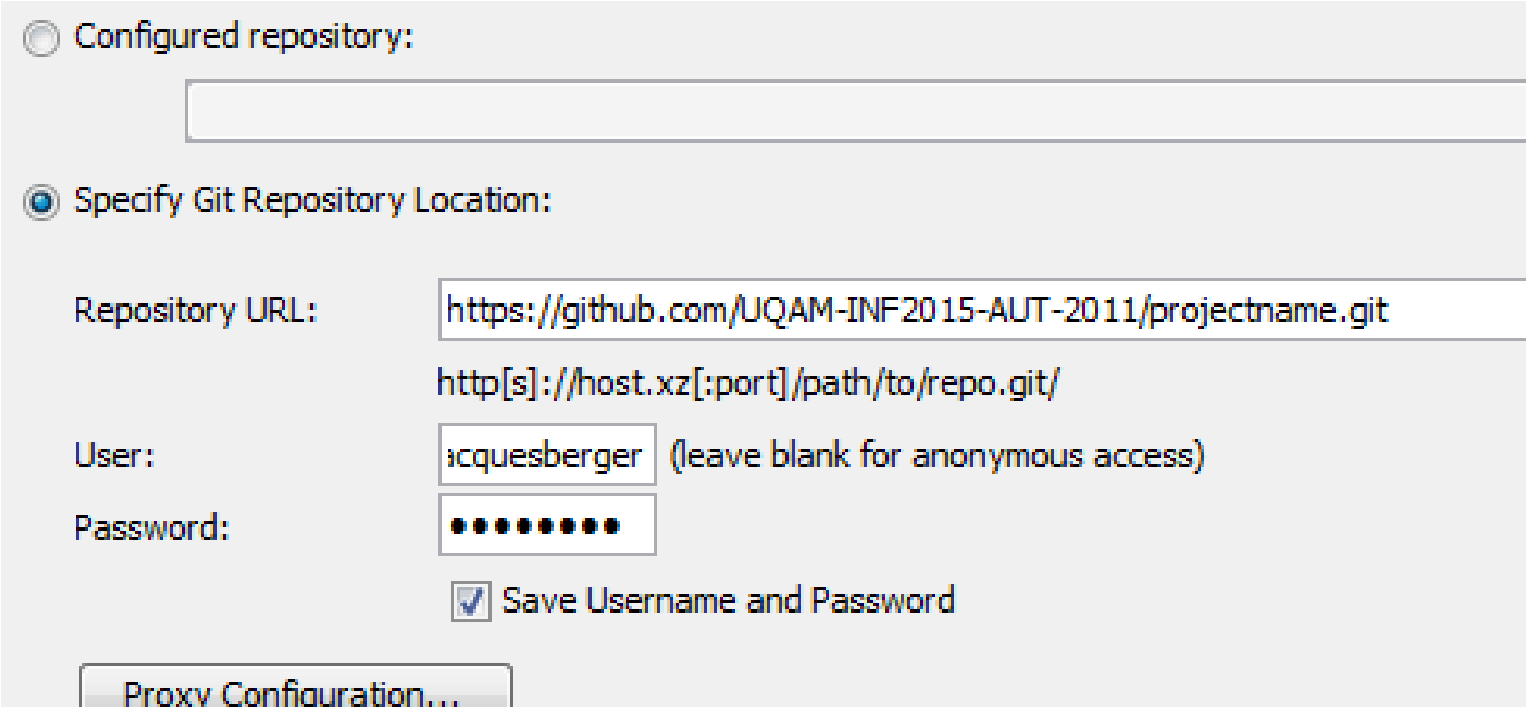
Git > Commit...

Ça nous amène à la fenêtre suivante :



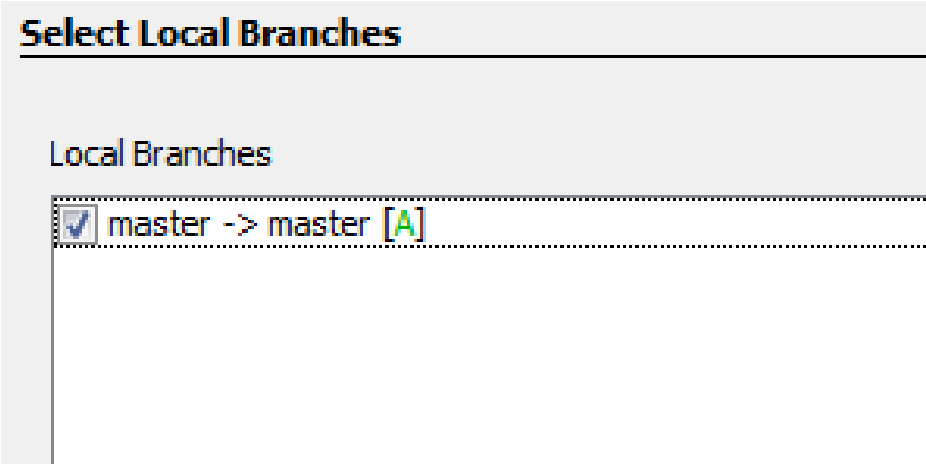
Appuyez sur le bouton Commit. Et voilà le commit est terminé. Maintenant, on doit faire le Push sur github. On clique à droite sur le projet et on sélectionne :

Git > Remote > Push



Dans le champ Repository URL, mettez l'URL de votre dépôt github suivi de ".git". Le champ User est votre nom d'utilisateur sur github et votre mot de passe est votre mot de passe github. Appuyez sur le bouton Next.

Si tout se passe bien, on arrive à la fenêtre suivante :



Sélectionnez la branche master et appuyez sur Finish.

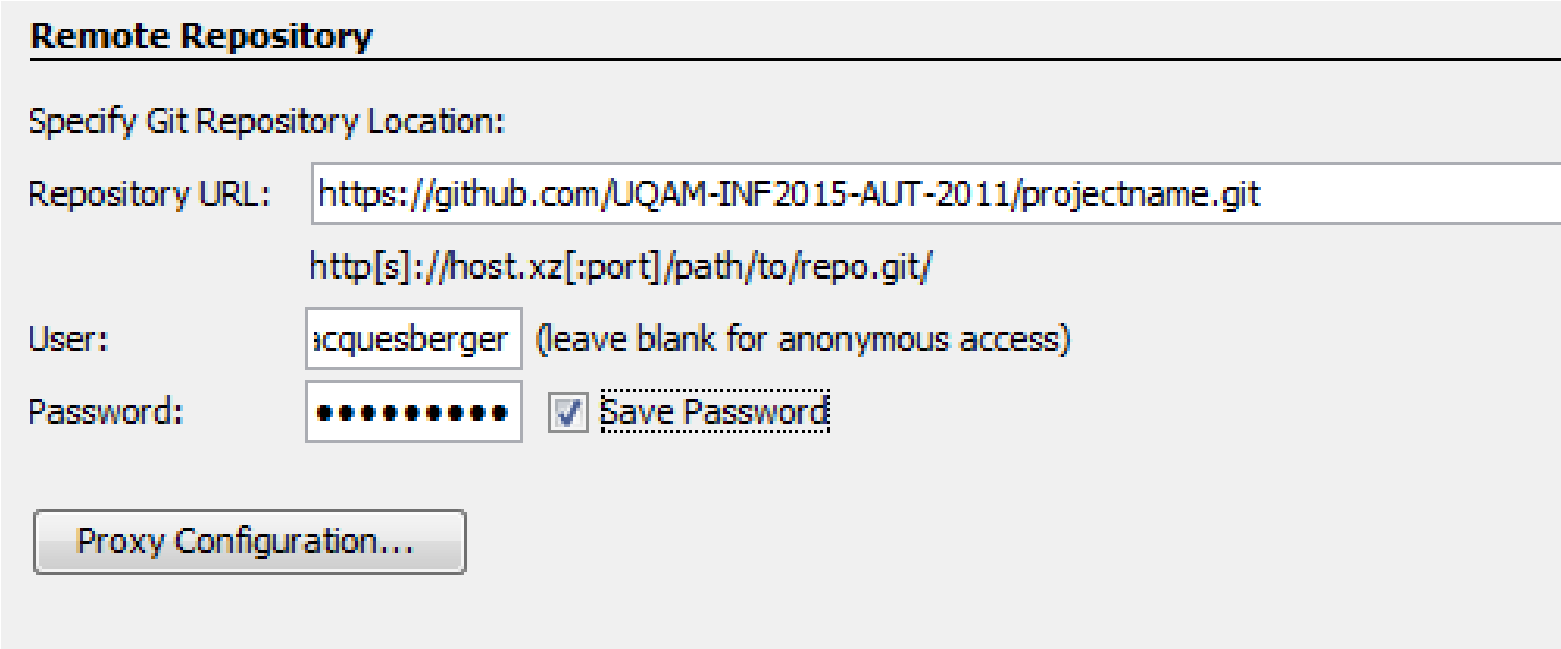
Et voilà, c'est terminé.

## Cloner le dépôt

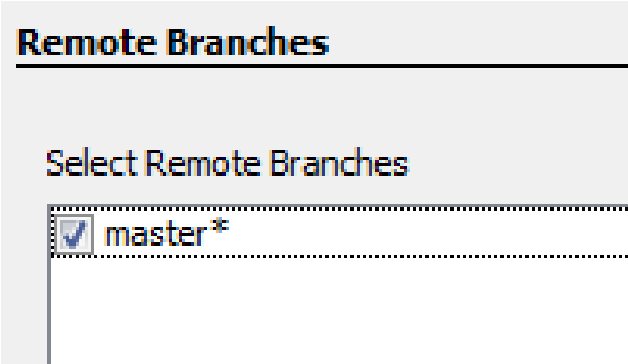
Pour les autres membres de l'équipe, ils devront se créer une copie locale du dépôt pour pouvoir y travailler. Ce concept s'appelle "cloner un dépôt". Pour le faire dans NetBeans 7.1 :

Projet > Git > Remote > Clone...

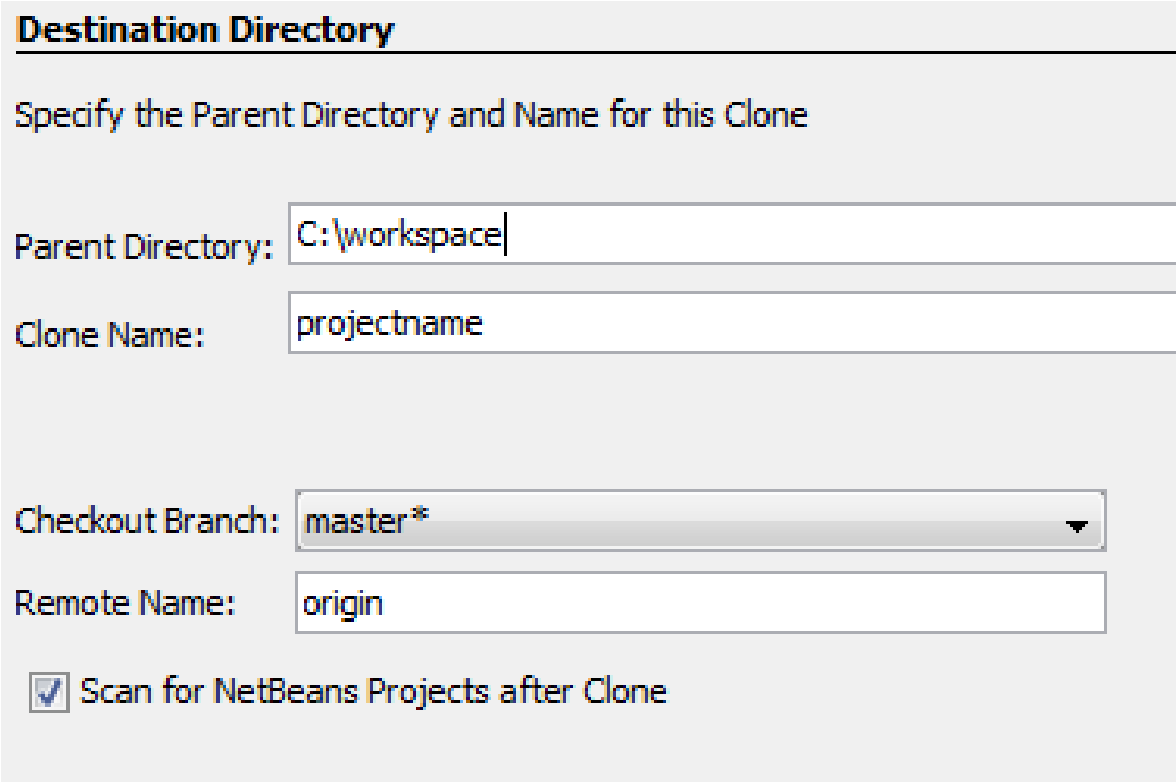
Entrez les informations du dépôt github comme lors de la création du dépôt :



Appuyez sur Next. Choisissez la branche master et appuyez sur Next encore.

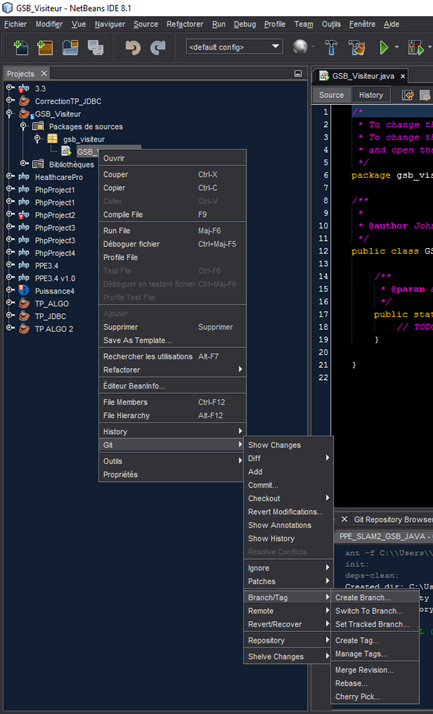


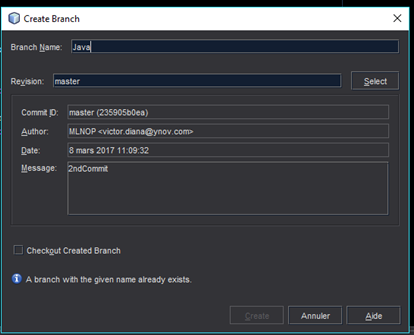
Placez le répertoire du projet où vous voulez, habituellement les valeurs proposées par NetBeans sont correctes. Appuyez sur Finish quand vous êtes prêt.



Si tout se passe bien, NetBeans vous demandera d'ouvrir le projet et c'est terminé.

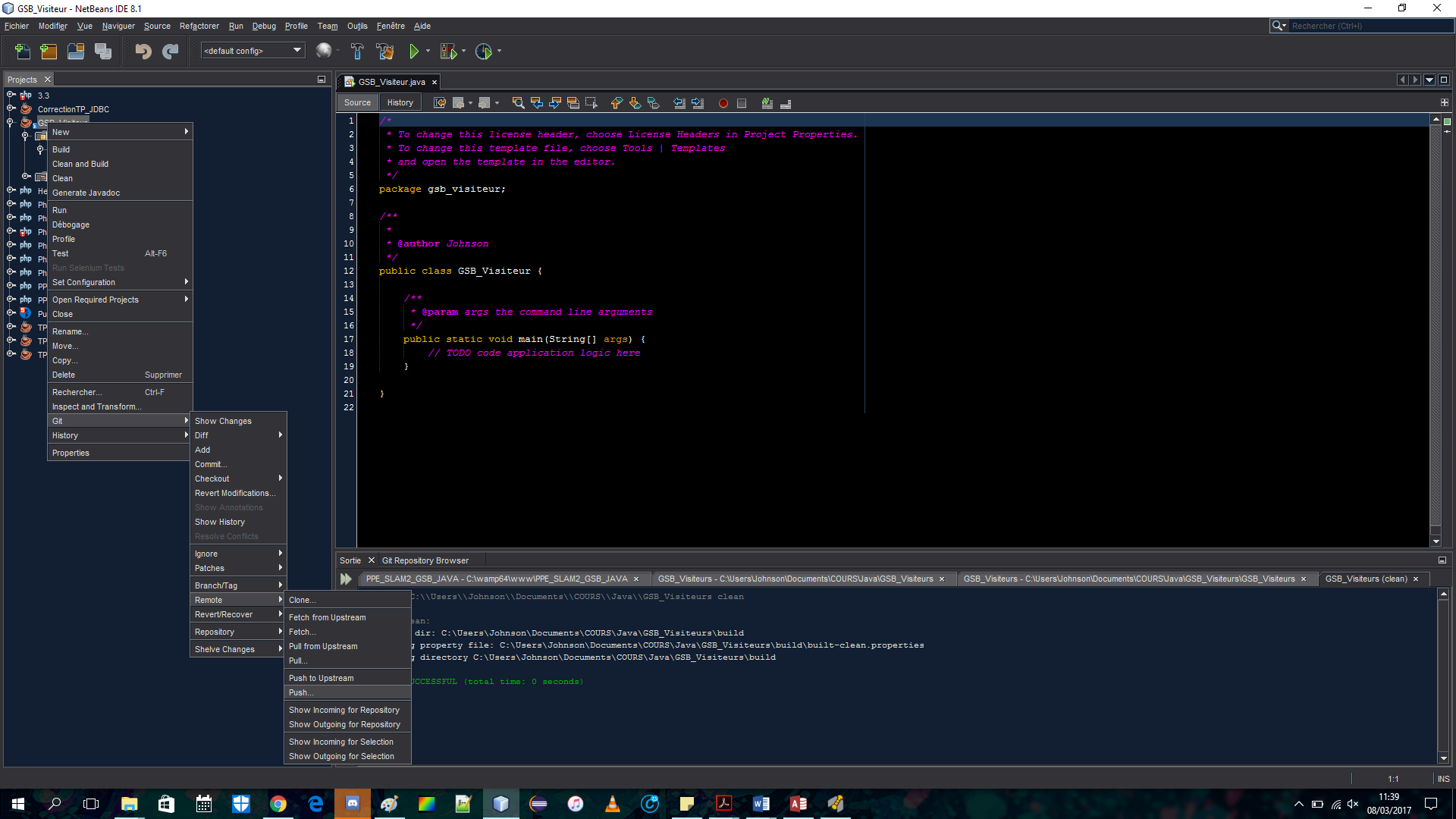
## Créer Branches

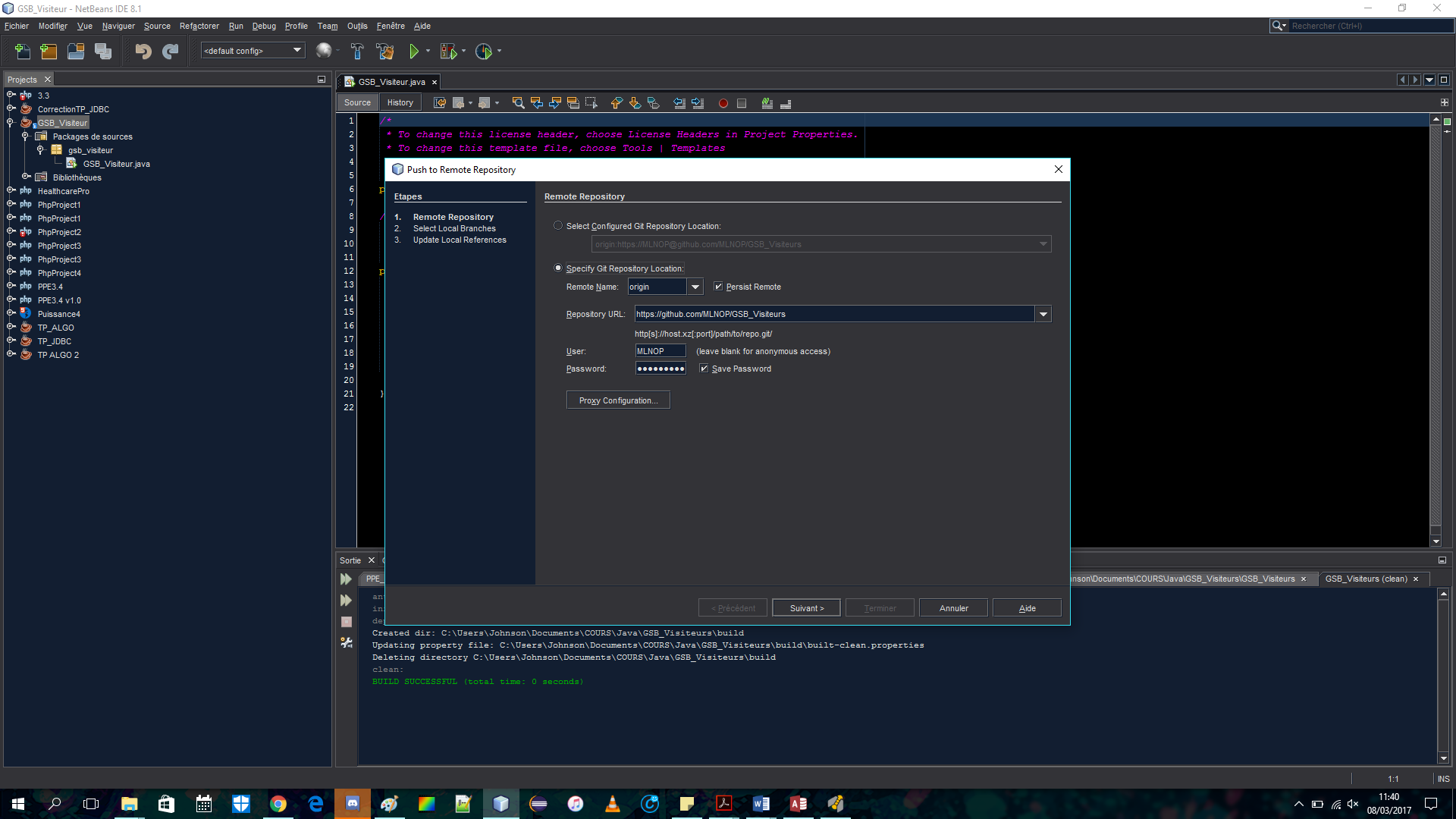




Révision : Branche mère (branche créée au-dessus de la nouvelle)

Bien sur ne pas oublier de push la request de création de branche





Copier l’url du Dépot GitHub en ligne et rentrez vos identifiants