TP1: GIT & GITHUB

Objectifs :

- Comprendre l'utilité d'un logiciel de gestion de version décentralisé .
- Savoir utiliser Git de façon basique.
- Savoir utiliser GitHub de façon basique

Partie 1. Installation d'EGit :

1. Installer les plugins egit qui vont bien au sein d'Eclipse

   a. Dans la barre de menu, cliquez sur le menu *Help*, puis sélectionner *Install New Software...* :
   b. *Ajouter un nouveau site distant* : http://download.eclipse.org/egit/updates/
   c. *Une fois l'installation terminée, Eclipse va vous demander si vous voulez le redémarrer, faites le (bouton Restart Now).*

Jetez un coup d'œil sur :

- Les préférences: `Window > Preferences > Team > Git`
- La perspective: `Window > Open perspective > Others > Git Repository Exploring`
- Vues: `Window > Show View > Others ... > Git`

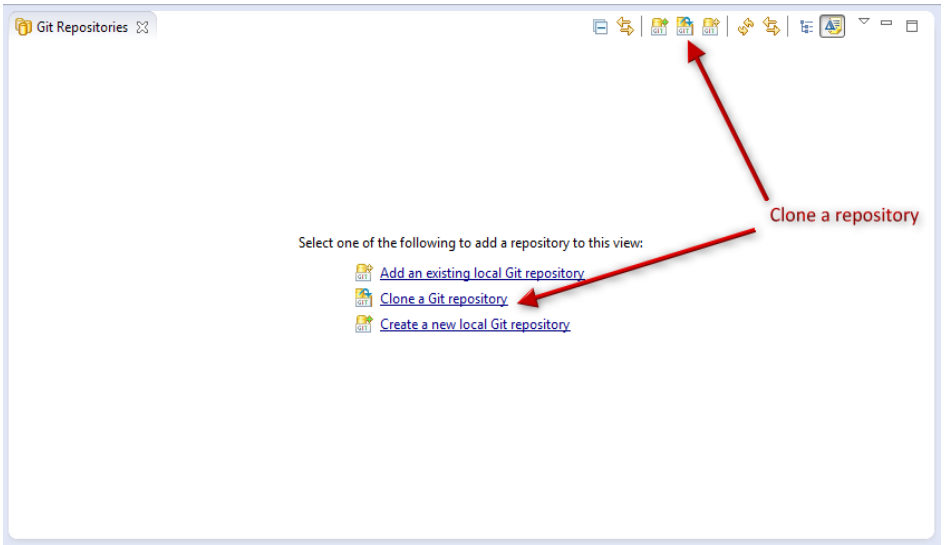
Partie 2. Création du repository sous GitHub

1. Pour commencer, rendez-vous dans le backoffice de **GitHub** après vous être authentifier. Lorsque vous êtes sur la liste de vos repositories, cliquez juste sur le petit bouton « add repository ».
2. Ensuite, vous nommez votre Repository, ajoutez une description**. *SURTOUT, n'oubliez pas de sélectionner d'initialiser le repository avec un README***. A la suite de quoi, vous arriverez sur le récapitulatif des commandes git et des liens servant à rattacher votre projet eclipse.

Partie 3. Cloner le dépôt

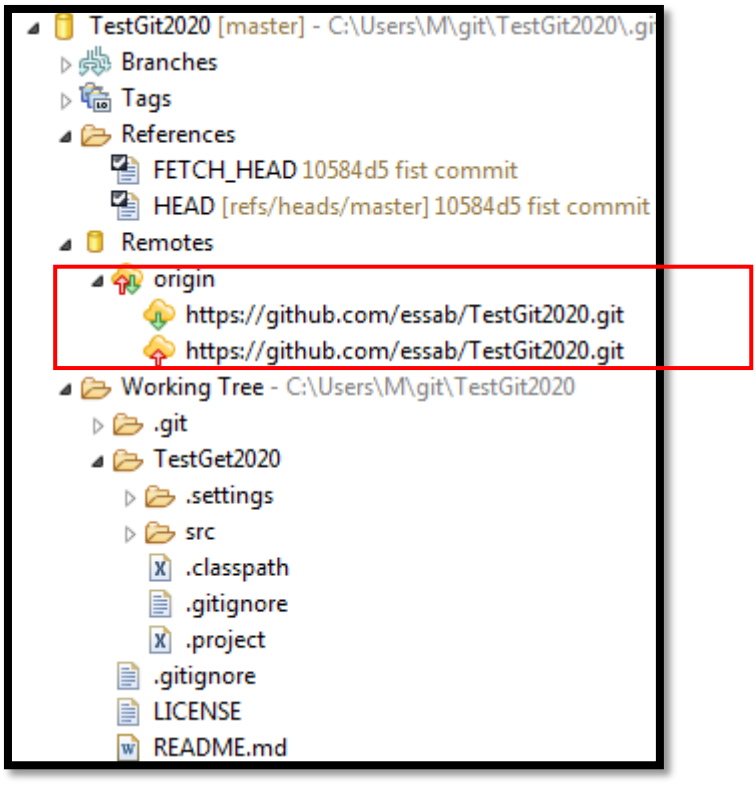
Vous allez maintenant accéder à votre dépôt à partir d'Eclipse.

1. Passer en perspective GIT



2. Dans la vue 'Git Repositories', cliquer sur 'clone a Git Repository'.
3. Dans 'Clone Git Repository', remplissez les champs. Vous pouvez trouver l'URL de votre dépôt sur la page web de celui-ci dans gitHub.

A cette étape, vous devez avoir une copie locale de votre dépôt. Celui ci est visible dans la vue 'Git Repositories'.



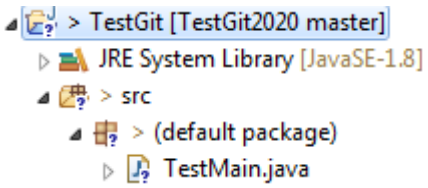
origin est le nom par défaut que Git donne au serveur a partir duquel vous avez cloné

Partie 4. Rattachement de son projet à GitHub

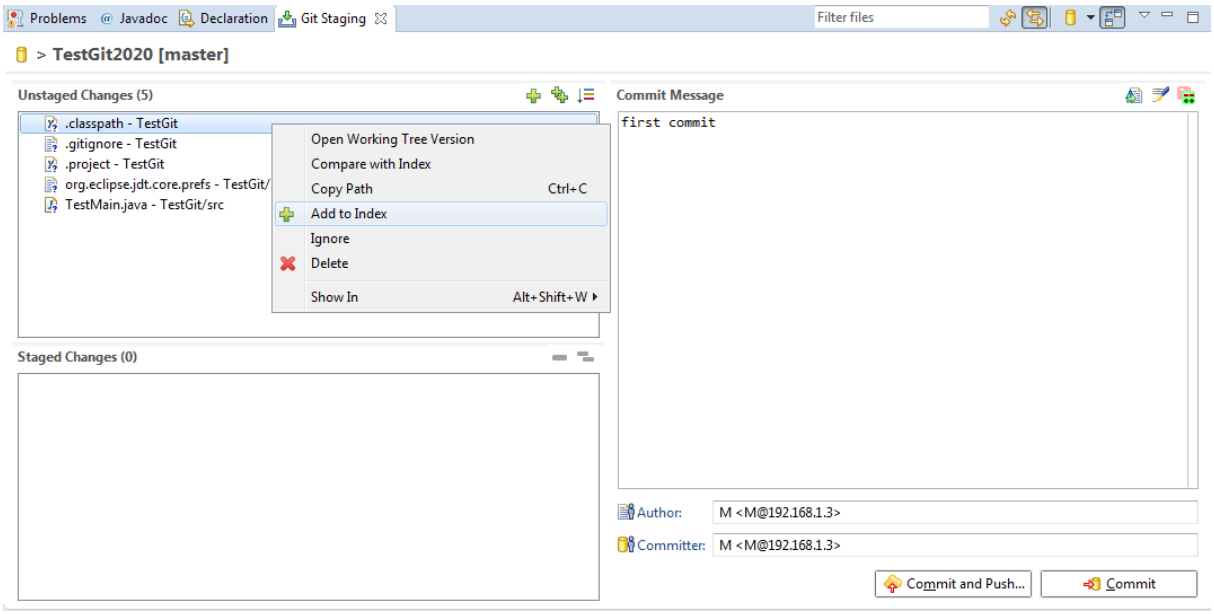
4. Sous Eclipse, créer un projet java.
5. Créer une classe de TestMain.java

6. Quand le projet est créé, vous devez indiquer à quel Repository GIT vous voulez le relier. Pour cela, faites un **clic droit sur votre** `projet > Team > Share Project`

A cette étape, vous devez avoir un projet contrôlé par GIT. Le nom de votre projet doit être suivi du nom du dépôt et de la branche.

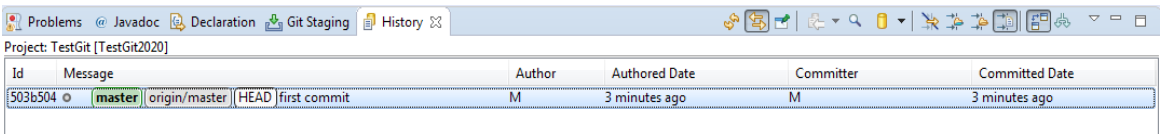


A partir de maintenant, on peut commencer les commits locaux. Oui, car GIT utilise deux choses lors du partage des données. Le commit local qui permet de commiter seulement à soi (dépôt local) le commit distant, appelé **Push**, permettant d'envoyer toutes ses modifications aux autres collaborateurs.



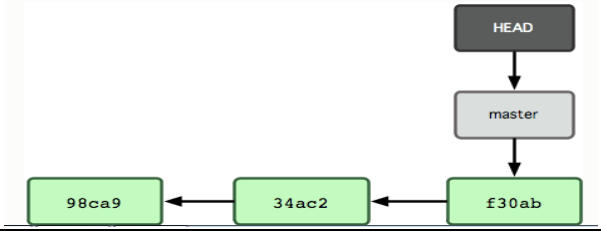
7. Faire un Commit local : vous n'avez qu'à cliquer sur le `projet => team => commit` (exactement comme pour celle de SVN).
8. Il ne nous reste plus qu'à envoyer nos commit locaux sur GitHub. Il suffit de refaire clic droit sur le `projet => team =>Push to Upstream`.

   faites un **clic droit sur votre** `projet > Team >show in History`

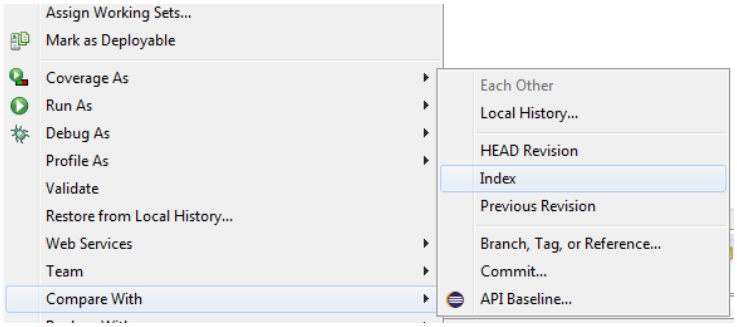


La branche par défaut dans Git s'appelle **master**. Au fur et à mesure des validations, la branche master pointe vers le dernier des commits réalisés. À chaque validation, le pointeur de la branche master avance automatiquement.

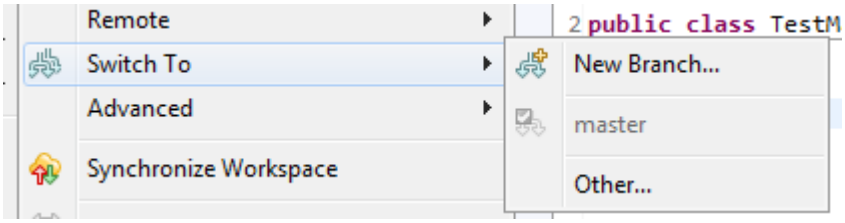
Comment Git connaît-il la branche sur laquelle vous vous trouvez ➔ Il conserve un pointeur spécial appelé **HEAD**.



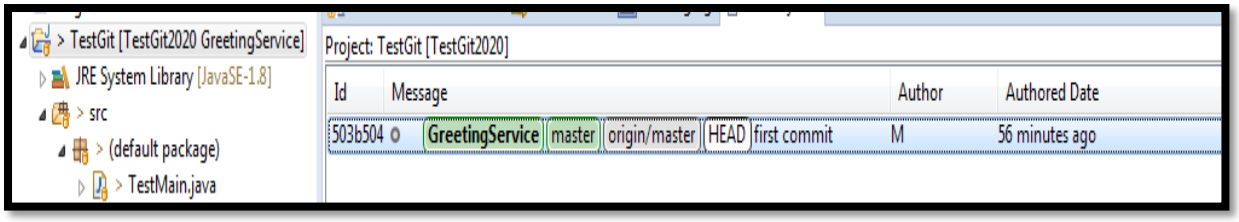
9. Modifiez la classe TestMain.java.
10. Faites un clic droit sur Main.java et cliquez sur `Compare with > Git Index.`



11. Créez la branche *GreetingService* : Choisissez Create Branch... dans le menu contextuel Historique. Ou choisissez `Team > Switch To > New Branch...` dans le menu contextuel de l' Explorateur de packages .

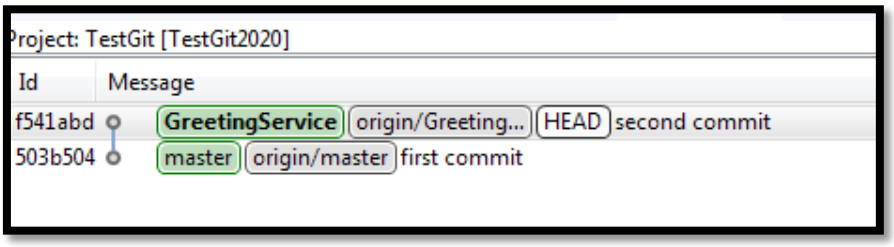


12. La branche actuelle est *GreetingService*



13. Créer une classe de *TestMainGreetingService*.java
14. Envoyer les modification : `projet => team =>Push to Upstream`

15. Faites un **clic droit sur votre** `projet > Team >show in History`



16. Retourner vers la branche master.
17. Créer une classe de *TestMainMaster*.java
18. Fusionner la branche master avec GreetingService (Choisissez `Team > Merge.`.)