Guide utilisation git

Installation :

Afin de pouvoir télécharger le logiciel de gestion de version Git sur Windows, rendez-vous sur la page suivante : https://git-scm.com/ et cliquez sur Download.

Pour procéder à l'installation, cliquez sur le fichier exécutable que vous venez de télécharger et suivez la procédure d'installation en conservant les paramètres recommandés. Toutefois, dans la section « **Adjusting your PATH environment** », vous pouvez sélectionner « **Use Git from Git Bash only** » car de toute façon vous n'aurez certainement pas besoin de lancer Git depuis un invité de commande Windows car il est préférable d'utiliser les commandes Git depuis l'invité de commande spécifique à Git, à savoir **Git Bash**.

Une fois l'installation terminé, décocher la deuxième case pour que Git n'ouvre pas les Releases Notes.

Ensuite, il faut procéder à une petite étape de configuration où il est nécessaire de renseigner un nom d'utilisateur ainsi qu'une adresse mail à Git. Vous pouvez renseigner globalement ce que vous voulez, pas besoin que ce soit en raccord avec BitBucket. Ouvrez l'invite de commande Git Bash et rentrez les commandes suivantes :

- **git config --global user.name "{username}"**
- **git config --global user.email "{email}"**

Ajouter une clé SSH sur BitBucket :

Au moment de soumettre des changements sur BitBucket (au travers de la commande « git push »), Git vous demandera peut-être de renseigner vos identifiants de connexions BitBucket à chaque fois que vous aurez envie de push.

Pour éviter cela, il faut créer une clé SSH qui permet une connexion sécurisée avec BitBucket au travers d'une technique de chiffrement asymétrique, ce qui impliquera que vous n'aurez plus besoin de renseigner vos identifiants.

1. Ouvrez Git Bash et tapez la commande suivante : **ssh-keygen** 2.
   Appuyez sur Entrer pour accepter le chemin par défaut à savoir :
   /c/Users/<username>/.ssh/id_rsa
3. Appuyez sur Entrer deux fois de plus pour valider une passphrase a null .

Dans le dossier **/c/Users/<username>/.ssh/** vous devriez retrouver deux fichiers : **id_rsa** qui correspond à la clé privé et **id_rsa.pub** qui correspond à la clé publique.

Il faut maintenant ajouter la clé publique sur son compte BitBucket :

1. Cliquez sur **Personal Settings** depuis son avatar
2. Cliquez sur **SSH Keys**
3. Ouvrez le fichier **id_rsa.pub** et **copiez** son contenu
4. Depuis BitBucket, cliquez sur **Add key**
5. Entrer un **Label**, n'importe lequel

6. **Collez** le contenu de la clé publique dans le champ **Key**
7. Sauvegarder
8. Pour tester si ça marche : tapez la commande **ssh -T git@bitbucket.org** depuis Git Bash

Récupérer le projet depuis Bitbucket :

Attention : Pour cette étape, il y a peut-être besoin d'être connecté sur BitBucket au préalable.

Dans un premier temps, ouvrez Git Bash depuis le dossier dans le lequel vous souhaitez cloner le projet. Pour cela, ouvrez l'explorateur de fichier et rendez-vous sur le dossier désiré et faite **Click Droit -> Git Bash Here**.

Ensuite, tapez la commande suivante :
git clone git@bitbucket.org:elmitsuki/ciad-lab-spring-boot.git
(Remplacez cette adresse par celle de votre répertoire git)
Toutefois, pour pouvoir cloner le projet avec la commande citée précédemment, il est nécessaire d'avoir réalisé l'étape précédente concernant l'ajout d'une clé SSH sur son compte BitBucket car ce clonage correspond à un clonage en **SSH**. Vous pouvez aussi cloner en **HTTPS** avec la commande suivante : **git clone https://ElMitsuki@bitbucket.org/elmitsuki/ciad-lab-spring-boot.git(Remplacez cette adresse par celle de votre répertoire git)** mais dans ce cas-là il faudra renseigner ses identifiants BitBucket et il se peut que ça ne marche pas.

Commandes de bases de Git

git clone : permet de de cloner un projet **git branch** : permet de créer une nouvelle branche ->

git branch <nom_de_la_branche> git checkout : permet de changer de branche -> **git**

checkout <nom_de_la_branche > git add : permet d'ajouter un ou plusieurs fichiers dans la

pré-sauvegarde -> **git add index.html**

git commit : permet d'ajouter un ou plusieurs fichiers dans la sauvegarde -> le paramètre **-m** permet de préciser un message au commit pour donner un message concernant le changement -> **git commit -m "ajout de la table liste_conference"**

git push : permet de pousser les changements effectuer vers le repository git qui se trouve sur

BitBucket dans notre cas **git status** : permet de connaitre l'état des fichiers **git pull** : permet

de mettre à jour votre branche en local avec le repository

Une fois qu'on a terminé d'effectuer une tâche, la série d'instructions à réaliser pour pousser les changements vers BitBucket est la suivante :

1. Ouvrir Git Bash depuis le dossier root du projet à savoir **ciad-lab-spring-boot**
2. git add . (le point indique qu'on sélectionne tous les fichiers modifiés)
3. git commit -m "<message>"
4. git push

Utiliser Git depuis Eclipse :

Depuis l'IDE cliquez sur le logo **Open Perspective** en haut à droite puis cliquez sur **Git** et enfin **Open**.

Vous aurez alors accès à la vue Git qui vous permettra d'effectuer certaines commandes directement depuis eclipse sans avoir à les rentrer à la main.

Depuis le champ « Unstaged Changes » vous pouvez voir tous les fichiers que vous avec modifiés par rapport aux repository et vous pouvez cliquer sur les deux boutons + verts pour faire l'équivalent d'un git add.

Depuis le champ « Commit Message » vous pouvez saisir le message de votre commit et ensuite vous pouvez appuyer sur le bouton « Push HEAD » pour faire l'équivant du git commit suivi du git push.

A partir de la vue à gauche de l'écran vous pouvez également voir la liste des branches du projet en local sur votre PC en cliquant sur **Branches -> Local**. A partir de là vous pouvez choisir de changer de branche ce qui revient à être l'équivalent d'un git checkout.

Procédure de développement d'une nouvelle tâche :

Pour commencer une nouvelle tâche, il faut tout d'abord créer une nouvelle branche à partir du master.

Vous pouvez créer la nouvelle branche depuis BitBucket en vous rendant dans la rubrique **Branches** à gauche du projet.

Ensuite, cliquez sur **Create Branch**. Choisissez le Type Feature si la tâche correspond à une nouvelle fonctionnalité (ce sera le plus souvent ça qu'on veut) ou bien sur Bugfix si la tâche correspond à une correction de bug.

Il faut bien avoir sélectionné la branche **master** comme branche source car on crée toujours une branche à partir du master. Il faut que le master soit toujours le plus à jour possible.

Donner un nom à la branche qui correspond à la tâche que vous voulez réaliser. Ne mettez pas d'espaces ou d'accents car ils seront ignorés par BitBucket de toute façon.

Ensuite, il faut faire un git checkout pour avoir accès à la nouvelle branche créée en local sur son pc. Pour cela faite un **git checkout <nouvelle_branche>** depuis le Git Bash ou bien depuis Eclipse à partir de la vue Git en allant sur **Branches / Remote Tracking** et en faisant **Clic droit** sur la nouvelle branche puis **Check Out…**

Une fois la tâche terminée vous pouvez faire toute la procédure pour pousser vos modifications sur BitBucket.

Procédure pour les pulls requests pas encore définie.

Utilisation d'un login personnel

Guide supplémentaire concernant l'utilisation du fichier application.yml. Il permettra d'utiliser son propre fichier application.yml en dehors du projet global et ainsi éviter de devoir changer le mot de passe de la base de données à chaque fois qu'on commencera une nouvelle tâche (en effet, tout le monde n'aura pas le même mot de passe de connexion).

Une fois que vous avez cloné le projet, la première chose à faire et donc de paramétrer l'application pour avoir son propre fichier application.yml.

1. Copier le fichier application.yml du projet et collez le dans un dossier en dehors du projet git pour qu'il ne soit pas pris en compte au moment de push des modifications. Par exemple vous pouvez créer un dossier **/c/Users/<username>/Documents/SpringBootProperties** et collez le fichier application.yml dedans.
2. Ouvrez le fichier application.yml à l'aide d'un éditeur de texte et renseigner le mot de passe de votre base de données (si vous avez bien suivi le tuto d'Antoine, votre base de données s'appelle bien **librarymanager** et le username par défaut devrait être **root** donc vous n'avez pas besoin de modifier ces informations dans le fichier).
3. Maintenant, il faut préciser au démarrage du projet que Spring doit chercher le fichier de configuration à l'endroit où vous venez de le ranger. Pour cela, il faut ajouter un argument à la commande de démarrage qui permet de spécifier ce chemin. Etant donné que nous utilisons Eclipse pour lancer le projet, on peut spécifier les arguments directement depuis Eclipse. Pour cela allez dans **Run -> Run Configurations**. Dans le menu à gauche, vous devez vous assurez que le fichier sélectionné est bien **Java Application -> PubProviderApplication**. Cliquez sur la section **Arguments** puis dans **Program arguments** saisissez l'argument suivant : **--spring.config.location=<chemin_du_dossier>/application.yml**. Le chemin spécifié dépend évidemment du dossier dans lequel vous avez rangé votre fichier application.yml. Cliquez ensuite sur **Apply** puis vous pouvez **Close** la fenêtre.
   Néanmoins, si vous utilisez un autre IDE qu'Eclipse, vous devez chercher par vous-même comment on ajoute des arguments au démarrage.