Git - Séance 2

Le travail en équipe

# Avant-propos

* Afin d’illustrer les problématiques du travail en équipe, ce TP sera réalisé en binôme. Une partie des actions sera réalisée par un élève, une autre partie par l’autre, et enfin certaines actions devront être effectuées par les deux.
* Avant de débuter, remplacez tous les termes “eleve\_1” par le nom en minuscule et sans espace ou lettre spéciale de l’un de vous (par exemple, dans mon cas ce serait “barbier”) et “eleve\_2” par le nom de l'autre.
* Les actions décrites ci-dessous devront être réalisées de préférence sur <https://gitlab.com/>, afin de bénéficier de certaines fonctionnalités intéressantes proposées. En cas d’inaccessibilité du serveur, elles pourront être adaptées à [https://github.com/](https://github.com/.).

# Prérequis

Si vous ne possédez pas de compte Gitlab (ou Github, le cas échéant), commencez chacun par créer un compte Gitlab en utilisant votre adresse mail Ynov.

Sous Windows, supprimer tous les mots de passe Gitlab enregistrés dans le “Gestionnaire d’identification” (ou le cas échéant les mots de passe Github).

Rappel

*Toutes les lignes commençant par $ sont à lancer dans l'interpréteur de lignes de commande “bash”.*

# Partie 1 – Manipuler un dépôt distant

*Cette partie sera réalisée sur le poste de eleve\_1 uniquement.*

## Créer un dépôt distant

1. Créer un nouveau projet :
   * Nommé **seance2-*eleve\_1*** ;
   * **Privé ;**
   * Initialisé avec un **fichier Readme ;**
2. Changer la protection de la branche “master” :
   * Dans “Settings / Repository / Protected branches”, pour “**Allowed to push**” et “**Allowed to merge**”, choisir “**Developers + Mainteners**” pour la branche “master” ;

## Récupérer le dépôt distant sur sa machine

Tester la récupération d’un dépôt distant via les deux méthodes disponibles :

Première méthode : via SSH

1. Créer un répertoire “GitSeance2” ;
2. Se placer dans ce répertoire ;
3. Créer une clé SSH et l’associer au dépôt distant en suivant les instructions décrites ici : <https://gitlab.com/help/ssh/README#generating-a-new-ssh-key-pair> (privilégier la création d’une clé ED25519) ;
4. Lancer la commande *$ git clone url\_ssh\_fournie\_par\_gitlab seance2-eleve\_1-ssh* ;
5. Se placer dans le répertoire créé ;
6. Lancer la commande *$ git status* pour vérifier l’état du dépôt.

Question 1 : Quel est le résultat de la commande ?

Deuxième méthode : via HTTPS

1. Se placer dans le répertoire “GitSeance2” ;
2. Lancer la commande *$ git clone url\_https\_fournie\_par\_gitlab seance2-eleve\_1-https* et renseigner les informations d’identification ;
3. Se placer dans le répertoire créé ;
4. Lancer la commande *$ git status* pour vérifier l’état du dépôt local.

Question 2 : Quel est le résultat de la commande ? Est-il différent de celui de la question 1 ?

*Pour la suite du TP, vous pourrez utiliser au choix l’un des 2 répertoires de dépôt local créés.*

## Envoyer une validation sur le serveur distant

1. Dans le dépôt local choisi, créer un fichier “File-1.txt” ;
2. Ajouter le fichier à l’index Git ;
3. Valider la modification en utilisant un message expliquant l’action réalisée (par exemple “Add File-1.txt file”) ;
4. Lancer les commandes *$ git status* et *$ git log* pour vérifier l’état du dépôt local et son historique ;

Question 3 : Que signifie le résultat “Your branch is ahead of 'origin/master' by 1 commit” que vous avez dû obtenir ?

1. Suivre l’instruction proposée pour envoyer les validations locales sur le serveur distant.

Question 4 : Quelle instruction avez-vous utilisée ?

Question 5 : A présent, quel est l’état du dépôt local ?

# Partie 2 – Manipuler un dépôt distant en équipe

*Cette partie sera réalisée sur le poste de eleve\_1 ou eleve\_2 suivant les besoins.*

## Ajouter un développeur au projet

1. Dans Gitlab, dans la rubrique “Settings / Members” du projet **seance2-*eleve\_1***, inviter *eleve\_2* en lui octroyant les droits de **développeur**.
2. Sur le poste de *eleve\_2*, utiliser l’une des 2 méthodes décrites plus haut (https ou ssh) pour récupérer le dépôt distant de *eleve\_1* (ne pas préciser de répertoire de destination afin d’utiliser le répertoire par défaut) ;

Question 6 : Quelle est la commande à utiliser ?

## Manipuler le dépôt distant de **son binôme**

*Les opérations suivantes sont à réaliser sur le* ***poste de eleve\_2****.*

1. Se placer dans le répertoire du dépôt local et créer un nouveau fichier appelé “File-2.txt” ;
2. L’ajouter à l’index et le valider localement ;
3. Le pousser sur le dépôt distant de *eleve\_1* ;
4. Lancer la commande *$ git log* pour voir l'historique du dépôt local ;

Question 7 : Combien y a-t-il de validations sur ce dépôt ? Combien d’utilisateurs différents peut-on recenser ?

1. Sur Gitlab, visualiser le graphe du dépôt distant (“Repository / Graph”).

Question 8 : Combien y a-t-il de validations sur ce dépôt ? Combien d’utilisateurs différents peut-on recenser ?

## Récupérer les validations effectuées par un tiers sur **son dépôt distant**

*Les opérations suivantes sont à réaliser sur le* ***poste de eleve\_1****.*

1. Lancer la commande *$ git log* ;

Question 9 : Combien y a-t-il de validations sur ce dépôt ? Pourquoi est-ce moins qu’aux questions 7 et 8 ?

1. Lancer la commande *$ git log --remotes* ;

Question 10 : Combien y a-t-il de validations recensées ?

1. Lancer la commande *$ git fetch* ;
2. Relancer les 2 commandes des questions précédentes ;

Question 11 : Y a-t-il une évolution sur l’historique des validations dans l’un ou l’autre cas ou les deux ? Pourquoi ?

1. Lancer la commande *$ git pull* ;
2. Lancer à nouveau les 2 commandes des questions précédentes ;

Question 12 : Quel est le résultat à présent ? Pourquoi ?

## Effectuer une fusion de modifications

*Les opérations suivantes sont à réaliser sur les postes des 2 élèves.*

1. Chacun de son côté :
   1. Créer un fichier portant un nom différent (en le suffixant du nom de famille du créateur par exemple) ;
   2. Valider le fichier créé sur le dépôt local ;
2. L’un après l’autre, envoyer les modifications sur le serveur distant via la commande *$ git push* ;

Question 13 : Que se passe-t-il lors de la seconde exécution de la commande push ? Pourquoi ?

1. Sur le dépôt local n’ayant pu envoyer les modifications sur le distant :
   1. Exécuter la commande *$ git pull* ;

Question 14 : Que se passe-t-il ?

* 1. Accepter le message de fusion proposé en fermant l’éditeur de texte ;
  2. Ré-exécuter la commande *$ git push* ;

Question 15 : Que se passe-t-il à présent ?

1. Sur Gitlab, visualiser le graphe du dépôt distant (“Repository / Graph”)

Question 16 : Qu’y a-t-il de différent par rapport à l’affichage effectué dans les étapes précédentes ?

# Partie 2 – Utiliser des branches

*Cette partie sera réalisée sur le poste de eleve\_1 ou eleve\_2 suivant les besoins.*

## Créer une branche

1. Sur le poste de eleve\_1 et eleve\_2, lancer la commande *$ git pull* afin d’avoir un dépôt local identique au dépôt distant ;
2. Sur le poste de eleve\_1, lancer la commande *$ git branch test-branching-eleve\_1* ; puis la commande *$ git checkout test-branching-eleve\_1* ;
3. Sur le poste de eleve\_2, lancer la commande *$ git branch test-branching-eleve\_2* ; puis la commande *$ git checkout test-branching-eleve\_2* ;

Question 17 : Après l’appel à la commande "git checkout …" quel changement voit-on apparaitre dans l’interface bash ? Que cela signifie-t-il ?

## Travailler sur une branche

1. Chacun sur son poste, créer un nouveau fichier nommé différemment ;
2. L’ajouter au dépôt local ;
3. Lancer les commandes *$ git log -n 3* et *$ git log -n 3 master* afin d’afficher l’historique de la branche courant et de la branche master ;

Question 18 : Quelles différences observe-t-on entre les 2 appels ?

## Envoyer une branche nouvelle sur le dépôt distant

1. Sur Gitlab, visualiser le graphe du dépôt distant (“Repository / Graph”) et constater qu’aucune référence aux branches créées localement n’existe ;
2. Sur le poste de eleve\_1, lancer la commande *$ git push --set-upstream origin test-branching-eleve\_1* ;
3. Sur le poste de eleve\_2, lancer la commande *$ git push --set-upstream origin test-branching-eleve\_2* ;
4. Sur Gitlab, visualiser le graphe du dépôt distant (“Repository / Graph”) ;

Question 19 : Qu’est-il apparu ?

## Fusionner une branche dans une autre

1. Sur le poste de eleve\_1, lancer les commandes suivantes :
   1. *$ git checkout master* ;
   2. *$ git merge test-branching-eleve\_1* ;
   3. *$ git push* ;
2. Sur Gitlab, visualiser le graphe du dépôt distant (“Repository / Graph”) ;

Question 20 : Que s'est-il passé sur la branche master ?

Question 21 : Si l’on souhaite récupérer les modifications de master sur la branche test-branching-eleve\_2, comment doit-on s’y prendre ?

## Supprimer une branche

1. Sur le poste de eleve\_1, s’assurer que l’on est sur la branche master ;
2. Lancer les commandes suivantes :
   1. *$ git branch -d test-branching-eleve\_1* ;

Question 22 : A quoi sert cette commande ?

* 1. *$ git push origin --delete test-branching-eleve\_1* ;

Question 23 : A quoi sert cette commande ?