Objectifs

L'objectif de cet atelier est d'apprendre à utiliser GIT pour versionner ses projet et pour travailler à plusieurs.
Vous allez commencer par apprendre à travailler sur un dépôt local (clone d'un dépôt partagé), puis vous allez travailler sur un dépôt partagé.
Pour cet atelier, il est préférable de travailler dans un nouveau *projet*.

GIT

Pour revoir rapidement le fonctionnement de git :
Try GIT
https://try.github.io/levels/1/challenges/1

Le livre suivant, en français, décrit comment utiliser GIT.
http://git-scm.com/book/fr/v1

Choisir un hébergeur GIT

Commencez par créer un compte sur un hébergeur GIT si vous n'en avez pas encore. Vous pouvez par exemple créer un tel compte sur le GITHUB, BITBUCKET ou GITLAB.

Voici les liens pour ces différentes plateformes :

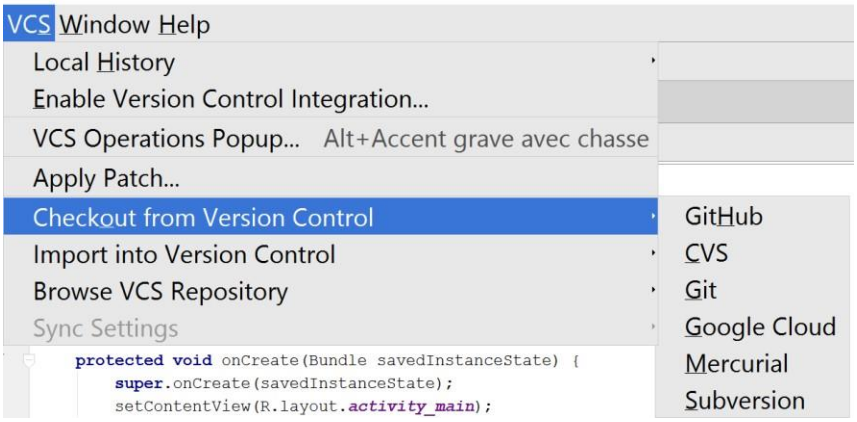
- https://about.gitlab.com/
- https://github.com/
- https://bitbucket.org/
- Il faudra installer GitBash https://gitforwindows.org/
- Il faut utiliser une clé ssh : ssh-keygen
    - Mettre la clé publique dans le GITLab ou autre
    - Indiquer la clé privée à AndroidStudio

Créer un projet Android

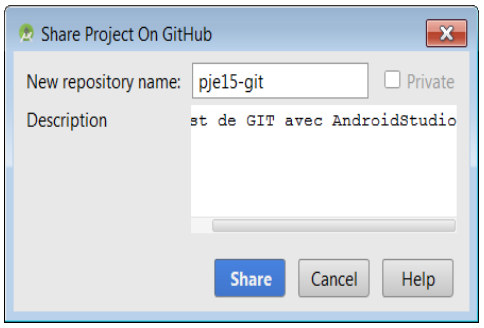
Créez un projet Android avec une "Blank Activity".
Vous devez maintenant avoir un nouveau projet contenant des fichiers.

Mettre son projet sous GIT

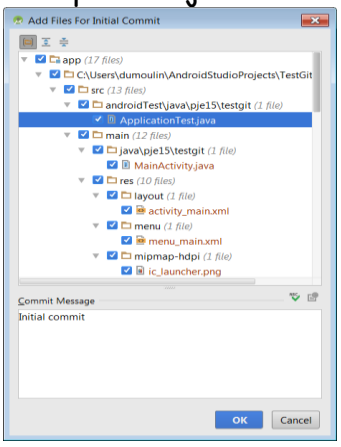
Vous allez maintenant mettre votre projet sous GIT.



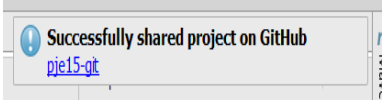
VCS -> Import into Version Control -> Share project on Git (ou Github)
Entrez votre login et password github (si on vous le demande) ou utilisez la clé ssh (GITLab)



Acceptez l'ajout de tout les fichiers pour le 1er commit



Vous devriez avoir le message suivant en haut à droite



Vérifiez sur votre compte gitlab ou github. Vous devriez voir votre projet qui est maintenant sous GIT.

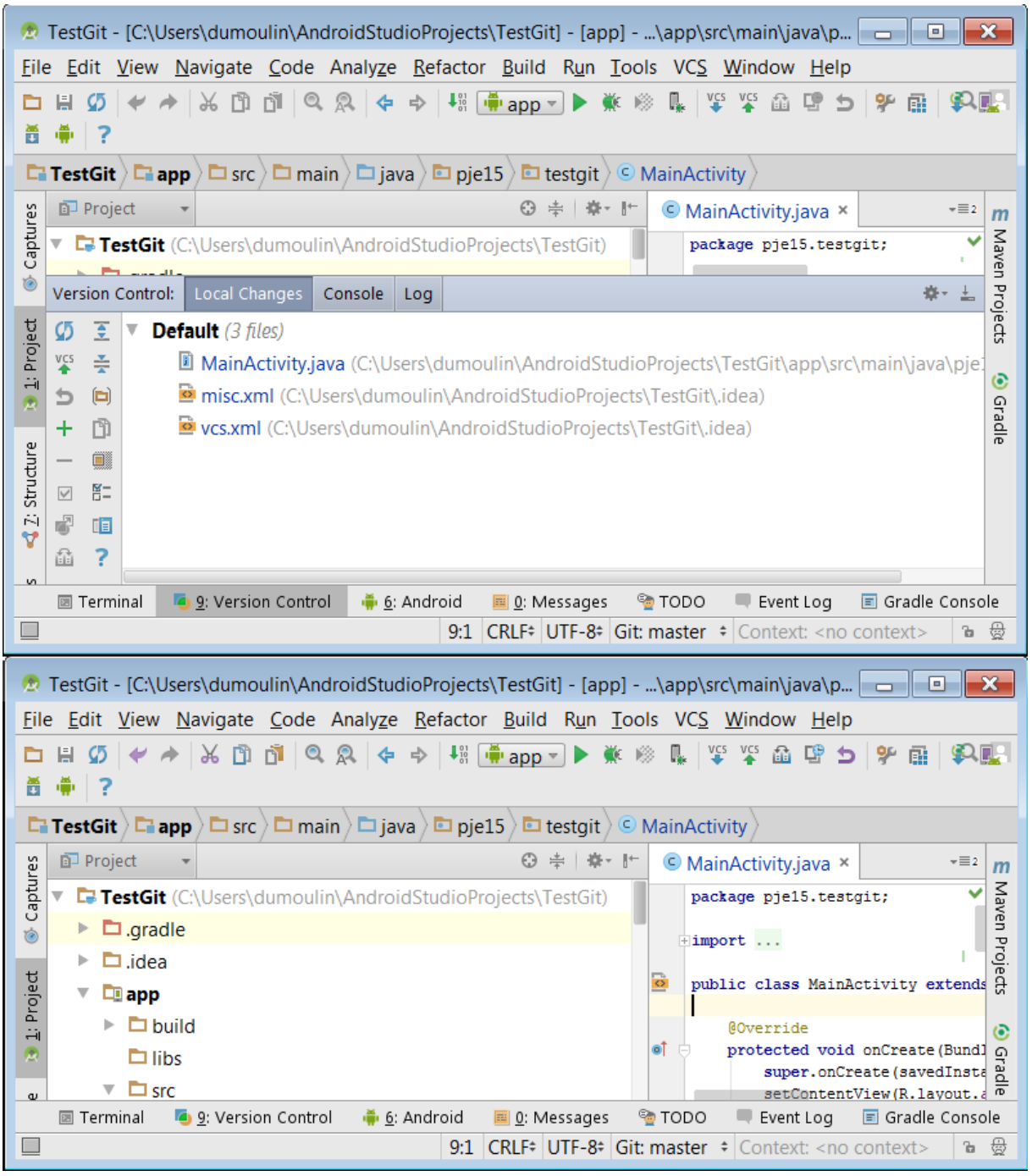
Voir les fichiers modifiés

AndroidStudio utilise un code de couleur pour signaler l'état des fichiers (modifié, ajouté, en conflit ...). La page suivante explique ce code : https://www.jetbrains.com/idea/help/file-status-highlights.html

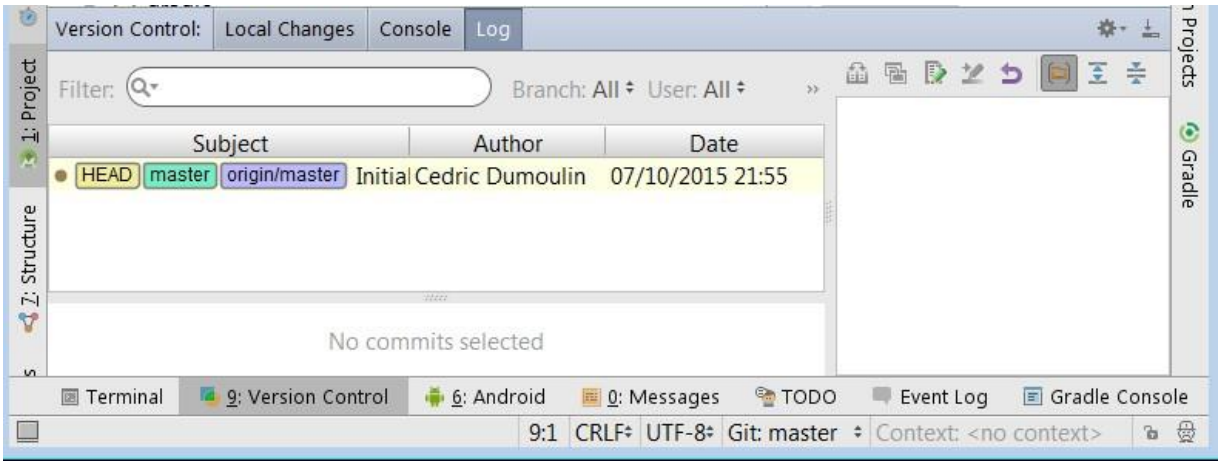
Modifiez un des fichiers Java, par exemple en ajoutant un espace et en le sauvant. Vérifiez son nouveau statut.

Ouvrir la fenêtre VCS

AndroidStudio propose une fenêtre dédié à GIT.
Cliquer sur l'onglet 'Version Control' en bas de l'éditeur.
Vous devriez voir la liste des fichiers modifié.



Vous pouvez aussi voir l'historique des commit, et les branches, en selectionnant l'onglet Log :



Essayer GIT

Vous allez vous exercer à utiliser les commandes de GIT dans AndroidStudio ou GitBash. Pour cela, vous allez créer et modifier des classes Java dans votre projet en suivant les indications.

Travailler en local

Exercez-vous à travailler en local. Basculez entre les vues "Local Changes" et "Log" pour vérifier les résultats.
Vous travaillez en local : les commit se font sur votre dépôt local. Ils ne seront pas encore propagés sur github.

Premier commit

- Créez une classe C1,
- Commit
  - Dans la vue "Commit Changes", n'oubliez pas de spécifier l'auteur ET le message
- Vérifiez l'historique

Second Commit

- Créez une classe C2,
- Commit
- Vérifiez l'historique

Créer une Branche

- Créez une branche 'br1'
  - Voir Comment avoir le menu pour les branches
- Créez une classe C3,
- Commit
- Vérifiez l'historique, regardez bien les étiquette !

- Créez une classe C4,
- Commit
- Vérifiez l'historique.

Vous avez maintenant une branche

Premier merge : Fast-Forward

Vous allez merger votre branche br1 dans master

- Changer de branche, aller sur *master*
  - Voir Comment changer de branche
- demander le merge de la branche br1
  - Demander le menu des branches
  - "br1 -> checkout"
  - Comme l'historique est linéaire, GIT fait un fast-forward. Il ne faite que avancer les étiquetes locales.
- Vérifiez l'historique, regardez bien les étiquette ! Où est br1 ? master ? origin/master ?

Second merge : vraie merge

- Basculer sur br1
- Créez une classe C5,
- Commit
- basculer sur master
- Créez une classe C6,
- Commit
- Vérifiez l'historique,
  - regardez bien les étiquette. Vous devez voir maintenant deux branches distinctes.
  -
- demander le merge de la branche br1
  - Le merge s'effectue sur deux branches distinctes. GIT crée un nouveau noeud pour merger les deux branches
- Vérifiez l'historique afin de visualiser le nouveau noeud.

Travailler à plusieurs

Jusqu'ici, vous avez travaillé avec votre dépôt local. C'est ce que vous devez faire quand vous développez une nouvelle fonctionnalité. Une fois la fonctionnalité finalisée, il faut la partager dans le dépôt commun.

Créer un second clone

Pour simuler le partage, il faut créer un second clone de votre dépôt distant.
Pour cela, ouvrez un autre AndroidStudio. Vous pouvez le faire sur votre machine, ou sur le compte de votre binôme.
Voir Cloner le dépôt.

Par la suite, nous distinguerons les deux espaces de travail AndroidStudio par **'AndroidStudio1'** et **'AndroidStudio2'**.
Dans **AndroidStudio2**, vous devriez voir le projet de **AndroidStudio1**, ainsi que son historique.
A ce niveau, les espaces de travail de **'AndroidStudio1'** et **'AndroidStudio2'** sont différents. Si ce n'est pas le cas, c'est que vous avez fait des 'pull' à la place de 'commit'. Dans ce cas, créez une nouvelle classe dans **AndroidStudio1** afin d'avoir un espace de travail différent.

Pousser les modifications

Pour partager les modifications, il faut les 'pousser' (push) dans le dépôt distant.
Dans votre historique, vous avez une étiquette 'origin/master' et une étiquette 'master'. Ces étiquettes référencent le dernier commit de la branche correspondante. La première étiquette 'origin/master' correspond au commit de 'master' dans le dépôt distant. La seconde correspond au dernier commit que vous avez fait dans la branche locale 'master'.
Le but de push est de pousser les commits que vous avez fait dans la branche locale master vers la branche distante master, puis de faire avancer l'étiquette origin/master.
Comment pousser les commit :

- Dans AndroidStudio1
- sélectionnez un fichier du projet
- git -> repositories -> push
- Vérifiez l'historique;
    - ou est l'étiquette 'origin/master' ? master ?

Récupérer (tirer) les modifications

Pour récupérer les modifications du dépôts distant, il faut les tirer (pull) dans votre dépôt local.
GIT fait le pull se fait en deux étapes :

- un fetch - qui rapatrie dans votre dépot local tous les commits distant que vous n'avez pas encore.
    - Dans cette étape, origin/master peut etre modifié par GIT si des commit distants existent dans cette branche.

- un merge - qui merge la branche origin/master avec votre branche master locale
    - GIT fera un fast-forward si possible
    - Sinon, il créera un nouveau noeud merge

Ces deux étapes peuvent se faire séparément, mais GIT offre une commande faisant les deux à la fois : 'pull'.

- Allez dans AndroidStudio2,
- VCS -> Git ->pull
    - Cela doit synchroniser votre dépot local et fusionner votre branche master.
- Vérifiez l'historique;
    - ou est l'étiquette 'origin/master' ? master ?

Gérer les conflits

La plupart du temps, le merge se passes sans probleme. GIT arrive a fusionner les modifications de chacun.
La fusion se fait par fichier, ligne à ligne: GIT remplace les lignes modifiés par la nouvelle ligne.
Les conflits arrivent si vous modifiez des lignes qui ont été modifié par quelqu'un d'autre depuis votre dernière synchronisation avec le dépôt distant. Dans ce cas, GIT interrompt le merge, et vous demande de régler le conflit.
GIT indique le conflit dans le fichier concerné à l'aide de chevron
'<<<<<<<<<<<<
et
>>>>>>>>>>>>'.
Vous devez résoudre le conflit, enlever les chevrons, et finir le processus de merge en plaçant le fichier résolu dans la *stagged area*, puis en faisant un commit.

- Dans AndroidStudio1 et AndroidStudio2, modifiez la même ligne du même fichier.
- Allez dans AndroidStudio1
- pousser vos modifications
- Allez dans AndroidStudio2
- pousser vos modifications
    - AndroidStudio refuse, il vous propose de faire un 'merge' auparavant
- faite le merge
    - Le merge s'interrompt
    - AndroidStudio signale un conflit

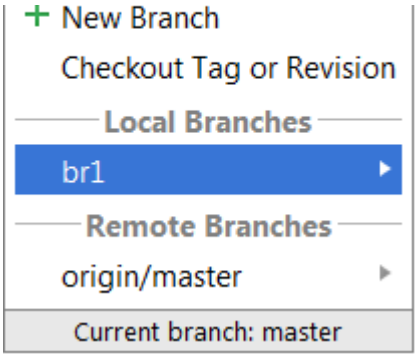
    - il propose de prendre les modifs du dépôt OU les vôtres OU de régler vous même le conflit
    - réglez le conflit
    - regardez l'historique, et la position de la branche 'master' et 'remote/master'. Elle ne sont pas identique, ce qui indique que vos modifications n'ont pas été propagée sur le dépôt distant.
- Vous avez maintenant une version locale contenant vos modifications et les modifications du serveur
- Faites un push afin pousser vos modifications sur le serveur.

- Allez dans AndroidStudio1
- Tirez les modifications du serveur
    - Vous devez voir les modifications effectué dans AndroidStudio2

How To

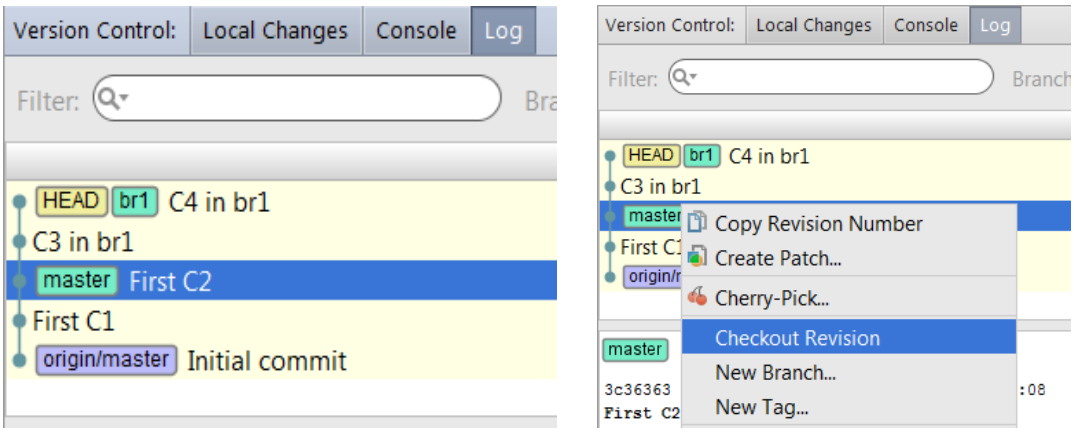
Comment avoir le menu pour les branches

https://www.jetbrains.com/idea/help/accessing-git-branches-popup-menu.html

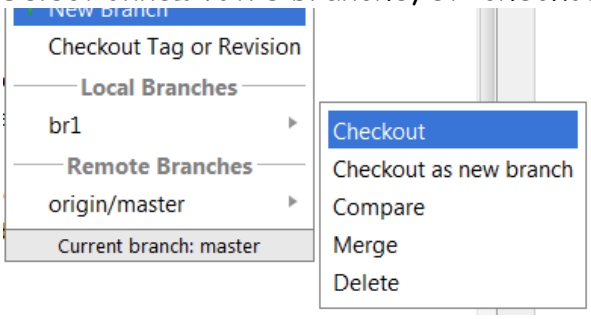
- Sélectionner un fichier de votre projet
- "clique droit -> Git -> Repositories -> Branches ..."
- Vous obtenez le menu "branches"
- 

Comment changer de branche

- aller dans la vue "Version Control -> Log"
- Sélectionner le commit correspondant à la branche que vous voulez
- clique droit -> checkout revision
-

- 

Ou

- Sélectionner le menu branches
- Selectionnez votre branche, et "checkout"
- 

Cloner un dépôt

A partir d'un AndroidStudio ouvert :

- File -> New -> Project from Version Control -> github
- Choisir d'ouvrir dans une nouvelle fenetre
- Si un message indiquant que votre projet n'est pas sous git aparait
  - VCS -> import into version control -> Share project on github