Oefening Git en Gitlab

Deze oefeningen hebben als doel u een beetje vertrouwd te maken met GIT als Version Control System.

De eerste 3 oefeningen gebruiken de CLI – Command Line Interface ofwel GIT-Bash dat u op uw computer installeerde. In de realiteit zal u vermoedelijk meer met IntelliJ of Android Studio werken voor uw GIT interactie al moet u beseffen dat deze IDE achterliggend precies dezelfde commando's gebruiken. De laatste 2 oefeningen gebruiken Android Studio.

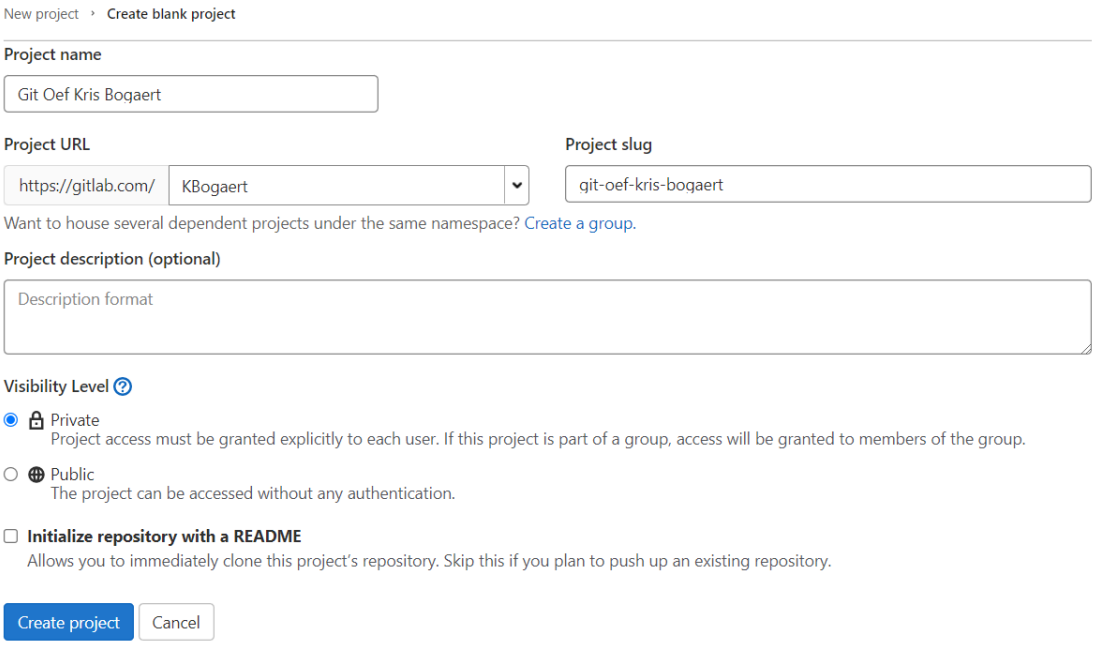
En passant leert u ook nog wat Linux commando's mocht u daar niet mee vertrouwd zijn.

Nota: Volg elke stap nauwgezet en ga niet verder als een stap niet lukt want meestal wordt het alleen erger. Soms is het beter enkele stappen terug te gaan en opnieuw te beginnen of gelijk een nieuw GIT project aan te maken.

Oefening 1 – A team of 2

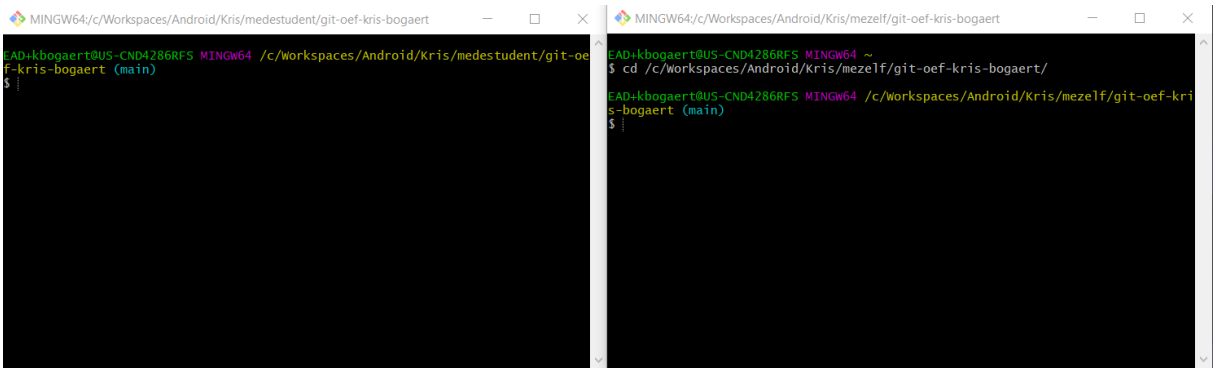
Hier gaan we een 'team' van 2 personen simuleren: 'mezelf' en 'medestudent'. Je gaat namelijk eenzelfde repository 2 keer clonen in verschillende folders

1. Log in op gitlab.com en maak een nieuw blanco project aan: Git Oef <uw naam>.
   Bijvoorbeeld:



2. Na de create navigeer linksboven naar Project Information / Members
   Vul in:
   GitLab member or Email address: **Jorn.Jossa@ucll.be** en **Nadir.Aboulkassimi@ucll.be**
   Choose a role permission: **Maintainer**
   Klik Invite
   Nu heb ik toegang tot jouw project en kan ik het verbeteren.

3. Zoals in de les, open de GIT-Bash Command Line Editor. Tik het commando: cd om naar je 'Home' directory te gaan. Dat wordt aangegeven met een ~
4. Ga in de folder UCLL-Projects die we in de les hebben aangemaakt of indien nog niet aanwezig maak die folder met het commando: mkdir UCLL-Projects
   Navigeer naar die folder (cd = Change Directory): cd UCLL-Projects
5. Hierbinnen maak 2 nieuw folders (naast elkaar) met als namen: **mezelf** en **medestudent**
   Je kan dit met Windows Explorer doen of met het mkdir commando.
6. Kloon de gitoef repo 2 keer:
   cd mezelf
   Kloon hier het project dat je hebt aangemaakt in stap1. Zie slides voor juiste commando.
   cd ../medestudent
   Kloon hier opnieuw hetzelfde project dat je hebt aangemaakt in stap1
   cd git-oef-<uw naam>
7. Open een 2e GIT-Bash commando en navigeer naar het 'mezelf' project en daarbinnen je gitoef_... folder. Zet de 2 'terminals' naast elkaar:



8. In de terminal met de 'mezelf' repo. Maak een nieuw bestand aan genaamd: mijn_bestand.txt
   Je kan dit makkelijk doen met het Linux commando: touch mijn_bestand.txt

   Zoals in de les gedaan. Doe een git add, commit (inclusief de verplicht commit message) en push om mijn_bestand.txt op de Remote server (gitlab.com) te krijgen. Controleer in de browser of het bestand is aangekomen op gitlab.com

9. Ga nu naar de andere terminal (met de 'medestudent' repo) en doe een git pull
   Valideer of er een bestand werd toegevoegd met het (Windows) commando dir of met het (Mac/Linux) commando ls -l

10. Nog steeds in de 'medestudent' terminal, wijzig het bestand.
    Je kan er via Windows Explorer naartoe navigeren
    Of je tikt: notepad mijn_bestand.txt
    Of je tikt: nano mijn_bestand.txt
    Of, voor de echte die-hards, je tikt: vi mijn_bestand.txt

11. Save, (Add), Commit en push je bestand

12. Terug in de ‘mezelf’ editor. Pull de wijzigingen en valideer de inhoud van het bestand bv. met het commando: cat mijn_bestand.txt

Oefening 2 – Merge conflicts

Met nog steeds de 2 terminalen open uit oefening 1

1. In de terminal met de ‘mezelf’ repo. Wijzig nog een keer het bestand mijn_bestand.txt. Commit met de boodschap ‘wijziging van mezelf’ en push
2. In de terminal met de medestudent repo.
   Wijzig ook hier het bestand mijn_bestand.txt maar doe het een beetje anders dan hierboven
   Commit met als boodschap ‘wijziging van medestudent’
   Push => Gaat dit zomaar lukken?
   Nope! Lees de hint:
   Updates were rejected because the remote contains work that you do not have locally. This is usually caused by another repository pushing to the same ref. You may want to first integrate the remote changes (e.g., 'git pull ...') before pushing again.
3. De git push is dus niet gelukt. Doe nu eerst een git pull
   In GIT Bash staat nu op het einde **MERGING**
4. Open het bestand mijn_bestand.txt in je editor van voorkeur
   Je zal normaal gezien ergens volgende structuur tegenkomen
   <<<<<<< HEAD
   (Hier staat uw ‘medestudent’ wijziging)
   =======
   (Hier staat de wijziging die je van ‘mezelf’ hebt binnengetrokken)
   >>>>>>>

   Corrigeer nu het bestand mijn_bestand.txt zodat beide wijzigingen erin blijven en verwijder al die <<<<<<<, >>>>>>>> en =======. Het resultaat is dus het gemergde bestand waarin zowel uw als de andere zijn wijzigingen een plaats kregen en dat er weer goed uitziet.

5. Nu moet je (beetje raar misschien maar het is zo) opnieuw een **git add** doen van je bestand gevolgd door een **git commit -m “Merge conflictje opgelost”** en een **push**

6. In de andere terminal doe je een git pull.
   Wat verwacht je? Gaat dit werken?

7. (Als het allemaal niet lukt kan je met **git merge --abort** altijd eventueel eens opnieuw beginnen)

8. Ook in de browser, gitlab.com, selecteer nog eens het bestand om de laatste versie te zien. Klik eens op de ‘History’ knop rechtsboven om alle versies en commit messages te zien die je intikte.

Oefening 3 – Commit messages

In eender welke terminal

1. Begin met een git pull zodat je zeker de laatste wijziging hebt
2. Breng een wijziging aan in het bestand
3. Commit eens zonder een message dus zonder de -m param
4. Er opent een soort inline editor waar je uw commit message kan intikken. Je kan die message opslaan en die editor verlaten door volgende 4 toetsen na elkaar in te drukken: ESC : w q
5. Optioneel: met het commando **git commit --amend** kan je een foutje in je git commit boodschap nog veranderen (als je een beetje van vi kent tenminste)

Oefening 4 – Afsplitsen/Forken

1. Zoals in de les. Open Android Studio en kloon (voor de 2e keer dus in een **nieuwe** lokale directory) het project https://gitlab.com/ucll2021-22/java-mobile-appusage.git

2. Maak een nieuw project aan uw Gitlab.com account genaamd:
   Java-mobile-appusage-<uw naam hier>
   Maak **Jorn.Jossa@ucll.be** en **Nadir.Aboulkassimi@ucll.be** ook opnieuw maintainer op dit project. Zie oef 1.

3. In Android Studio, onder Git, Manage Remotes… overschrijf de URL die daar staat met de Clone URL van je nieuwe gitlab.com project uit stap 2.
   Push alles naar jouw Central/Remote repository op Gitlab.com

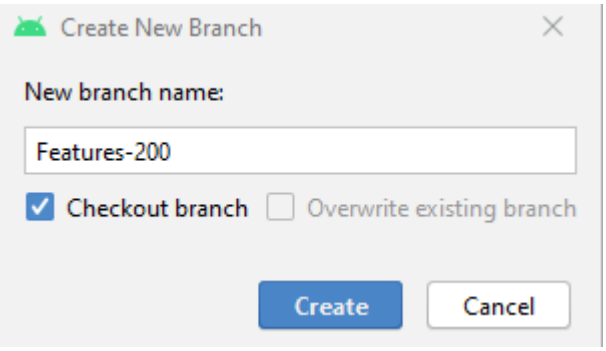
   Nu heb je echt uw eigen kopie waarvan je verder zou kunnen ontwikkelen

4. Doe een Git push om dit alvast naar de cloud / Gitlab.com te brengen

Oefening 5 – Branches

Nog steeds met het project uit Oefening 4 open in Android Studio.

1. Maak een nieuwe Branch aan. Klik daartoe onderaan rechts op Git:Master en vervolgens New Branch met als naam Features-200



   Voeg een bestand toe bv. een Java class.
   Add, Commit en Push dit alles.

2. Keer terug naar de Master branch door rechtsonder te klikken op Git: Features-200 / Master / Checkout

Begrijp je waarom je uw nieuw aangemaakt bestand niet meer ziet?

In het Menu Git / Merge Changes… Selecteer je branch en klik Merge.
Nu zou het bestand ook hier zichtbaar moeten zijn

3. Push nog een laatste keer alles naar de Central Server

Well done!