

Java Application Development

Graduaten programmeren

Werken met GIT en Gitlab.com

Versie 1.0, 14/09/2021



UC Leuven
Limburg
MOVING MINDS



Agenda

- Inleiding
 - Belang van broncodebeheer
 - Version Control System (VCS) types
 - Wat is GIT?
- Git repo in de Cloud
- Kernbegrippen / vocabulaire
- Command line oefening
- Android Studio oefening
- Branches



Wat is broncode?

- De programmeren cyclus, 10 talen keren per dag:
 - Programmeer => Build (compile, test, package, deploy) => Test
- Broncode – Source code (in de src folder)
 - Is de verzameling van bestanden ontstaan of bijgewerkt bij het programmeren en die samen een toepassing vormen
 - Onder andere *.java, *.xml, *.properties, *.gradle, *.gif, *.css, *.sql, ...
 - Vaak gegroepeerd in een 'project' in een editor zoals bv. IntelliJ, Visual Studio
 - Is de basis voor elke aanpassing of wijziging. Verlies ervan is ernstig!
 - Is NIET:
 - De gegenereerde output van het compilatie proces (de uitvoerbare binaries) of de bibliotheken die bij die compilatie gebruikt werden.
 - Cache, log of andere output bestanden ontstaan (of bijgewerkt) tijdens de uitvoering van het programma



Broncodebeheer / Versiebeheer

- Geen enkele professionele ontwikkelaar werkt zonder een of andere vorm van een **VCS – Version Control System**.
- Door gebruik te maken van een VCS kan je:
 - de source code **backuppen** (op het netwerk of in de cloud)
 - elk wijziging aan een bestand als een nieuwe **versie** opslaan
 - Vergelijken met een voorgaande versie om de wijzigingen te zien
 - De historiek inkijken en bv. een voorgaande versie actueel maken (restoren)
 - **samenwerken** met collega's/studenten op dezelfde broncode en terug versies vergelijken. Met aanduiding van wie, wat schreef.
 - Locking versus Merging
 - met "**feature branches**" werken
 - Nieuwe features/functionaliteiten toevoegen/uitproberen op een soort tijdelijke kopie zonder de "hoofdversie" te beschadigen



VCS types

- 3 types VCS
 - JUST BACKUP
 - Niet echt een VCS System
 - CENTRAAL
 - Microsoft Team Foundation Server (TFS)
 - Apache Subversion (SVN)
 - GEDISTRIBUEERD
 - GIT
 - Mercurial



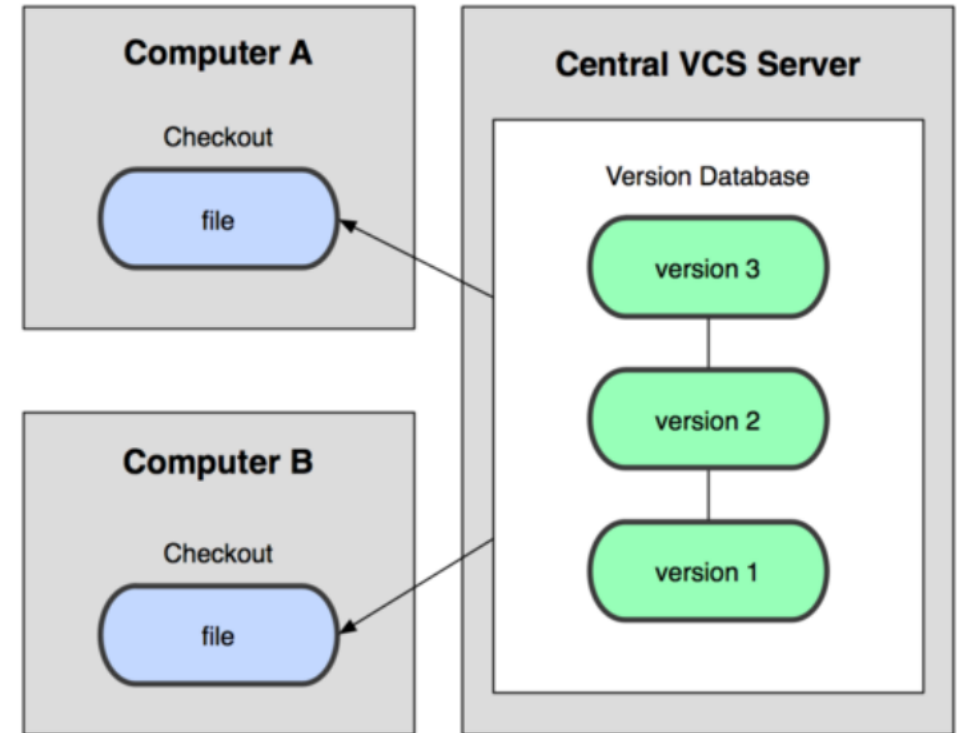
JUST BACKUP – Geen echt VCS

- Op uw persoonlijke Laptop/Desktop
 - Kopieer uw project folder of maak ZIP archief op dagelijkse basis
 - Kopieer de folder of zip
 - Op een externe harde schijf
 - In the cloud (bij voorkeur)
- Dit is simpel en OK indien
 - **Je op uw eigen werkt** of wat experimenteel werk doet
 - In de initiële **opstart fase** waarbij nog weinig aandacht is voor benaming, packages etc.....
- MAAR
 - Restoring betekent gaan 'zoeken' in de 'juiste' zip files
 - Kan na een tijdje veel disk space innemen



CENTRAL server based VCS

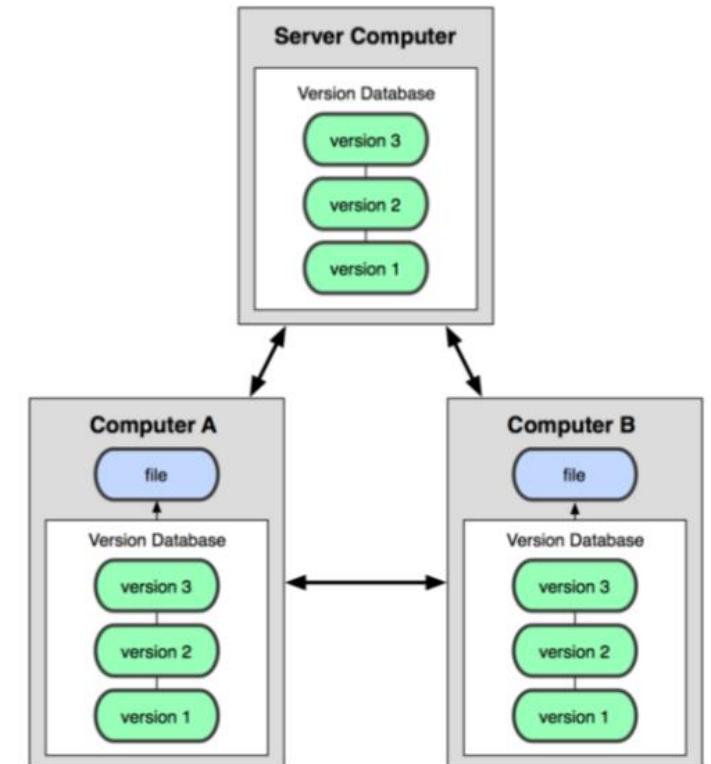
- Broncode wordt opgeslagen op de Central of Remote VCS server
- Zonder connectiviteit met de server kan je niks doen.
- Je hebt de server nodig om
 - Versiehistoriek te bekijken
 - Andere branch te openen
 - Code te committen / pushen
 - (te locken)
- Vaak gaat dit moeizaam en **traag**





GEDISTRIBUEERD VCS

- Er is nog altijd een **centrale server** MAAR elke progr. heeft een volledige kopie met alle versies en branches
- Grootste voordeel is de **snelheid!**
 - Switching feature branches
 - Compare branches
- Nadeel
 - Meer disk space nodig
- Locking niet mogelijk noch wenselijk
 - Merging als oplossing van broncode conflicten





Wat is GIT?

- Git (Global Info Tracker) is uitgevonden door [Linus Torvalds](#) in 2005 voor de verdere ontwikkeling van de Linux kernel.
- Git is een volledig **gedistribueerd VCS**
 - Het is **zeer snel** omdat de locale repository alles bevat
 - Het werkt erg goed in grote projectteams waarbij verschillende programmeurs aan verschillende onderdelen van het project werken in parallel elk op hun **eigen feature branch**.
- Een Git Repository is de verzamelnaam van **uw project onder versiebeheer**. Dit is inclusief alle versies, branches, commit messages, gebruikersinformatie, ...
- Meestal 1 project -> 1 Git Repo. Maak beide op hetzelfde moment aan



Centrale Git repo in de Cloud Github / Gitlab / Bitbucket ?

- Microsoft Github
 - focussed meer op Publiek opengestelde projecten
 - beperkt de team grootte van Privé projecten tot max 3 personen. Indien meer moet je een licentie betalen.
- Atlassian BitBucket
 - beperkt de team grootte van Privé projecten tot max 5 personen. Indien meer moet je een licentie betalen.
- Bij **Gitlab.com** is er geen beperking op de team grootte
 - Handig voor klas als 1 team op te zetten
 - Handig voor bij de geïntegreerde proeven
- Verder is de werking en de GIT commando's overal precies hetzelfde



Gitlab.com

- Gitlab account
 - Kan elke student een account aanmaken op gitlab.com (tenzij u er al een heeft)
 - Ga daartoe naar <https://gitlab.com>, Klik rechtsboven "Register".
 - Gebruik uw UCLL email adres:
<voornaam>.<naam>@student.ucll.be
- Iemand toegang geven tot uw GIT project
 - Voor elk toekomstig GIT project onder "Members" uw leerkracht toegang geven:
 - Jorn.Jossa@ucll.be en Nadir.Aboulkassimi@ucll.be
 - Role permission: **Maintainer** (Push ability required)
 - (Access Expiration Date: 31 augustus 2022)

Register for GitLab

First name	Last name
<input type="text"/>	<input type="text"/>

Username

Email

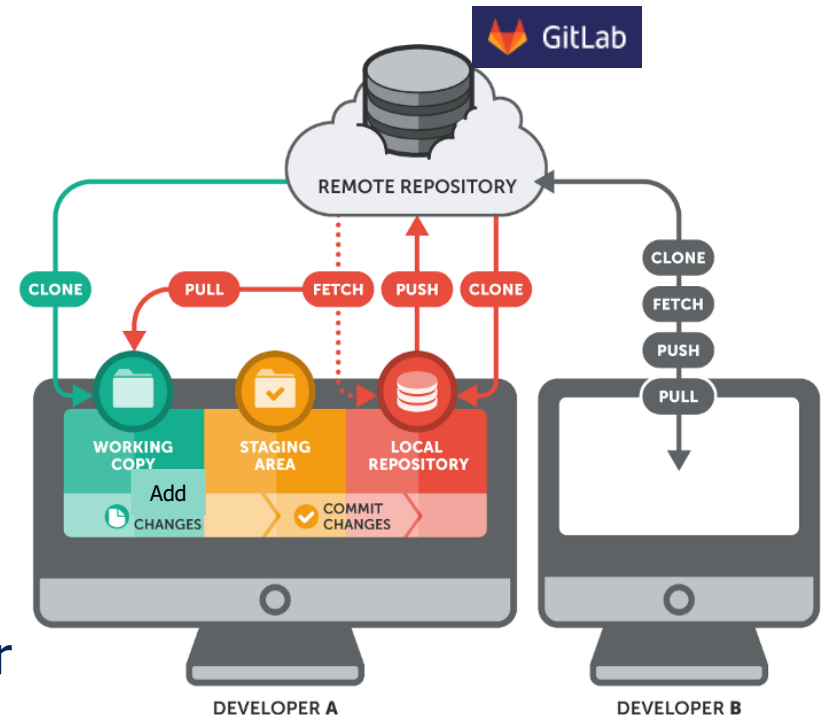
Password

Minimum length is 8 characters



Enkele kernbegrippen / vocabulaire

- Op de laptop/desktop
 - **Working directory / copy**
 - Het pad naar uw project (Android Studio project, IntelliJ project, ...) lokaal op je computer
 - 1 project komt normaal gezien overeen met 1 Git project/repo
 - **Local repository**
 - "onzichtbare" **.git** folder in de Working dir
- In het netwerk of in de cloud
 - Remote repository
 - GIT project/repo in het netwerk of in de cloud bv. op Github/Gitlab/Bitbucket





Working directory

- Groepeer al je projecten ergens op een zinvolle plaats
- Bv. C:\User\uw userid\UCLL-Projects
- Sla projecten NIET op op een OneDrive of zo. Dat vertraagt bv. het compilatie process enorm



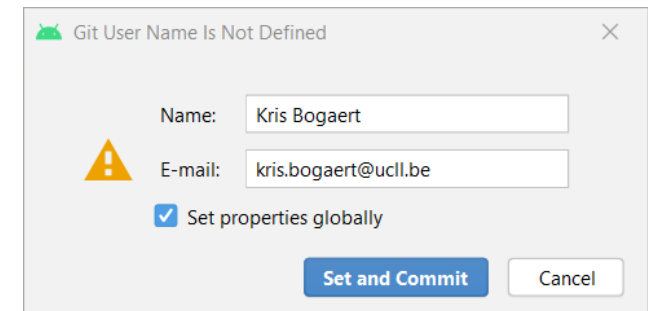
Kernbegrippen (Engelstalig)

- From local repo to central repo: **ADD**, **COMMIT** en **PUSH** ➡
 - Via **ADD** breng je bestanden onder versiebeheer. Dit doe je typisch NIET voor gecompileerde classes, log files, test output etc... Zie ook verder (.gitignore)
 - Via **COMMIT** zal je een "logisch" geheel van bestanden committen in de lokale repo samen met een verplichte COMMIT MESSAGE.
 - Via **PUSH** breng je uw wijzigingen van de lokale repo over naar de centrale repo en later via die weg naar de andere teamleden.
- From remote repo to local repo: **FETCH**, **MERGE** en **PULL** ⬅
 - Via **FETCH** krijg je de wijzigingen binnen van de centrale/remote repo in uw lokale repo
 - **MERGE** is het proces van het integreren van lokale repo wijzigingen in uw working directory. Maar soms zou die 'update' uw wijzigingen overschrijven. Dan hebben we een MERGE conflict.
 - **PULL = FETCH meteen gevolgd door MERGE**



Kernbegrippen (Engelstalig)

- CLONE
 - Je 'kloont' een remote GIT project in een working directory en die operatie maakt in de working directory je lokale .GIT repository aan die een volledige kopie is van de remote repo.
- CONFIG – Globale instellingen (overheen repo's):
 - je naam en e-mail instellen omdat zo iedereen zal zien wie/wat deed op de repo.
Gaat mee met de commit message.

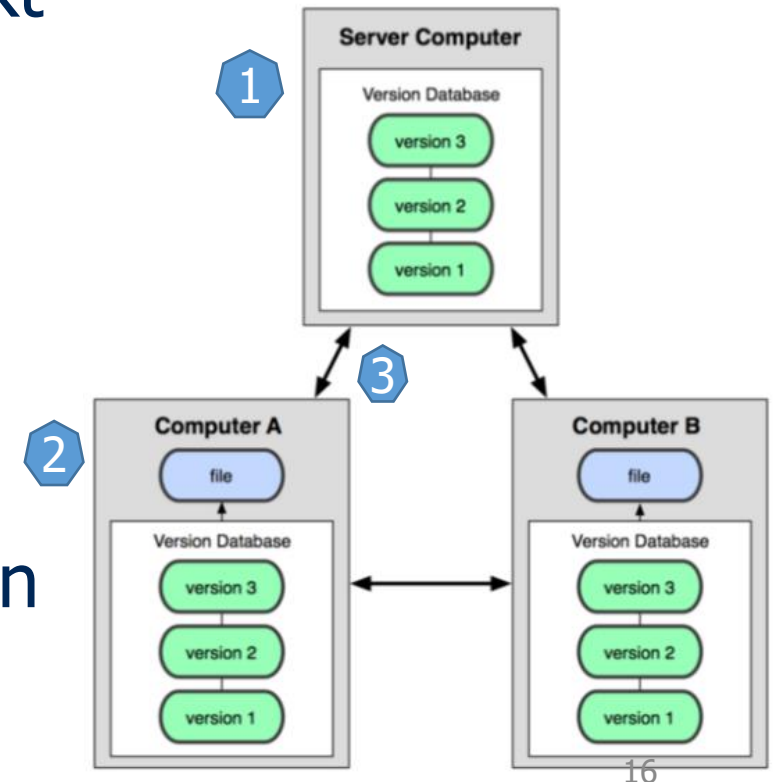


- `git config --global user.name "... uw voornaam naam ..."`
- `git config --global user.email x.y@student.ucll.be`



Command line oefening - Overzicht

1. Een project onder "Source Control" brengen begint vanop **gitlab.com** (de centrale/remote GIT repo) waar een nieuw GIT-project wordt aangemaakt
2. Navigeer naar de workspace of de locatie op je locale PC waar je dit project wil klonen
3. Clone project
4. Voeg bestand toe
5. Laat iemand anders bestand wijzigen
6. Update uw bestand





1. Remote GIT repo

New project

- Login to Gitlab.com met uw account en maak een nieuw Blank Privé project aan genaamd: TestGit_<uw naam>
- “Initialize repository with a README” **NIET** aanvinken

New project › Create blank project

Project name

TestGit Kris Bogaert

Project URL

https://gitlab.com/ KBogaert

Project slug

testgit-kris-bogaert

Want to house several dependent projects under the same namespace? [Create a group](#).

Project description (optional)

Description format

Visibility Level

☒ Private
Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.

☐ Public
The project can be accessed without any authentication.

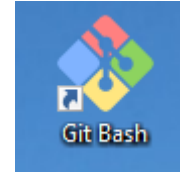
☐ Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project Cancel



2. Kloon project

- Op de lokale machine
 - Open Git Bash als command line editor
 - Eenmalig handelingen: (doe dit met UW gegevens natuurlijk)



```
KBoga@HPPav MINGW64 ~  
$ git config --global user.name "Kris Bogaert"  
  
KBoga@HPPav MINGW64 ~  
$ git config --global user.email "kris.bogaert@uc11.be"  
  
KBoga@HPPav MINGW64 ~  
$ git config --global -l  
user.name=Kris Bogaert  
user.email=kris.bogaert@uc11.be  
  
KBoga@HPPav MINGW64 ~  
$
```



2. Kloon project

- Op de lokale machine
 - Navigeer bv. naar de Home directory (C:\Users\<uw userid>)
 - Maak er een folder aan "UCLL-Projects" en ga in die folder
 - Kloon je project. Voorbeeldje: (maar jij moet **uw URL** gebruiken)
 - git clone <https://gitlab.com/KBogaert/testgit-kris-bogaert.git>
 - Een klein popup venstertje komt je userid en paswoord vragen van Gitlab
 - OF indien er geen password popup komt (Authentication failed):
 - git clone https://<userid>:<Password>@gitlab.com/ucll2020-21/testgit_kris_bogaert.git.
 - Vervang de @ in je userid, door %40
 - OF kuis een en ander op in Windows Configuratiescherf \ Gebruikersaccounts \ Referentiebeheer)



2. Kloon project

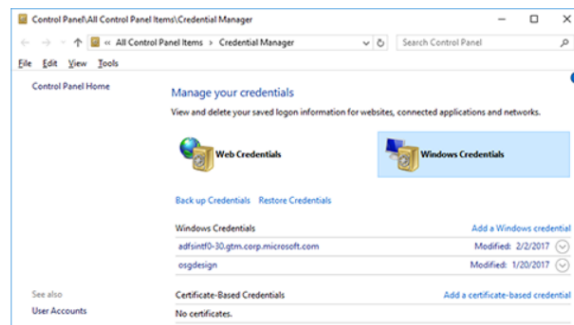
- Op de lokale machine
 - Kloon je project. Voorbeeldje: (maar jij moet **uw URL** gebruiken)
 - **Bij login problemen**, bekijk credentials in Windows Configuratiescherm \ Gebruikersaccounts \ Referentiebeheer)

Accessing Credential Manager

Windows 10

Credential Manager lets you view and delete your saved credentials for signing in to websites, connected applications, and networks.

1. To open Credential Manager, type **credential manager** in the search box on the taskbar and select **Credential Manager Control panel**.
2. Select **Web Credentials** or **Windows Credentials** to access the credentials you want to manage.





3. Maak source code aan

- CD in je project en maak een bestand aan bv.

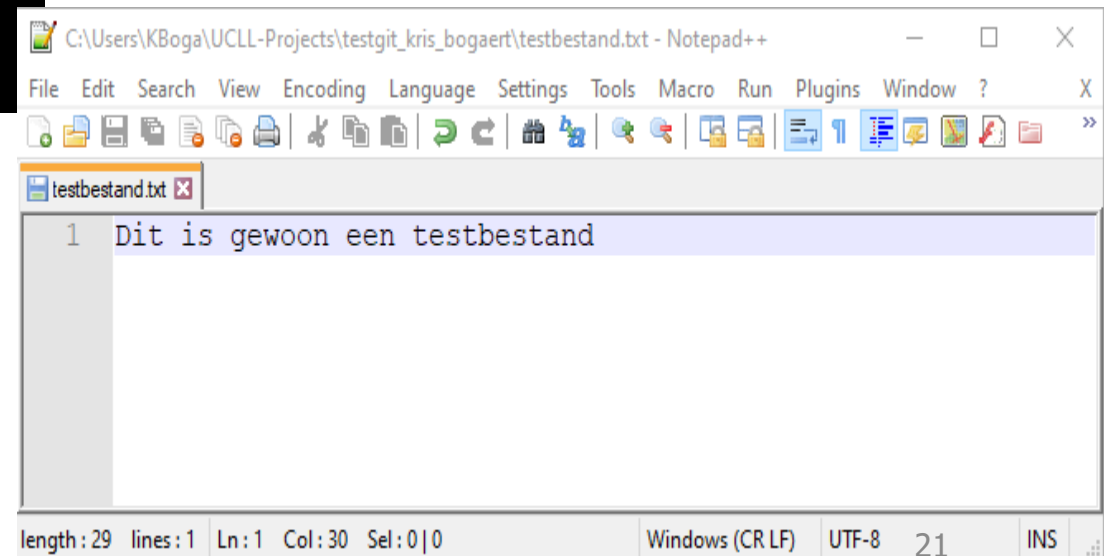
```
Cloning into 'testgit_kris_bogaert'...
warning: You appear to have cloned an empty repository.

KBoga@HPPav MINGW64 ~/UCLL-Projects
$ ls -al
total 16
drwxr-xr-x 1 KBoga 197612 0 sep 20 18:08 ./
drwxr-xr-x 1 KBoga 197612 0 sep 20 17:18 ../
drwxr-xr-x 1 KBoga 197612 0 sep 20 18:08 testgit_kris_bogaert/

KBoga@HPPav MINGW64 ~/UCLL-Projects
$ cd testgit_kris_bogaert/

KBoga@HPPav MINGW64 ~/UCLL-Projects/testgit_kris_bogaert (master)
$ touch testbestand.txt
```

- Bekijk het even in Windows Explorer en wijzig het





3. Git ADD, COMMIT en PUSH

```
KBoga@HPPav MINGW64 ~/UCLL-Projects/testgit_kris_bogaert (master)
$ git add testbestand.txt

KBoga@HPPav MINGW64 ~/UCLL-Projects/testgit_kris_bogaert (master)
$ git commit -m "Basisversie van het project"
[master (root-commit) f42427d] Basisversie van het project
1 file changed, 1 insertion(+)
create mode 100644 testbestand.txt

KBoga@HPPav MINGW64 ~/UCLL-Projects/testgit_kris_bogaert (master)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 257 bytes | 257.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://gitlab.com/uc112020-21/testgit_kris_bogaert.git
* [new branch]      master -> master
```

- Resultaat op de centrale repo:

Kris Bogaert > TestGit Kris Bogaert > Repository

main


testgit-kris-bogaert / testbestand.txt


Find file



Blame

History

Permalink

 Eerste versie
Kris Bogaert authored 1 minute ago

856c259a 




 testbestand.txt  29 Bytes

Edit

Web IDE

Replace

Delete

1	Dit is gewoon een testbestand
---	-------------------------------



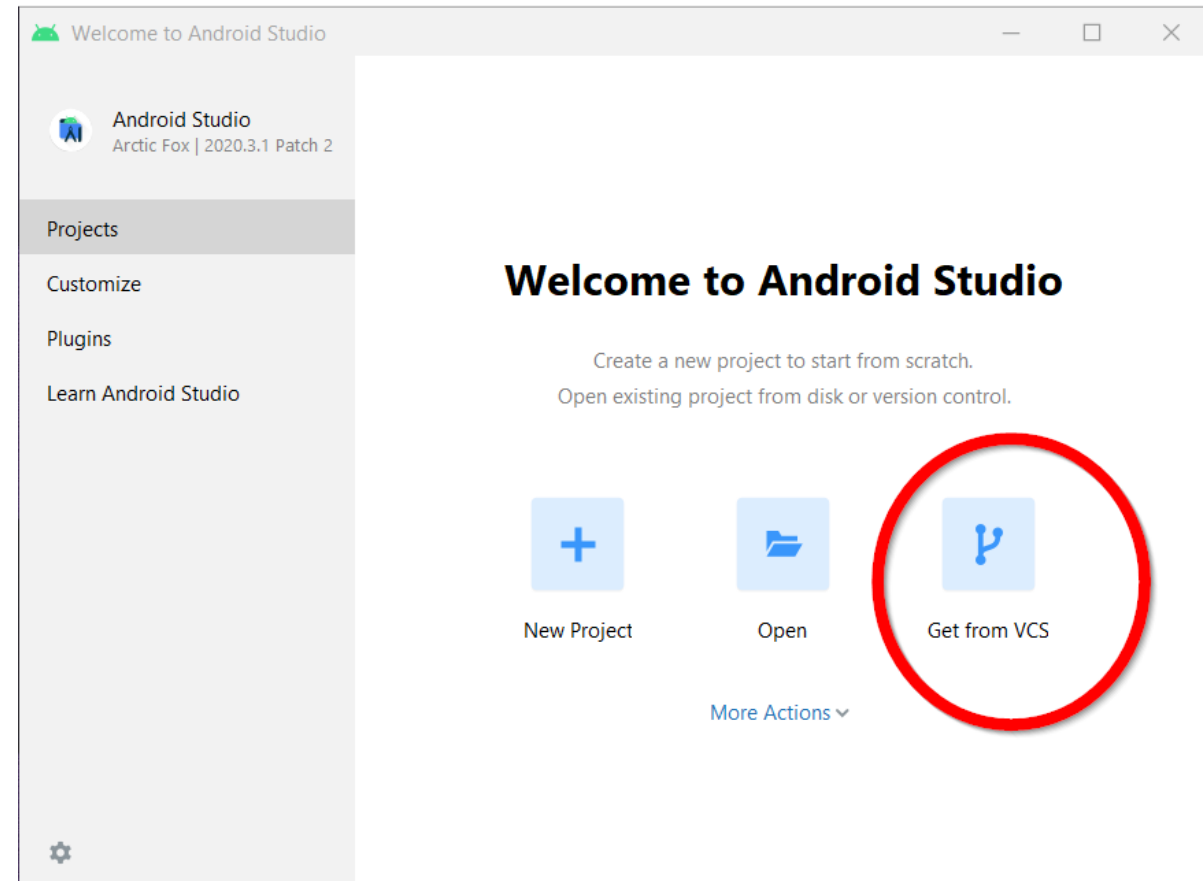
Afsplitsen - Forking

- In dit scenario vertrek je van een Kloon van een bestaande centrale repo om je eigen centrale repo te maken en daarop verder te werken.
Je splitst dus af om je eigen weg te gaan.
- Dit scenario moeten jullie bij vrijwel elke les of elke oefening uitvoeren!
 1. Kloon de repo met de aanzet van de les of de oefening
 2. Maak een nieuw gitlab project aan
 3. Switch de 'Remote' zodat er niet langer naar de originele repo maar wel naar de nieuwe repo gerefereerd wordt
 4. Push into new repo



Android Studio oefening

- Start Android Studio
- On the welcome screen select "Get from VCS"
- Of indien Android Studio al open is
In het Menu, selecteer File / New / Project from Version Control...





Existing Git repo

- Kloon de volgende repo en save lokaal in uw projectenfolder:
<https://gitlab.com/ucll2021-22/java-mobile-appusage.git>

Version control: Git

URL: https://gitlab.com/ucll2021-22/java-mobile-appusage.git

Directory: C:\Workspaces\Android\Kris\java-mobile-appusage

Log In to gitlab.com

Enter credentials

Username: kris.bogaert@ucll.be

Password:

☒ Remember

☐ Use credentials helper

Log In Cancel

- Yes, Trust en Open gelijk het project (in this window)

Open Project

Where would you like to open the project?

☐ Don't ask again This Window New Window Cancel

Trust Gradle Project?

If you don't trust the source, stay in safe mode.

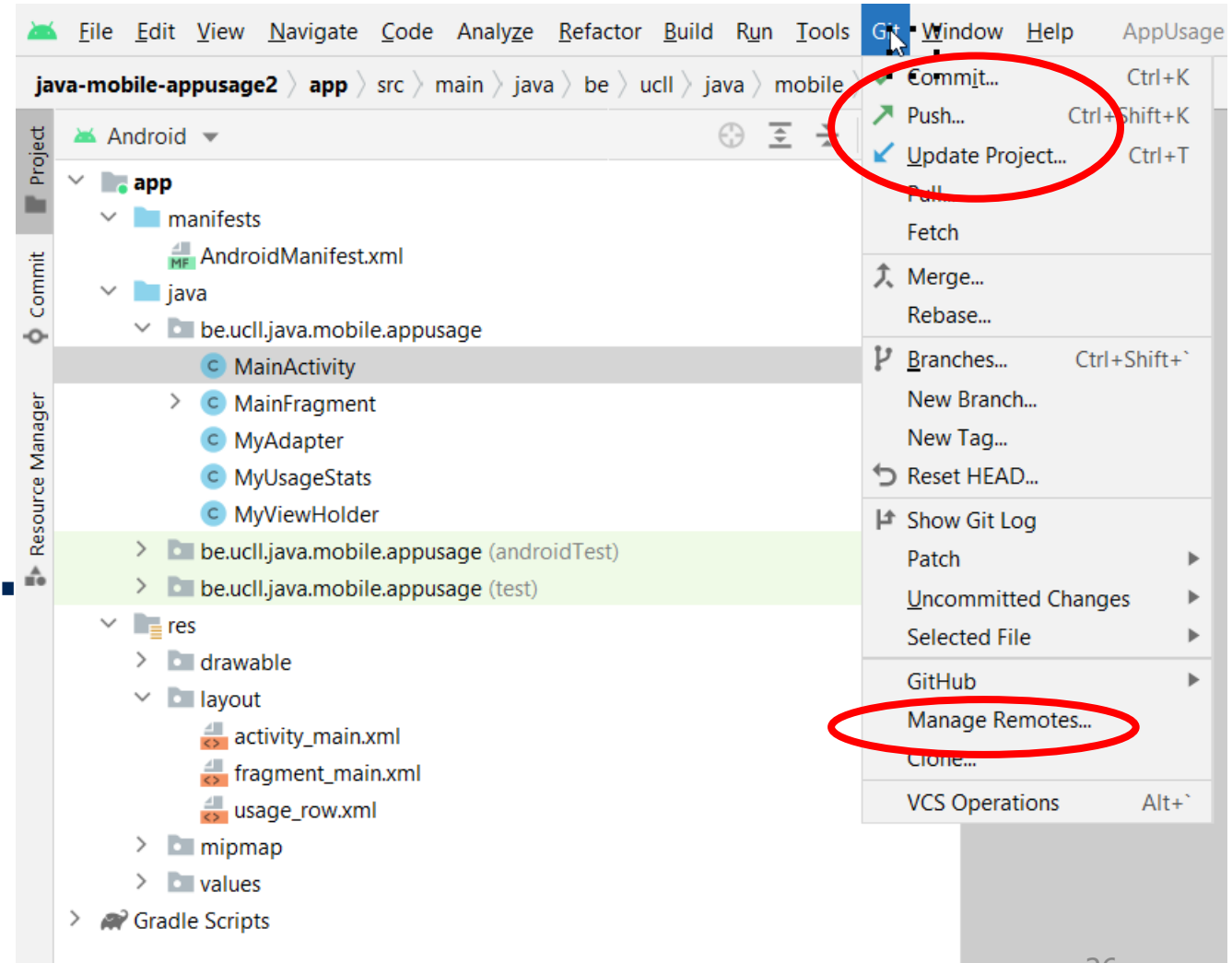
Loading, running or building a Gradle project may execute potentially malicious code from its build scripts.

Trust Project Stay in Safe Mode



Android Studio - VCS

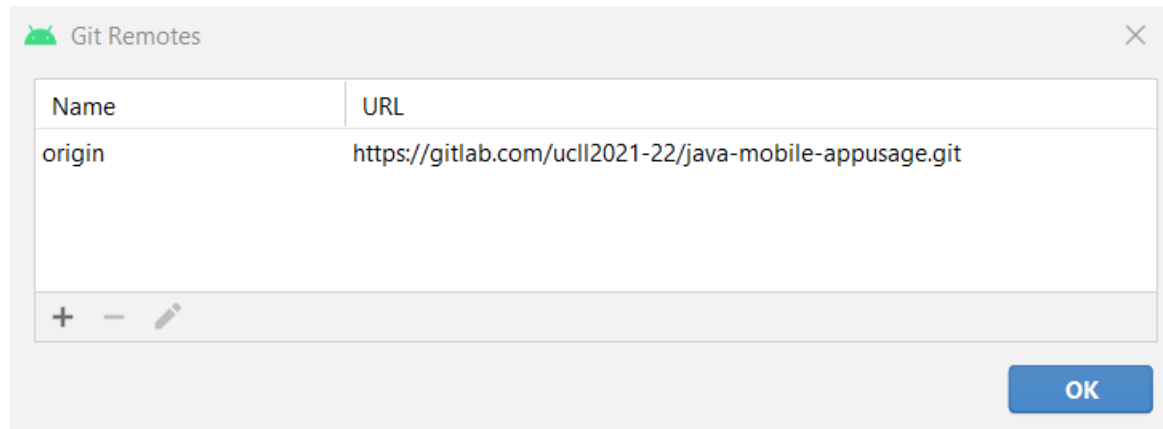
- Er is een Git Menu met herkenbare commandos. Idem als in IntelliJ
- Gebruik **Manage Remotes...** om te switchen van een Remote central GIT server naar uw eigen afsplitsing (Fork)





GIT Remotes

- Onder het menu Git / Manage Remotes...

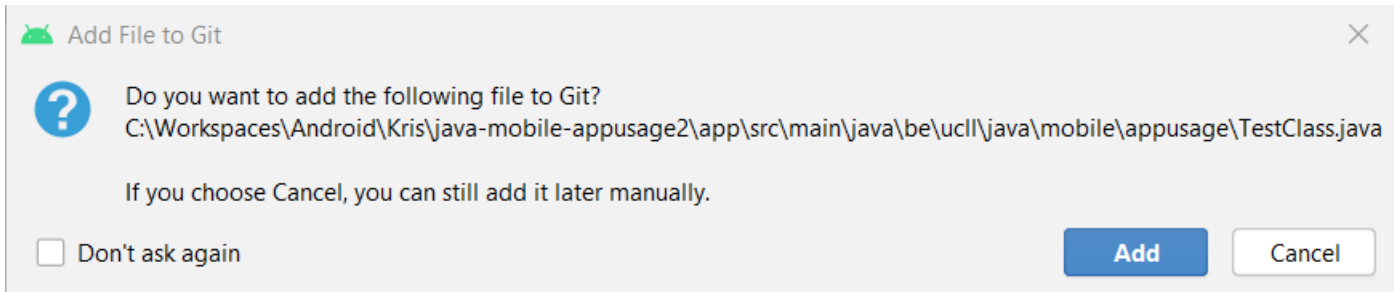


- Om te forken moeten we 2 dingen doen
 - Eerst een nieuwe lege Centrale GIT repo aanmaken op Gitlab.com
 - Bovenstaande Remote wijzigen ✎ naar de zonet aangemaakte GIT
- Push de volledige lokale repo naar de nieuwe remote repo



Add & Modify some content

- Create a new Java class file
 - For example, create "Test Class" in `be.ucll.java.mobile.appusage`
 - Add the file to GIT

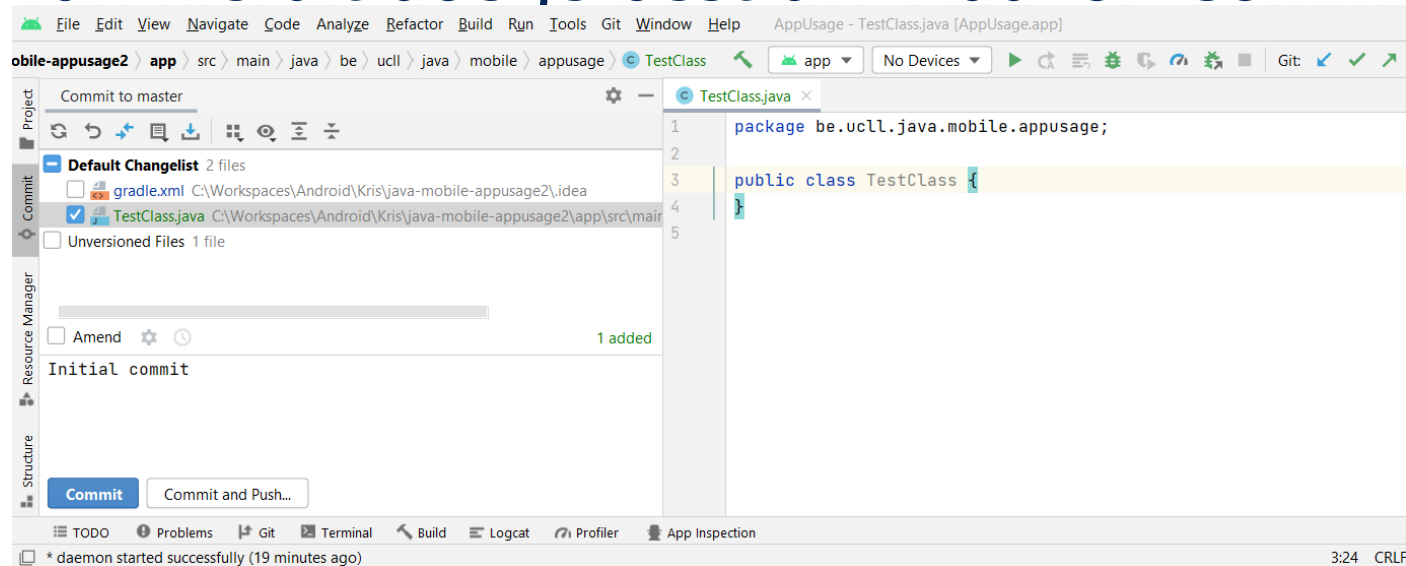


- If you select no, the file will have a "brown" color indicating it is not part of the GIT repo
- Modify some existing Java code
 - Optionally try doing a Revert of your changes



Commit tab en Push

- Links heb je de Commit tab.
Van hieruit doe je best uw 'Add' of 'Commit' en Push



- Expand Default and select relevant files
- Enter a commit message like: "Initial commit"
A commit message is MANDATORY



Commit Content

- Normaal 'commit' je een 'eenheid van werk'. Een aantal bestanden die bij elkaar horen en samen aangemaakt of gewijzigd werden
- Commit NOOIT code dat niet compileert
 - Test dus je code vooraleer je commit
 - Of Stash je code
Hierbij parkeer je even je code opzij om ze later (op dezelfde brach) weer te "apply"-en



Git Excluded files & folders

- Normally you'll want to start excluding some files from GIT
 - You don't want your **target folder** to be in your VCS, the target folder is generated by IntelliJ (or maven) when it compiles your code.
- A VCS should only contain sources, images, javascript, html, xml and so forth (sources)
 - **NO generated files**, because they are generated on your machine based on some metadata (class files are generated based upon your source, your IntelliJ project files are generated by maven, and so forth)



Git Excluded files & folders

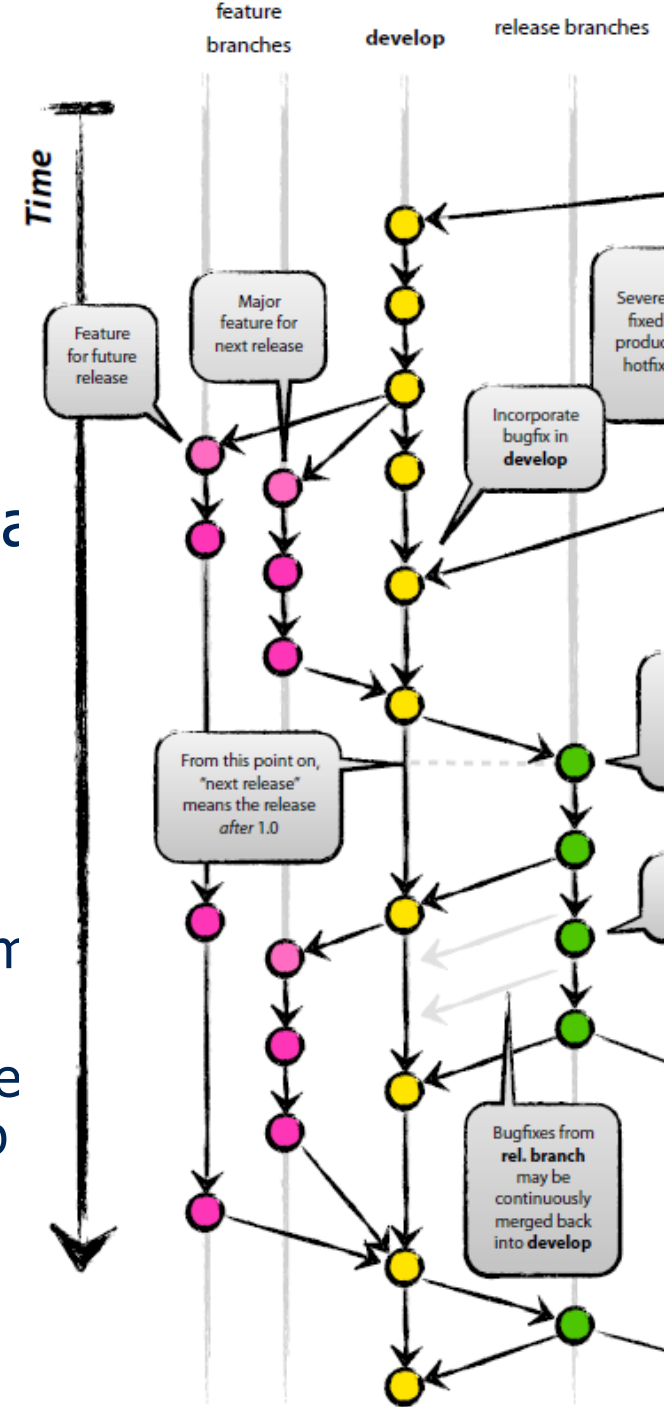
- The file containing the files and/or directories to ignore is called **.gitignore**
 - Typically sits in the project root folder, but this is not mandatory
 - See <http://git-scm.com/docs/gitignore> for syntax and usages
 - See <http://gitignore.io> for support in generating a suitable file
- Switch to 'Project' mode to see the file since files starting with . are hidden files.
- You can edit the .gitignore by hand (using a text editor) or use your IDE
- For the repo you've cloned, the .gitignore was already added.



Branches

- **Features branches**

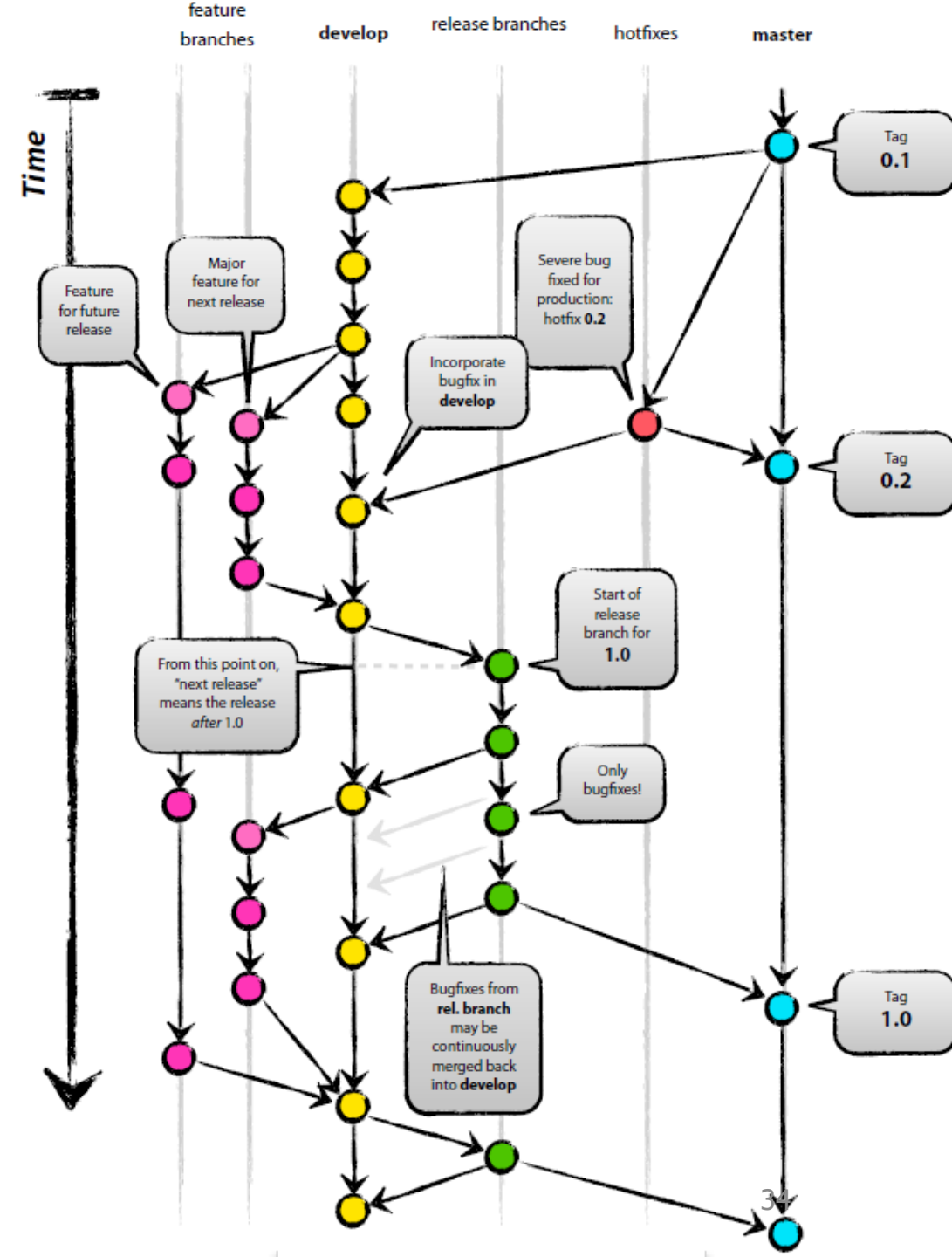
- A Feat. branch is a temporary copy to fix a defect or develop a new feature / functionality. Followed by a peer review.
- Branch name: prefix + nr.
 - F.i. FEAT001, DEF001, CR001
 - The Prefix and number often follows a separate Ticketing system such as f.i. Atlassian Jira/Trello.
 - The Ticketing system describes the feature/change request to develop or the defect to solve
 - Analysis documents
 - Mails, Schema's, Screenshots
 - Costing and Planning





Branches

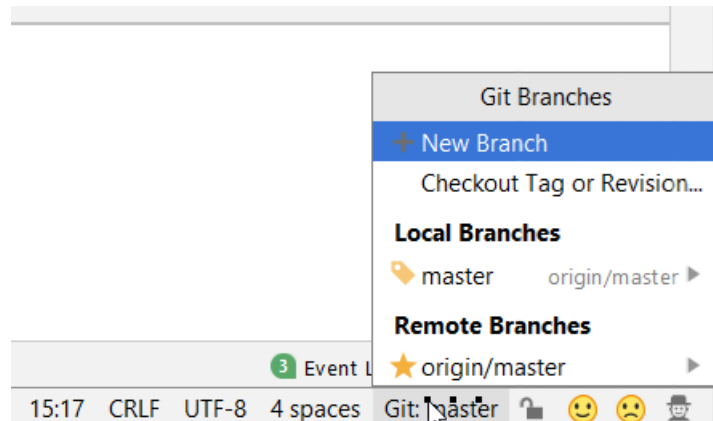
- **Release branches**
 - Stabilize the code by separating the next periodical release in a dedicated Release branch
 - Deploy on Test environment
 - Branch name often like
 - V1.0
 - Release 1.0
 - Version numbering Ma.Mi.Bu.Fi
 - Major version
 - Minor version
 - Build
 - Fix





Branch aanmaken

- Klik rechtsonder op Git:Master en selecteer New Branch

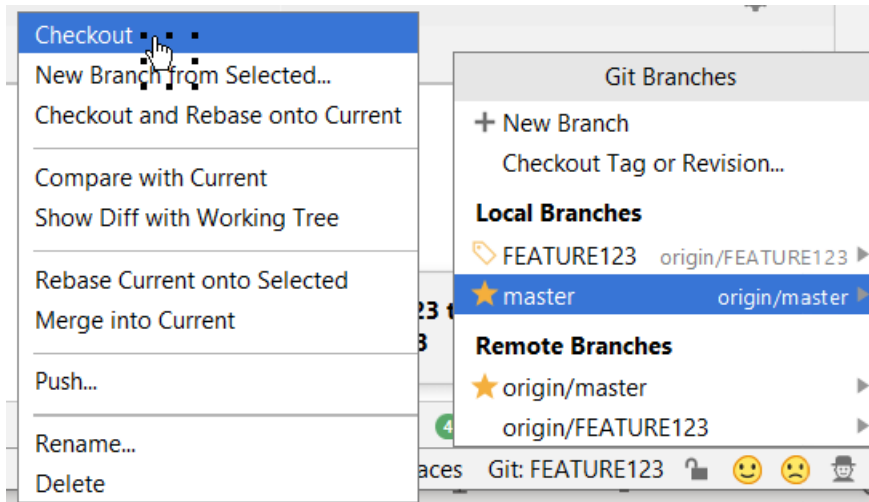


- Naam FEATURE123 en laat Checkout branch aangevinkt
- Voeg een Nieuw bestand toe. Git Add, Commit en Push



Checkout branch

- Checkout Master. Waar is mijn bestand heen???

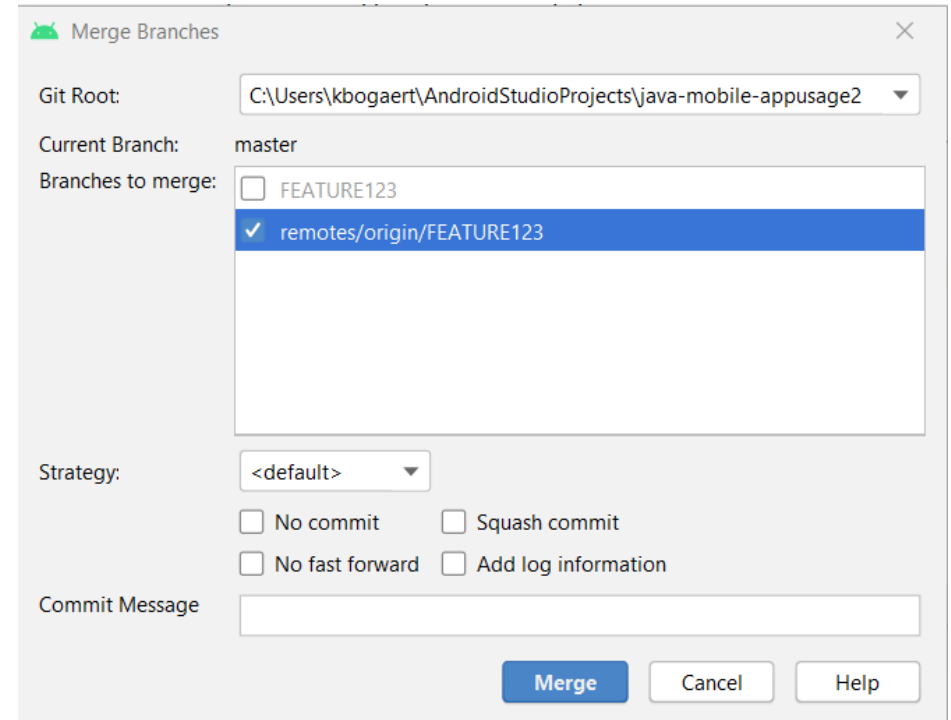


- Checkout Feature123 terug. Is bestand terug?



Merge Branch into Master

- Checkout terug naar de Master
- Merge de FEATURE123 branch in de Master via VCS / Git / Merge Changes
- Push
- Bekijk dit eens op [Gitlab.com](https://gitlab.com)
- Hierna kan je uw branch deleten.





GIT Documentatie

- Git algemeen: <https://git-scm.com/doc>
- Gitlab.com permissies en rollen:
<https://docs.gitlab.com/ee/user/permissions.html>



Extra slides

- Hierna wat verdiepingsinformatie



Werken met SSH en PPK

- Uiteraard zijn de GIT repos in de cloud (gitlab.com) beveiligd qua toegang.
 - Je kan toegang verkrijgen tot je repo via HTTPS + Userid + Paswoord
 - Of je kan als alternatief een Public Private Key encryptiesleutel gebruiken
 - Die zal de communicatie tussen uw computer en de GIT repo in de cloud versleutelen. Ofwel er wordt een Secure Shell gecreerd (SSH)
 - Die zal u identificeren / authenticeren zodat je geen userid paswoord meer in hoeft te geven



Werken met SSH en PPK

- Open terug Git-bash en ga naar je home dir met het commando: `cd`
- Controleer of er al een directory bestaat: **.ssh** met het commando: `ls -al`
 - Indien niet, maak de folder aan met: `mkdir .ssh`
- Ga in de folder met: `cd .ssh`
- Controleer even of deze 2 bestanden al bestaan in de .ssh folder: `id_rsa.ppk` en `id_rsa.ppk.pub`
 - Indien niet, voer volgend commando uit: `ssh-keygen -t rsa -b 2048`
 - Klik 3 maal Enter om de defaults te accepteren



Werken met SSH en PPK

```
EAD+kbogaert@US-CND4286RFS MINGW64 ~/.ssh
$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/kbogaert/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/kbogaert/.ssh/id_rsa
Your public key has been saved in /c/Users/kbogaert/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:R1eA5oNG0vXhGPagwL7OMDZW5BmIhwQGppjwPMYCNN8 EAD+kbogaert@US-CND4286RFS
The key's random image is:
+---[RSA 2048]-----+
|XB o...ooo.o..|
|O== = ..ooo= o|
|++*= E o ++ +|
| o..= o.o.|
| . . .S ..|
| *|
| o *|
| o|
|-----[SHA256]-----+

EAD+kbogaert@US-CND4286RFS MINGW64 ~/.ssh
$ ls -al
total 17
drwxr-xr-x 1 EAD+kbogaert 4096 0 Sep 20 18:48 ./
drwxr-xr-x 1 EAD+kbogaert 4096 0 Sep 20 18:42 ../
-rw-r--r-- 1 EAD+kbogaert 4096 1831 Sep 20 18:48 id_rsa
-rw-r--r-- 1 EAD+kbogaert 4096 408 Sep 20 18:48 id_rsa.pub

EAD+kbogaert@US-CND4286RFS MINGW64 ~/.ssh
```

- Er werden nu 2 bestanden aangemaakt
 - id_rsa.ppk => Dit is uw PRIVATE sleutel
 - id_rsa.ppk.pub => Dit is uw PUBLIEKE sleutel
- Open id_rsa.ppk.**pub** met een editor zoals notepad
 - Kopieer de volledige inhoud



Werken met SSH en PPK

- In gitlab.com, ga naar uw persoonlijk profile en klik links op SSH Keys
- Paste de inhoud van uw id_rsa.ppk.pub in het tekstvak

The screenshot shows the GitLab web interface. The browser address bar displays 'gitlab.com/~profile/keys'. The left sidebar contains the 'User Settings' menu with 'SSH Keys' highlighted. The main content area is titled 'SSH Keys' and includes a search bar. Below the title, there is a description of SSH keys and a large text input field for pasting the public key. At the bottom, there are fields for 'Title' and 'Expires at', followed by an 'Add key' button.

SSH Keys - User Settings - GitLab

gitlab.com/~profile/keys

GitLab Menu Search GitLab

User Settings > SSH Keys

Search settings

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

To add an SSH key you need to [generate one](#) or use an [existing key](#).

Key

Paste your public SSH key, which is usually contained in the file '~/.ssh/id_ed25519.pub' or '~/.ssh/id_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Do not paste your private SSH key, as that can compromise your identity.

Typically starts with "ssh-ed25519 ..." or "ssh-rsa ..."

Title

e.g. My MacBook key

Expires at

mm/dd/yyyy

Give your individual key a title. This will be publicly visible.

Key will be deleted on this date.

Add key



Git important commands

- Checkout
 - Switch to a different branch or back to develop or master
- Branch
 - Shows on which branch you are
 - Can create a branch if provided with a name
 - -a shows all branches
 - -r shows all remote branches
- Stash



GIT

Branching

- Create a feature branch
 - Git branch <yourbirthdate>
 - *git branch ddmmyyyy*
 - Ex git branch 01012000
 - Checkout the branch
 - *git checkout <yourbirthdate>*
 - Check the files in the branch
 - *ls*
 - They should be the same as on the master branch



GIT Branching

- We will now add a file on the branch
- Add any Java file to your project and commit the file.



GIT

Branching

- Go back to the master branch
 - *git checkout master*
 - You'll notice that your new file is "gone"
 - Make sense, the new file exists on the branch, not on the master
 - *git checkout <yourbirthdate>*
 - The file is back again, since we are back on the branch



GIT Merging

- We will now merge the master with the feature
 - *git checkout master*
 - *git merge <yourbirthdate>*
 - *git commit -m "merging <yourbirthdate>"*
 - You should now see your file added on the branch on your master branch



GIT

Stashing

- When you are working on code which has modified items, you cannot switch to another branch
 - Git will tel you: *“error: you have local changed to “xxx”; cannot switch branch*
- When you “stash” you save those modifications which you can then replay later
 - **git stash**
 - After this you will be allowed to change branch
 - **git stash apply**
 - When you come back on the branch you were working on you can apply the changes again
- Instead of using Stash you could have made a “Commit”. But it is VERY BAD practice to commit untested/unfinished code. Therefore: Stash!