

Version control GIT

Elektronica – ICT

Sven Mariën

(sven.marien01@ap.be)

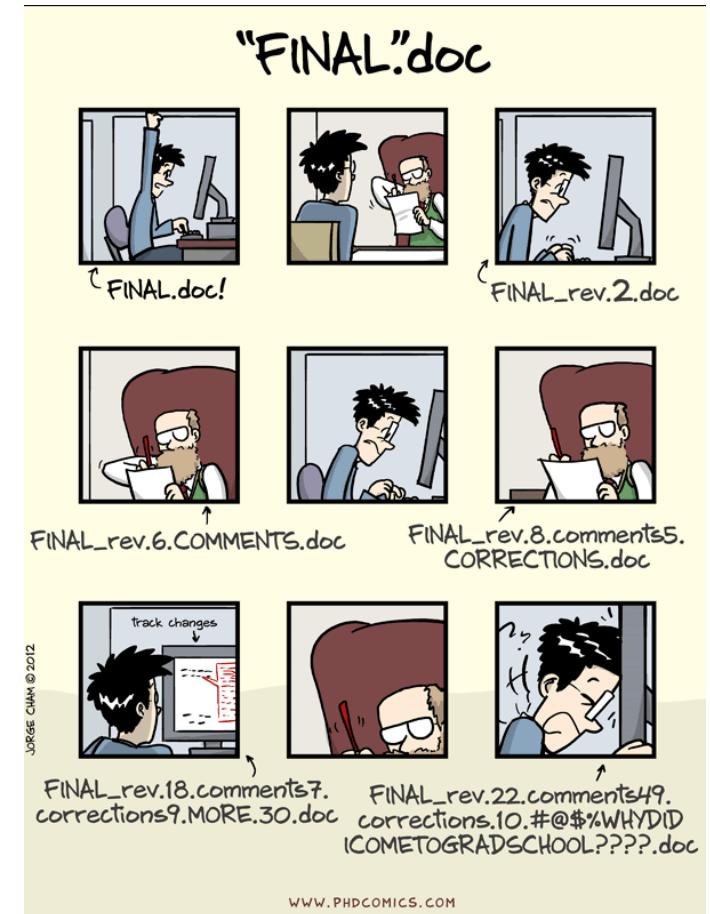
2018-2019



ARTESIS PLANTIJN
HOOGESCHOOL ANTWERPEN

Wat is version control ?

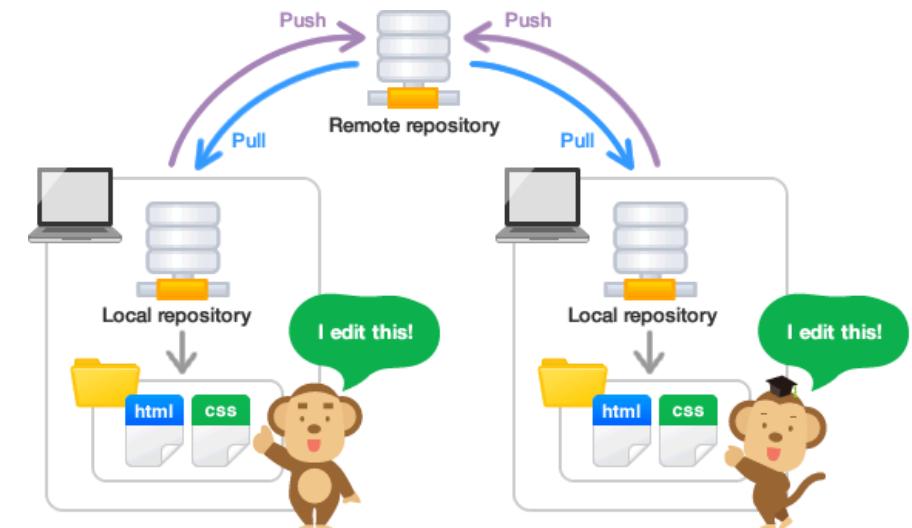
- Version control
- Meest gebruikte shortcut in elk programma: CTRL+Z
- Version control is een UNDO knop op steroïden
- Ga terug of herstel van wijzigingen die je een minuut geleden hebt gemaakt ...
 - ... of een week geleden.
 - Spring tussen verschillende versies
- Version control systemen **tracken alle wijzigingen** die je maakt in je code, configuratie, tekstbestanden, ...



Source code control

- Source code control = version control
- + efficiënt bestanden delen en samenwerken in dezelfde code
- + wie maakte een wijziging?
- + wat is het verschil tussen de staat nu en vorige week?
- + welke bestanden werden vaak aangepast?
- + identificeer releases van je code => betrouwbare builds

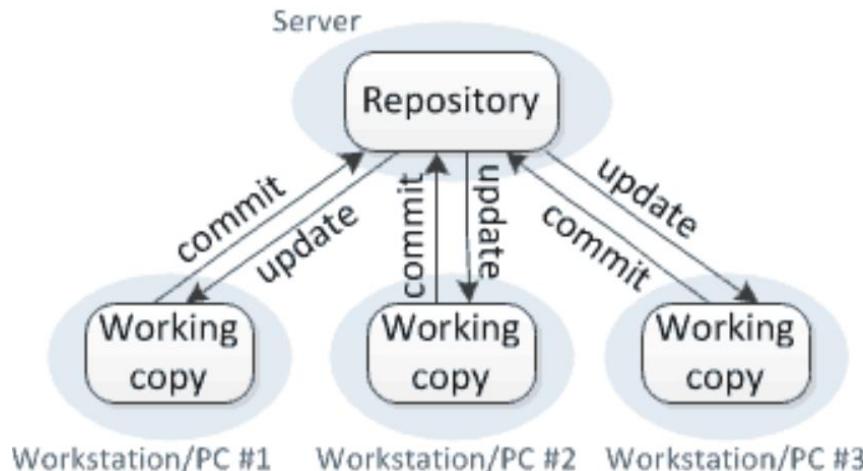
Gebruik altijd source code control! Zelfs al werk je alleen aan een project.



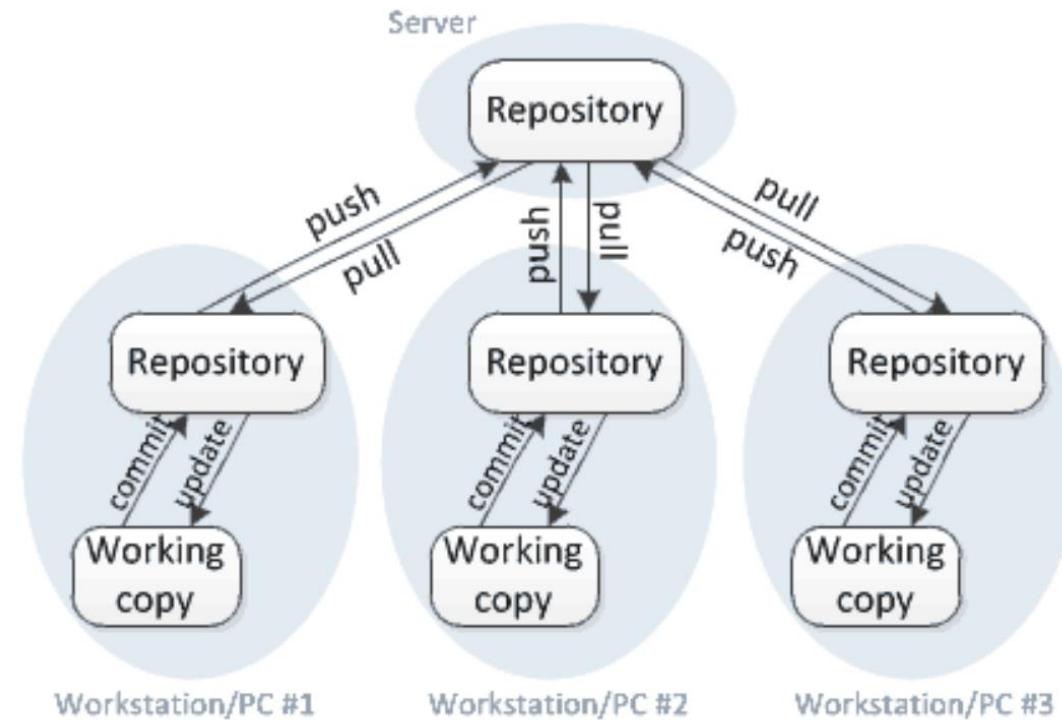
Types van systemen

- Centralized vs. Distributed
- Subversion, TFS,.. vs GIT,..

Centralized version control

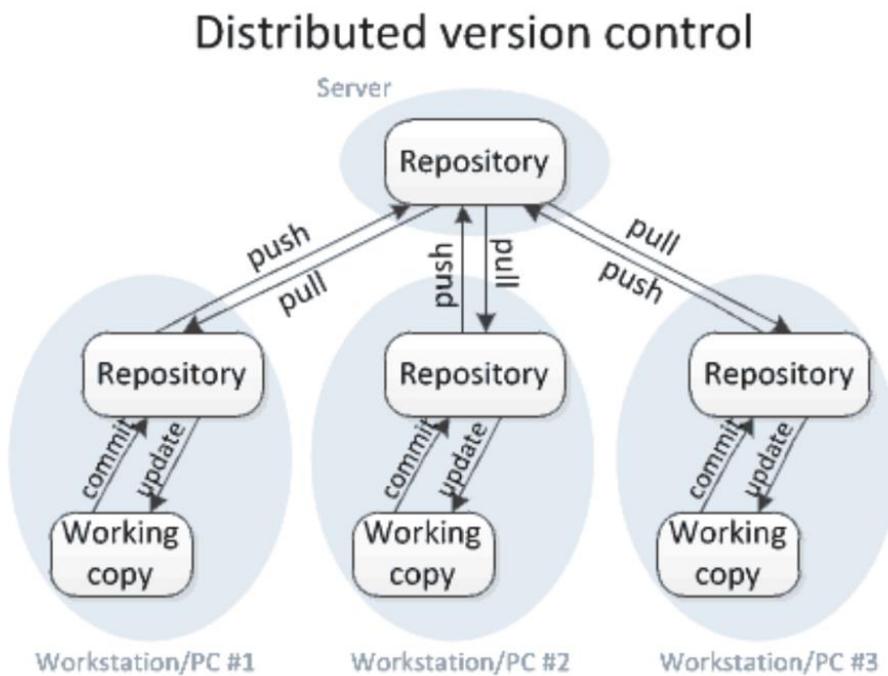


Distributed version control



Voordelen van “Distributed” Systemen

- Aangezien de repository lokaal staat zijn de meeste operaties lokaal en dus snel.
 - Het opvragen van historiek.
 - Het opvragen van de wijzigingen die gebeurd zijn tussen bepaalde versies.
 - Je kan dus gerust “offline” werken (netwerk problemen, vliegtuig,...)
 - Je hebt steeds een backup bij de hand van de volledige server repository
 - ...



Git Junior (local)

Elektronica – ICT

Sven Mariën

(sven.marien01@ap.be)

2017-2018



ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

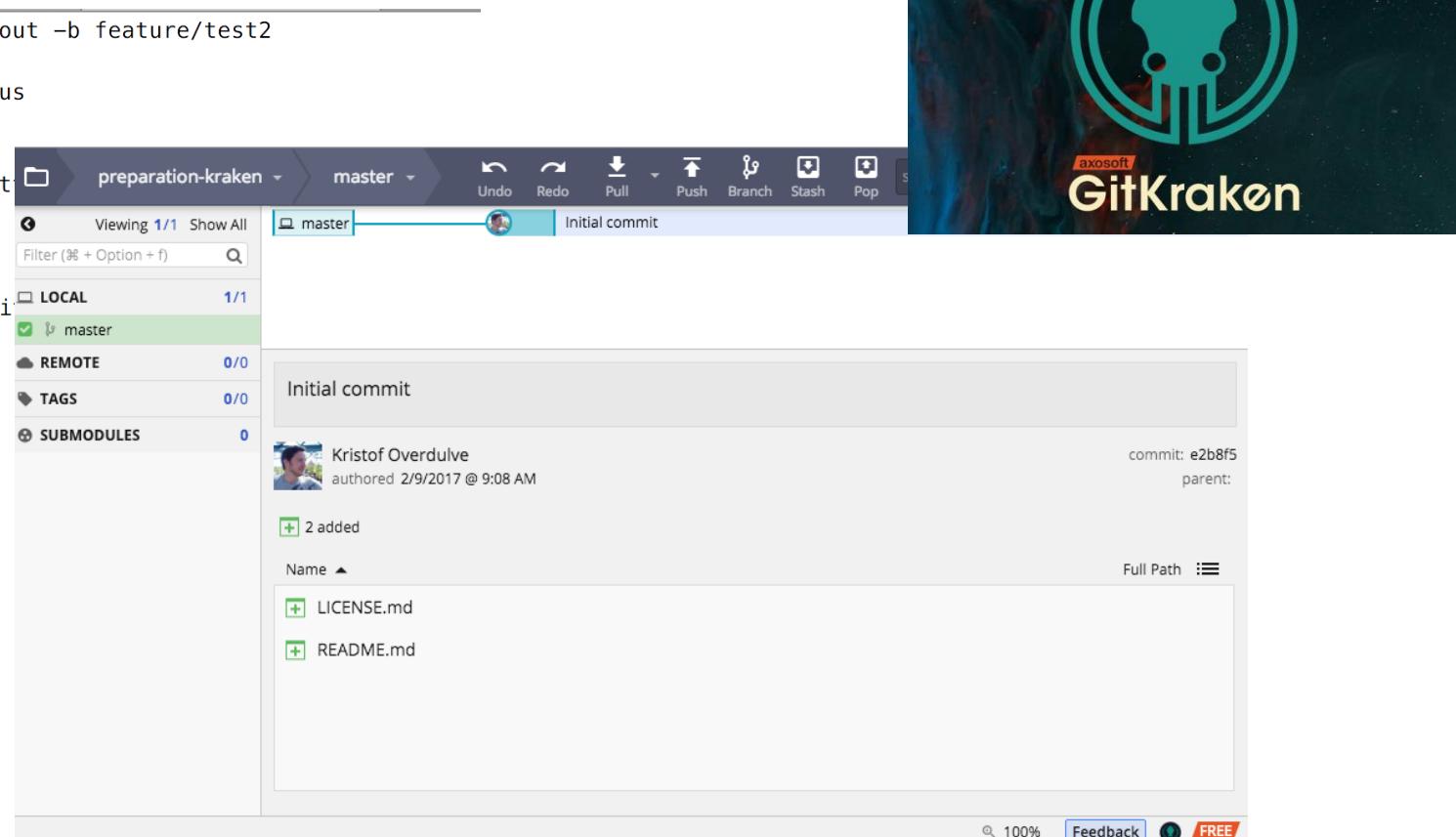
Installatie

- Installeer Git: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
- Installeer Gitkraken: <https://www.gitkraken.com/>

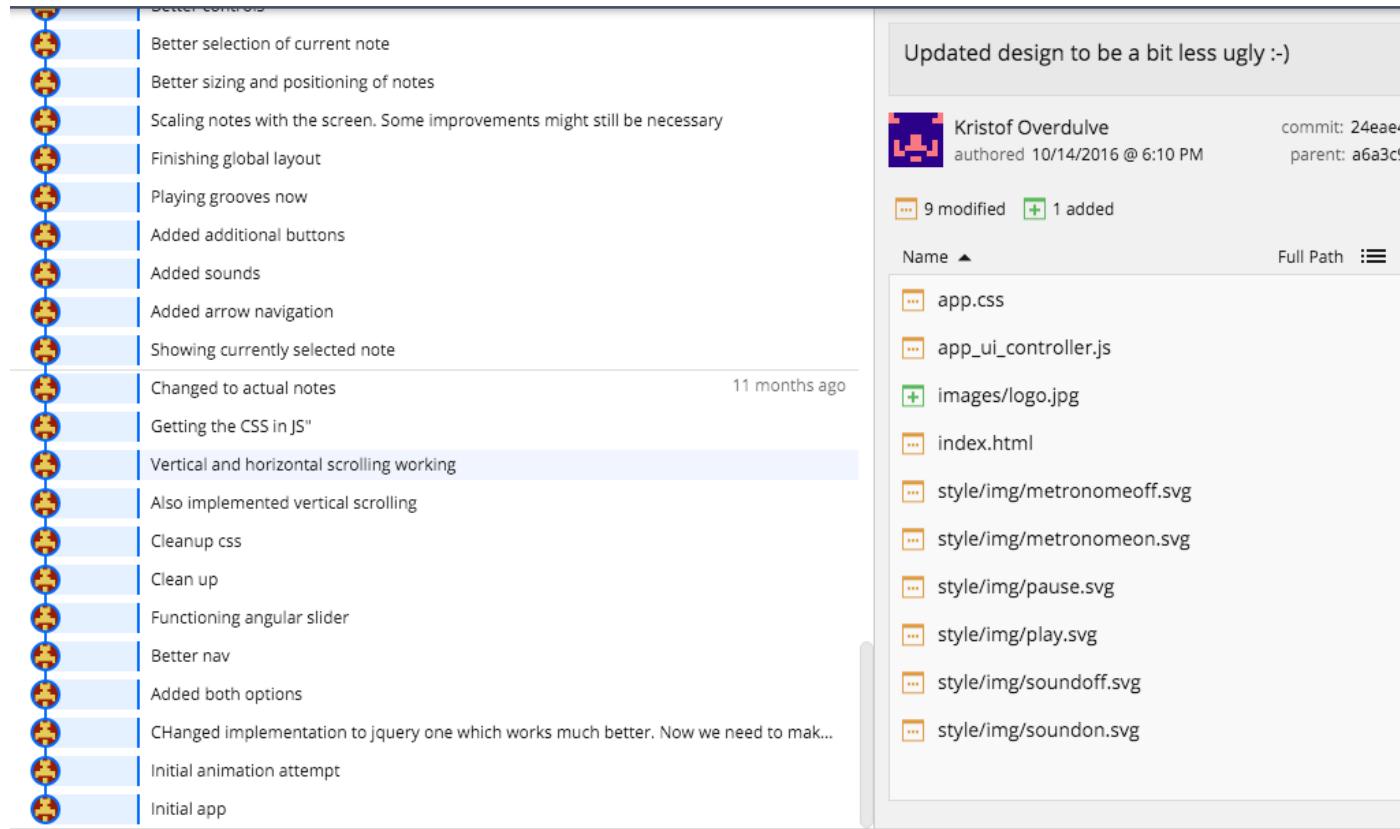
```
~/Development/LoRa-PW0/mieweb ➜ git checkout -b feature/test2
Switched to a new branch 'feature/test2'
~/Development/LoRa-PW0/mieweb ➜ git status
On branch feature/test2
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    dbexport.sql

nothing added to commit but untracked files present (use "git add" to track)
~/Development/LoRa-PW0/mieweb ➜ git status
```



Git concepten: “Commit”



Een commit is een snapshot van een bepaalde staat van je code op het moment dat de commit werd gemaakt.

1. Een commit message dat beschrijft wat je hebt gemaakt/gewijzigd.
2. Een SHA1 hash van je commit. Deze zorgt voor integriteit van je code zodat geen bestand kan worden aangepast zonder dat Git het weet.
3. De bestanden die toegevoegd/gewijzigd zijn.

Git Concepten: “Commit”

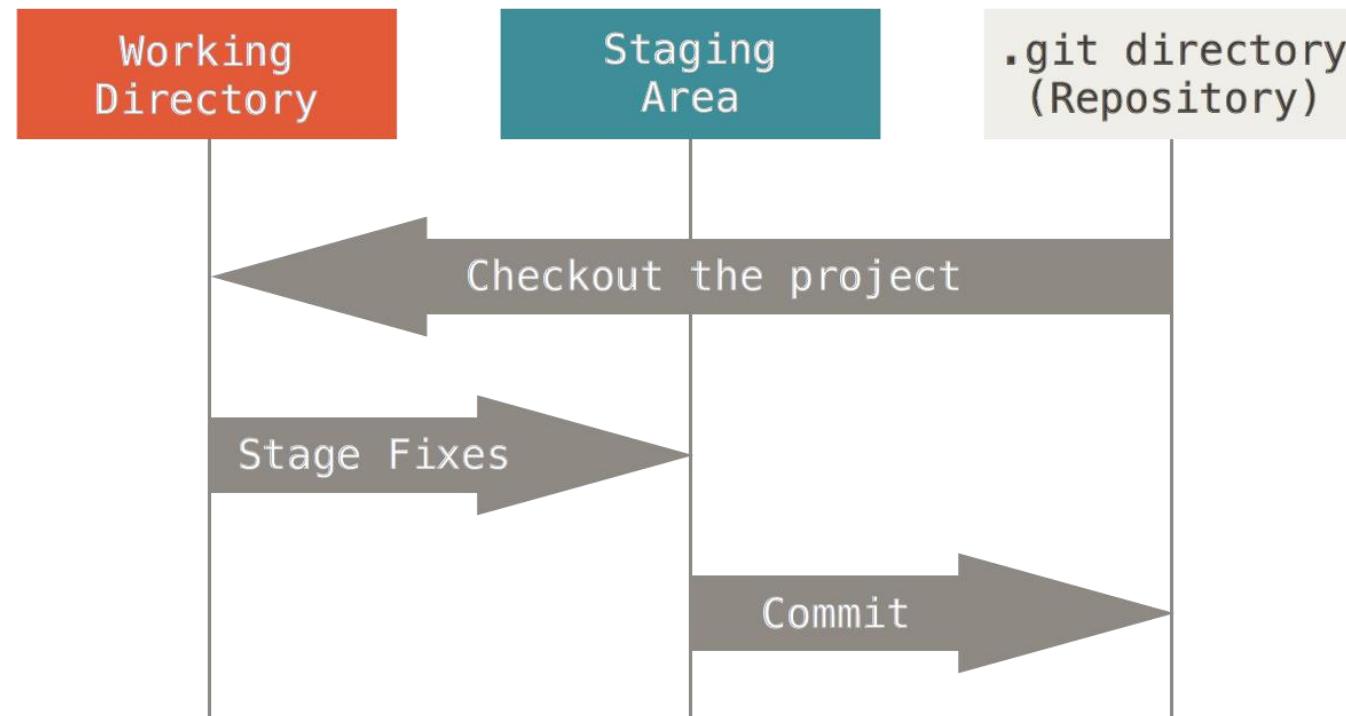
- **Elke commit reflecteert een werkend stuk code!**
- Maak een aparte commit voor elke verbetering, bug fix, ...
- Commit messages:
 - **de eerste lijn moet een samenvatting en beschrijving van je wijzigingen omvatten in < 80 karakters**
 - de tweede lijn (als die er is) moet leeg zijn
 - Vanaf de derde lijn kan je extra uitleg geven over wat je hebt gedaan en waarom

COMMENT	DATE
CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
ENABLED CONFIG FILE PARSING	9 HOURS AGO
MISC BUGFIXES	5 HOURS AGO
CODE ADDITIONS/EDITS	4 HOURS AGO
MORE CODE	4 HOURS AGO
HERE HAVE CODE	4 HOURS AGO
AAAAAAA	3 HOURS AGO
ADKFJSLKDFJSOKLFJ	3 HOURS AGO
MY HANDS ARE TYPING WORDS	2 HOURS AGO
HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

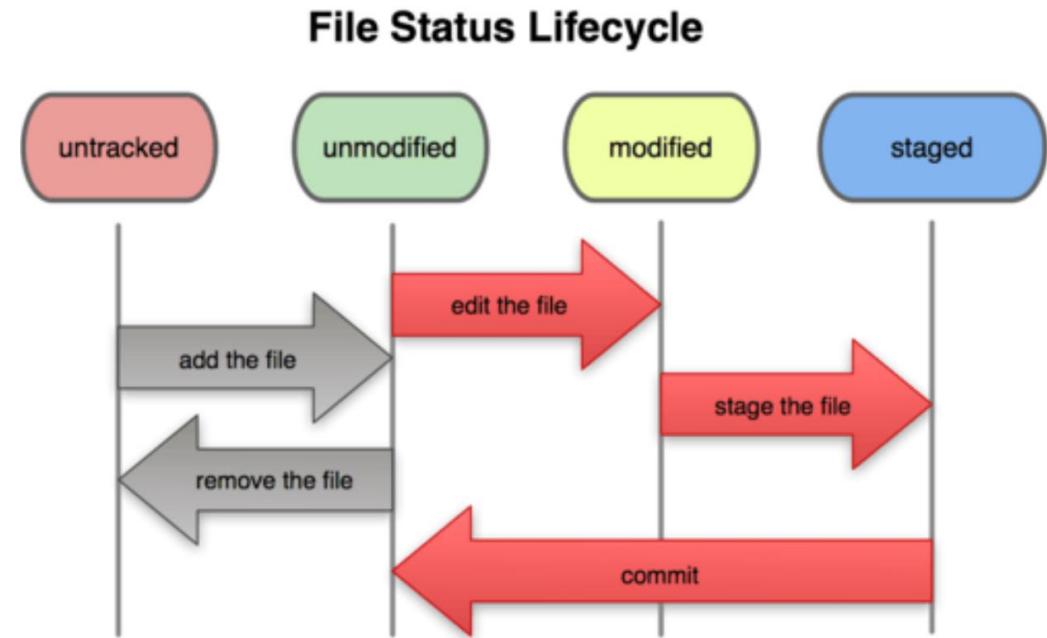
Git concepten: 3 niveau's

- Working directory: bevat **1 bepaalde versie** van je project waarop je werk verricht
- Staging: **klaarzetten van de wijzigingen** die je wil gaan “committen” naar de repository
- Repository: bevat **alle versies** van het project.



Git concepten: File status

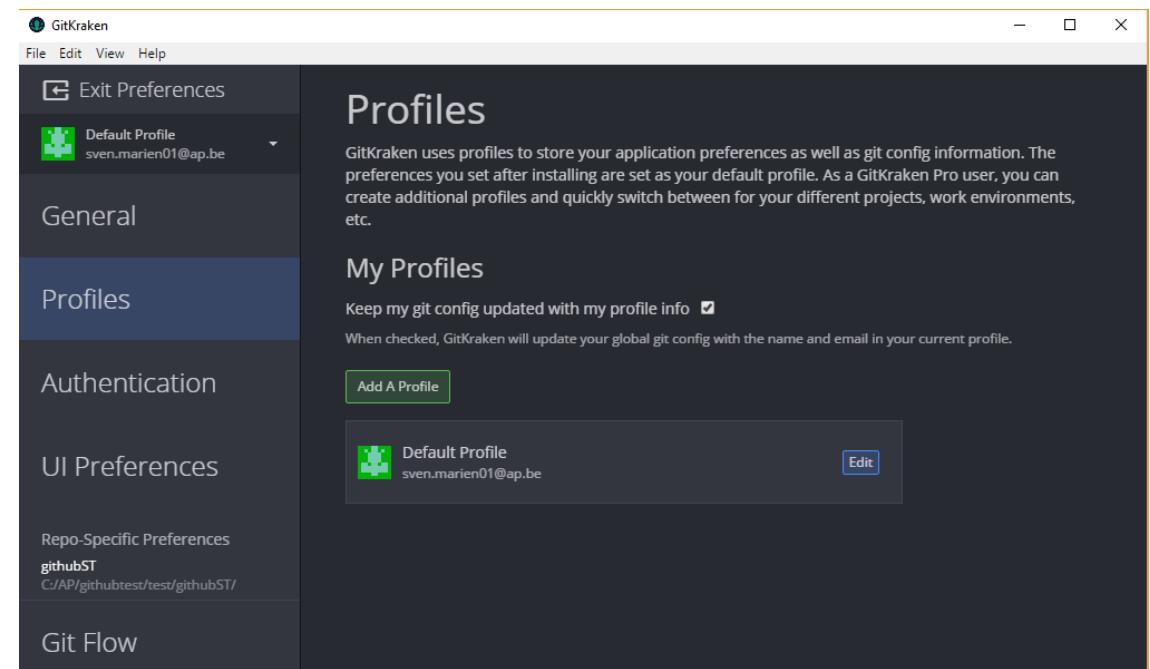
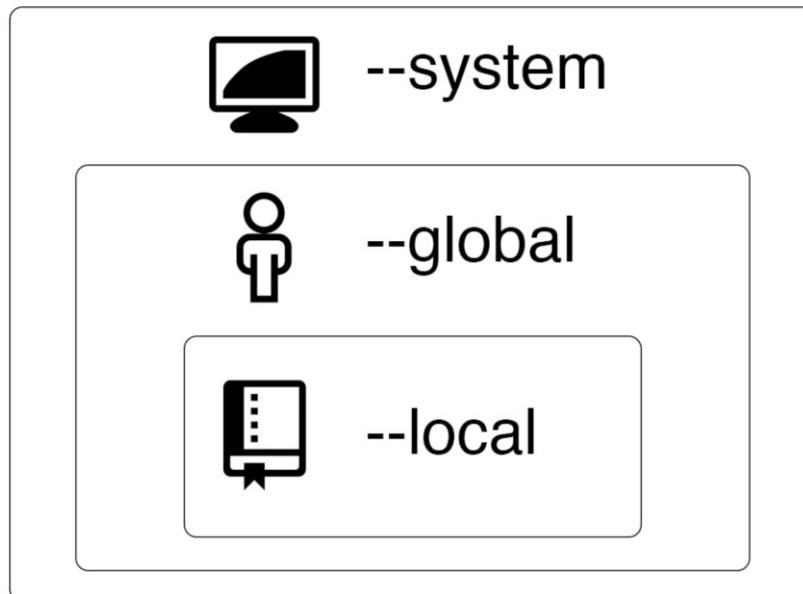
- **Untracked:** het bestand is (nog) niet onder controle van Git.
- **Unmodified:** het bestand is een exacte kopij van het dezelfde versie die zich in de repository bevindt.
- **Modified:** je hebt een bestand gewijzigd maar nog niet ge"commit"
- **Staged:** je hebt een aangepast bestand gemarkeerd om in zijn huidige toestand te worden meegenomen bij de volgende commit
- **Committed:** de aanpassingen zitten veilig gestockeerd in de repository en dus wordt de status terug ingesteld op 'unmodified'



Git configuratie:

- **git config**
- Configuratie op 3 niveau's
 - Stel gebruikersinfo in op: "global"
 - **git config --global**
 - Opvragen van de info
 - **git config --global --list**
 - Verwijderen
 - **git config--unset-all [user.email]**
- Gebruikersnaam: **user.name**
- E-mail: **user.email**

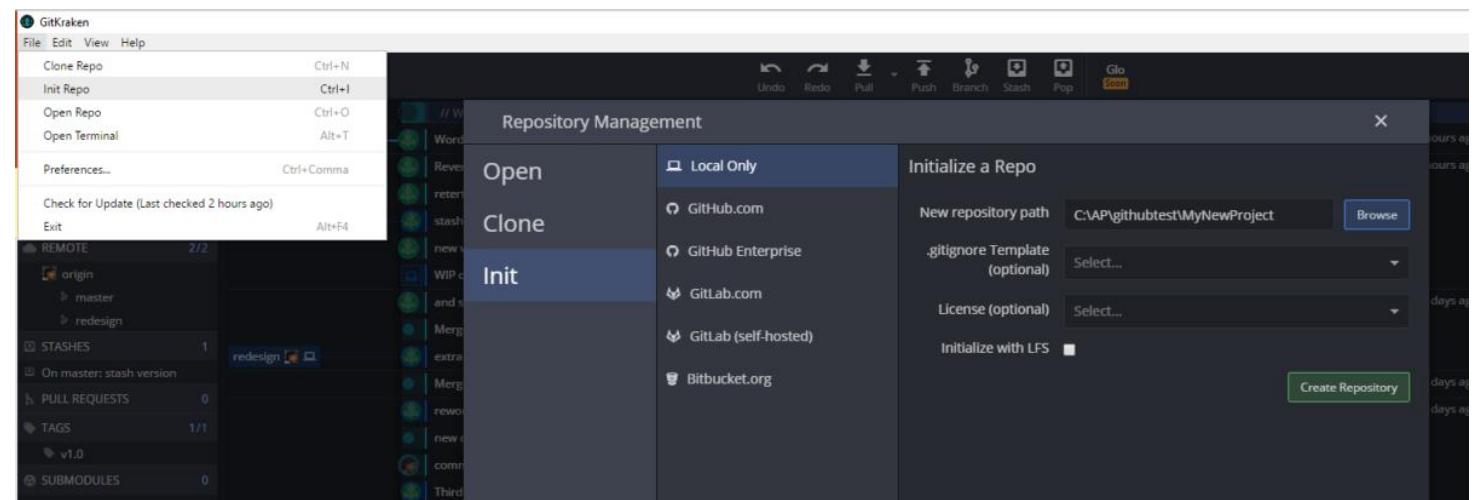
```
>git config --global user.name "Sven"
>git config --global user.email "sven.marien01@ap.be"
```



Nieuwe repository maken

- **git init**
- Een (verborgen) map voor de repository wordt aangemaakt : **.git**

```
C:\AP\githubtest>mkdir MyNewProject  
  
C:\AP\githubtest>cd MyNewProject  
  
C:\AP\githubtest\MyNewProject>git init  
Initialized empty Git repository in C:/AP/githubtest/MyNewProject/.git/
```



Repository status

- **git status**
- Toont bestanden die nog niet opgenomen zijn in Git, gewijzigde bestanden die nog niet gestaged zijn en gewijzigde bestanden die al gestaged zijn. Toont ook het verschil in staat tussen de huidige repository en de remote repository.

```
C:\AP\githubtest\MyNewProject>git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Bestand toevoegen aan git

- **git add [naam]**
- **git add .** => alle bestanden toevoegen
- Voeg bestanden die **nog niet in git staan of gewijzigde bestanden** toe aan de staging area, maar commit de wijzigingen nog niet.

```
C:\AP\githubtest\SoftwareEngineering>git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   main.js

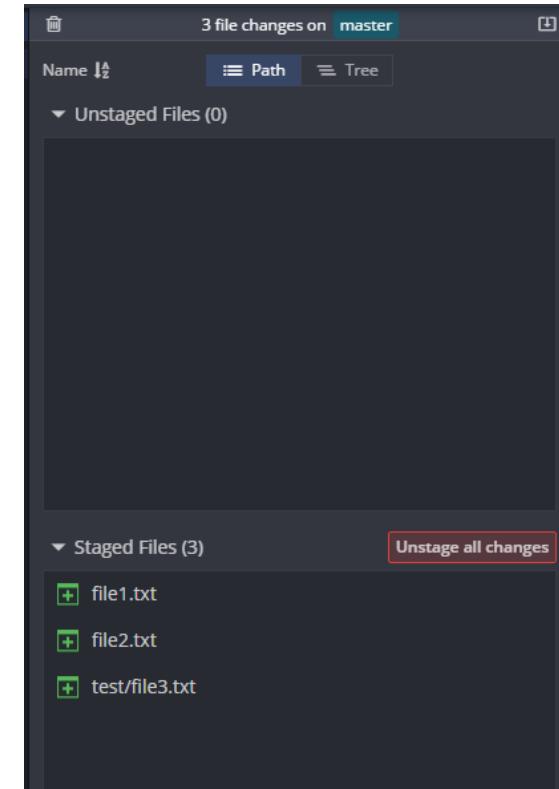
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    helper.js

no changes added to commit (use "git add" and/or "git commit -a")

C:\AP\githubtest\SoftwareEngineering>git add .
C:\AP\githubtest\SoftwareEngineering>git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

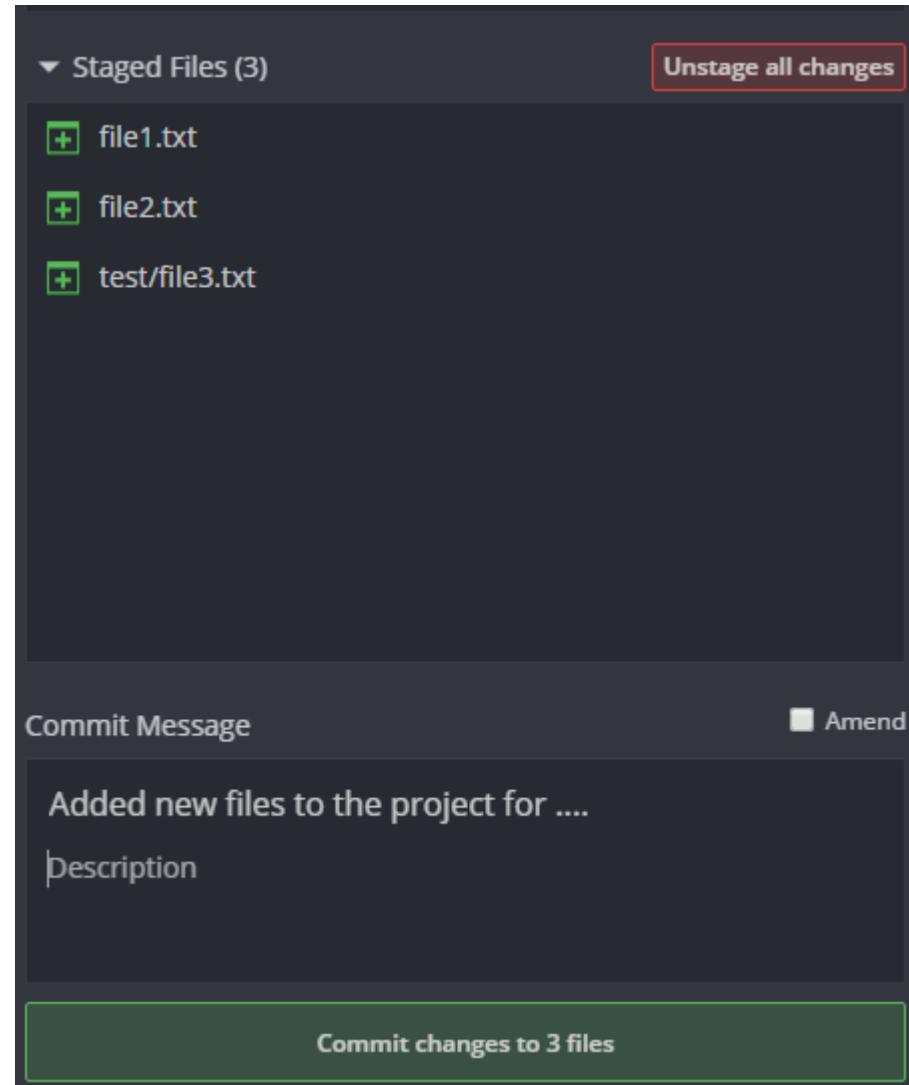
    new file:   helper.js
    modified:   main.js
```



“Commit” uitvoeren

- **git commit**
- **git commit -m “[beschrijving]”**
- Voert alle wijzigingen in de staging area door in een commit met een optionele (maar ten sterkst aan te raden) beschrijving.

```
C:\AP\githubtest\test3>git commit -m "Added new files to the project for ..."
[master 1c1e0aa] Added new files to the project for ...
 3 files changed, 3 insertions(+)
 create mode 100644 file1.txt
 create mode 100644 file2.txt
 create mode 100644 test/file3.txt
```



Overzicht van de “commit log”

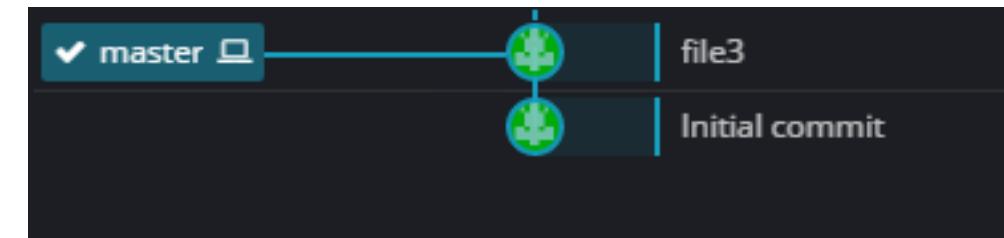
- **git log**
- Toon alle commits die gebeurd zijn voor de huidige HEAD (HEAD is een pointer naar de huidige staat van de snapshot. Meestal zal de HEAD gewoon naar de branch wijzen).

```
C:\AP\githubtest\test3>git log
commit 9631ae832a9562767dd4e67b39880c2d665879e3 (HEAD -> master)
Author: Unknown <sven.marien01@ap.be>
Date:   Tue Feb 13 13:33:30 2018 +0100

    file3

commit d18e15943bf5e4e31a8d5fce80481c745c4666f
Author: Unknown <sven.marien01@ap.be>
Date:   Mon Feb 12 22:52:23 2018 +0100

    Initial commit
```



“Undo staging”

- **git reset**
- Zet de veranderingen die klaar staan in de staging area terug bij de ‘unstaged files’.

```
C:\AP\githubtest\test3>git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

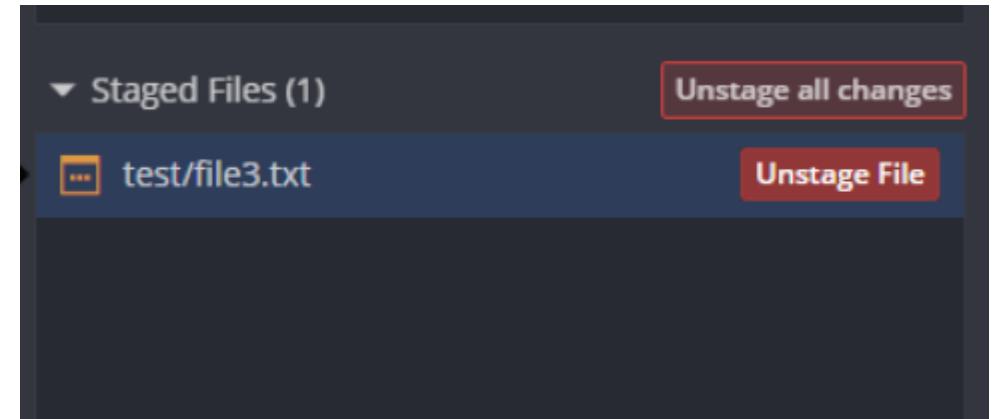
    modified:   test/file3.txt

C:\AP\githubtest\test3>git reset
Unstaged changes after reset:
M       test/file3.txt

C:\AP\githubtest\test3>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   test/file3.txt

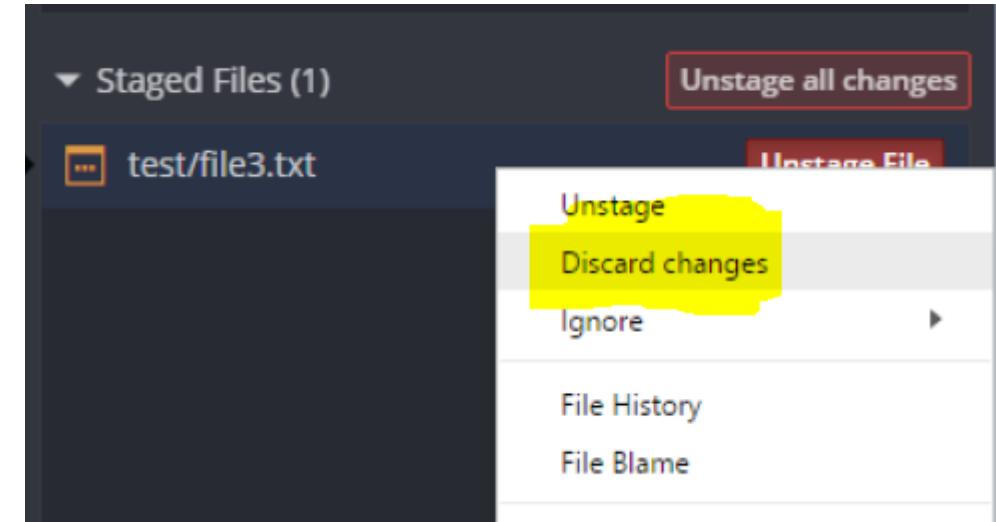
no changes added to commit (use "git add" and/or "git commit -a")
```



“Undo” van alle laatste wijzigingen

- **git reset –hard**
- Reset alle getrack'te bestanden terug naar de staat van de commit.

```
C:\AP\githubtest\test3>git reset --hard  
HEAD is now at 9631ae8 file3
```



Details weergeven

- **git show [CommitID]**
- Toon de details van een specifieke commit.

```
C:\AP\githubtest\test3>git log
commit 76bf6d679d51ecc939ae5bd00a636eb97ccce4bf (HEAD -> master)
Author: Sven <sven.marien01@ap.be>
Date:   Tue Feb 13 14:19:36 2018 +0100

    tekst toegevoegd

commit 9631ae832a9562767dd4e67b39880c2d665879e3
Author: Unknown <sven.marien01@ap.be>
Date:   Tue Feb 13 13:33:30 2018 +0100

    file3

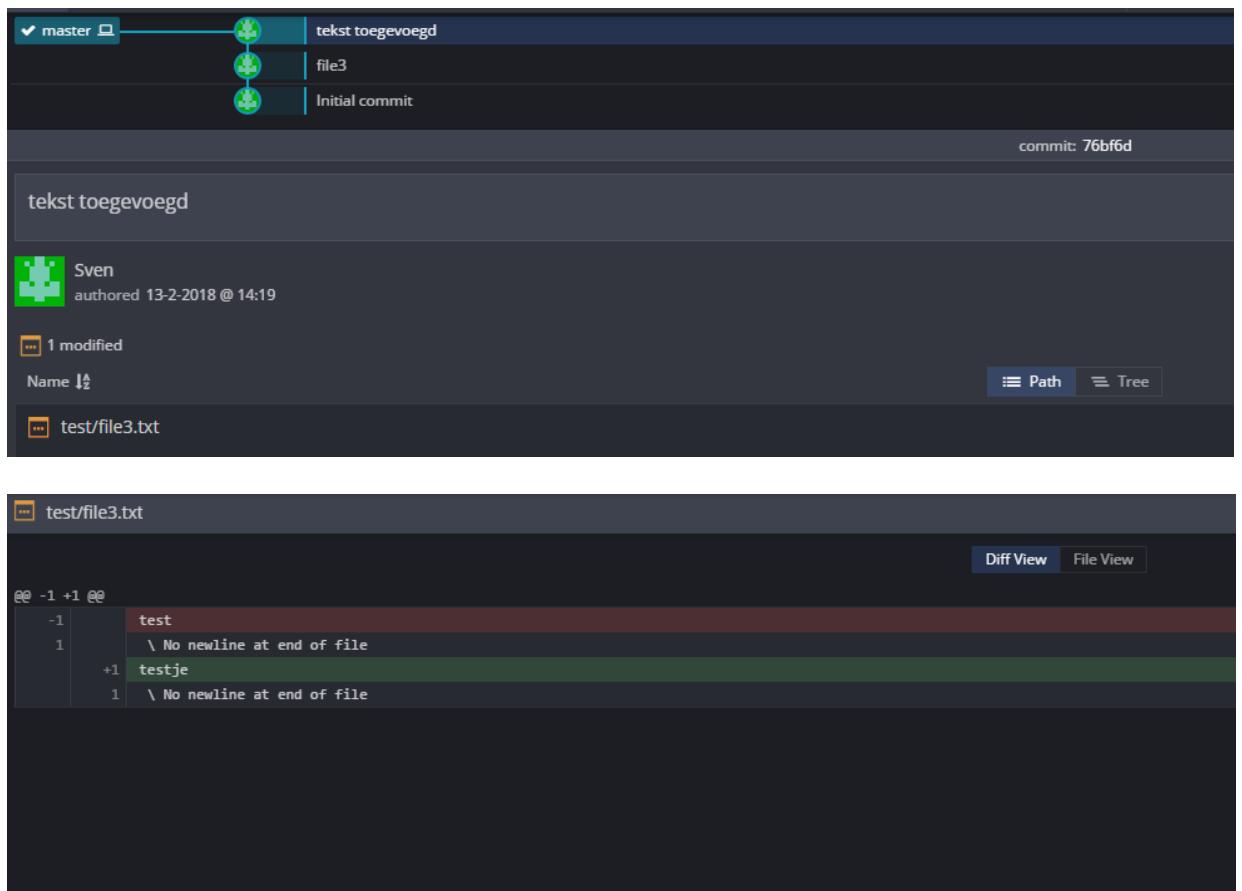
commit d18e15943bf5e4e31a8d5dfce80481c745c4666f
Author: Unknown <sven.marien01@ap.be>
Date:   Mon Feb 12 22:52:23 2018 +0100

    Initial commit

C:\AP\githubtest\test3>git show 76bf6d
commit 76bf6d679d51ecc939ae5bd00a636eb97ccce4bf (HEAD -> master)
Author: Sven <sven.marien01@ap.be>
Date:   Tue Feb 13 14:19:36 2018 +0100

    tekst toegevoegd

diff --git a/test/file3.txt b/test/file3.txt
index 30d74d2..7eb9cc7 100644
--- a/test/file3.txt
+++ b/test/file3.txt
@@ -1 +1 @@
-test
\ No newline at end of file
+testje
\ No newline at end of file
```



Niet alles moet in git...

- Bestanden die specifiek zijn aan een gebruiker of een lokale opstelling (e.g. IDE configuratie)
- Bestanden die automatisch gegenereerd worden bij o.a. het compileren van code, het installeren van dependencies, enzovoorts.
- Binaire bestanden (tricky, uitzondering is mogelijk als je er zeker van bent dat slechts 1 persoon dit bestand tegelijkertijd aanpast)

=> **.gitignore**

Gitignore stappen

1. Maak .gitignore aan of open het
2. Creëer reguliere expressie met pad naar de code die niet in Git hoort (files, directories, extensies of combinatie)
3. Check met 'git status' of die code nu niet meer bij de wijzigingen staat.
4. Optioneel: pas README.md aan om de stappen te beschrijven die nodig zijn om bestanden te genereren, e.g. 'do npm install'.
5. Add en commit .gitignore en eventueel relevante andere wijzigingen.

Git Junior (local & remote)

Elektronica – ICT

Sven Mariën

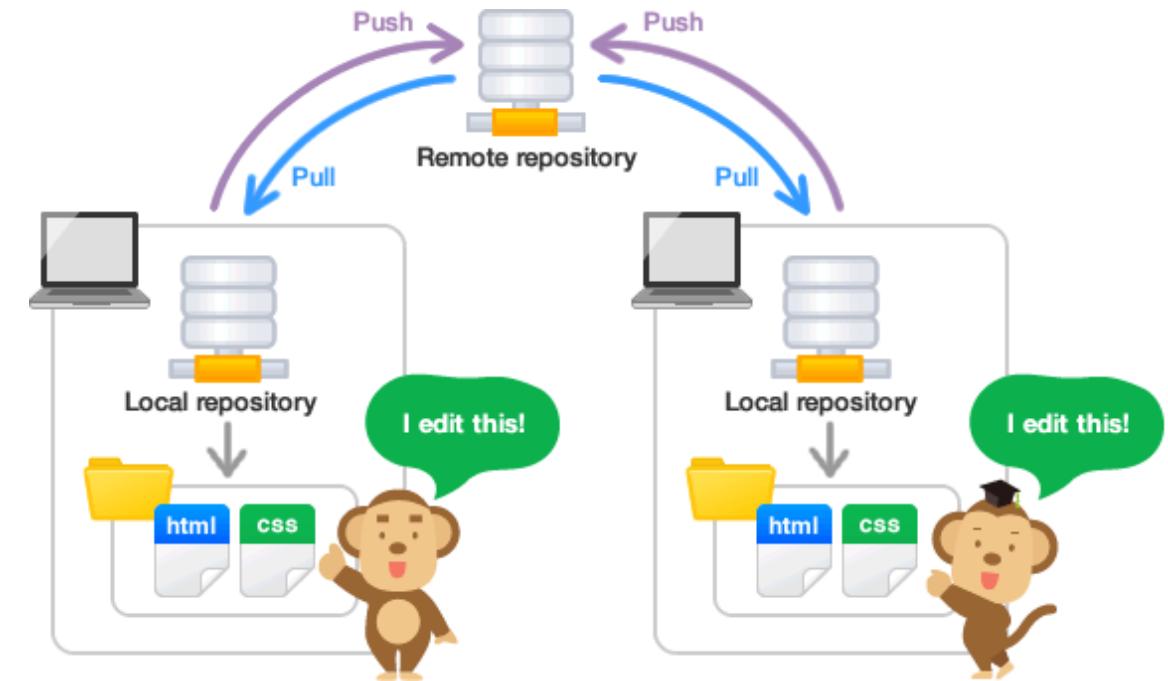
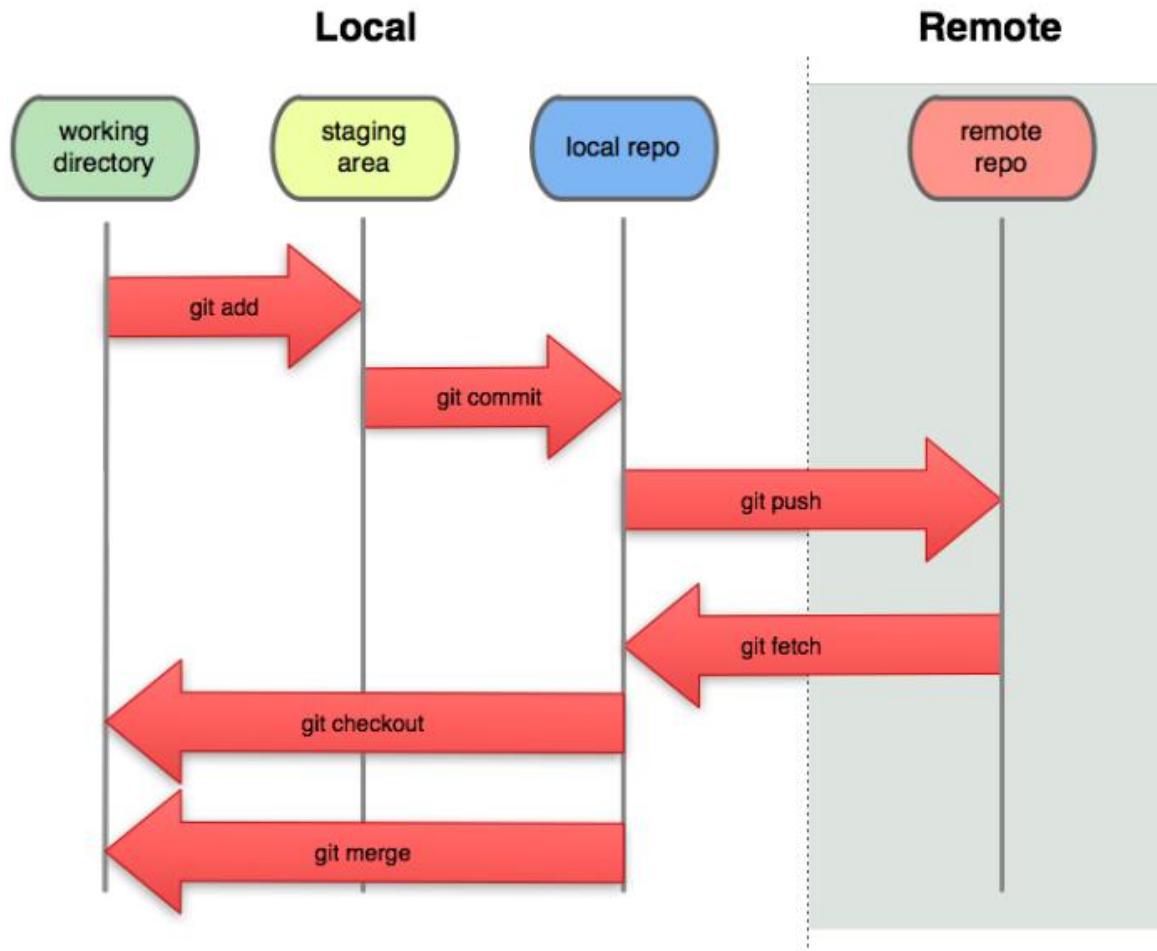
(sven.marien01@ap.be)

2017-2018



ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

Local + Remote repository



Git hosting service

- Een remote Git repository kan worden ondergebracht bij een hosting service.
- <https://www.git-tower.com/blog/git-hosting-services-compared/>
- Deze bestaan gratis en betalend: <http://comparegithosting.com/>

Site	Price	# of Repos	Type	Users	Storage	Defects/ Issues	Agile/ Tasks	Wiki	Other SCM	Deploy
GitHub	\$0	Unlimited	Public Only	Unlimited	1GB^	Yes	-	Yes	-	Compare
 GitLab	\$0	Unlimited	Public/ Private	Unlimited	10GB/ Project	Yes	-	Yes	-	GitLab CI Compare
 DEVEO	\$0	Unlimited	Private Only	Unlimited	1GB	Yes	-	Yes	SVN, Mercurial	- Compare
 Atlassian Bitbucket	\$0	Unlimited	Public/ Private	5	1GB^	Yes	-	Yes	Mercurial	Pipelines (BETA) Compare
 Visual Studio	\$0	Unlimited	Private Only	5	Unlimited	Yes	Yes	Yes	TFVC	CI Compare
 amazon webservices™	\$0	Unlimited	Private Only	5	50GB	-	-	-	-	CodePipeline Compare

[^] per repository is recommended

Samenwerking in teamverband

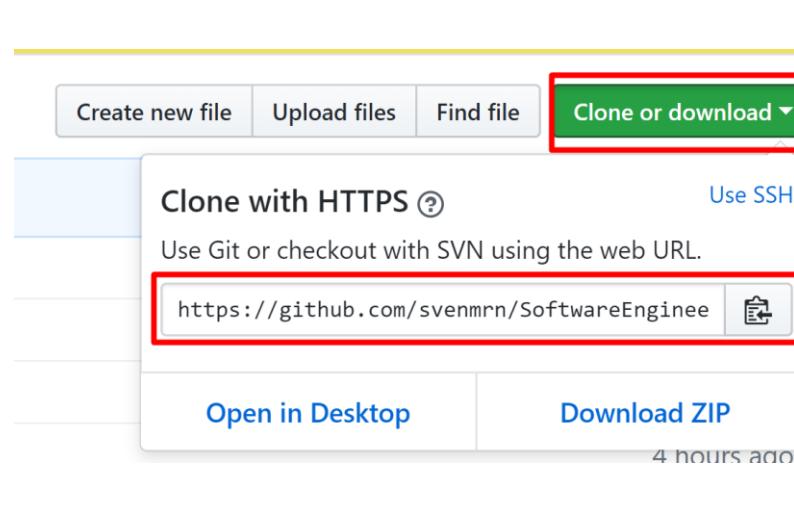
- Werk per 2
- Iemand van het team maakt op github een repo aan.
- Zorg dat je collega hierop “schrijf” toegang heeft.

The screenshot shows a GitHub repository page for 'svenmrn / SoftwareEngineering'. At the top, there are buttons for 'Watch' (0), 'Star' (0), and 'Fork' (0). Below these are links for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', and 'Insights'. A red box highlights the 'Settings' button, which is also highlighted with an orange bar. On the left, a sidebar has 'Options' at the top, followed by 'Collaborators' (which is highlighted with a red box), 'Branches', 'Webhooks', 'Integrations & services', and 'Deploy keys'. The main content area is titled 'Collaborators' and contains the message: 'This repository doesn't have any collaborators yet. Use the form below to add a collaborator.' It includes a search bar labeled 'Search by username, full name or email address' and a note: 'You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.' A large red box surrounds the search bar and the 'Add collaborator' button.

Bestaande repository downloaden

- **git clone [url]**
- Maakt een nieuwe map aan en haalt de volledige repository over + aanmaak working directory.

```
C:\AP\githubtest>git clone https://github.com/botpress/botpress.git
Cloning into 'botpress'...
remote: Counting objects: 15869, done.
remote: Compressing objects: 100% (58/58), done.
remote: Total 15869 (delta 44), reused 63 (delta 26), pack-reused 15781
Receiving objects: 100% (15869/15869), 10.77 MiB | 242.00 KiB/s, done.
Resolving deltas: 100% (10334/10334), done.
```



The screenshot shows two side-by-side interfaces for cloning a GitHub repository. On the left is the GitHub web interface for the repository 'svenmrn/SoftwareEngineer'. It features a 'Clone or download' button highlighted with a red box. Below it is a 'Clone with HTTPS' field containing the URL <https://github.com/svenmrn/SoftwareEngineer>, also highlighted with a red box. Other options shown include 'Open in Desktop' and 'Download ZIP'. On the right is a 'Repository Management' dialog box. It has tabs for 'Open', 'Clone', and 'Init', with 'Clone' currently selected. Under 'Clone', there's a 'Clone with URL' section where 'GitHub.com' is selected. The 'Where to clone to' field contains the path `c:\ap\githubtest\`. The 'URL' field contains the same GitHub URL as the web interface. The 'Full path' field shows the complete local path `c:\ap\githubtest\botpress`. A large green 'Clone the repo!' button is at the bottom right of the dialog.

Lokale “commits” doorsturen naar “remote” repository

- **git push**

- Wijzigingen in je lokale repository (die nog niet in de remote repo zitten) aan de remote repository toevoegen. De lokale repo zal dan “commits” voorlopen op de remote repo.

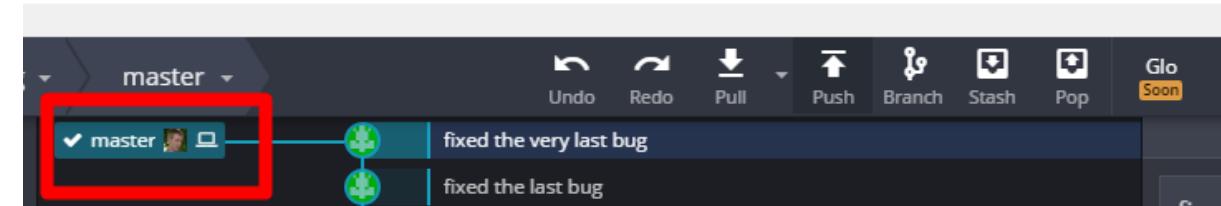
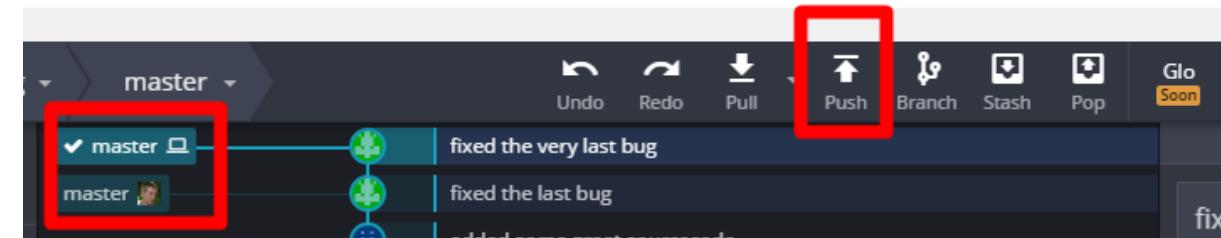
```
C:\AP\githubtest\SoftwareEngineering>git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\AP\githubtest\SoftwareEngineering>git push
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 332 bytes | 332.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/svenmrn/SoftwareEngineering.git
  b0b29e8..f644a0c  master -> master

C:\AP\githubtest\SoftwareEngineering>git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```



Controleren op “remote” wijzigingen

- **git fetch**
- Controleren of er wijzigingen zijn in de remote repository (synchroniseert je lokale branch nog niet)

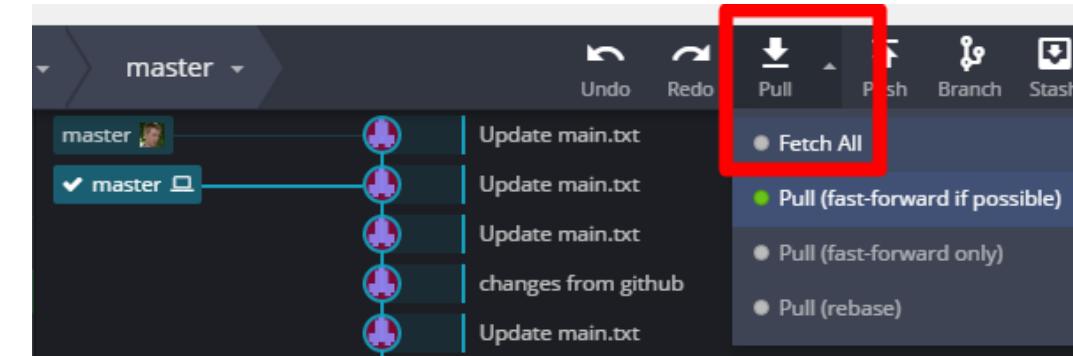
```
C:\AP\githubtest\SoftwareEngineering>git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

C:\AP\githubtest\SoftwareEngineering git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/svenmrn/SoftwareEngineering
  3379aa0..d2df886  master      -> origin/master

C:\AP\githubtest\SoftwareEngineering>git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean
```



Ophalen van de nieuwere remote “commits”

- **git pull**
- Synchroniseert je lokale repository met de laatste wijzigingen in de remote repository.

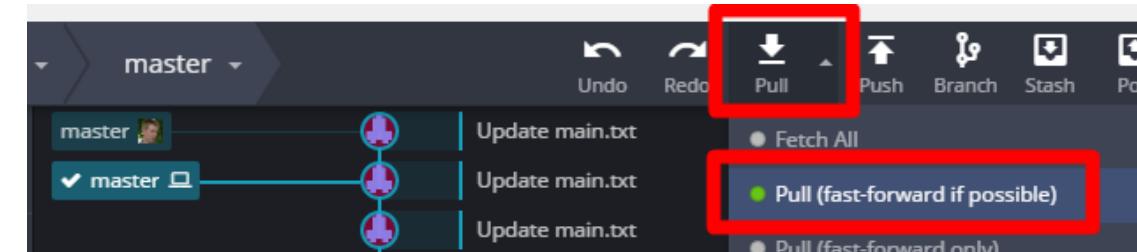
```
C:\AP\githubtest\SoftwareEngineering>git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean

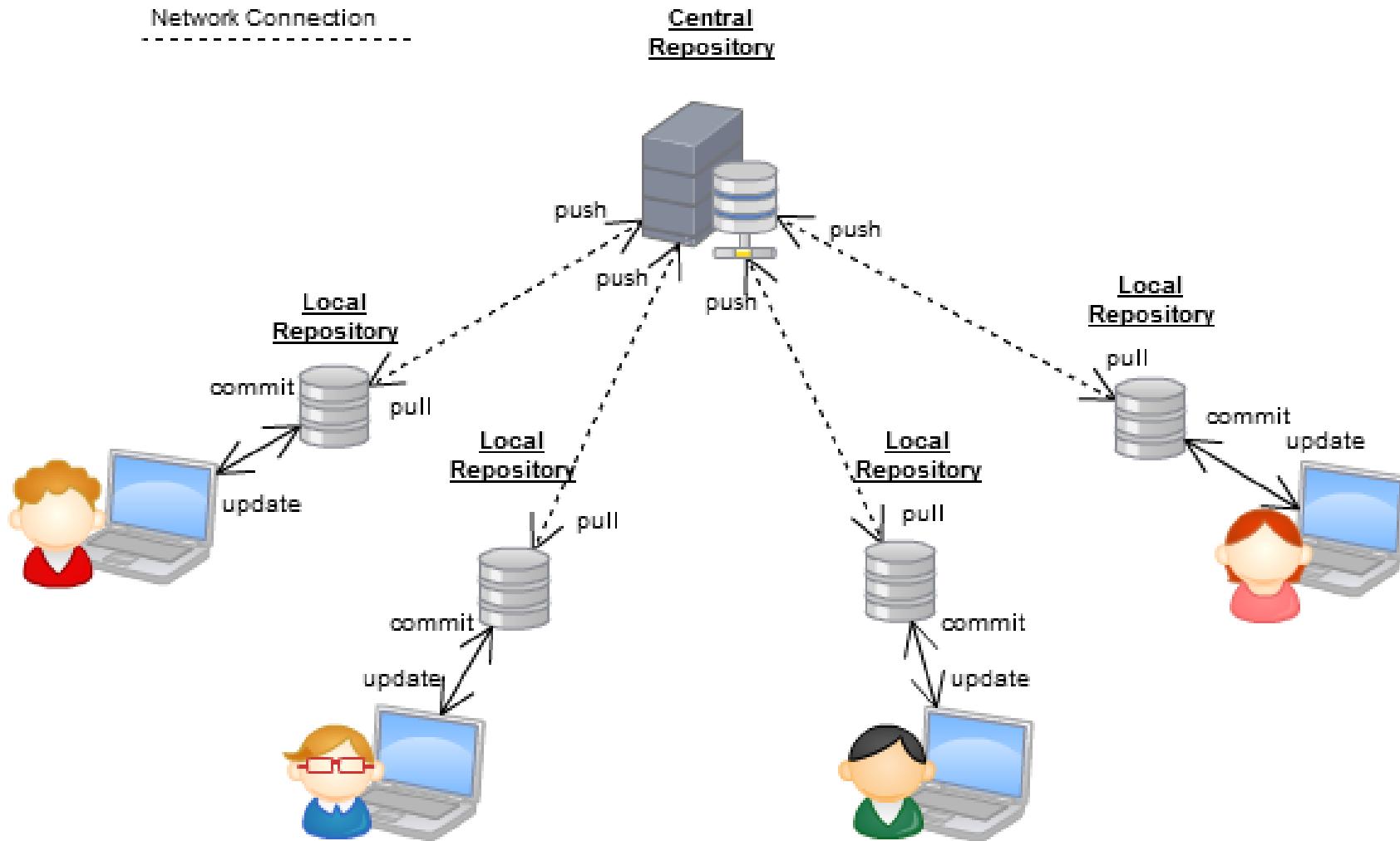
C:\AP\githubtest\SoftwareEngineering>git pull
Updating 3379aa0..d2df886
Fast-forward
  main.txt | 2 +-
  1 file changed, 1 insertion(+), 1 deletion(-)

C:\AP\githubtest\SoftwareEngineering>git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```



Samenwerken aan een project



Scenario's

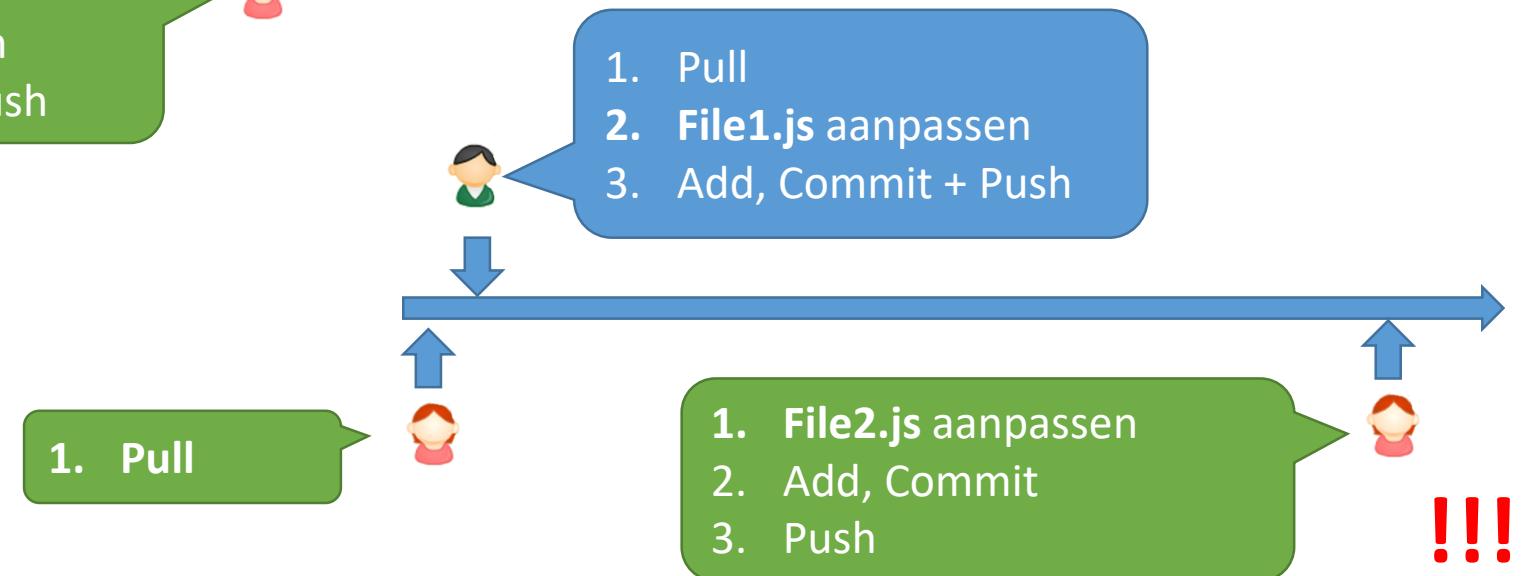
- Robin en June werken op hetzelfde project en nemen ieder story's op, hiervoor moeten ze elk bepaalde bestanden aanpassen.
- Robin werkt bijvoorbeeld aan **file1.js** en June aan **file2.js**
- Er zijn nu verschillende scenario's die kunnen voorkomen.
- Het zou eveneens kunnen dat ze voor hun story in hetzelfde broncode bestand aanpassingen moeten uitvoeren....

Scenario 1: ieder werkt aan een ander bestand

1a: Doe steeds een 'pull' voor je start, persoon 2 start pas nadat persoon 1 klaar is

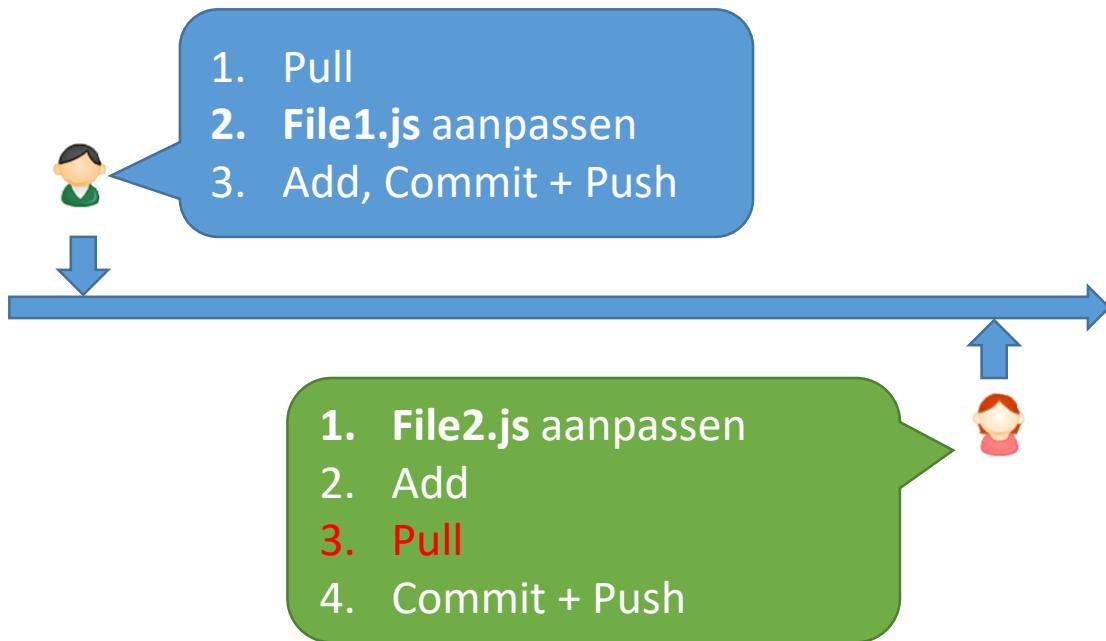


1b: je vergeet een 'pull' te doen voor je start, of ondertussen heeft iemand anders 'commit'(s) ge'pushed'

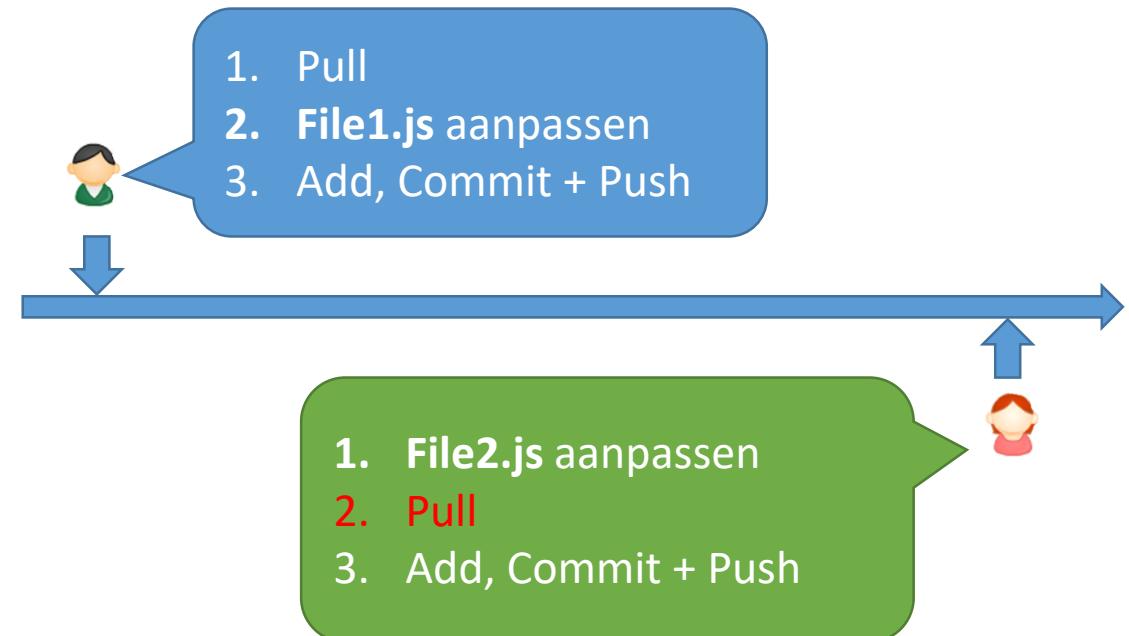


Scenario 1: ieder werkt aan een ander bestand

1c: Moet je eerst een “commit” doen voor een “pull” ?



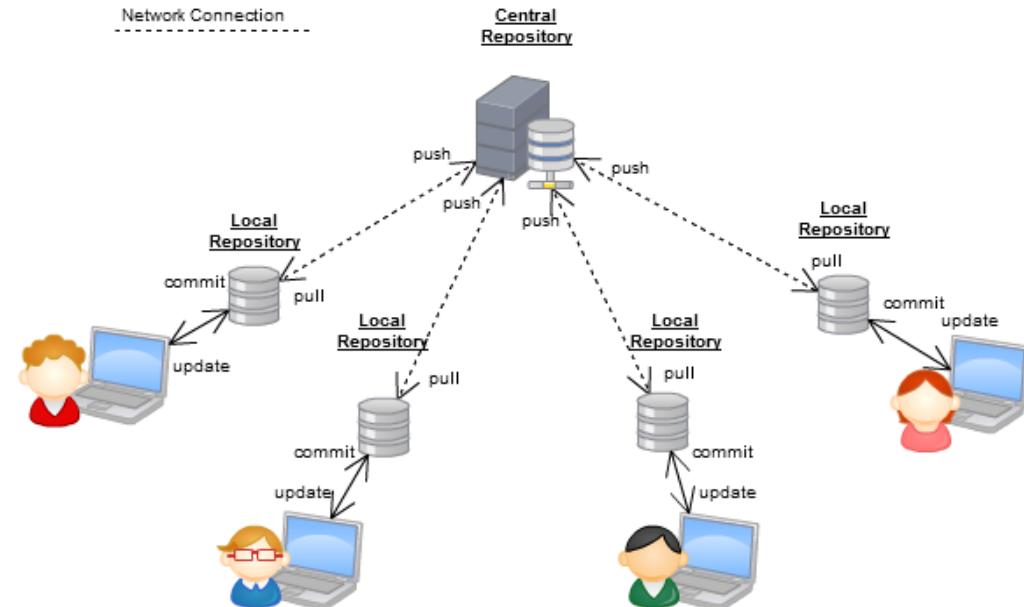
1d: Moet je eerst een “add” doen voor een “pull” ?



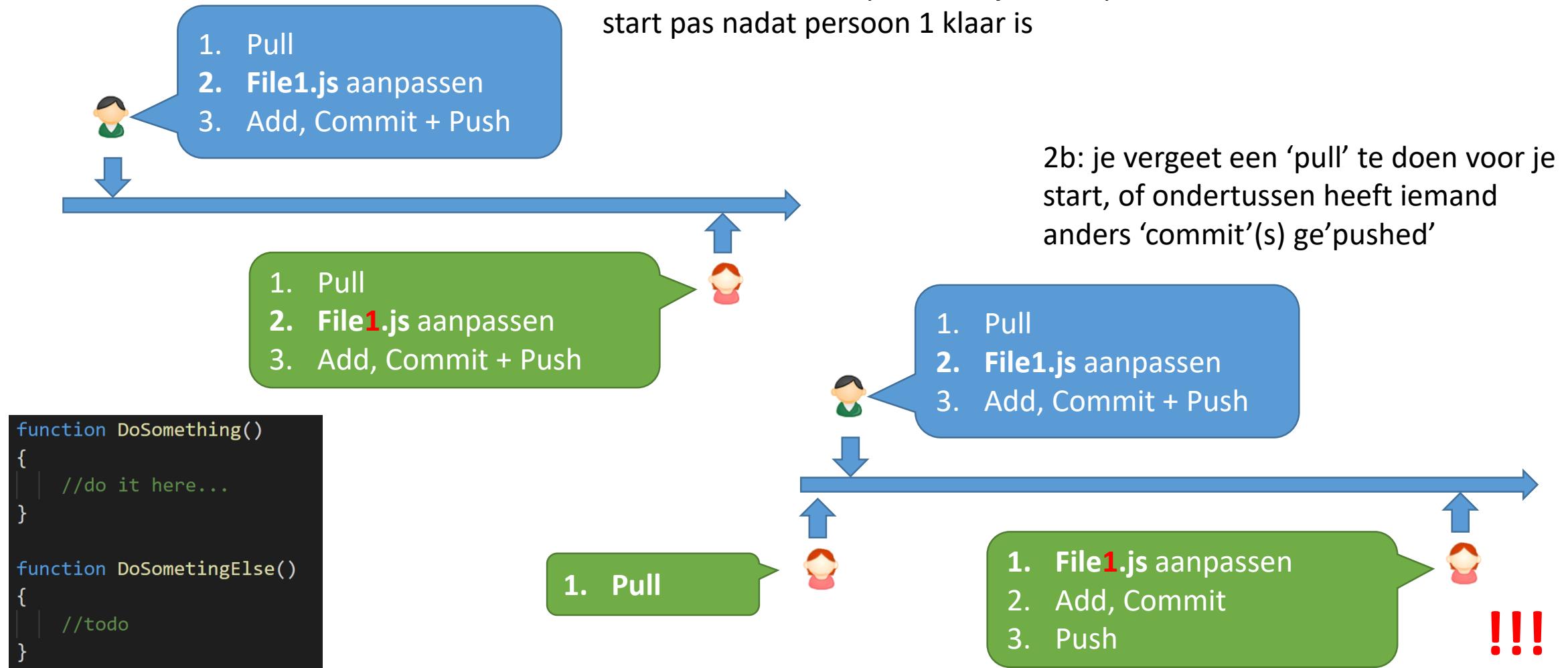
Scenario1: ieder werkt aan ander bestand

Samenvatting:

- Pull: Zolang je niet aan hetzelfde bestand(en) werkt kan je op eender welk moment een ‘pull’ doen om de wijzigingen (commits) van je andere teamgenoten te krijgen
- Push: Je kan enkel een ‘push’ doen van je eigen commit(s) nadat je eerst alle andere nieuwe commits hebt opgehaald (pull)

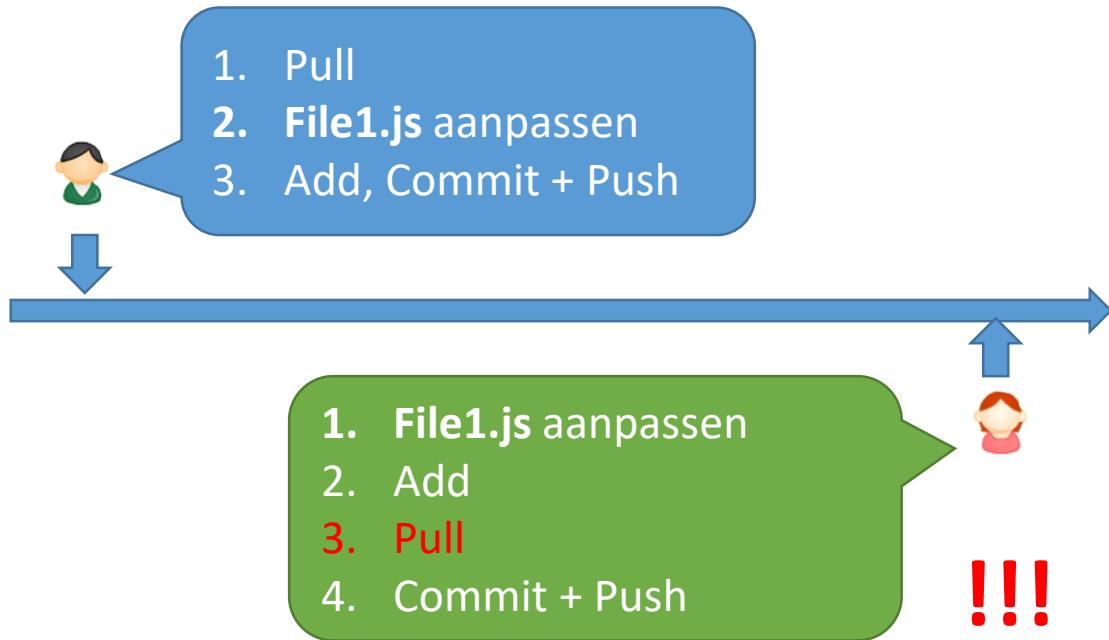


Scenario 2: werken aan hetzelfde bestand maar in een andere functie

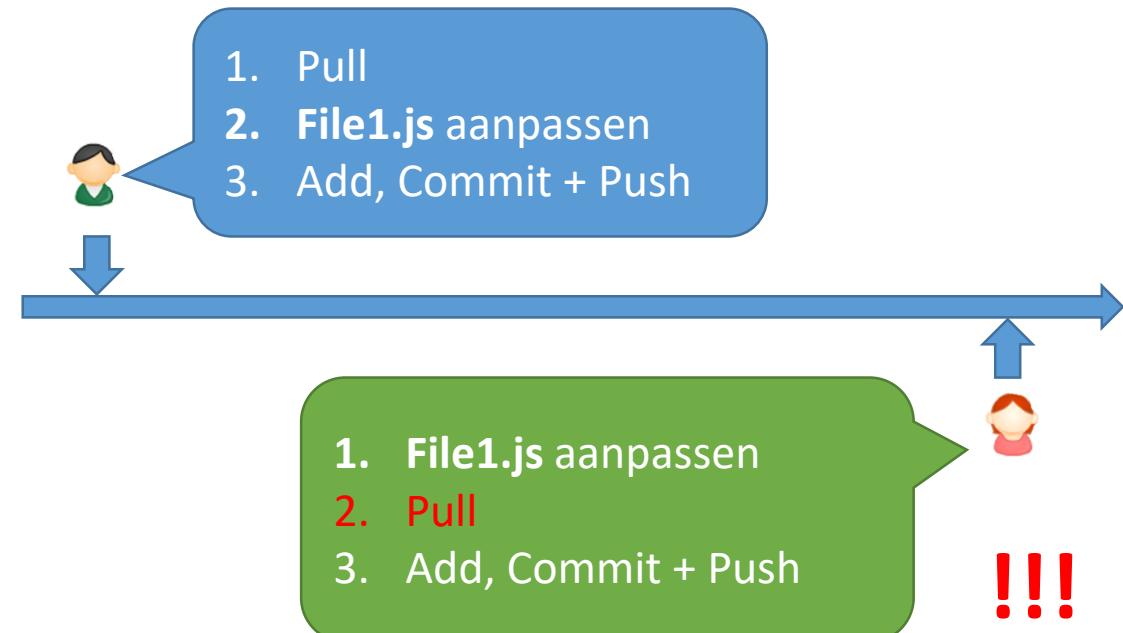


Scenario 2: werken aan hetzelfde bestand maar in een andere functie

2c: Moet je eerst een “commit” doen voor een “pull” ?



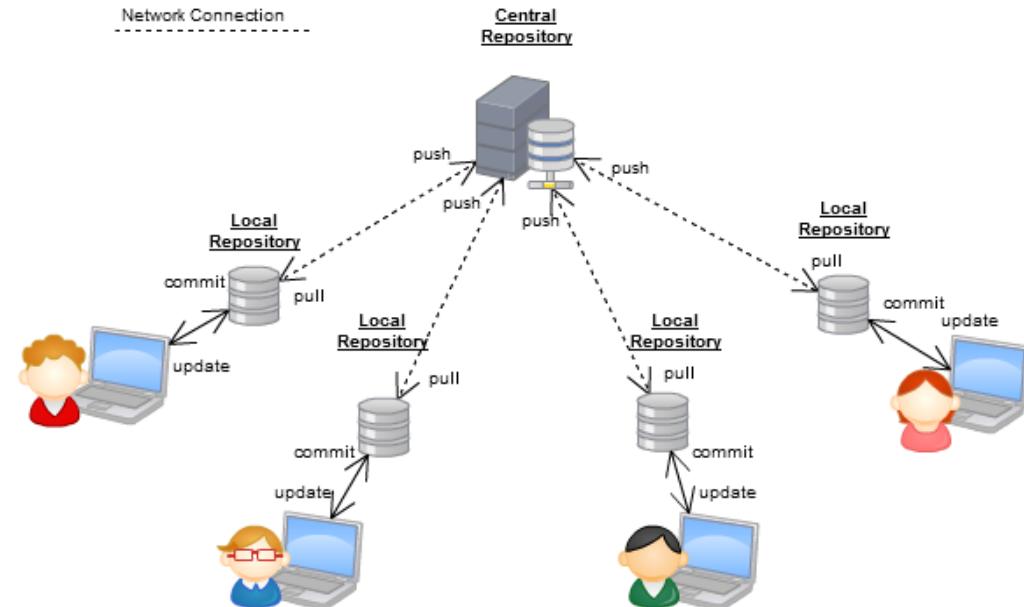
2d: Moet je eerst een “add” doen voor een “pull” ?



Scenario2: werken aan hetzelfde bestand maar in een andere functie/regio

Samenvatting:

- Pull: Wanneer je aan hetzelfde bestand(en) werkt kan je **ENKEL** een ‘**pull**’ doen nadat je je eigen wijzigingen hebt ge’**commit**’. Bij de pull zal er een ‘**merge**’ gebeuren waarbij de beide wijzigingen automatisch worden samengevoegd door git.
- Push: Je kan enkel een ‘**push**’ doen van je eigen commit(s) nadat je eerst alle andere nieuwe commits hebt opgehaald (pull)



Git Medior (merge conflicten)

Elektronica – ICT

Sven Mariën

(sven.marien01@ap.be)

2017-2018



ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

Scenario 3: werken aan hetzelfde bestand in dezelfde code

- 
- 1. Pull
 - 2. **File1.js** aanpassen
 - 3. Add, Commit + Push

3a: Doe steeds een ‘pull’ voor je start, persoon 2 start pas nadat persoon 1 klaar is

- 
- 1. Pull
 - 2. **File1.js** aanpassen
 - 3. Add, Commit + Push

3b: je vergeet een ‘pull’ te doen voor je start, of ondertussen heeft iemand anders ‘commit’(s) ge’pushed’

- 
- 1. Pull
 - 2. **File1.js** aanpassen
 - 3. Add, Commit + Push

```
function DoSomething()  
{  
    let number = 5;  
}
```

- 
- 1. Pull

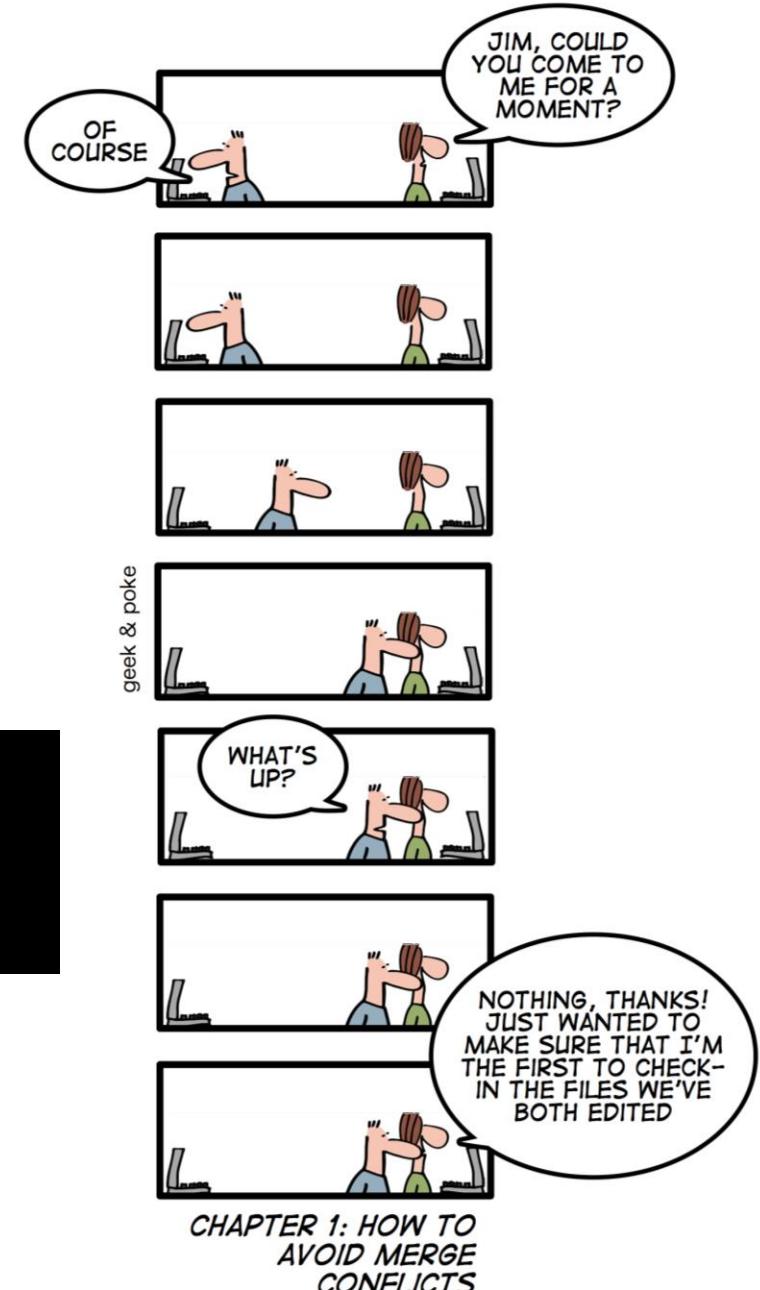
- 
- 1. **File1.js** aanpassen
 - 2. Add, Commit
 - 3. Push

!!!

Wat zijn “merge” conflicten...

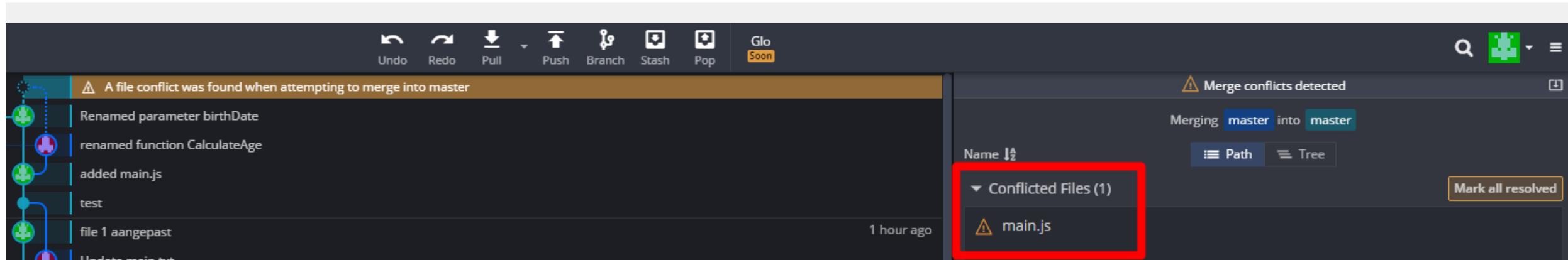
- Meerdere personen hebben aanpassingen gedaan aan dezelfde lijnen code of tekst
- Git kan zelf niet beslissen welke wijzigingen de “juiste” zijn.
- De “merge” kan dus **niet automatisch** gebeuren => **manuele ingreep** nodig.
- Git geeft wel aan waar er “merge conflicten” zijn.
- De ontwikkelaar die laatst is zal de “merge” moeten uitvoeren 😞
- Uiteraard kunnen er conflicten in 1 of meerdere bestanden voorkomen.

```
C:\AP\githubtest\SoftwareEngineering>git pull
Auto-merging main.txt
CONFLICT (content): Merge conflict in main.txt
Automatic merge failed; fix conflicts and then commit the result.
```



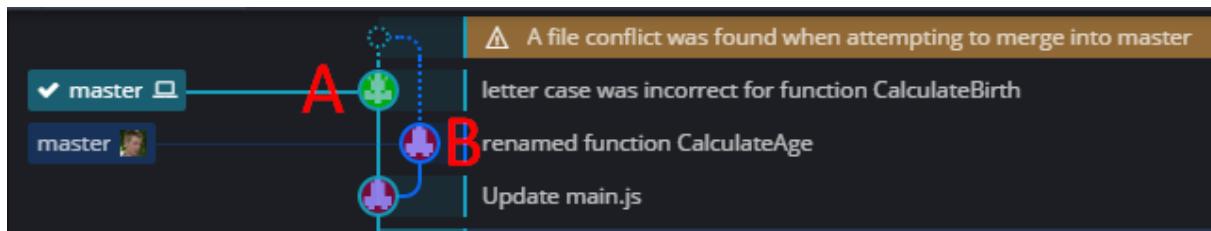
Oplossen van conflicten (via Gitkraken)

- **Stap 1:** in welk(e) bestand(en) doet er zich een conflict voor ?



Oplossen van conflicten (via Gitkraken)

- **Stap 2:** vergelijk de wijzigingen voor elk bestand
 - Git zal in de bestanden een indicatie geven van de conflicten
 - <<<<< HEAD => jouw aanpassing
 - ======> aanpassing door collega
 - >>>>> => CommitID
- Eenvoudiger via een mergetool (bv. Gitkraken)



```

main.js - Kladblok
Bestand Bewerken Opmak Beeld Help
<<<<< HEAD
function CalculateAge(birthdate) ←
=====
function CalculateAgeFromBirthDate(birthdate) ←
>>>>> 6e96f807bd6ccbe01cf4168f7c6498d707610bf4
{
    //....TODO
}

GitKraken
File Edit View Help
SoftwareEngineering master Undo Redo Pull Push Branch Stash Pop Glo Soon
Open in external merge tool Save
main.js (1 conflict)
[A] Commit e3dfbe on
[B] Commit on
function CalculateAge(birthdate) ←
=====
function CalculateAgeFromBirthDate(birthdate) ←
conflict 1 of 1
1 !function Calculateage(birthdate)
2 f
3 
4 //....TODO
5 }
6

```

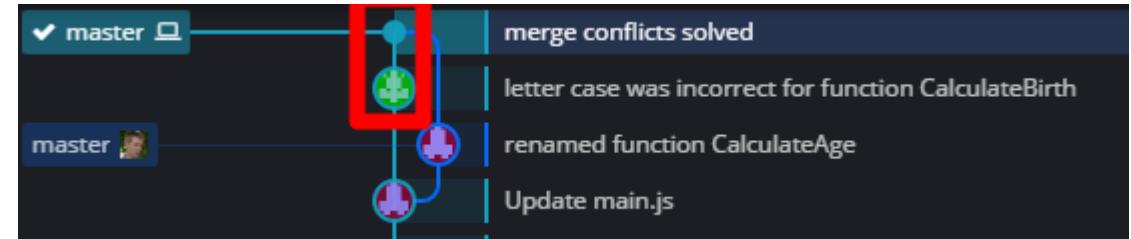
Oplossen van conflicten (via GitKraken)

- **Stap 3:** los het conflict op.
 - Ofwel kies versie A
 - Ofwel kies versie B
 - Ofwel kies beiden
 - Ofwel via een manuele ingreep
- Manuele aanpassing niet mogelijk vanuit GitKraken Free version
 - Ofwel via code editor na “save”
 - Ofwel via externe merge tool (Kdiff, P4Merge, ...)

```
GitKraken
File Edit View Help
SoftwareEngineering master Undo Redo Pull Push Branch Stash Pop Glo Soon
main.js (1 conflict)
[A] Commit e3dfbe on
1
[B] Commit on
1
2 function CalculateAge(birthdate)
3 {
4 //....TODO
5 }
6
2 function CalculateAgeFromBirthDate(birthdate)
3
4 //....TODO
5
6
Open in external merge tool Save
conflict 1 of 1 Edit file output with GitKraken Pro
```

Oplossen van conflicten (via GitKraken)

- Wanneer alle conflicten zijn opgelost...
- **Stap 4:** “commit” & “push”
 1. Je eigen versie was reeds ge”commit”
 2. “Commit” de oplossing van de merge
 3. Push beide commits



```
C:\AP\githubtest\SoftwareEngineering>git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)
```

Ander “merge” scenario

Je wil zowel A als B (of en deel ervan) meenemen in het resultaat

The screenshot shows a code editor with two branches, A and B, merged into a single file.

- Branch A:** Commit 20d102 on *master*. The code contains a check for `birthDate != null`.
- Branch B:** Commit on *origin/master*. The code contains a check for `birthDate` (not `null`).
- Merge Result:** The final code in the editor includes both checks: `//Check if birthDate was not empty` and `if (birthDate != null)`.
- Output:** The output window shows the merged code, which includes both the original check from A and the new check from B.

This screenshot shows a code editor where a specific line from branch B is being rejected.

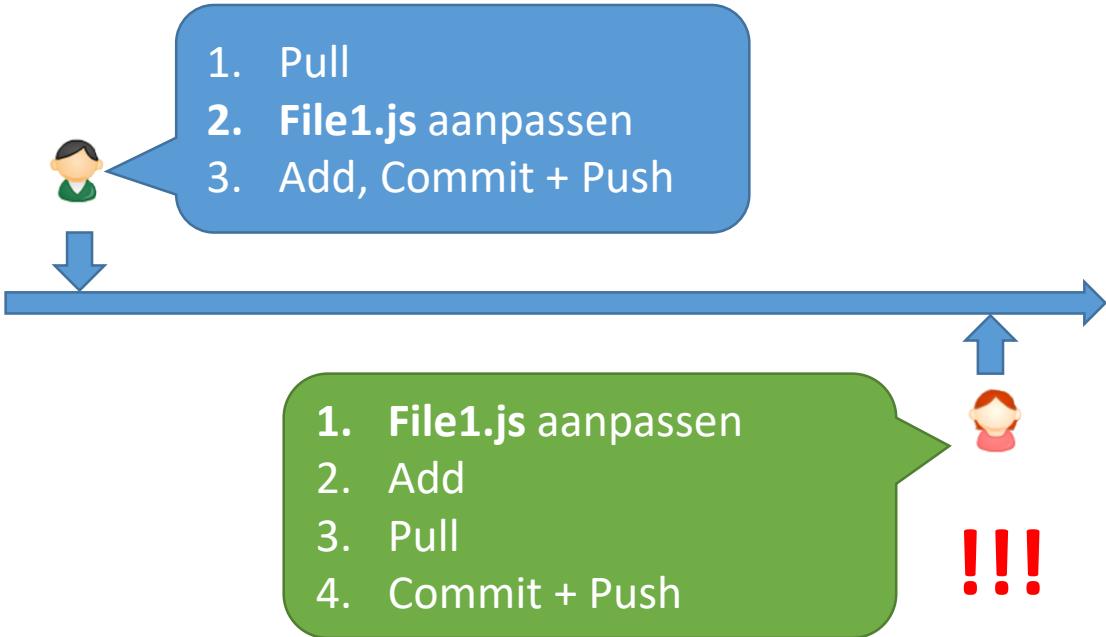
- Branch A:** Commit 20d102 on *master*.
- Branch B:** Commit on *origin/master*. The code contains a check for `birthDate` (not `null`).
- Merge Result:** The code editor highlights the line `if (birthDate)` with a red minus sign, indicating it is being rejected.
- Message:** A large white box with the text "Don't take this line" is overlaid on the code editor.

This screenshot shows a code editor where both branches' code is included in the merge result.

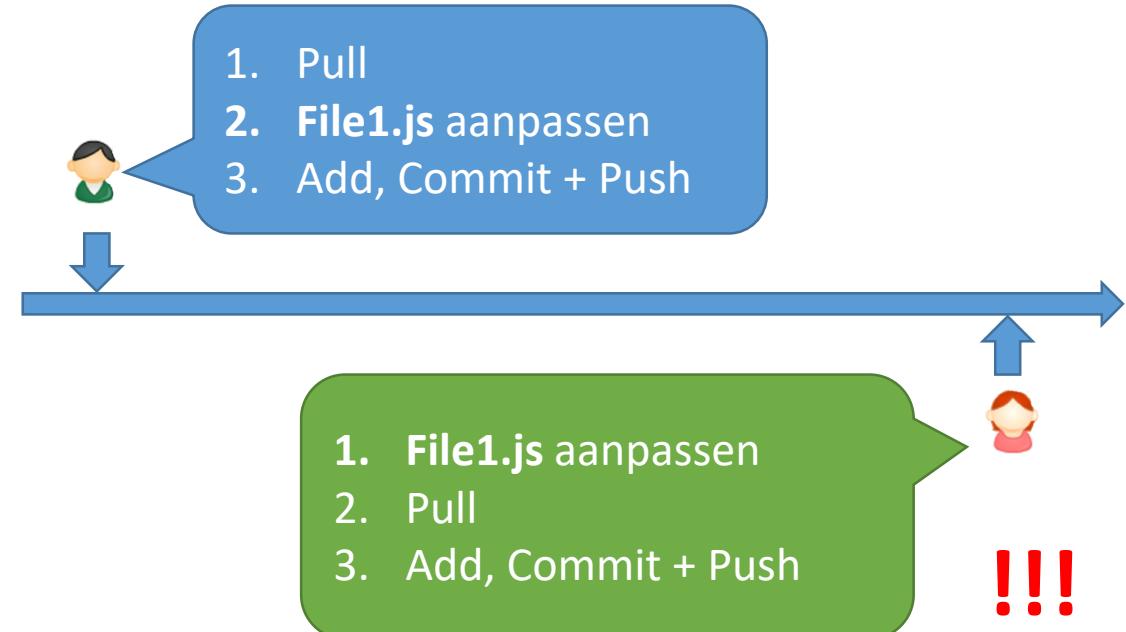
- Branch A:** Commit 20d102 on *master*.
- Branch B:** Commit on *origin/master*. The code contains a check for `birthDate` (not `null`).
- Merge Result:** The code editor shows both the original check from A and the new check from B, with each line preceded by either an A or a B indicator.
- Output:** The output window shows the merged code, which includes both the original check from A and the new check from B.

Scenario 3: werken aan hetzelfde bestand in dezelfde code

3c: Moet je eerst een “commit” doen voor een “pull” ?



3d: Moet je eerst een “add” doen voor een “pull” ?

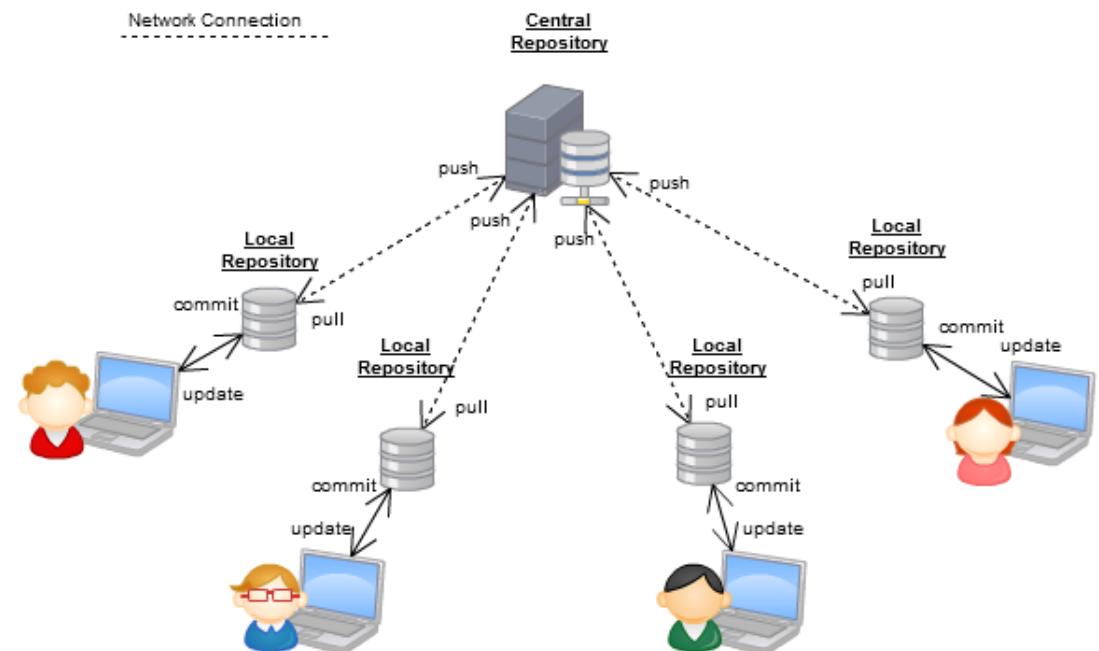


Scenario 3: werken aan hetzelfde bestand in dezelfde code

Samenvatting:

- Pull: Wanneer je aan dezelfde code werkt kan je **ENKEL** een ‘pull’ doen nadat je je eigen wijzigingen hebt ge’**commit**’. Bij de pull zal er een **manuele ‘merge’** moeten gebeuren waarbij de juiste wijzigingen door de gebruiker zal moeten worden aangegeven. Deze manuele merge zal resulteren in een 2^e commit.
- Push: Je kan enkel een ‘push’ doen van je eigen commit(s) nadat je eerst alle andere nieuwe commits hebt opgehaald (pull)

Tip: Als je een push doet van je wijzigingen en je hebt eerst een pull moeten doen, **TEST dan eerst (terug) vooraleer je de push uitvoert zodat je zeker bent dat jouw wijzigingen + die van je collega’s tezamen ook nog correct werken !** Zeker als je een (manuele) merge hebt moeten doen !

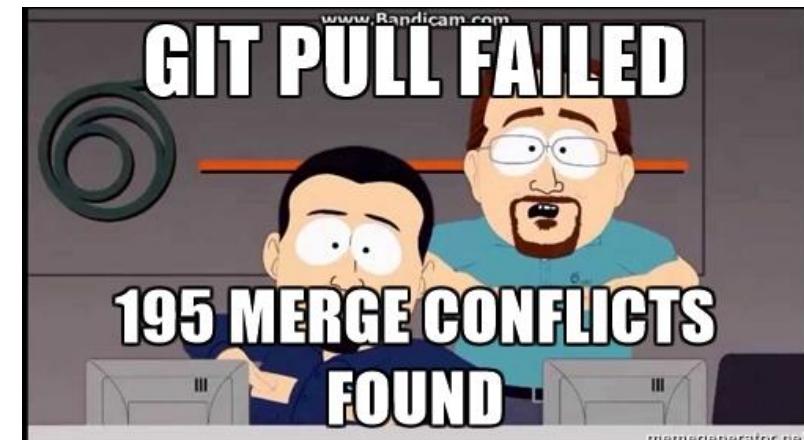


Impact van conflicten beperken



Impact conflicten beperken

1. Maak kleine commits met kleine wijzigingen (1 commit = 1 bugfix/feature/verbetering/... niet “Fixed a ton of bugs”).
2. Push en pull vaak! Zo vaak mogelijk dezelfde staat v. code houden.
3. Pull voor je een push doet.
4. Geef duidelijke commit berichten zodat andere developers weten waarom je iets hebt veranderd.
5. Zet geen binaire bestanden in Git. Nagenoeg altijd merge conflict.



Git Medior (Branching)

Elektronica – ICT

Sven Mariën

(sven.marien01@ap.be)

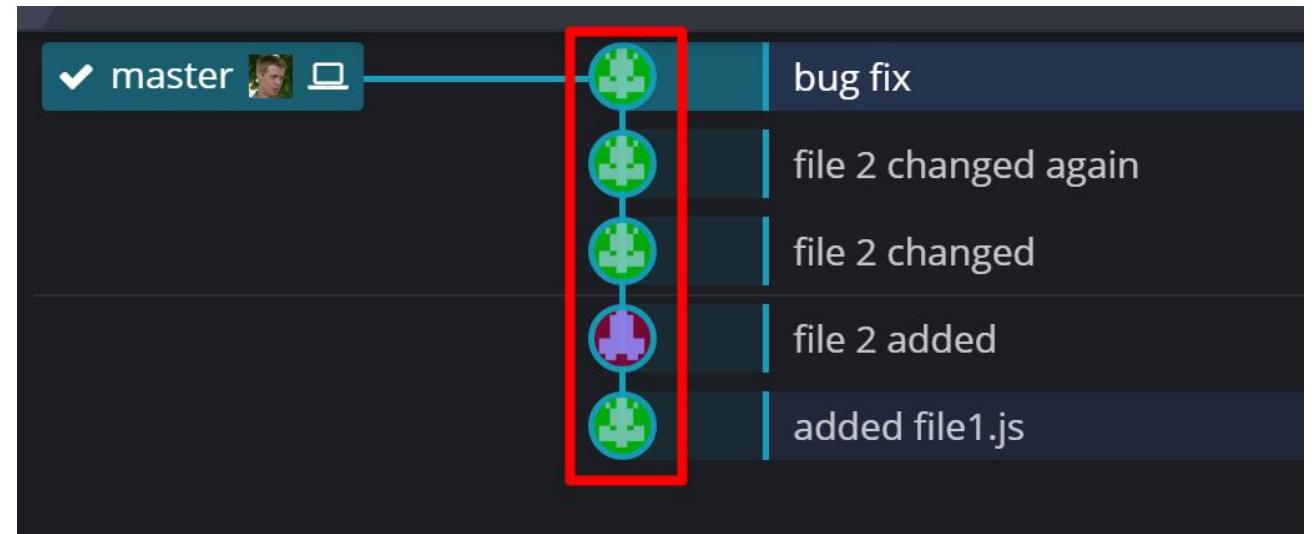
2017-2018



ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

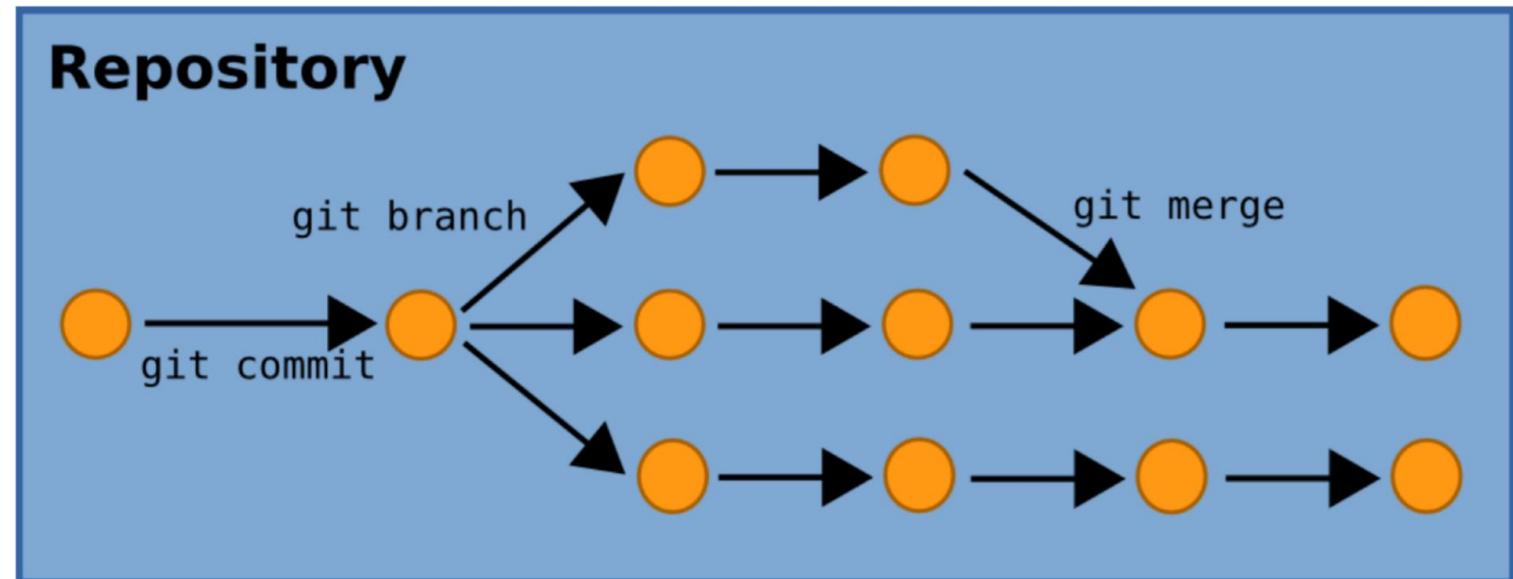
Branching

- Lineaire commit log
- Elke commit wordt onmiddellijk zichtbaar voor de andere teamleden.
- Elke commit wordt ook naar de andere teamleden gestuurd bij een Pull



Branching

- Soms kan het handig / nodig zijn om een tijdje apart te kunnen werken zonder dat je andere teamleden jouw wijzigingen automatisch ontvangen bij een pull:
 - Je wil aan een grotere story werken en je commits wel pushen maar niet automatisch voor iedereen beschikbaar stellen.
 - Je wil iets uitproberen of aanpassen dat je nu wil committen, maar pas op een later moment (of misschien nooit) wil toevoegen aan het project (master branch).
 - Je wil een grotere refactoring doen, die eerst een tijdje grondig getest en geëvalueerd moet worden vooraleer aan het project (master branch) kan worden toegevoegd
 - ...



Branching: overzicht van de bestaande branches

- **git branch**
- Toon de lokale branches
- Hoofdbranch noemt “by default”: **master**

```
C:\AP\githubtest\SEWork>git branch  
* master
```

- **git branch -a**
- Toon de lokale en remote branches
- Remote repo noemt “by default”: **origin**

```
C:\AP\githubtest\SEWork>git branch -a  
* master  
remotes/origin/master
```

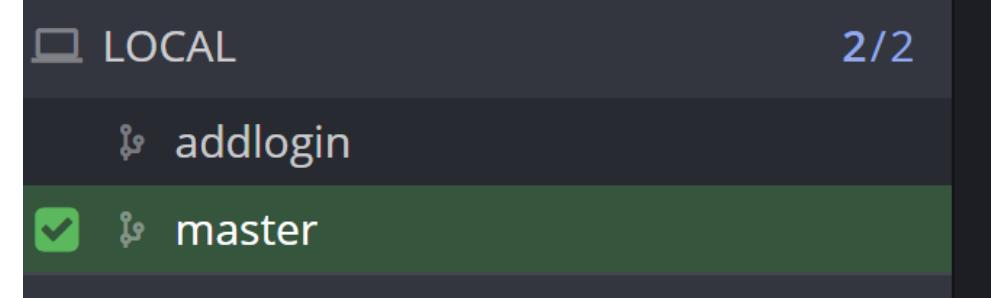
	LOCAL	1/1
<input checked="" type="checkbox"/>	master	
	REMOTE	1/1
 origin		
	 master	

Branch aanmaken

- **git branch [name]**
- Maak een nieuwe branch aan die vertrekt vanop de huidige commit.
- Deze branch wordt dan enkel in de lokale repo aangemaakt.

```
C:\AP\githubtest\SEWork>git branch addlogin
```

```
C:\AP\githubtest\SEWork>git branch  
  addlogin  
* master
```

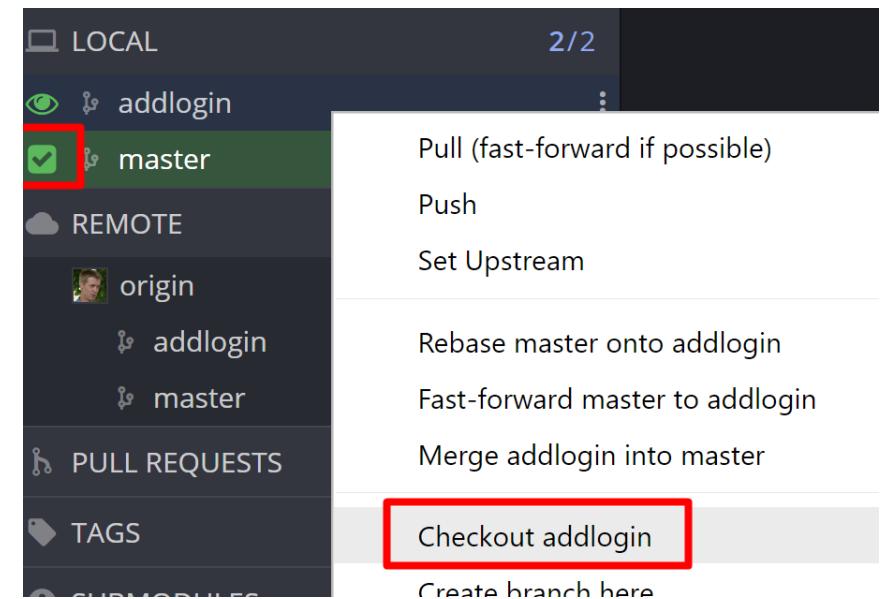


Switchen naar een branch

- **git checkout [name]**
- Open een andere branch (of een specifieke commit).
- Je working directory zal nu deze branch (of commit) weergeven.
- Je kan nu op deze branch verder werken zoals voordien (add, commit, push,...)
- Wanneer je de branch pushed zal ook op de remote repo een branch worden aangemaakt.

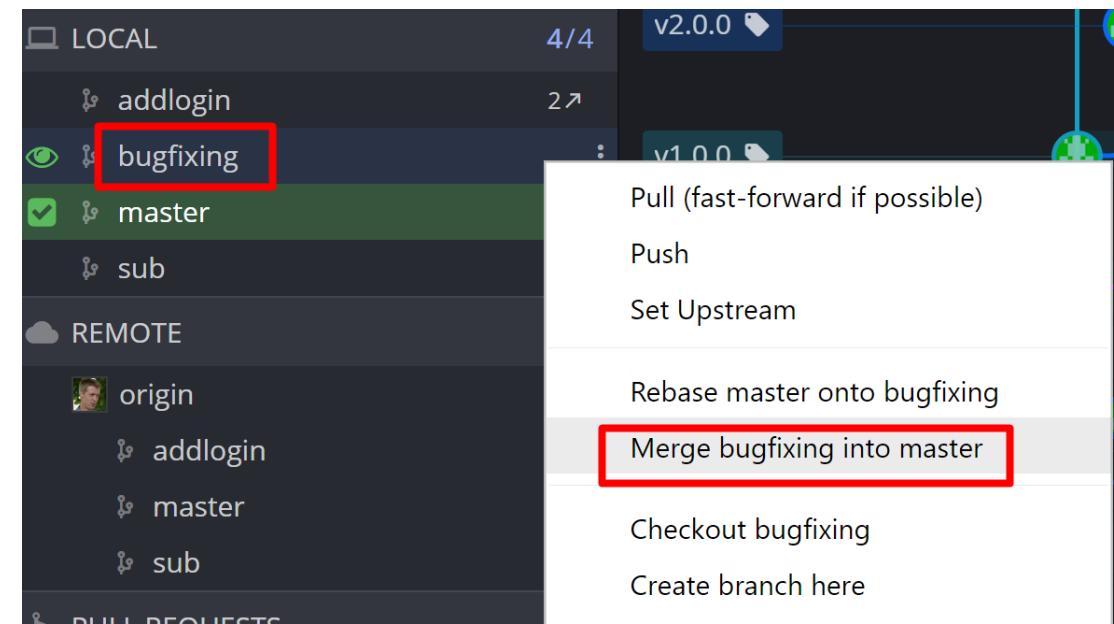
```
C:\AP\githubtest\SEWork>git checkout addlogin
Switched to branch 'addlogin'
Your branch is up to date with 'origin/addlogin'.

C:\AP\githubtest\SEWork>git branch
* addlogin
  master
```



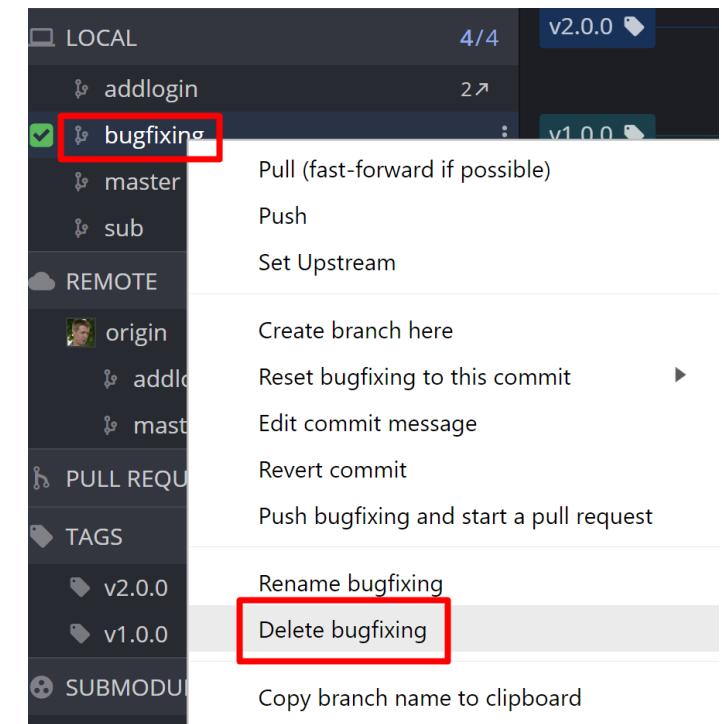
Merge van de branch naar de master

- **git merge [branchname]**
- Neemt de verschillen tussen de huidige branch en de opgegeven branch en maakt een commit die de wijzigingen binnen neemt.
- Doe eerst een checkout op de branch waar je naartoe wil “mergen”



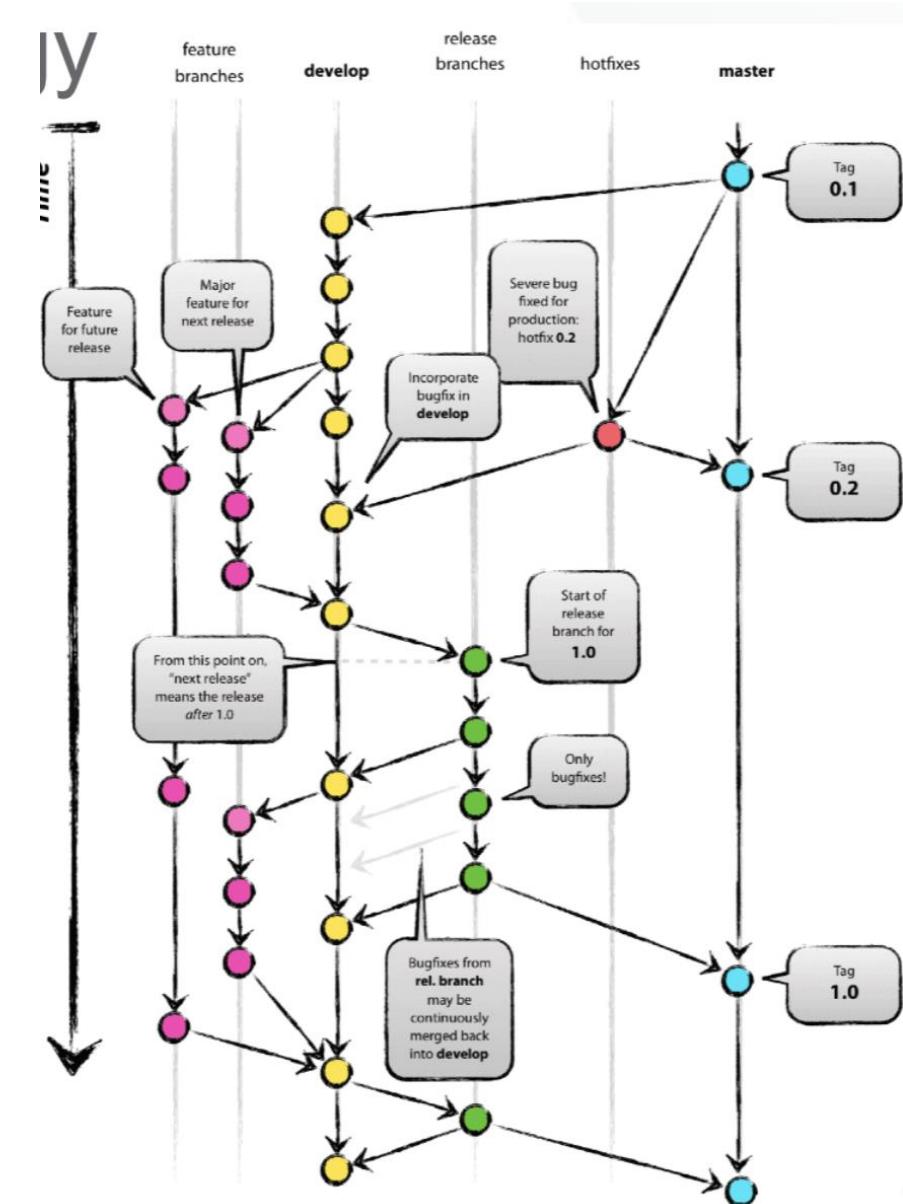
Branches verwijderen

- **git branch –d [name]**
- Verwijder de lokale branch met de opgegeven naam.
- **git push [remote repo name] –delete [branch name]**
- Verwijder de remote branch met de opgegeven naam.



Branching strategy

- Master = stabiele code (huidige release of na QA)
- Develop = dev branch (volgende release of voor QA)
- Aparte release branches worden niet heel vaak gebruikt, eerder gewoon tags
- Feature branches, aparte branch per feature / userstory en nadien merge naar de master en/of development branch



Branching oefening

- Werk per 2 en maak een branch aan ‘development’
- Iemand werkt op de master branch en iemand op de development branch.
- Beiden pushen hun wijzigingen naar de remote repo (dus remote branch aanmaken)
- Zelf al doe je aanpassingen en hetzelfde bestand er zal dus nooit een conflict optreden aangezien je op een andere branch werkt.
- Na enkele wijzigingen doet iemand een merge naar de master branch
 - Op dit moment zullen eventuele conflicten manueel moeten worden opgelost
- Op het einde verwijder je de branch, zowel lokaal als remote

Git Cheat Sheet

<http://git.or.cz/>

Remember: `git command --help`

Global Git configuration is stored in `$HOME/.gitconfig` (`git config --help`)

Create

From existing data
`cd ~/projects/myproject`
`git init`
`git add .`

From existing repo
`git clone ~/existing/repo ~/new/repo`
`git clone git://host.org/project.git`
`git clone ssh://you@host.org/proj.git`

Show

Files changed in working directory
`git status`

Changes to tracked files
`git diff`

What changed between \$ID1 and \$ID2
`git diff $id1 $id2`

History of changes
`git log`

History of changes for file with diffs
`git log -p $file $dir/ec/tory/`

Who changed what and when in a file
`git blame $file`

A commit identified by \$ID
`git show $id`

A specific file from a specific \$ID
`git show $id:$file`

All local branches
`git branch`
(star ¹⁸¹ marks the current branch)

Cheat Sheet Notation

\$id : notation used in this sheet to represent either a commit id, branch or a tag name
\$file : arbitrary file name
\$branch : arbitrary branch name

Concepts

Git Basics

master : default development branch
origin : default upstream repository
HEAD : current branch
HEAD^ : parent of HEAD
HEAD~4 : the great-great grandparent of HEAD

Revert

Return to the last committed state
`git reset --hard` ⚠ you cannot undo a hard reset

Revert the last commit
`git revert HEAD` Creates a new commit

Revert specific commit
`git revert $id` Creates a new commit

Fix the last commit
`git commit -a --amend`
(after editing the broken files)

Checkout the \$id version of a file
`git checkout $id $file`

Branch

Switch to the \$id branch
`git checkout $id`

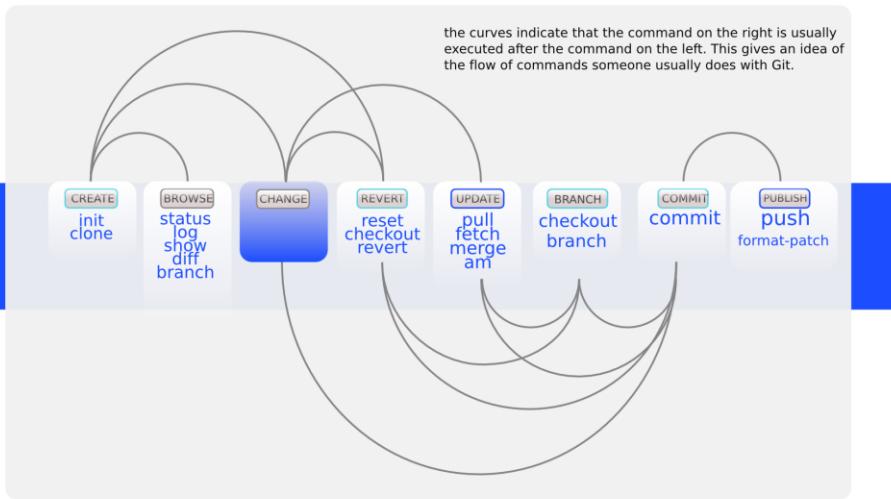
Merge branch1 into branch2
`git checkout $branch2`
`git merge branch1`

Create branch named \$branch based on the HEAD
`git branch $branch`

Create branch \$new_branch based on branch \$other and switch to it
`git checkout -b $new_branch $other`

Delete branch \$branch
`git branch -d $branch`

Commands Sequence



Update

Fetch latest changes from origin
`git fetch`
(but this does not merge them)

Pull latest changes from origin
`git pull`
(does a fetch followed by a merge)

Apply a patch that some sent you
`git am -3 patch.mbox`
(in case of a conflict, resolve and use git am --resolved)

Publish

Commit all your local changes
`git commit -a`

Prepare a patch for other developers
`git format-patch origin`

Push changes to origin
`git push`

Mark a version / milestone
`git tag v1.0`

Useful Commands

Finding regressions

`git bisect start` (to start)
`git bisect good $id` (\$id is the last working version)
`git bisect bad $id` (\$id is a broken version)

`git bisect bad/good` (to mark it as bad or good)
`git bisect visualize` (to launch gitk and mark it)
`git bisect reset` (once you're done)

Check for errors and cleanup repository
`git fsck`
`git gc --prune`

Search working directory for foo()
`git grep "foo()"`

Resolve Merge Conflicts

To view the merge conflicts

`git diff` (complete conflict diff)
`git diff --base $file` (against base file)
`git diff --ours $file` (against your changes)
`git diff --theirs $file` (against other changes)

To discard conflicting patch

`git reset --hard`
`git rebase --skip`

After resolving conflicts, merge with

`git add $conflicting_file` (do for all resolved files)
`git rebase --continue`

References

- <https://git-scm.com/book/en/v2>
- <http://learngitbranching.js.org/>
- <https://try.github.io/levels/1/challenges/1>
- <http://gitreal.codeschool.com/levels/1>
- <http://marklodato.github.io/visual-git-guide/index-en.html>

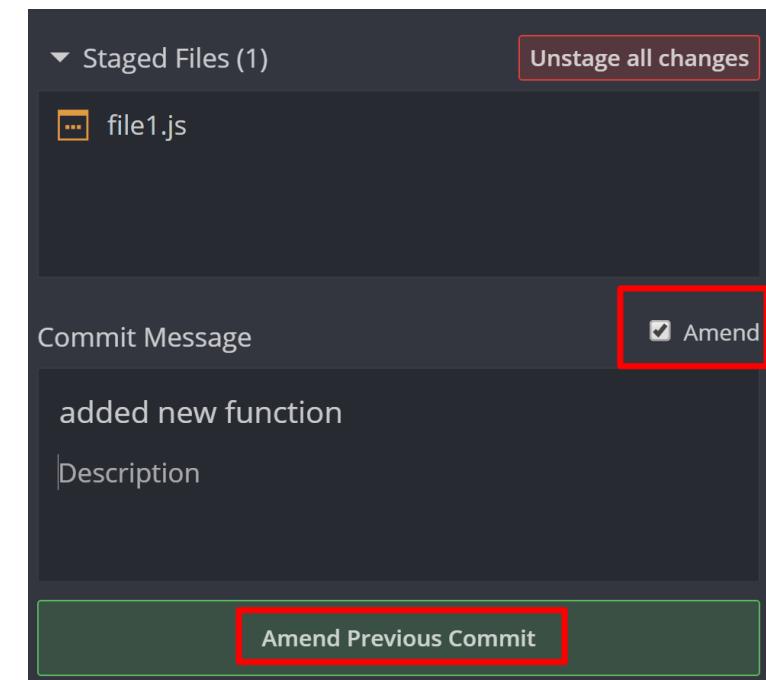
Inhoud deel 2

- Commit met Amend
- Commit revert
- Stashing
- Tags
- Cherry picking
- Rebasing
- Pull requests

Wijzigingen toevoegen aan de laatste commit

- **git commit --amend**
- Voeg de wijzigingen in de staging area toe aan de wijzigingen van de laatste commit **zonder** een nieuwe commit te maken.
- Kan handig zijn als je iets was vergeten toe te voegen aan de laatste commit

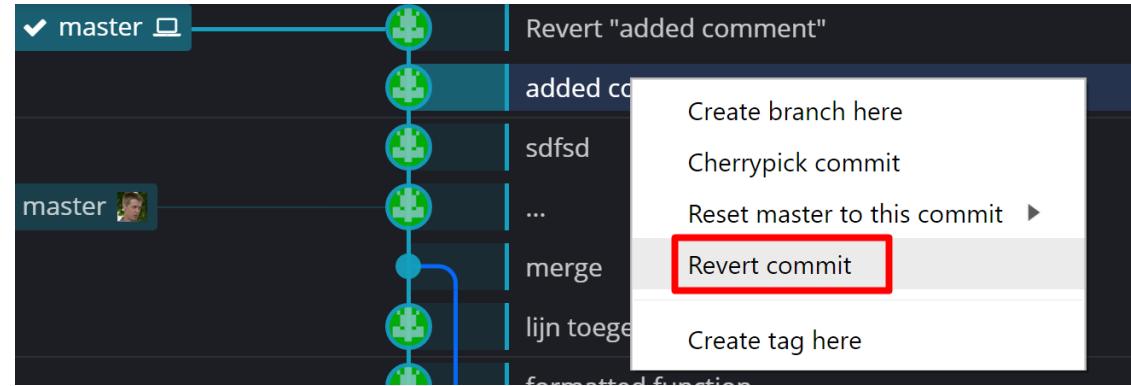
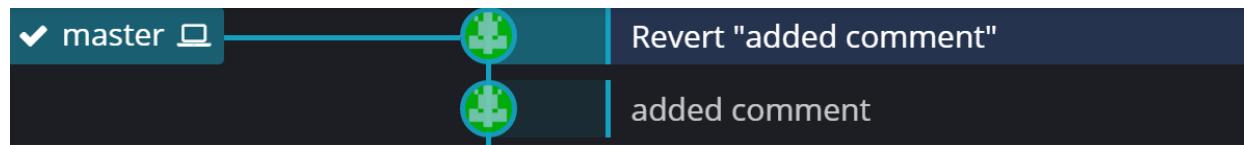
```
C:\AP\githubtest\SEWork>git commit --amend  
[master 967dab6] added neww function  
Date: Sun Feb 25 14:45:58 2018 +0100  
1 file changed, 2 insertions(+), 2 deletions(-)
```



Een commit ongedaan maken

- **git revert [hash]**
- Maak een nieuwe commit dat de opgegeven commit omdraait (herschrijft de geschiedenis dus niet)

```
C:\AP\githubtest\SEWork>git revert 8a2b0a
[master 0673b6c] Revert "added comment"
 1 file changed, 1 insertion(+), 1 deletion(-)
```



Revert "added comment"

This reverts commit 8a2b0ab8604c4e29eb63e7c3d5ce3f14fd53 e1bf.

Sven authored 23-2-2018 @ 15:09

parent: 8a2b0a

1 modified

Name ↴ Path Tree View all files

file1.js

A detailed view of the 'Revert "added comment"' commit. It shows the commit message, author information, and a list of modified files. The 'Path' tab is selected.

Wijzigingen tijdelijk opzij plaatsen

- **git stash [push –m “message...”]**
- Maakt de lokale niet-gecommitte wijzigingen (van bestanden onder git controle) tijdelijk ongedaan (in de working directory)
- Maar deze wijzigingen worden wel bijgehouden in een achterliggende “stack”
- De wijzigingen worden dus niet ge’commit.

```
C:\AP\githubtest\SEWork>git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

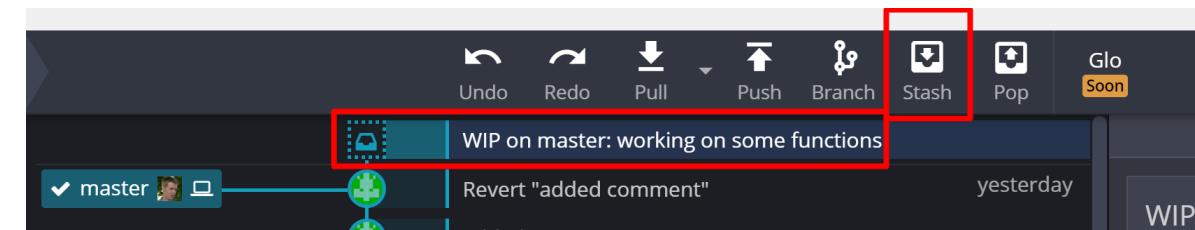
        modified:   file1.js

no changes added to commit (use "git add" and/or "git commit -a")

C:\AP\githubtest\SEWork>git stash push -m "working on function Test"
Saved working directory and index state On master: working on function Test

C:\AP\githubtest\SEWork>git status
On branch master
Your branch is up to date with 'origin/master'.

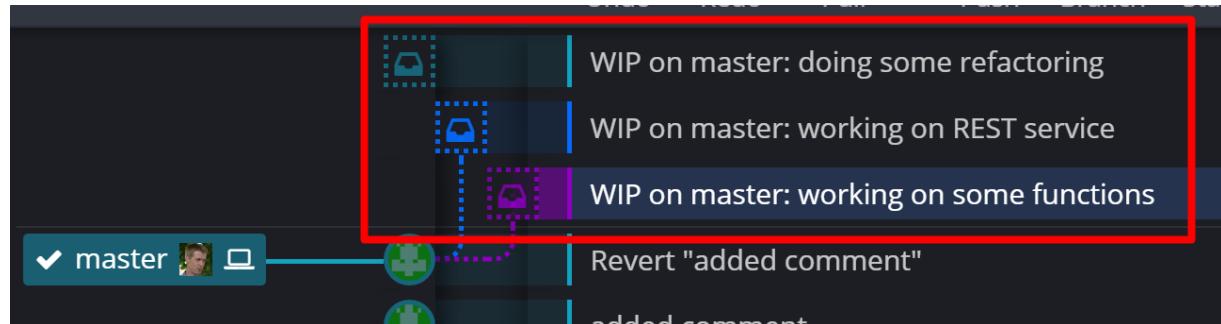
nothing to commit, working tree clean
```



Meerdere wijzigingen na elkaar “stashen”

- **git stash list**
- Geeft een overzicht van de wijzigingen op de stash stack (=LIFO)

```
C:\AP\githubtest\SEWork>git stash list
stash@{0}: On master: doing some refactoring
stash@{1}: On master: working on REST service
stash@{2}: On master: working on some functions
```



Wijzigingen terug ophalen naar de “working directory”

- **git stash pop**
- Haal de meest recent ge”stash”de wijzigingen terug op uit de stash stack

```
C:\AP\githubtest\SEWork>git stash list
stash@{0}: On master: doing some refactoring
stash@{1}: On master: working on REST service
stash@{2}: On master: working on some functions

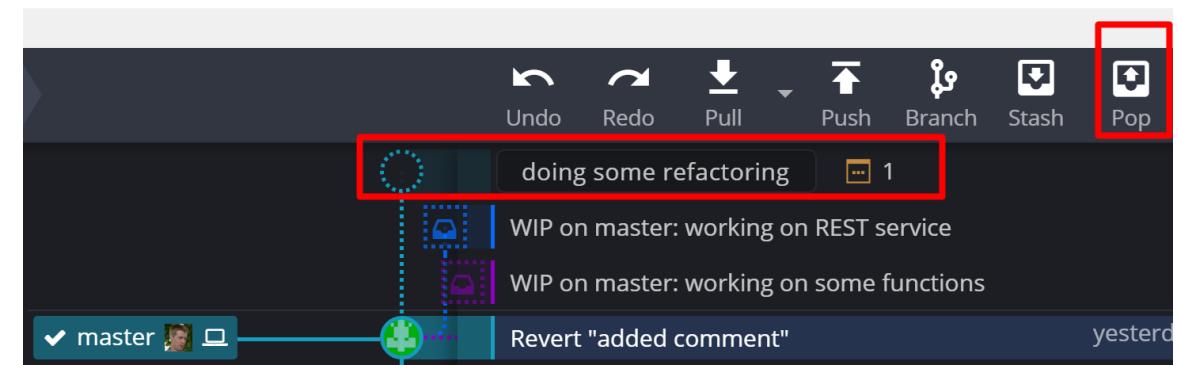
C:\AP\githubtest\SEWork>git stash pop
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

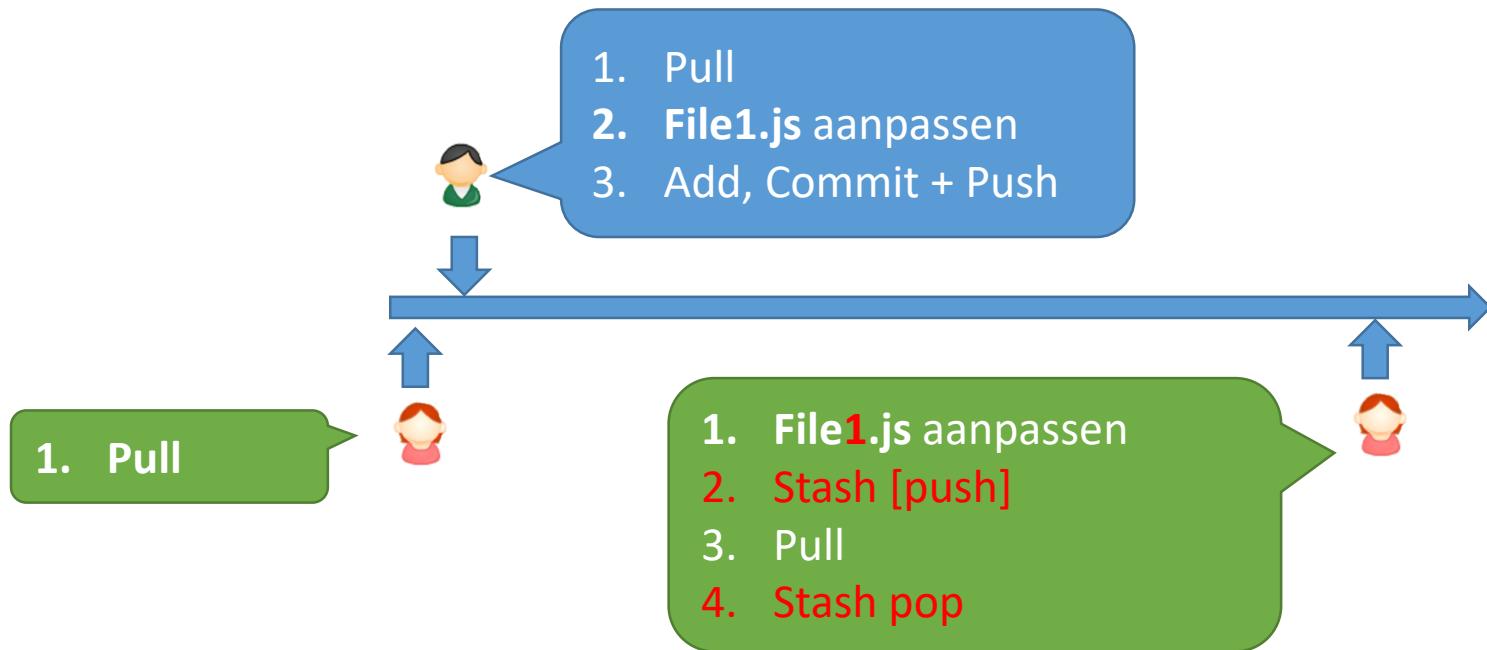
    modified:   file2.js

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (bae168053d9866dbb08d51038d63c13962d74d8c)

C:\AP\githubtest\SEWork>git stash list
stash@{0}: On master: working on REST service
stash@{1}: On master: working on some functions
```



Scenario : Je wil toch de wijzigingen van je collega's (die ook aan hetzelfde bestand hebben gewerkt) pullen zonder eerst te moeten committen



1. “Stash” eerst je wijzigingen
2. Doe vervolgens een ‘pull’ request
3. Haal je wijzigingen terug met een “stash pop”

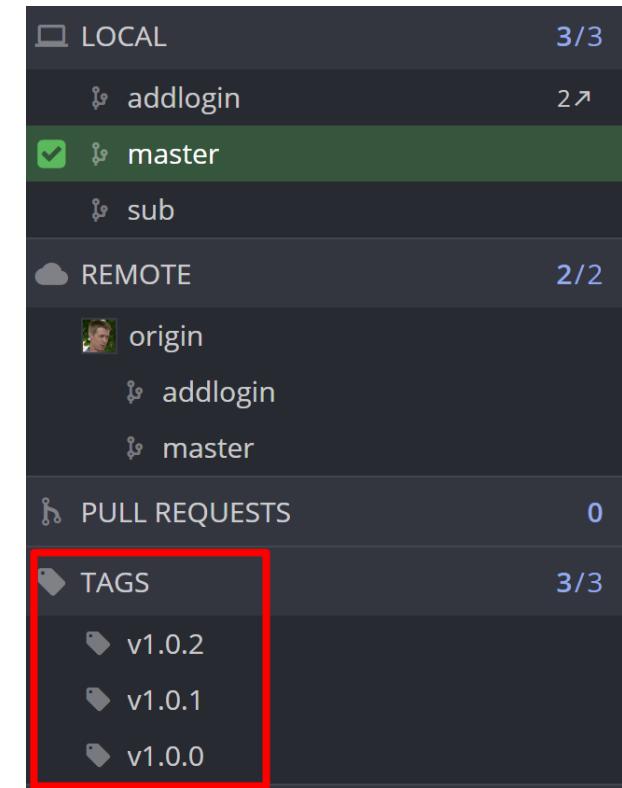
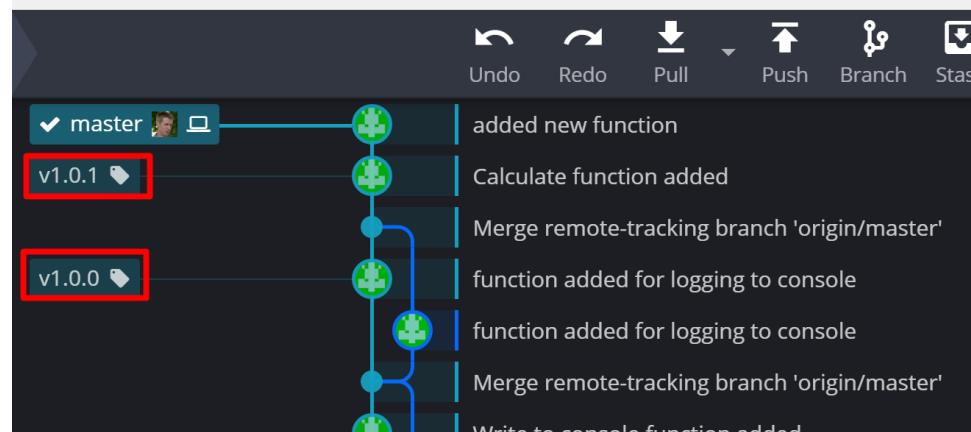
“Tags” toevoegen (versies)

- **git tag [message / label]**
- Maak een tag bij een specifieke commit. Een tag is vaak een belangrijke snapshot van je project. B.v. een **specifieke versie** of een release.

```
C:\AP\githubtest\SEWork>git tag v1.0.2
```



```
C:\AP\githubtest\SEWork>git tag
v1.0.0
v1.0.1
v1.0.2
```



Versie met “tag” terug opvragen

- **git checkout [tag]**
- Checkout van een van een tag. De working directory zal de versie bevatten die bij die tag behoort.
- Je kan vanaf deze versie dan terug commits doen, maar deze zijn **tijdelijk** en voor niemand anders zichtbaar. Om ze permanent bij te houden moet je dan een branch maken of deze mergen naar een bestaande branch.

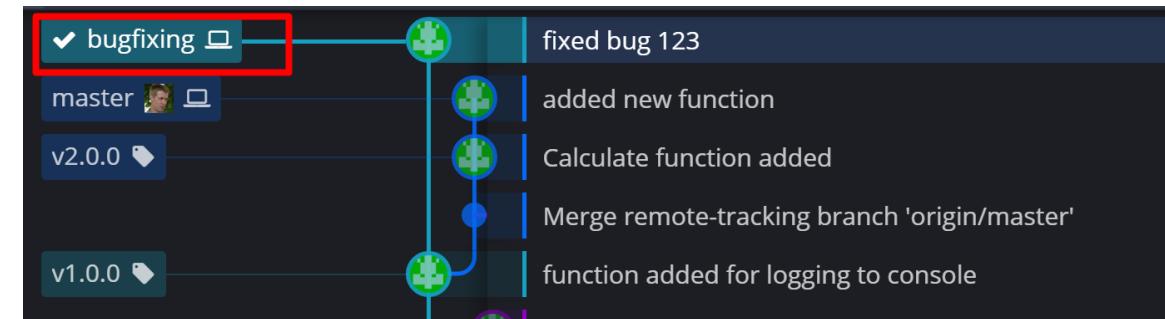
```
C:\AP\githubtest\SEWork>git checkout v1.0.0
Note: checking out 'v1.0.0'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 4aa939c... function added for logging to console
```

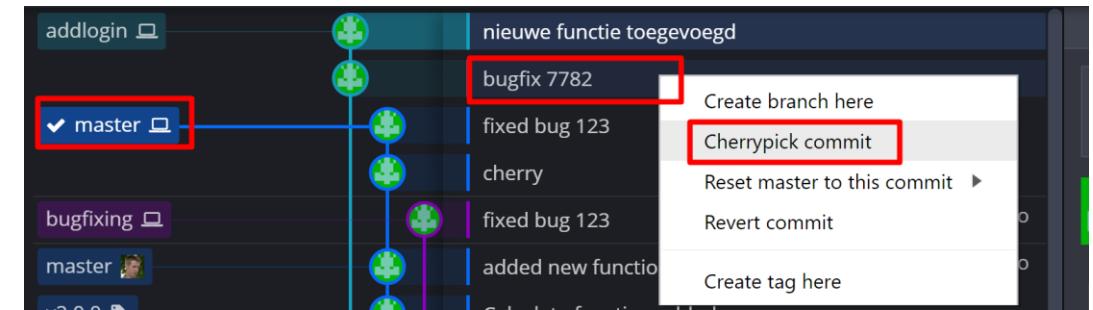


Een specifieke commit samenvoegen

- **git cherry-pick [commitID]**
- Neemt de wijzigingen van de opgegeven commit (die ergens anders in de repo staat) en pas de commit toe in de huidige branch.
- Je hebt bijvoorbeeld de oplossing voor een bug reeds ge'commit op een development branch en je wil **enkel die commit** ook gaan toevoegen aan de master branch.

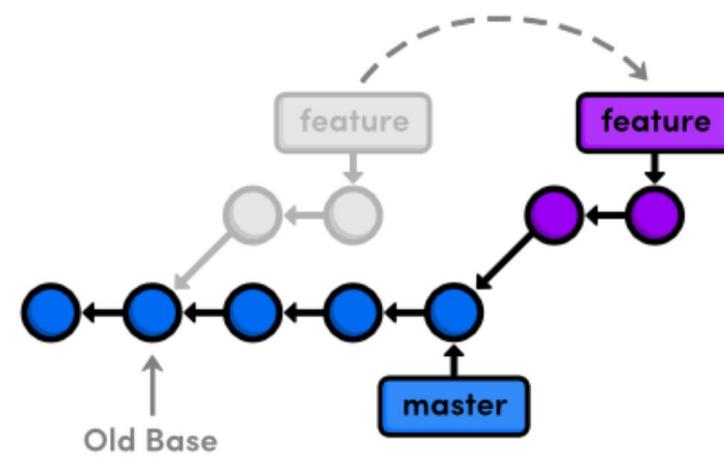
```
C:\AP\githubtest\SEWork>git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

C:\AP\githubtest\SEWork>git cherry-pick 8c49e6d7
[master 084e2c4] bugfix 7782
  Date: Sun Feb 25 18:21:03 2018 +0100
  1 file changed, 1 insertion(+)
  create mode 100644 file3.js
```



Alternatief voor merge => rebase

- **git rebase [basebranch]**
- Voer een rij van commits uit bovenop een bepaalde basis commit of branch...
- Zo behoud je een lineaire history (in tegenstelling tot de aftakking bij “Branching”)
- Opgelet, wees hiermee voorzichtig:
 - **de geschiedenis wordt herschreven !**
 - Doe dit enkel van een (feature) branch (waar jij enkel op werkt, maw. een local branch)
 - Doe dit beter NIET van een branch waar het team op werkt (bv. Master branch) OF van een branch die je moet pushen naar een remote repo (git push werkt niet meer) !



Samenwerken adhv. “Pull requests”

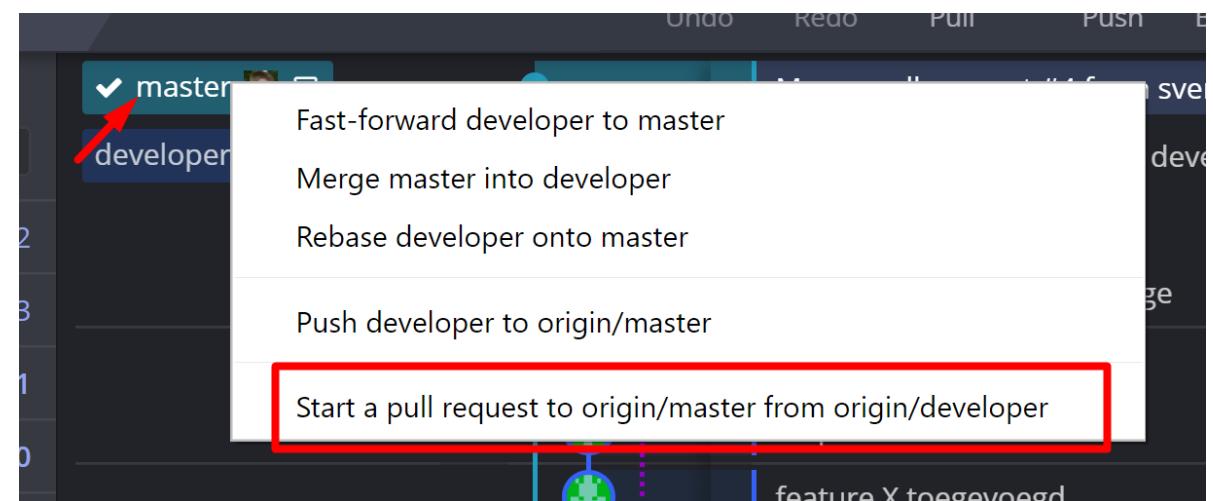
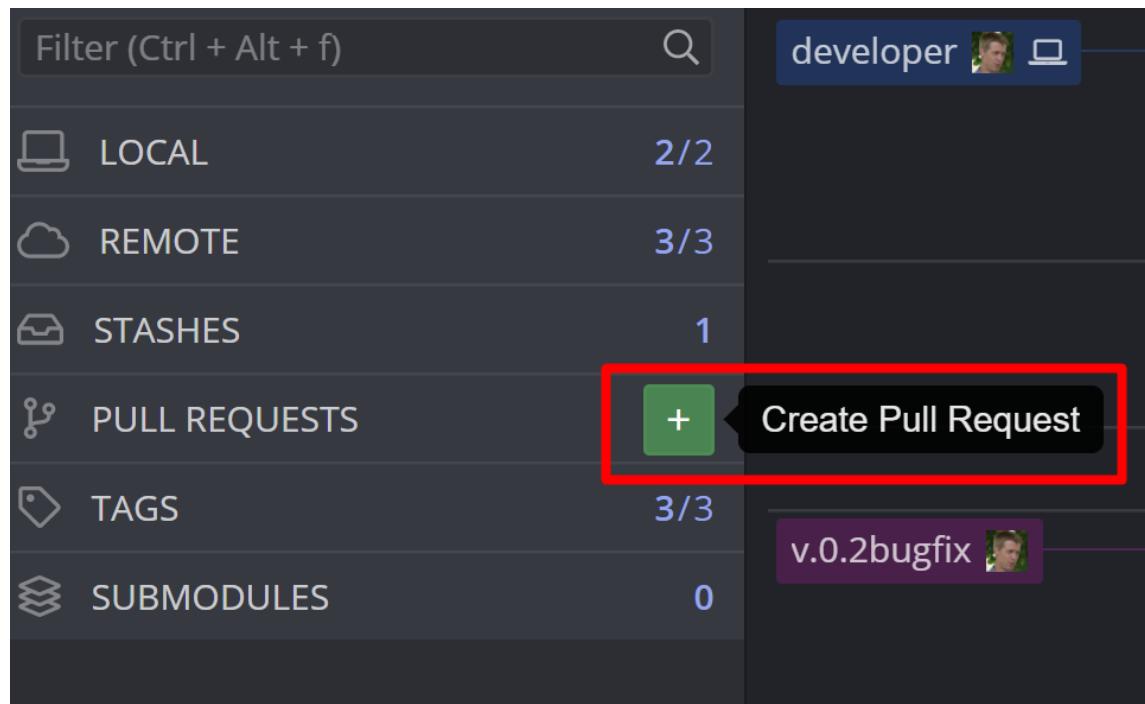
- **Github** biedt de mogelijkheid om te werken met zogenaamde “**pull requests**”
- Dit is dus een functie van de “remote” repo. (dus niet lokaal van git)
- Dit is een **extra stap vóór de merge** van je feature branch naar de main (development, master ...) branch
- Geeft de kans aan **andere teamleden** om eerst de aanpassingen te bekijken, eventueel te testen en feedback te geven.
- Naar aanleiding van die feedback kunnen er mogelijk nog enkele aanpassingen / fixes worden toegevoegd. We blijven voorlopig werken op de feature branch.
- Als iedereen akkoord is kan dan de merge worden uitgevoerd en komt de feature volledig beschikbaar op de main branch.
- Merk op dat nu de **merge** zal gebeuren aan de ‘**remote**’ zijde (dus op github). Je zal dus nadien ook zelf een pull moeten doen om deze “merge commit” in je lokale repo te krijgen.

Workflow met pull-requests



Pull request aanmaken

- Zorg dat je zeker alles ge'pushed' hebt naar github.
- Kan zowel vanuit Gitkraken als vanuit Github
- Gitkraken: via drag and drop , of met de + knop



Pull request aanmaken vanuit github

References

- Merging vs. Rebasing: <https://www.atlassian.com/git/tutorials/merging-vs-rebasing>

Oefeningen

- Vertrek van de master branch en voeg enkele commits toe
- Maak vervolgens een lokale feature branch waarop je ook enkele commits doet
- Ondertussen werkt je collega verder op de master branch
- Doe een cherry picking van 1 bepaalde commit uit die master branch naar de feature branch
- Doe een rebase van de master branch naar de feature branch
- Werk verder met enkele commits op de feature branch en je collega op de master branch
- Merge uiteindelijk alle wijzigingen van de feature branch naar de master branch en doe een push
- Geef deze versie de tag “v1.0.0”

A large graphic element consisting of two overlapping shapes. On the left is a triangle pointing upwards, filled with a bright red color. To its right is a circle, also filled with a bright red color. The two shapes overlap slightly, with the triangle's apex pointing towards the circle's center. The entire graphic is set against a white background.

IP