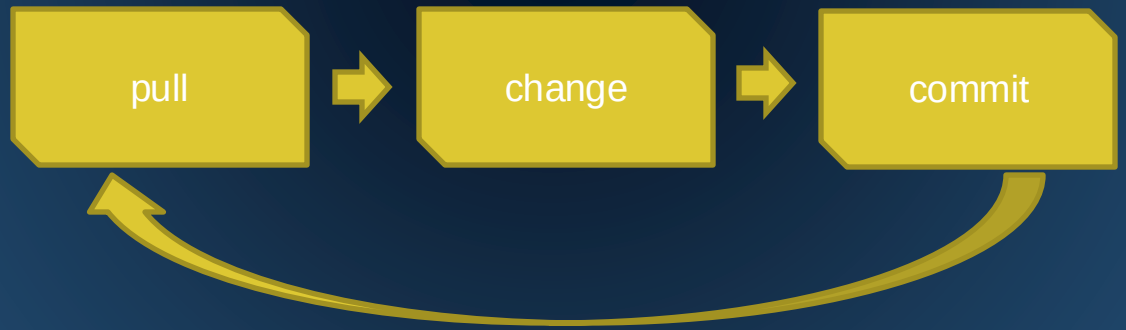
4 Git workflow



Overzicht

- Workflow
  - Pull
  - Scenario’s
    - Verder werken met git
    - Nieuwe feature/bugfix
  - Totale cyclus
- opdrachten

Git workflow

Tussenbeoordeling skill-git

- Versiebeheer |GD

Versiebeheer |GD

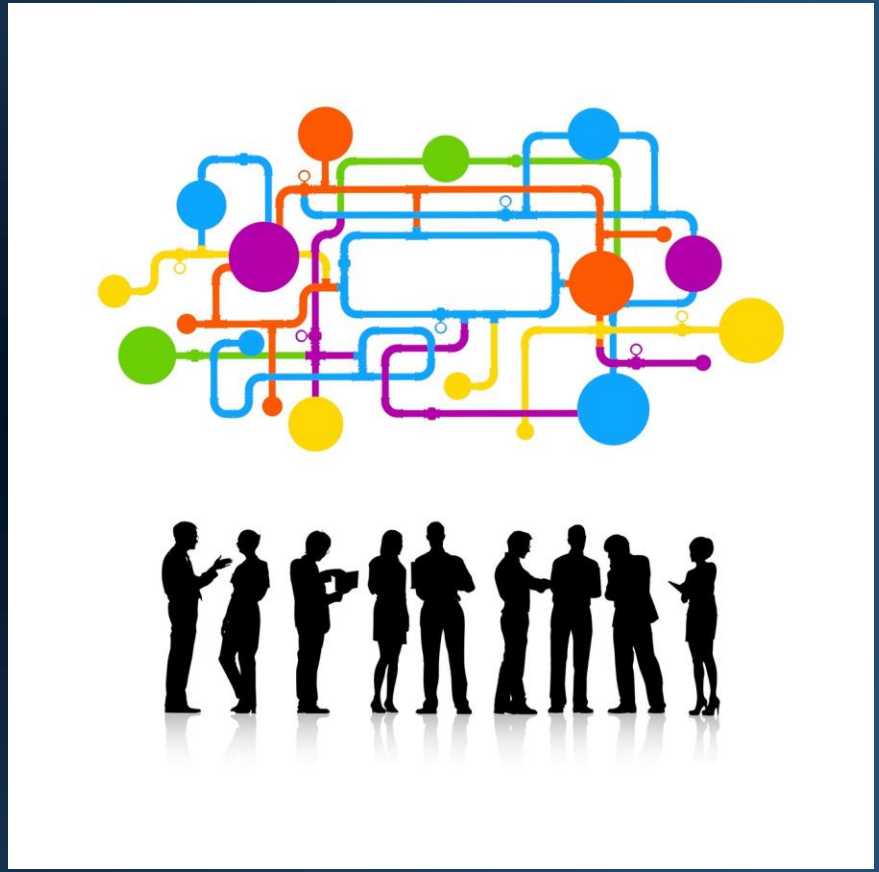
100% / 100%

Ik kan nog niet bewijzen dat ik op niveau ben.	Ik kan code naar GitHub pushen via de commandline.	Ik kan beheren welke bestanden ik toevoeg aan Git.	Ik kan Git configureren voor game engines. of webproject	Ik kan werken met Gitflow en pull requests maken.	Ik kan effectief versiebeheer toepassen in projecten.

Workflow

Is hoe je dagelijks met git omgaat

- Changes van je teamgenoten binnenhalen
- Je eigen changes maken
- Alles netjes bij elkaar zetten



Fetch

Met fetch:

- haal je de laatste staat van de remote op.
- Je haalt op of er nieuwe branches zijn.
- Je haalt op of er branches verwijderd zijn.

Commando:

```
git fetch
```

Pull

Met pull:

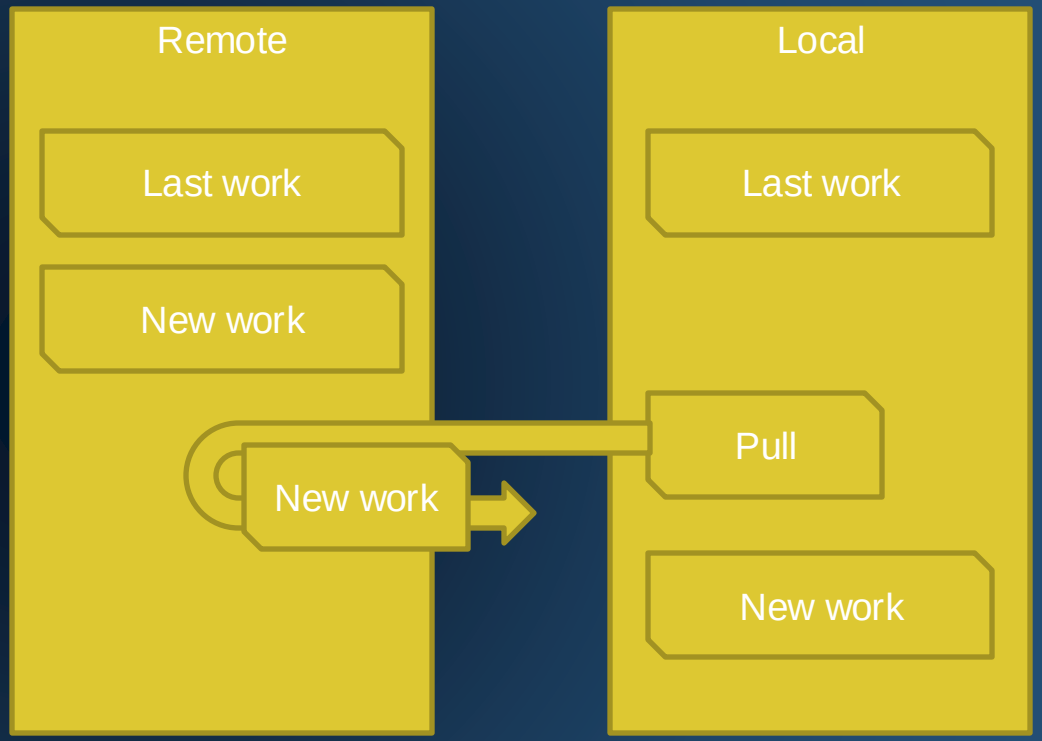
- haal je de laatste staat van de remote op
- sla je die status in je repo op

Commando:

```
git pull
```

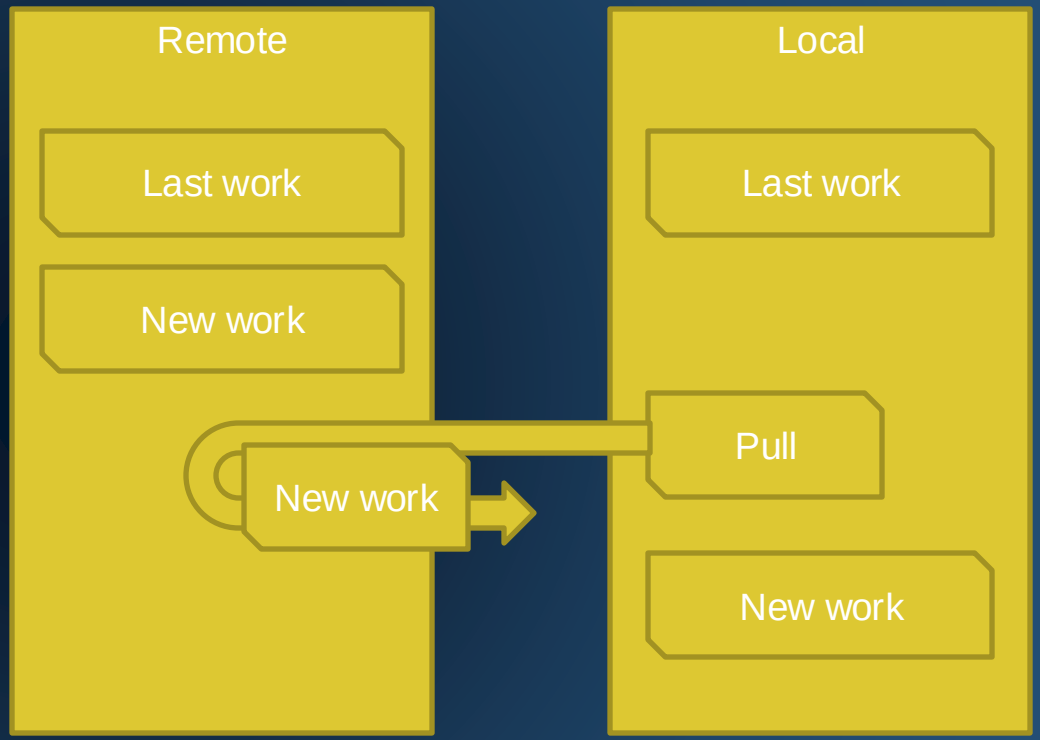
Update flow

1. Switch naar de correcte branch
   `(git checkout main)`
2. Haal je de laatste staat van de remote op
   `(git pull)`
3. Switch naar jouw werk branch
   `(git checkout feature_xxxx)`
4. Voeg je de geupdate branch samen met jouw locale branch
   `(git merge main)`
5. Verstuur de wijzigingen naar gitHub
   ```
   git add .
   git commit
   git push
   ```



Update flow is dus:

- Switch
- Pull
- Merge
- Add
- Commit
- Push



2 Hoofd scenarios

Nieuwe feature/fix etc

Begint met een nieuwe branch

Verder werken

Gaat verder op je huidige branch

Verder werken

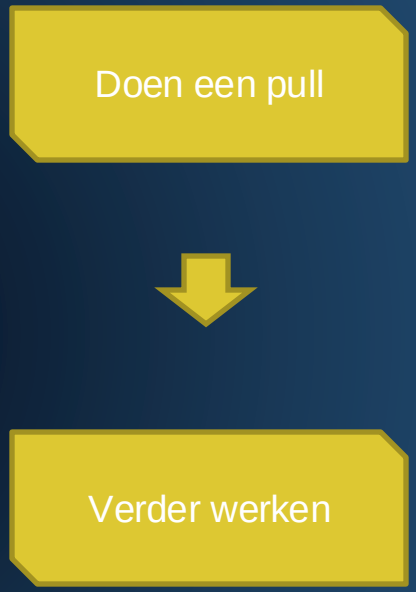
Je blijft op je huidige branch (main)

Als je begint doe je eerst een Pull

- Dit om de laatste complete changes van andere mee te nemen

- Als er conflicten zijn, nu oplossen, scheelt straks meer werk

- Nu kunnen we werken!



Je changes commiten

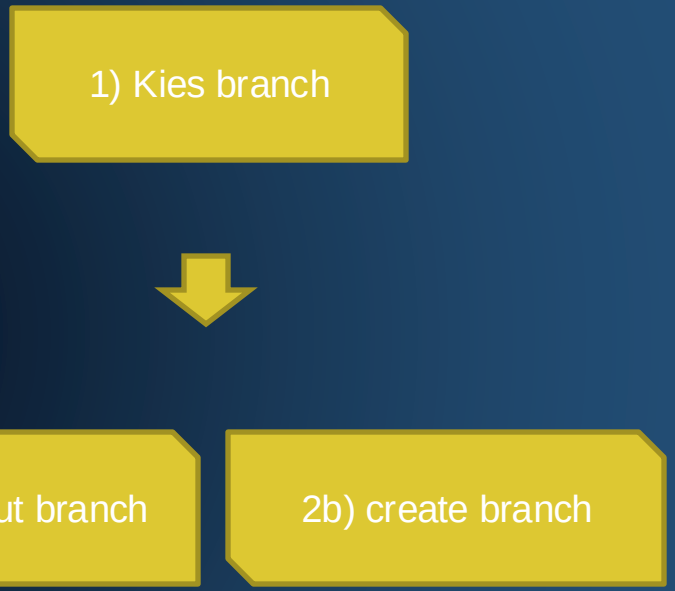
1. Check je werk eerst!
    - (compiled? Errors? Getest?)
2. Stage je werk
3. Check je stage (git status)
4. Maak een commit
5. Pull nogmaals
    - iemand kan nieuw werk hebben gepushed
6. Conflicts? Oplossen!
7. Push

Conflicts komen in een ander les

Nieuwe feature/bug prep

Je begint met zorgen dat je op de juiste plek werkt:

1. Kies de juiste branch (of maken)
2. Die branch haal je binnen of maak je



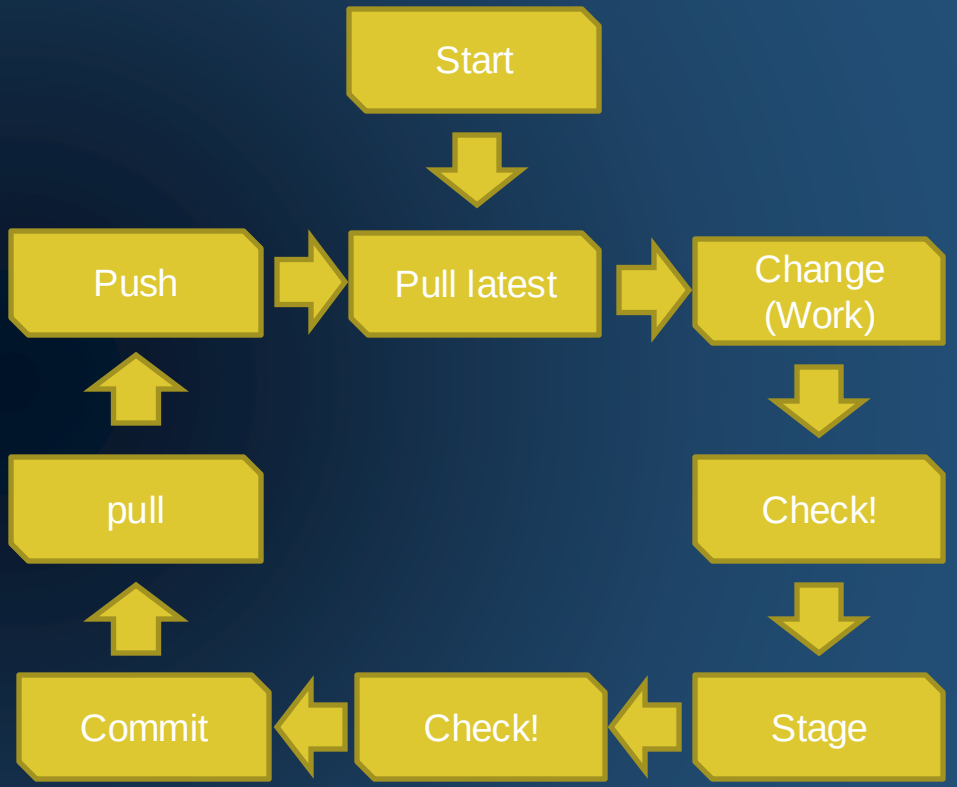
Branching komt de volgende les!

Gebruik de voor deze les de main

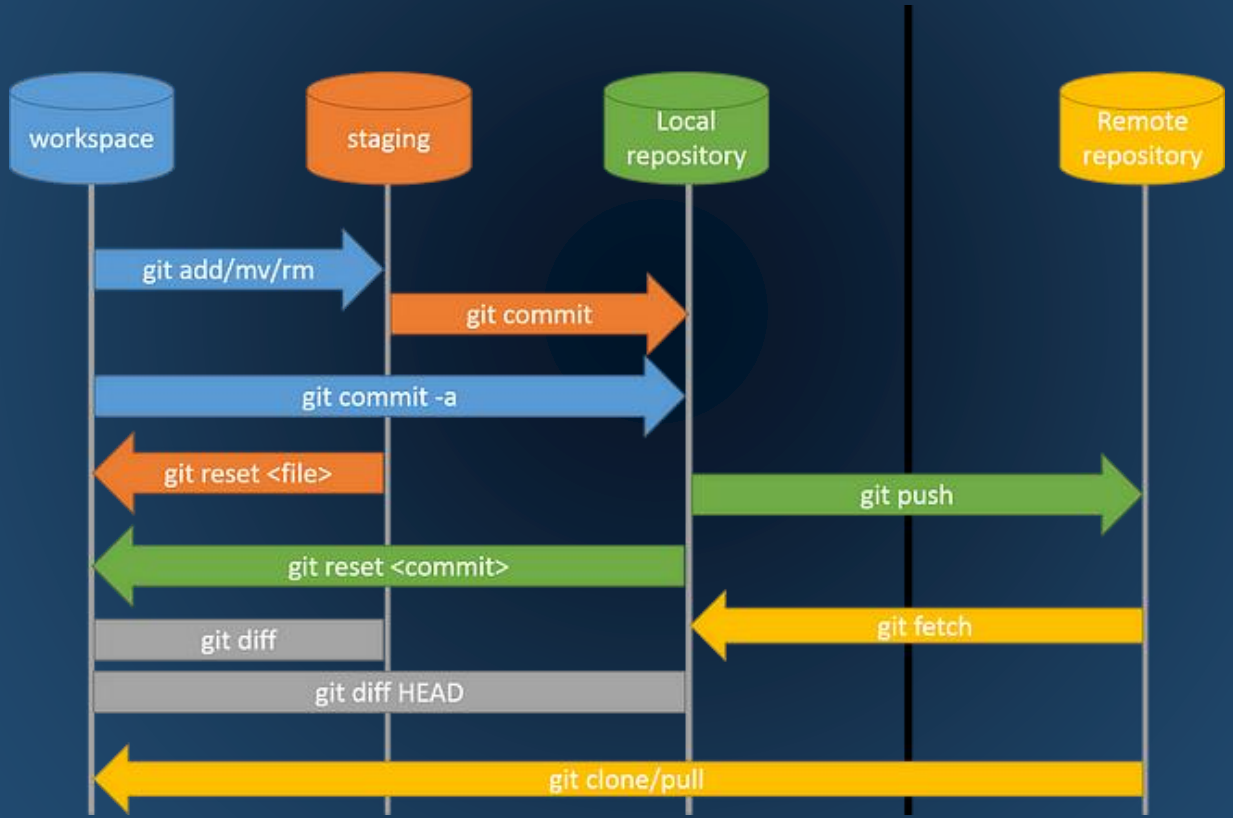
Totale cyclus

Nu zijn we klaar voor de start, je dag begint.

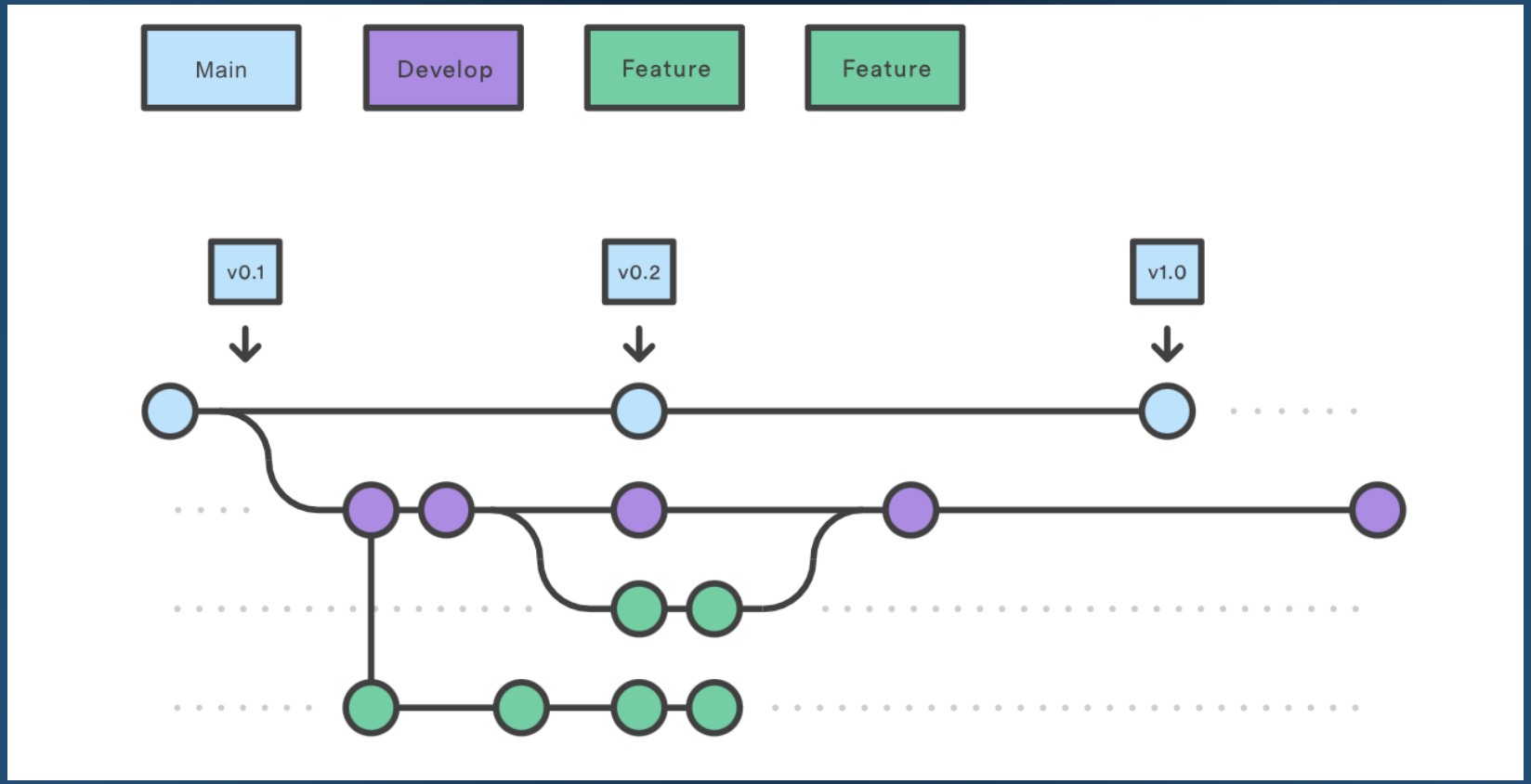
- Eerst haal je de laatste status op (*Dit voorkomt problemen*)
- Je doet je changes & test/checked deze
- Stage je changes
- Een laatste check (*commit moet goed zijn!*)
- Maak een commit
- Pull de laatste status (*problemen komen hier naar boven*)
- Als dat goed gaat of gefixed is push je



Overzicht workflow



Overzicht workflow



Opdracht

Aan de slag

Team:

1. Maak een team met 2 tot 5 teamleden.
2. Bepaal wie de RepoMaster wordt
3. Bepaal welke taken de teamleden krijgen

RepoMaster:

1. RepoMaster zet een lokale repository op
2. Start de branche `main` met een README.md (met als inhoud de namen van de teamleden)
3. Plaats het minimarket project
4. Start een branch `development` Ze daar de bestanden neer waar je mee wil beginnen
5. Stuur de repo naar de remote en
6. Voeg de teamleden als collaborator toe

Teamlid

1. Clone de remote REPO
2. Checkout de branch development
3. Maak een eigen branch met jouw eigen naam als branchnaam
4. Ontwikkel jouw aandeel van het teamprobject
5. Gebruik hetzelfde project !
6. Push jouw aandeel naar de remote

Team:

1. Bespreek de toevoegingen
2. Merge alle branches in `development`
3. Merge develop in de branch `main`
4. Lever de URL in bij simulise