

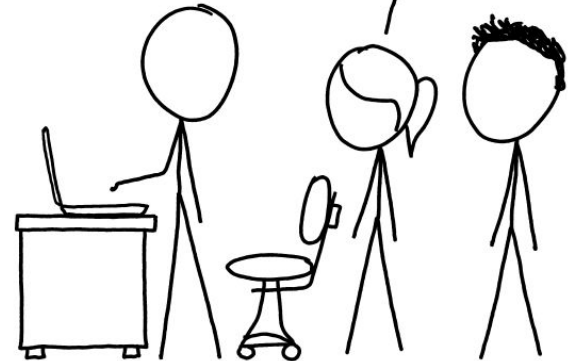
# Workshop GIT

Wesley Klop

THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.





# Dit ben ik

- Software Engineering student
- Al ±8 jaar aan het programmeren
- 
- Mail: [wesley19097@gmail.com](mailto:wesley19097@gmail.com)
- GitHub: <https://github.com/WesleyKlop>
- Ook te vinden op Teams voor vragen



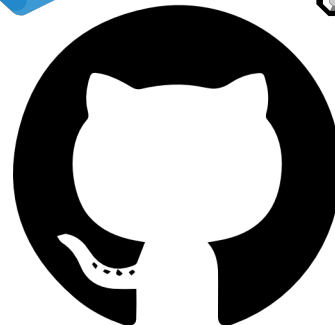
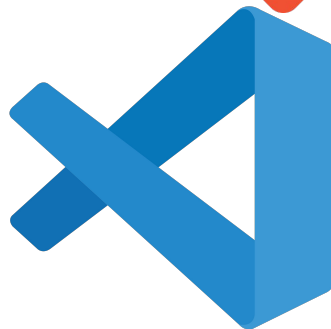


# Dit heb je nodig

- Git CLI (<https://git-scm.com>)
- Een editor / IDE naar keuze bijv.
  - VSCode
  - Vim
  - JetBrains
- Een GitHub account
  - Of Gitlab
  - Of Bitbucket
  - Of whatever



# git





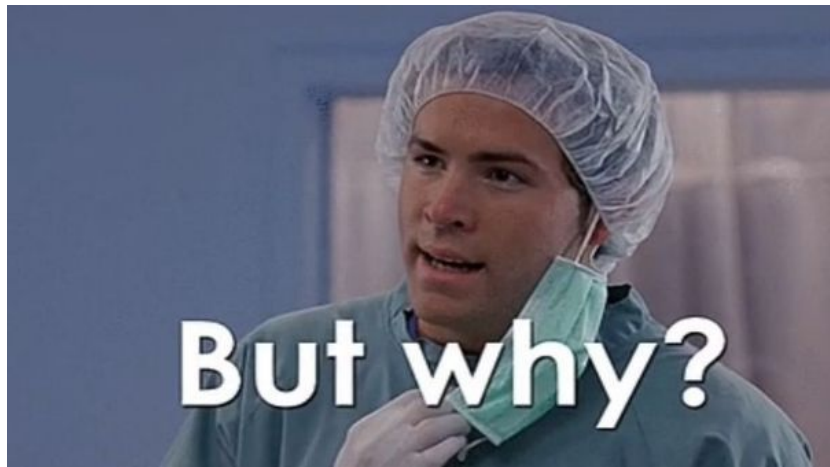
# Dit gaan we behandelen

- Waarom deze workshop?
- Wat is git?
- Zo moet het niet
- Zo kan het wel
- Aan de slag
  - Mini mini crash course git commando's
  - De opdracht(en)
- Tips & Tricks



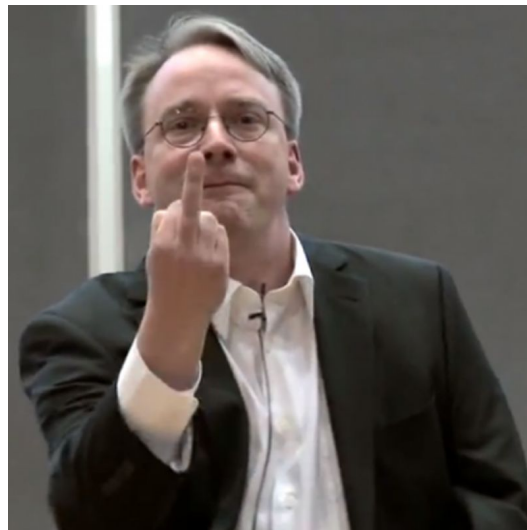
# Waarom deze workshop

- Veel programmeurs gezien die geen git kunnen of maar wat doen
- Easy to learn, difficult to ~~master~~ main
- Essentieel in het bedrijfsleven en tijdens je studie
- Beter samenwerken
- Dit mist in het curriculum



# Wat is Git?

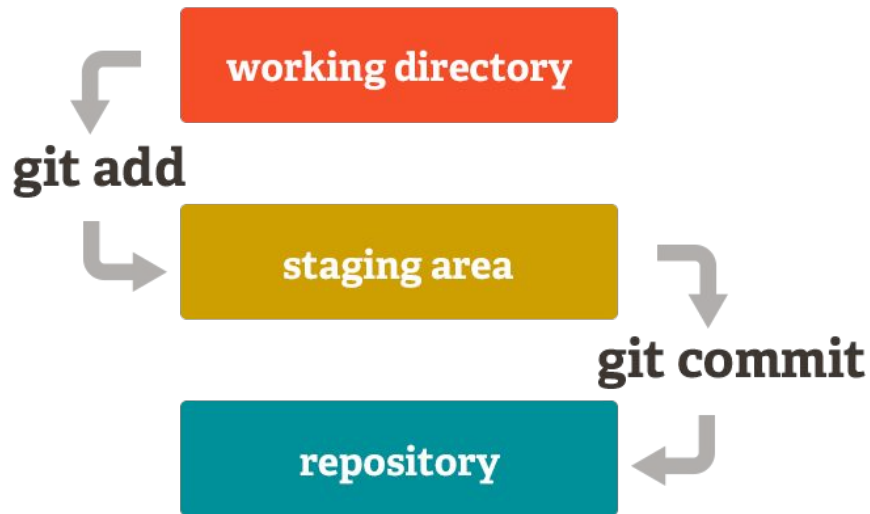
- Versiebeheersysteem
- Gedistribueerd
- Gemaakt door Linus Torvalds
  - Op de foto maakt hij zijn mening over NVIDIA duidelijk
- Onafhankelijk van elkaar aanpassingen maken en samenvoegen
- Een must-have skill voor alle IT'ers (zoals jij!)





# Basis basis git

- Working directory
- Staging
- Local repository
- Remote repository
- Branches



The image features a solid orange background. In the top-left corner, there are three vertical bars of varying heights, each composed of three overlapping circles. In the bottom-right corner, there are four vertical bars of increasing height from left to right, each also composed of three overlapping circles.

**ZO MOET HET NIET!!!!11!**



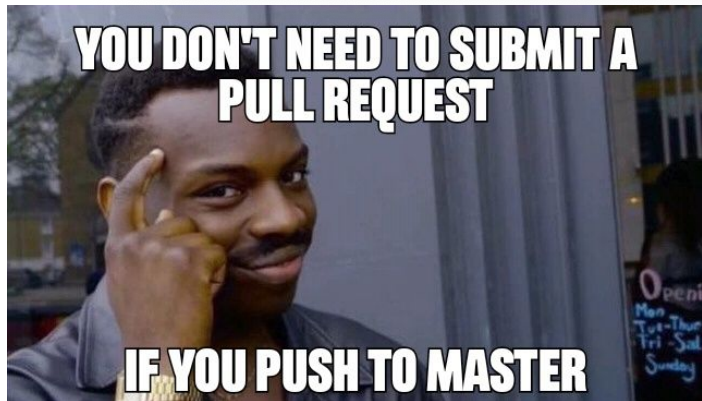


# Voorbeelden

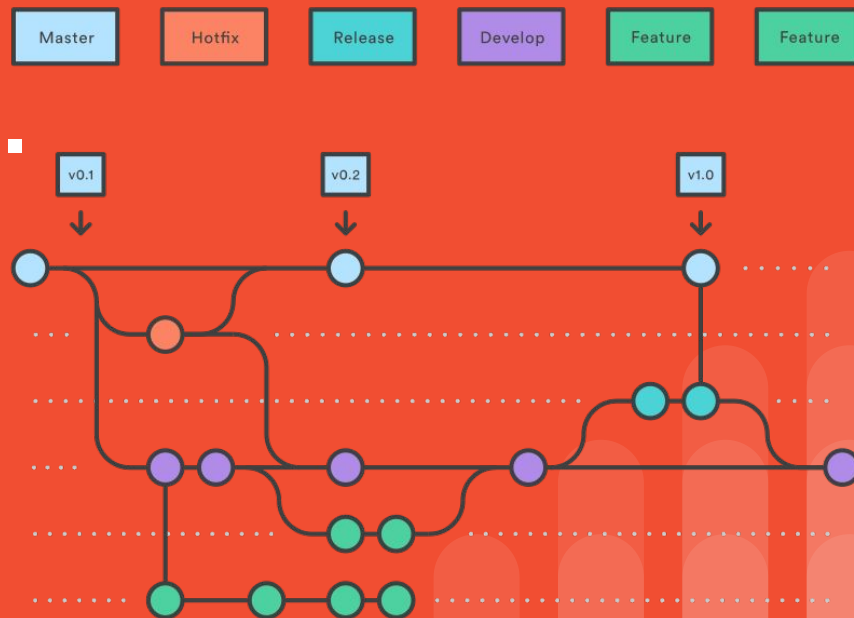
- Branches zijn geen mensen
- Direct committen op de hoofd branch
- `git commit -am "changes"`
- `git push -f`
- Multiple talen
- Heel veel verschillende dingen in 1 commit die niks met elkaar te maken hebben
- Ongerelateerde code style veranderingen

WesleyKlop wants to merge 442 commits into `develop`

```
wesley@snijplank:~/Projects/Web/git-...  
  
wesley in snijplank in git-workshop on ✎ main  
> git branch  
ben-jij-al-syntax-lid-btw  
guus  
homepage  
* main  
thom  
veranderingen  
  
wesley in snijplank in git-workshop on ✎ main  
> |  
  
🕒 03/12, 3:35 PM  📁 -zsh  ✎ main
```



# ZO KAN HET WEL. feat. Git Flow



# Over het algemeen

```
wesley@polaris: ~/Projects/Valet/wesley
wesley on develop via v14.1.0 via v7.4.5
> git gud
git: 'gud' is not a git command. See 'git --help'.

The most similar commands are
  add
  bug
  guilt
  sed
```

- Zorg dat je main branch altijd werkt en branch protection heeft
- Spreek een taal af waarin je je commit messages schrijft(NL, EN)
- Schrijf in toekomstige tegenwoordige tijd “Implement creation of users”
- Branch naar een doel (user stories)
- Commit **vaak** (het is gratis!)
- Gebruik een .gitignore!
  - .vscode, .idea
  - node\_modules, vendor
- Dwing dezelfde code style af
  - .editorconfig (basics)
  - PSR-12, PEP-8, etc...
  - Prettier

```
✖ Run yarn lint --no-fix
1 ▶ Run yarn lint --no-fix
4 yarn run v1.22.4
5 $ vue-cli-service lint --no-fix
6 error: Replace `h` with `(h)` (prettier/prettier) at src/main.ts:27:13:
7   25 |     router,
8   26 |     store,
9   > 27 |     render: h => h(App),
10      |               ^
11   28 |   }).$mount('#app')
12   29 |
13
```



## Over het algemeen (2)

- Gebruik Pull Requests (PRs)
- Doe aan Code Reviews!
- Kleine branches
  - Anders een losse “epic” branch
- Vergeet niet te pullen voordat je aan je volgende feature / bugfix begint
- Maar het belangrijkste:
  - Wees **consistent** met je team

In case of fire



1. git commit



2. git push

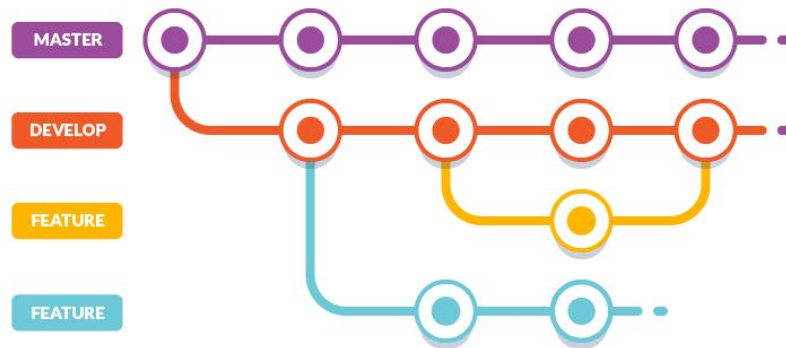


3. leave building



# Git Flow

- 2 branches
  - Main (production)
  - Develop (development)
- Main zou altijd moeten werken
- Branch op basis van develop
  - feature/implement-auth
  - feature/42-update-logo
  - bugfix/user-nullpointer-error
  - bugfix/1337-missing-translation
- Release? Merge develop in main
  - Meerdere environments (OTAP)
  - Rollbacks



Geweldige bron: <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>



# Voordelen

- Minder merge conflicts
- Meer duidelijkheid
- Mogelijkheden meer automatisering
- Geen ruzie meer
- Betere code schrijven
- Hogere ontwikkelsnelheid
- Grip op je project
- Flexibel
- Makkelijker debuggen

```
wesley@polaris: ~/Projects/ipsenh/journali-frontend
journali-frontend on  master via  v14.1.0
> git flow
usage: git flow <subcommand>

Available subcommands are:
  init      Initialize a new git repo with support for the branching model.
  feature   Manage your feature branches.
  bugfix    Manage your bugfix branches.
  release   Manage your release branches.
  hotfix    Manage your hotfix branches.
  support   Manage your support branches.
  version   Shows version information.
  config    Manage your git-flow configuration.
  log       Show log deviating from base branch.

Try 'git flow <subcommand> help' for details.
```



# Een miniem stukje theorie





# Basis git

- 4 “omgevingen”
  - working directory - De bestanden die je ziet in je git project
  - stage - De veranderingen die je gaat registreren met “git commit”
  - local repository - De lijn van veranderingen die lokaal opgeslagen staan
  - remote repository - Centrale versie van alle gepushde veranderingen
- Basic workflow
  - `git init` / `git clone` - Initialiseer of kloon een project
  - `git add` - Voeg gemaakte veranderingen toe aan je stage
  - `git commit` - Registreer je stage naar de lokale repository
  - `git push` - Push de veranderingen in je lokale repository naar het centrale git versiebeheersysteem
- Meer commando's
  - `git status` - Zie de huidige status van je project
  - `git checkout (-b)` - Wissel van branch (en maak deze eventueel aan) of bezoek een commit hash
  - `git branch` - Zie een lijst van branches
  - `git log` - Zie een log van commits



# Uitleg repository maken

1. Maak een mapje aan waar je project in komt
2. Open je project in een terminal / git bash
3. Initialiseer je project met git init
4. Maak wijzigingen zoals een README aanmaken
5. Voeg je gemaakte wijzigingen toe aan de stage ( `git add .` ) voegt alle bestanden in je huidige directory ( `.` ) recursief aan je stage toe
6. Commit je gemaakte wijzigingen (gebruik `git commit -m "Voeg README toe"` om meteen een bericht mee te geven)
7. Maak een github/gitlab/bitbucket repository aan
8. Registreer je repository als remote (`git remote add origin [url van je repo]`)
9. Push naar de repository (`git push -u origin main`)

```
wesley@polaris: ~/tof-project

~/tof-project
> git init
Initialized empty Git repository in /Users/wesley/tof-project/.git/

tof-project
> echo "# Tof project" > README.md

tof-project
> git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)

tof-project
> git add README.md

tof-project
> git commit -m "Initial commit"
[master (root-commit) 4ba15e7] Initial commit
1 file changed, 1 insertion(+)
 create mode 100644 README.md

tof-project on   master
> git remote add origin git@github.com:WesleyKlop/tof-project.git

tof-project on   master
> git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 881 bytes | 48.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:WesleyKlop/tof-project.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

tof-project on   master took 4s
>
```

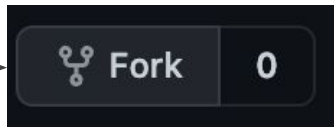


# Aan de slag



# Een kleine web applicatie

- Ik heb een git repo gemaakt: <https://github.com/WesleyKlop/git-workshop/>
- Voeg iets toe aan de repo!
  - Je naam aan de public/assets/contributors.json file
  - Fix een issue (plaats eerst een reactie om dubbel werk te voorkomen)
  - Als je een beter idee hebt, ga je gang!
- Maar hoe? 





# Tips & Tricks





# Tips & Tricks

- **git blame**
  - zie wie iets gedaan heeft
- **git fetch origin**
  - Haal de remote repository genaamd origin op
- **git bisect**
  - Zoek naar de commit die een bug geïntroduceerd heeft
- **git rebase**
  - Herbaseer jouw branch op een andere branch, “herhaalt” de commits (ongeveer)
- **git cherry-pick**
  - Haal een enkele commit van een andere branch af
- **git revert**
  - Maak een commit ongedaan
- **github actions etc.**
  - auto labels
  - testen
  - builden
  - deployen
  - changelogs
- **git tags**
  - Checkpoints op bepaalde commits
- **commit hooks**
  - husky
- **git flow plugin**
  - `brew install git-flow(-avh)`
  - `apt install git-flow`
  - Ingebouwd met Windows git client

Vaak oefenen, dan git je vanzelf gud

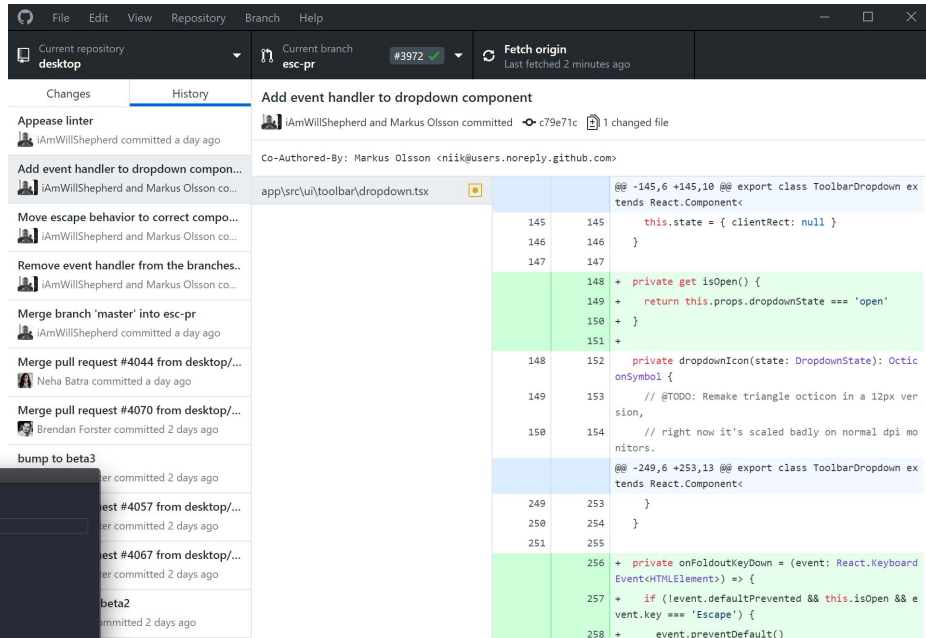
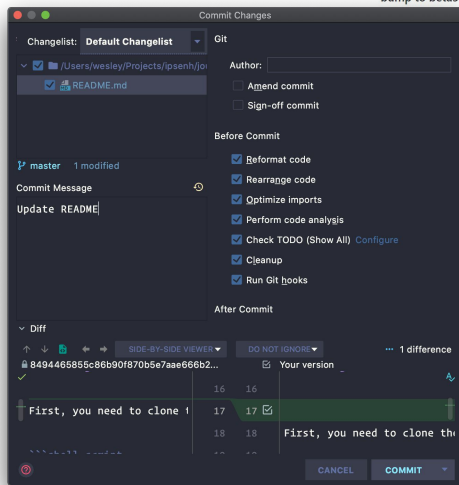


# Geen CLI fan?

Veel IDE's hebben git support ingebouwd

Git Kraken

Github desktop





**Vragen?**

