

# **Digitale Werkomgeving 1**

## **Git en GitHub**

# GIT

# Versiebeheersysteem

(= Version Control System - VCS)

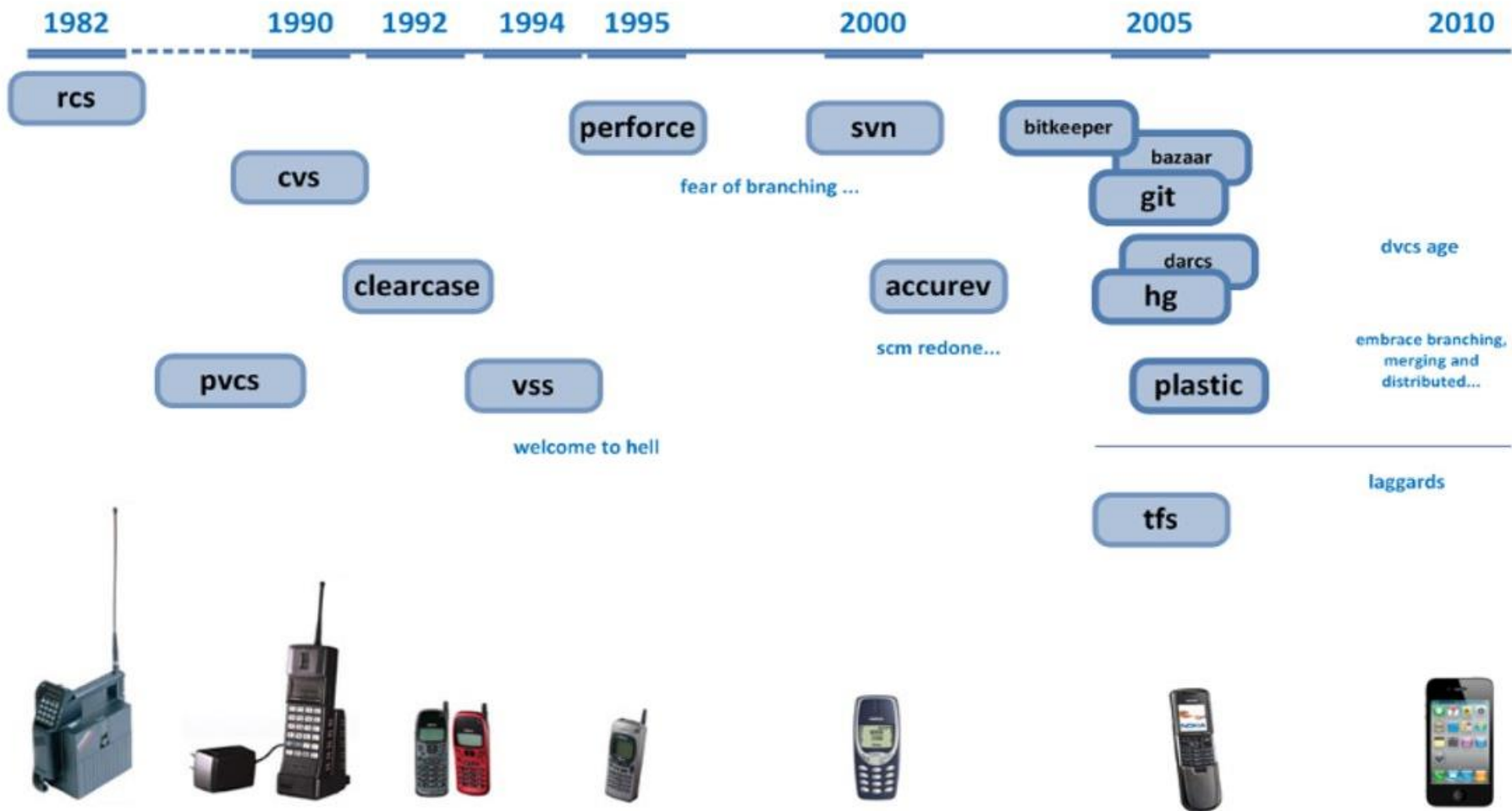
- verschillende versies van bestanden opslaan
- bekijken wie, wat, wanneer gewijzigd heeft
- specifieke wijzigingen ongedaan maken  
(teruggaan naar bepaalde versie van een bestand)

# Versiebeheersysteem

Ideaal voor source code

Ideaal om samen te werken

Ideaal als er aan verschillende versies/features  
in parallel gewerkt wordt (m.b.v. branches)



# Welke VCS kiezen?

Professional Developers

Learning to Code

53,374 responses

Git



96.65%

SVN



5.96%

I don't use one



1.38%

Mercurial



1.22%

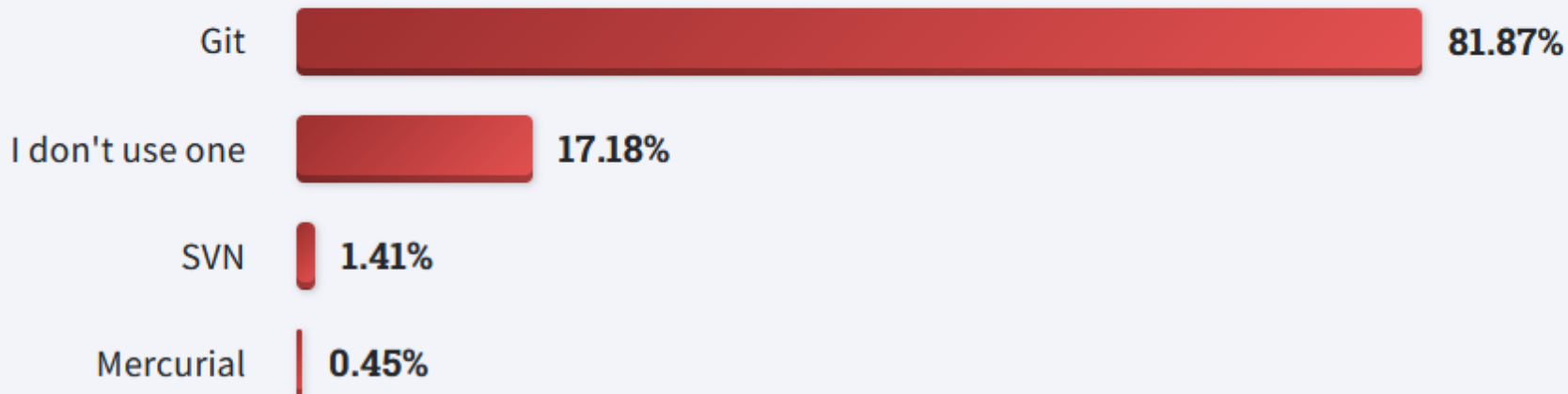
GIT

# Welke VCS kiezen?

Professional Developers

Learning to Code

6,157 responses



# Git

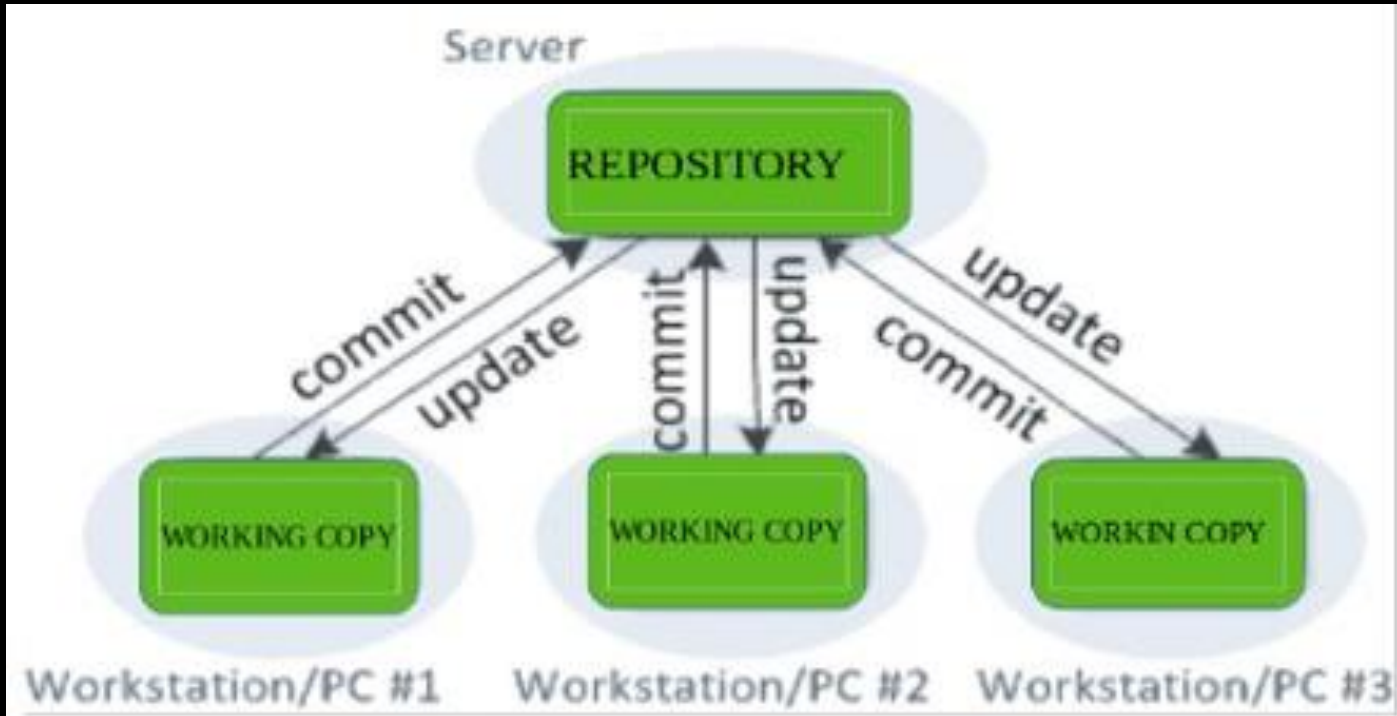
Git is een gedistribueerd versiebeheersysteem (VCS).

Elke gebruiker heeft een lokale kopie van de repository.

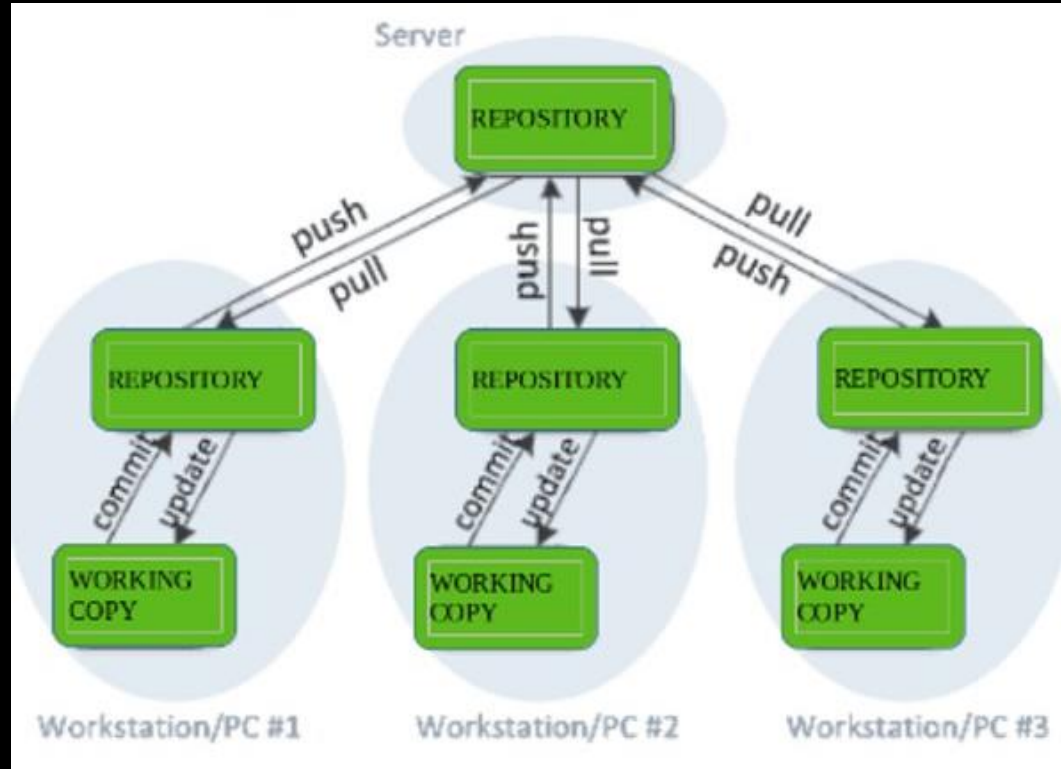
repository = verzameling source code-bestanden



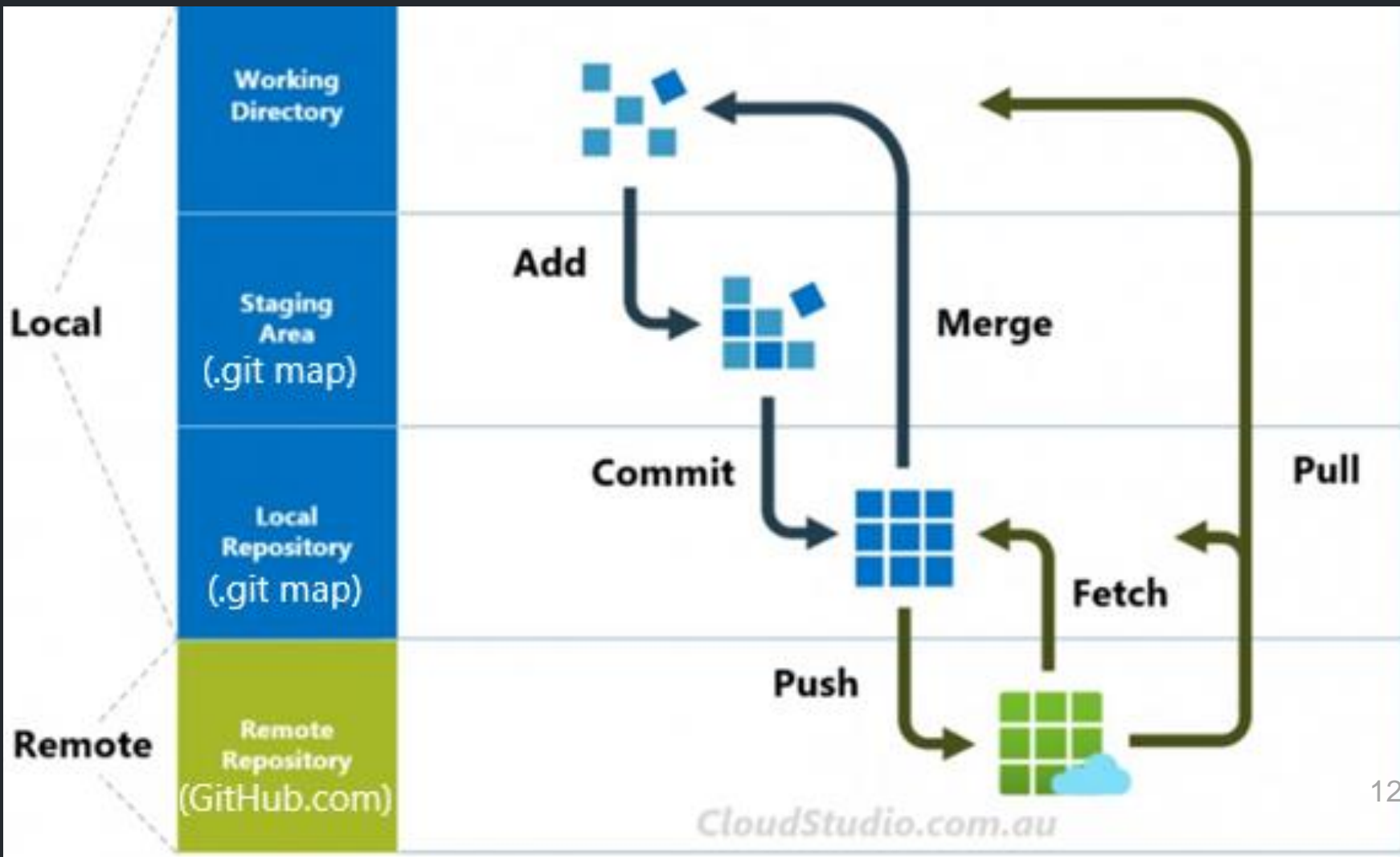
# (oudere) gecentraliseerde VCS'en



# Gedistribueerde VCS'en (bvb. Git)



# Git Workflow



# Toestanden van een bestand

- untracked
- modified
- staged
- committed

## Toestanden van een bestand

- **Untracked:** Dit is de toestand van een bestand dat nog niet is toegevoegd aan de Git-repository. Het bestand wordt niet bijgehouden door Git en wordt niet opgenomen in commits. Wanneer je een nieuw bestand aanmaakt in een Git-repository wordt het als "untracked" gemarkeerd.
- **Modified:** Dit is de toestand van een bestand dat is gewijzigd nadat het aan de Git-repository is toegevoegd, maar nog niet is opgenomen in de staging area met "git add". Git ziet de wijzigingen in het bestand, maar deze zijn nog niet opgeslagen in de repository.
- **Staged:** Dit is de toestand van een bestand dat is toegevoegd aan de staging area met het "git add" commando. Het bestand is gemarkeerd als gereed voor een commit en de wijzigingen zijn klaar om permanent te worden opgeslagen in de (lokale) Git-repository.
- **Committed:** Dit is de toestand van een bestand dat is opgeslagen in de Git-repository met een commit. De wijzigingen in het bestand zijn permanent opgeslagen in de repository en kunnen worden bekeken of teruggenomen in de toekomst.

## **Commando's** (Commandprompt)

- git add
- git commit
- git push
- git pull

# Git ≠ GitHub

Git:

Version Control System (VCS) software

GitHub:

Hosting voor centrale Git-repo's

(alternatieven: GitLab, BitBucket, Beanstalk, Codebase, ...)

# Maak een GitHub-account

(of gebruik een bestaande account)

<https://github.com/>

- Denk na over je username  
(bvb: goedertw, DavidBr89, lucvervoort, adsr, jmchen28, grosal, alejandro5042, protobear, liegebeest, ...)
- Kies een sterk wachtwoord  
(<https://passwordsgenerator.net/>)



Als je reeds een GitHub-account hebt, mag je die gebruiken. Als je er nog geen hebt, maak er dan een aan via <https://github.com> >> “Sign up”.

Denk even na over de gebruikersnaam die je gaat gebruiken, want die zal publiek zichtbaar zijn. Misschien ga je later ook naar je GitHub-account verwijzen als referentie op je sollicitatiebrief. Als je later in een bedrijf werkt met tientallen ontwikkelaars is het aangewezen om in je naam ook een **identificatie** te steken, zoals je familienaam/voornaam combinatie.

- Voorbeeld: GitHub-naam van Wim Goedertier is “goedertw”)
- Tegenvoorbeeld: GitHub-naam van Joris Custers is “liegebeest”)

Tip voor het genereren van een sterk paswoord: <https://passwordsgenerator.net/>.

Eens je een GitHub-account hebt, en je hebt je HOGENT-adres toegevoegd als één van je e-mailadressen, dan kan je via <https://education.github.com/students> een “**Student Developer Pack**” aanvragen. Dit is niet nodig voor de les, maar bevat wel een aantal interessante voordelen.

# Installeer Git

Download via

<https://gitforwindows.org>

en installeer.



Kies overal voor de "default" instellingen.  
Kies dus steeds gewoon "Next".

# Test de installatie

Start "Opdrachtprompt" (cmd)  
en voer volgende instructie uit:

```
C:\Users\els>git --version  
git version 2.42.0.windows.2
```

```
C:\Users\els>
```

# Configuratie

## Vertel Git wie je bent (via CMD)

```
git config --global user.name <jouw-github-username>  
git config --global user.email <jouw-mail-adres>  
git config --global core.editor notepad  
git config --global --list
```

# Configuratie

In plaats van je HOGENT-mailadres kan je ook een “no-reply”-mailadres van GitHub.com instellen als “**user.mail**”. Dit is vooral nuttig bij "public" repo's!

Je vindt dit no-reply-adres  
via <http://github.com/> >> profiel-icoontje rechtsbovenaan >> Settings >> Emails.

Daar vind je bvb. iets van de vorm “33165762+goedertw@users.noreply.github.com”.

Gebruik bij “**core.editor**” een tekst-editor die zeer “licht” is en zeer snel opstart. Deze editor wordt alleen maar gebruikt om (in sommige situaties) “commit-messages” in te tikken. “notepad” is daarom een zeer goede keuze.

# Git-repo gebruiken

1. We maken een lege repo op GitHub.com
2. We "clonen" die repo op onze laptop
3. We maken een tekstbestand in die repo
4. We plaatsen dat bestand onder versiebeheer ("add" + "commit")
5. We "pushen" de wijziging naar GitHub.com
6. We controleren op GitHub.com

## 1. Lege repository aanmaken via <http://github.com/>

Ga via <http://github.com/> >> profiel-icoontje [rechtsbovenaan](#) >> [“Your repositories”](#).




Klik daarna op  en kies een naam. Kies voor een [“public”](#) repo en [“Add a README file”](#).

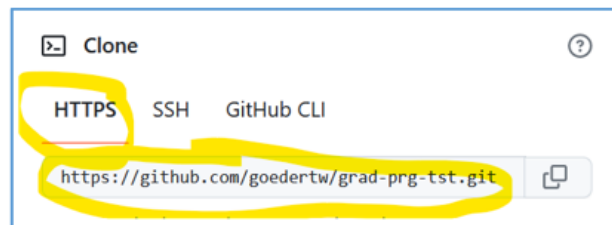
Gebruik deze [repo](#) *enkel* voor de opdrachten in deze les.

Als je nog extra dingen verder wilt uitproberen met Git en GitHub, doe dat dan een aparte [repo](#).

## 2. Maak een lokale kloon van je nieuwe repo

Ga via <http://github.com/> >> profiel-icoontje rechtsbovenaan >> "Your repositories" naar de

nieuwe, lege repo. Klik daarna op  en kopieer de "connection-string"



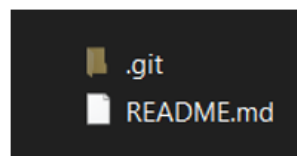
Ga, via de command-line, naar de folder waar je de repo wilt zetten.

**Opgelet:** Deze folder zit best **niet** onder OneDrive of DropBox !!

```
C:\Users\els\dw1> git clone <connection-string-voor-het-clonen>
```

Dit maakt een directory met de naam van je repository. (Je kan deze directory verplaatsen en hernoemen. Dit heeft geen invloed op de naam van je repository.)

In deze directory staat er reeds 1 bestand: README.md. Dit is omdat we dit aangevinkt hebben bij het aanmaken van de repo. Als je niks aanvinkte dan heb je uiteraard geen README.md.

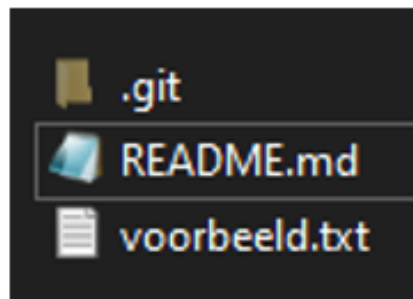


In deze directory vind je ook een "hidden" subdirectory ".git". Deze ".git"-directory bevat de lokale kopie van de volledige repository op GitHub. In de ".git"-directory worden ook alle oude versies bewaard, en ook alle commit-messages, allebei in een gezippt formaat. Deze ".git"-directory mag je dus zeker niet wissen.



### 3. Maak een tekst-bestand in die repo

Maak naast de “.git”-directory en het “README.md”-bestand nog een extra bestand “voorbeeld.txt” (gebruik exact “voorbeeld.txt”, gebruik geen andere naam).



Zet enkele willekeurige lijnen tekst in “voorbeeld.txt”

Bekijk de situatie:

```
C:\xxx> git status
```

## 4. Plaats de nieuwe bestanden onder versiebeheer

Nieuw bestand (of wijziging) klaarzetten in de staging area:

```
C:\xxx> git add voorbeeld.txt
```

Bekijk het resultaat:

```
C:\xxx> git status
```

Lokaal committen:

```
C:\xxx> git commit -m "<korte-zinnige-boodschap>"
```

Bekijk het resultaat:

```
C:\xxx> git status
```

## 5. Nieuw bestand “pushen” naar GitHub.com

```
C:\xxx> git push
```

Bekijk het resultaat:

```
C:\xxx> git status
```

## 6. Resultaat controleren op GitHub.com

# Help!

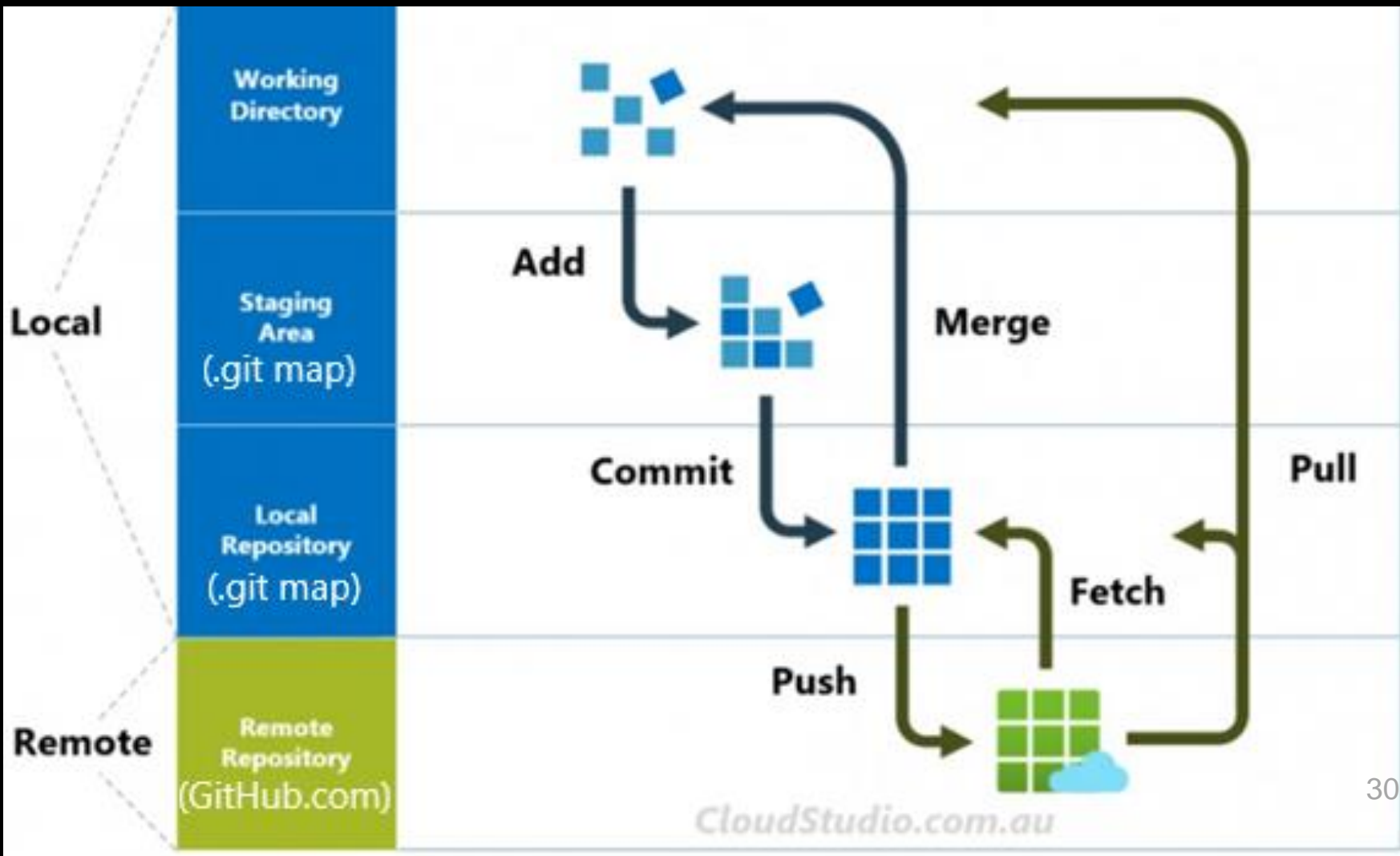
```
C:\xxx> git <commando> -h
```

Ofwel:

```
C:\xxx> git <commando> --help
```

Dit opent op Windows een “man-page” in je browser.

# Git Workflow



# Typische Git-workflow

1. `git pull` (zeker zijn dat we de laatste versie van GitHub.com hebben)
2. `git status` (zeker zijn dat er geen lokale wijzigingen zijn)
3. bestand aanpassen
4. `git add <bestandsnaam>` (klaarzetten in staging area)
5. `git commit -m "<message>"`
6. `git push`

# Enkel source-code committen

## → .gitignore

1. maak een bestand "tst.bin" en "tst.exe"
2. `git status` (wat zie je ?)
3. maak een bestand ".gitignore" met als inhoud:  
    \*.bin  
    \*.exe
4. `git status` (wat zie je ?)
5. `git add .gitignore`
6. `git commit -m "ignore *.bin en *.exe"`
7. `git push`

## Commits bekijken

Om te zien wie-wat-wanneer gewijzigd heeft, zijn volgende commando's handig:

```
git log
git whatchanged
git blame <filename>
```

Als je vastzit na één van deze commando's, druk je "q" (van quit)

Enkele van mijn favoriete varianten:

```
git log --pretty="%h (%ad) %aN: %s%C(red)%d" --date=relative
git log --pretty="%h (%ad) %aN: %s" --date=iso
git whatchanged --pretty="%h (%ad) %aN: %s" --date=iso
git show <commit-fingerprint>
git blame -s <filename>
git blame --date=short <filename>
```



# Samenwerken / Branching / Merge-conflicten

Het werken met meerdere gebruikers,  
het werken met verschillende branches  
en het oplossen van merge-conflicten  
leren we in **Digitale Werkomgeving 2**

# Grafische Git-clients

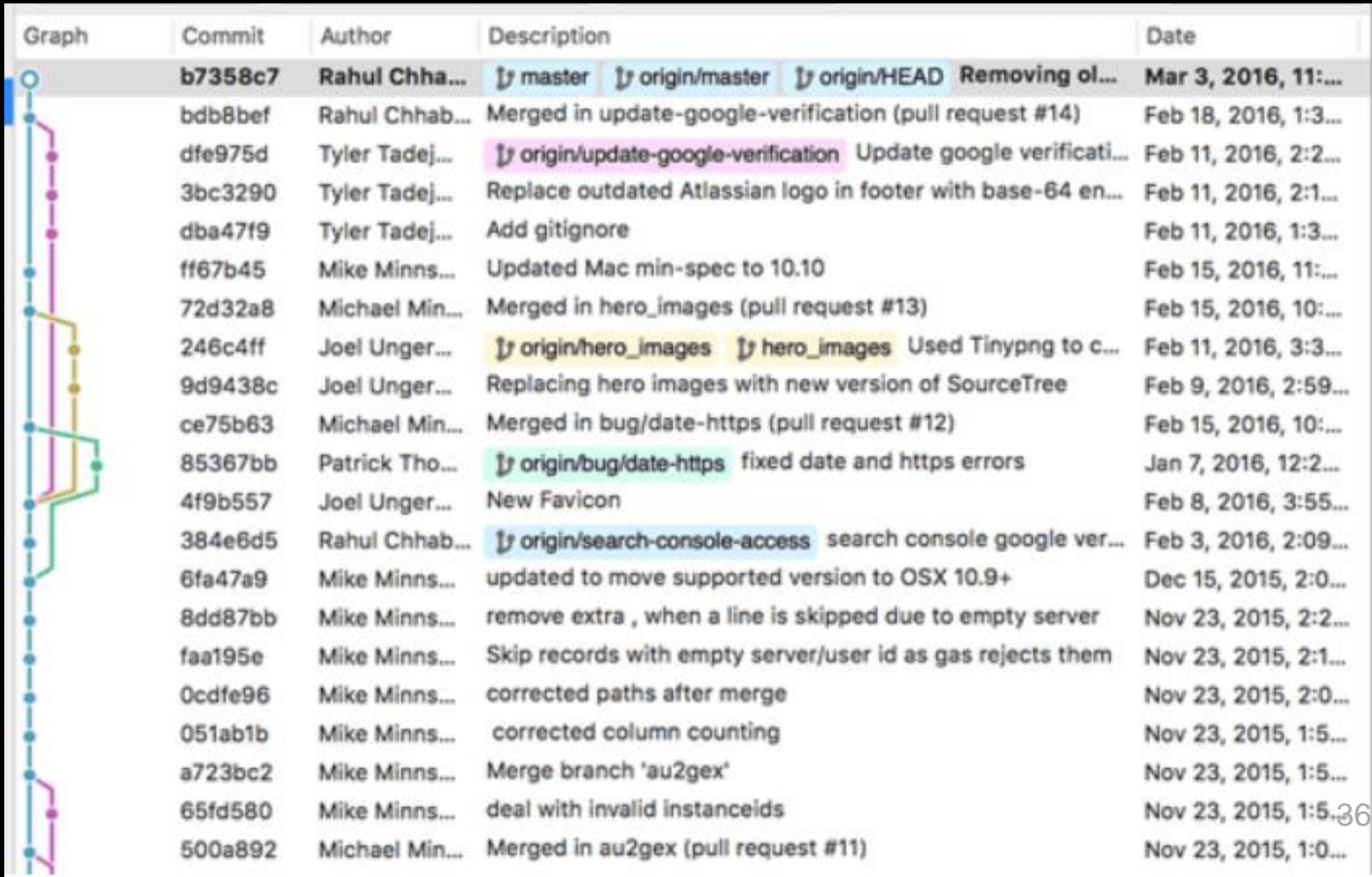
Sommige dingen worden gemakkelijker

Visuele voorstelling van branches is zeer handig

Nadeel: men weet minder goed wat men juist doet

- Sourcetree (werkt goed, volledig gratis)
- GitHub Desktop (nogal beperkt)
- GitKraken (gratis versie beperkt tot public repo's)

# Codetree: visualisatie branching



# Opdracht (algemeen)

Maak een nieuwe GitHub-repo en gebruik deze voor alle individuele codeer-taken van de opleiding Graduaat Programmeren

- Programmeren Basis
- Web 1
- Programmeren Gevorderd
- Web 2
- ...

# Opdracht (details)

Maak een nieuwe repo (via github.com)

1. kies "Private"
2. voeg "README" toe
3. voeg ".gitignore" toe voor "VisualStudio"
4. voeg daarna "goedertw" toe als "Collaborator" (via "Settings")

Op je laptop:

1. maak een clon (git clone ...)
2. maak 2 nieuwe mappen "prog1" en "web1" (via Verkenner)
3. verplaats de ".gitignore" naar "prog1" (git mv .gitignore prog1) en commit
4. verplaats je VisualStudio-project-folders naar "prog1" (+ add/commit)
5. zet minstens 1 html-file in de map "web1"
6. maak indien nodig een passende ".gitignore" in de "web1"-map

Commit en synchroniseer (push) regelmatig je werk met github.com