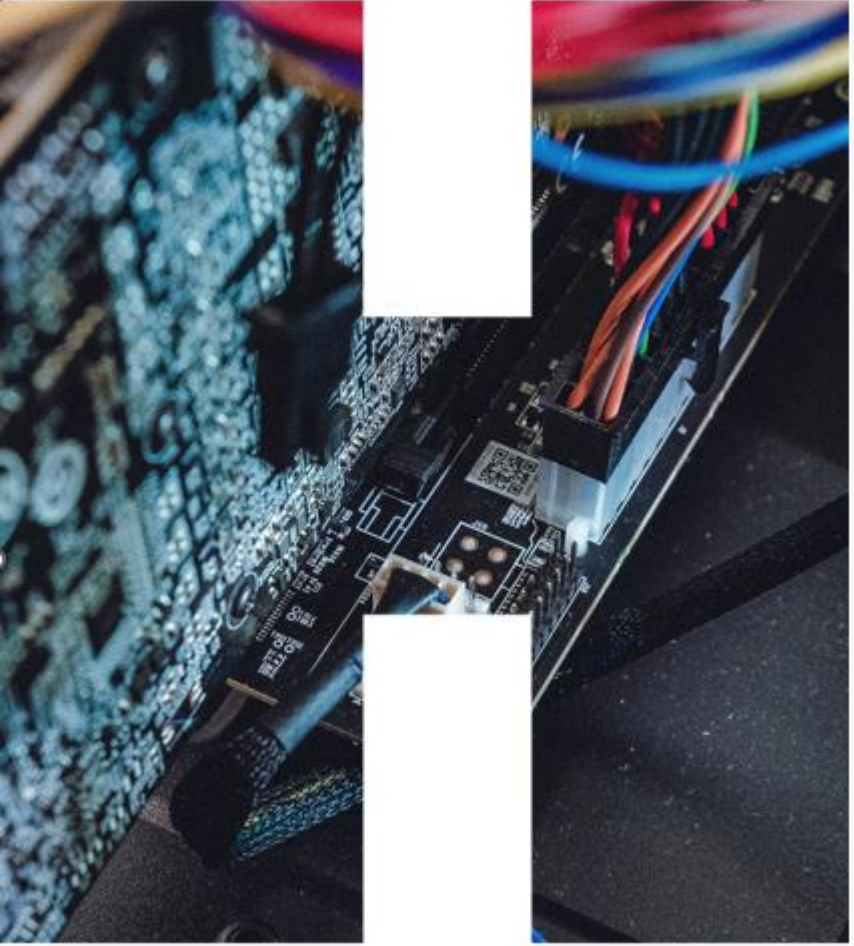
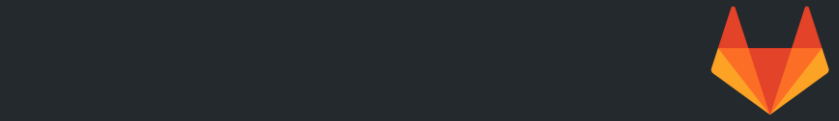


Digitale Werkomgeving 1

Git en GitHub



GIT

Versiebeheersysteem

(= Version Control System - VCS)

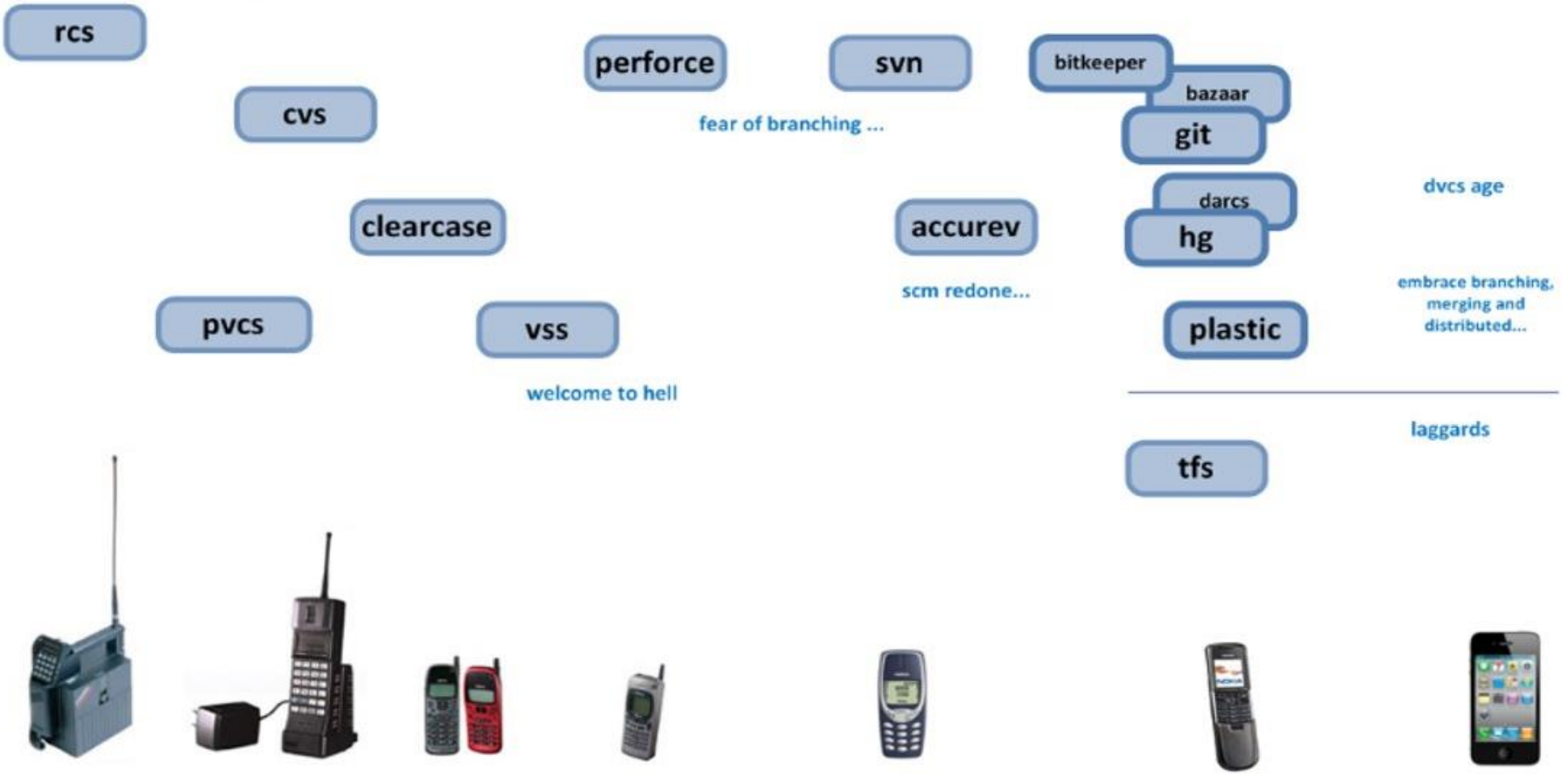
- verschillende versies van bestanden opslaan
- bekijken wie, wat, wanneer gewijzigd heeft
- specifieke wijzigingen ongedaan maken (teruggaan naar bepaalde versie van een bestand)

Versiebeheersysteem

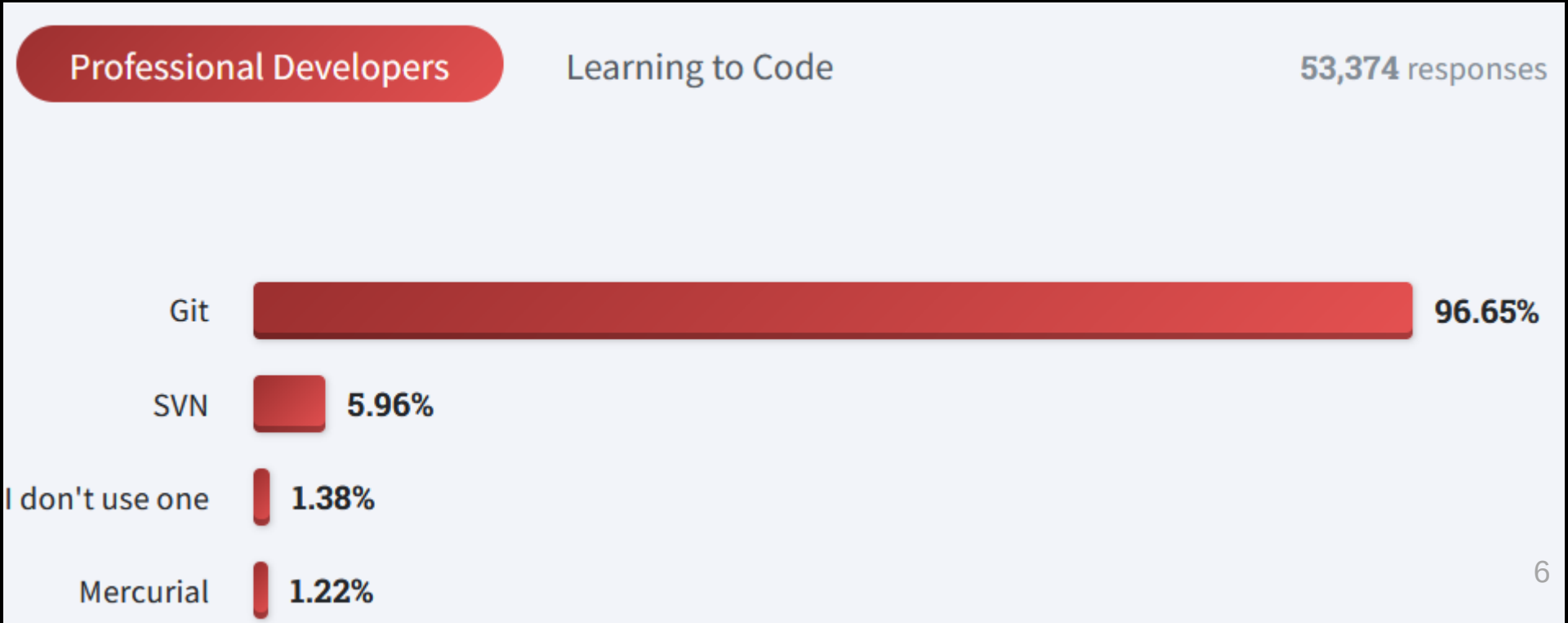
Ideaal voor source code

Ideaal om samen te werken

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

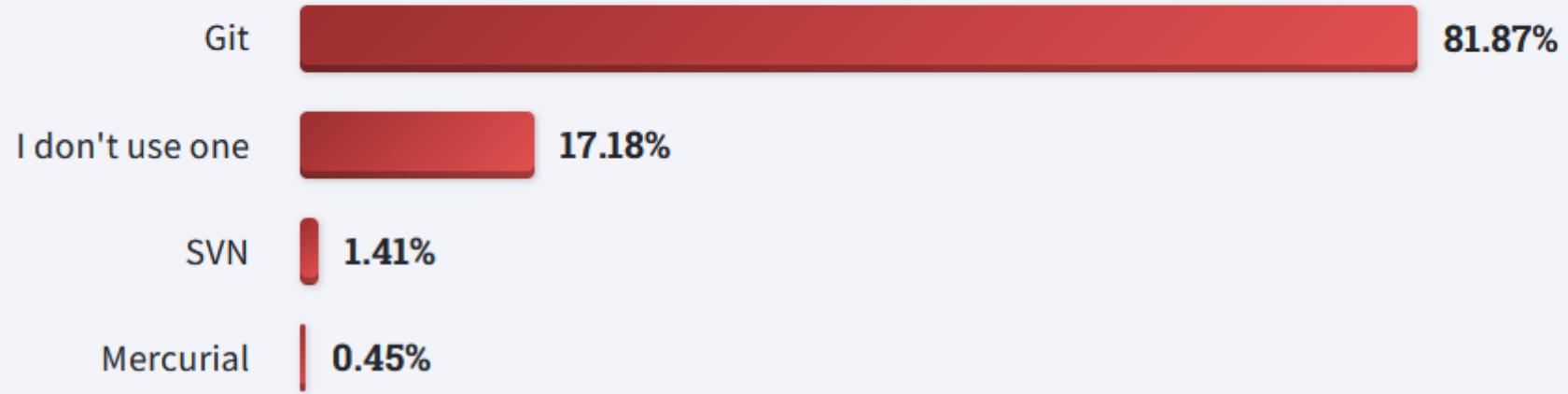

1982
1990
1992
1994
1995
2000
2005
2010
rcs
cvs
clearcase
pvcs
vss
welcome to hell
perforce
fear of branching ...
svn
accurev
scm redone...
bitkeeper
bazaar
git
darcs
hg
dvcs age
embrace branching, merging and distributed...
plastic
laggards
tfs

Welke VCS kiezen?



Welke VCS kiezen?

Professional Developers | Learning to Code | 6,157 responses



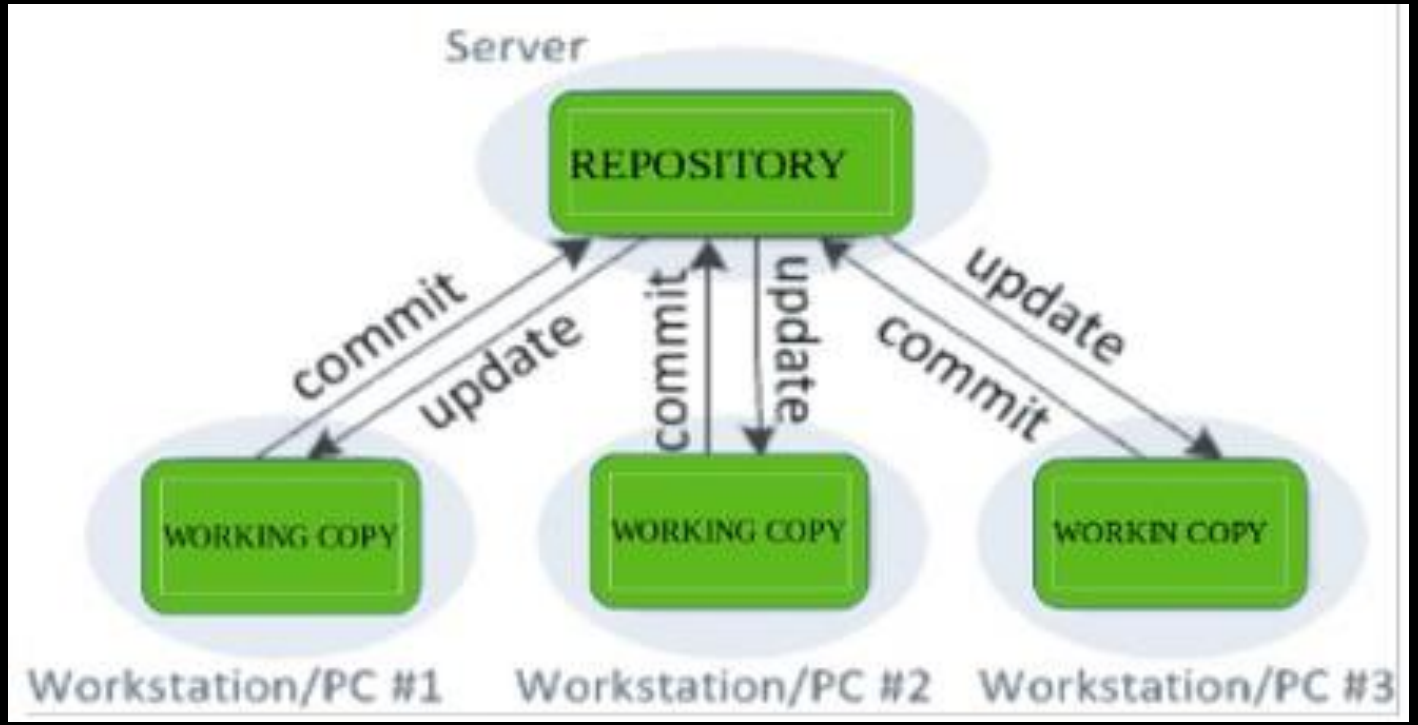
Git

Git is een **gedistribueerd** versiebeheersysteem (VCS).

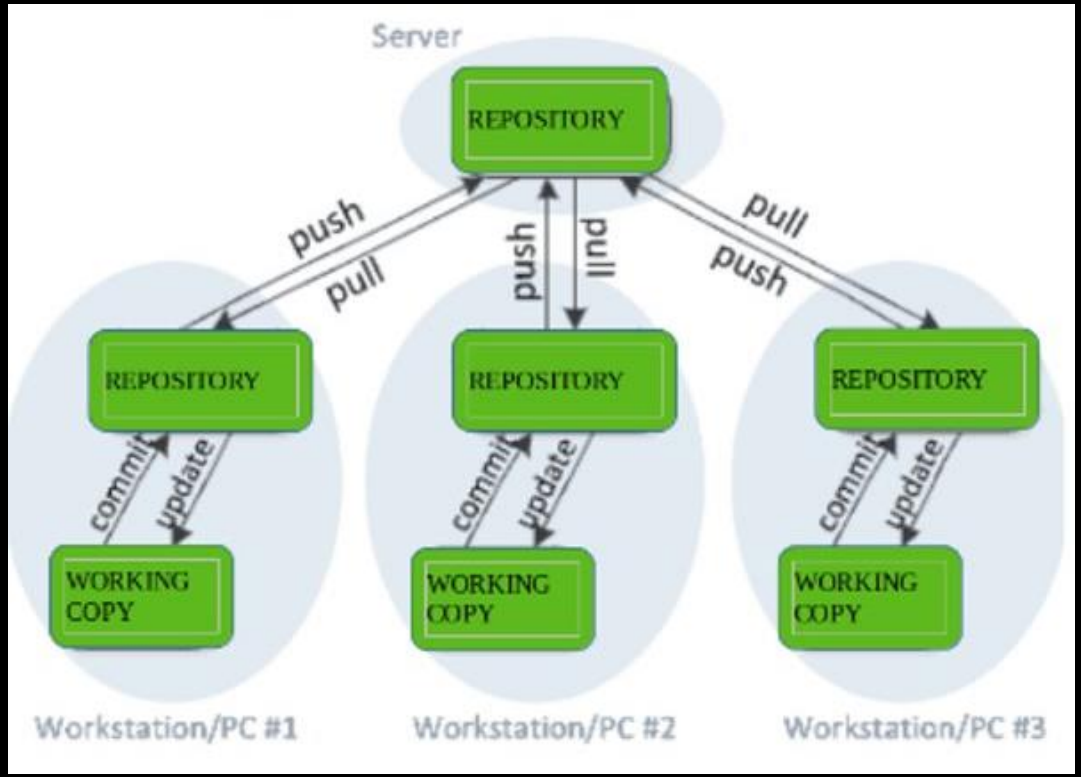
Elke gebruiker heeft een **lokale kopie** van de repository.

repository = verzameling **source code**-bestanden

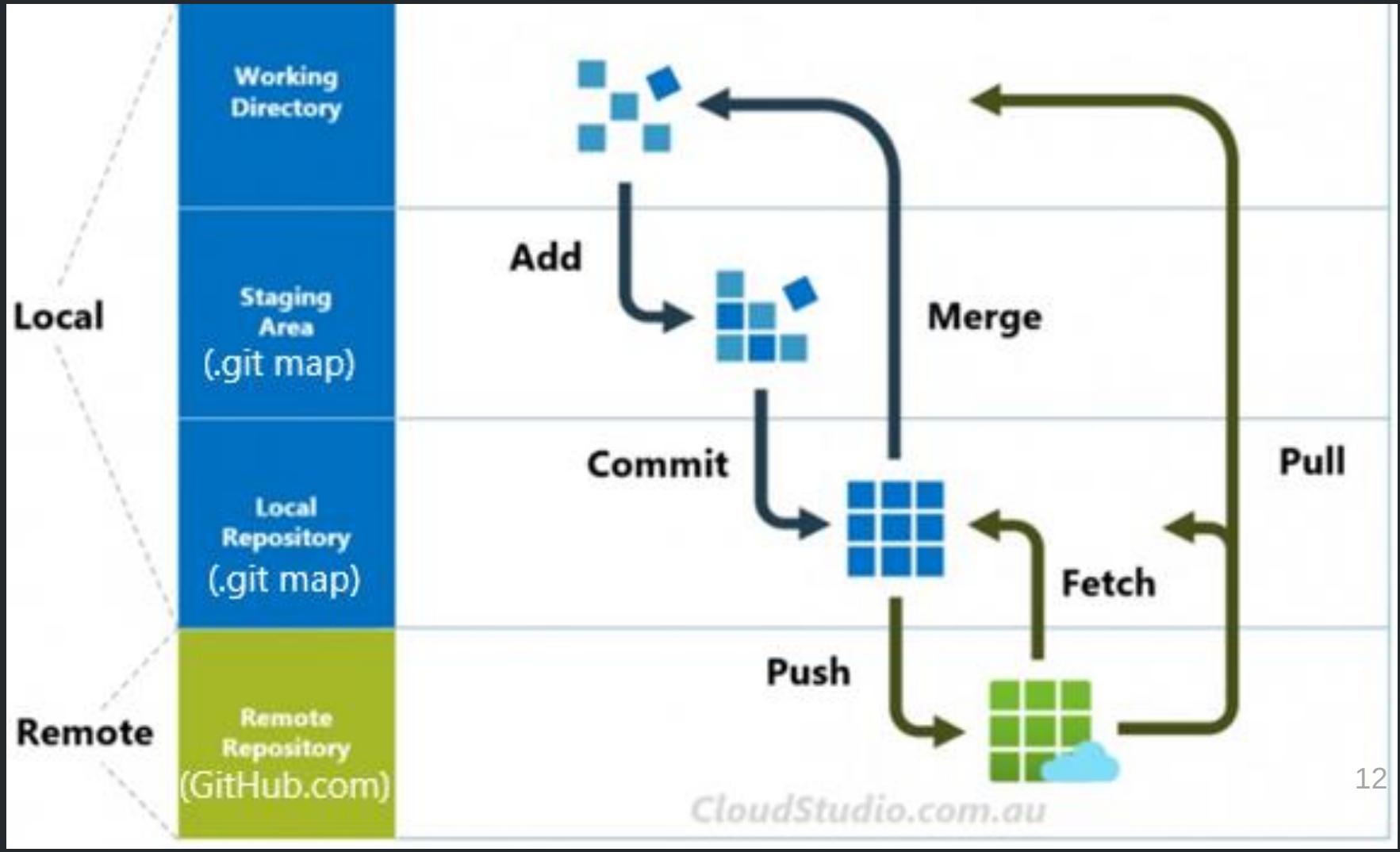
(oudere) gecentraliseerde VCS'en



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, grosa1, 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 “331885762+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.

Opdr 1a: Git-repo maken & gebruiken.

Overzicht stappen (details op volgende slides)

1. Maak een lege repo op GitHub.com (naam naar keuze)
2. "Kloon" de repo op je laptop (*niet* in je OneDrive-map)
3. Maak een file "voorbeeld.txt" in je gekloonde repo
4. Plaats "voorbeeld.txt" onder versiebeheer ("add" + "commit")
5. "Push" de wijziging naar GitHub.com
6. Bekijk/controleer op GitHub.com

1. Lege repository aanmaken via http://github.com/

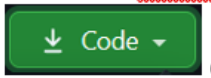
Ga via http://github.com/ >> profiel-icoontje rechtsbovenaan >> “Your repositories”.

Klik daarna op 

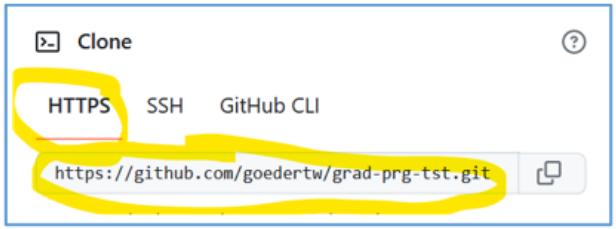
en kies een naam. Kies voor een “**public**” repo en “**Add a README file**”.

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”

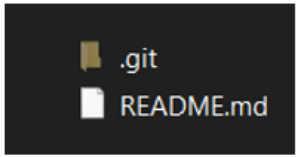


Ga, via de command-line, naar de folder waar je de repo wilt zetten.
Opgelet: Deze folder zit best **niet** onder **OneDrive** of DropBox !!

```
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.)

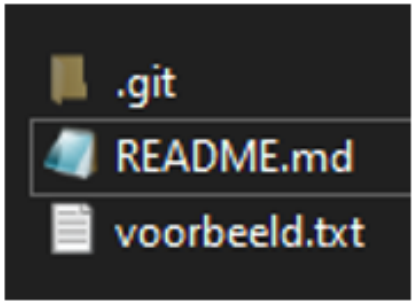
In deze directory staat er reeds 1 bestand: README.md. Dit is omdat we dit aangevinkt hebben bij het aanmaken van de repo. Als je niks aanvinkte dan heb je uiteraard geen README.md.



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 gezipt 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-zinvolle-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!

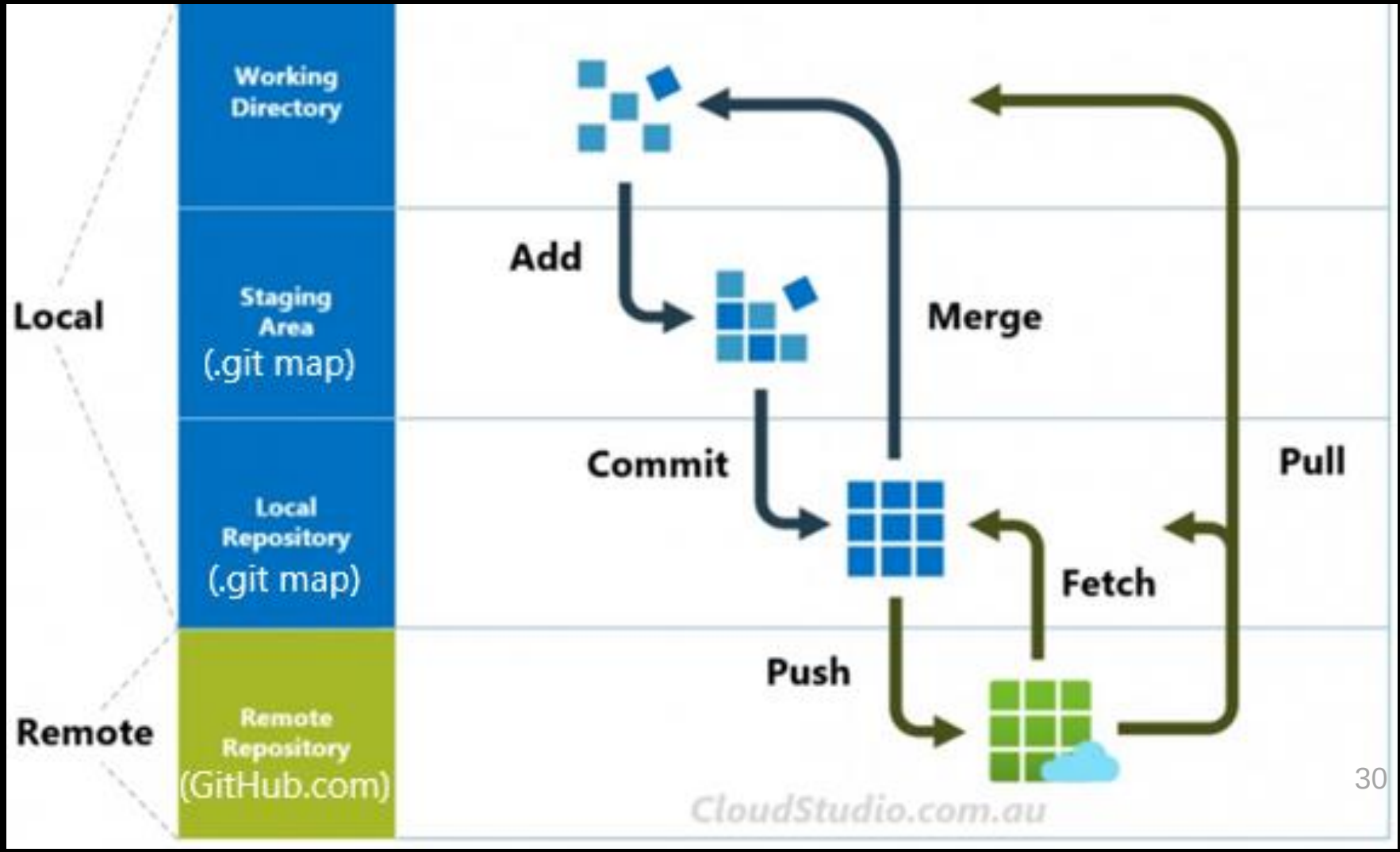
```
C:\xxx> git <commando> -h
```

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. code schrijven/bugs fixen/... : of dus "bestanden wijzigen"
4. `git add <bestandsnaam>` (klaarzetten in staging area)
5. `git commit -m "<message>"`
6. `git push`

Opdr 1b: .gitignore gebruiken

(om bepaalde files *niet* op Github te zetten)

1. zet "`04-git.pdf`" in je gekloonde repo
2. maak een bestand "`hello-world.exe`"
3. `git status` (wat zie je ?)
4. maak een bestand "`.gitignore`" met als inhoud:

```
*.exe
*.pdf
```

5. `git status` (wat zie je ?)
6. `git add .gitignore`
7. `git commit -m "ignore *.exe en *.pdf"`
8. `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>
```

Opletten bij hernoemen, verplaatsen en verwijderen

Eens een bestand (of folder) gekend is door "Git", kan je het beter niet meer zomaar hernoemen, verplaatsen of verwijderen.

Gebruik hiervoor de Git-commando's:

- **`git mv ...`** (doe evt. `git mv --help`)
- **`git rm ...`** (doe evt. `git rm --help`)

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

Leer eerst Git gebruiken op de command-line!
Daarna

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)

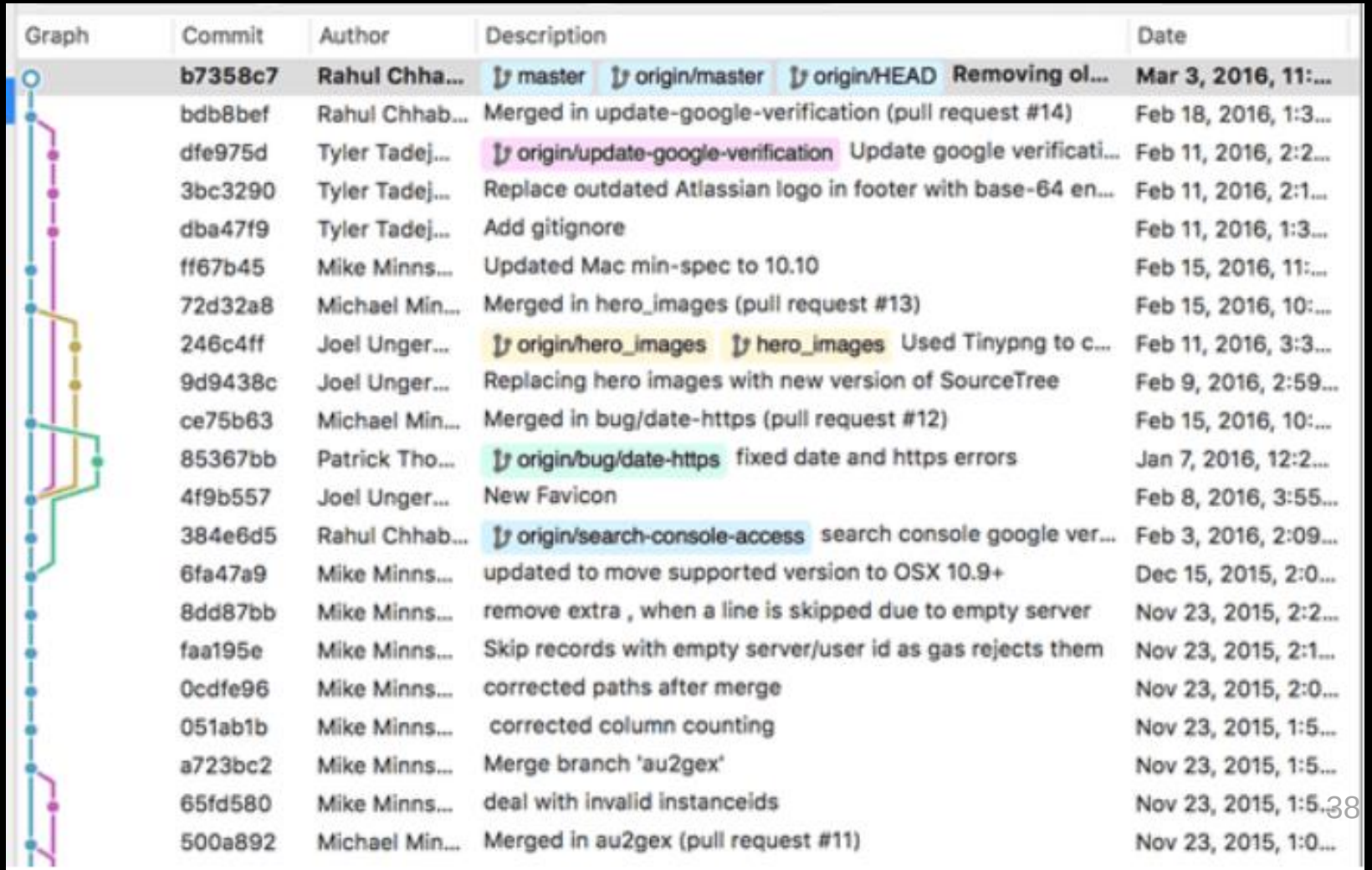
Grafische Git-clients

Leer eerst Git gebruiken op de command-line!
Daarna

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)

Sourcetree: visualisatie branching



Graph	Commit	Author	Description	Date
	b7358c7	**Rahul Chha...**	**master** **origin/master** **origin/HEAD** **Removing ol...**	**Mar 3, 2016, 11:...**
	bdb8bef	Rahul Chhab...	Merged in update-google-verification (pull request #14)	Feb 18, 2016, 1:3...
	dfe975d	Tyler Tadej...	origin/update-google-verification Update google verificati...	Feb 11, 2016, 2:2...
	3bc3290	Tyler Tadej...	Replace outdated Atlassian logo in footer with base-64 en...	Feb 11, 2016, 2:1...
	dba47f9	Tyler Tadej...	Add gitignore	Feb 11, 2016, 1:3...
	ff67b45	Mike Minns...	Updated Mac min-spec to 10.10	Feb 15, 2016, 11:...
	72d32a8	Michael Min...	Merged in hero_images (pull request #13)	Feb 15, 2016, 10:...
	246c4ff	Joel Unger...	origin/hero_images hero_images Used Tinypng to c...	Feb 11, 2016, 3:3...
	9d9438c	Joel Unger...	Replacing hero images with new version of SourceTree	Feb 9, 2016, 2:59...
	ce75b63	Michael Min...	Merged in bug/date-https (pull request #12)	Feb 15, 2016, 10:...
	85367bb	Patrick Tho...	origin/bug/date-https fixed date and https errors	Jan 7, 2016, 12:2...
	4f9b557	Joel Unger...	New Favicon	Feb 8, 2016, 3:55...
	384e6d5	Rahul Chhab...	origin/search-console-access search console google ver...	Feb 3, 2016, 2:09...
	6fa47a9	Mike Minns...	updated to move supported version to OSX 10.9+	Dec 15, 2015, 2:0...
	8dd87bb	Mike Minns...	remove extra , when a line is skipped due to empty server	Nov 23, 2015, 2:2...
	faa195e	Mike Minns...	Skip records with empty server/user id as gas rejects them	Nov 23, 2015, 2:1...
	0cdfe96	Mike Minns...	corrected paths after merge	Nov 23, 2015, 2:0...
	051ab1b	Mike Minns...	corrected column counting	Nov 23, 2015, 1:5...
	a723bc2	Mike Minns...	Merge branch 'au2gex'	Nov 23, 2015, 1:5...
	65fd580	Mike Minns...	deal with invalid instanceids	Nov 23, 2015, 1:5...
	500a892	Michael Min...	Merged in au2gex (pull request #11)	Nov 23, 2015, 1:0...

Opdracht 2 (algemeen)

We gaan een nieuwe GitHub-repo maken en gebruiken voor alle individuele codeer-taken van de opleiding Graduaat Programmeren

- Programmeren Basis
- Web 1
- Programmeren Gevorderd
- Web 2
- …

Opdracht 2 (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 lokale copy (git clone ...), best niet in een OneDrive-folder
2. maak 2 nieuwe mappen "progbasis" en "web1" (via CMD)
3. verplaats de ".gitignore" naar "progbasis" (git mv .gitignore progbasis)
4. maak minsten 1 C#-project in "progbasis" (of verplaats al je huidige VisualStudio-project-folders naar "progbasis"
5. zet minstens 1 html-file in de map "web1"
6. maak indien nodig een passende ".gitignore" in de "web1"-map

Vergeet geen "git status", "git add", ..., én "git push" te doen!!