

Continuous Integration Basics

"A commit message shows whether a developer is a good collaborator"

Goede afspraken rond commit messages (stijl, content & metadata):

- Teamwork!!
- Log wordt overzichtelijk en bruikbaar instrument
- Makkelijk om te zien wat wanneer door wie is aangepast (en waarom) = context
- Projecten hebben vaak lange levensduur (maanden tot vele jaren...)

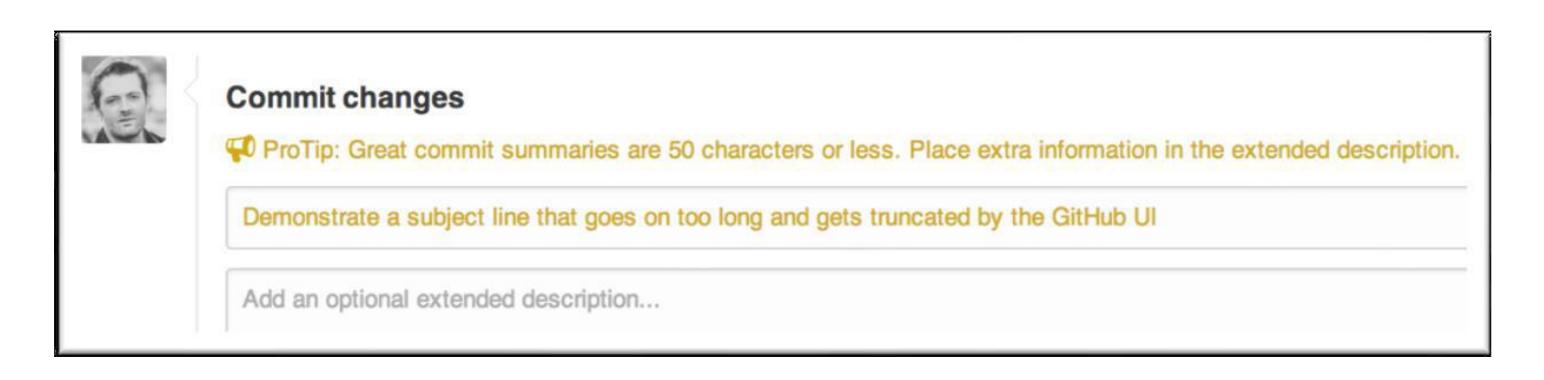
```
$ git log
$ git log --oneline
$ git shortlog
```

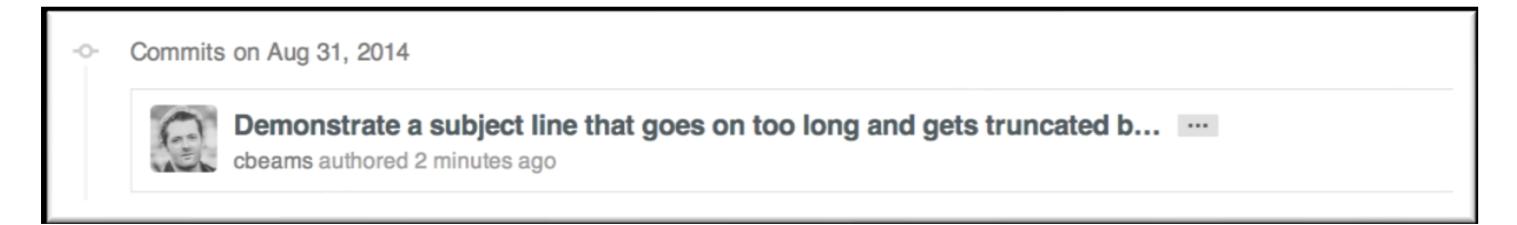


Basisregels voor commit message (titel)

- 1. Maak je commit message in het **Engels**
- 2. Beperk de titel tot 50 karakters (maar beschrijf toch zo goed mogelijk de change!)
- 3. Begin de titel met een hoofdletter
- 4. Eindig de titel niet met een punt of ander leesteken
- 5. Gebruik de **gebiedende wijs**







Goede commit message	GEEN goede commit message
Refactor subsystem X for readability	Subsystem refactored
Update getting started documentation	docs
Remove deprecated methods	Various cleanup.
Release version 1.1.0	release new version
Fix bug with login screen	Fixed bug with login screen
Ideaal: meerdere goede commits (1 per fix!)	More fixes for broken stuff



Indien je ook de body gebruikt (voor extra uitleg)

- 1. Hier geen gebiedende wijs.
- 2. Wrap body zelf op 72 karakters.
- 3. Leg uit waarom, niet hoe (dat zie je in de code zelf!)

```
$ git commit -m "Fix typo on login screen"
$ git commit -m "Titel" -m "Body"
$ git commit
```

Enkel titel

Titel en body

Titel en/of body (via editor)



Commit message via editor (bv. Visual Studio Code):

Remove portal to the underworld



Apply primal spells, rephrase incantations because the spawn was too powerful Close #666

Body

Titel en body gescheiden door lege lijn



	COMMENT	DATE
Q	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
φ	ENABLED CONFIG FILE PARSING	9 HOURS AGO
o	MISC BUGFIXES	5 HOURS AGO
φ	CODE ADDITIONS/EDITS	4 HOURS AGO
Q.	MORE CODE	4 HOURS AGO
Ò	HERE HAVE CODE	4 HOURS AGO
þ	AAAAAAA	3 HOURS AGO
Ø.	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
\(\rightarrow \)	MY HANDS ARE TYPING WORDS	2 HOURS AGO
þ	HAAAAAAAANDS	2 HOURS AGO

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

Blijf volharden en aandacht besteden aan goede commit messages!!



hogeschool

Gitignore

Continuous Integration Basics

Doel van .gitignore

Niet altijd gewenst om alle bestanden op te nemen in versiebeheer, bv:

- Configuratiebestanden (wachtwoorden, keys, ...)
- Bepaalde folders, by met gecompileerde binaries (.vs, bin, obj, ...)
- Zaken die voor jezelf van belang zijn maar niet voor anderen

Deze bestanden/mappen steeds manueel uitsluiten bij stagen is veel werk en foutgevoelig!

.gitignore -> AUTOMATISCH en ALTIJD genegeerd



Waar plaats je .gitignore?

CONVENTIE = 1 .gitignore bestand in ROOT van repository

Technisch mogelijk om meerdere .gitignore bestanden te gebruiken in je repo (buiten scope van de cursus)



Hoe maak je .gitignore?

Gewoon tekstbestand met naam ".gitignore"

!! Opgelet: ZONDER extensie

Inhoud regel	Betekenis	Voorbeeld
Lege regel	Wordt genegeerd	
Regel die begint met #	Commentaar	# Ignore all PDF files
Exacte bestandsnaam (incl. extensie)	Bestand genegeerd (eender waar!) Opgelet: matcht ook op eventuele folders met deze exacte naam	manual.pdf
Naam van een folder (eindig met /)	Folder genegeerd (eender waar!)	bin/
* (in combinatie met bovenstaande)	* = wildcard (reeks ongedefinieerde karakters)	*.pdf (alle pdf bestanden) draft/*.xlsx (alle Excel bestanden in map 'draft')
Regel die begint met!	Omgekeerd: neem WEL op in versiebeheer, ondanks genegeerd door andere regel	*.pdf (alle pdf bestanden negeren) !docs.pdf (behalve specifieke bestand 'docs.pdf')



Demo: .gitignore

Zie opdracht op Leho

!! Belangrijk:

Eens bestand/map opgenomen is in versiebeheer heeft .gitignore hier **geen invloed** meer op! We starten dus met een **nieuwe repo** (bestanden downloaden via Leho)

Denk goed na over je .gitignore vanaf het moment dat je een repo aanmaakt en telkens wanneer je nieuwe bestanden toevoegt! **LET OP met commando's zoals** git add .

