



# Conventies commit messages

Continuous Integration Basics

# Conventies rond commit messages

---

*“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
```

# Conventies rond commit messages

---

## 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**

# Conventies rond commit messages





## Commit changes


 **ProTip:** Great commit summaries are 50 characters or less. Place extra information in the extended description.

Demonstrate a subject line that goes on too long and gets truncated by the GitHub UI

Add an optional extended description...

 Commits on Aug 31, 2014



**Demonstrate a subject line that goes on too long and gets truncated b...** 

cbeams authored 2 minutes ago

# Conventies rond commit messages

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
<b><i>Ideaal: meerdere goede commits (1 per fix!)</i></b>	More fixes for broken stuff

# Conventies rond commit messages: body

---

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"
```

Enkel titel

```
$ git commit -m "Titel" -m "Body"
```

Titel en body

```
$ git commit
```

Titel en/of body (via editor)

# Conventies rond commit messages: body

---

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

Remove portal to the underworld

Titel

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

Body

Titel en body gescheiden door **lege lijn**

# Conventies rond commit messages

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	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!!**





# Gitignore

Continuous Integration Basics

# Doel van .gitignore

---

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

- Configuratiebestanden (wachtwoorden, keys, ...)
- Bepaalde folders, bv 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**

**!!! Opgelet:** ROOT = map waar .git folder staat  
**NIET IN** je .git folder

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