Git via command-line

Tim Van den Eynde

ULYSSIS

2 april 2015

Wat is git?

ULYSSIS

- Geschreven door Linus Torvalds, uit onvrede met bestaande systemen
- Gedistribueerd versiebeheersysteem
- Snel!
- Alternatieven: svn, mercurial...

Git via command-line

ULYSSIS

- Windows: git-scm.com
- OS X: brew of bovenstaande website
- Linux en BSD: via package manager

Structuur en naamgeving

ULYSSIS

- Veranderingen groeperen in een *commit*
- Git houdt *diffs* bij voor elke commit
- Commits volgen elkaar op in een boomstructuur, een tak hiervan heet een *branch*
- Branches kunnen op elk moment ontstaan of mergen
- Twee stappen: lokaal en *remote*

Structuur en naamgeving (2)

ULYSSIS

- Alles is een verwijzing naar een commit
- De *index* is de kennis die git heeft over de *working directory*
- `git status` toont het verschil tussen de working directory en de index
- *HEAD* verwijst naar de huidige commit waarop gewerkt wordt
- Als HEAD niet naar een branch verwijst, spreken we van een *detached HEAD*
- Relatieve verwijzingen met $^\wedge$ en $\sim$

Een repository maken

ULYSSIS

- `git init`: initieer een git repository in de huidige map
- Enkel lokaal
- Git trackt standaard geen files

Een commit maken

ULYSSIS

- Mini-demo!
- Git *trackt* alleen files die toegevoegd zijn
- Files *stagen*: `git add`
  = toevoegen aan de volgende commit
- Commit finaliseren: `git commit`
- Mini-demo die wél iets doet!

Wat hebben we net gedaan?

ULYSSIS

- `git add`: voeg een file toe aan de index of update de inhoud van een tracked file
- `git commit`: maak de diff tussen HEAD en de index en voeg deze samen in een commit. Dit zal:
  - HEAD verplaatsen naar de nieuwe commit
  - Als HEAD verwees naar de top van een branch, ook deze branch verplaatsen

Commits bekijken

ULYSSIS

- Op basis van:
  - commit hash
  - HEAD
  - branch
- `git show`: toon commit message en veranderingen van een gegeven commit
- `git log`: toon commitgeschiedenis
- `git diff`: toon het verschil tussen twee states

Werken met remote

ULYSSIS

- Tot nu toe: alles lokaal
- Pushen naar remote: `git push`
- Waar is remote?
  - `git remote add origin` *url*: handmatig URL toevoegen
  - `git clone` *url*: download een remote repository
- Pullen van remote: `git pull`

`git add`

ULYSSIS

- `git add` neemt ook directories aan als argument
- `.gitignore`
- `git add -n`: dry-run
- `git add -u`: update index van tracked files

`.gitignore`

ULYSSIS

- Beschrijft welke bestanden `niet` toegevoegd mogen worden door git
- `.gitignore` geldt voor zijn map én alle subfolders
- Uitzonderingen: begin met een ! om een pattern toch toe te laten
- Git houdt *diffs* bij, maar dit werkt enkel goed bij plaintext bestanden
- Met een goeie `.gitignore` kom je vaak toe met `git add .`

`git commit`

ULYSSIS

- `git commit -a`: voeg alle **tracked** files toe (hetzelfde als eerst `git add -u`)
- `git commit -m`: voeg een inline commit message toe
- `git commit --amend`: voeg veranderingen toe aan de laatste commit

`git log`

ULYSSIS

- `git log --oneline`: zet commit hash + message op één regel
- `git log --graph`: toont de graph tussen branches
- `git log --decorate`: toont labels voor de verschillende branches

Branches: de basis

ULYSSIS

- Branches zijn verwijzingen naar een commit
- `git branch`: branches beheren
- `git checkout` *`identifier file`*: updatet files in de working directory om te matchen met een commit
  - Indien geen identifier: HEAD
  - Indien geen file: current working directory
- `git checkout -b` *`branch`*: `git branch` *`branch`* `&& git checkout` *`branch`*
- `git branch -d`: verwijdert een branch lokaal

Merging

ULYSSIS

- `git merge` *`identifier`*: merget een commit en zijn parents in de huidige branch
- In praktijk is dit (bijna) altijd een branch
- Indien deze branch volgt op de huidige: fast-forward
- Anders: creëert een merge commit met twee parents: één op elke branch

Conflicts

ULYSSIS

- Veranderingen in hetzelfde bestand worden slim samengevoegd
- Veranderingen op dezelfde regel $\Rightarrow$ conflict!
- Plaintext of met een merge tool (meld, je IDE...): `git mergetool`

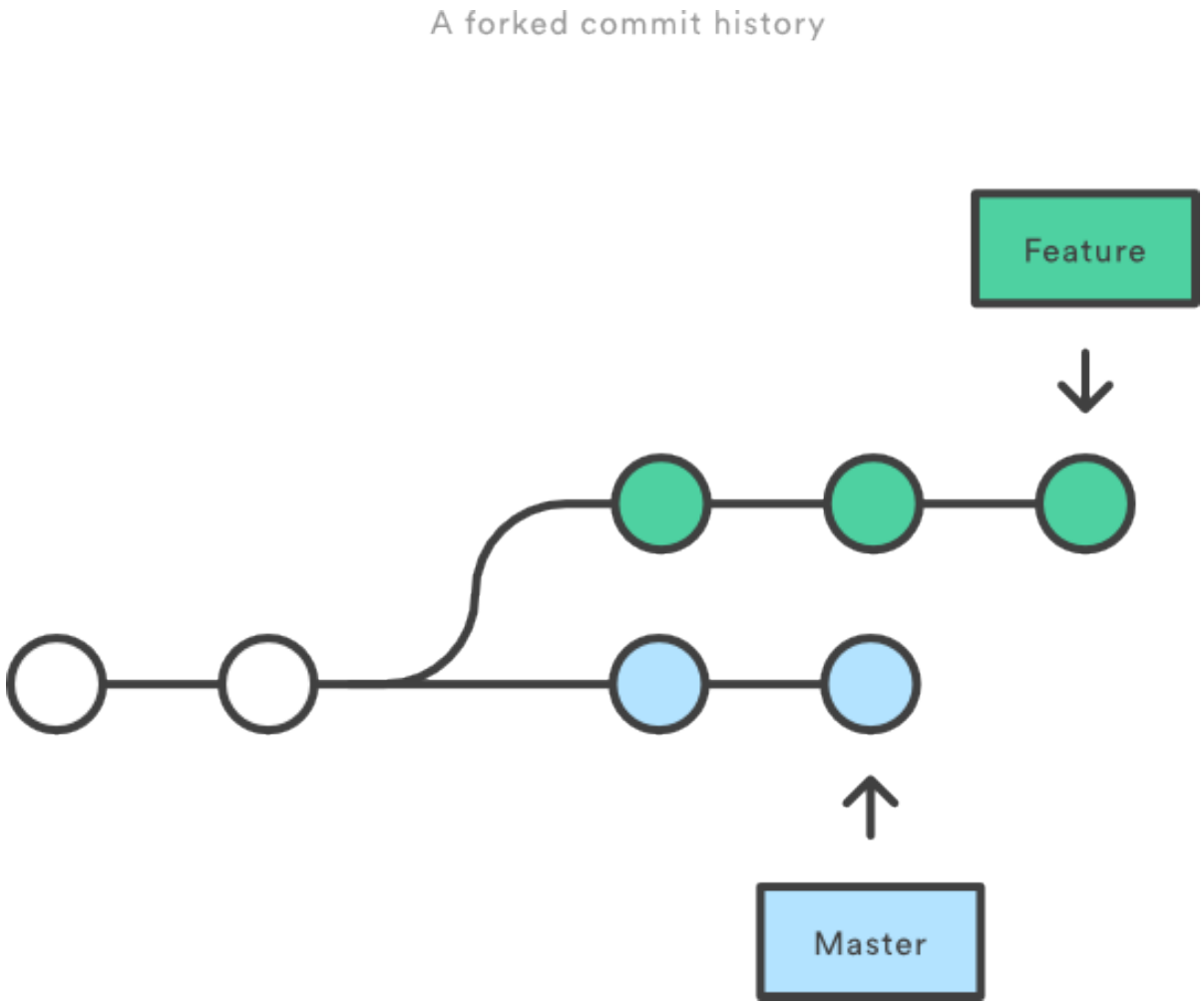
Resetting

ULYSSIS

- Laat de current branch (en HEAD) naar iets anders wijzen
- `git reset --hard`: volledige reset, ook van de working directory
- `git reset --soft`: het verschil tussen oude en nieuwe HEAD is *staged*
- `git reset --mixed`: het verschil tussen oude en nieuwe HEAD blijft *unstaged* staan in de working directory

Wat gebeurt er bij een merge?

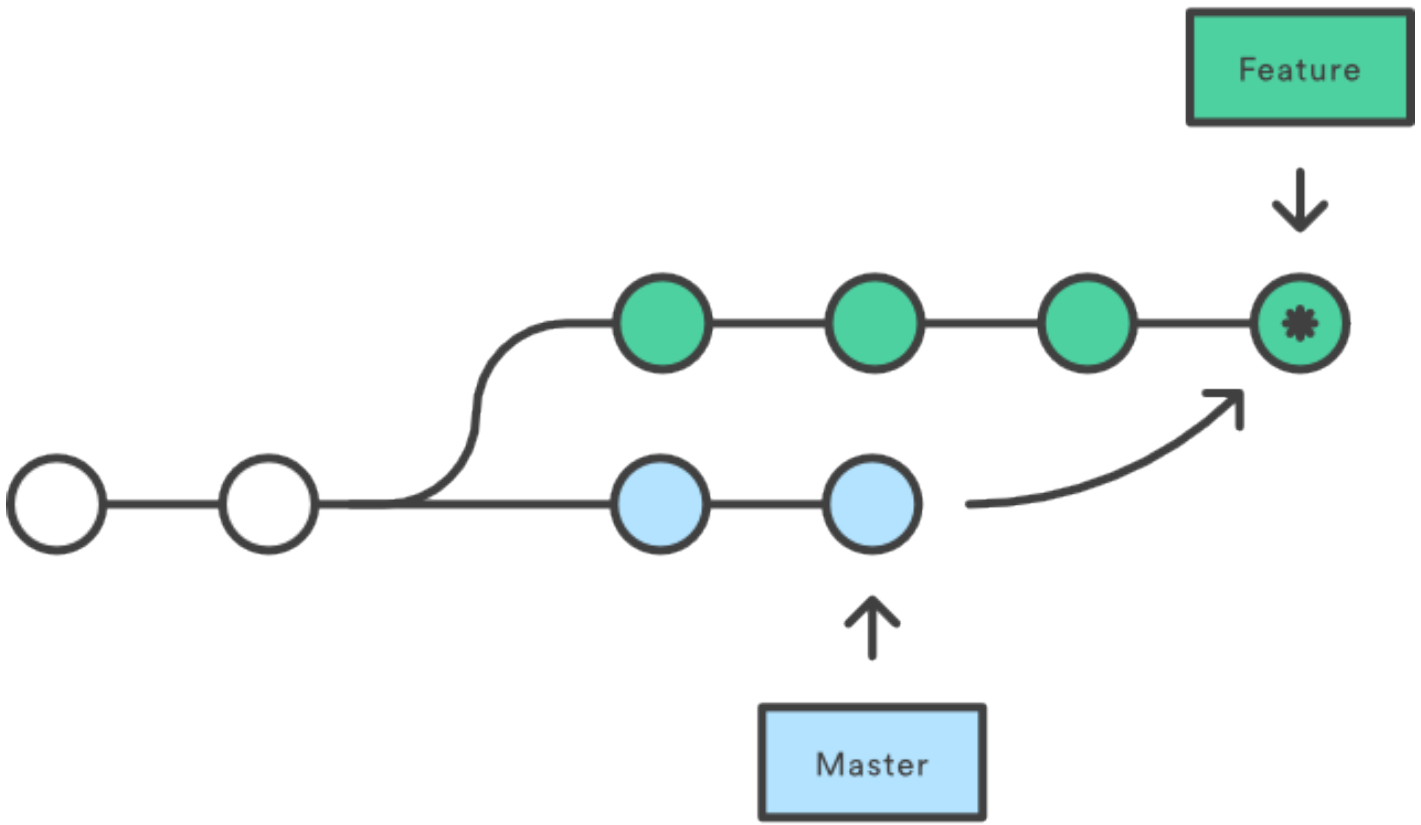
ULYSSIS



Wat gebeurt er bij een merge?

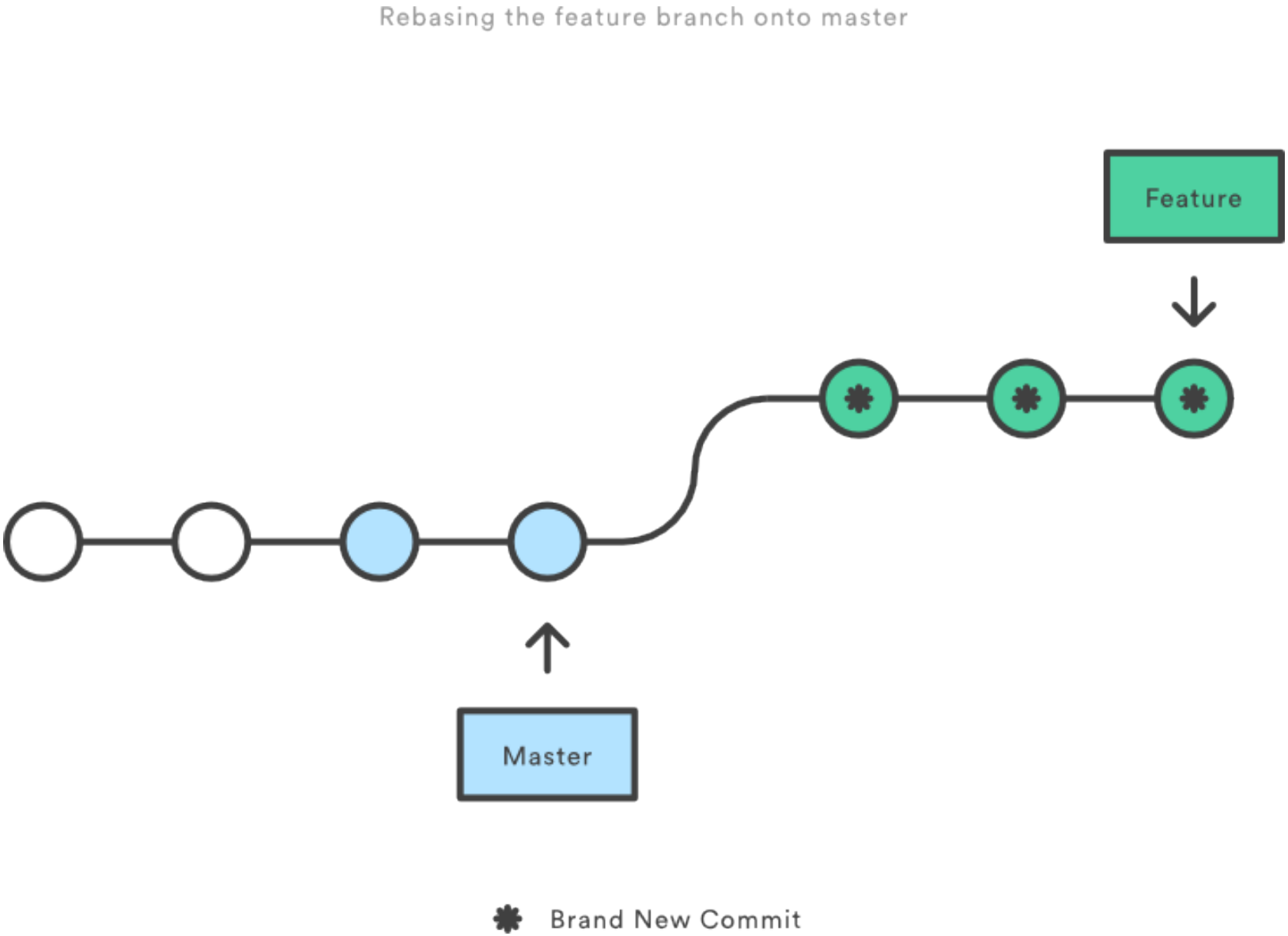
ULYSSIS

Merging master into the feature branch



Wat gebeurt er bij een rebase?

ULYSSIS



Rebasing: voordelen

ULYSSIS

- Lineaire geschiedenis
- Geen extra *merge commit*

Rebasing: nadelen

ULYSSIS

- Herschrijft geschiedenis van een branch
- Meer moeite dan mergen :)

Rebasing: andere toepassingen

ULYSSIS

- Rebasen op eigen ancestor: geschiedenis herschrijven
- `git rebase -i`: interactive
  - **p**ick: Gebruik commit as-is
  - **r**eword: Gebruik commit, maar edit commit message
  - **e**dit: Gebruik commit, maar stop hier om amend mogelijk te maken
  - **s**quash: Gebruik commit, maar smelt samen met de vorige
  - Regel weglaten: commit verwijderen (inclusief inhoud!)
  - Uitleg staat eronder

Filebeheer

ULYSSIS

- Gelijkaardig aan UNIX
- `git rm [-r]`: verwijder een bestand (of een map recursief)
- `git rm --cached`: bewaar het bestand lokaal
- `git mv`: verplaats een bestand

Tags

ULYSSIS

- `git tag` *naam*: geef een naam aan een commit (bv. versies)
- `git tag -d` *naam*: verwijder een tag

Blaming your colleagues :)

ULYSSIS

- `git blame`: annotate een bestand met de auteur

Veranderingen ophalen van origin

ULYSSIS

- `git fetch`: haal veranderingen op, maar verplaats branch pointers **niet**
- Default: "interessante" tags. Overschrijf met `-n` (`--no-tags`) of `-t` (`--tags`)

Arbitraire commits gebruiken

ULYSSIS

- `git cherry-pick`: apply een enkele commit
- `git cherry-pick -e`: laat toe eerst de commit message aan te passen
- Initieert mogelijk een merge state

Git stash

ULYSSIS

- Veranderingen in je working directory tijdelijk opslaan
- `git stash`: sla uncommitted veranderingen tijdelijk op en reset je working directory
- `git stash list`: bekijk alle stashes
- `git stash apply [stash]`: pas de opgeslagen veranderingen weer toe op de huidige working directory
- `git stash drop` *`stash`*: verwijder een stash
- `git stash pop [stash]`: apply en drop een stash
- `git stash branch` *`branch`* `[stash]`: pas de opgeslagen veranderingen weer toe in een eigen branch

Submodules

ULYSSIS

- `git submodule add` *`url locatie`*:
- `git submodule update [--init] [--recursive]`: fetch alle submodules en checkout naar juiste commit
- `git submodule foreach [git pull origin master]`: voer command uit in elke submodule zijn directory, moet geen git commando zijn