Workshop Git

multiplayer notepad

Anthony Clays

21 november 2016

ULYSSIS

Inleiding

De basis

Eerste stappen

Branches en samenwerken

Geavanceerde features

Inleiding

Wat is git?

ULYSSIS

- Auteur: Linus Torvalds
- Beter versiecontrolesysteem:
  - Snel
  - Gedistribueerd
  - Data-integriteit ("You know you can trust the data")

Installatie

ULYSSIS

Windows `https://git-scm.com/`.
Na installatie: gebruik “Git Bash”

macOS Standaard geïnstalleerd?
Anders: Homebrew of `https://git-scm.com/`

Linux Installeer via package manager
(`sudo apt-get install git`)

- `git config --global user.email "donald@trump.org"`
- `git config --global user.name "Donald Trump"`
- Windows: `git config --global core.editor notepad`

De basis

Naamgeving

ULYSSIS

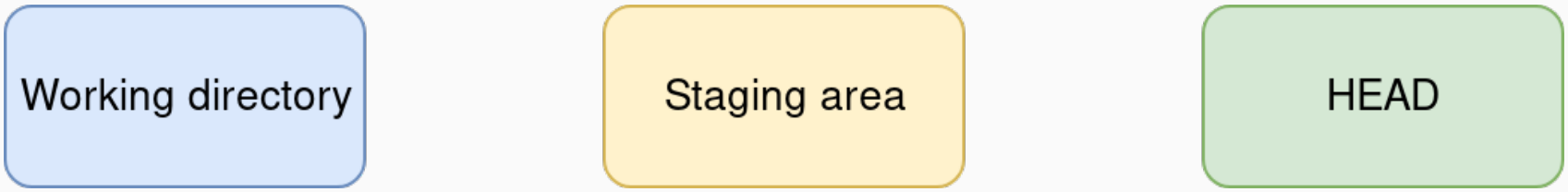
Naamgeving (2)

Staat van je repository: drie verschillende "stadia"

Working Tree De bestanden zoals ze op je harde schijf staan

HEAD De commit waarop je aan het verderwerken bent

Staging area De veranderingen die je in de *volgende* commit wilt opslaan



Eerste stappen

Een repository aanmaken

ULYSSIS

- Voor we beginnen: `git help`
- `git init`: creëer een nieuwe git repository in de huidige map
- Volledig lokaal
- Demo: volg mee!

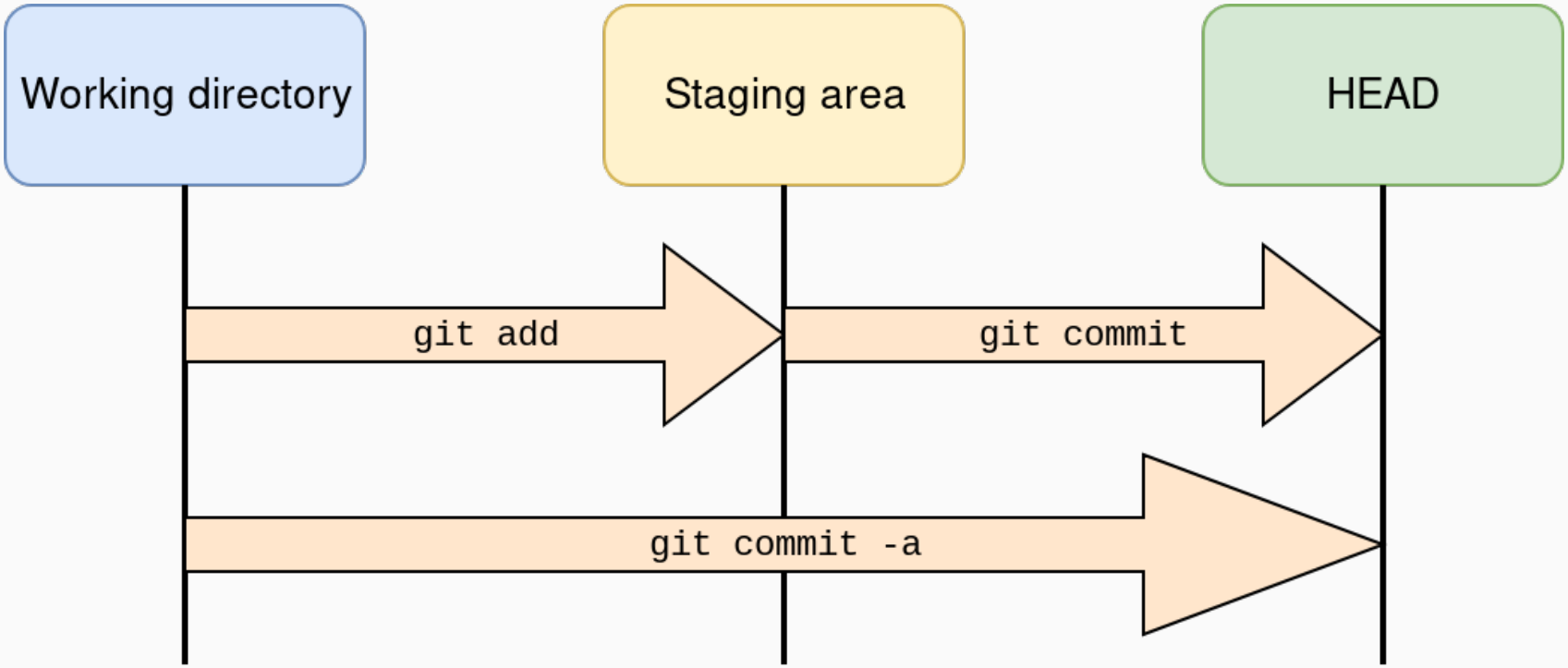
Een commit maken

ULYSSIS

- `git add`: toevoegen aan *staging area*
- `git commit`: nieuwe commit maken verderbouwend op HEAD
- Demo

Working tree / Staging area / HEAD

ULYSSIS



Geschiedenis

ULYSSIS

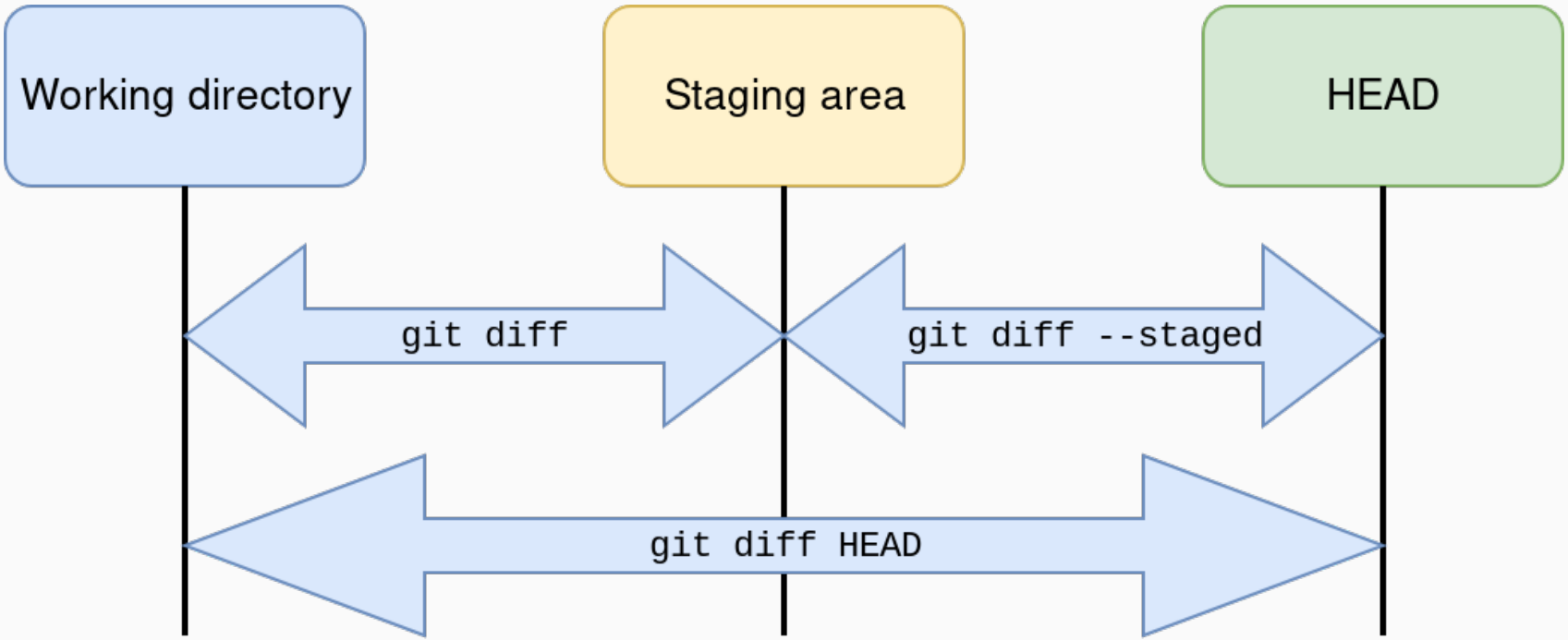
Git can remember it for you wholesale

- `git log` — Simpele oplijsting
- `git log --oneline` — zet commit hash + message op één regel
- `git log --decorate` — toont labels voor de verschillende branches
- `git log` *`identifier`* — tot *identifier* ipv HEAD

`git diff`

ULISSIS

- `git diff` Staging area → Working directory
- `git diff --staged` HEAD → Staging area
- `git diff` *id* Commit *id* → Working directory
- `git diff` *id1* *id2* Commit *id1* → Commit *id2*

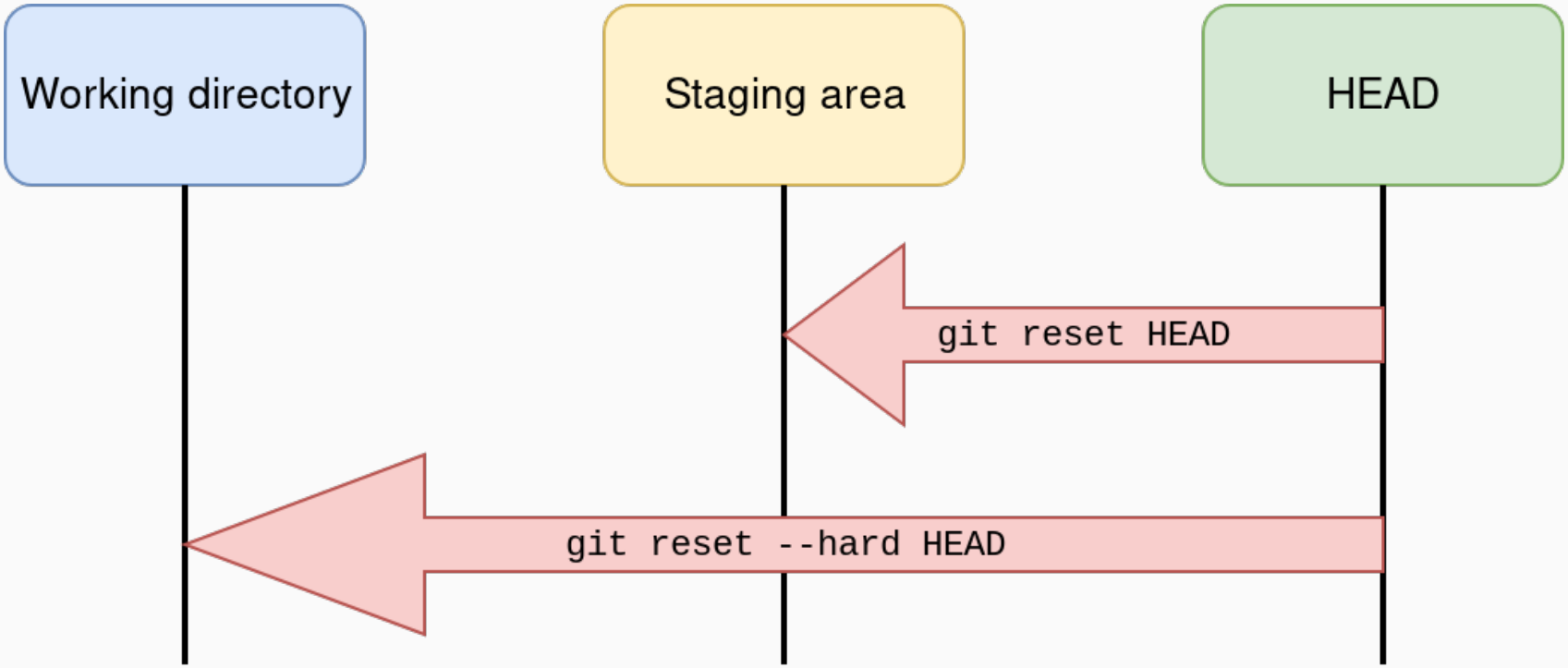


`git reset`

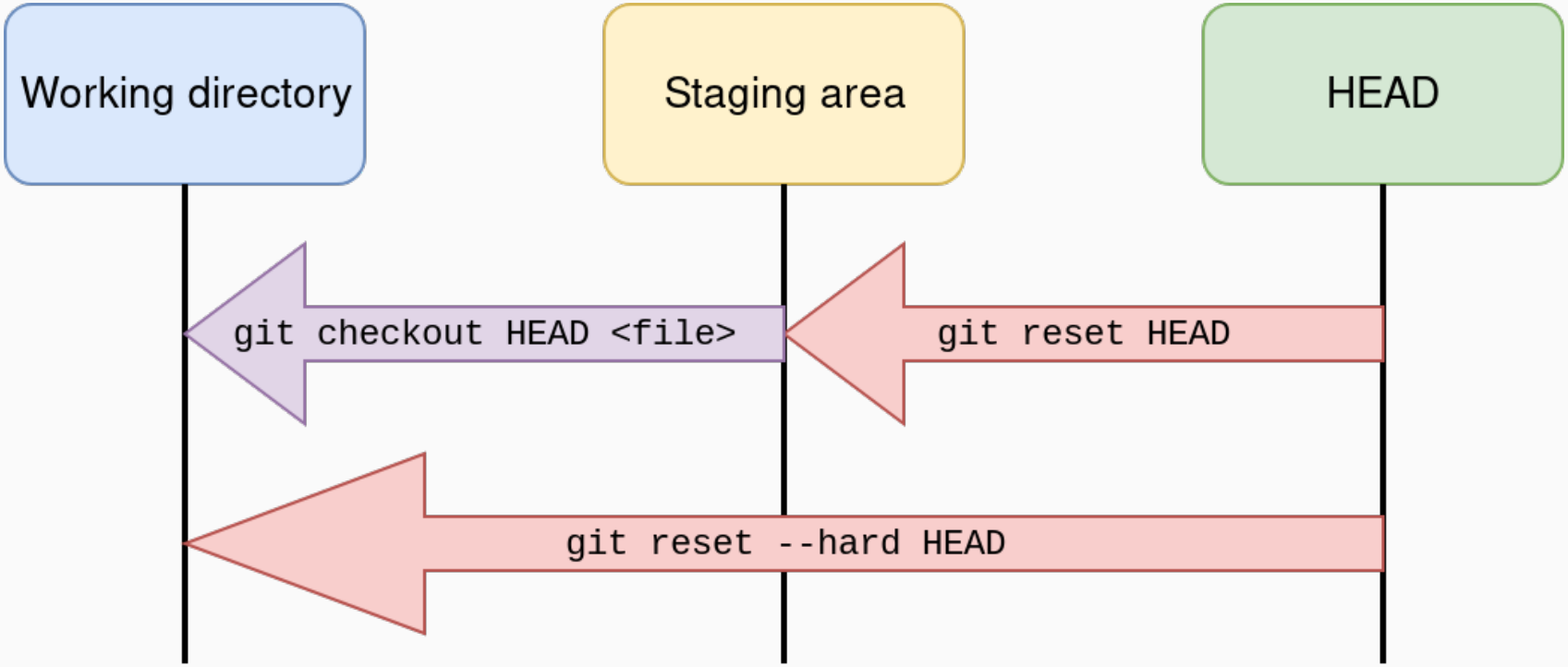
ULYSSIS

- `git reset --soft` *`id`*
  Update HEAD naar een bepaalde commit en stop.
- `git reset` *`id`*
  Hetzelfde, maar reset ook de staging area.
- `git reset --hard` *`id`*
  Hetzelfde, maar reset zowel de staging area als de working directory.

`git reset` (2)


Working directory
Staging area
HEAD
git reset HEAD
git reset --hard HEAD

`git checkout <file>`: zet het bestand (in de working directory) terug naar de versie in de staging area



`.gitignore`

ULYSSIS

- `.gitignore` vertelt git welke bestanden te negeren
- Meestal mee in repository
- Generatie: `https://gitignore.io`

Branches en samenwerken

Branches

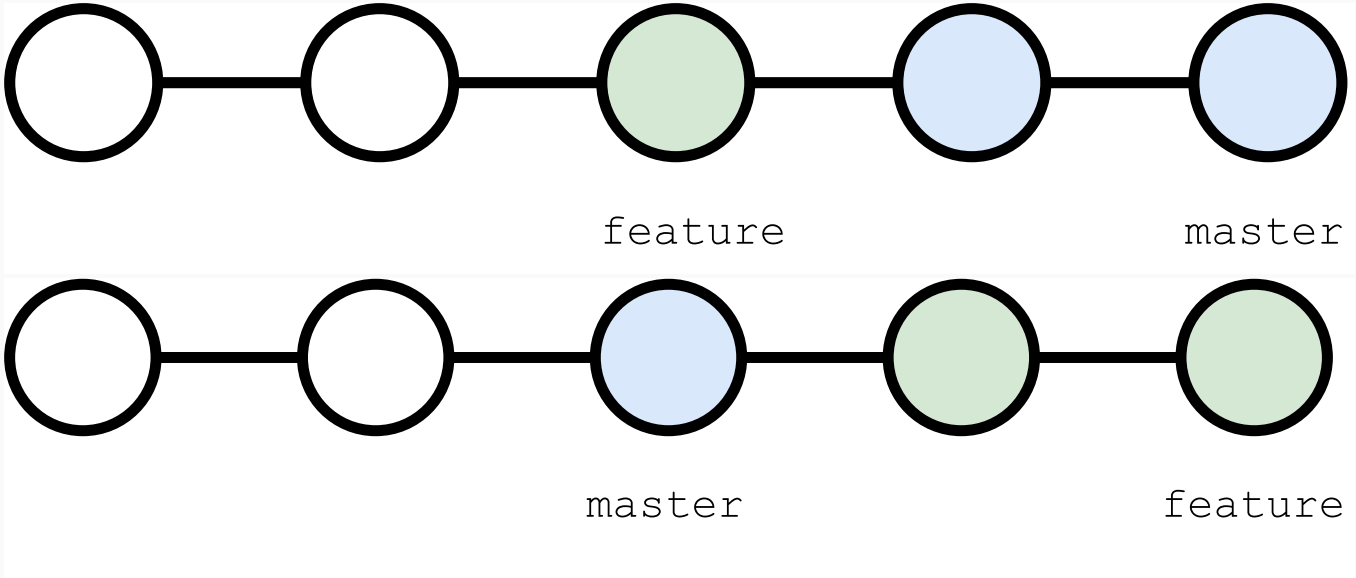
ULYSSIS

- Branches zijn verwijzingen naar een commit
- `git branch`: branches beheren
- `git checkout` *`branch`*: switch naar branch *branch*
- `git checkout -b` *`branch`*: `git branch` *`branch`* `&&` `git checkout` *`branch`*

Merging

ULYSSIS

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



Merge conflicts

ULYSSIS

- Veranderingen in hetzelfde bestand worden slim samengevoegd
- Veranderingen op dezelfde regel $\Rightarrow$ conflict!
- Voordat de merge commit gefinaliseerd wordt: los het conflict op.
- `git merge --abort`

Een repository clonen

ULYSSIS

- Tot nu toe: alles lokaal
- Waar is remote?
  - `git clone` *`url`*: download een remote repository
  - `git remote add origin` *`url`*: handmatig URL toevoegen
  - `git remote set-url origin` *`url`*: URL van remote aanpassen
- Veranderingen ophalen die in de remote zijn doorgevoerd: `git fetch`
- Deze veranderingen ook direct lokaal doorvoeren: `git pull`
- Pushen naar remote: `git push`

Een repository clonen (2)

ULYSSIS

```
git clone
https://gitlab.ulyssis.org/anthony/git-demo demo2
```

`git checkout`, **revisited**

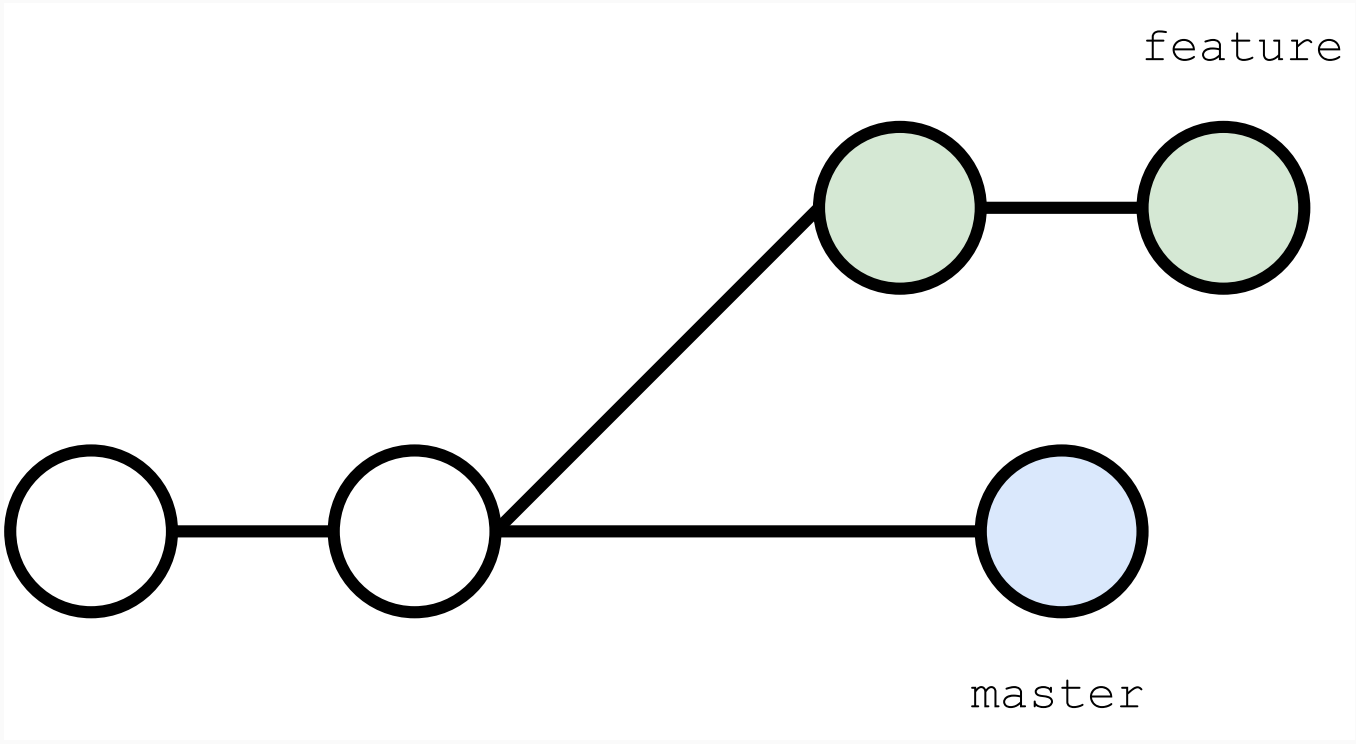
`git checkout` *`identifier path`*: update files in `path` naar de versie in `identifier`

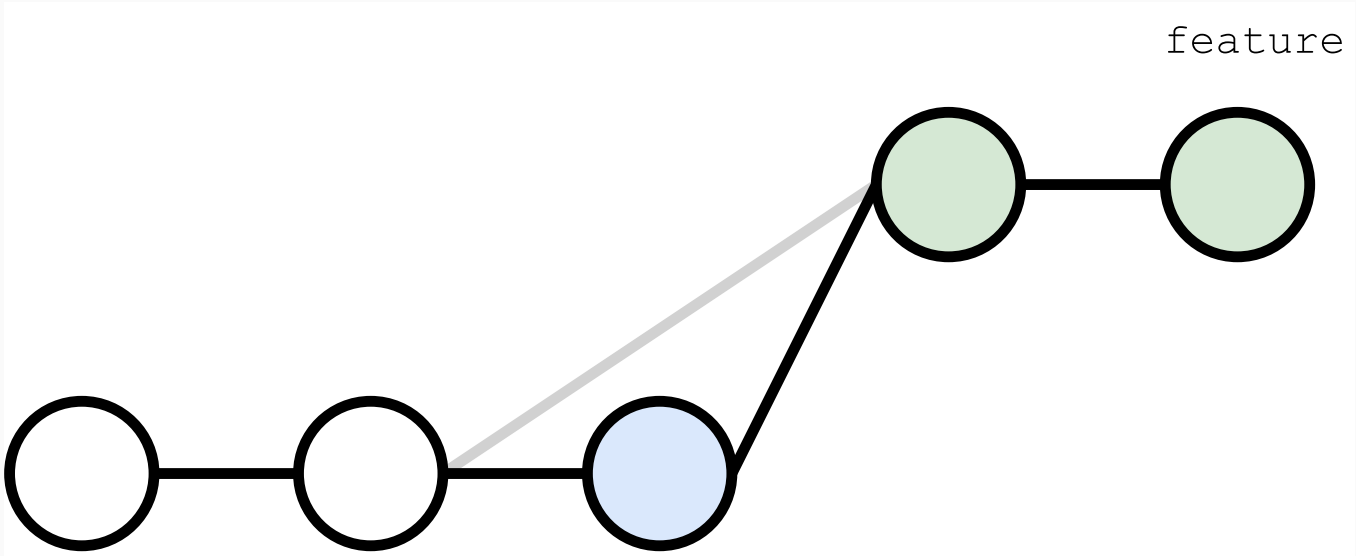
- Indien geen identifier: HEAD
- Indien geen path: working tree

Geavanceerde features

- rebase = tak verplaatsen naar een nieuwe base commit
- Als alternatief voor merge: branch "vooruitbewegen" op base branch
- Voordelen
  - Lineaire geschiedenis
  - Geen extra *merge commit*
- Nadelen
  - Herschrijft geschiedenis (gevaarlijk)
  - Moeilijker
- `rebase --interactive`

git rebase (2)





`git commit --amend`

ULYSSIS

`git commit --amend`: voeg veranderingen toe aan vorige commit

`git stash`

ULYSSIS

the sock drawer of version control

- `git stash [save]` — sla huidige veranderingen op
- `git stash (apply|pop) [stash]` — pas de laatste stash toe
- `git stash drop [stash]` — verwijder deze stash
- `git stash list` — toon alle stashes
- `git stash branch` *`branch`* `[stash]` — pas toe in een nieuwe branch

`git tag`

ULYSSIS

- `git tag` *`tag`* — tag maken op huidige commit
- `git tag -d` *`tag`* — tag verwijderen
- `git push --tags` — lokale tags naar remote kopiëren

git blame

Destroying friendships since 2006

Toon de auteur(s) van een bestand

Lijn per lijn

`git reflog`

ULYSSIS

Toont een lijst van alle staten waarin de repository zich heeft bevonden

Vind verloren commits terug

Conclusies

ULYSSIS

- Als je iets niet snapt: lees de manual, zoek op op google
- Doe nooit een force push naar master git commit crime
- Schrijf goede, beschrijvende commit messages
- ULYSSIS biedt ook GitLab aan (private repositories)