Version Control met GIT

plus: Advies over gebruik met SCRUM

Agenda

- Introductie
- Resources
- Model
- Basics
- Tags
- Branching
- Merge
- Rebase
- Remote repos
- GIT en SCRUM

RESOURCES

GIT Resources

- Gebruik de command line!!
  - Vooral aan het begin
  - Later misschien plugins
    - Meer controle

- Gebruik zeker niet de tool van GITHUB!

GIT Resources

- At the command line: (where verb = config, add, commit, etc.)

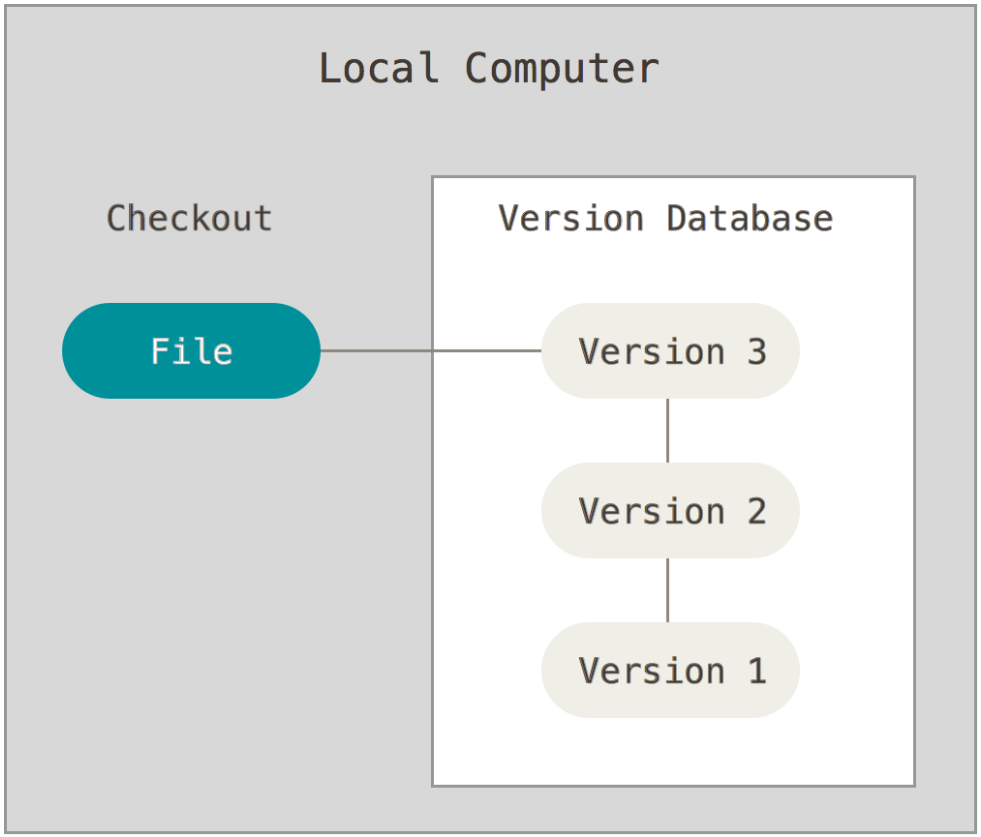
```
$ git help <verb>
$ git <verb> --help
$ man git-<verb>
```

- Free on-line book: http://git-scm.com/book
- Git tutorial: http://schacon.github.com/git/gittutorial.html
- Reference page for Git: http://gitref.org/index.html
- Git website: http://git-scm.com/

- Git for Computer Scientists (http://eagain.net/articles/git-for-computer-scientists/)

MODEL

Version Control (Lokaal)

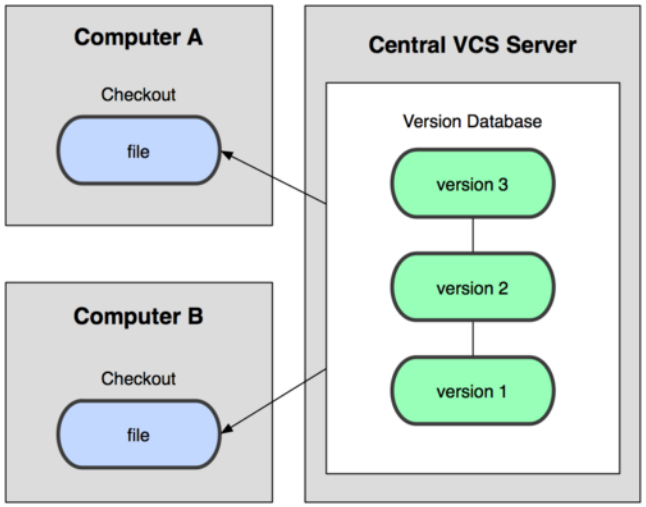


git

HOGESCHOOL UTRECHT

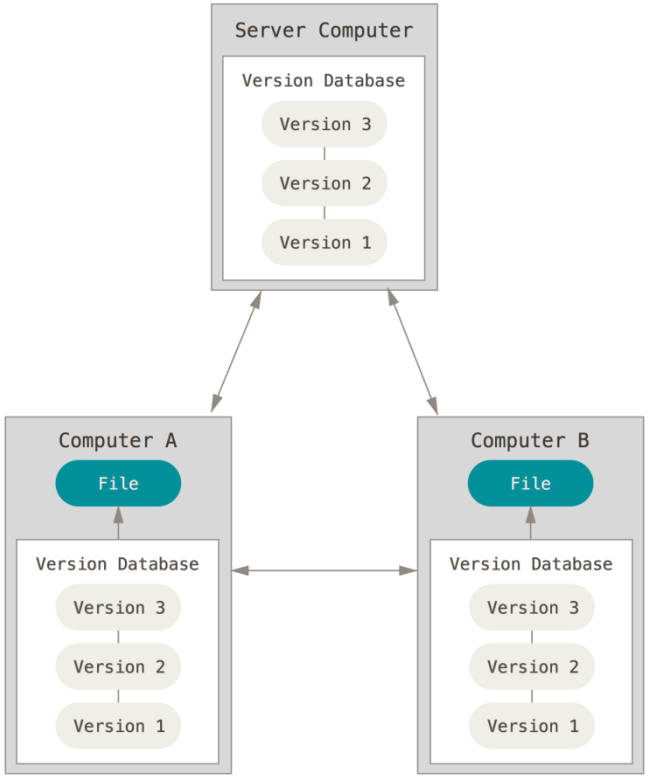
Version Control (Distributed)

Centralized Model



(CVS, Subversion, Perforce)

Distributed Model



(Git, Mercurial)

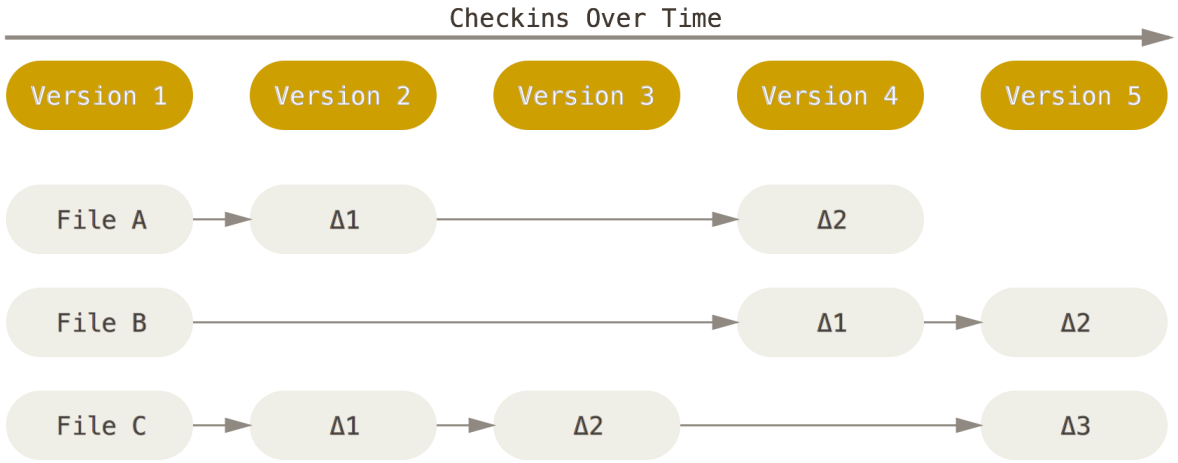
Result: Many operations are local

git

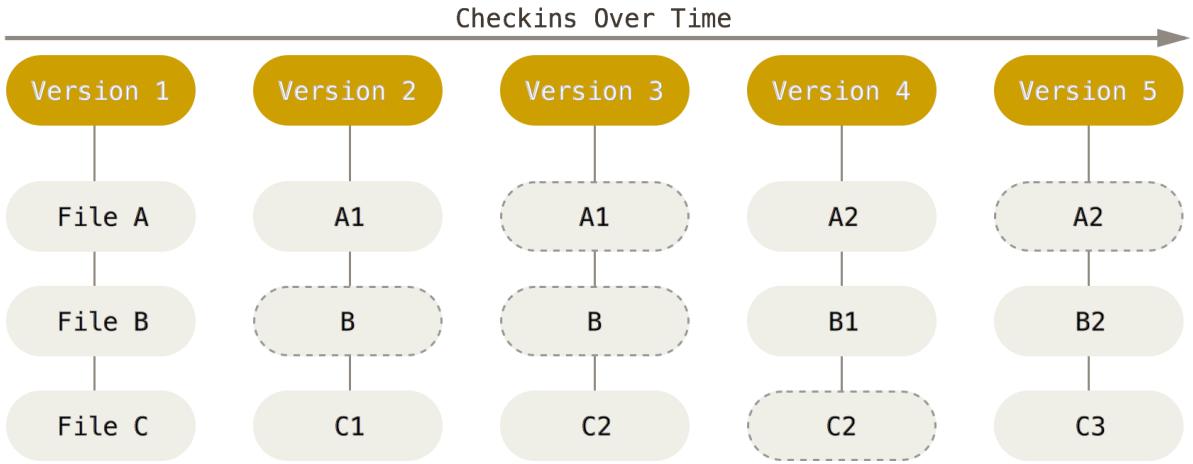
HOGESCHOOL UTRECHT

Snapshots, geen verschillen

CVS
Subversion



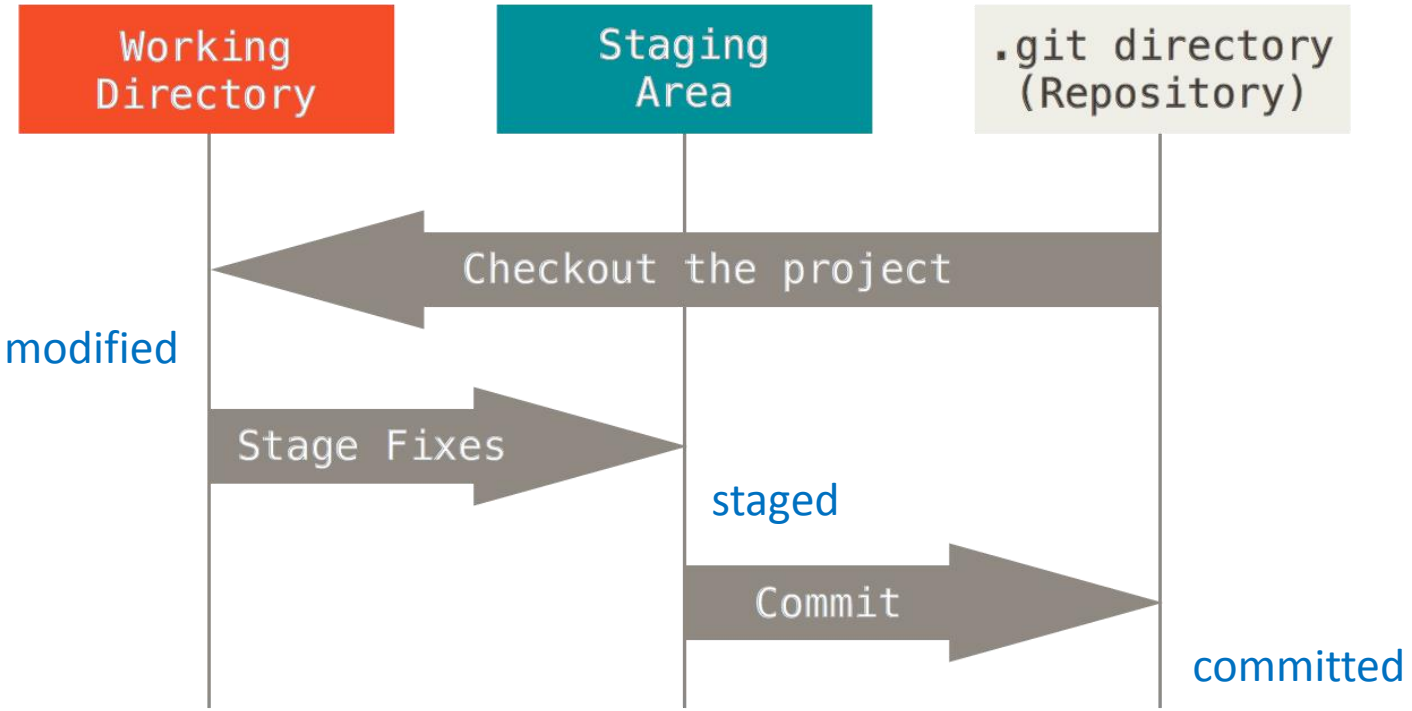
GIT



git

HOGESCHOOL UTRECHT

Drie statussen (Local)



BASICS

Creeër een repo

- Twee scenarios:
  - Clonen van een bestaande **repo** in de huidige directory:

    ```
    $ git clone <url> [local dir name]
    ```

    Dit zorgt voor een **directory** met de naam `local dir name`, met een **working copy** van de files in de **repo** en een **.git** directory
  - Volledig nieuwe repo in je huidige directory:

    ```
    $ git init
    ```

    Dit maakt een **.git** directory in je huidige directory.

Commando's

command	description
`git clone` ***`url [dir]`***	copy a git repository so you can add to it
`git add` ***`files`***	**adds** file contents to the **staging area**
`git commit`	**records** a snapshot **of the staging area**
`git status`	view the **status** of your files in the **working** directory and **staging area**
`git diff`	shows diff of what is **staged** and what is **modified but unstaged**

Al deze commando's werken op jouw **local versie van repo**.

Voorbeeld

- Zie command line en whiteboard…

Commits

- Een commit maakt een object en verplaatst de labels
  - HEAD
  - Actieve branch (op dit moment master)

Commit Objecten

- ID
  - Content
  - Author
  - Date
  - Log
  - Previous commit ID
- Checksum

ELKE ID IS UNIEK

ELKE COMMIT IS UNIEK

COMMITS VERANDEREN NOOIT

Staging area en opbouw commit

- Staging area
  - Alle files die je hebt toegevoegd met `git add`
- Vormeren van verschillende commits
- Bijv:
  - 6 files zijn verandert
  - Eerste drie files horen bij één ding

```
$ git add file1 file2 file3; git commit
```

  - Andere drie files horen bij ander ding

```
$ git add file4 file5 file6; git commit
```

TAGS

Tags zijn gewoon ook weer labels

- Toevoegen van een tag:

```
$ git tag v.10
$ git tag v.20 b3a5aff8
```

- Tags zijn lokaal, wil je deze delen moet je ze pushen

```
$ git push origin --tags
```

BRANCHING

Omgaan met branches

- Om een **branch** te maken met de naam **experimental**:

```
$ git branch experimental
```

- Om alle **branches** te zien:

```
$ git branch
```

- Om te wisselen naar een **branch** met de naam **release**

```
$ git checkout release
```

Voorbeeld

- Zie command line en whiteboard…

Omgaan met branches

- Eigenlijk zijn branches alleen maar labels
  - Commits verplaatsen labels en maken objects
- Niets bijzonders dus
- Actieve branch zie je altijd in de git-shell
  - Zie ook gitk

Algemene werkwijze

branch feature

checkout feature

edit

test

add

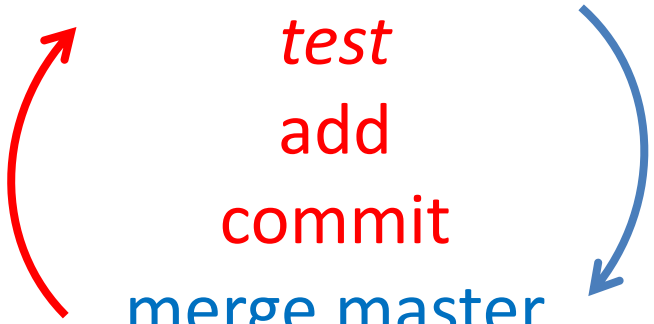
commit

merge master

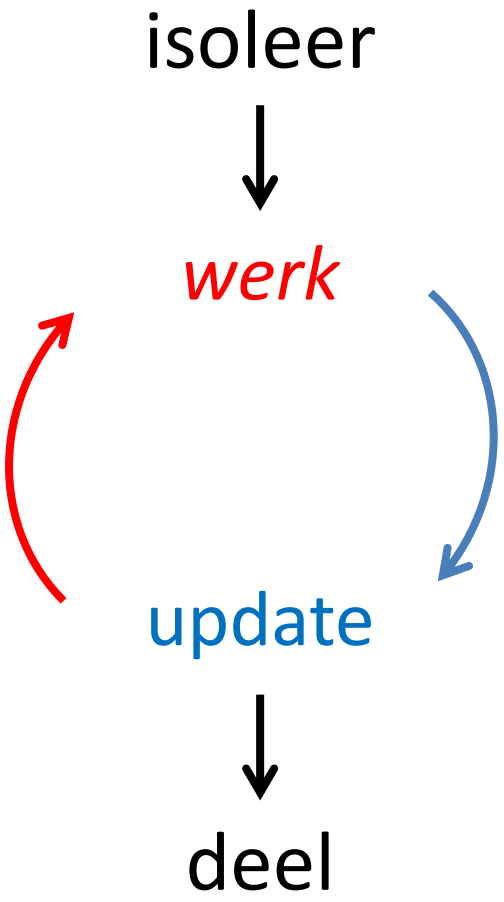
test

checkout master

merge feature



Algemene werkwijze



MERGE

Omgaan met merges

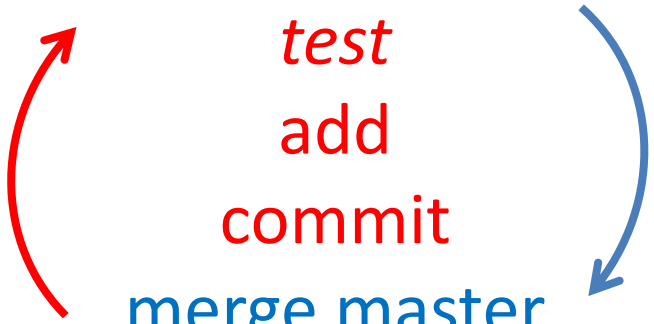
- Om een **merge** te doen met de naam **experimental** (doe je in de branch waar je naartoe wil mergen):

```
$ git merge experimental
```

- Er zijn verschillende soorten merges
  - Fast-forward strategy
  - Recursive strategy
  - …
- Git probeert conflicten zoveel mogelijk te voorkomen

Denk daarbij aan:

branch feature
checkout feature
edit
test
add
commit
merge master
test
checkout master
merge feature



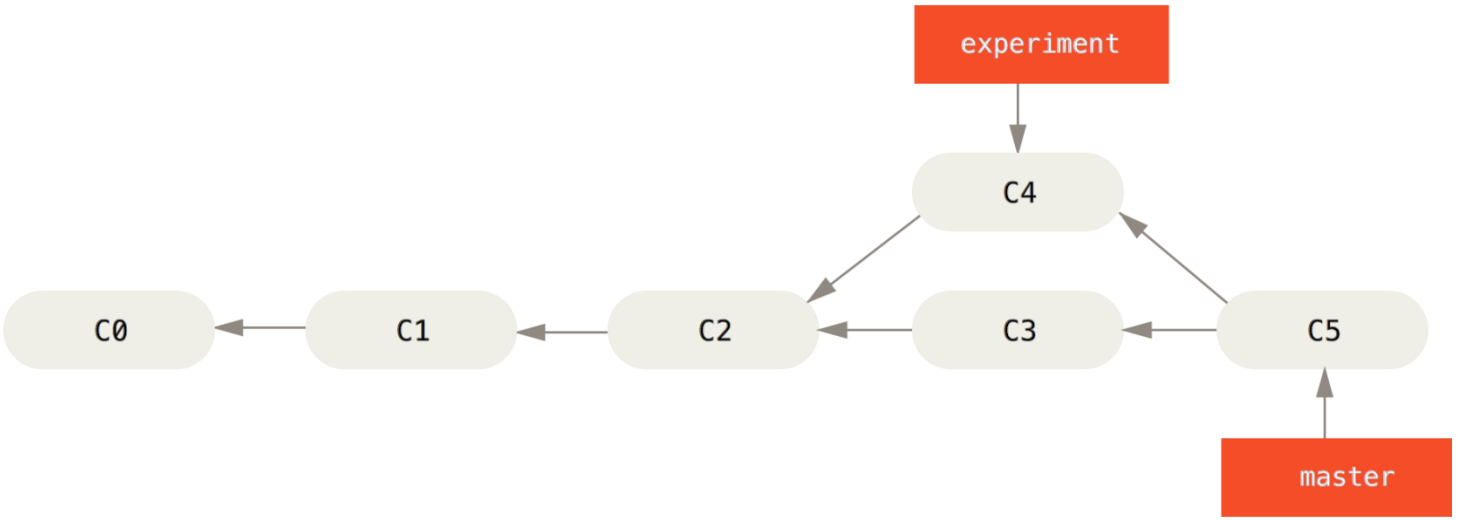
Voorbeeld

- Zie command line en whiteboard…

Merge of Rebase

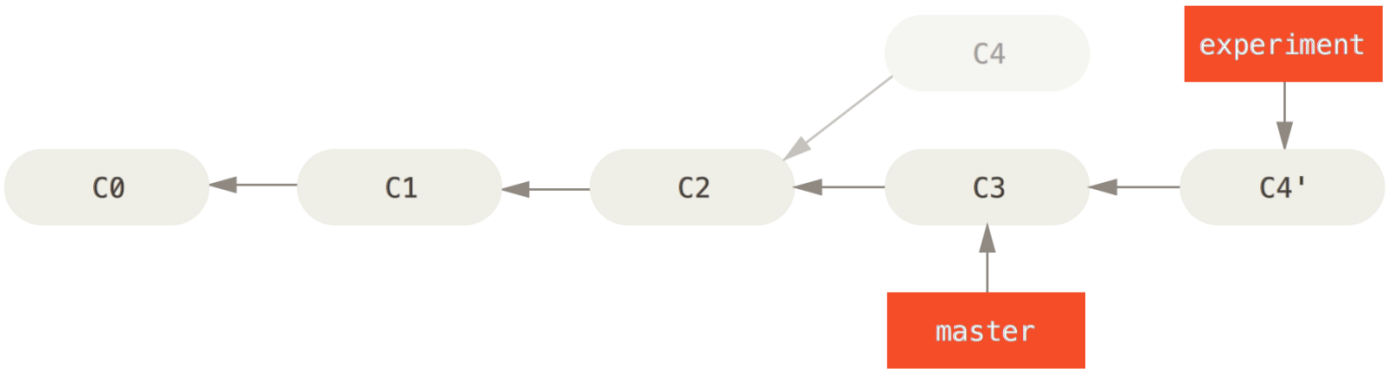
- Om te voorkomen dat je veel verschillende commit-objects krijgt en een verwarrende history
  - Rebase kan gebruikt worden om een merge te doen
  - Doe alleen een Rebase op je lokale branches, dus geen public branches rebasen!!

Merge



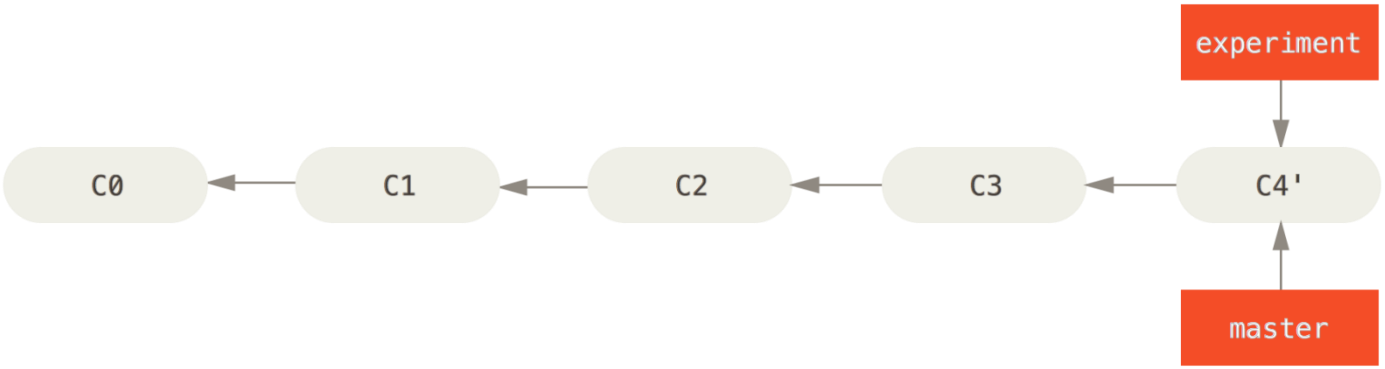

```
$ git checkout master
$ git merge experiment
```

Rebase (1)




```
$ git checkout experiment
$ git rebase master
```

Rebase (2)




```
$ git checkout experiment
$ git rebase master
$ git checkout master
$ git merge experiment
```

HOGESCHOOL UTRECHT

Waarvoor/Wanneer?

- Zeer geschikt om up-to-date te blijven (zie later)
- Kleinere history

- Nadeel:
  - Je verliest wat informatie (geen gegevens, alleen overzicht)

REMOTE REPOS

Intermezzo om een remote te maken

- … domdiedom…

Werken met remote repos

command	description
`git push`	push your new branches and data to a remote repository
`git pull`	fetch from a remote repo and try to merge into the current branch
`git fetch`	fetch from a remote repo

Voorbeeld

- Zie command line en whiteboard…

GIT EN SCRUM

Branch structuur

- Er zijn vele manieren om te werken met branches

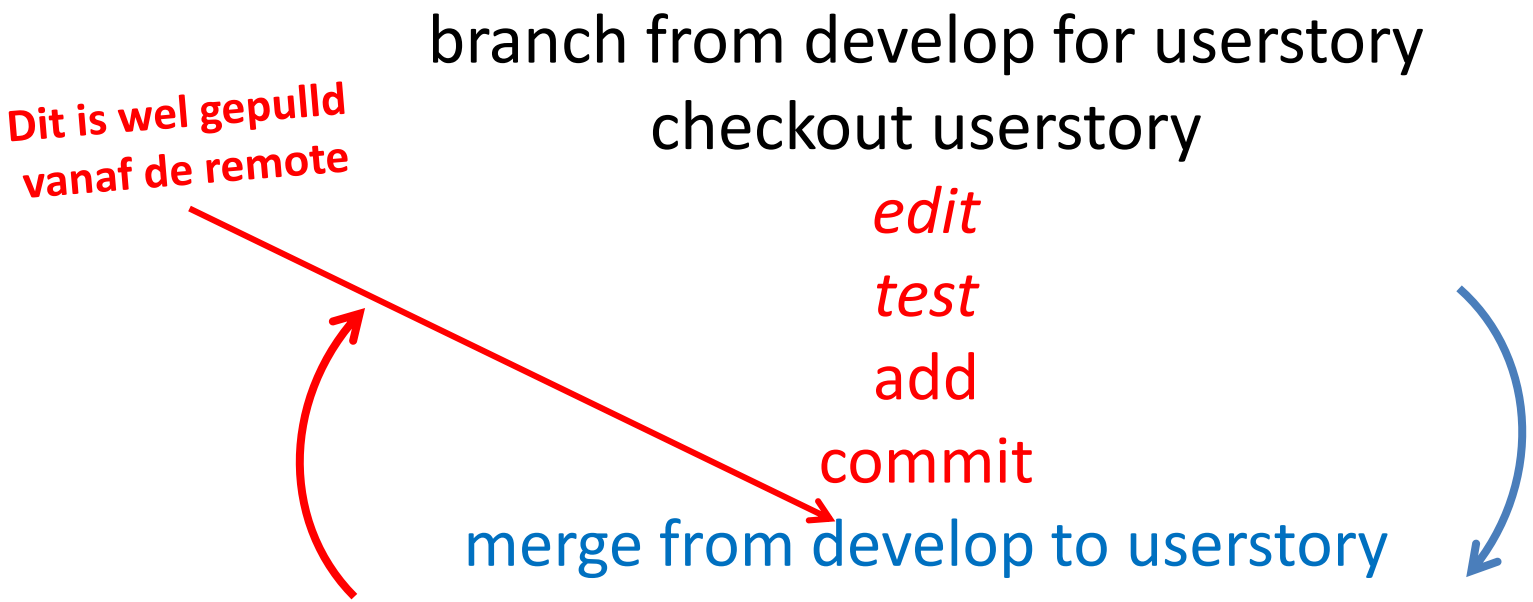
- Dit is slechts één manier
  - Wel een vaak gebruikte manier
  - Vooral voor situaties dat je voor het eerst begint met Git

Branch structuur

- Twee standaard branches:
  - **Master (for releases)**
  - **Develop**

- Daarnaast voor elke **user story** een **branch**

Algemene werkwijze

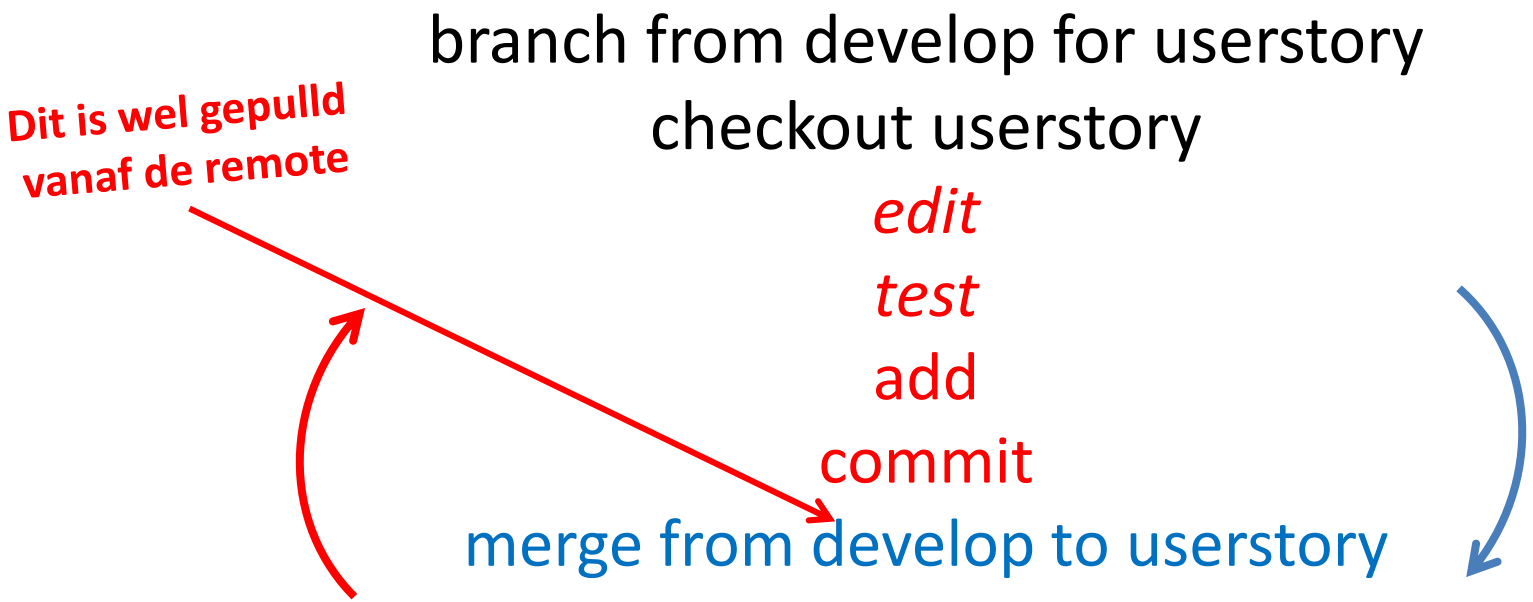


Hoe doe je het bijblijven?

- De standaard wijze dat je heel veel doet:

```
$ git checkout develop
$ git pull
$ git checkout -
$ git merge develop –no-ff
        (git rebase develop)
```

Algemene werkwijze

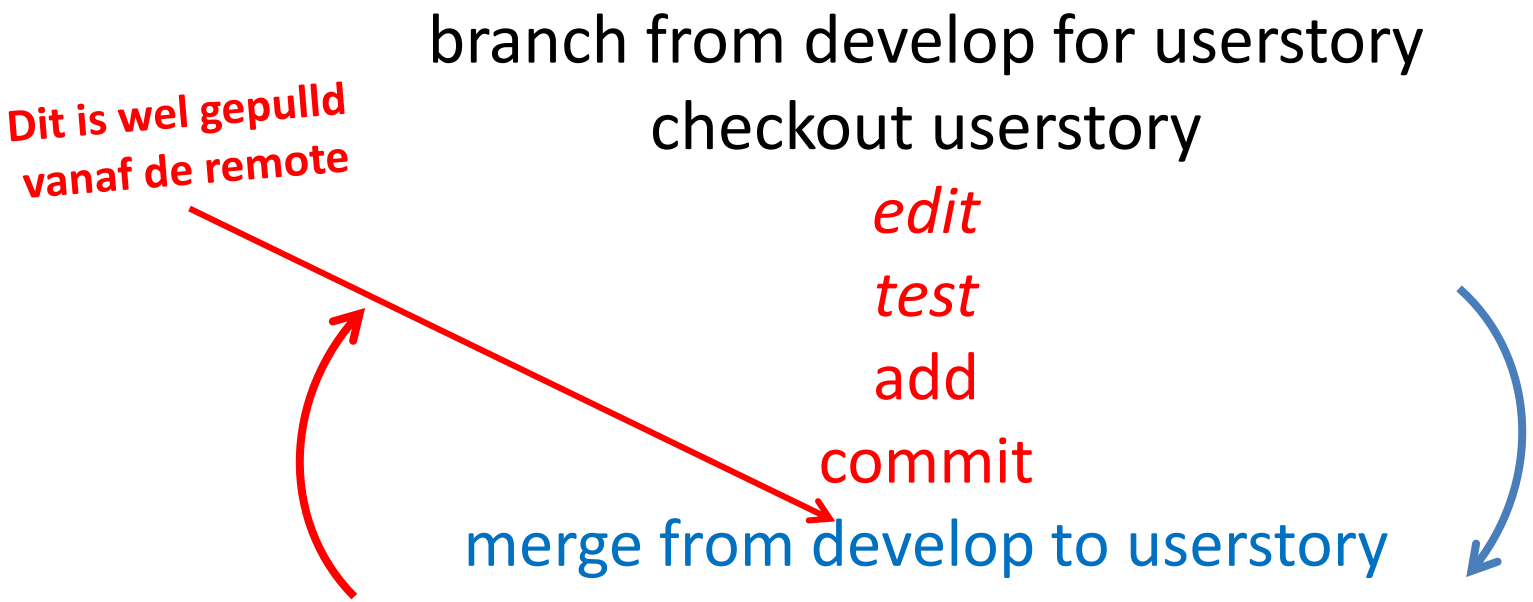


Merging a userstory to develop

- Om de hele history te blijven zien is het verstandig om geen Fast-Forward merge te doen
- Dus:

```
$ git checkout develop
$ git merge --no-ff mystory
```

Algemene werkwijze



Na een sprint

- Merge de develop naar de master branch
- Dus:

```
$ git checkout master
$ git merge --no-ff develop
$ git tag <version>
```

VRAGEN?