ZooRoyal IT

Git

Sebastian Knott

10. April 2018

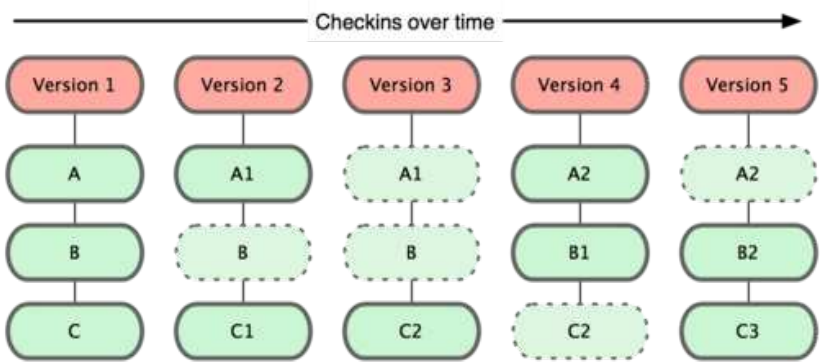
ZooRoyal IT
Git
Sebastian Knott
10. April 2018

Überblick

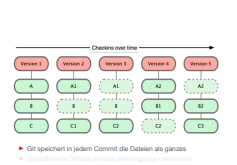
git
Git unter der Haube
Datenhaltung in Git
History
Branch
Merge
Rebase
Arbeiten mit Git
Repositories organisieren
Git Cli

Überblick

Datenhaltung in Git
Versionen und Dateien

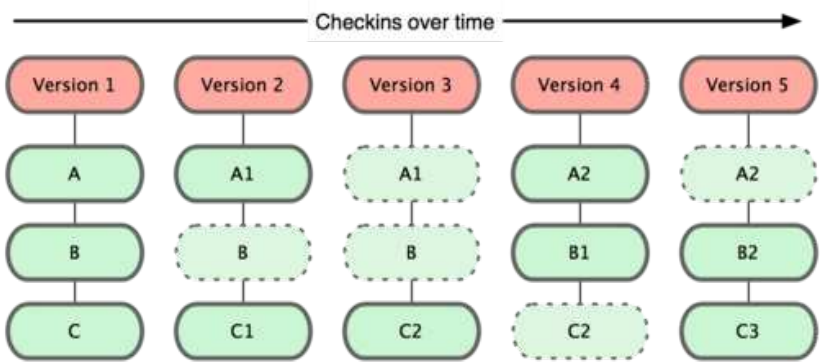


- Git speichert in jedem Commit die Dateien als ganzes
- Unveränderte Dateien werden allerdings nur referenziert

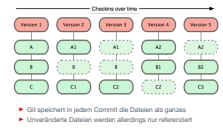


1. Entspricht cp -r quelle ziel

Datenhaltung in Git
Versionen und Dateien



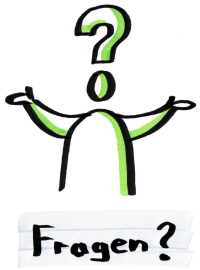
- Git speichert in jedem Commit die Dateien als ganzes
- Unveränderte Dateien werden allerdings nur referenziert



1. Entspricht cp -r quelle ziel

Fragen

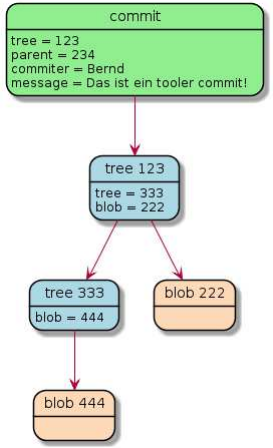
Fragen?

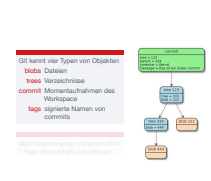


Git kennt vier Typen von Objekten

- **blobs** Dateien
- **trees** Verzeichnisse
- **commit** Momentaufnahmen des Workspace
- **tags** signierte Namen von commits

Alle Objekte werden mit einem SHA-1 Hash ihres Inhalts identifiziert

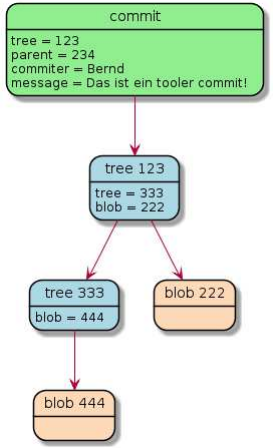


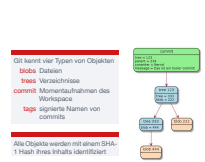


Git kennt vier Typen von Objekten

- **blobs** Dateien
- **trees** Verzeichnisse
- **commit** Momentaufnahmen des Workspace
- **tags** signierte Namen von commits

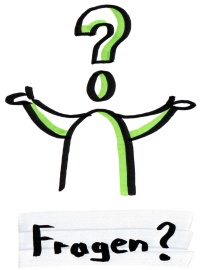
Alle Objekte werden mit einem SHA-1 Hash ihres Inhalts identifiziert





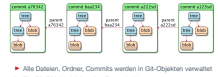
Fragen

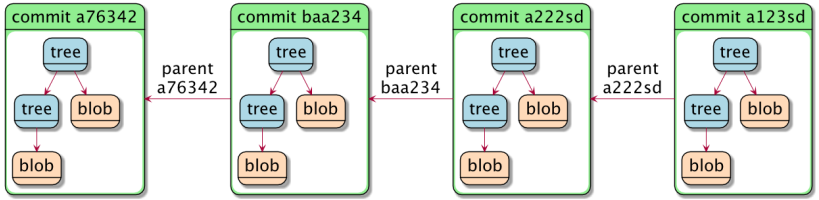
Fragen?



Aufgabe: Verzeichnisbaum anzeichnen und neuen Git Commit mit Verzeichnissen und Datien konstruieren lassen.

Überblick

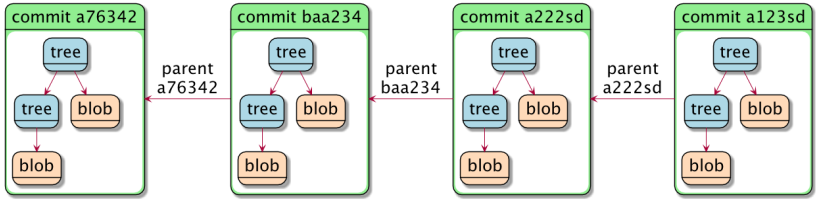




- Alle Dateien, Ordner, Commits werden in Git-Objekten verwaltet
- Die Git-Objekte bilden eine Baumstruktur

1. Aufgabe: Commit anzeichnen. Eine Datei verändert. Neuen Commit anzeichnen lassen.

History
Verkettung von commits

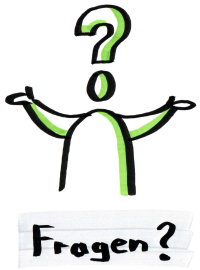


- Alle Dateien, Ordner, Commits werden in Git-Objekten verwaltet
- Die Git-Objekte bilden eine Baumstruktur

1. Aufgabe: Commit anzeichnen. Eine Datei verändert. Neuen Commit anzeichnen lassen.

Fragen

Fragen?



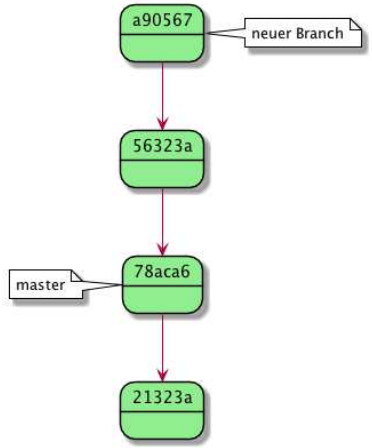
Überblick

Branches
Organisation von Commits

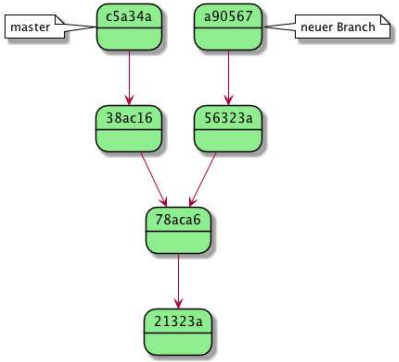
- Branches sind Pointer auf commits
- Verschiedene Commit-ishs für gleichen Commit

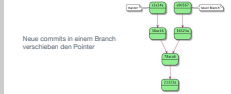
```
git checkout 78aca6
git checkout master
```

(Vorsicht! Detached Head!)



Neue commits in einem Branch verschieben den Pointer

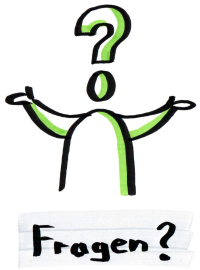




- Aufgabe1: Git Commit anzeichnen. Weitere Branches erzeugen lassen. Weitere Commits erzeugen lassen. Einen Branch resetten.
- Hashes verändern sich nicht!

Fragen

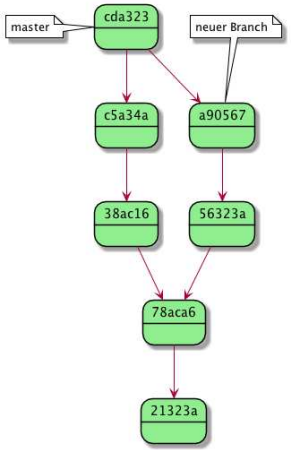
Fragen?



Überblick

Merge
Zusammenführen von zwei Branches

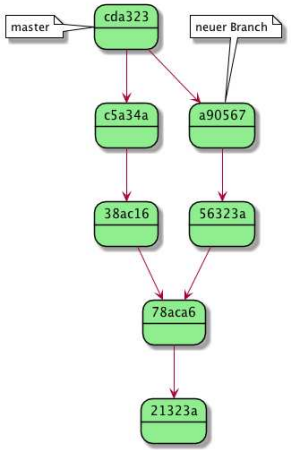
- Neuer Branch → master
- Quellbranch bleibt bestehen
- Pointer im Quellbranch wird nicht bewegt
- Neuer commit (`cda323`) hat zwei Parents
- `git branch -d neuer Branch` entfernt nur den Pointer, nicht die commits

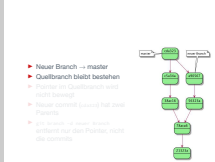


1. Mit HEAD 1 wird uneindeutig. HEAD^1 gibt Parent an

Merge
Zusammenführen von zwei Branches

- Neuer Branch → master
- Quellbranch bleibt bestehen
- Pointer im Quellbranch wird nicht bewegt
- Neuer commit (`cda323`) hat zwei Parents
- `git branch -d neuer Branch` entfernt nur den Pointer, nicht die commits

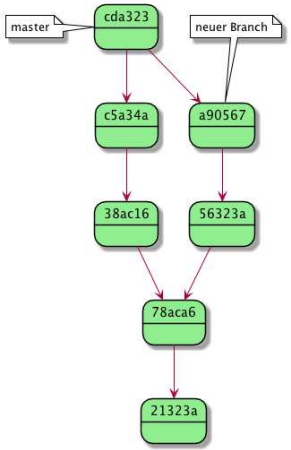


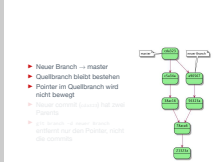


1. Mit HEAD 1 wird uneindeutig. HEAD^1 gibt Parent an

Merge
Zusammenführen von zwei Branches

- Neuer Branch → master
- Quellbranch bleibt bestehen
- Pointer im Quellbranch wird nicht bewegt
- Neuer commit (`cda323`) hat zwei Parents
- `git branch -d neuer Branch` entfernt nur den Pointer, nicht die commits

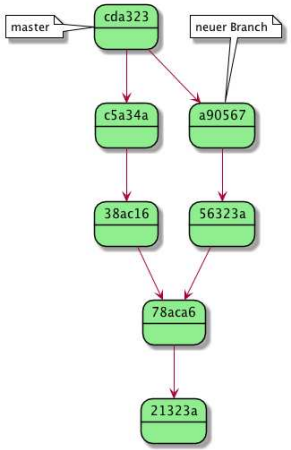


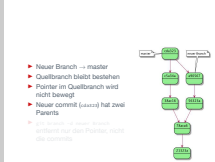


1. Mit HEAD 1 wird uneindeutig. HEAD^1 gibt Parent an

Merge
Zusammenführen von zwei Branches

- Neuer Branch → master
- Quellbranch bleibt bestehen
- Pointer im Quellbranch wird nicht bewegt
- Neuer commit (`cda323`) hat zwei Parents
- `git branch -d neuer Branch` entfernt nur den Pointer, nicht die commits

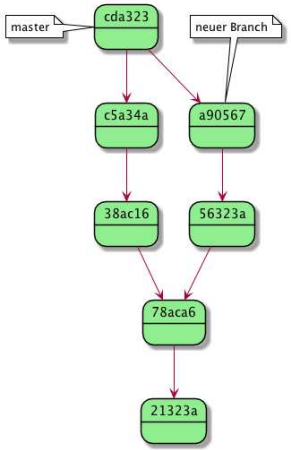


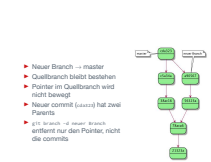


1. Mit HEAD 1 wird uneindeutig. HEAD^1 gibt Parent an

Merge
Zusammenführen von zwei Branches

- Neuer Branch → master
- Quellbranch bleibt bestehen
- Pointer im Quellbranch wird nicht bewegt
- Neuer commit (`cda323`) hat zwei Parents
- `git branch -d neuer Branch` entfernt nur den Pointer, nicht die commits

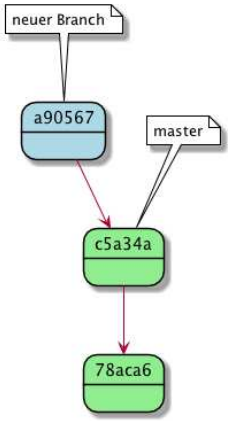




1. Mit HEAD 1 wird uneindeutig. HEAD^1 gibt Parent an

Merge
Mergen mit Fast Forward

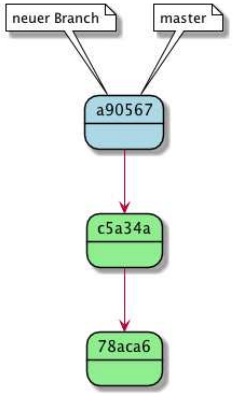
Vorher

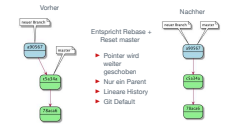


Entspricht Rebase + Reset master

- Pointer wird weiter geschoben
- Nur ein Parent
- Lineare History
- Git Default

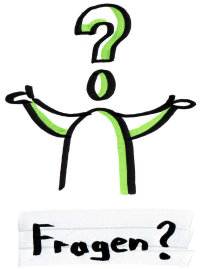
Nachher





Fragen

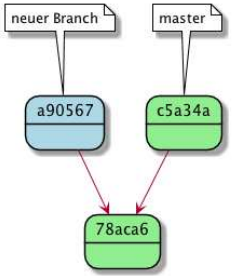
Fragen?



Überblick

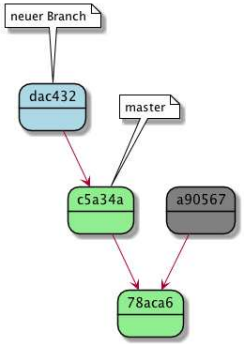
Rebase
Was passiert bei einem Rebase

Vorher



- Commit wird kopiert und Parent gesetzt
- master Pointer wird nicht verschoben
- Lineare History

Nachher



Rebase gibt einem weitere Möglichkeiten Commits umzuorganisieren.

Drop Einzelne Commits entfernen

Squash, Fixup Commits zusammenführen

Reword Einzelne Commit Messages ändern

Edit Einzelne Commits amenden

Operationen können beliebig kombiniert werden

Rebase gibt einem weitere Möglichkeiten Commits umzuorganisieren.
Drop Einzelne Commits entfernen
Squash, Fixup Commits zusammenführen
Reword Einzelne Commit Messages ändern
Edit Einzelne Commits amenden
Operationen können beliebig kombiniert werden

Rebase gibt einem weitere Möglichkeiten Commits umzuorganisieren.

Drop	Einzelne Commits entfernen
Squash, Fixup	Commits zusammenführen
Reword	Einzelne Commit Messages ändern
Edit	Einzelne Commits amenden

Operationen können beliebig kombiniert werden

Rebase gibt einem weitere Möglichkeiten Commits umzuorganisieren.
Drop Einzelne Commits entfernen
Squash, Fixup Commits zusammenführen
Reword Einzelne Commit Messages ändern
Edit Einzelne Commits amenden
Operationen können beliebig kombiniert werden

Rebase gibt einem weitere Möglichkeiten Commits umzuorganisieren.

Drop	Einzelne Commits entfernen
Squash, Fixup	Commits zusammenführen
Reword	Einzelne Commit Messages ändern
Edit	Einzelne Commits amenden

Operationen können beliebig kombiniert werden

Rebase gibt einem weitere Möglichkeiten Commits umzuorganisieren.

Drop	Einzelne Commits entfernen
Squash, Fixup	Commits zusammenführen
Reword	Einzelne Commit Messages ändern
Edit	Einzelne Commits amenden

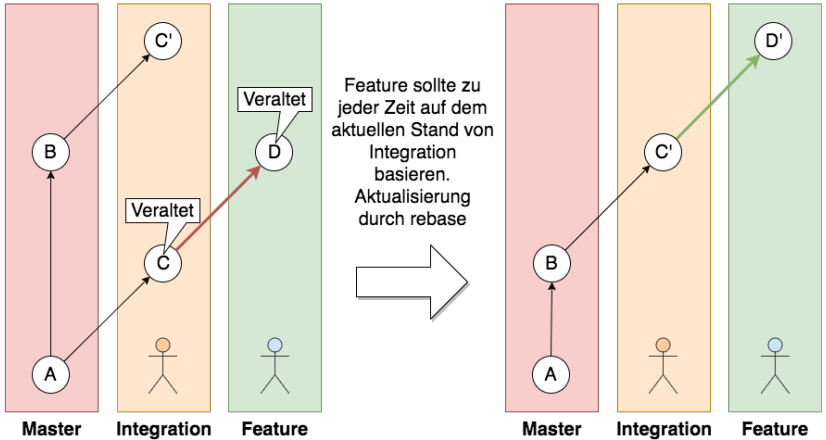
Operationen können beliebig kombiniert werden

Mit großer Macht kommt große Verwirrung

- Vorsicht vor Branches, auf denen andere Entwickler Arbeiten
- Überkomplexe Branches machen Rebases extrem schwer
- Mitdenken erforderlich

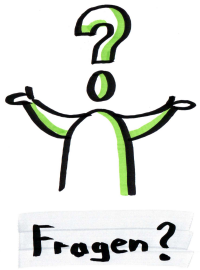
Mit großer Macht kommt große Verwirrung

- Vorsicht vor Branches, auf denen andere Entwickler Arbeiten
- Überkomplexe Branches machen Rebases extrem schwer
- Mitdenken erforderlich


C'
Veraltet
D
Veraltet
B
C
A
Master
Integration
Feature
Feature sollte zu jeder Zeit auf dem aktuellen Stand von Integration basieren. Aktualisierung durch rebase
D'
C'
B
A
Master
Integration
Feature

Fragen

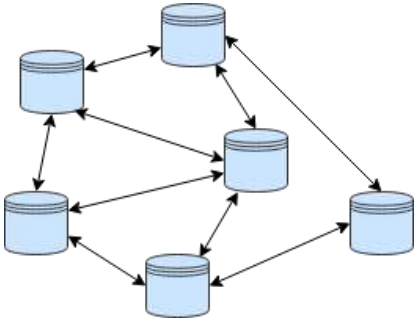
Fragen?



Überblick

Repositories organisieren

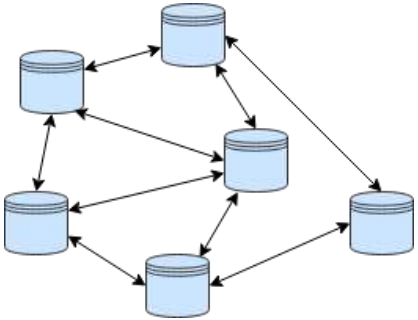
Repositories beliebig verknüpfen



- Repositories existieren prinzipell unabhängig
- Können mit Prozesseregel organisiert werden
- Daten könne mit Push, Pull oder Patch ausgetauscht werden

Repositories organisieren

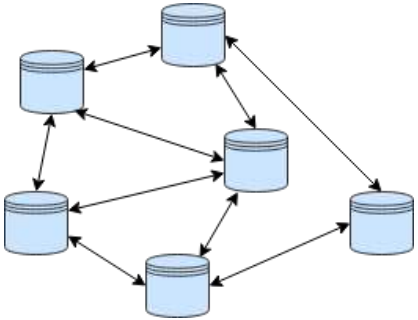
Repositories beliebig verknüpfen



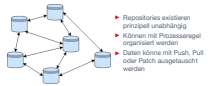
- Repositories existieren prinzipell unabhängig
- Können mit Prozesseregel organisiert werden
- Daten könne mit Push, Pull oder Patch ausgetauscht werden

Repositories organisieren

Repositories beliebig verknüpfen



- Repositories existieren prinzipell unabhängig
- Können mit Prozesseregel organisiert werden
- Daten könne mit Push, Pull oder Patch ausgetauscht werden

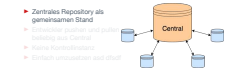


Git bietet einige Protokolle zur Datenübertragung

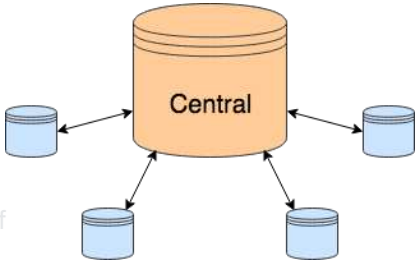
- git
- file
- ssh
- http
- mailto

Git bietet einige Protokolle zur Datenübertragung
- git
- file
- ssh
- http
- mailto

Repositories organisieren
Zentrales Rpositry



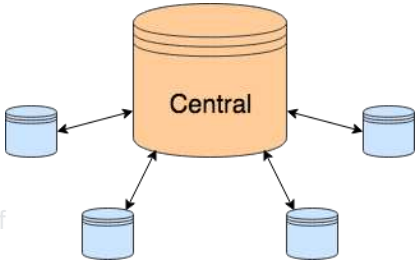
- Zentrales Repository als gemeinsamen Stand
- Entwickler pushen und pullen beliebig aus Central
- Keine Kontrollinstanz
- Einfach umzusetzen asd dfsdf



Repositories organisieren

Zentrales Rpositry

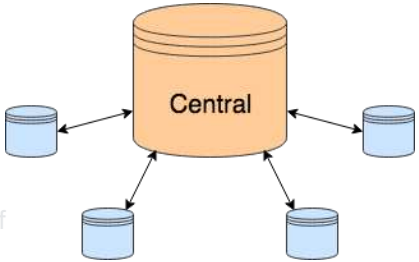
- Zentrales Repository als gemeinsamen Stand
- Entwickler pushen und pullen beliebig aus Central
- Keine Kontrollinstanz
- Einfach umzusetzen asd dfsdf



Repositories organisieren

Zentrales Rpositry

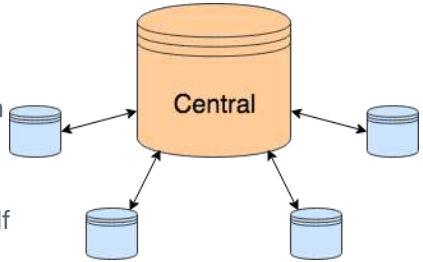
- Zentrales Repository als gemeinsamen Stand
- Entwickler pushen und pullen beliebig aus Central
- Keine Kontrollinstanz
- Einfach umzusetzen asd dfsdf



Repositories organisieren

Zentrales Rpositry

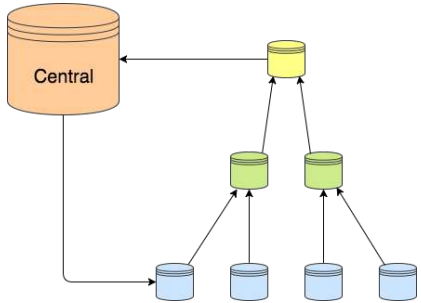
- Zentrales Repository als gemeinsamen Stand
- Entwickler pushen und pullen beliebig aus Central
- Keine Kontrollinstanz
- Einfach umzusetzen asd dfsdf

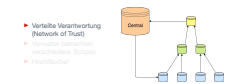


Repositories organisieren

Zentrales Rpositry

- Verteilte Verantwortung (Network of Trust)
- Verwalter betrachten verschiedene Scopes
- Hochflexibel

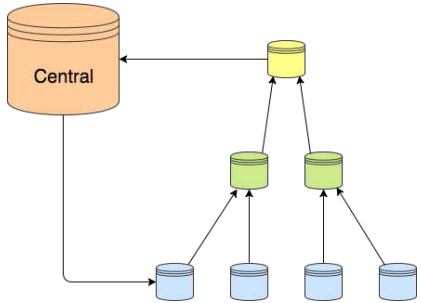


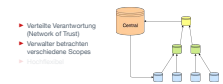


Repositories organisieren

Zentrales Rpositry

- Verteilte Verantwortung (Network of Trust)
- Verwalter betrachten verschiedene Scopes
- Hochflexibel

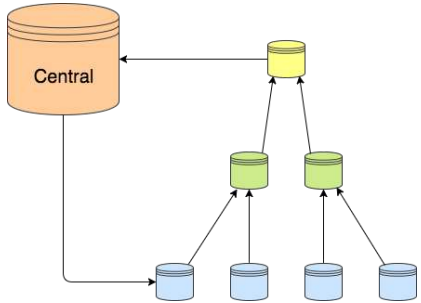


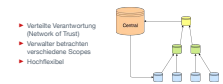


Repositories organisieren

Zentrales Rpositry

- Verteilte Verantwortung (Network of Trust)
- Verwalter betrachten verschiedene Scopes
- Hochflexibel

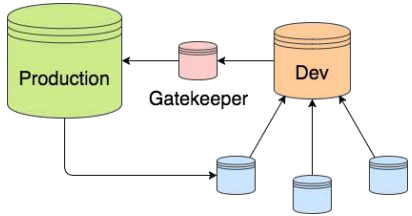




Repositories organisieren

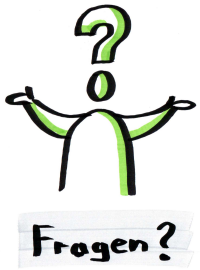
Zentrales Rpositry

- Continuous integration
- Automatische Tests
- Abnahme
- Einheitlicher Entwicklungsstand
- Releases



Fragen

Fragen?



Überblick

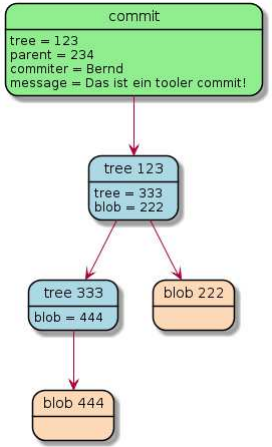
- Use Git the Git way
- Umfangreicher Werkzeugkofer
- Schnell
- Zielsicher
- Stabil

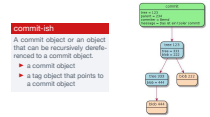
```
~/talks/latex  master
└─ git reflog | cat
d5f470e HEAD@{0}: commit: Git strukturen halb durch
3518d7b HEAD@{1}: reset: moving to HEAD
3518d7b HEAD@{2}: commit: Mockery Farbcodes in Listings geändert
ec41e65 HEAD@{3}: commit: Gitvortrag angefangen
4659920 HEAD@{4}: commit (amend): Komprimierter Vortrag. Bereinigtes leeres Projekt
c84a20c HEAD@{5}: commit: Komprimierter Vortrag. Bereinigtes leeres Projekt
1716399 HEAD@{6}: commit: Initial commit
9ac8809 HEAD@{7}: reset: moving to HEAD
9ac8809 HEAD@{8}: reset: moving to HEAD
9ac8809 HEAD@{9}: commit: Mockery Vortrag
d846250 HEAD@{10}: commit (amend): UnitTest vorversion; FeatherTemplateZR release
724716a HEAD@{11}: commit (amend): UnitTest vorversion; FeatherTemplateZR release
7fdc995 HEAD@{12}: commit (amend): UnitTest vorversion; FeatherTemplateZR release
1a0514e HEAD@{13}: commit: UnitTest vorversion; FeatherTemplateZR release
d854940 HEAD@{14}: commit (initial): Initial Commit

../talks/latex  master
└─ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Git/Git.pdf
        modified:   Git/Git.tex

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Git/Images/git_logo.png

no changes added to commit (use "git add" and/or "git commit -a")

../talks/latex  master
└─
```

commit-ish

A commit object or an object that can be recursively dereferenced to a commit object.

- a commit object
- a tag object that points to a commit object

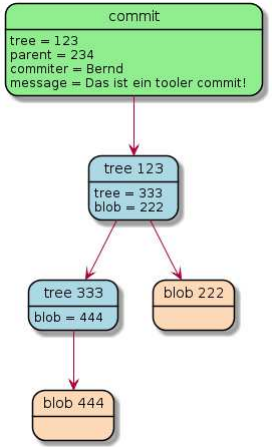


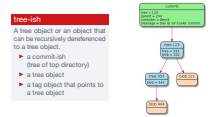


tree-ish

A tree object or an object that can be recursively dereferenced to a tree object.

- a commit-ish
  (tree of top directory)
- a tree object
- a tag object that points to a tree object

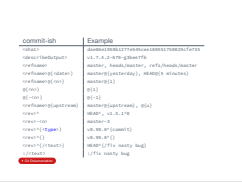




Git CLI
Adressierung von Git-Objekten

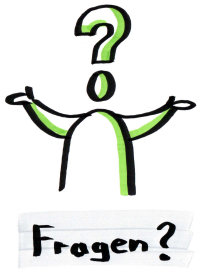
commit-ish	Example
<sha1>	dae86e1950b1277e545cee180551750029cfe735
<describeOutput>	v1.7.4.2-679-g3bee7fb
<refname>	master, heads/master, refs/heads/master
<refname>@{<date>}	master@{yesterday}, HEAD@{5 minutes}
<refname>@{<n>}	master@{1}
@{<n>}	@{1}
@{-<n>}	@{-1}
<refname>@{upstream}	master@{upstream}, @{u}
<rev>^	HEAD^, v1.5.1^0
<rev>~<n>	master~3
<rev>^{<type>}	v0.99.8^{commit}
<rev>^{}	v0.99.8^{}
<rev>^{/<text>}	HEAD^{/fix nasty bug}
:/<text>	:/fix nasty bug

▸ Git Dokumentation



Fragen

Fragen?



git reflog

Git Dokumentation

- Historie der letzten Git-Operationen
- Commit-ish benutzen um zum Stand zurück zu kehren

```
git reset --hard HEAD@{5}
```

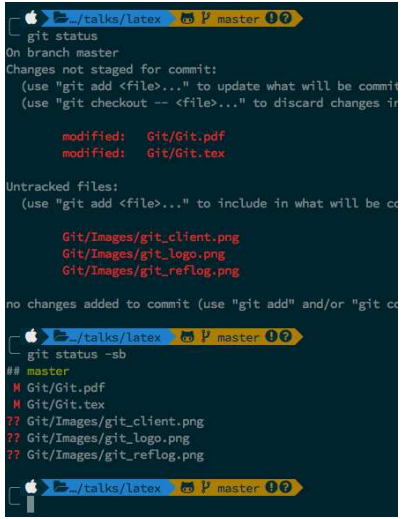
```
d5f470e (HEAD -> master) HEAD@{0}: commit: G
3518d7b HEAD@{1}: reset: moving to HEAD
3518d7b HEAD@{2}: commit: Mockery Farbcodes
ec41e65 HEAD@{3}: commit: Gitvortrag angefan
4659920 HEAD@{4}: commit (amend): Komprimier
c84a20c HEAD@{5}: commit: Komprimierter Vort
1716399 HEAD@{6}: commit: Initial commit
9ac8809 HEAD@{7}: reset: moving to HEAD
9ac8809 HEAD@{8}: reset: moving to HEAD
9ac8809 HEAD@{9}: commit: Mockery Vortrag
d846250 HEAD@{10}: commit (amend): UnitTest
724716a HEAD@{11}: commit (amend): UnitTest
7fdc995 HEAD@{12}: commit (amend): UnitTest
1a0514e HEAD@{13}: commit: UnitTest vorversi
d854940 HEAD@{14}: commit (initial): Initial
(END)
```

git status

Git Dokumentation

- Working mode (Rebasing, Merging …)
- Branch, Remotes
- Staged, Unstaged

```
git status

git status -sb #aka STFU-Mode
```



Add
Dateien stagen

git add

Git Dokumentation

- Fügt unbekannte Dateien zum index hinzu
- Staged Änderungen an bekannten Datien

```
git add NewFile.php
git add --all

git add -i #Interactive
```

```
~/.../git_repositories/zooroyalweb  remotes/origin/HEAD
$ git status
HEAD detached at origin/master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .idea/inspectionProfiles/profiles_settings.xml

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    .idea/inspectionProfiles/profiles_settings.xml
        modified:   build/config/phpcs/ZooroyalDefault/ruleset.xml

~/.../git_repositories/zooroyalweb  remotes/origin/HEAD
$ git add .idea/inspectionProfiles/profiles_settings.xml
~/.../git_repositories/zooroyalweb  remotes/origin/HEAD
$ git status
HEAD detached at origin/master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   build/config/phpcs/ZooroyalDefault/ruleset.xml

no changes added to commit (use "git add" and/or "git commit -a")
~/.../git_repositories/zooroyalweb  remotes/origin/HEAD
$ git add -i
           staged     unstaged path
  1:    unchanged        +0/-1 build/config/phpcs/ZooroyalDefault/ruleset.xml

*** Commands ***
  1: status       2: update       3: revert       4: add untracked
  5: patch        6: diff         7: quit         8: help
What now>
```

git rm / git mv

Git Dokumentation

- Git-Historie pflegen

```
git rm NewFile.php
git mv NFile.php NewFile.php
```

git checkout

Git Dokumentation

Stellt Git-Objekte im Workspace bereit

```
# checkout branch
git checkout master
# create and checkout branch
git checkout -b new_Branch
# checkout remote branch at date
git checkout origin/master@{
     yesterday}
# checkout single file
git checkout <commit-ish> -- <
     file_path>
# checkout last used branch
git checkout - # alt: @{-1}
```

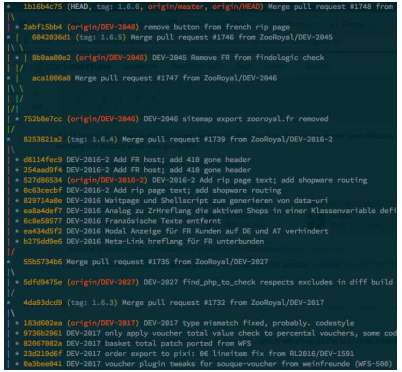
```
.../git_repositories/zooroyalweb  UnitTestPOC 2
 git checkout @{yesterday}
M       build/config/phpcs/ZooroyalDefault/ruleset.xml
Note: checking out '@{yesterday}'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at fa9e2faad zwischenstand mit Mockery

.../git_repositories/zooroyalweb  fa9e2fa...
 git checkout UnitTestPOC
M       build/config/phpcs/ZooroyalDefault/ruleset.xml
Previous HEAD position was fa9e2faad zwischenstand mit Mockery
Switched to branch 'UnitTestPOC'
Your branch is ahead of 'origin/UnitTestPOC' by 2 commits.
  (use "git push" to publish your local commits)

.../git_repositories/zooroyalweb  UnitTestPOC 2
 git checkout origin/UnitTestPOC
M       build/config/phpcs/ZooroyalDefault/ruleset.xml
Note: checking out 'origin/UnitTestPOC'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 960cda1f0 POC zum UnitTest für Plugins

.../git_repositories/zooroyalweb  remotes/origin/UnitTestPOC
```

git log

Git Dokumentation

- Zeigt die Historie
- Verschiedene Ansichten
- Durchsuchen

```
git log
# detailed view
git log -p
# short view with graph
git log --graph --online
# grep message for pattern
git log --grep=Bugfix
# show commits since date
git log --since=yesterday
```

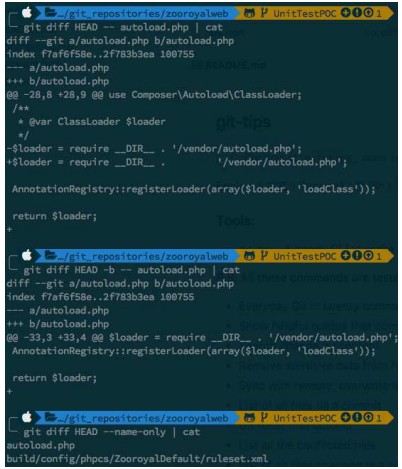


git diff

Git Dokumentation

- Zeigt unterschiede zwischen Commit-ishs
- Durchsuchen

```
git diff
# only names of unmerged
git diff --name-only --diff-filter
    =U
# changed chars only
git diff --word-diff
# ignore whitespace
git diff HEAD~4..HEAD~2 -b
```

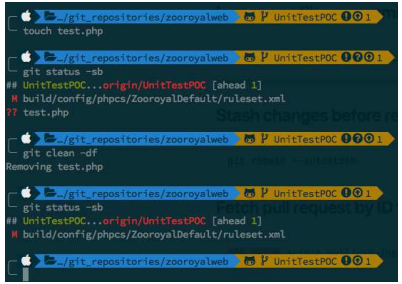


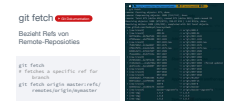
git clean

Git Dokumentation

- Entfernt nicht indizierte Dateien
- .gitignore wird beachtet

```
git clean
# deletes directories too
git clean -df
```



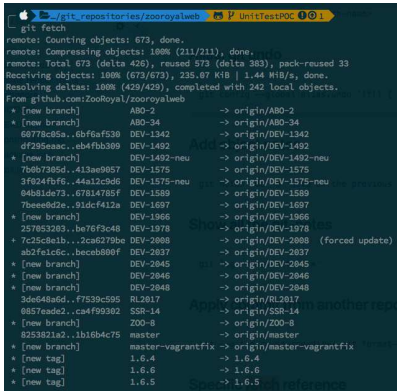


git fetch

Git Dokumentation

Bezieht Refs von Remote-Reposioties

```
git fetch
# fetches a specific ref for
     branch
git fetch origin master:refs/
     remotes/origin/mymaster
```



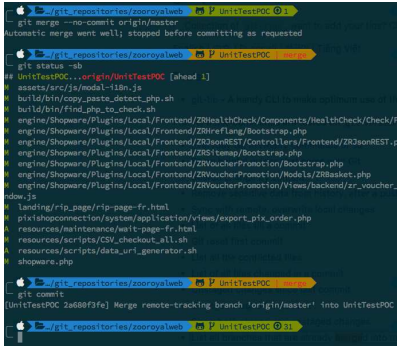
Merge
Zwei Branches zusammen führen

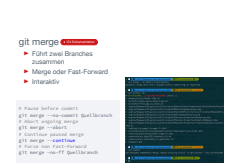
git merge

Git Dokumentation

- Führt zwei Branches zusammen
- Merge oder Fast-Forward
- Interaktiv

```
# Pause before commit
git merge --no-commit Quellbranch
# Abort ongoing merge
git merge --abort
# Continue paused merge
git merge --continue
# Force non Fast-Forward
git merge -no-ff Quellbranch
```





Pull

Daten aus dem anderem Branch integrieren

git pull

Git Dokumentation

- Integriert Daten aus anderem Branch
- Fetch und merge

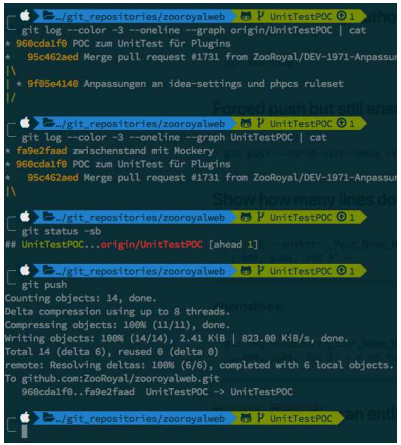
```
# fetch + merge FETCH_HEAD
git pull
# pull ref from remote
git pull origin master
# fetch + rebase instead
git pull --rebase master
# Config always rebase
git config --global pull.rebase
     true
```

git push

Git Dokumentation

Remote-Repository-Daten schreiben oder löschen

```
# force push to origin/branch
git push -f origin branch
# delete remote branch
git push origin :branch
# protected force push
git push --force-with-lease
```



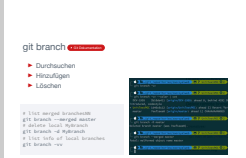
git revert

Git Dokumentation

Erzeugt einen neuen Commit der Änderungen eines anderen Commits rückgängig macht

```
# undo <commit-ish>
git revert <commit-ish>
# undo merge
git revert -m 1 <commit-ish>
```

```
…/git_repositories/zooroyalweb  UnitTestPOC
└ git status -sb
## UnitTestPOC...origin/UnitTestPOC

…/git_repositories/zooroyalweb  UnitTestPOC
└ touch test.txt

…/git_repositories/zooroyalweb  UnitTestPOC ?
└ git add test.txt

…/git_repositories/zooroyalweb  UnitTestPOC +
└ git commit -m "broken commit"
[UnitTestPOC e0a2e4039] broken commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt

…/git_repositories/zooroyalweb  UnitTestPOC ⦿ 1
└ git log -1 | cat
commit e0a2e4039d9298840ab35738a2e529d21c5bea98
Author: Sebastian Knott <s.knott@zooroyal.de>
Date:   Fri Jan 26 13:33:36 2018 +0100

    broken commit

…/git_repositories/zooroyalweb  UnitTestPOC ⦿ 1
└ git revert e0a2e4039d9298840ab35738a2e529d21c5bea98
[UnitTestPOC 1d46c6c12] Revert "broken commit"
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 test.txt

…/git_repositories/zooroyalweb  UnitTestPOC ⦿ 2
└ git log -2 --oneline --color | cat
1d46c6c12 Revert "broken commit"
e0a2e4039 broken commit
```

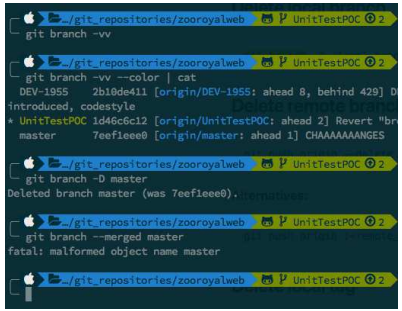



git branch

Git Dokumentation

- Durchsuchen
- Hinzufügen
- Löschen

```
# list merged branchesNN
git branch --merged master
# delete local MyBranch
git branch -d MyBranch
# list info of local branches
git branch -vv
```



Weiteres

- Einsteigerkurs von GitHub
- Roger Dudlers Git Guide
- Git Tricks

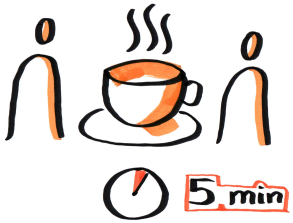
- Einsteigerkurs von GitHub
- Roger Dudlers Git Guide
- Git Tricks

Danke für eure Aufmerksamkeit!

Danke für eure Aufmerksamkeit!

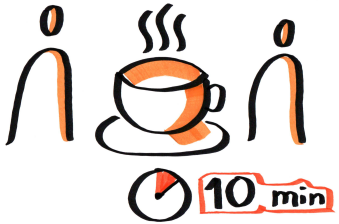
Pause

5 Minuten Pause



Pause

10 Minuten Pause



Pause
Pause

Fragen

Fragen?