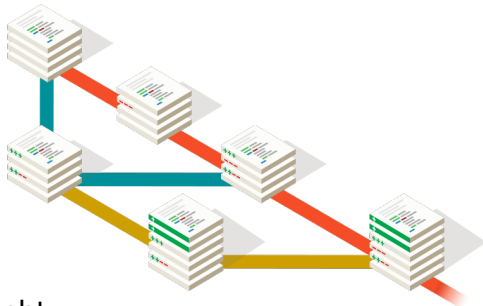
Git-Workshop

Sven Greiner

24. März 2017

Versionsverwaltung

- Was eine Versionsverwaltung ist:
  - Erfassen von Änderungen
  - Archiv mit Zeitstempel und Autor



- Wozu man eine Versionsverwaltung braucht:
  - Herausfinden, wann etwas geändert wurde
  - Änderungen rückgängig machen
  - Mit mehreren Leuten zusammenarbeiten

Was ihr hier lernt . . .

. . . reicht für **95%** aller Fälle.

Was ihr hier lernt . . .

. . . reicht für **95%** aller Fälle.

Für den Rest gibt es Google.

Die Kommandozeile ist dein Freund

Wie ist dein Name?

- Einmalig deinen Namen und deine E-Mail-Adresse festlegen:

```
$ git config --global user.name "Dein Name"
$ git config --global user.email foobar@example.com
```

- Wird in `~/.gitconfig` gespeichert
- Viele weitere Einstellungen über `git config` möglich

Alles hat ein Anfang: Init

1. Konsole öffnen
2. Verzeichnis betreten
3. `$ git init`
4. Fertig

Alles hat ein Anfang: Init

1. Konsole öffnen
2. Verzeichnis betreten
3. `$ git init`
4. Fertig

- Lege deinen Namen und deine E-Mail-Adresse fest
- Erstelle einen neuen Ordner und lege ein Git-Repository an

```
$ git config --global user.name . . .
$ git config --global user.email . . .
```

Mein erster Commit

- (Alle) Dateien in den *Index* hinzufügen:

```
$ git add -A
```

- Den Commit erzeugen:

```
$ git commit
```

Mein erster Commit

- (Alle) Dateien in den *Index* hinzufügen:

  ```
  $ git add -A
  ```

- Den Commit erzeugen:

  ```
  $ git commit
  ```

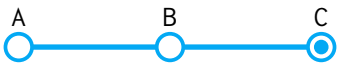
Mein erster Commit

- (Alle) Dateien in den *Index* hinzufügen:

  ```
  $ git add -A
  ```

- Den Commit erzeugen:

  ```
  $ git commit
  ```



Ein Commit im Detail

```
80cadb22bcd2cc2dd4b9a337. . .
SammysHP <sven@sammyshp.de>
Fri Mar 24 14:42:14 2017 +0100
Kurzbeschreibung
Und ein längerer Text, welcher
diesen Commit genauer beschreibt.
```

SHA1-Hash

Autor

Zeit

Kurzbeschreibung (eine Zeile)

Ausführliche Beschreibung

Und die Änderungen

Vereinfachung und Ausnahmen

- Alle bereits versionierten Dateien committen:

  ```
  $ git commit -a
  ```

- Änderungen am Index (`git add`) rückgängig machen:

  ```
  $ git reset [-- <DATEI>]
  ```

- Dateien ignorieren: `.gitignore`
  - Textdatei im Stammverzeichnis des Repository
  - Ein Dateiname pro Zeile

- Erstelle ein paar Dateien, füge sie dem Index hinzu und committe sie!
- Ändere Dateien, füge welche hinzu und lösche eine Datei. Erstelle zwischendurch Commits.
- Was konntest du beobachten? Worauf musstest du achten?

```
git add -A,  git add <DATEI>
git commit,  git commit -a,  git commit <DATEI>
git status
```

Was war, was ist und was sein wird

Was geändert wurde

- Zwischen Workspace und Index

```
$ git diff
```

- Zwischen zwei Commits

```
$ git diff A..B
```

- In einem Commit

```
$ git show A
```

History

- Mit viel Text

```
$ git log
```

- „Grafisch“

```
$ git log --graph --oneline
```

Aktueller Status

```
$ git status
```

- Schau dir an, was du gemacht hast.
- Probier auch mal, Dateien zu ändern (ohne sie zu committen) und schau dir die Änderungen an.
- Finde heraus, wie man sich eine Liste der geänderten Dateien im Log anschauen kann.

```
git diff,  git diff A..B
git show,  git show A
git log,  git status,  man git-log
```

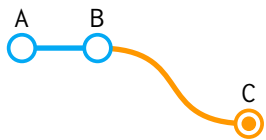
Auf verzweigten Wegen: Branches

Auf verzweigten Wegen: Branches

```
$ git checkout -b <NAME>
```

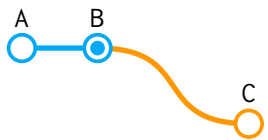
Auf verzweigten Wegen: Branches

```
$ git checkout -b <NAME>
$ git commit -a
```



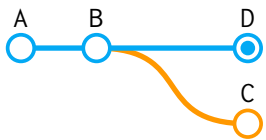
Auf verzweigten Wegen: Branches

```
$ git checkout -b <NAME>
$ git commit -a
$ git checkout master
```



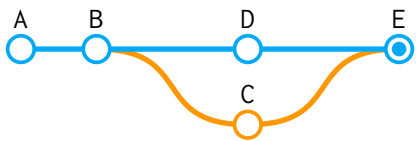
Auf verzweigten Wegen: Branches

```
$ git checkout -b <NAME>
$ git commit -a
$ git checkout master
$ git commit -a
```



Und wieder vereint: Merges

```
$ git checkout -b <NAME>
$ git commit -a
$ git checkout master
$ git commit -a
$ git merge <NAME>
```



Exkurs: References

- Zeiger auf Commits
- Häufige refs:
  - `HEAD` – „aktueller“ Commit
  - `master` – lokaler master-Branch
  - `origin/master` – master-Branch im Remote
- Tags
  - Beliebige Referenzen auf Commits
  - z. B. Releases
  - `git tag <NAME> [-a]`

Mergekonflikte

- Wenn etwas nicht passt . . .

```
$ git merge foo
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.
```

- Konflikte in der Datei hervorgehoben:

```
ullamcorper eget posuere diam. Morbi
<<<<<<< HEAD
nunc master-asdf, vestibulum ac accumsan
=======
nunc foobar, vestibulum ac accumsan
>>>>>>> foo
vitae, sollicitudin vitae est. Fusce
```

Branches auflisten und löschen

Auflisten

- Lokale Branches

  ```
  $ git branch
  ```

- Remote Tracking Branches

  ```
  $ git branch -r
  ```

- Alle Branches

  ```
  $ git branch -a
  ```

Löschen

- Lokale Branches

  ```
  $ git branch -d <NAME>
  ```

- Remote Tracking Branches

  ```
  $ git branch -r -d <NAME>
  ```

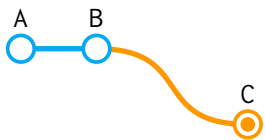
- Remote Branches

  ```
  $ git push origin :<NAME>
  ```

- Lege Branches an, mache Commits, wechsle Branches
- Merge Branches, achte auf Konflikte!
- Lösche einen Branch nach dem Mergen

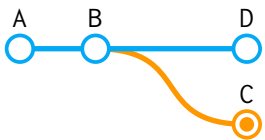
```
git branch <NAME>,  git checkout -b <NAME>
git checkout <NAME>,  git merge <NAME>
git branch -d <NAME>
```

Schönere History: Rebase



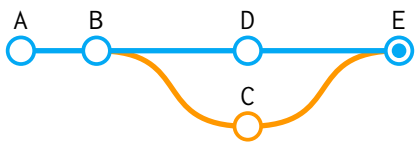
Schönere History: Rebase

- Commit in Master



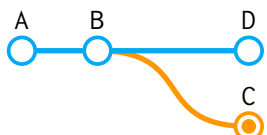
Schönere History: Rebase

- Commit in Master
- Merge würde einen Merge-Commit erzeugen



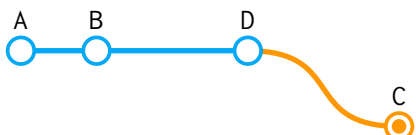
Schönere History: Rebase

- Commit in Master
- Merge würde einen Merge-Commit erzeugen
- Stattdessen:



Schönere History: Rebase

- Commit in Master
- Merge würde einen Merge-Commit erzeugen
- Stattdessen: `$ git rebase master`



Schönere History: Rebase

- Commit in Master
- Merge würde einen Merge-Commit erzeugen
- Stattdessen: `$ git rebase master`
- `$ git checkout master && git merge <NAME>`

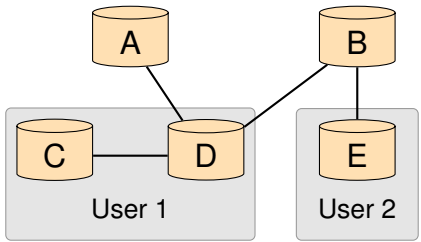
  $\Longrightarrow$ Fast-Forward

- Committe etwas in einem anderen Branch
- Erstelle einen Commit in `master`
- Rebase den ersten Branch auf `master`
- Mache einen Fast-Forward Merge in `master`

```
git checkout
git rebase
```

Verteiltes Arbeiten: Remotes

- Git ist dezentral – kann aber zentral genutzt werden



- Jedes Remote hat (meistens) alle Daten
- `$ git remote add <NAME> <URL>`

Exkurs: Branch Tracking

- Beziehungen zwischen Branches
- Lokale Branches, z. B. `master`
  - Nur lokal, wurden mit keinem anderen Remote geteilt
- Remote Branches, z. B. `origin/master`
  - Referenz zu einem Branch in einem anderen Remote
  - Kann man nicht bearbeiten
- Upstream Branches, z. B. `origin/master` für `master`
  - Der lokale Branch `master` ist eine Kopie des Branches `master` in dem Remote `origin`

clone, push, pull

- Ein vorhandenes Remote herunterladen:

  ```
  $ git clone <URL>
  ```

- Änderungen hochladen:

  ```
  $ git push [-u] [<REMOTE>]
  ```

  - `git commit` nicht vergessen
  - `-u` legt den *upstream tracking branch* fest

- Lokale Kopie aktualisieren:

  ```
  $ git pull [<REMOTE>]
  ```

In case of fire

1. git commit

2. git push

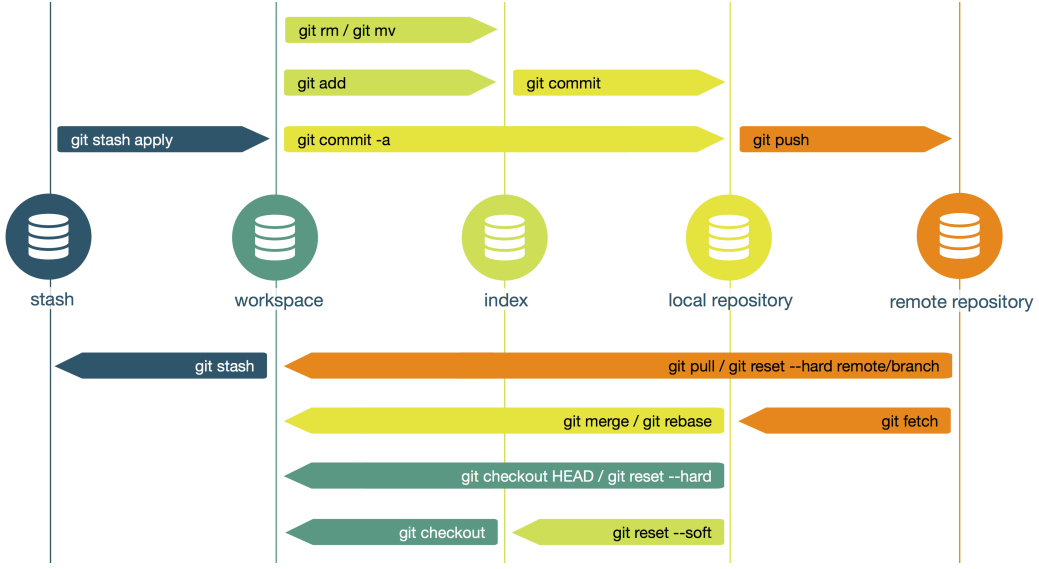
3. leave building

Pull im Detail

- `git pull` = `git fetch && git merge` (Jedenfalls fast)
- `git fetch` holt Daten von einem Remote und aktualisiert Remote Tracking Branches
- Merge lässt sich durch Rebase ersetzen:

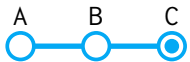
```
$ git pull --rebase
```

```
$ git config branch.<name>.rebase true
$ git config pull.rebase true
```


git rm / git mv
git add
git commit
git stash apply
git commit -a
git push
stash
workspace
index
local repository
remote repository
git stash
git pull / git reset --hard remote/branch
git merge / git rebase
git fetch
git checkout HEAD / git reset --hard
git checkout
git reset --soft

Wenn’s mal schief läuft: Revert

- Commit B macht Mist und soll rückgängig gemacht werden



Wenn’s mal schief läuft: Revert

- Commit B macht Mist und soll rückgängig gemacht werden
- `$ git revert B`

Wenn’s mal schief läuft: Revert

- Commit B macht Mist und soll rückgängig gemacht werden
- `$ git revert B`

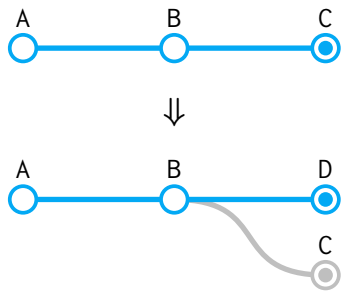
- Wir können Vergangenes nicht ändern!
- (Es gibt Möglichkeiten – die wollen wir aber möglichst nie nutzen)

Wenn’s mal schief läuft: Amend

- Situation:
  - Commit gemacht, etwas vergessen
  - Noch kein push!
- Lösung:
  - Änderungen durchführen
  - `$ git commit --amend -a`

Wenn’s mal schief läuft: Amend

- Situation:
  - Commit gemacht, etwas vergessen
  - Noch kein push!
- Lösung:
  - Änderungen durchführen
  - `$ git commit --amend -a`



- Ändere etwas am HEAD
- Reverte einen deiner Commits

```
git commit --amend
git revert
```

(Do NOT) Use the --force, Luke!

(Do NOT) Use the --force, Luke!

- Wir löschen damit Daten aus einem Remote!
- Andere haben diese Daten möglicherweise schon gepullt
  $\Longrightarrow$ Konflikte vorprogrammiert!

Und übrigens noch was . . .

- Git kann mehr!
- Manches ganz hilfreich:
  - bisect
  - blame
  - interactive rebase
  - stash
  - submodules
- Anderes braucht man nur selten

git-add
git-am
git-archive
git-bisect
git-branch
git-bundle
git-checkout
git-cherry-pick
git-citool
git-clean
git-clone
git-commit
git-describe
git-diff
git-fetch
git-format-patch
git-gc
git-grep
git-gui
git-init
git-log
git-merge
git-mv
git-notes
git-pull
git-push
git-rebase
git-reset
git-revert
git-rm
git-shortlog
git-show
git-stash
git-status
git-submodule

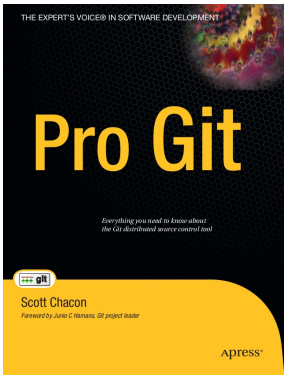
git-tag
git-worktree
gitk
git-config
git-fast-export
git-fast-import
git-filter-branch
git-mergetool
git-pack-refs
git-prune
git-reflog
git-relink
git-remote
git-repack
git-replace
git-annotate
git-blame
git-cherry
git-count-objects
git-difftool
git-fsck
git-get-tar-commit-id
git-help
git-instaweb
git-merge-tree
git-rerere
git-rev-parse
git-show-branch
git-verify-commit
git-verify-tag
git-whatchanged
gitweb
git-archimport
git-cvsexportcommit
git-cvsimport

git-cvsserver
git-imap-send
git-p4
git-quiltimport
git-request-pull
git-send-email
git-svn
git-apply
git-checkout-index
git-commit-tree
git-hash-object
git-index-pack
git-merge-file
git-merge-index
git-mktag
git-mktree
git-pack-objects
git-prune-packed
git-read-tree
git-symbolic-ref
git-unpack-objects
git-update-index
git-update-ref
git-write-tree
git-cat-file
git-diff-files
git-diff-index
git-diff-tree
git-for-each-ref
git-ls-files
git-ls-remote
git-ls-tree
git-merge-base
git-name-rev
git-pack-redundant

git-rev-list
git-show-index
git-show-ref
git-unpack-file
git-var
git-verify-pack
git-daemon
git-fetch-pack
git-http-backend
git-send-pack
git-update-server. . .
git-http-fetch
git-http-push
git-parse-remote
git-receive-pack
git-shell
git-upload-archive
git-upload-pack
git-check-attr
git-check-ignore
git-check-mailmap
git-check-ref-format
git-column
git-credential
git-credential-cache
git-credential-store
git-fmt-merge-msg
git-interpret-trailers
git-mailinfo
git-mailsplit
git-merge-one-file
git-patch-id
git-sh-i18n
git-sh-setup
git-stripspace

Hilfreiche Seiten

- man pages von Git: `$ man git-<BEFEHL>`
- Git Book: `https://git-scm.com/book/`
- Suchmaschinen!



Fragen Beispiele Ausprobieren