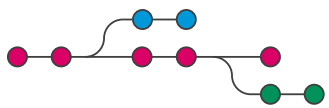
Hes·so VALAIS WALLIS

Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Systemdesign

Version Control



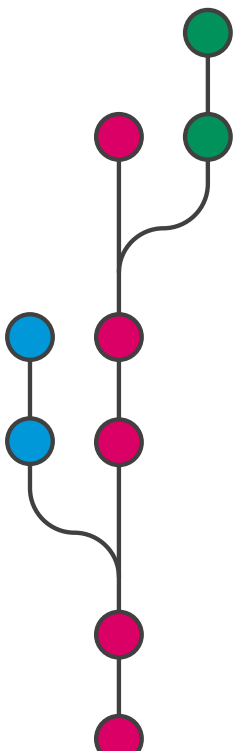
Studiengang Systemtechnik

Silvan Zahno silvan.zahno@hevs.ch

Ihr derzeitiges System

```
.
├── wichtige-datei copy.txt
├── wichtige-datei v1.txt
├── wichtige-datei v2.1 backup.txt
├── wichtige-datei v2.1 backup.txt.old
├── wichtige-datei v2.1.txt
├── wichtige-datei v2.txt
└── wichtige-datei.txt

1 Ordner, 7 Dateien
```

Version control



master v1.0.1 c4a9fee Merge branch 'release' into 'master' - Silvan Zahno <silvan.zahno@hevs.ch>

release 66ccde9 Merge branch 'develop' into 'release' - Silvan Zahno <silvan.zahno@hevs.ch>

2377d1e forgot to heat it up - Silvan Zahno <silvan.zahno@hevs.ch>

develop c43eda8 a whole cheese pizza, just for me - Silvan Zahno <silvan.zahno@hevs.ch>

fe3a9ee watter balloons (for the kids) - Silvan Zahno <silvan.zahno@hevs.ch>

v1.0.0 3b2d15b Merge branch 'release' into 'master' - Silvan Zahno <silvan.zahno@hevs.ch>

9dc3571 Merge branch 'develop' into 'release' - Silvan Zahno <silvan.zahno@hevs.ch>

fcd53f8 prepare for release - Silvan Zahno <silvan.zahno@hevs.ch>

9ddb202 added fabric softener - Silvan Zahno <silvan.zahno@hevs.ch>

a969898 added eggs - Silvan Zahno <silvan.zahno@hevs.ch>

ca10016 added milk - Silvan Zahno <silvan.zahno@hevs.ch>

e60c1b0 initial commit - Silvan Zahno <silvan.zahno@hevs.ch>

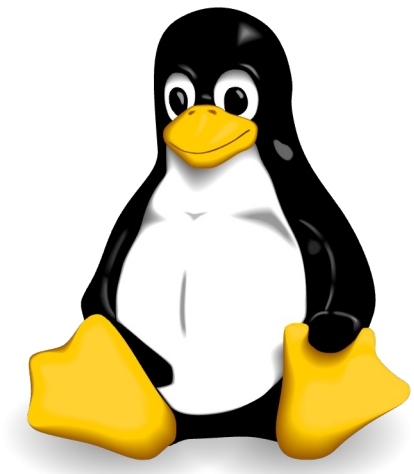
Possible Hilfsmittel

Git (git)

Subversion (svn)

Mercurial (hg)

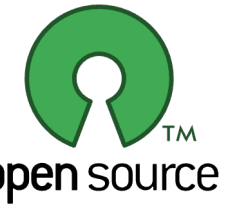
Linux Torvalds



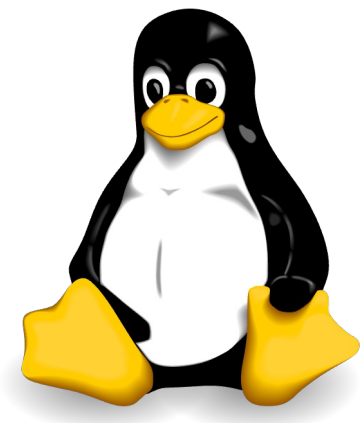
Linux (1991)

Git (2005)

Warum Linux braucht git



Linux hat sich in den letzten 30 Jahren zum grössten gemeinschaftlichen Entwicklungsprojekt in der Geschichte der Computertechnik entwickelt.



- 600 aktive Linux-Distributionen
- 85% aller Smartphones
- 500 Top-Supercomputer
- > 27,8 Millionen Codezeilen
- > 12'000 Mitwirkende
- > 1 Million commits

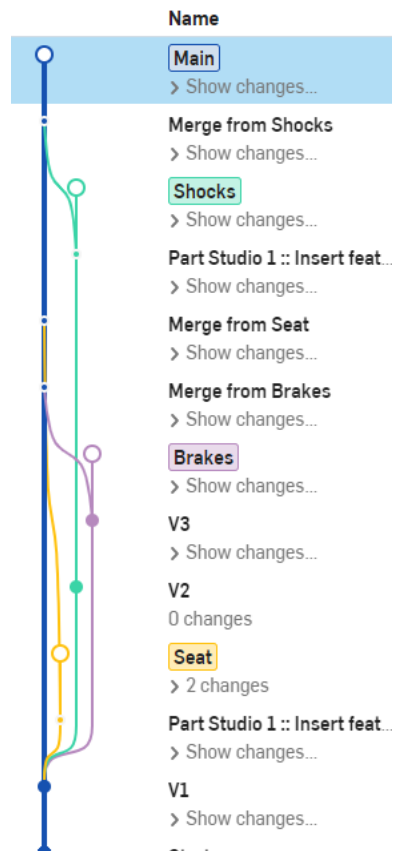
https://truelist.co/blog/linux-statistics/

Warum für ein Power&Control oder Design&Material git notwendig ist

Versionierung

- Verfolgung von Änderungen
- Zusammenarbeit
- Rollback-Fähigkeiten
- Dokumentation
- Bereitstellung

AUTODESK Vault



Git Plattformen

Gitlab

https://gitlab.com
https://gitlab.hevs.ch

Github

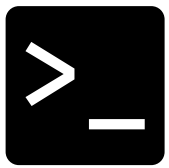
https://github.com

Bitbucket

https://bitbucket.com

Git Hilfsmittel

Kommandozeile



Fork

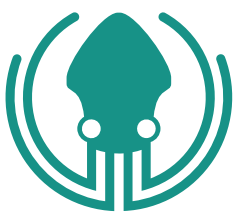
Sublime Merge

Tower

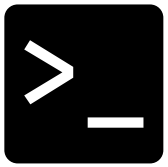
Git Cola

SmartGit

Git Kraken



Git Kommandozeile



```
git status
git diff
git add <file>
git diff --staged
git reset <file>
git commit -m "<commit message>"
git fetch <remote>
git merge <remote> <branch>
git push <remote> <branch>
git pull
```

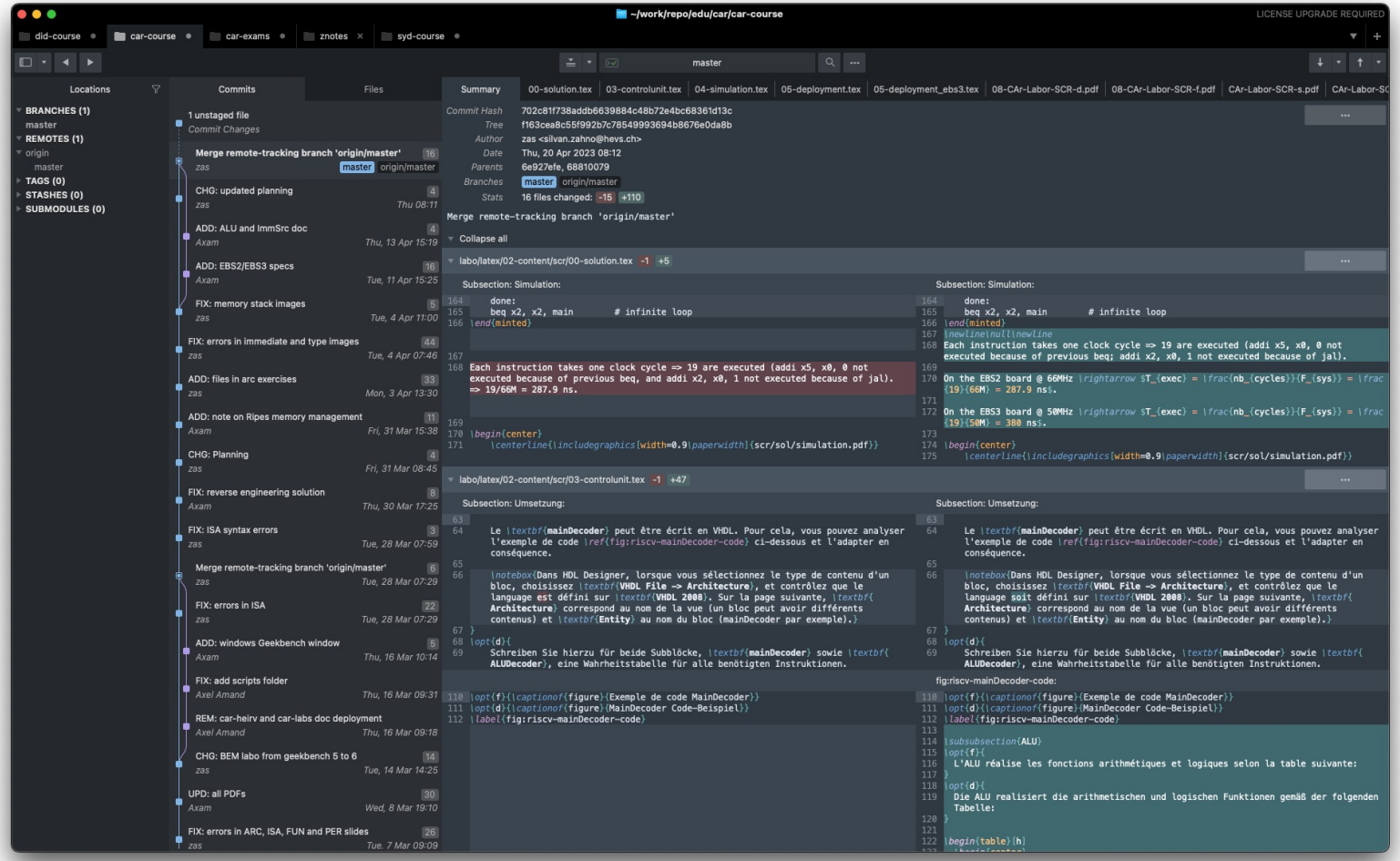
```
zas — zas@zac — ~ — -zsh — 99×52
Last login: Tue Mar  8 09:26:26 on ttys004
┌──(zas㉿zac)-[~] (base)
└─$ git --help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status

grow, mark and tweak your common history
   branch    List, create, or delete branches
   commit    Record changes to the repository
   merge     Join two or more development histories together
   rebase    Reapply commits on top of another base tip
   reset     Reset current HEAD to the specified state
   switch    Switch branches
   tag       Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch     Download objects and refs from another repository
   pull      Fetch from and integrate with another repository or a local branch
   push      Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

┌──(zas㉿zac)-[~] (base)
└─$
```

git Befehle

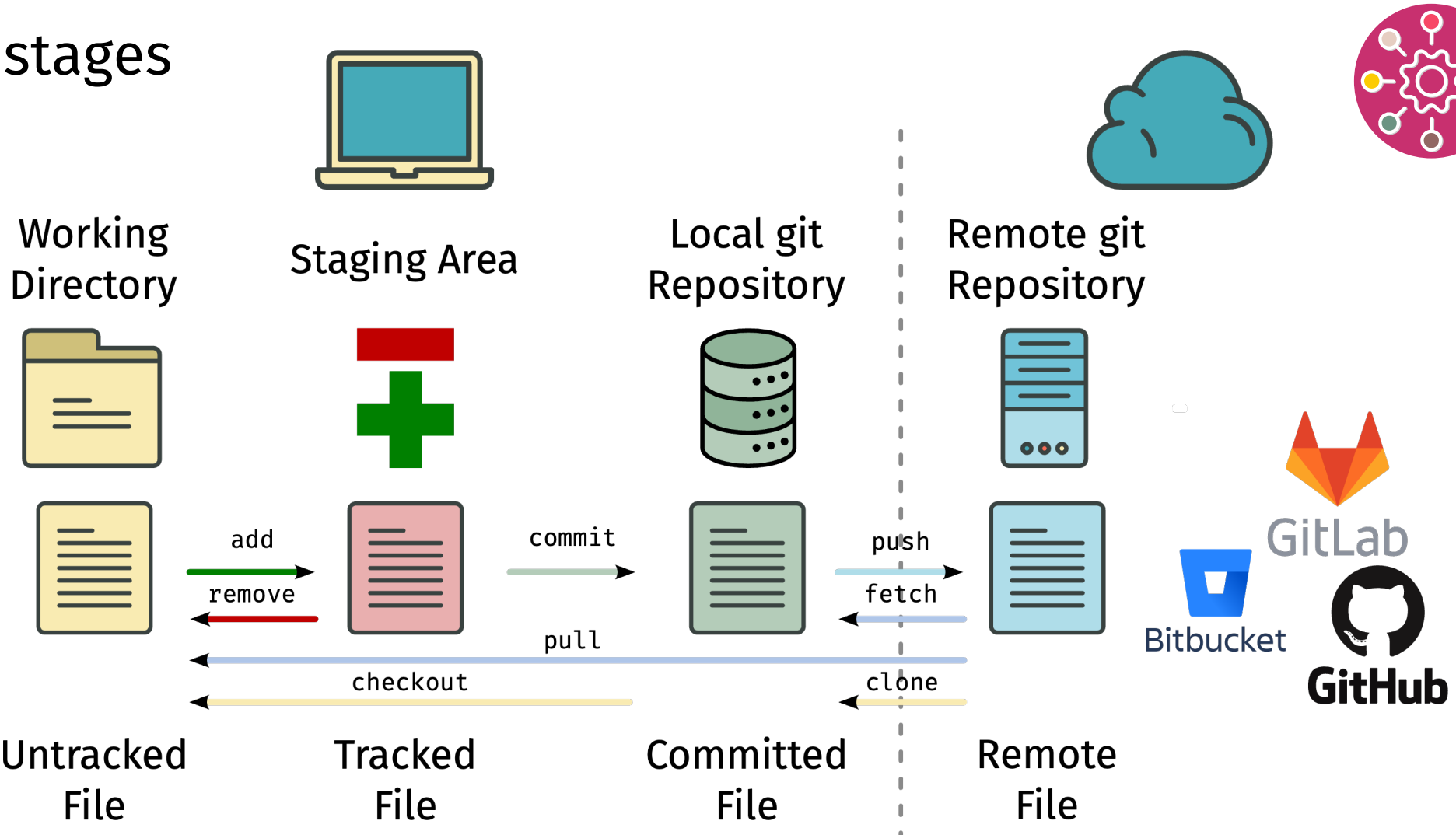
Befehl	Beschreibung
Eine working area beginnen	
clone	Klonen eines Repositorys in ein neues Verzeichnis
init	Erstellen eines leeren git-Repo oder Reinitialisieren eines vorhandenen Repo
An aktuellen Änderungen arbeiten	
add	Hinzufügen von Dateiinhalten zum Index
mv	Verschieben oder Umbenennen einer Datei, eines Verzeichnisses oder eines Symlinks
reset	Zurücksetzen des aktuellen HEAD auf den angegebenen Zustand
rm	Entfernen von Dateien aus dem Arbeitsbaum und aus dem Index
Untersuchen der Historie und des Status	
log	Übergabeprotokolle anzeigen
show	Verschiedene Arten von Objekten anzeigen
status	Show the working tree status

Befehl	Beschreibung
Wachsen, markieren und optimieren Sie Ihre gemeinsame Historie	
branch	Zweige auflisten, erstellen oder löschen Auschecken
checkout	Zweige wechseln oder Arbeitsbaumdateien wiederherstellen
commit	Änderungen an der Projektliste aufzeichnen
diff	Änderungen zwischen Commits, Commit und Arbeitsbaum anzeigen, etc.
merge	Zwei oder mehr Entwicklungsgeschichten zusammenführen
rebase	Commits auf einen anderen Basistipp anwenden tag
tag	Ein Tag-Objekt erstellen, auflisten, löschen oder verifizieren
Zusammenarbeiten	
fetch	Herunterladen von Objekten und Referenzen aus einem anderen Repo
pull	Abrufen und Integrieren aus einem anderen Projektarchiv oder einem lokalen Zweig schieben
push	Aktualisieren entfernter Referenzen zusammen mit zugehörigen Objekten

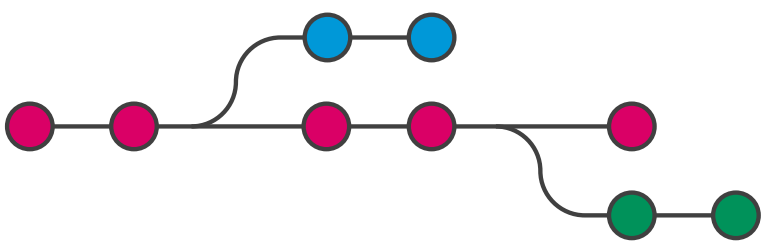
Sublime Merge



Git stages


Working Directory
Staging Area
Local git Repository
Remote git Repository
add
remove
commit
push
fetch
pull
checkout
clone
Untracked File
Tracked File
Committed File
Remote File
GitLab
Bitbucket
GitHub

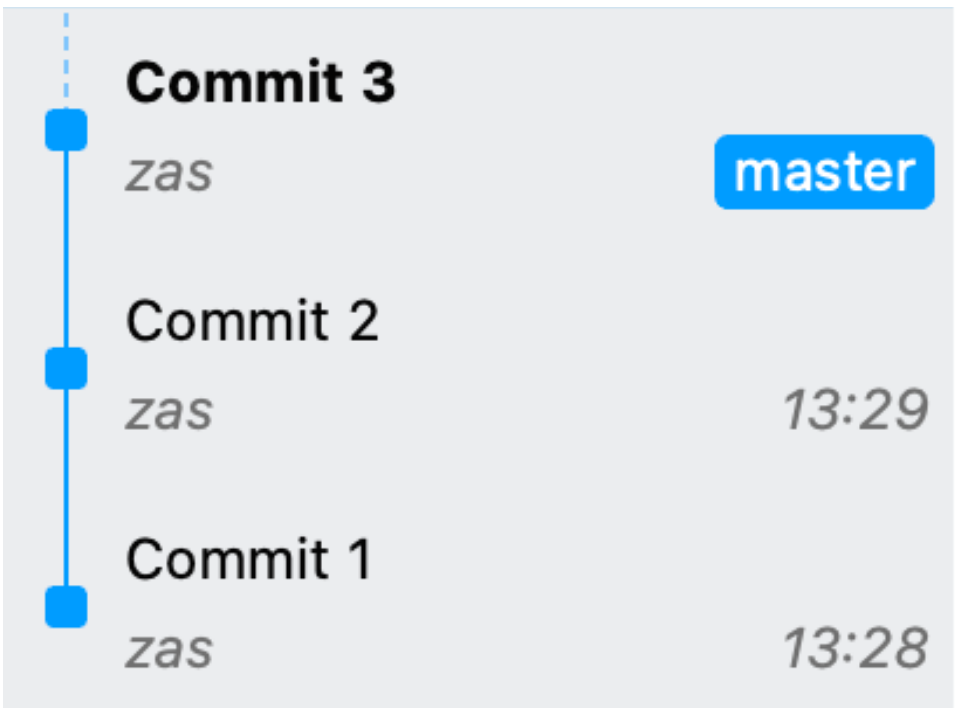
Git Branch und Merge Beispiele



Branch und Merge

Initial repo state

Jedes repo hat entweder ein `main` oder eine `master` branch als standart branch

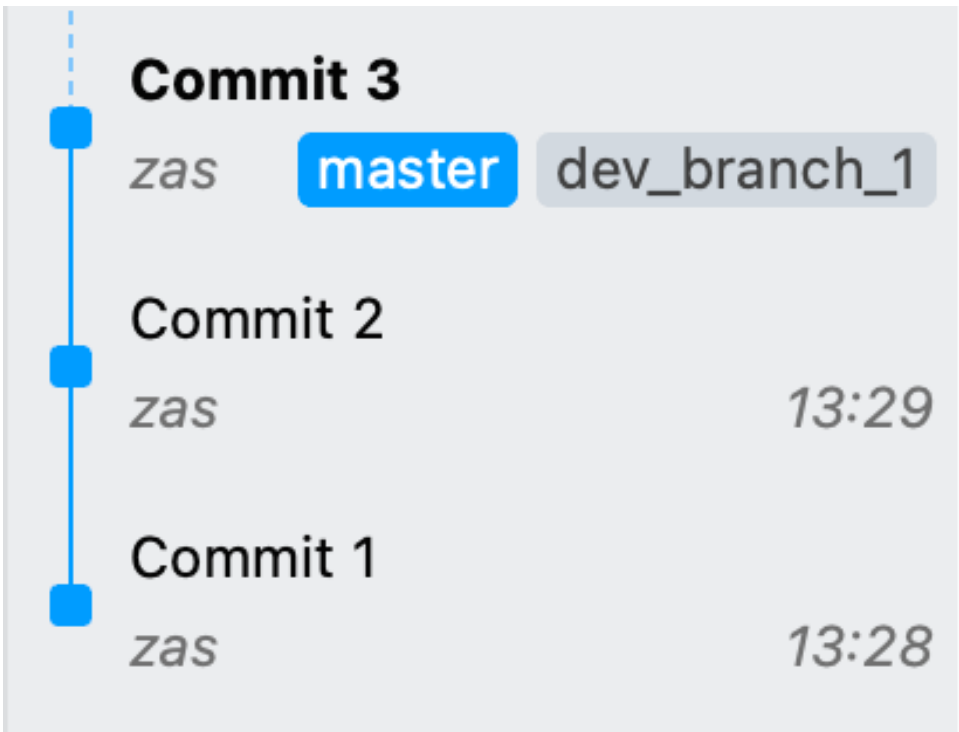


Branch und Merge

Create branch dev_branch_1

Erstelle branch dev_branch_1

```
$ git branch dev_branch_1
```



Branch und Merge

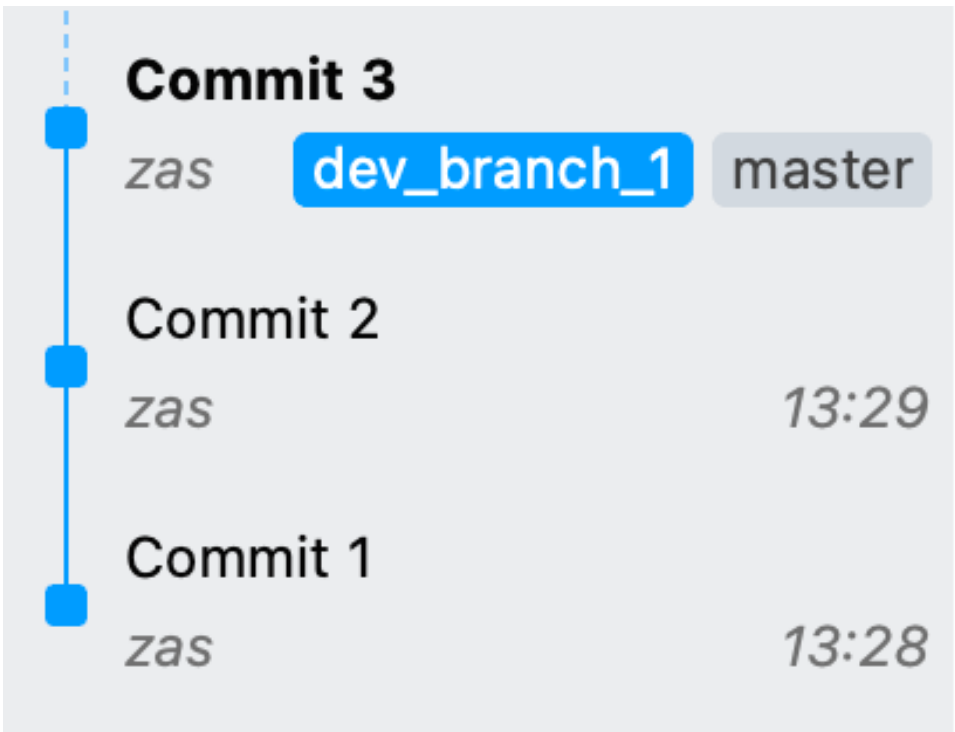
Checkout branch dev_branch_1

Prüfen Sie, in welchem Zweig wir uns befinden

```
$ git branch
```

Checkout branch dev_branch_1

```
$ git checkout dev_branch_1
```



Branch and Merge

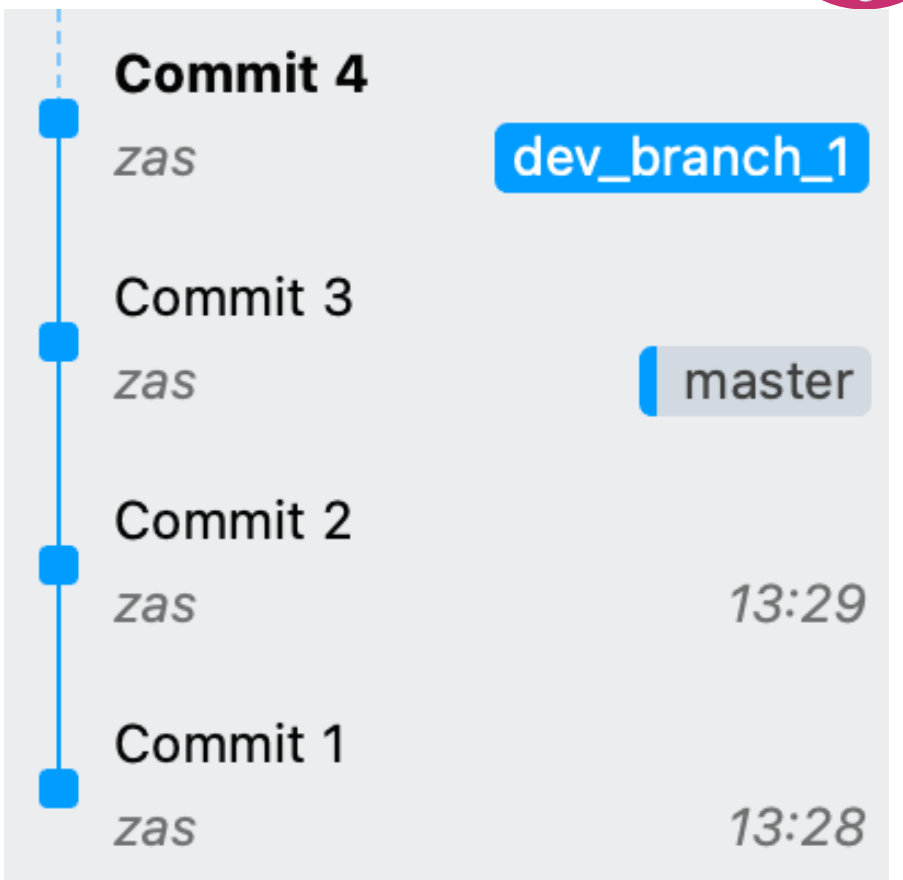
Commit on dev_branch_1

Stage new file

```
$ git add file.md
```

Commit stages files

```
$ git commit -m “Commit 4”
```



Branch und Merge

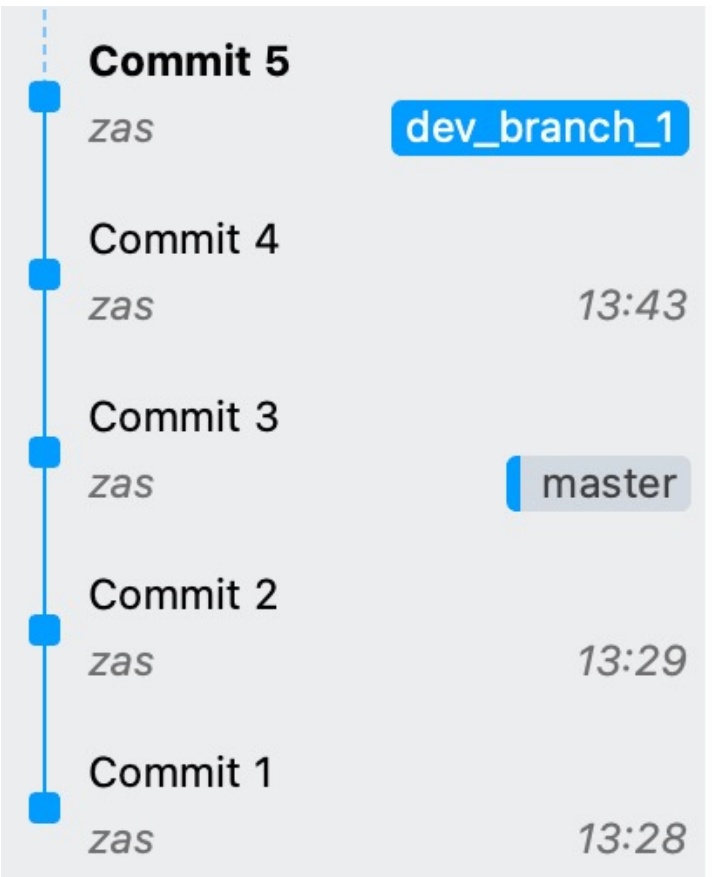
Commit on dev_branch_1

Stage new file

```
$ git add file.md
```

Commit stages files

```
$ git commit -m “Commit 5”
```



Branch und Merge

Commit on `master` branch

Checkout `master` branch

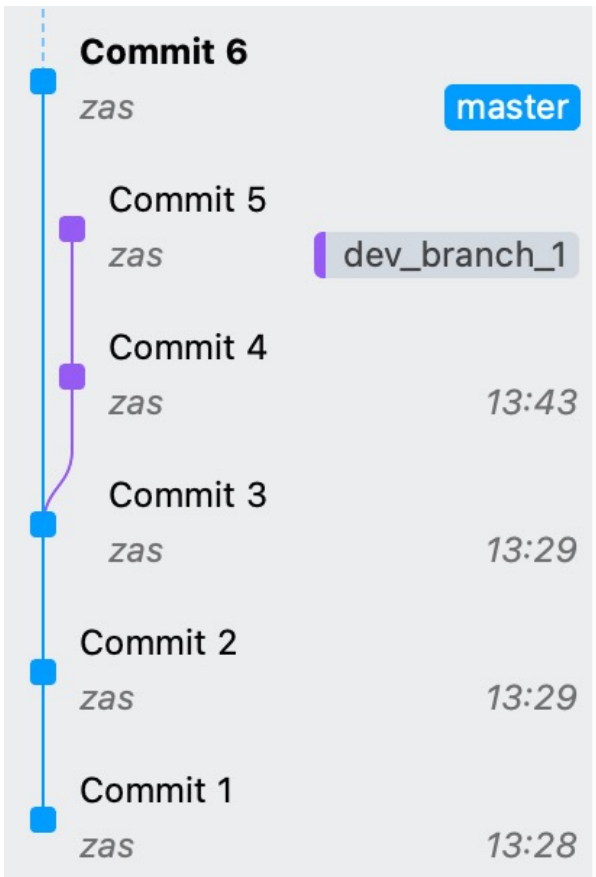
```
$ git checkout master
```

Stage new file

```
$ git add file.md
```

Commit stages files

```
$ git commit -m “Commit 6”
```

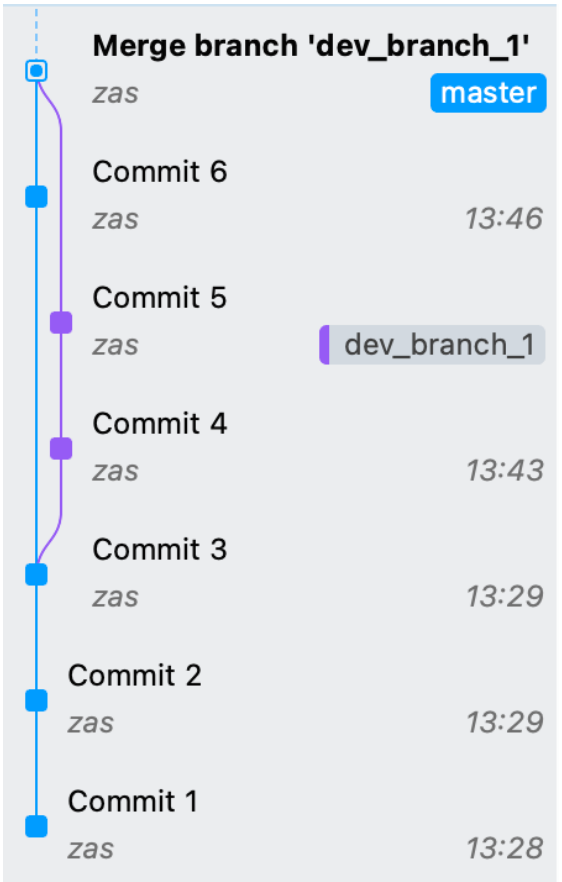


Branch und Merge

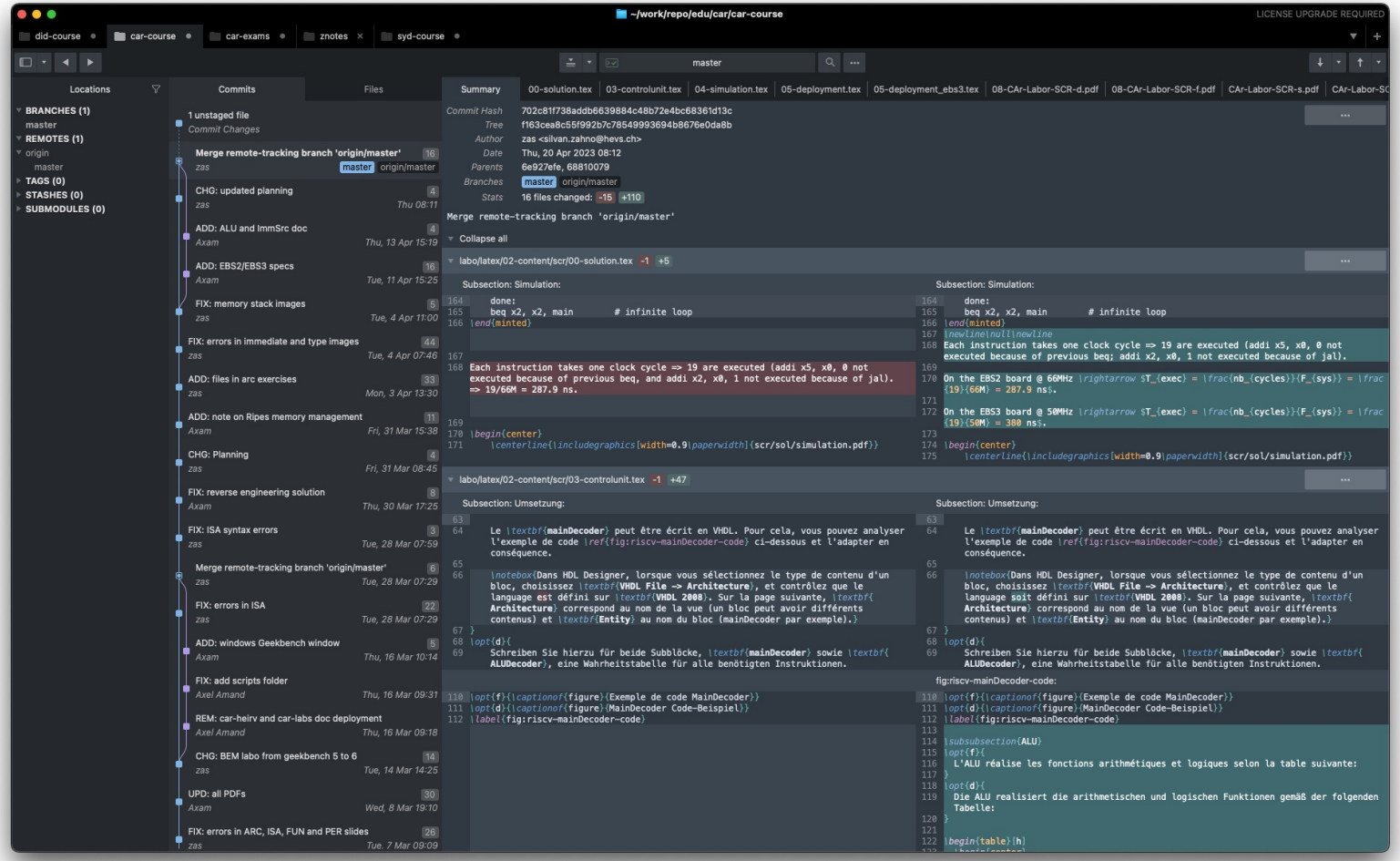
Three way merge `master` and `dev_branch_1`

Merge `dev_branch_1 into master`

```
$ git merge dev_branch_1
```

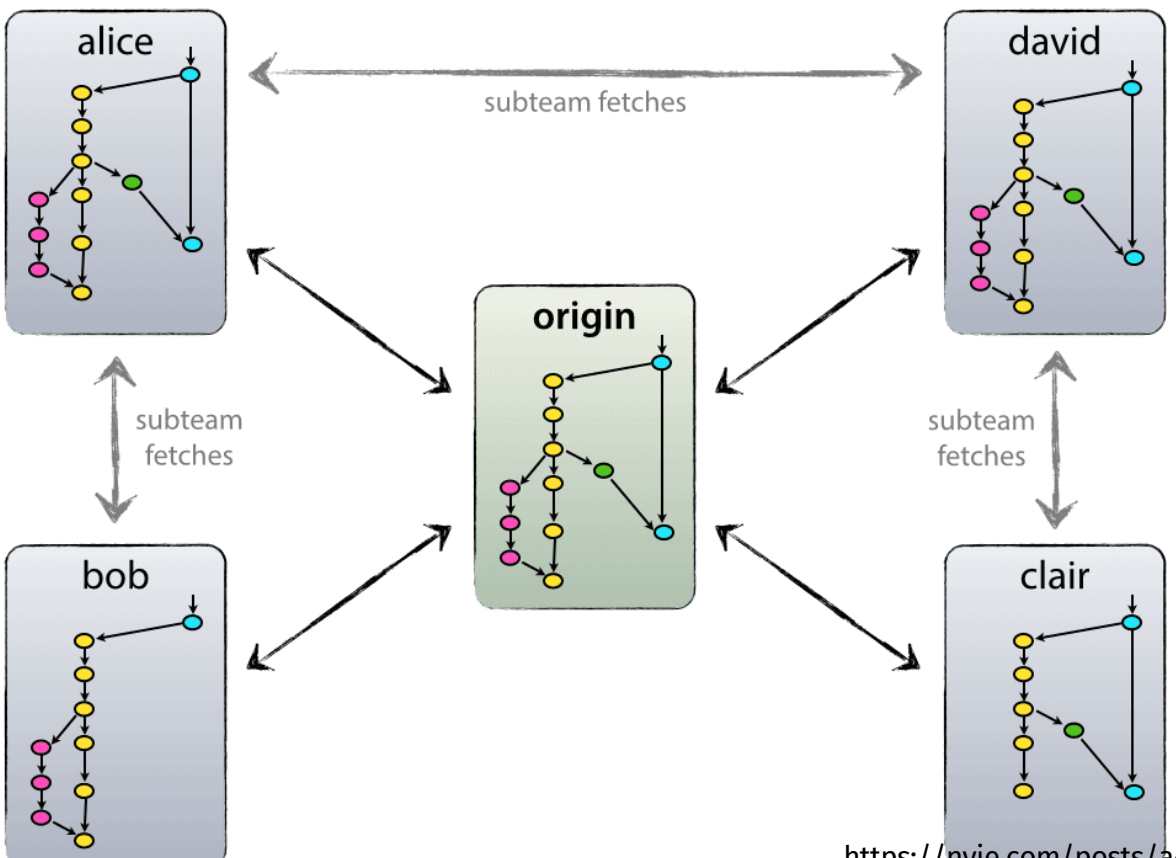


Demonstration



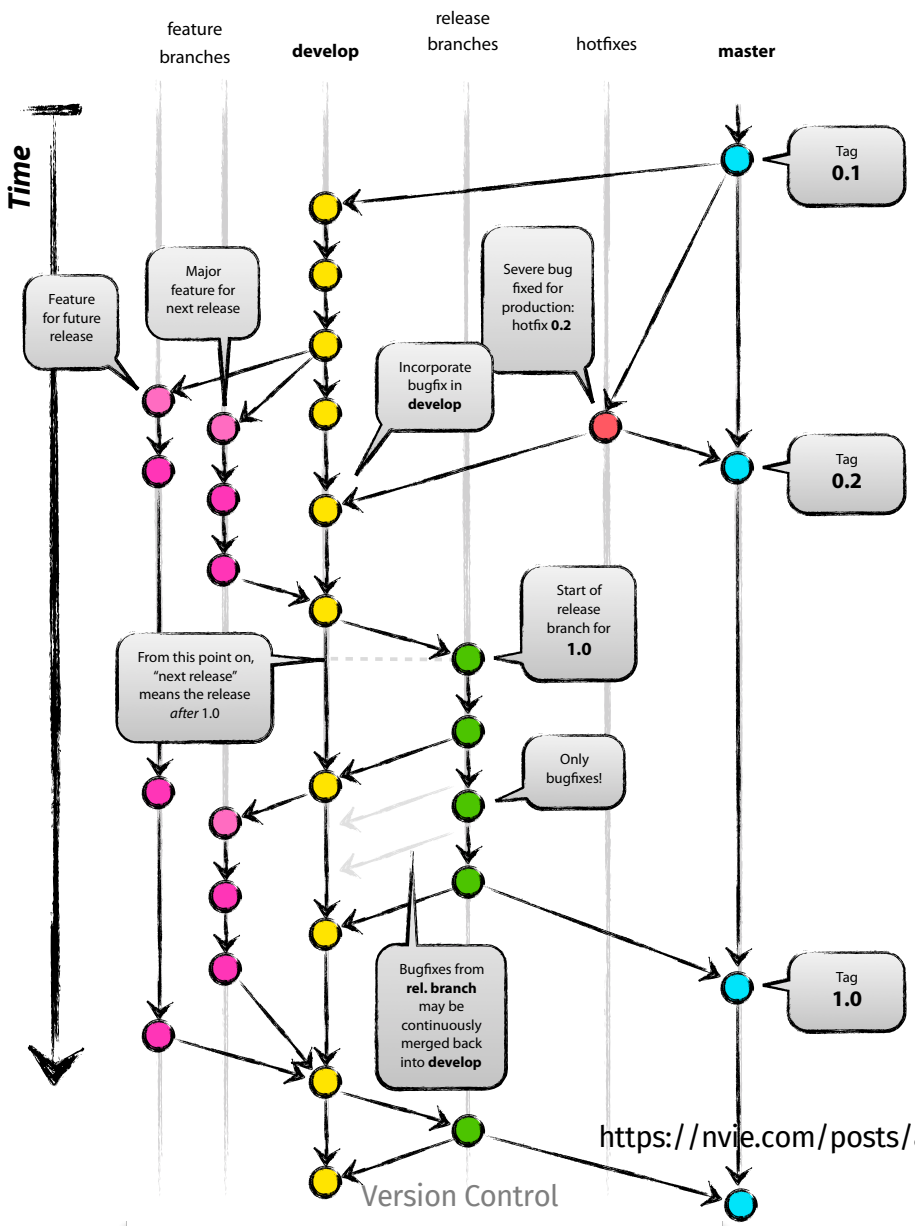
Gitflow

Gitflow Zusammenarbeit



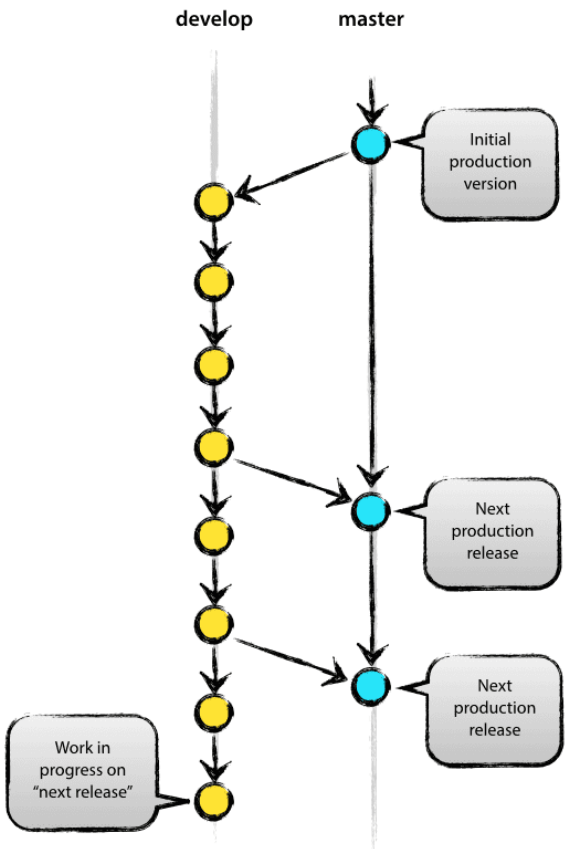
https://nvie.com/posts/a-successful-git-branching-model/

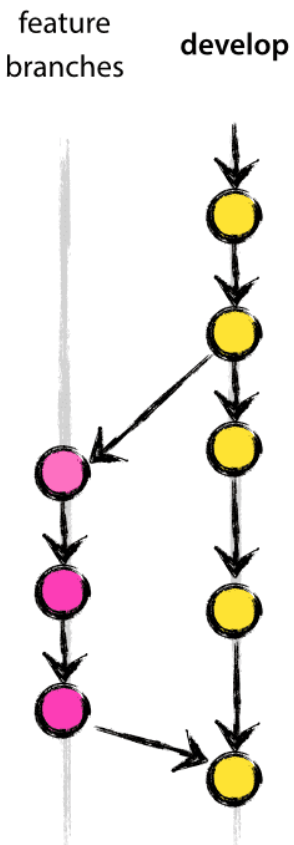
Gitflow

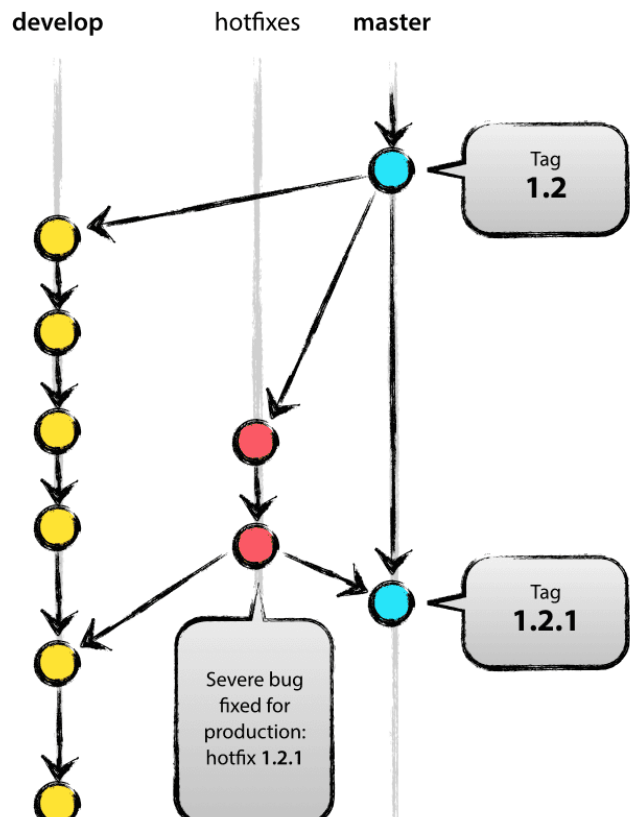


https://nvie.com/posts/a-successful-git-branching-model/

Gitflow Branching

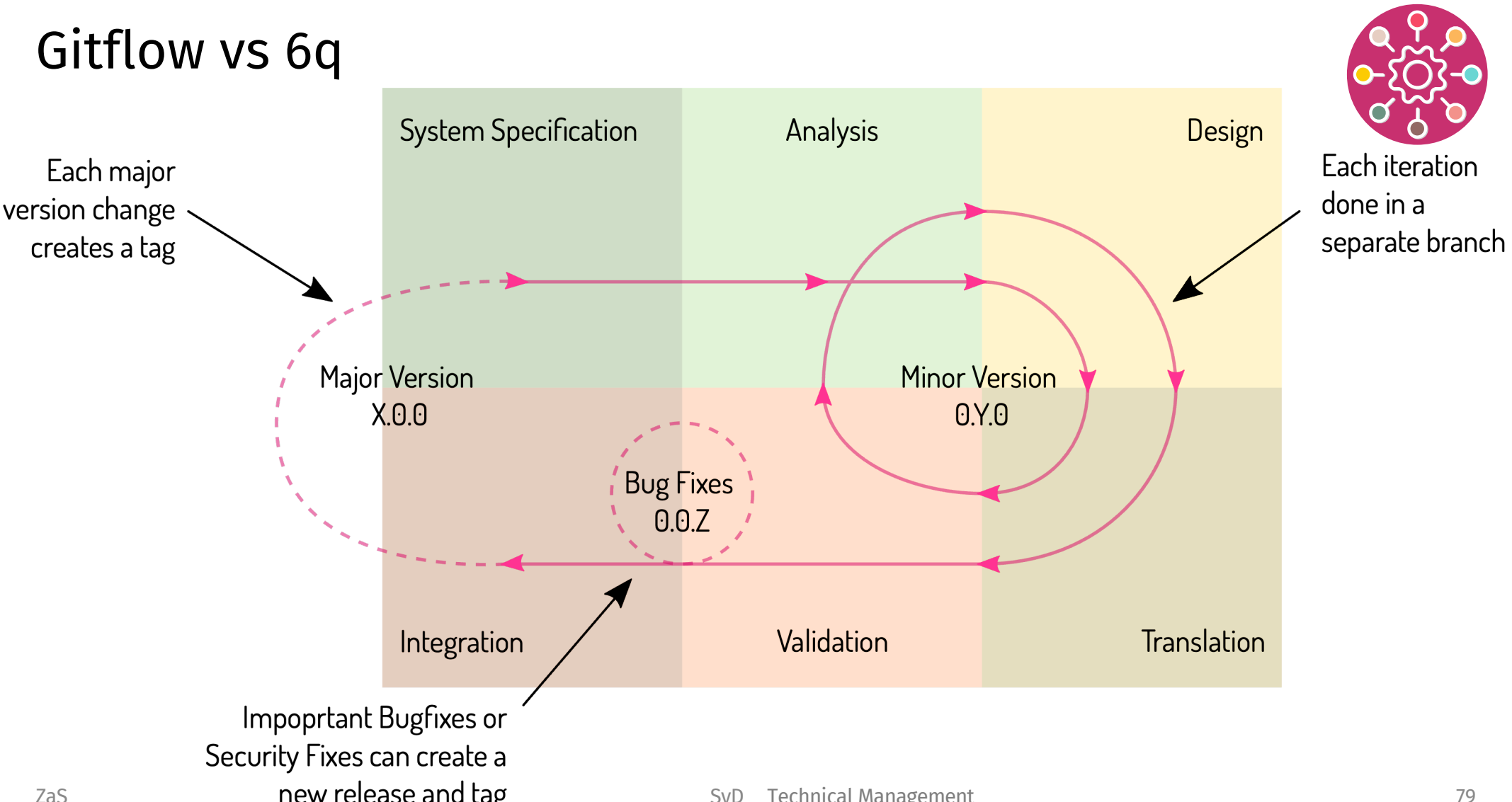






https://nvie.com/posts/a-successful-git-branching-model/

Gitflow vs 6q

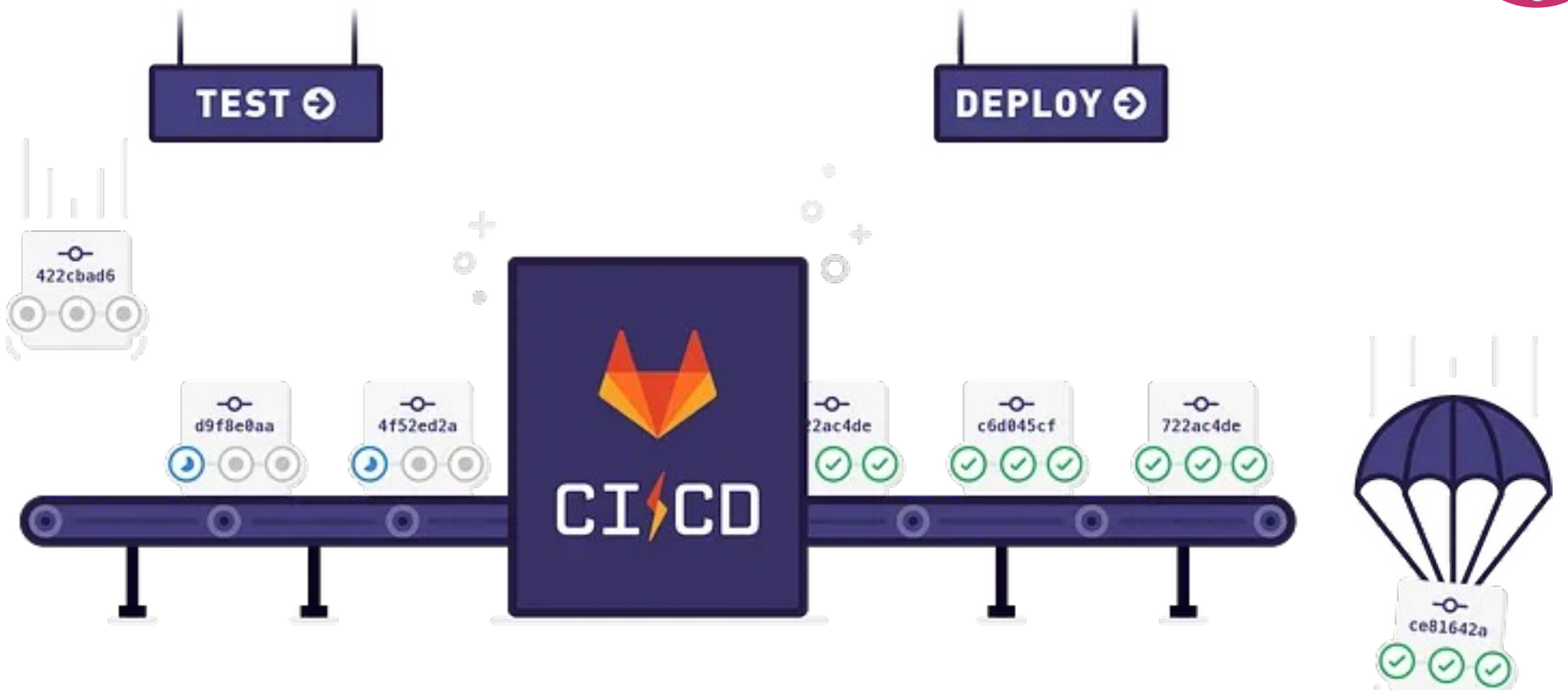


Git CI/CD

Was ist CI/CD?

- Bei der kontinuierlichen Integration (Continuous Integration, CI) werden Codeänderungen häufig in ein gemeinsames Repository integriert, das dann automatisch erstellt und getestet wird.
- Continuous Delivery (CD) geht noch einen Schritt weiter, indem Codeänderungen automatisch in produktionsähnlichen Umgebungen für weitere Tests und Validierungen bereitgestellt werden.
- Automatisierte Tests sind eine wichtige Komponente von CI/CD, da sie dazu beitragen, Bugs und andere Probleme frühzeitig im Entwicklungsprozess zu erkennen.
- Beliebte Tools sind GitLab CI/CD, Github Actions, Jenkins, CircleCI und Travis CI.

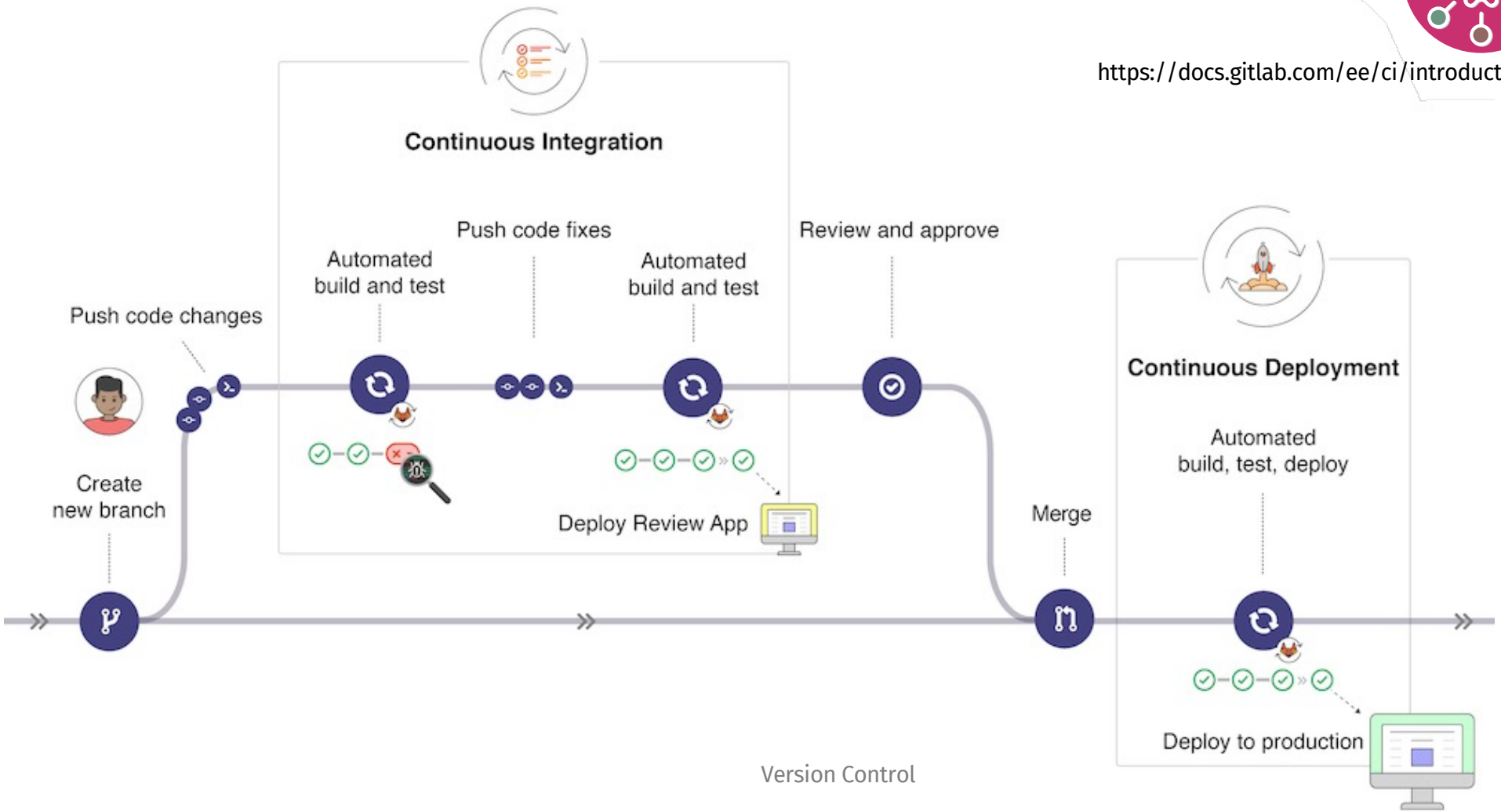
Was ist CI/CD?



https://about.gitlab.com/features/continuous-integration/

Gitlab Workflow

https://docs.gitlab.com/ee/ci/introduction/



Abschätzung von Aufgaben

Abschätzung 4

Erstelle einen Star Wars Star Destroyer (4784pcs)

X Punkt(e)

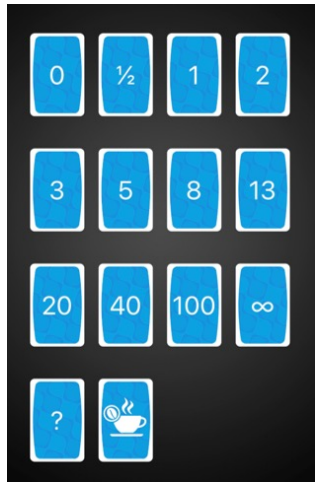
0	½	1	2
3	5	8	13
20	40	100	∞
?	☕		

Abschätzung von Aufgaben

Abschätzung 5

Erstelle einen Star Wars Star Destroyer (37pcs)

X Punkt(e)



Introduction to git - Part A

Installation & Setup

git

Contents

Introduction to git - Part B

git

Contents