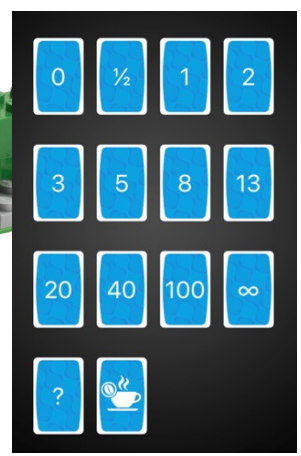
Abschätzung von Aufgaben

Abschätzung 2

Erstelle eine kleines Haus

X Punkt(e)

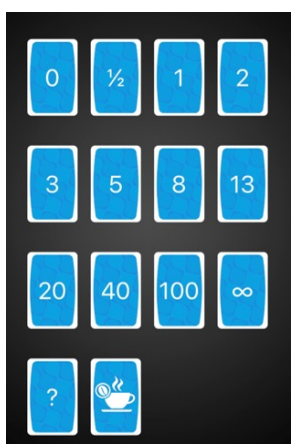


Abschätzung von Aufgaben

Abschätzung 3

Erstelle einen programmierbaren Rotober

X Punkt(e)

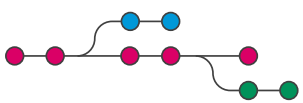


Hes·so VALAIS WALLIS

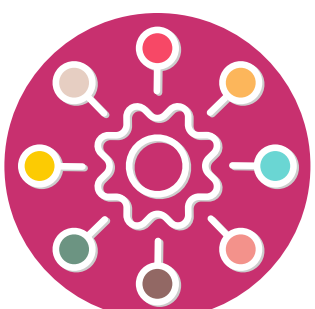
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Systemdesign

Version Control



Studiengang Systemtechnik

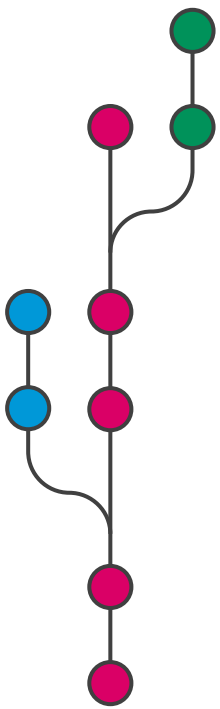


Silvan Zahno silvan.zahno@hevs.ch

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>

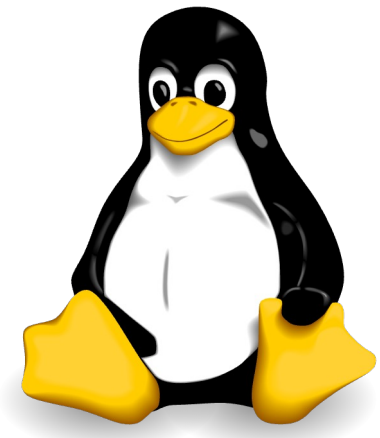
Possible Hilfsmittel

Git (git)

Subversion (svn)

Mercurial (hg)

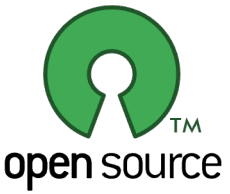
Linux Torvalds



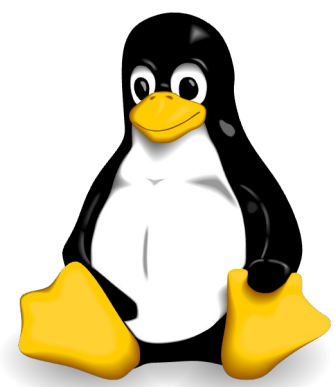
Linux (1991)

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/

Git Plattformen

Gitlab

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

Github

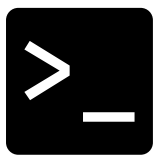
https://github.com

Bitbucket

https://bitbucket.com

Git Hilfsmittel

Kommandozeile



Fork

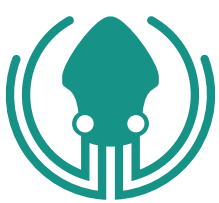
Sublime Merge

Tower

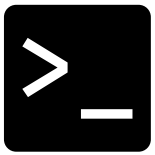
Git Cola

SmartGit

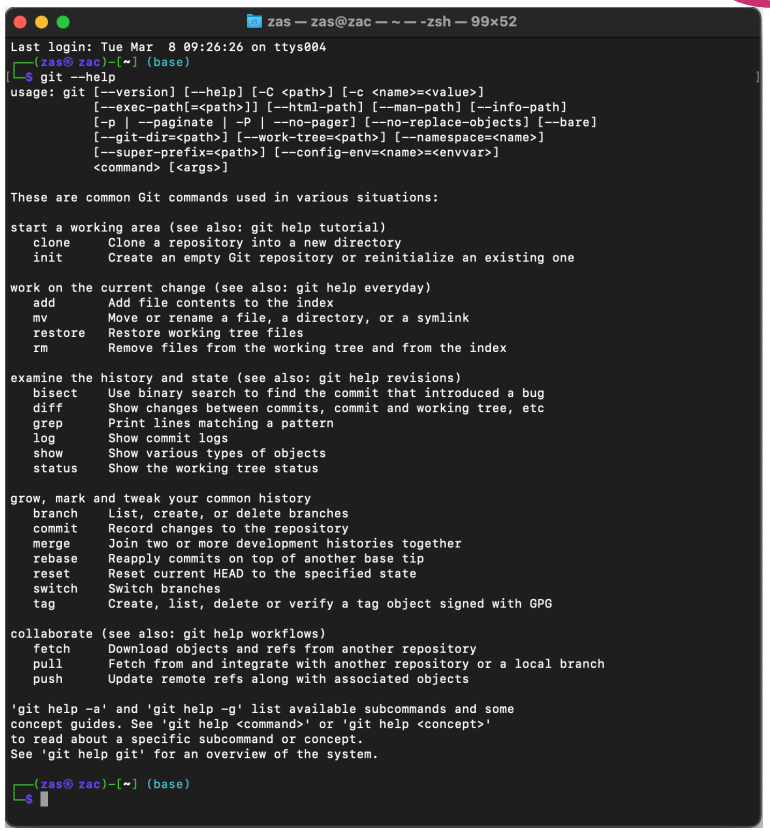
Git Kraken



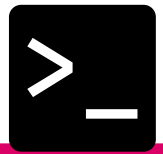
Git Kommandozeile



```
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
```



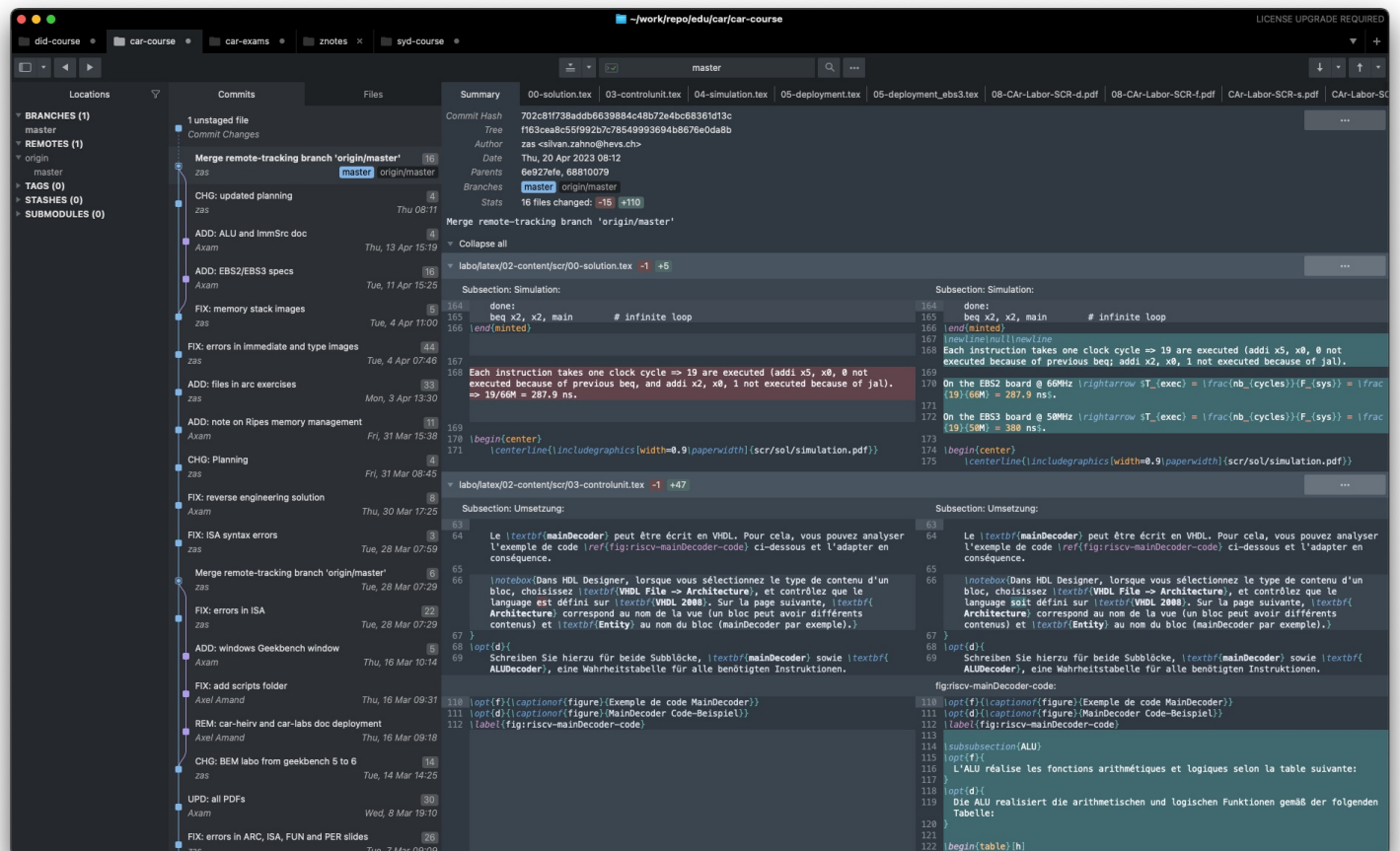
`git` Befehle



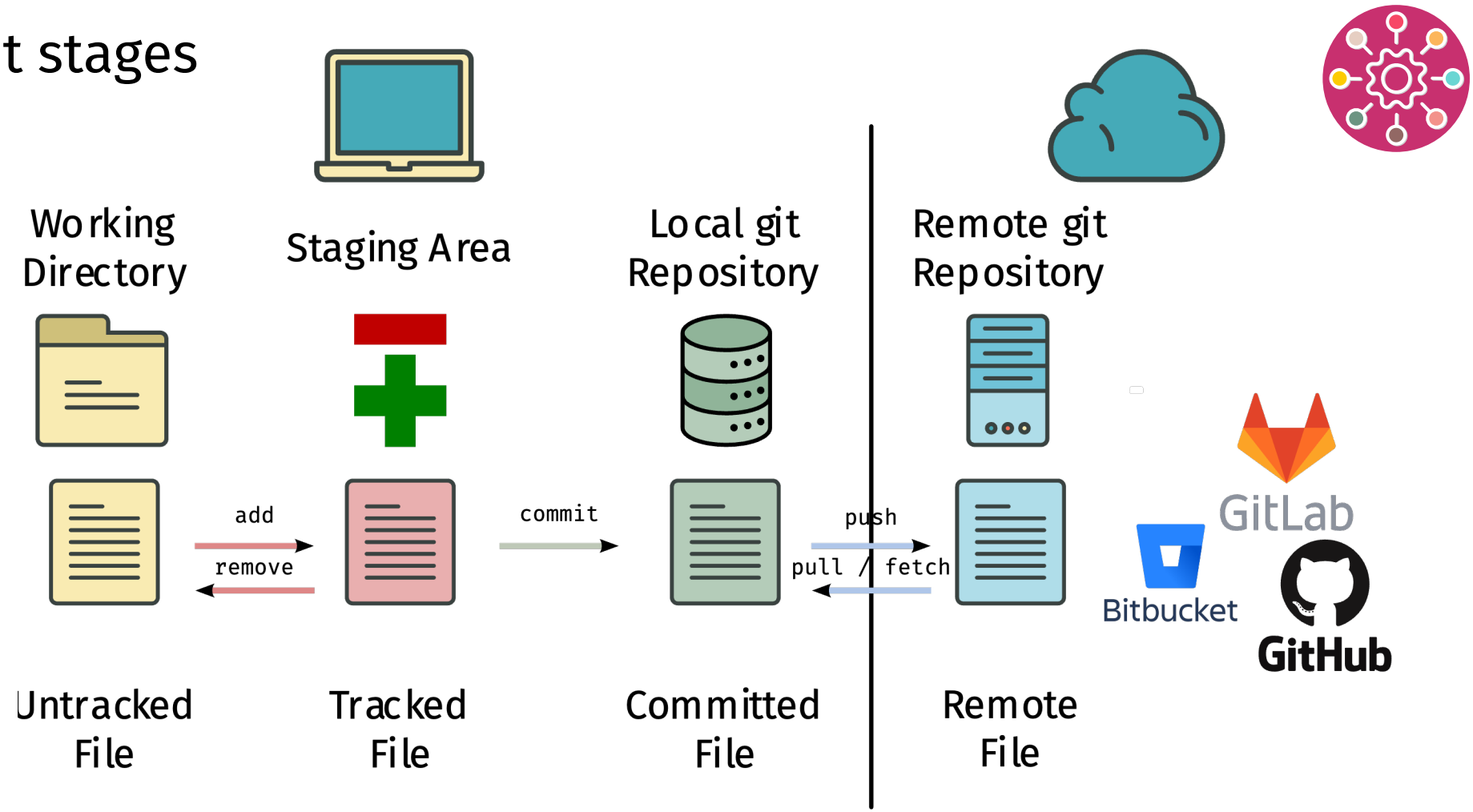
Befehl	Beschreibung
Eine working area beginnen	
`clone`	Klonen eines Repositorys in ein neues Verzeichnis
`init`	Erstellen eines leeren git-Repo oder Reinitialisieren eines vorhandenen Repo
An aktuellen Änderungen arbeiten	
`add`	Hinzufügen von Dateiinhalten zum Index
`mv`	Verschieben oder Umbenennen 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

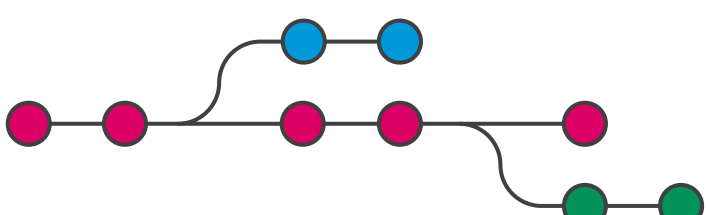
Sublime Merge



Git stages


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

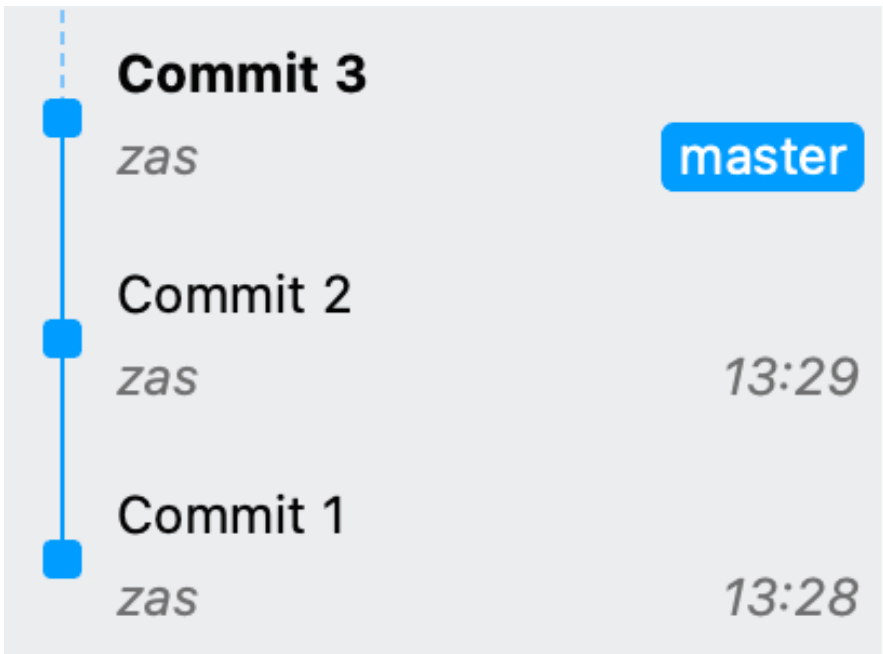
Git Branch und Merge Beispiele



Branch und Merge

Initial repo state

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

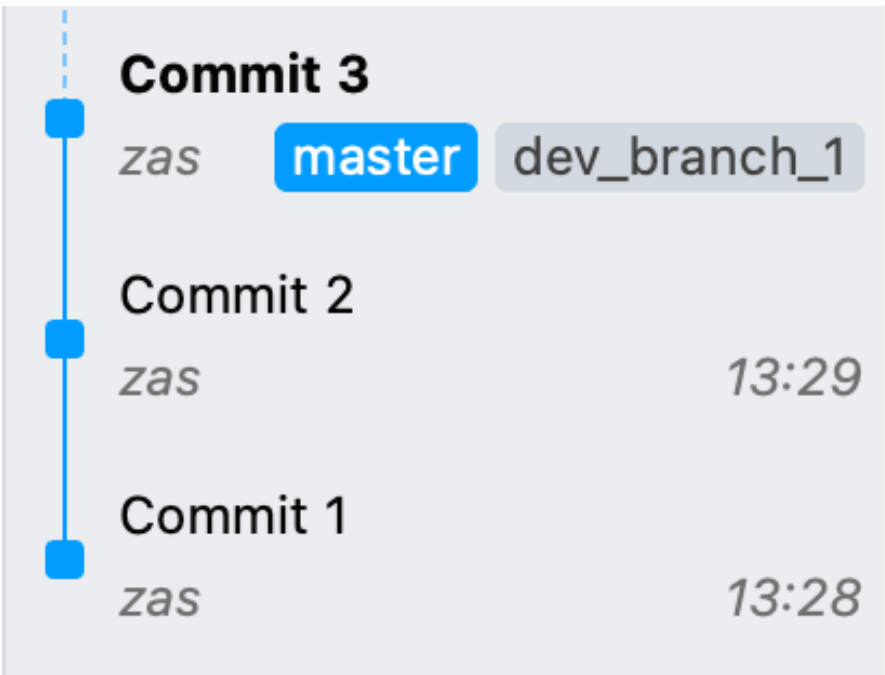


Branch und Merge

Create branch dev_branch_1

Erstelle branch dev_branch_1

```
$ git branch dev_branch_1
```



Branch und Merge

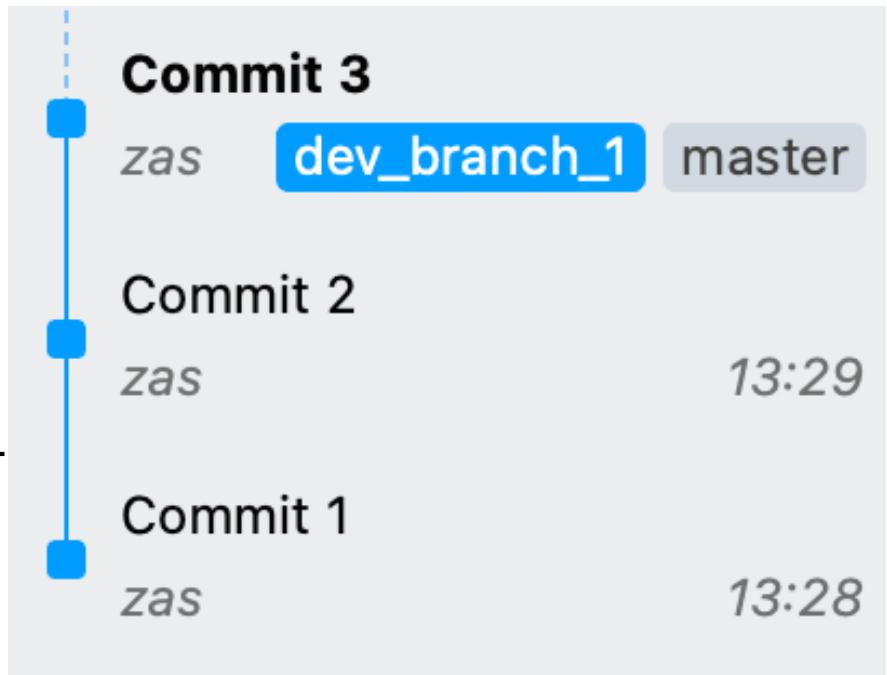
Checkout branch `dev_branch_1`

Prüfen Sie, in welchem Zweig wir uns befinden

```
$ git branch
```

Checkout branch `dev_branch_1`

```
$ git checkout dev_branch_1
```



Branch and Merge

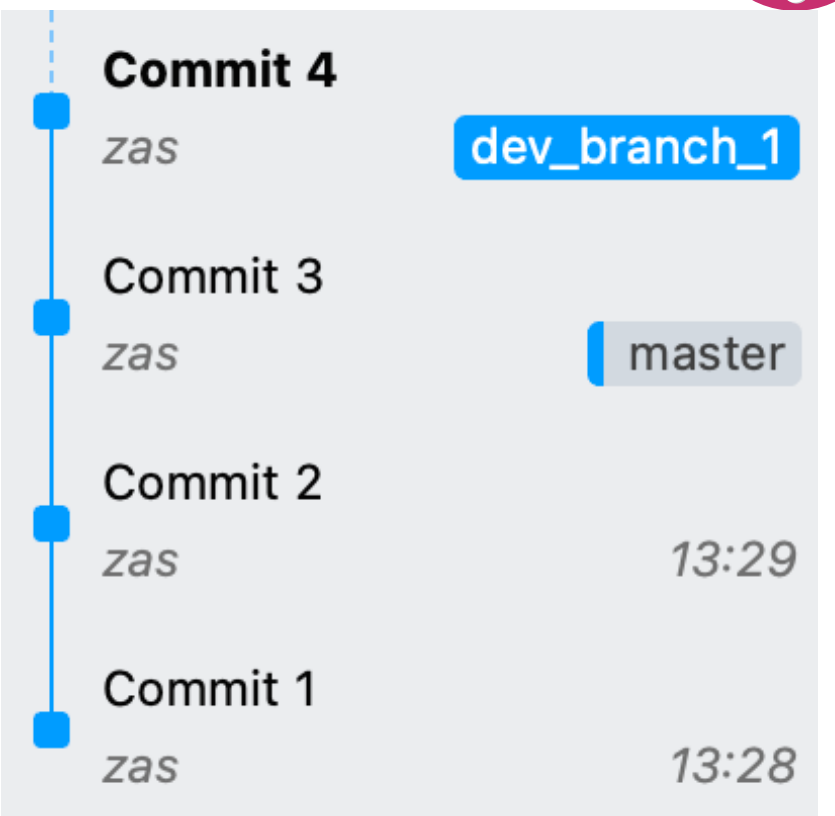
Commit on dev_branch_1

Stage new file

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

Commit stages files

```
$ git commit –m “Commit 4”
```



Branch und Merge

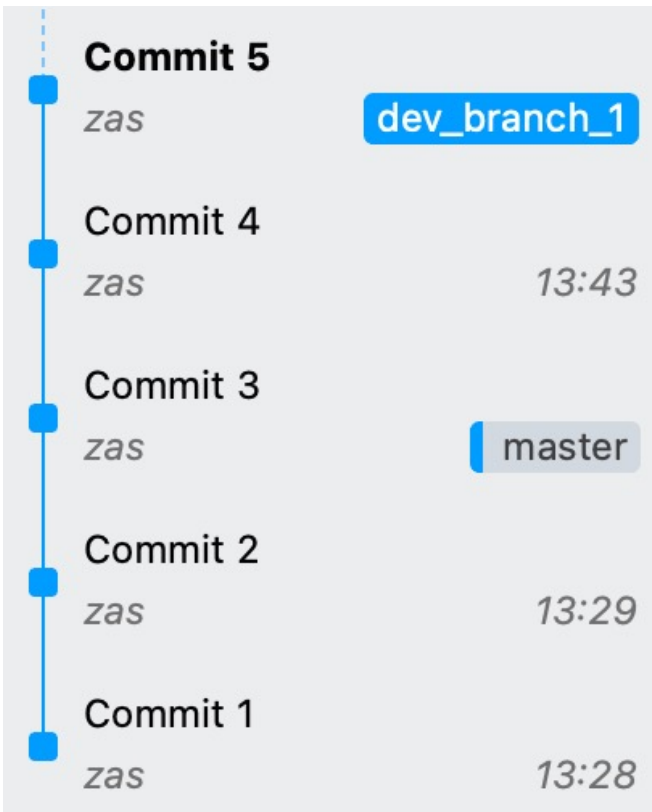
Commit on dev_branch_1

Stage new file

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

Commit stages files

```
$ git commit –m “Commit 5”
```



Branch und Merge

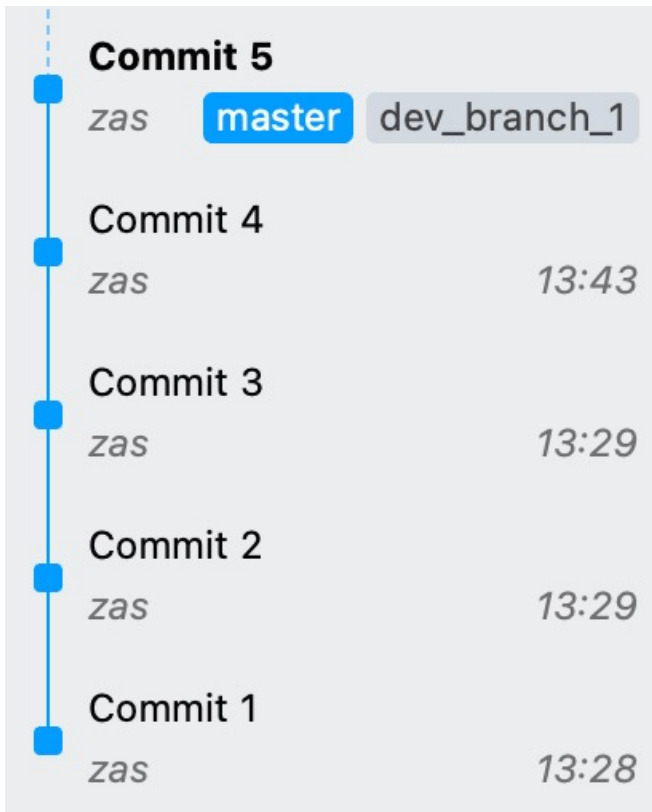
Merge `master` and `dev_branch_1`

Checkout `master` branch

```
$ git checkout master
```

Merge `dev_branch_1` into `master`

```
$ git merge dev_branch_1
```



Branch und Merge

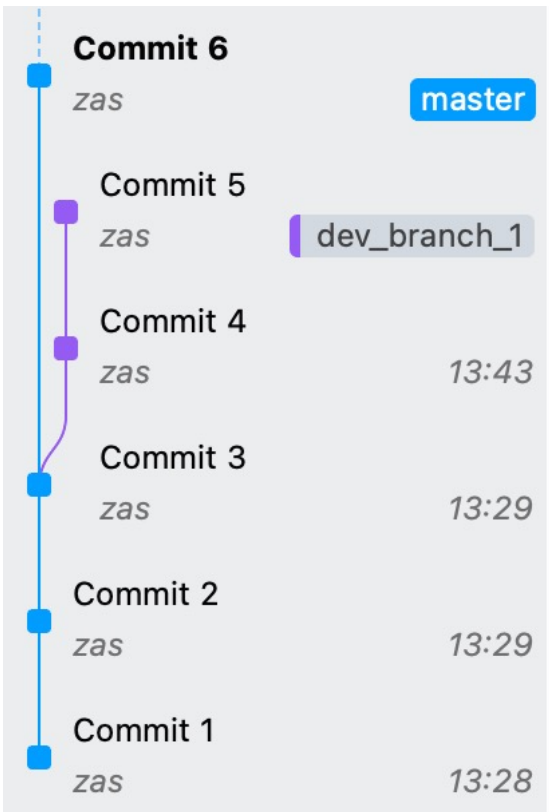
Commit on `master` branch

Stage new file

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

Commit stages files

```
$ git commit –m “Commit 6”
```

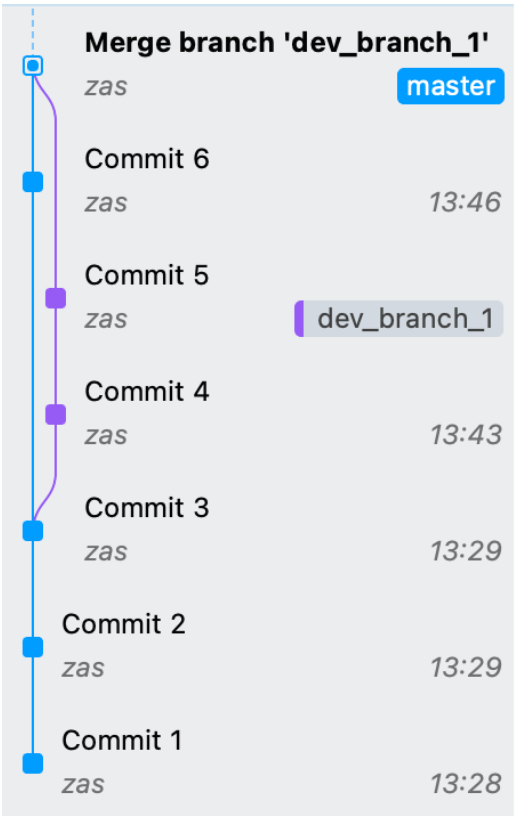


Branch und Merge

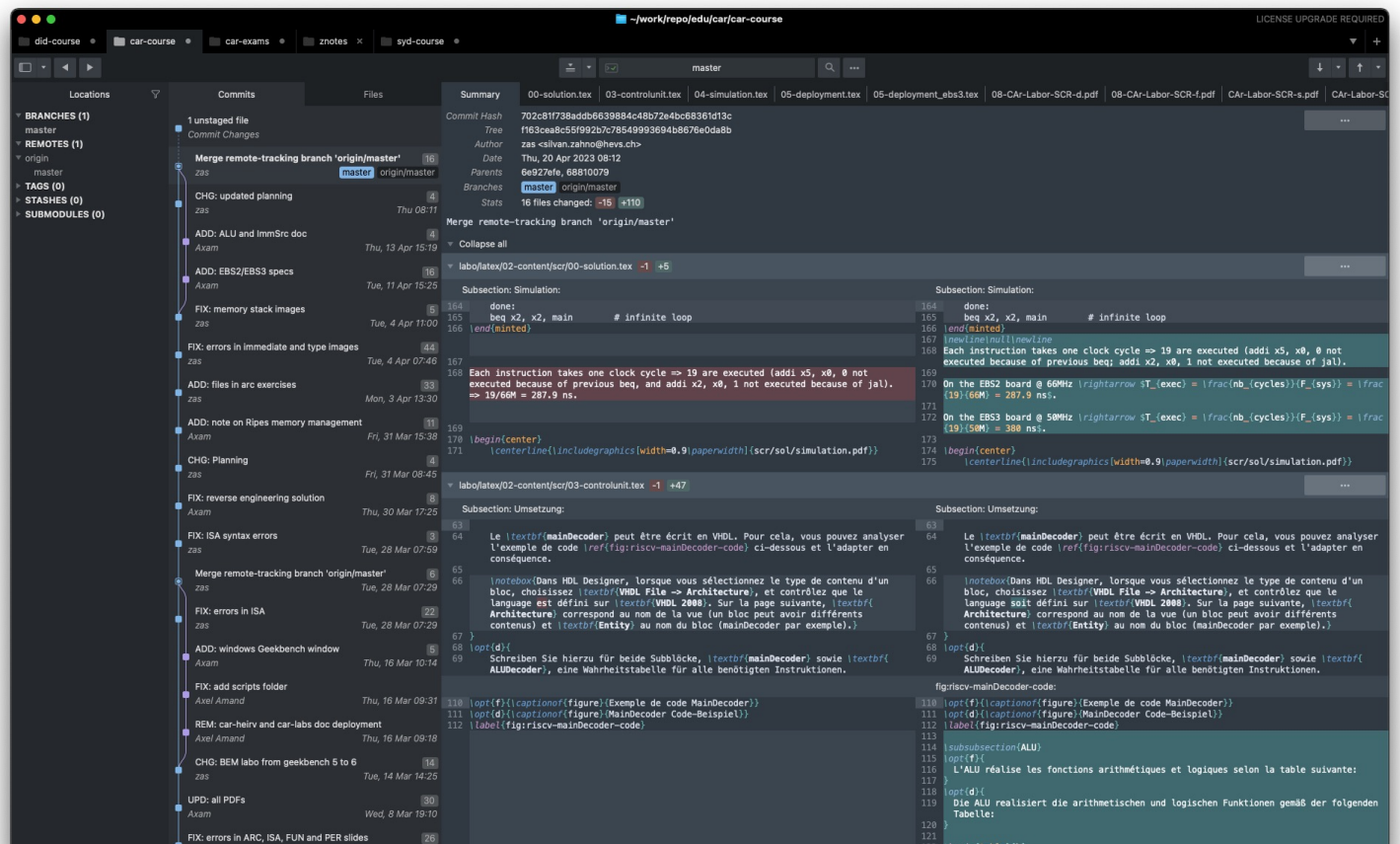
Three way merge `master` and `dev_branch_1`

Merge `dev_branch_1` into `master`

```
$ git merge dev_branch_1
```

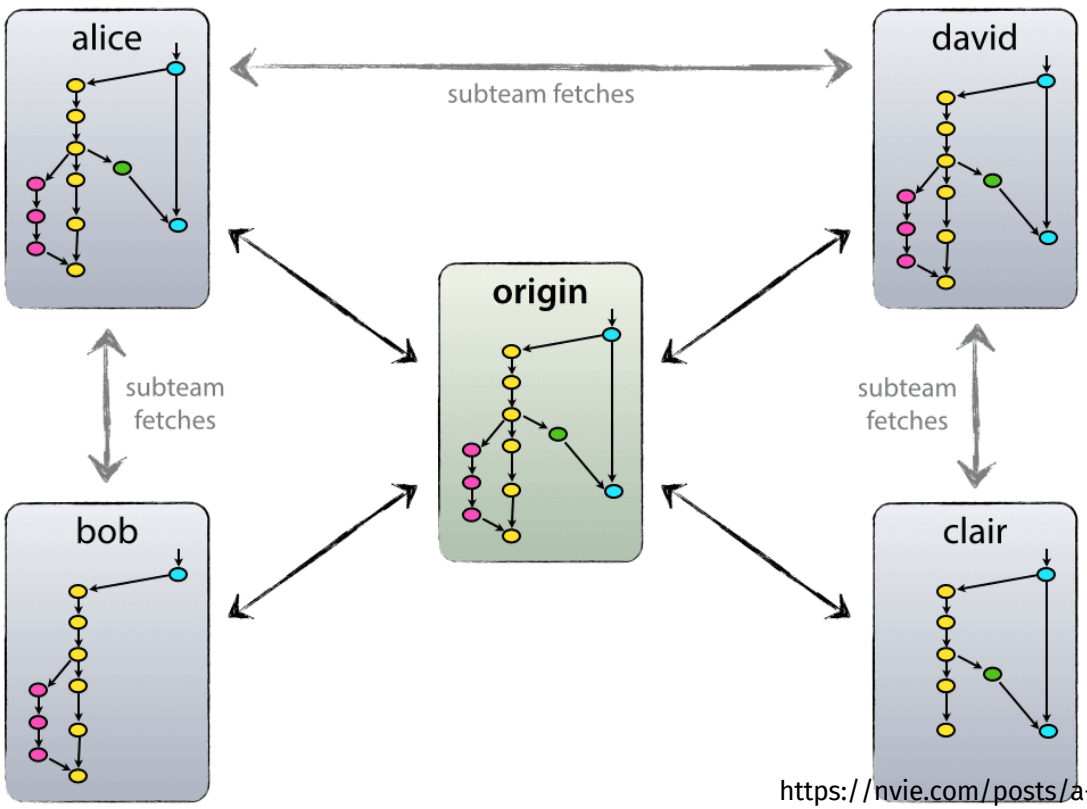


Demonstration



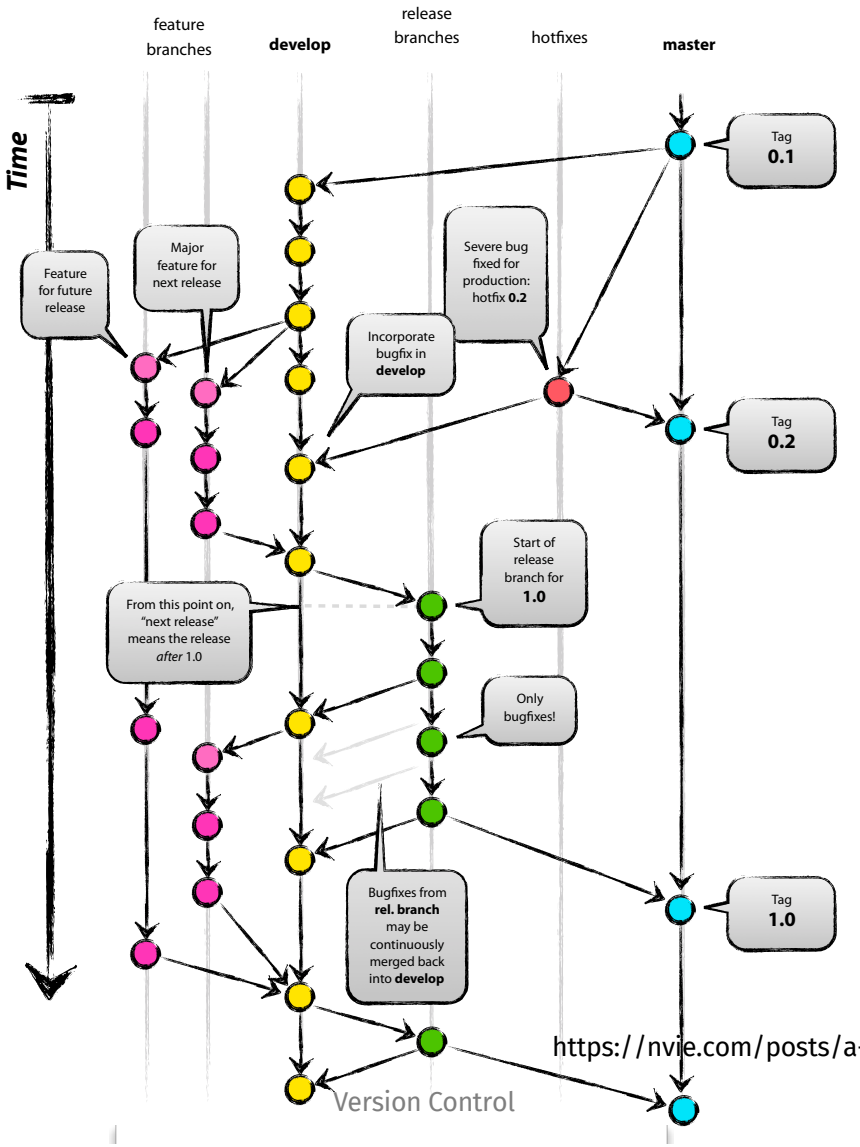
Gitflow

Gitflow Zusammenarbeit



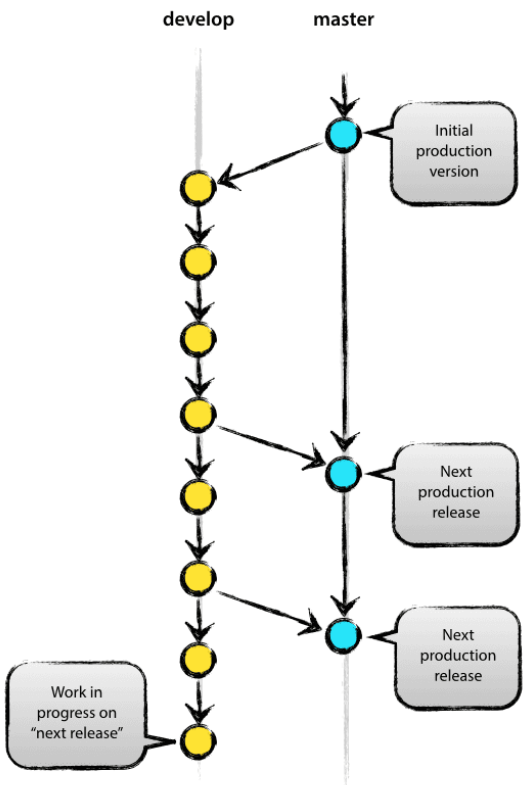
https://nvie.com/posts/a-successful-git-branching-model/

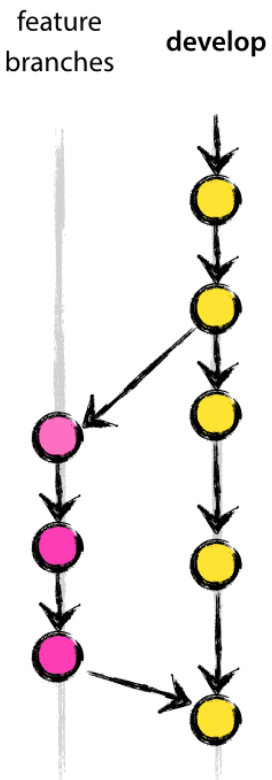
Gitflow

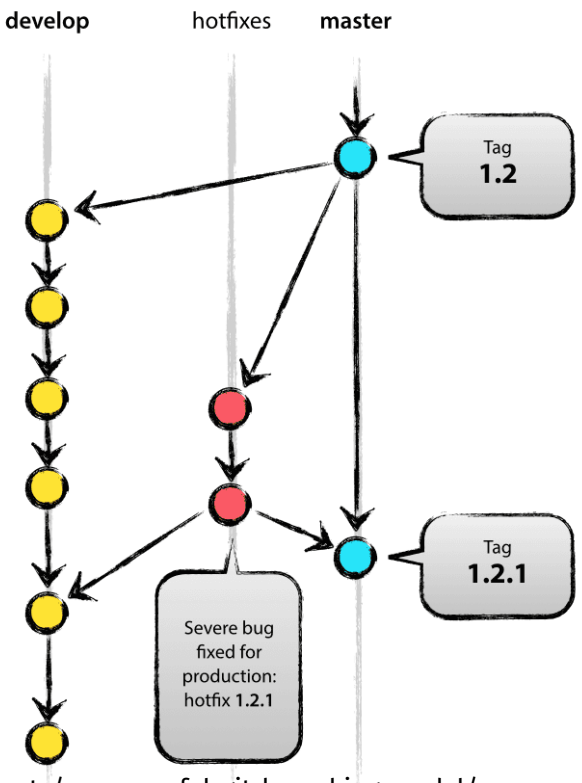


https://nvie.com/posts/a-successful-git-branching-model/

Gitflow Branching







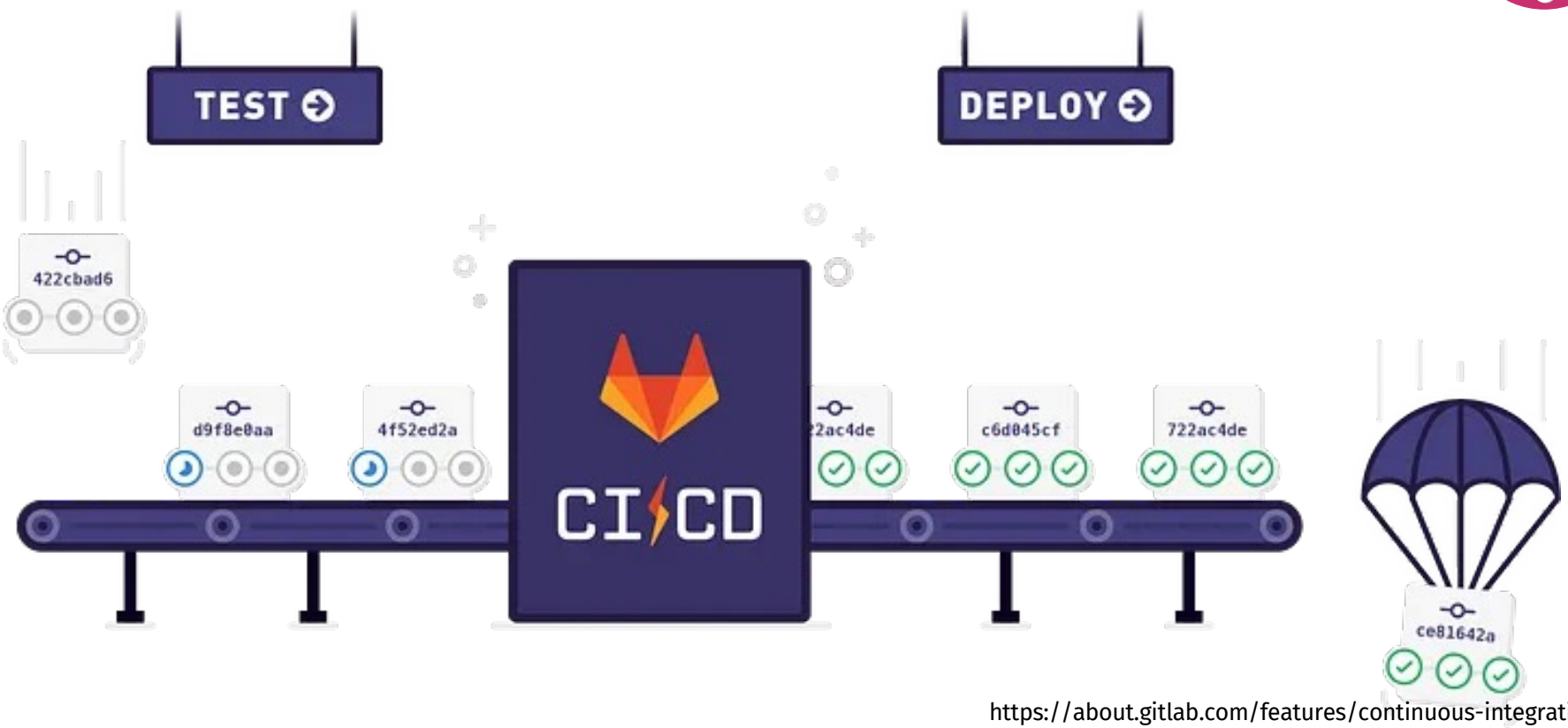
https://nvie.com/posts/a-successful-git-branching-model/

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.

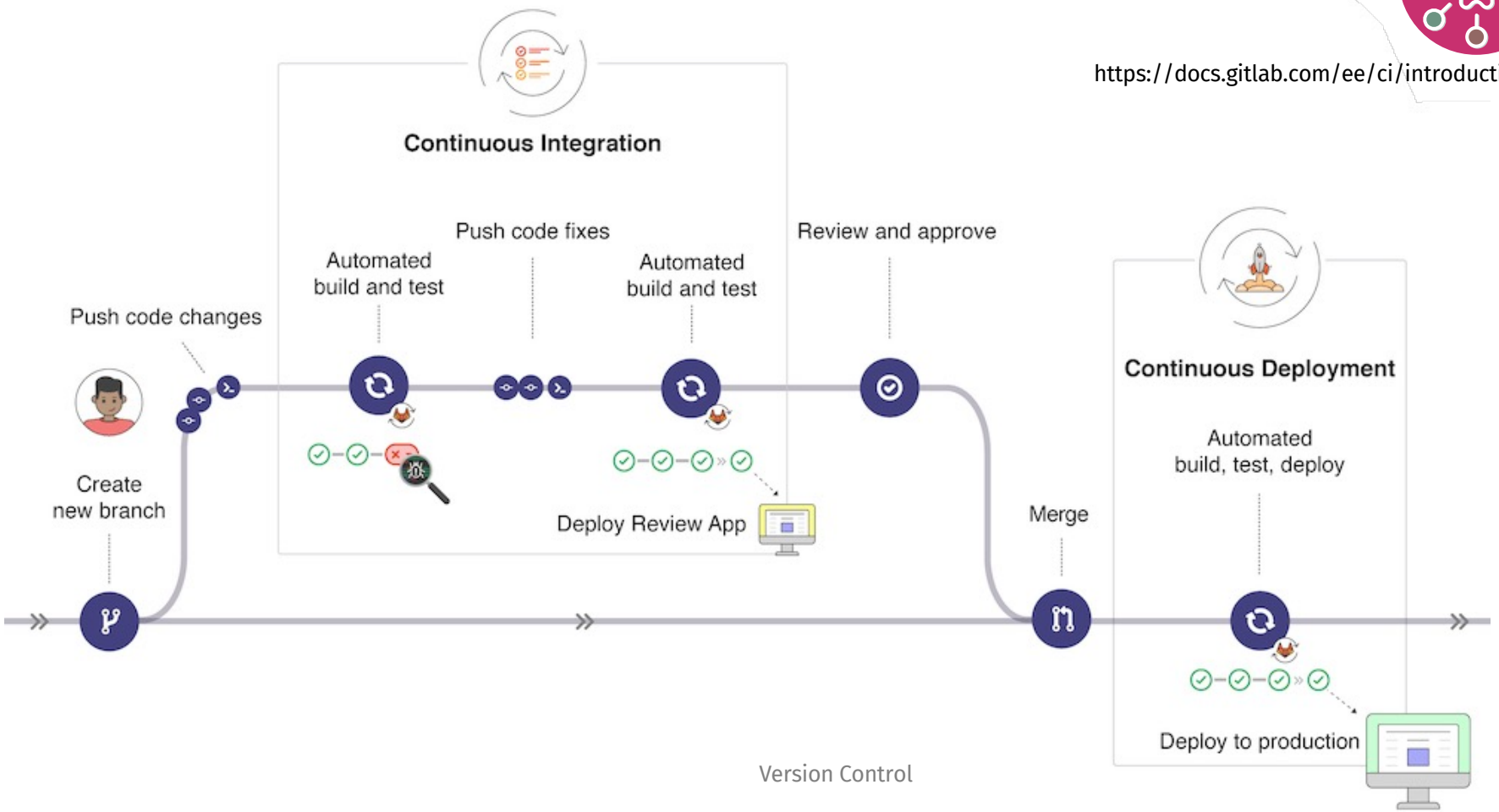
Was ist CI/CD?



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

Gitlab Workflow

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

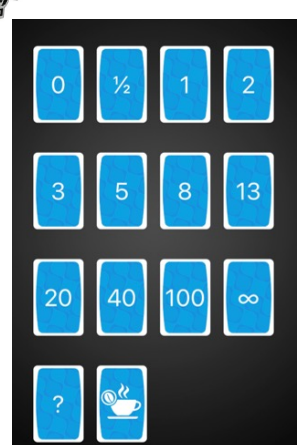


Abschätzung von Aufgaben

Abschätzung 4

Erstelle einen Star Wars Star Destroyer (4784pcs)

X Punkt(e)

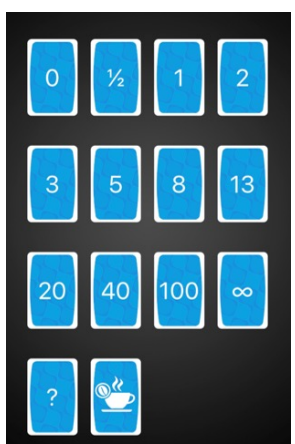


Abschätzung von Aufgaben

Abschätzung 5

Erstelle einen Star Wars Star Destroyer (37pcs)

X Punkt(e)



Introduction to git file versioning

Contents