

Abschätzung von Aufgaben

Abschätzung 2

Erstelle eine kleines Haus

X Punkt(e)



0	$\frac{1}{2}$	1	2
3	5	8	13
20	40	100	∞
?	☕		

Abschätzung von Aufgaben

Abschätzung 3

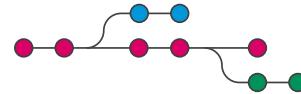
Erstelle einen
programmierbaren Rotober
X Punkt(e)



0	$\frac{1}{2}$	1	2
3	5	8	13
20	40	100	∞
?	☕		

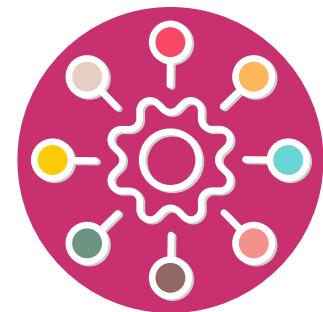


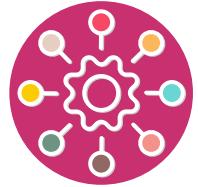
Systemdesign Version Control



Studiengang Systemtechnik

Silvan Zahno silvan.zahno@hevs.ch



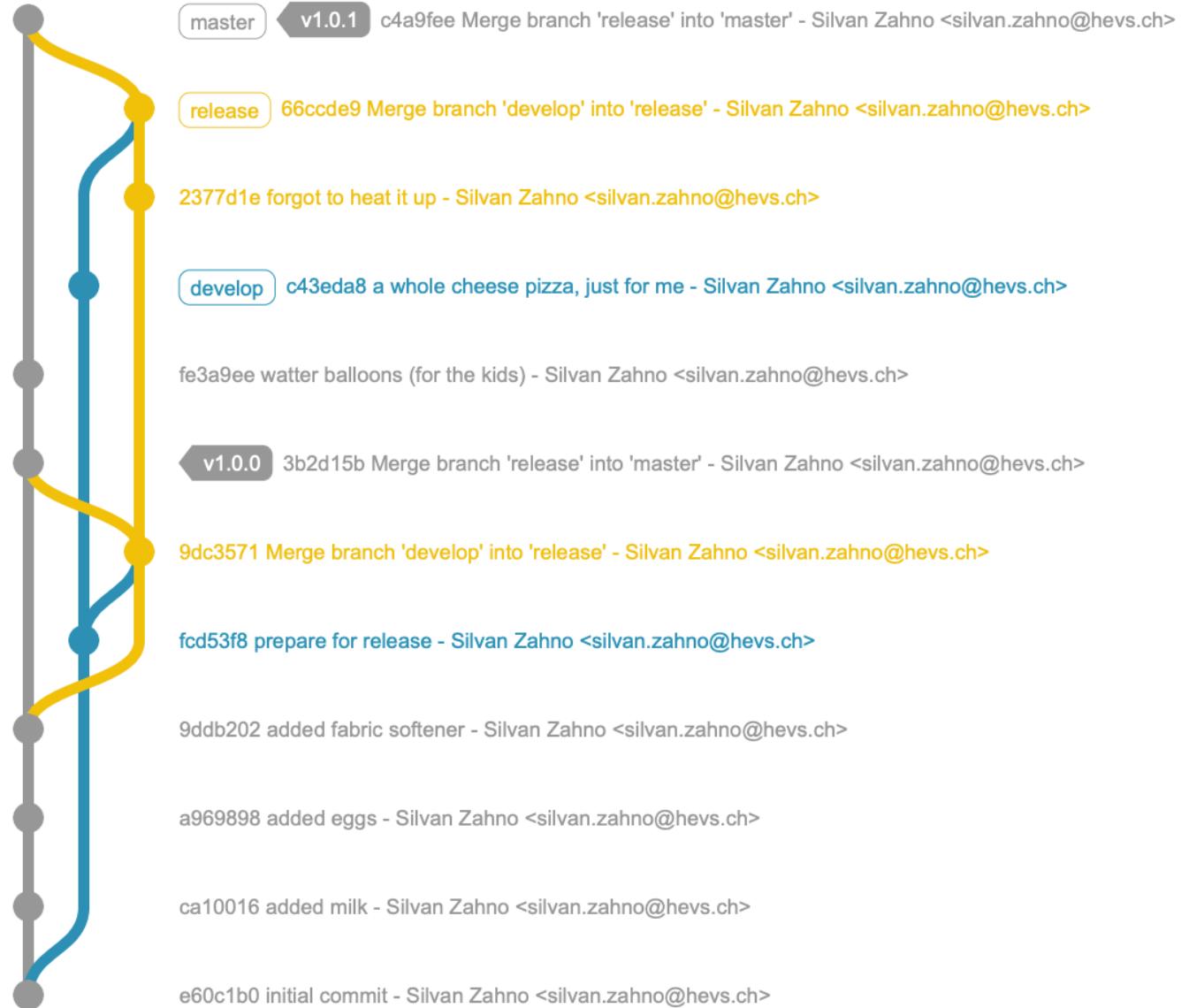
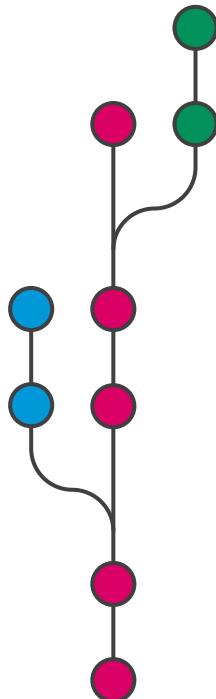


Ihr derzeitiges System

- - └─ 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



Possible Hilfsmittel



Git (git)



Subversion (svn)



Mercurial (hg)

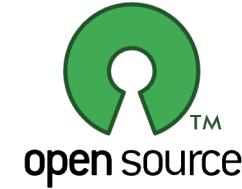
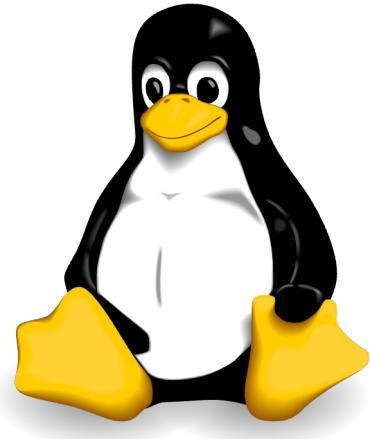


mercurial

SyD Version Control

Linux Torvalds

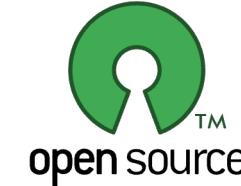
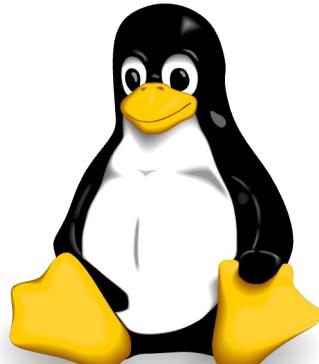
Linux (1991)



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>

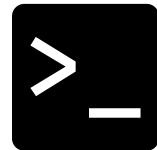
Bitbucket

<https://bitbucket.com>

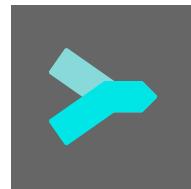


Git Hilfsmittel

Kommandozeile



Sublime Merge



Git Cola



Git Kraken



Fork



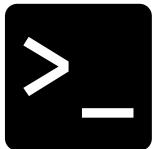
Tower



SmartGit



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
```

```
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

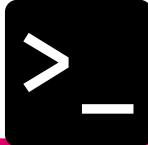
grow, mark and tweak your common history
branch  List, create, or delete branches
commit  Record changes to the repository
merge   Join two or more development histories together
rebase  Reapply commits on top of another base tip
reset   Reset current HEAD to the specified state
switch  Switch branches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

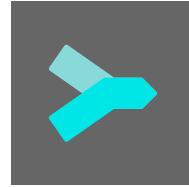
[zas@zac ~] (base)
$
```

git Befehle



Befehl	Beschreibung	Befehl	Beschreibung
Eine working area beginnen		Wachsen, markieren und optimieren Sie Ihre gemeinsame Historie	
clone	Klonen eines Repositorys in ein neues Verzeichnis	branch	Zweige auflisten, erstellen oder löschen Auschecken
init	Erstellen eines leeren git-Repo oder Reinitialisieren eines vorhandenen Repo	checkout	Zweige wechseln oder Arbeitsbaumdateien wiederherstellen
An aktuellen Änderungen arbeiten		commit	Änderungen an der Projektliste aufzeichnen
add	Hinzufügen von Dateiinhalten zum Index	diff	Änderungen zwischen Commits, Commit und Arbeitsbaum anzeigen, etc.
mv	Verschieben oder Umbenennen einer Datei, eines Verzeichnisses oder eines Symlinks	merge	Zwei oder mehr Entwicklungsgeschichten zusammenführen
reset	Zurücksetzen des aktuellen HEAD auf den angegebenen Zustand	rebase	Commits auf einen anderen Basistipp anwenden tag
rm	Entfernen von Dateien aus dem Arbeitsbaum und aus dem Index	tag	Ein Tag-Objekt erstellen, auflisten, löschen oder verifizieren
Untersuchen der Historie und des Status		Zusammenarbeiten	
log	Übergabeprotokolle anzeigen	fetch	Herunterladen von Objekten und Referenzen aus einem anderen Repo
show	Verschiedene Arten von Objekten anzeigen	pull	Abrufen und Integrieren aus einem anderen Projektarchiv oder einem lokalen Zweig schieben
status	Show the working tree status	push	Aktualisieren entfernter Referenzen zusammen mit zugehörigen Objekten

Sublime Merge



Sublime Merge interface showing a git repository and code editor.

Locations:

- BRANCHES (1)
 - master
- REMOTES (1)
 - origin
 - master
- TAGS (0)
- STASHES (0)
- SUBMODULES (0)

Commits:

- 1 untagged file Commit Changes
- Merge remote-tracking branch 'origin/master' (16 commits)
 - CHG: updated planning (4 commits, Thu 08:11)
 - ADD: ALU and ImmSrc doc (4 commits, Thu, 13 Apr 15:19)
 - ADD: EBS2/EBS3 specs (4 commits, Tue, 11 Apr 15:25)
 - FIX: memory stack images (5 commits, Tue, 4 April 11:00)
 - FIX: errors in immediate and type images (44 commits, Tue, 4 April 07:46)
 - ADD: files in arc exercises (33 commits, Mon, 3 April 13:30)
 - ADD: note on Ripes memory management (11 commits, Fri, 31 Mar 15:38)
 - CHG: Planning (4 commits, Fri, 31 Mar 08:45)
 - FIX: reverse engineering solution (8 commits, Thu, 30 Mar 17:25)
 - FIX: ISA syntax errors (3 commits, Tue, 28 Mar 07:59)
 - Merge remote-tracking branch 'origin/master' (16 commits, Thu, 28 Mar 07:59)
 - FIX: errors in ISA (22 commits, Tue, 28 Mar 07:29)
 - ADD: windows Geekbench window (5 commits, Thu, 16 Mar 10:14)
 - FIX: add scripts folder (Axel Amand, Thu, 16 Mar 09:31)
 - REM: car-herrv and car-labs doc deployment (Axel Amand, Thu, 16 Mar 09:18)
 - CHG: BEM labo from geekbench 5 to 6 (14 commits, Tue, 14 Mar 14:25)
 - UPD: all PDFs (Axam, Wed, 8 Mar 19:10)
 - FIX: errors in ARC, ISA, FUN and PER slides (26 commits, Tue, 7 Mar 09:09)

Summary:

- Commit Hash: 702c8f1738addb639884c48b724ab68361d3c
Tree: f163ce5b55f992b7c78549993694b867860ddab8
- Author: zas <silvan.zahn@hevs.ch>
- Date: Thu, 20 April 2023 08:12
- Parents: 6e927fe, 68810079
- Branches: master origin/master
- Stats: 16 files changed: -15 +110

Files:

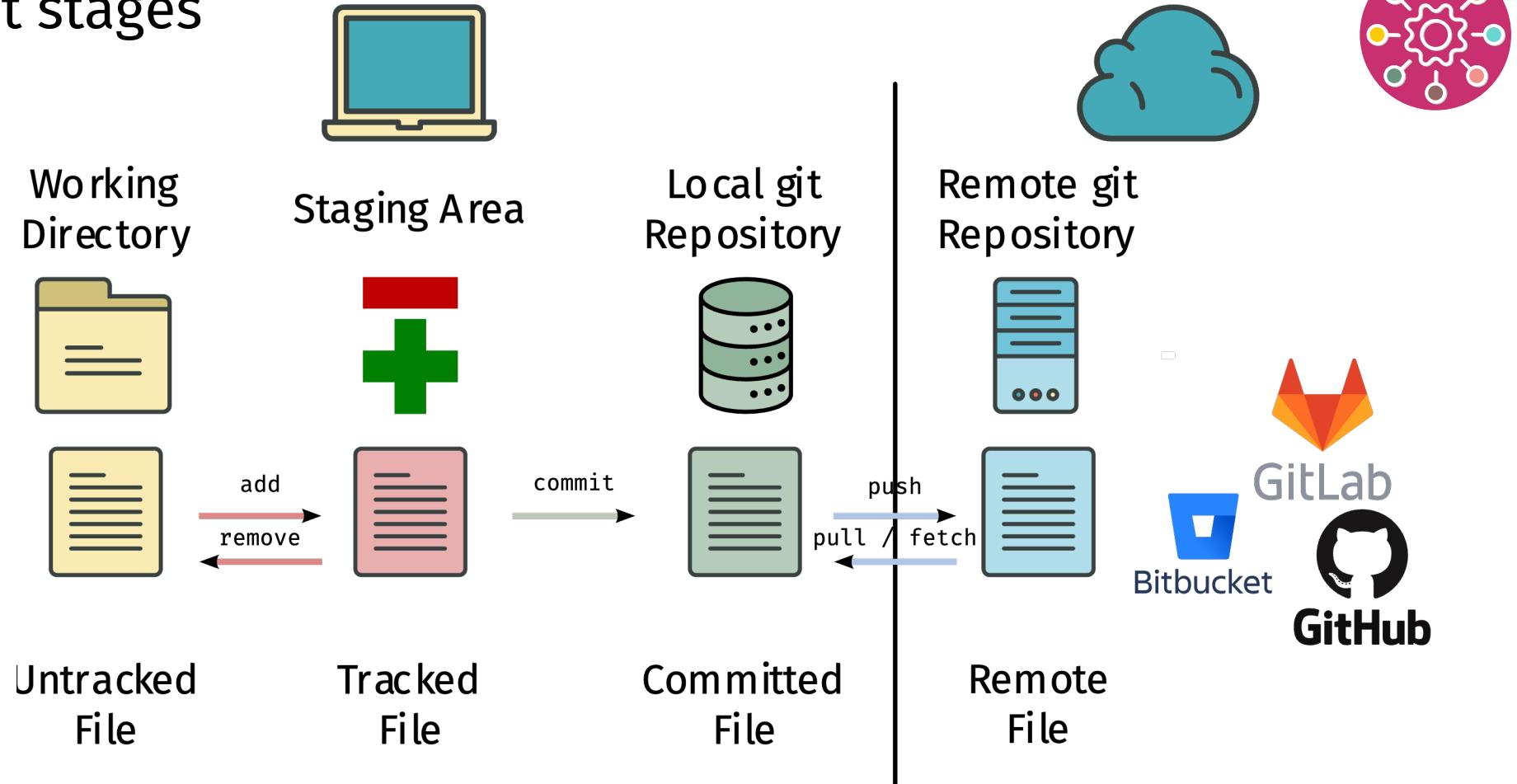
- 00-solution.tex
- 03-controlunit.tex
- 04-simulation.tex
- 05-deployment.tex
- 05-deployment_ebs3.tex
- 08-CAR-Labor-SCR-d.pdf
- 08-CAR-Labor-SCR-f.pdf
- CAR-Labor-SCR-s.pdf
- CAR-Labor-SCR-t.pdf

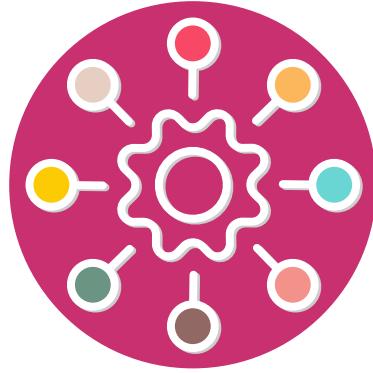
Code Editor:

```
00-solution.tex
03-controlunit.tex
04-simulation.tex
05-deployment.tex
05-deployment_ebs3.tex
08-CAR-Labor-SCR-d.pdf
08-CAR-Labor-SCR-f.pdf
CAR-Labor-SCR-s.pdf
CAR-Labor-SCR-t.pdf
```

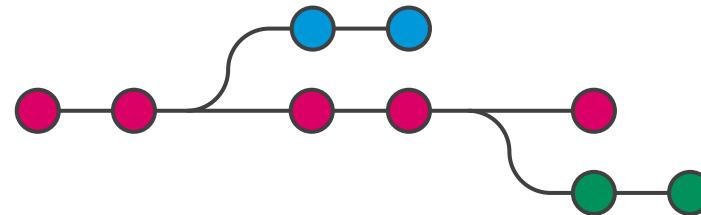
LICENSE UPGRADE REQUIRED

Git stages





Git Branch und Merge Beispiele

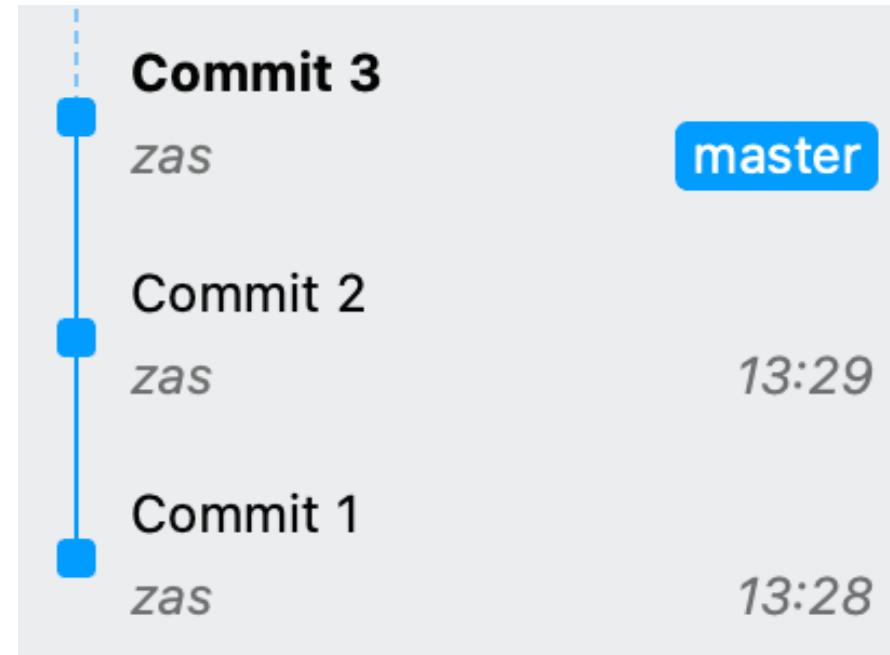




Branch und Merge

Initial repo state

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

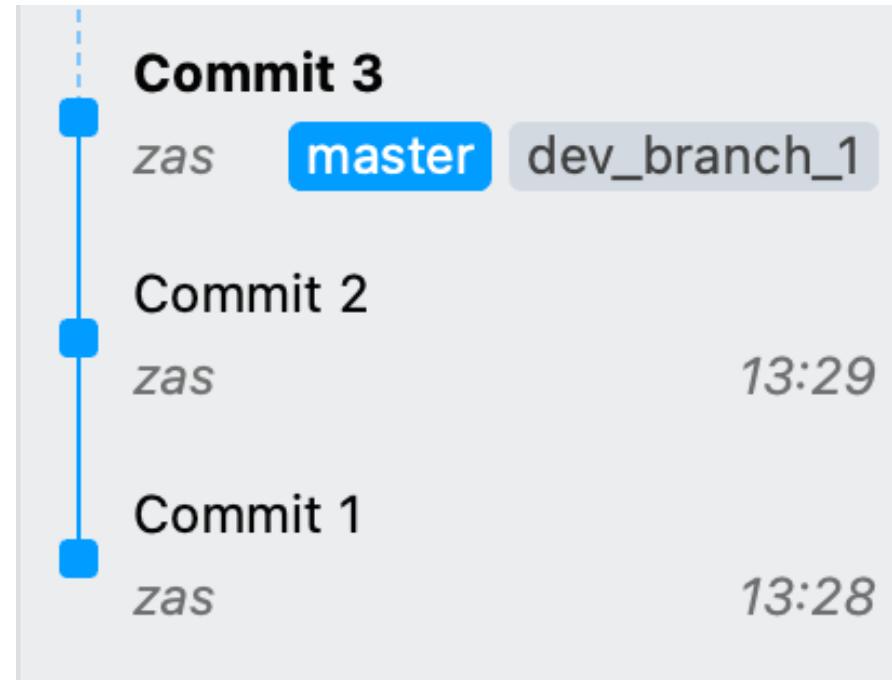




Branch und Merge

Create branch dev_branch_1

Erstelle branch dev_branch_1
\$ git branch dev_branch_1





Branch und Merge

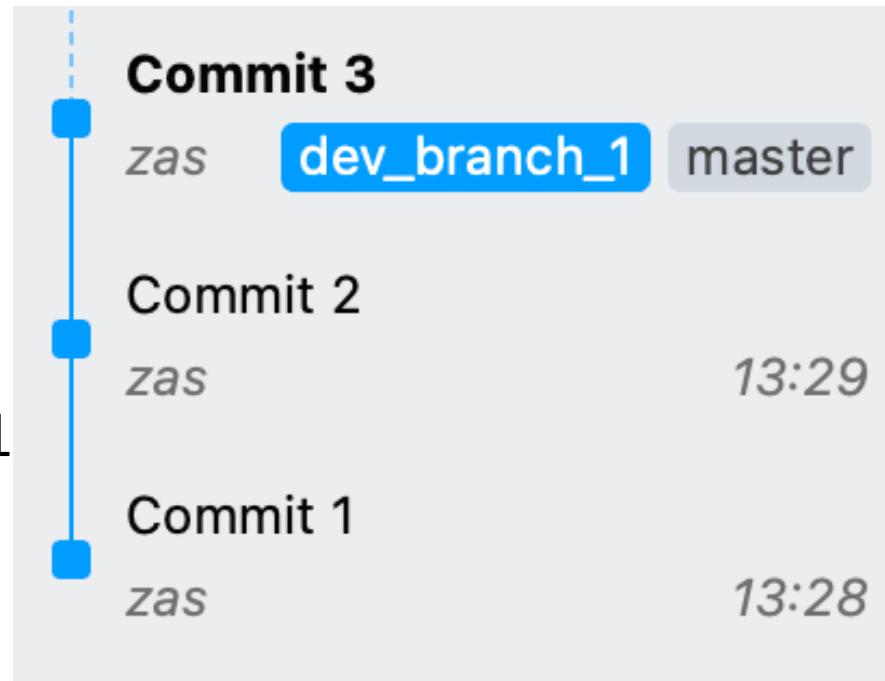
Checkout branch dev_branch_1

Prüfen Sie, in welchem Zweig wir uns befinden

```
$ git branch
```

Checkout branch dev_branch_1

```
$ git checkout dev_branch_1
```





Branch and Merge

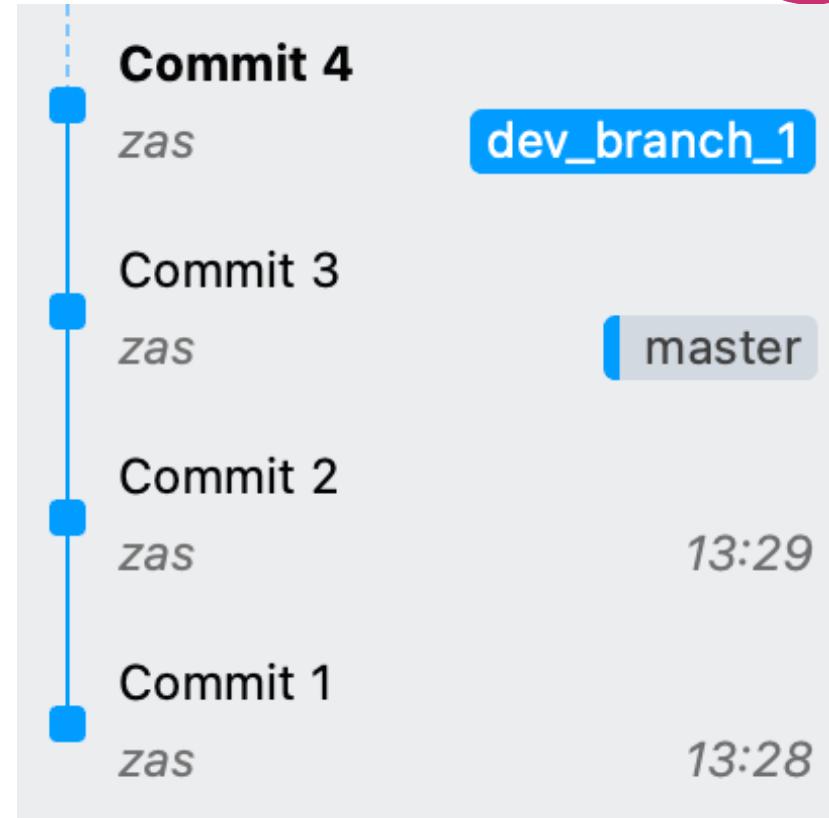
Commit on dev_branch_1

Stage new file

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

Commit stages files

```
$ git commit -m "Commit 4"
```





Branch und Merge

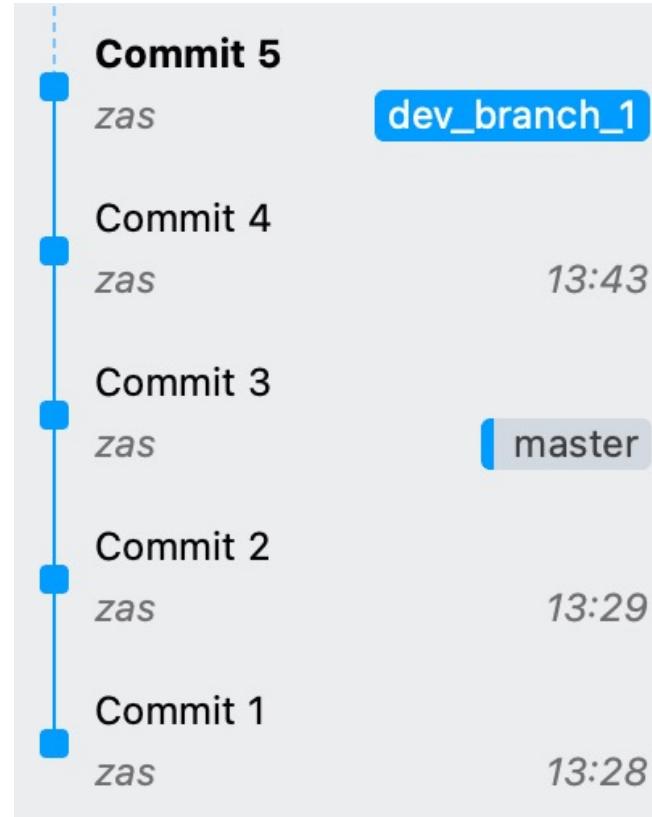
Commit on dev_branch_1

Stage new file

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

Commit stages files

```
$ git commit -m "Commit 5"
```





Branch und Merge

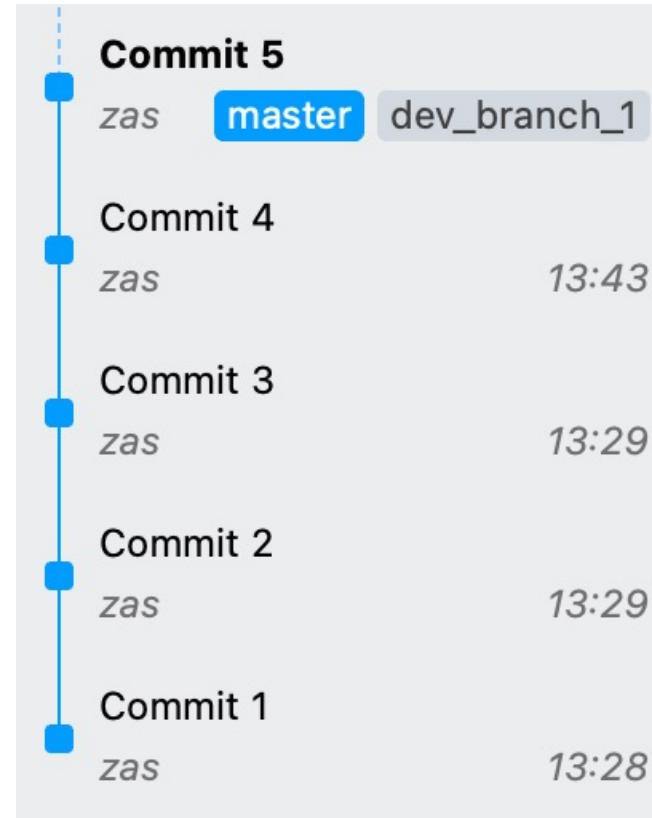
Merge master and dev_branch_1

Checkout master branch

```
$ git checkout master
```

Merge dev_branch_1 into master

```
$ git merge dev_branch_1
```





Branch und Merge

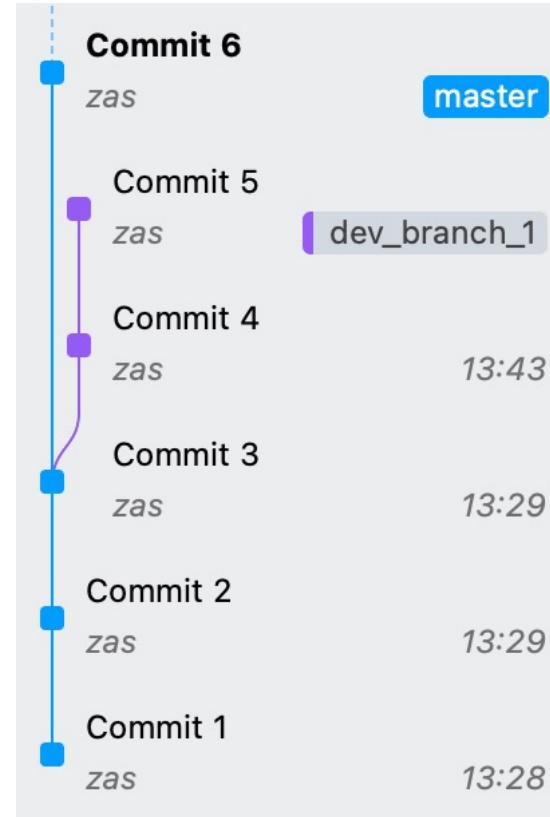
Commit on master branch

Stage new file

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

Commit stages files

```
$ git commit -m "Commit 6"
```



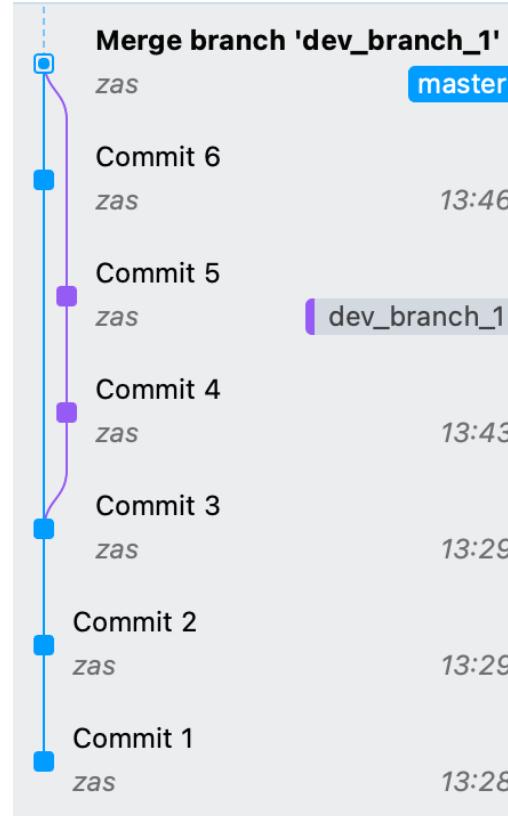
Branch und Merge

Three way merge master and dev_branch_1

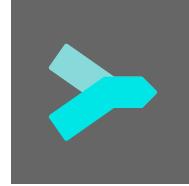


Merge dev_branch_1 into master

```
$ git merge dev_branch_1
```



Demonstration



The screenshot shows a GitHub pull request interface. The top navigation bar indicates the repository is 'edu/car-course' and the branch is 'master'. The pull request summary shows 16 commits, 16 files changed, and 16 additions/10 deletions. The commits are listed in chronological order, with details like author, date, and commit message. The right pane displays the code changes for each commit, with annotations explaining the purpose of specific lines of code.

Commits

- 1 unstaged file
- Commit Changes
- Merge remote-tracking branch 'origin/master'
- CHG: updated planning
- ADD: ALU and ImmSrc doc
- ADD: EBS2/EBS3 specs
- FIX: memory stack images
- FIX: errors in immediate and type images
- ADD: files in arc exercises
- ADD: note on Ripes memory management
- CHG: Planning
- FIX: reverse engineering solution
- FIX: ISA syntax errors
- Merge remote-tracking branch 'origin/master'
- FIX: errors in ISA
- ADD: windows Geekbench window
- FIX: add scripts folder
- REM: car-heirv and car-labs doc deployment
- CHG: BEM labo from geekbench 5 to 6
- UPD: all PDFs
- FIX: errors in ARC, ISA, FUN and PER slides

Files

Summary

Commit Hash: 702c81738addb66398b4c48b7e4acbf3631d13c
Tree: f163ceaa855f992b7c78549993694b8676e0de8b
Author: zas cslvahn.zahn@evs.ch
Date: Thu, 20 Apr 2023 08:12
Parents: 6e927fe, 68810079
Branches: master, origin/master
Stats: 16 files changed: -15 +10

Merge remote-tracking branch 'origin/master'

Subsection: Simulation:

164 done:
165 beq x2, x2, main # infinite loop
166 lendiftined

167 \begin{centerline}

168 Each instruction takes one clock cycle => 19 are executed (addi x5, x0, 0 not executed because of previous beq, and addi x2, x0, 1 not executed because of jal). => 19/664 = 287.9 ns.

169 \begin{center}

170 \centerline{\includegraphics[width=0.9\paperwidth]{scr/sol/simulation.pdf}}

171 \end{center}

172 On the EBS2 board @ 60MHz (\rightarrow sT_(exec) = (frac{nb_cycles}{F_sys}) = (frac{19}{664}) = 287.9 ns).

173

174 \begin{center}

175 \centerline{\includegraphics[width=0.9\paperwidth]{scr/sol/simulation.pdf}}

176

Subsection: Umsetzung:

63 Le \textit{txtrbf{mainDecoder}} peut être écrit en VHDL. Pour cela, vous pouvez analyser l'exemple de code \ref{fig:riscv-mainDecoder-code} ci-dessous et l'adapter en conséquence.

64 Le \textit{txtrbf{mainDecoder}} peut être écrit en VHDL. Pour cela, vous pouvez analyser l'exemple de code \ref{fig:riscv-mainDecoder-code} ci-dessous et l'adapter en conséquence.

65 \textit{txtrbf{Dsp3 HDL Designer}}, lorsque vous sélectionnez le type de contenu d'un bloc, choisissez \textit{VHDL File -> Architecture}, et contrôlez que le langage est défini sur \textit{txtrbf{VHDL 2008}}. Sur la page suivante, \textit{txtrbf{Architecture}} correspond au nom de la vue (un bloc peut avoir différents contenus) et \textit{txtrbf{Entity}} au nom du bloc (mainDecoder par exemple).

66 \textit{txtrbf{Dsp3 HDL Designer}}, lorsque vous sélectionnez le type de contenu d'un bloc, choisissez \textit{txtrbf{VHDL File -> Architecture}}, et contrôlez que le langage est défini sur \textit{txtrbf{VHDL 2008}}. Sur la page suivante, \textit{txtrbf{Architecture}} correspond au nom de la vue (un bloc peut avoir différents contenus) et \textit{txtrbf{Entity}} au nom du bloc (mainDecoder par exemple).

67 }

68 \opt{d}{

69 Schreiben Sie hierzu für beide Subblöcke, \textit{txtrbf{mainDecoder}} sowie \textit{txtrbf{ALUDecoder}}, eine Wahrheitstabelle für alle benötigten Instruktionen.

70 \textit{txtrbf{mainDecoder-code}}:

110 \opt{f}{\captionof{figure}{Exemple de code MainDecoder}}
111 \opt{d}{\captionof{figure}{MainDecoder Code-Beispiel}}
112 \label{fig:riscv-mainDecoder-code}

113

114 \subsubsection{ALU}

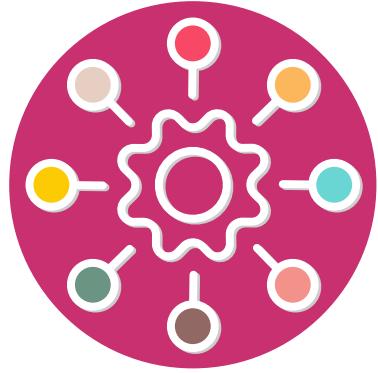
115 \opt{f}{

116 L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:

117 }
118 \opt{d}{
119 Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden Tabelle:

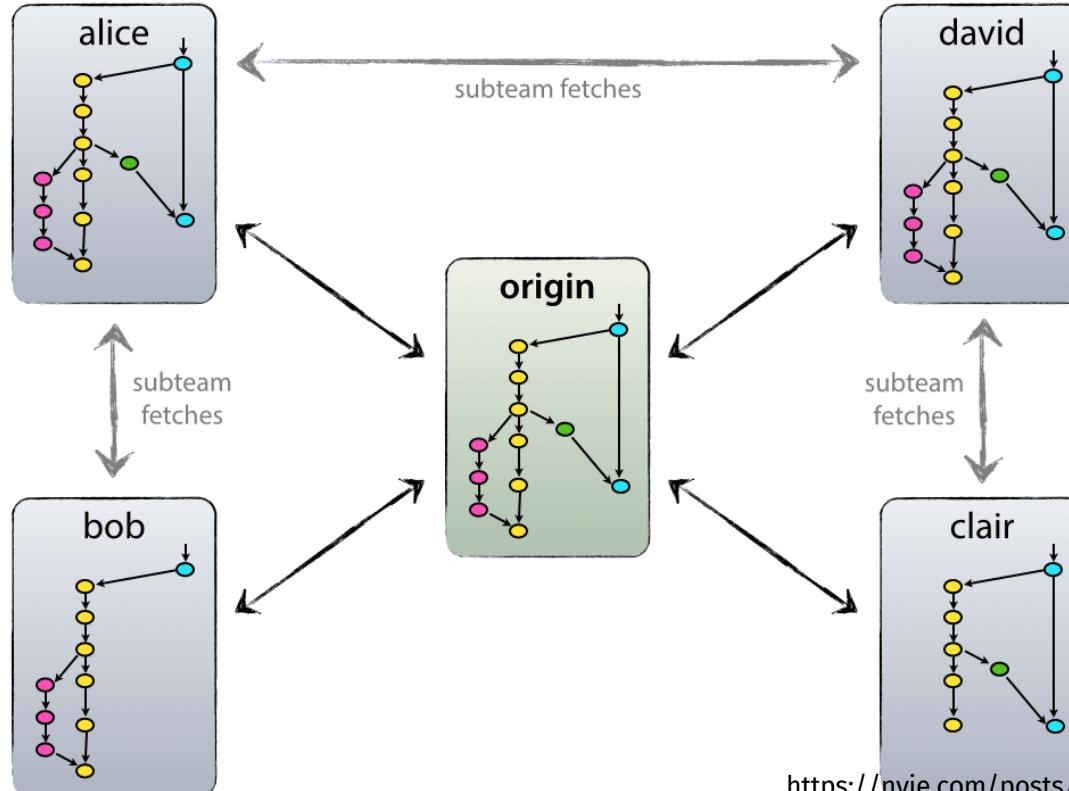
120 }
121

122 \begin{table}[h]



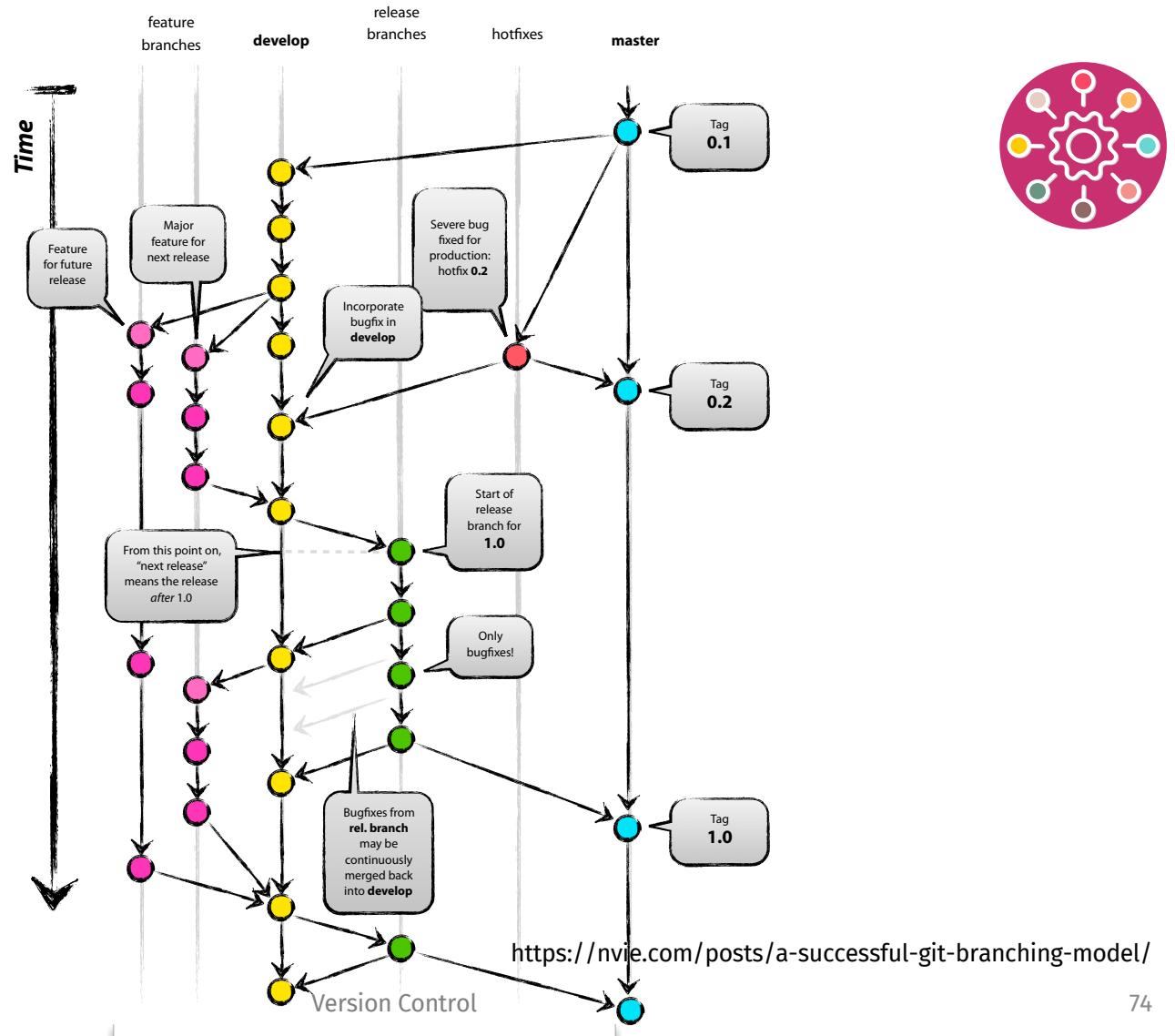
Gitflow

Gitflow Zusammenarbeit

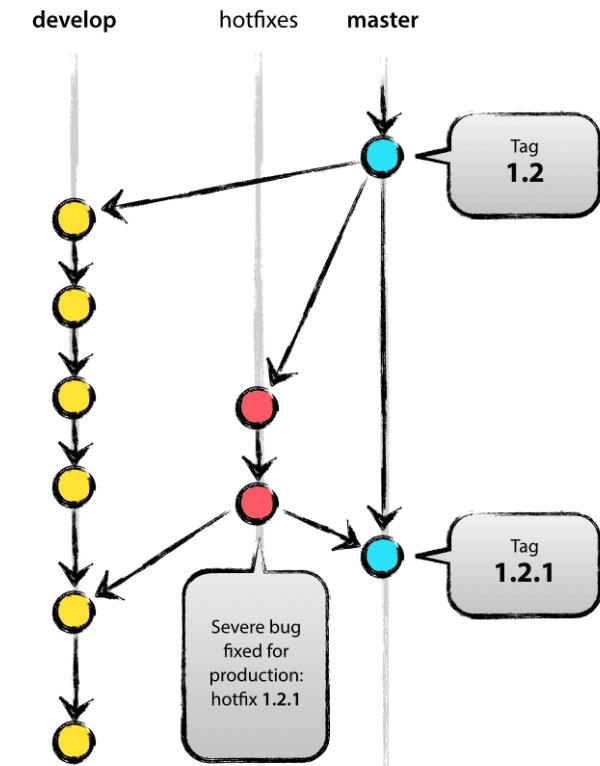
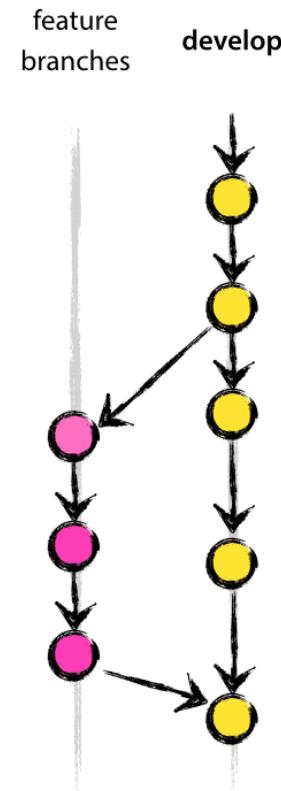
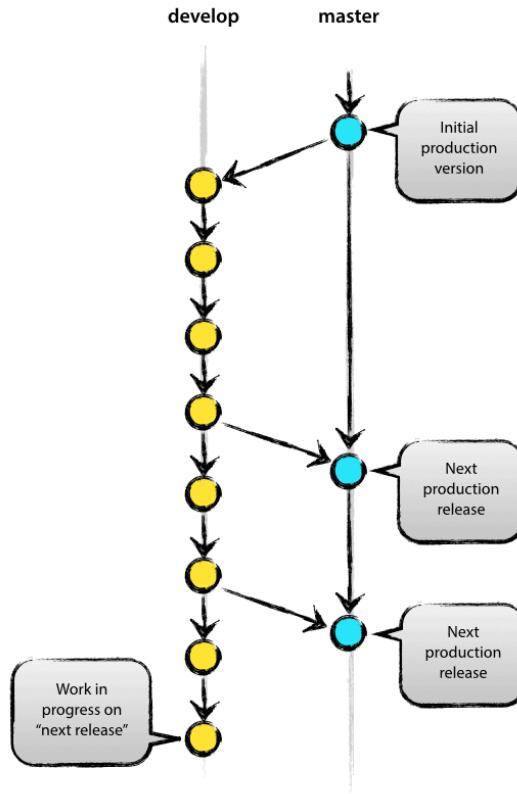


<https://nvie.com/posts/a-successful-git-branching-model/>

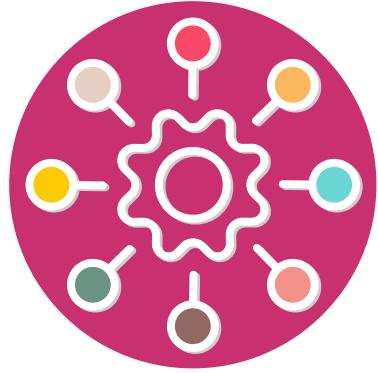
Gitflow



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?

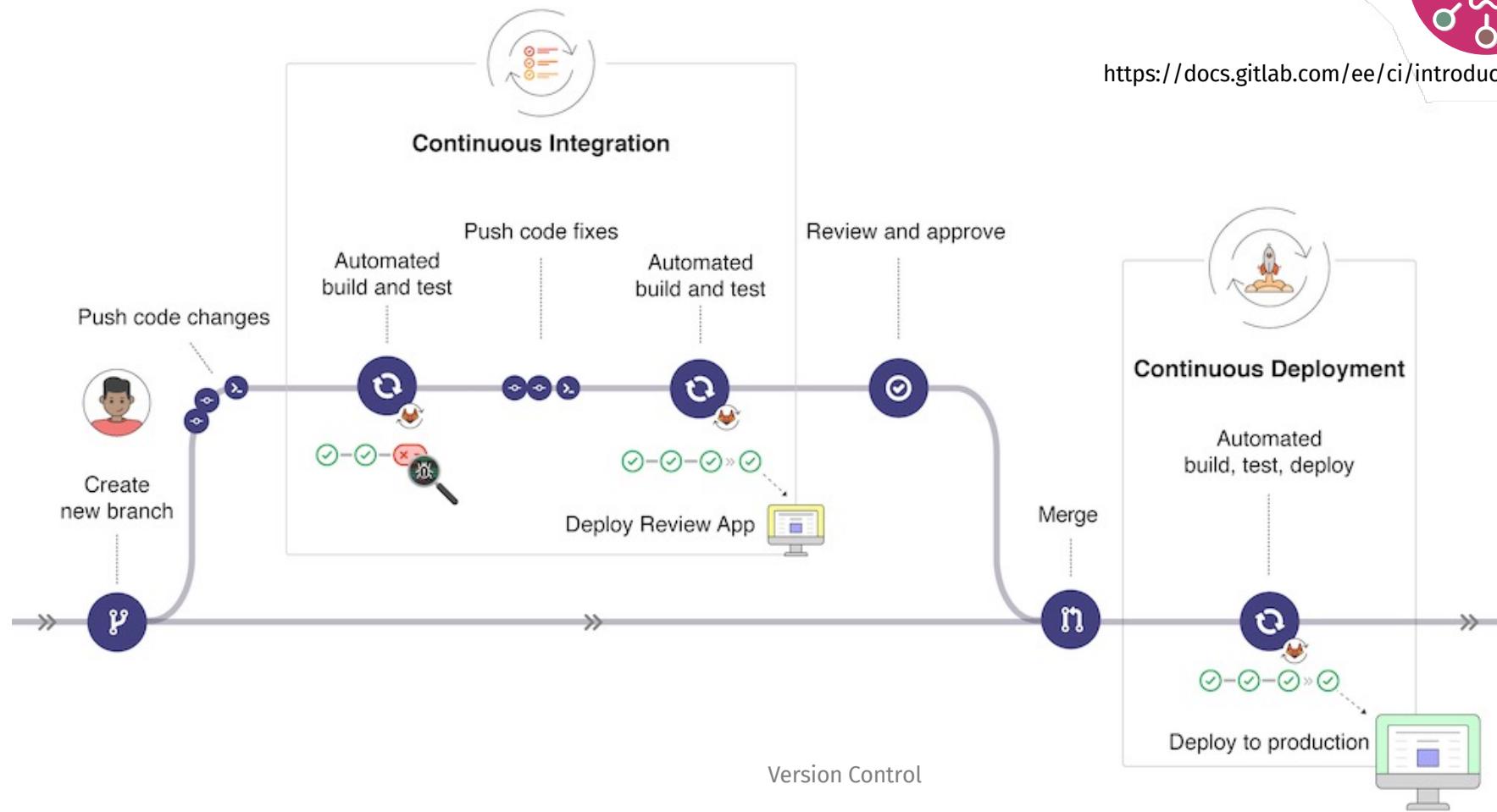


<https://about.gitlab.com/features/continuous-integration/>

Gitlab Workflow



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



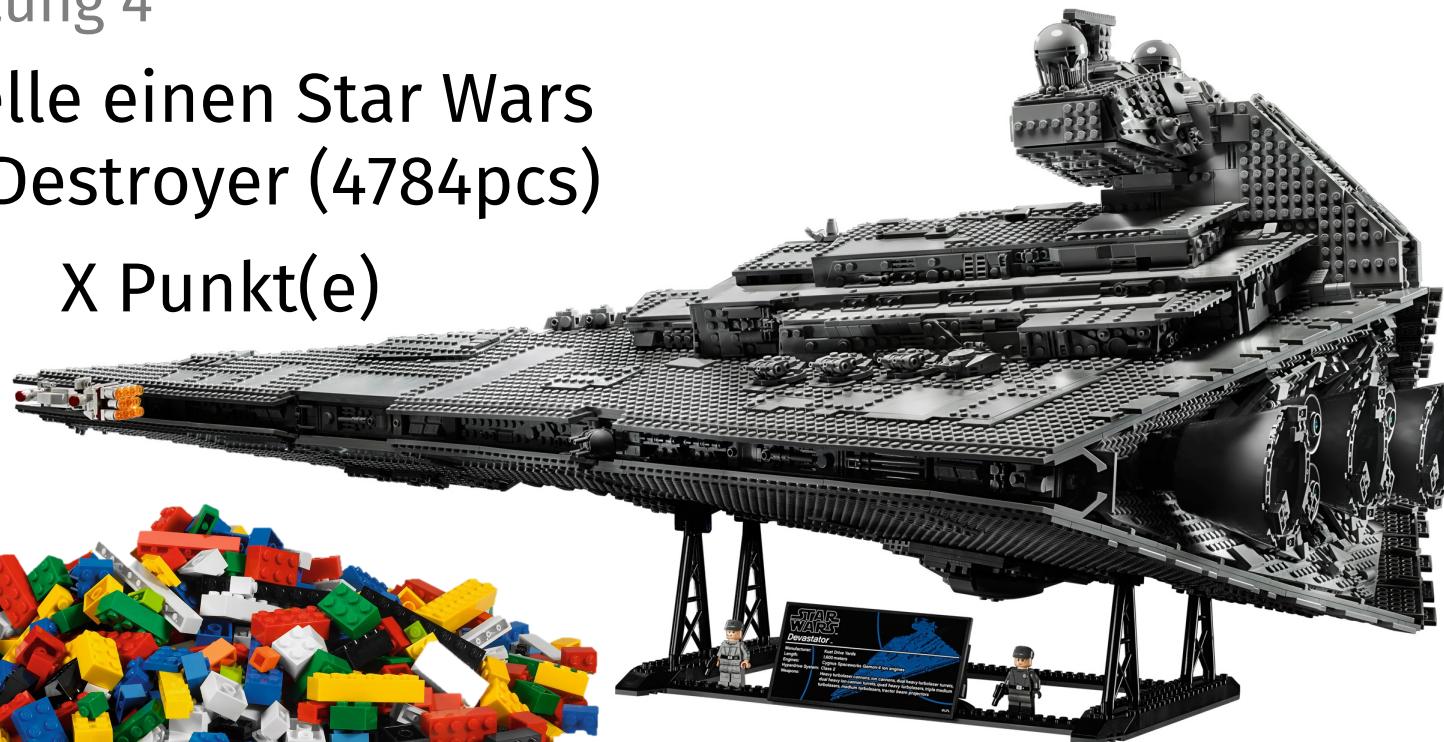


Abschätzung von Aufgaben

Abschätzung 4

Erstelle einen Star Wars
Star Destroyer (4784pcs)

X Punkt(e)



0	$\frac{1}{2}$	1	2
3	5	8	13
20	40	100	∞
?			

Abschätzung von Aufgaben

Abschätzung 5

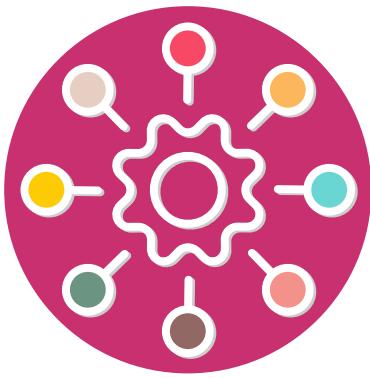


Erstelle einen Star Wars
Star Destroyer (37pcs)

X Punkt(e)



0	$\frac{1}{2}$	1	2
3	5	8	13
20	40	100	∞
?			



INTRODUCTION TO GIT FILE VERSIONING

Introduction to git file versioning



Contents

1 Goal	1
2 Installation	2
3 Basic operations	5
4 Branch and Merge	9
5 Gitflow	10
6 GIT commands	12
Bibliography	14