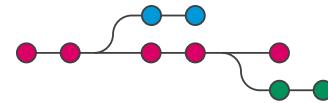




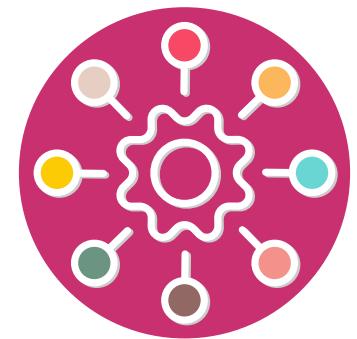
Systemdesign

Version Control



Studiengang Systemtechnik

Silvan Zahno silvan.zahno@hevs.ch



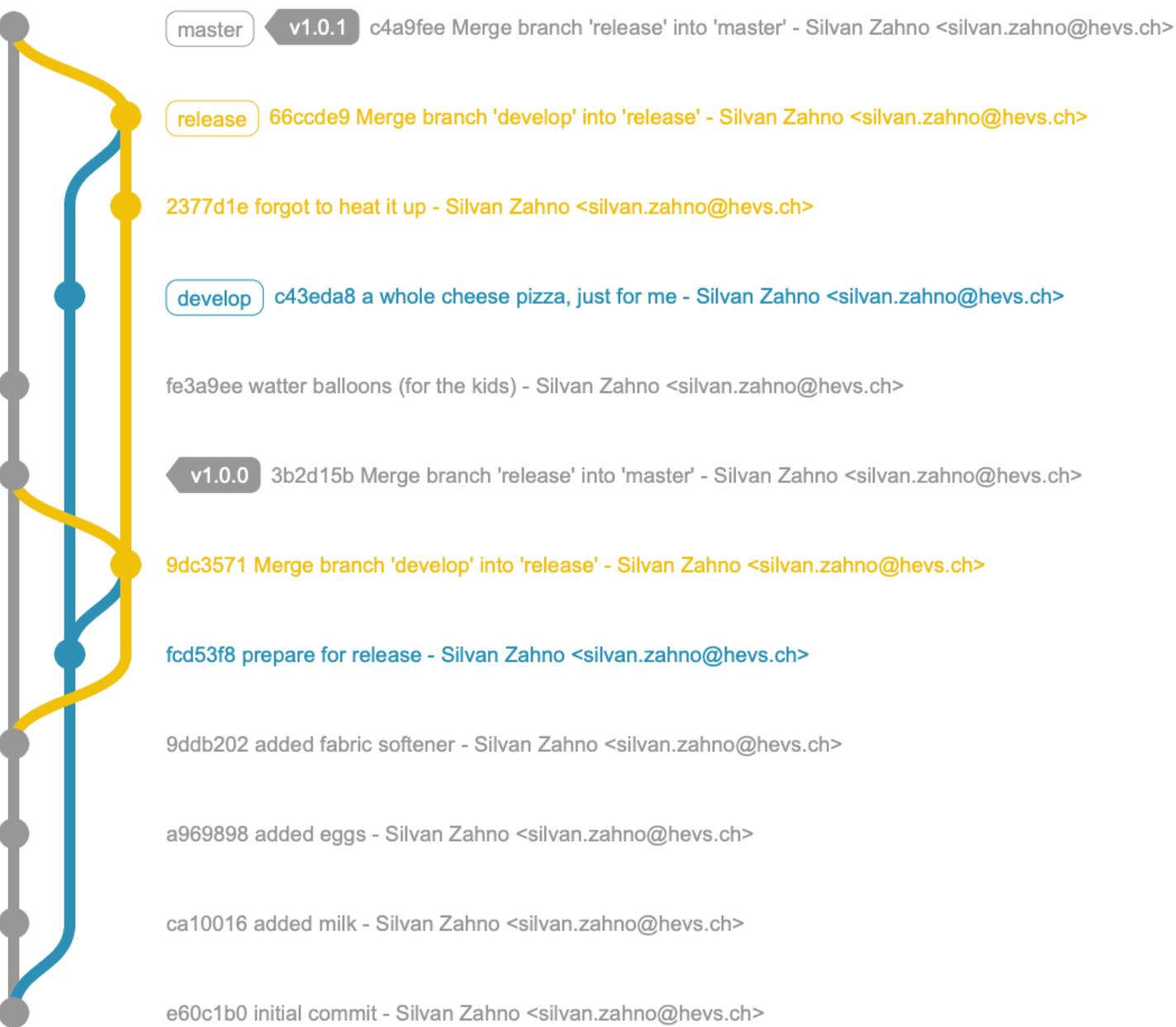
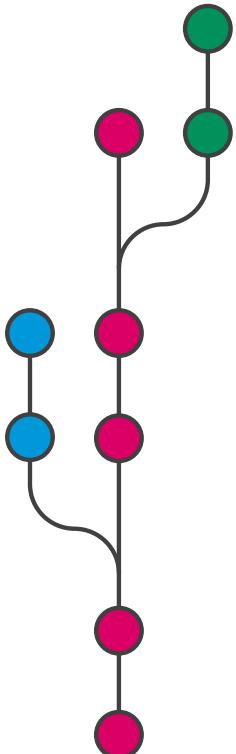


Ihr derzeitiges System

- .
- └── 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)

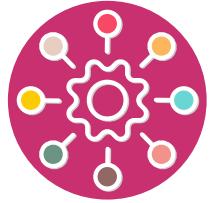
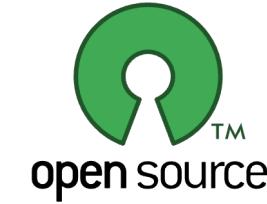
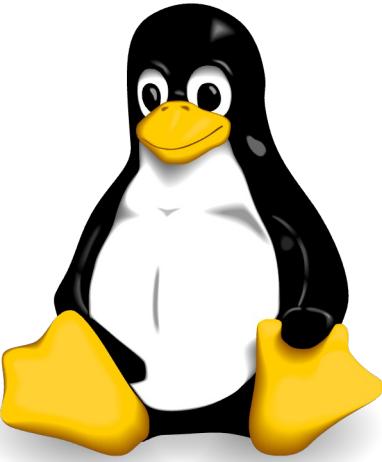


mercurial

Linux Torvalds

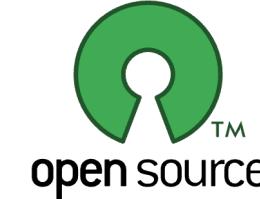
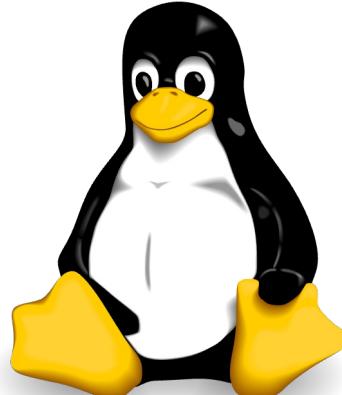
Linux (1991)

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

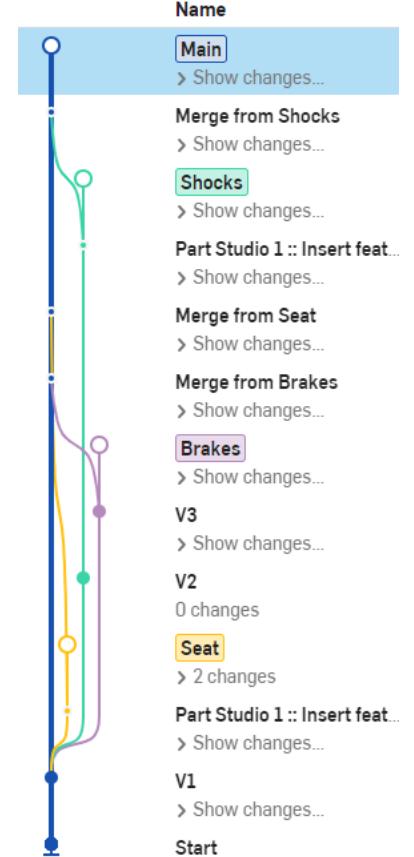


Versionierung

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



AUTODESK
Vault



Git Plattformen



Gitlab

<https://gitlab.com>

<https://gitlab.hevs.ch>

Github

<https://github.com>



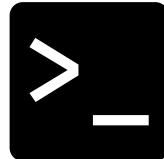
Bitbucket

<https://bitbucket.com>

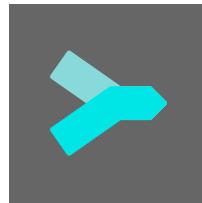


Git Hilfsmittel

Kommandozeile



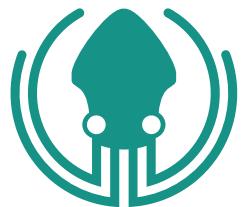
Sublime Merge



Git Cola



Git Kraken



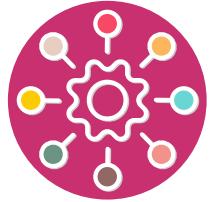
Fork



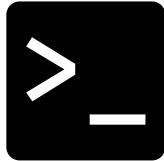
Tower



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

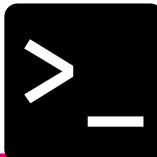
grow, mark and tweak your common history
branch  List, create, or delete branches
commit  Record changes to the repository
merge   Join two or more development histories together
rebase  Reapply commits on top of another base tip
reset   Reset current HEAD to the specified state
switch  Switch branches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

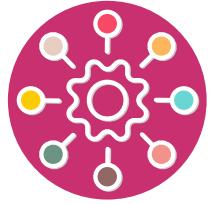
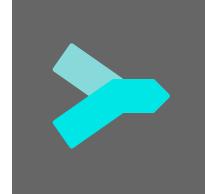
[zas@zac ~] (base)
$
```

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			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
rm	Entfernen von Dateien aus dem Arbeitsbaum und aus dem Index	tag	tag Commits auf einen anderen Basistipp anwenden
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



~/work/repo/edu/car/car-course

LICENSE UPGRADE REQUIRED

master

Commits Files Summary

BRANCHES (1) master

REMOTES (1) origin

TAGS (0) master

STASHES (0)

SUBMODULES (0)

1 unstaged file Commit Changes

Merge remote-tracking branch 'origin/master' 16

CHG: updated planning zas master origin/master Thu 08:11

ADD: ALU and ImmSrc doc Axam Thu, 13 Apr 15:19

ADD: EBS2/EBS3 specs Axam Tue, 11 Apr 15:25

FIX: memory stack images zas Tue, 4 Apr 11:00

FIX: errors in immediate and type images zas Tue, 4 Apr 07:46

ADD: files in arc exercises zas Mon, 3 Apr 13:30

ADD: note on Ripes memory management Axam Fri, 31 Mar 15:18

CHG: Planning zas Fri, 31 Mar 08:45

FIX: reverse engineering solution Axam Thu, 30 Mar 17:25

FIX: ISA syntax errors zas Tue, 28 Mar 07:59

Merge remote-tracking branch 'origin/master' 6

zас Tue, 28 Mar 07:29

FIX: errors in ISA zас Tue, 28 Mar 07:29

ADD: windows Geekbench window Axam Thu, 16 Mar 10:14

FIX: add scripts folder Axel Amand Thu, 16 Mar 09:31

REM: car-hevri and car-labs doc deployment Axel Amand Thu, 16 Mar 09:18

CHG: BEM labo from geekbench 5 to 6 zас Tue, 14 Mar 14:25

UPD: all PDFs Axam Wed, 8 Mar 19:10

FIX: errors in ARC, ISA, FUN and PER slides zас Tue, 7 Mar 09:09

Commit Hash 702c8ff738adbb639884c48b72e4bc68361d13c f163cea8c5f992b7c78549993694b6670e0da8b

Tree zas<silvan.zahn@nev.ch>

Author zas<silvan.zahn@nev.ch>

Date Thu, 20 Apr 2023 08:12

Parents 6e927ef6, 68810079

Branches master origin/master

Stats 16 files changed: 15 +10

...Collapse all

labo/latex/b2-content/scr/00-solution.tex -1 +5

Subsection: Simulation:

```
164 done;
165    beq x2, x2, main      # infinite loop
166  end(minted)
167 \newline\nullnewline
168 Each instruction takes one clock cycle => 19 are executed (addi x5, x0, 0 not
executed because of previous beq; addi x2, x0, 1 not executed because of jal).
=> 19/6M = 287.9 ns.
```

Subsection: Simulation:

```
164 done;
165    beq x2, x2, main      # infinite loop
166  end(minted)
167 \newline\nullnewline
168 Each instruction takes one clock cycle => 19 are executed (addi x5, x0, 0 not
executed because of previous beq; addi x2, x0, 1 not executed because of jal).
=> 19/6M = 287.9 ns.
```

On the EBS2 board @ 66MHz \rightarrow \\$T_{exec} = \frac{nb_cycles}{F_{sys}} = \frac{19}{(19)(6M)} = 287.9 ns.

On the EBS3 board @ 50MHz \rightarrow \\$T_{exec} = \frac{nb_cycles}{F_{sys}} = \frac{19}{(19)(5M)} = 380 ns.

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

Subsection: Umsetzung:

```
171 \begin{center}
172 \centerline{\includegraphics[width=0.9\paperwidth]{scr/sol/simulation.pdf}}
173 \end{center}
```

Subsection: Umsetzung:

```
174 \begin{center}
175 \centerline{\includegraphics[width=0.9\paperwidth]{scr/sol/simulation.pdf}}
176 \end{center}
```

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

Le \textbf{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.

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

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

g:riscv-mainDecoder-code:

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

Le \textbf{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.

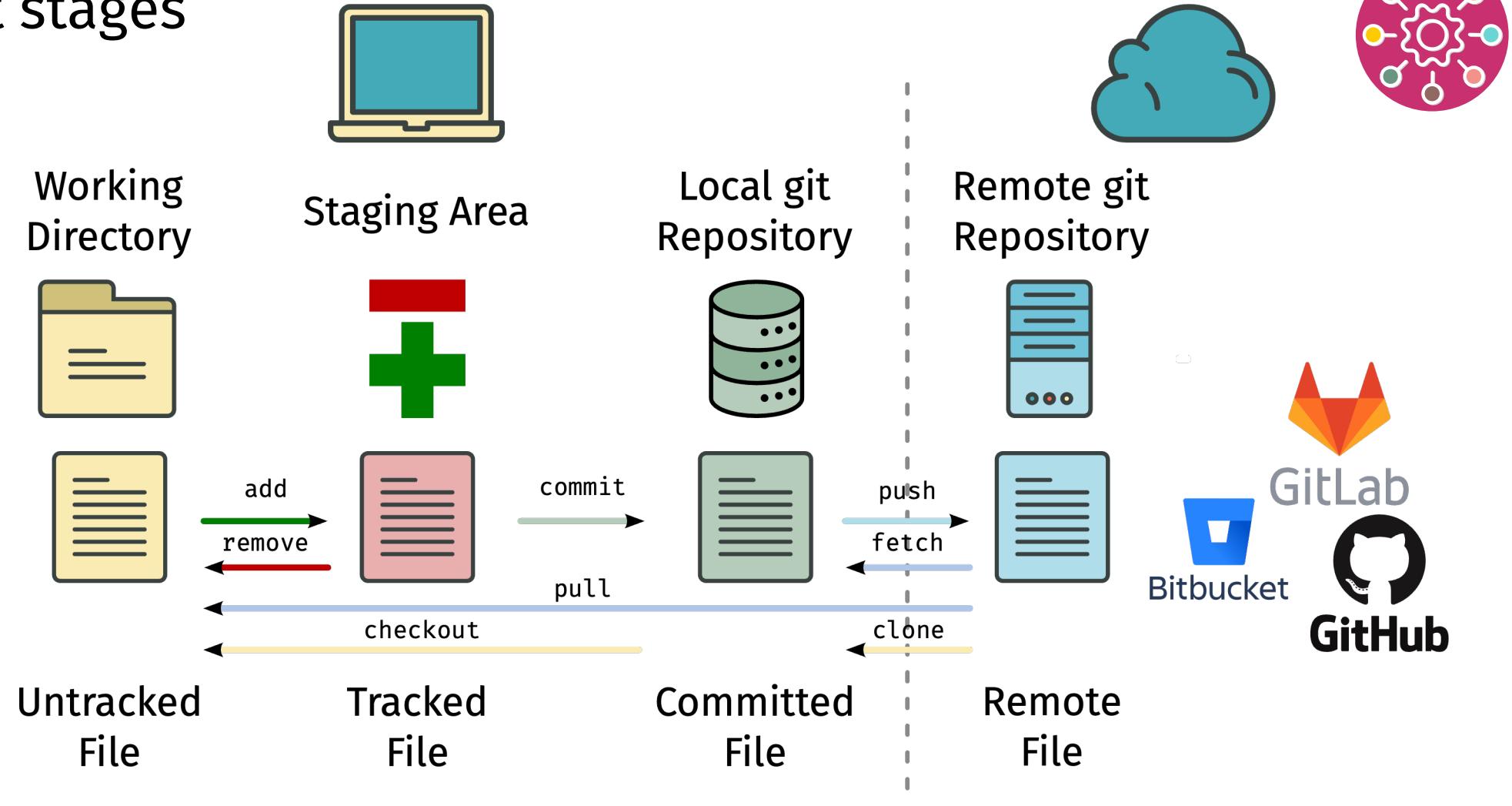
InnoDB (Dans HDL Designer, lorsque vous sélectionnez le type de contenu d'un bloc, choisissez \textit{VHDL File -> Architecture}, et contrôlez que le langage soit défini sur \textit{VHDL 2008}). Sur la page suivante, \textit{Architecture} correspond au nom de la vue (un bloc peut avoir différents contenus)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

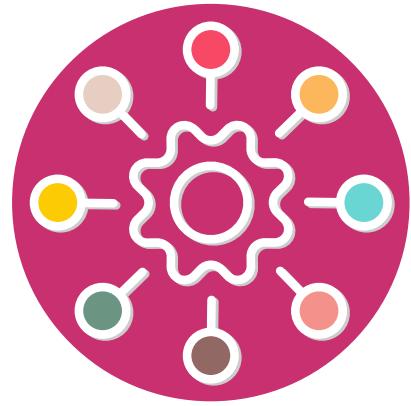
Schreiben Sie hierzu für beide Subblöcke, \textbf{mainDecoder} sowie \textbf{ALUDecoder}, eine Wahrheitstabelle für alle benötigten Instruktionen.

```
113 \subsubsection{ALU}
114 \opt{f}{%
115 \begin{table}[h]
116 \caption{L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:}
117 \end{table}
118 \opt{d}{%
119 \begin{table}[h]
120 \caption{Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden Tabelle:}
121 \end{table}
122 \begin{table}[h]
123 \caption{...}
```

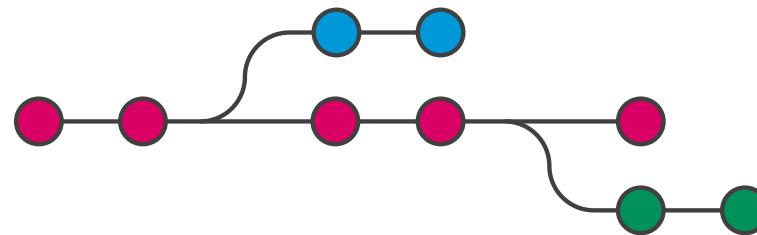
...begin[table][h]

Git stages





Git Branch und Merge Beispiele

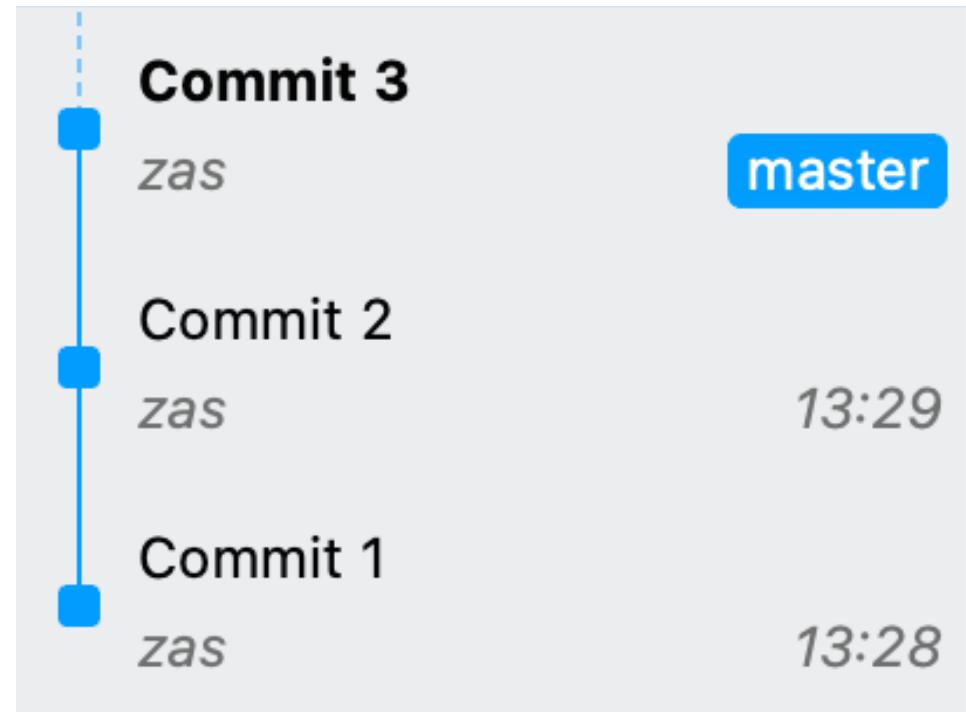




Branch und Merge

Initial repo state

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



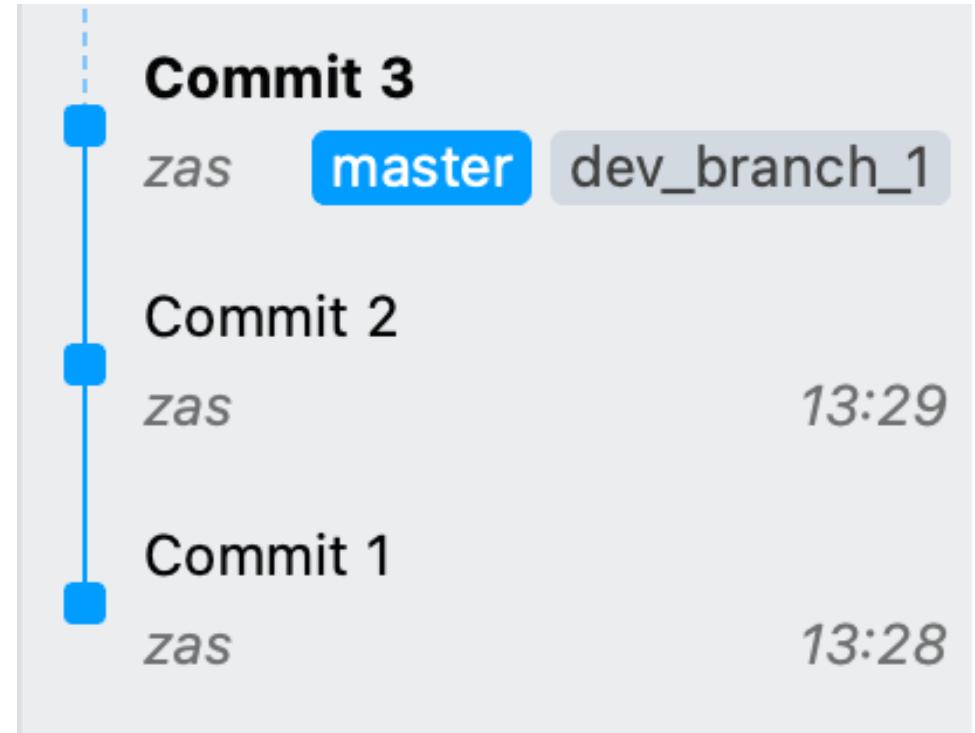


Branch und Merge

Create branch dev_branch_1

Erstelle branch dev_branch_1

```
$ git branch dev_branch_1
```





Branch und Merge

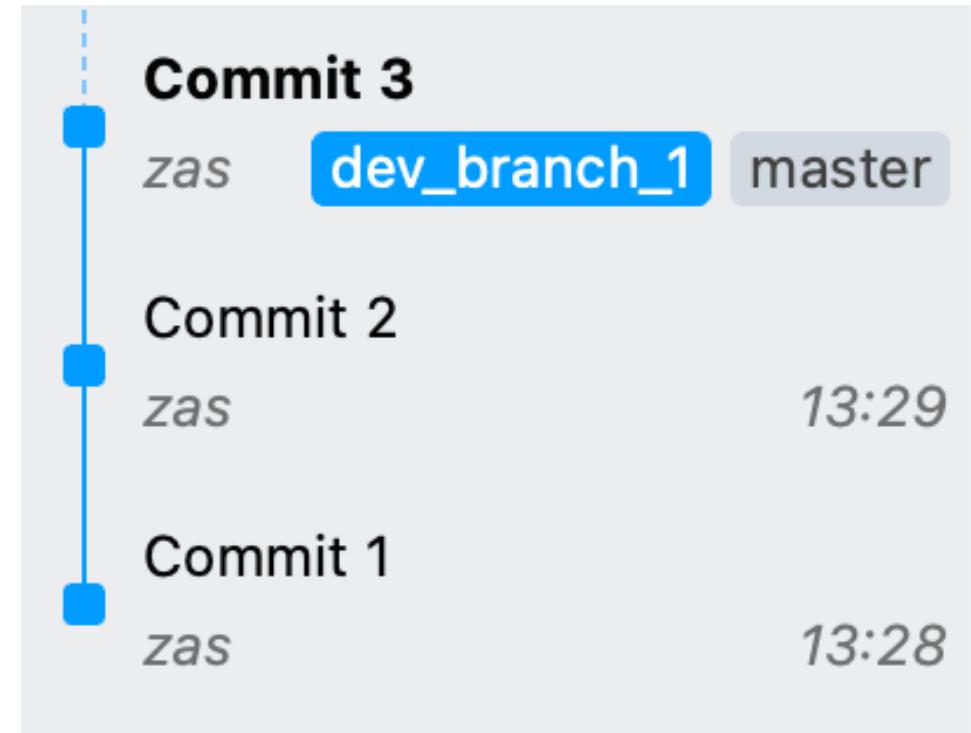
Checkout branch dev_branch_1

Prüfen Sie, in welchem Zweig wir uns befinden

```
$ git branch
```

Checkout branch dev_branch_1

```
$ git checkout dev_branch_1
```





Branch and Merge

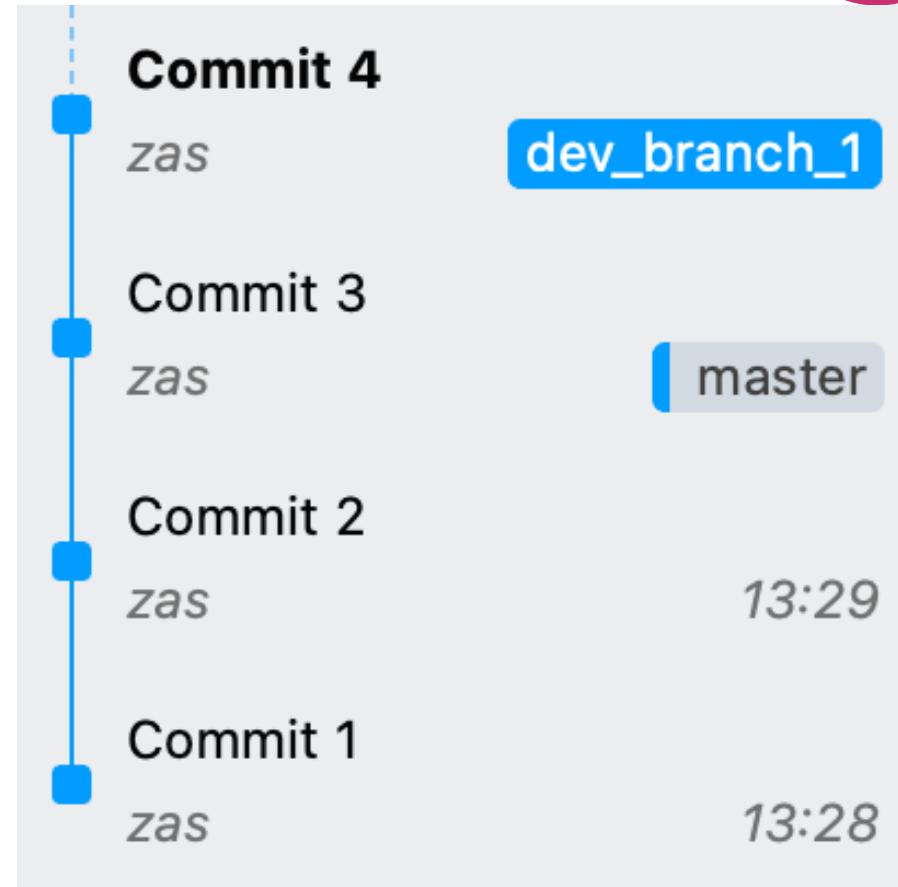
Commit on dev_branch_1

Stage new file

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

Commit stages files

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





Branch und Merge

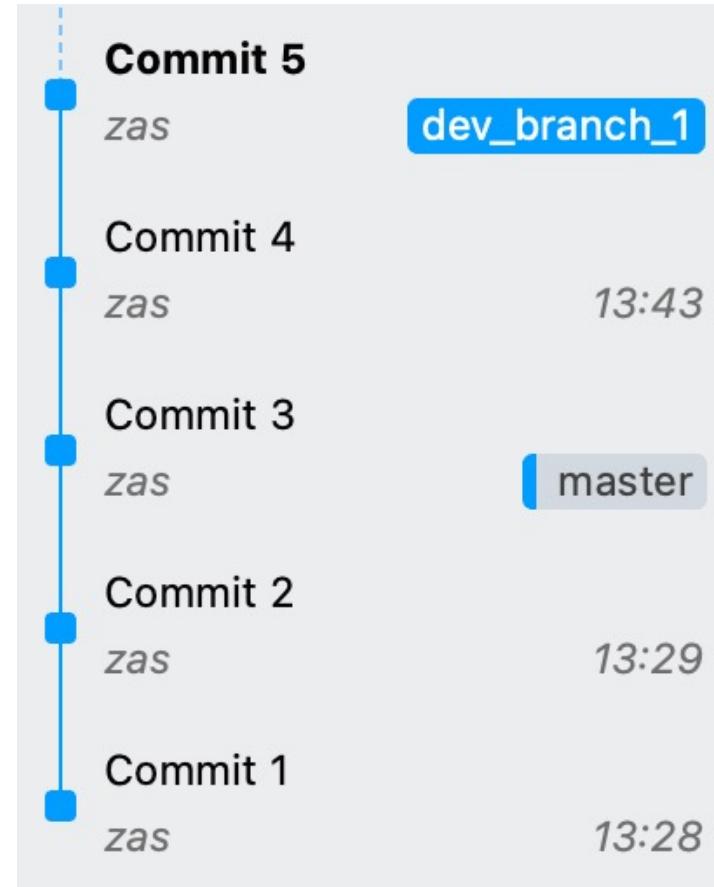
Commit on dev_branch_1

Stage new file

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

Commit stages files

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





Branch und Merge

Commit on master branch

Checkout master branch

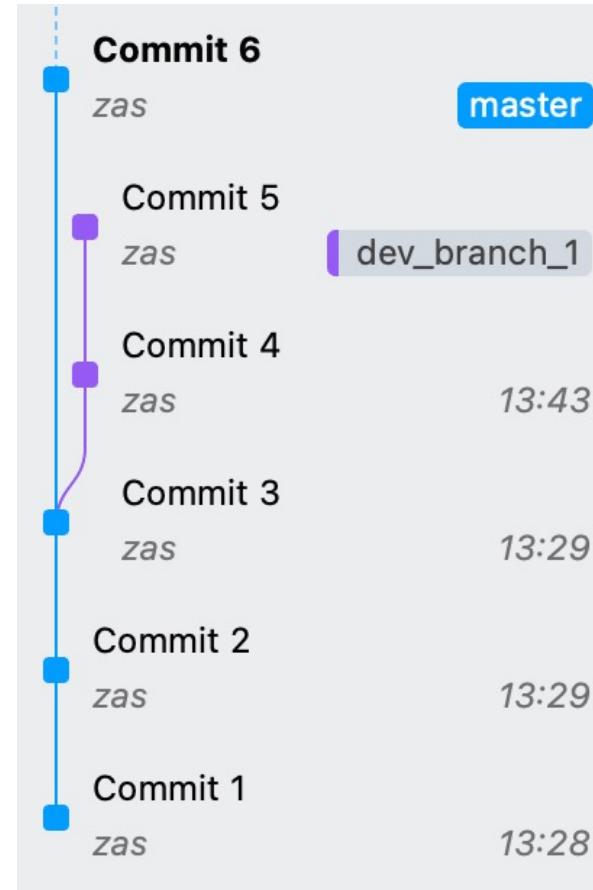
```
$ git checkout master
```

Stage new file

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

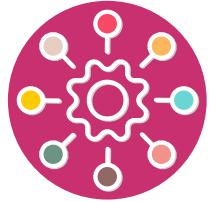
Commit stages files

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



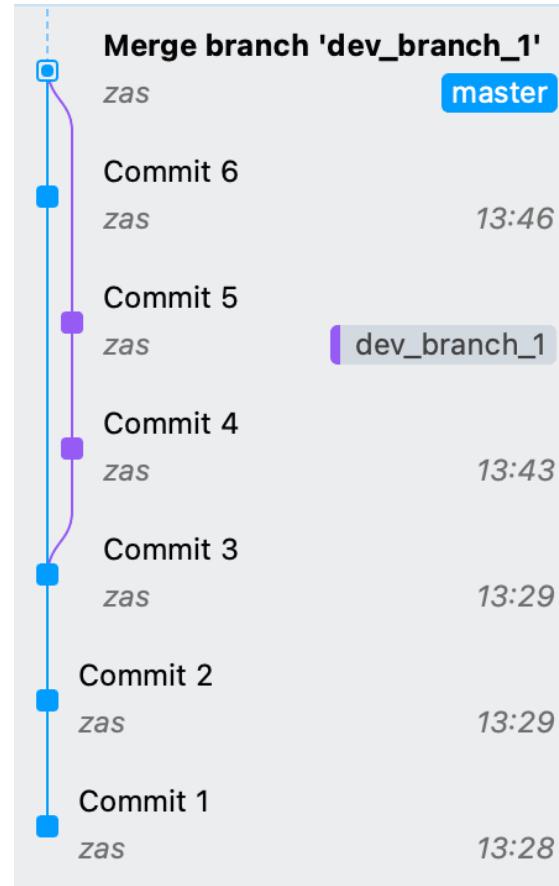
Branch und Merge

Three way merge master and dev_branch_1

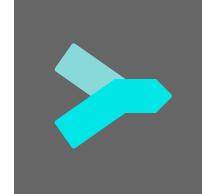


Merge dev_branch_1 into master

```
$ git merge dev_branch_1
```



Demonstration



~/work/repo/edu/car/car-course

LICENSE UPGRADE REQUIRED

master

Commits Files Summary

BRANCHES (1) master

REMOTES (1) origin

TAGS (0) master

STASHES (0)

SUBMODULES (0)

1 unstaged file Commit Changes

Merge remote-tracking branch 'origin/master' 16

CHG: updated planning zas master origin/master Thu 08:11

ADD: ALU and ImmSrc doc Axam Thu, 13 Apr 15:19

ADD: EBS2/EBS3 specs Axam Tue, 11 Apr 15:25

FIX: memory stack images zas Tue, 4 Apr 11:00

FIX: errors in immediate and type images zas Tue, 4 Apr 07:46

ADD: files in arc exercises zas Mon, 3 Apr 13:30

ADD: note on Ripes memory management Axam Fri, 31 Mar 15:38

CHG: Planning zas Fri, 31 Mar 08:45

FIX: reverse engineering solution Axam Thu, 30 Mar 17:25

FIX: ISA syntax errors zas Tue, 28 Mar 07:59

Merge remote-tracking branch 'origin/master' 6

zас Tue, 28 Mar 07:29

FIX: errors in ISA zас Tue, 28 Mar 07:29

ADD: windows Geekbench window Axam Thu, 16 Mar 10:14

FIX: add scripts folder Axel Amand Thu, 16 Mar 09:31

REM: car-hevry and car-labs doc deployment Axel Amand Thu, 16 Mar 09:18

CHG: BEM labo from geekbench 5 to 6 zас Tue, 14 Mar 14:25

UPD: all PDFs Axam Wed, 8 Mar 19:10

FIX: errors in ARC, ISA, FUN and PER slides zас Tue, 7 Mar 09:09

Commit Hash 702c8ff738adbf6639884c48b72e4bc68361d13c f163cea8c5f992b7c78549993694b6670e0da8b

Tree zas+silvan.zahn@nevs.ch

Author zas+silvan.zahn@nevs.ch

Date Thu, 20 April 2023 08:12

Parents 6e927ef6, 68810079

Branches master origin/master

Stats 16 files changed: 15 +10

...Collapse all

Subsection: Simulation:

done:
bed x2, x2, main # infinite loop
end(minted)

Each instruction takes one clock cycle => 19 are executed (addi x5, x0, 0 not executed because of previous beq; addi x2, x0, 1 not executed because of jal). => 19/6M = 287.9 ns.

164 done:
165 bed x2, x2, main # infinite loop
166 end(minted)
167 \newline\nullnewline
168 Each instruction takes one clock cycle => 19 are executed (addi x5, x0, 0 not executed because of previous beq; addi x2, x0, 1 not executed because of jal).
169 On the EBS2 board @ 66MHz \rightarrow \\$T_{exec} = \frac{nb_cycles}{F_{sys}} = \frac{19}{(19)(6M)} = 287.9 ns.
170 On the EBS3 board @ 50MHz \rightarrow \\$T_{exec} = \frac{nb_cycles}{F_{sys}} = \frac{19}{(19)(5M)} = 380 ns.
171
172 On the EBS3 board @ 50MHz \rightarrow \\$T_{exec} = \frac{nb_cycles}{F_{sys}} = \frac{19}{(19)(5M)} = 380 ns.
173
174 \begin{center}
175 \centerline{\includegraphics[width=0.9\paperwidth]{scr/sol/simulation.pdf}}
176 \end{center}

Subsection: Umsetzung:

Le \textbf{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.

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

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

g:riscv-mainDecoder-code:

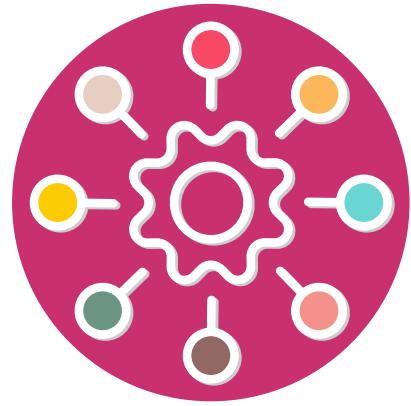
```
110 \opt{f}{\caption{figure}{Exemple de code MainDecoder}}
111 \opt{d}{\caption{figure}{MainDecoder Code-Beispiel}}
112 \label{fig:riscv-mainDecoder-code}
```

Le \textbf{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.

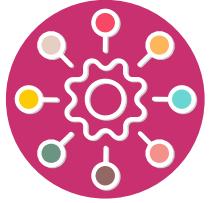
InnoDB (Dans HDL Designer, lorsque vous sélectionnez le type de contenu d'un bloc, choisissez \textit{VHDL File -> Architecture}, et contrôlez que le langage soit défini sur \textit{VHDL 2008}). Sur la page suivante, \textit{Architecture} correspond au nom de la vue (un bloc peut avoir différents contenus)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

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

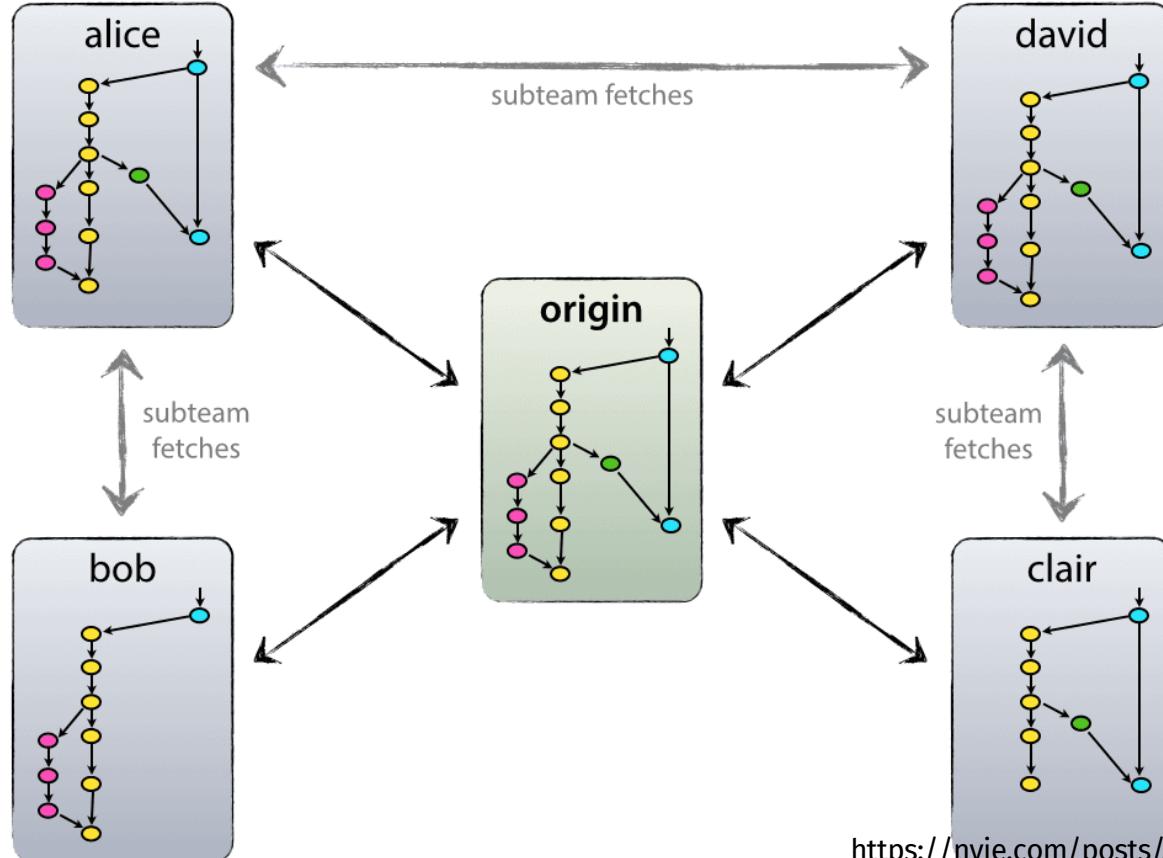
```
110 \opt{f}{\caption{figure}{Exemple de code MainDecoder}}
111 \opt{d}{\caption{figure}{MainDecoder Code-Beispiel}}
112 \label{fig:riscv-mainDecoder-code}
113
114 \subsubsection{ALU}
115 \opt{f}{%
116     \text{L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:}
117 }
118 \opt{d}{%
119     Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden Tabelle:
120 }
121
122 \begin{table}[h]
```



Gitflow

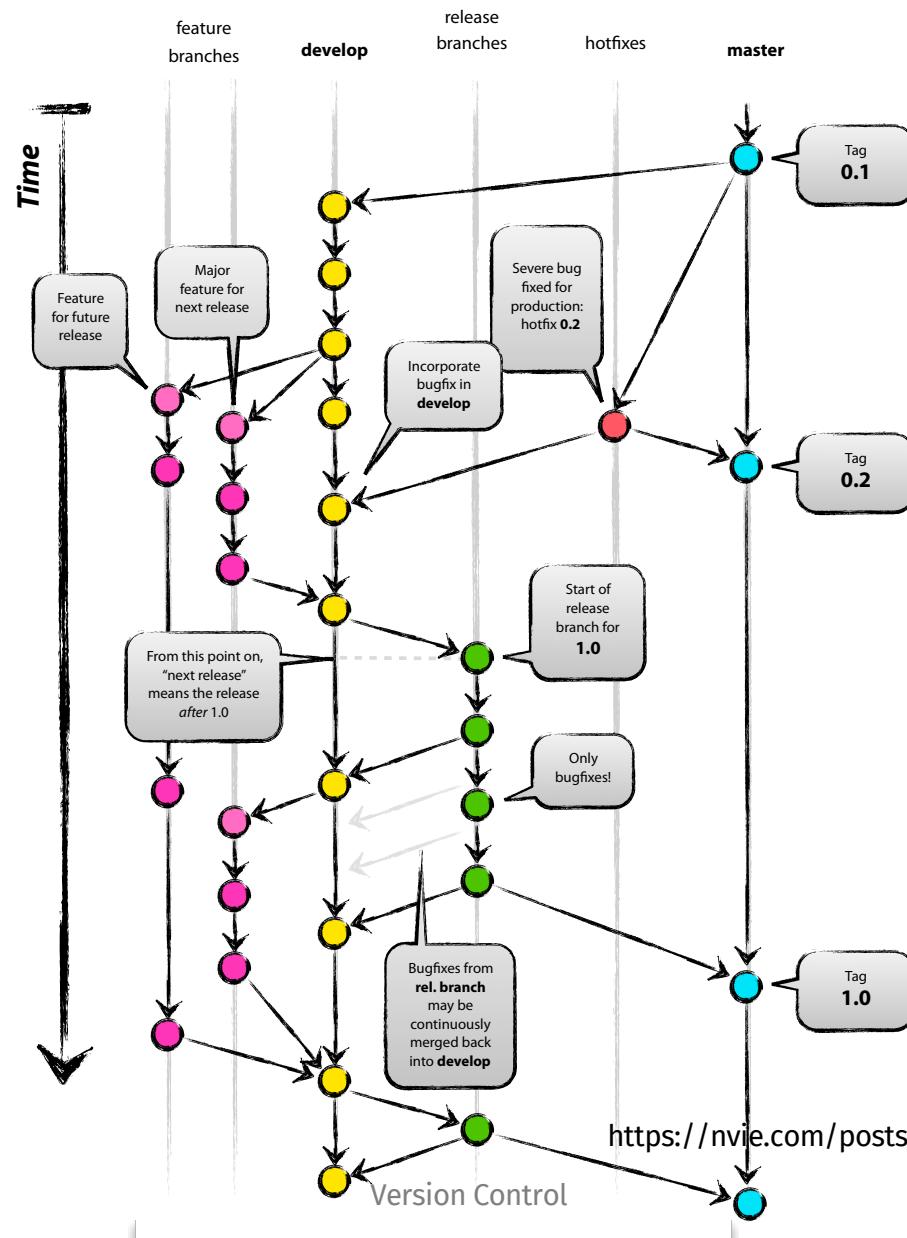


Gitflow Zusammenarbeit

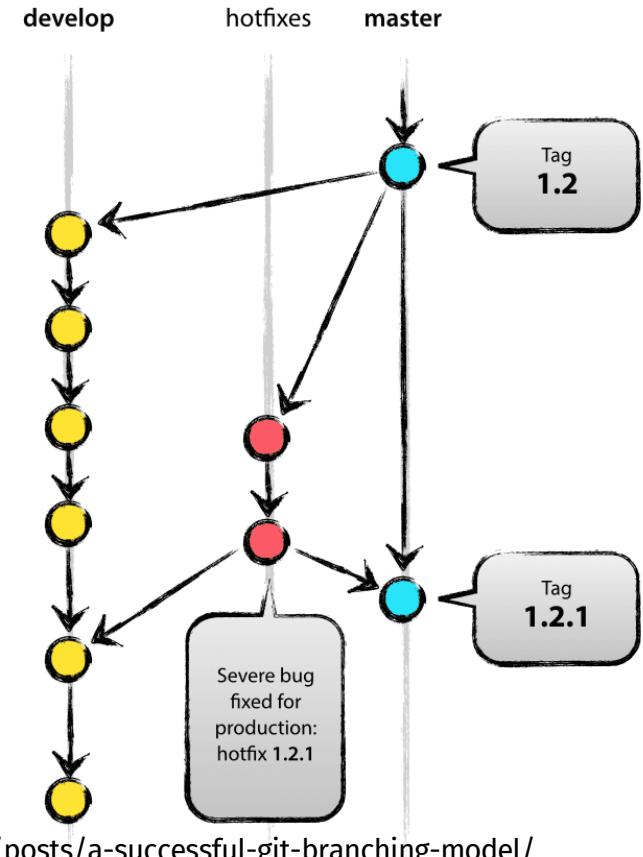
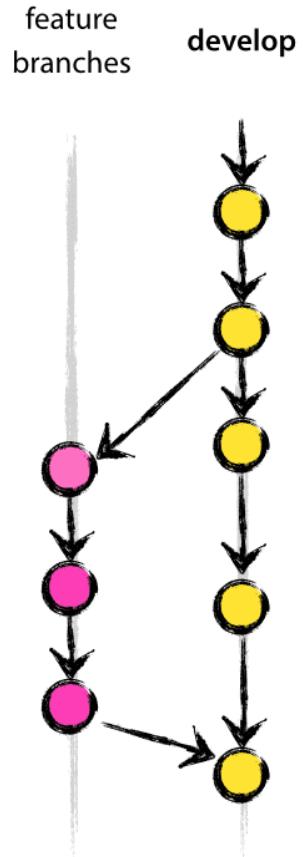
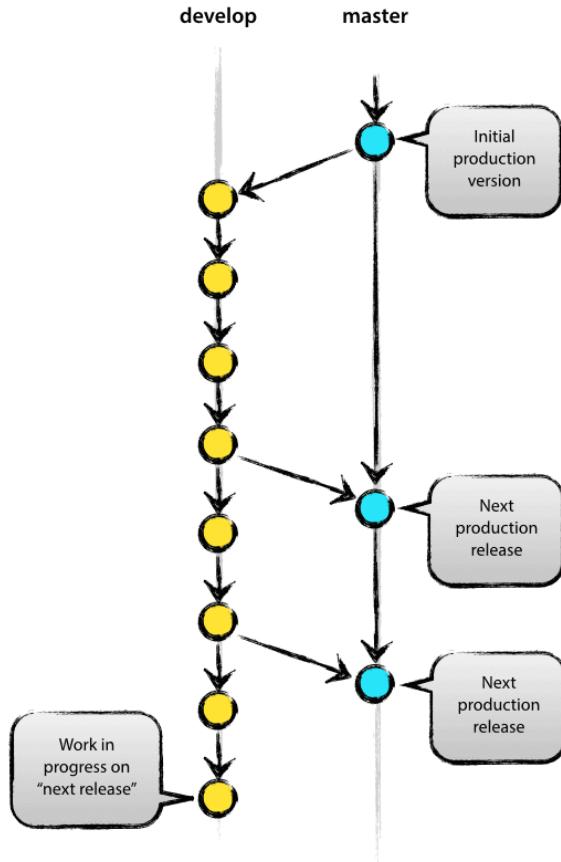


<https://nvie.com/posts/a-successful-git-branching-model/>

Gitflow



Gitflow Branching

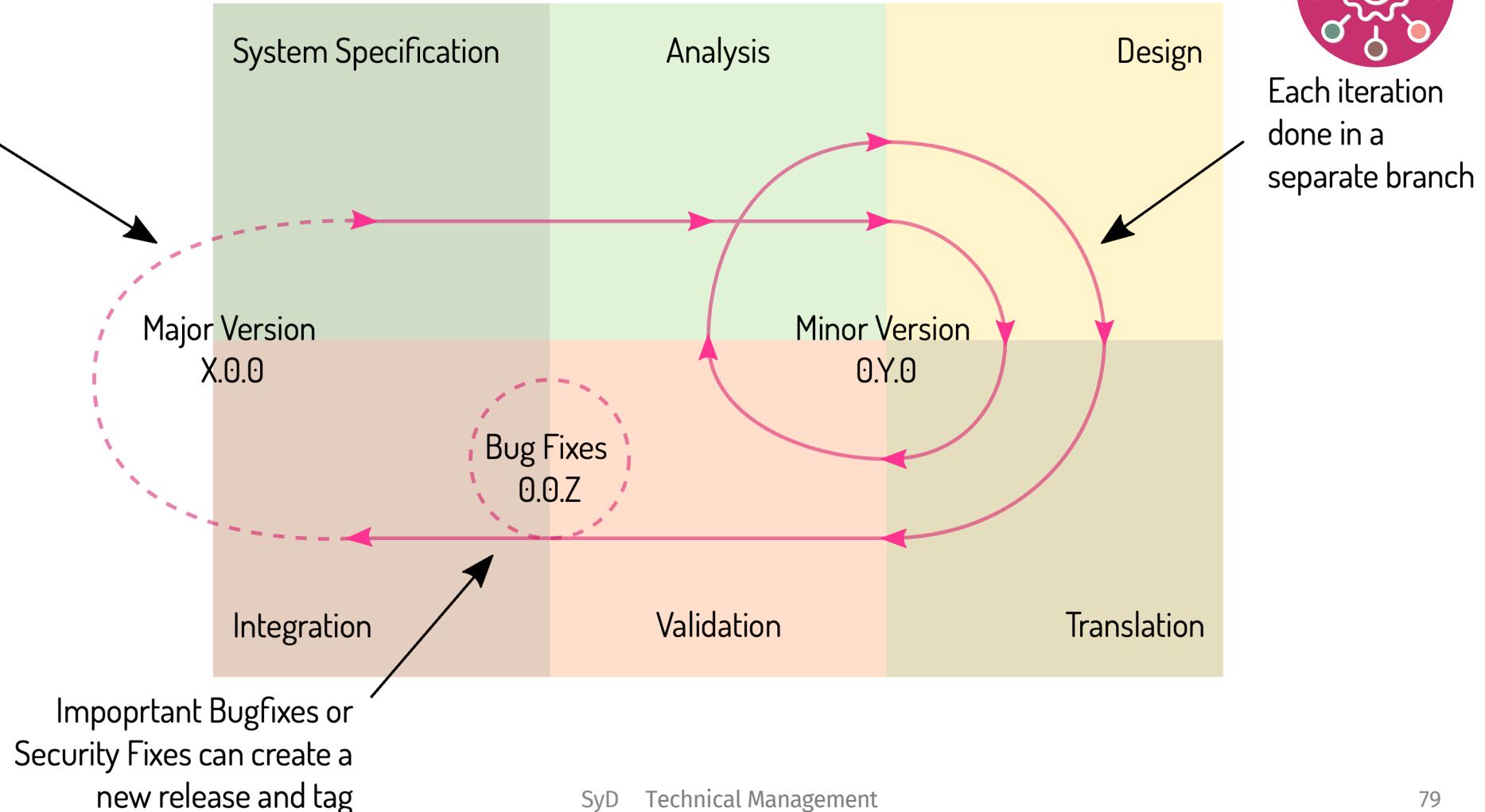


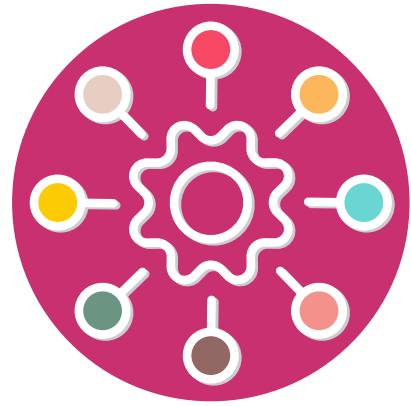
<https://nvie.com/posts/a-successful-git-branching-model/>

Gitflow vs 6q



Each major version change creates a tag



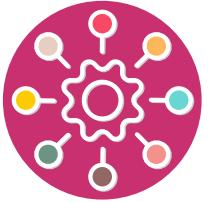


Git CI/CD

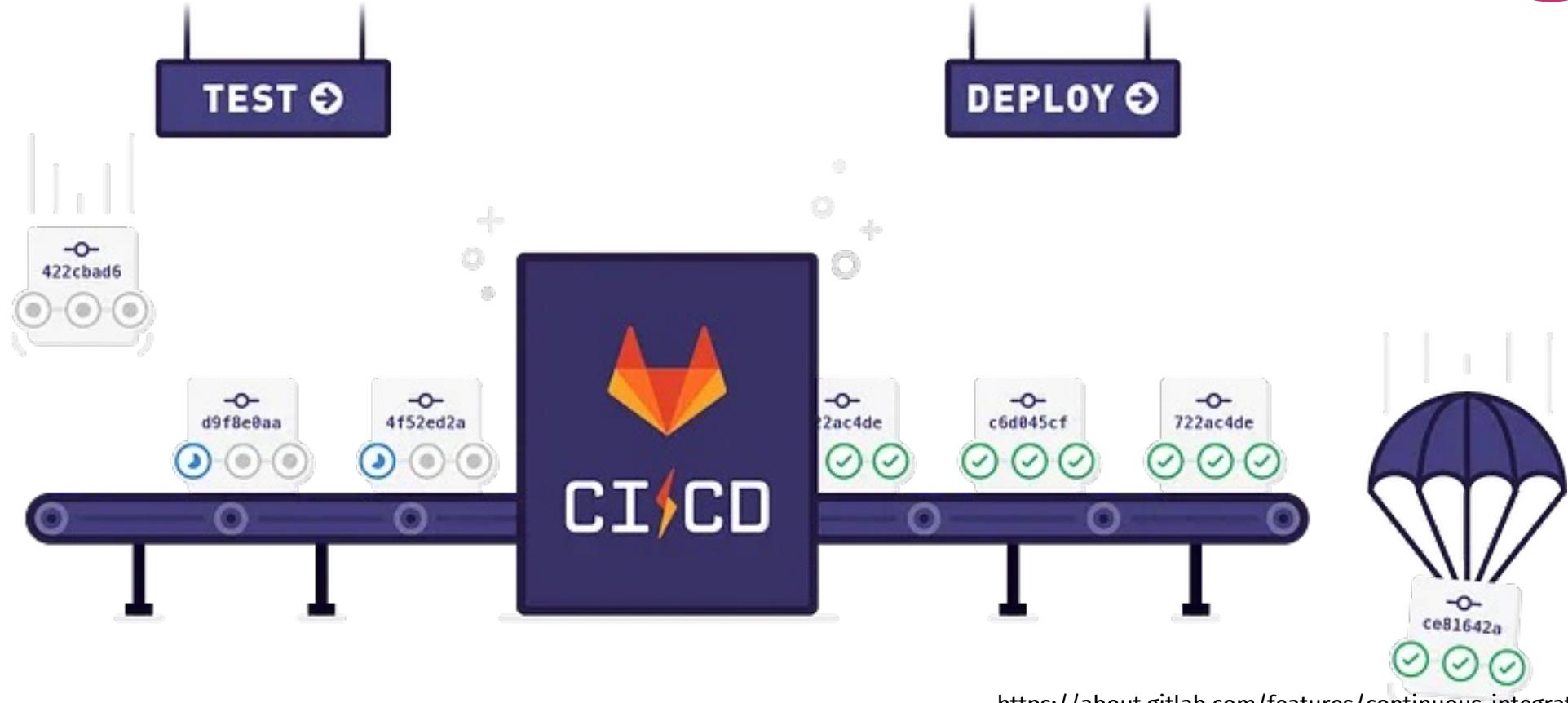
Was ist CI/CD?



- 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.

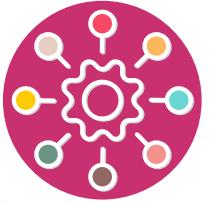


Was ist CI/CD?

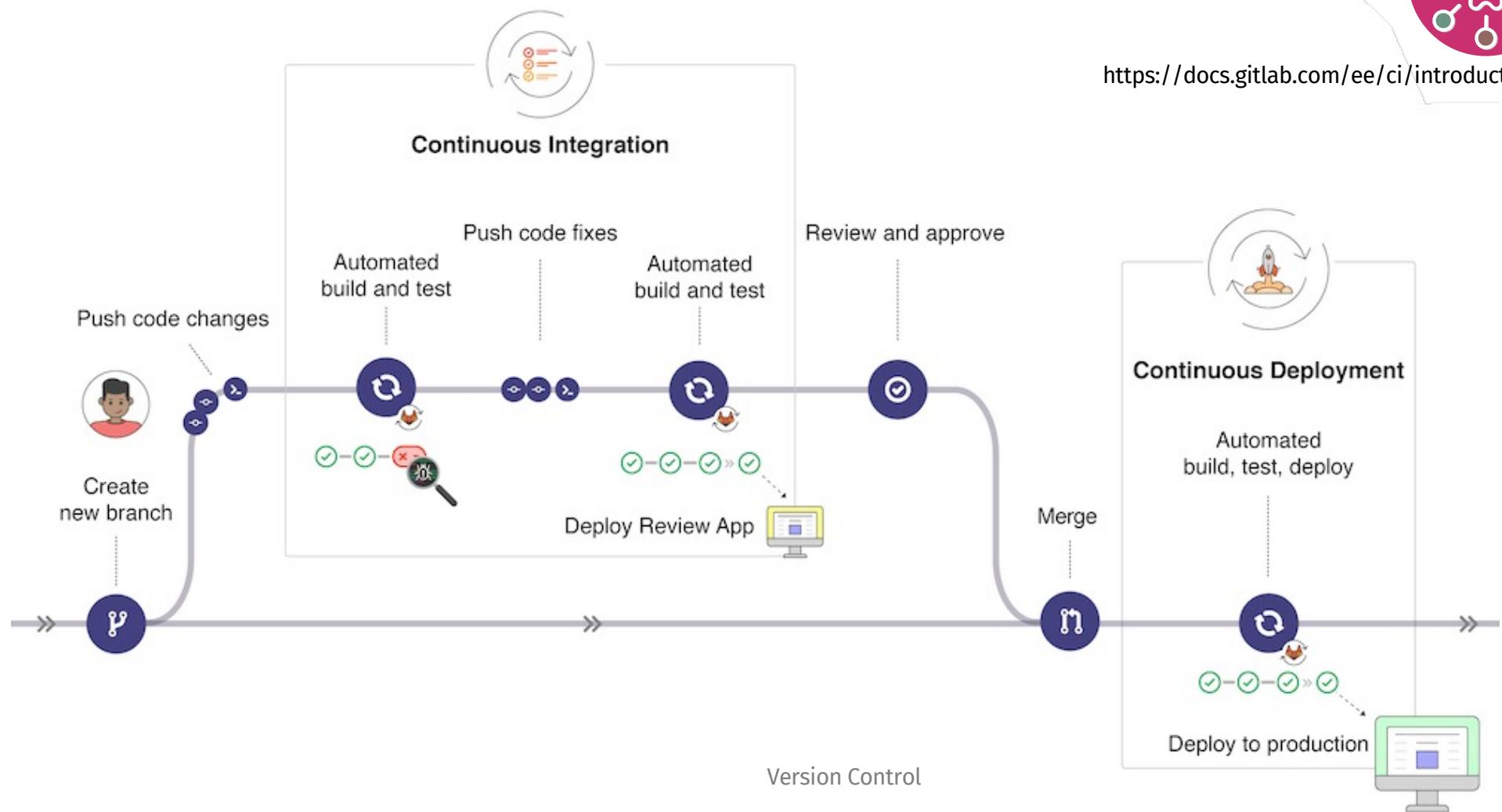


<https://about.gitlab.com/features/continuous-integration/>

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	∞
?	☕		

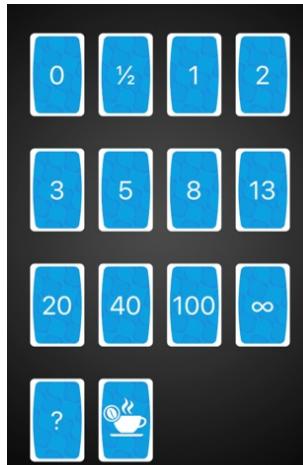
Abschätzung von Aufgaben

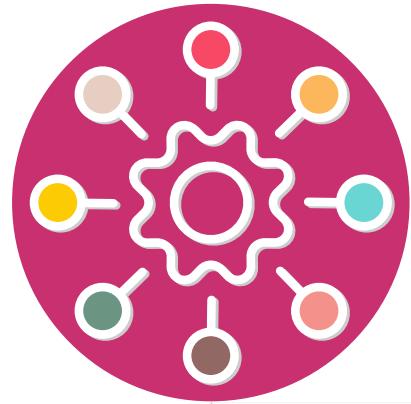
Abschätzung 5



Erstelle einen Star Wars
Star Destroyer (37pcs)

X Punkt(e)





SYSTEMS DESIGN / INTRODUCTION TO GIT - PART A

Introduction to git - Part A

Installation & Setup



Contents

1 Goal	1
2 Installation	2
3 Markdown	5
4 Outro	7
A GIT commands	8
B Most used Git commands	9

SYSTEMS DESIGN / INTRODUCTION TO GIT - PART B

Introduction to git - Part B



Contents

1 Goals	2
2 Outils	2
3 Basis Operationen	3
4 Branch and Merge	11
5 Gitgraph	15
6 Gitflow	16
7 Extras	18

