

Estimation des tâches

Estimation 2

Créer un petit maison
X Point(s)



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

Estimation des tâches

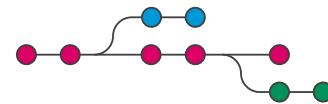
Estimation 3

Créer un programmable robot X Point(s)



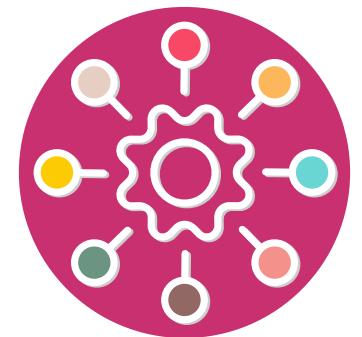


Conception des systèmes Version Control



Filière Systèmes industriels

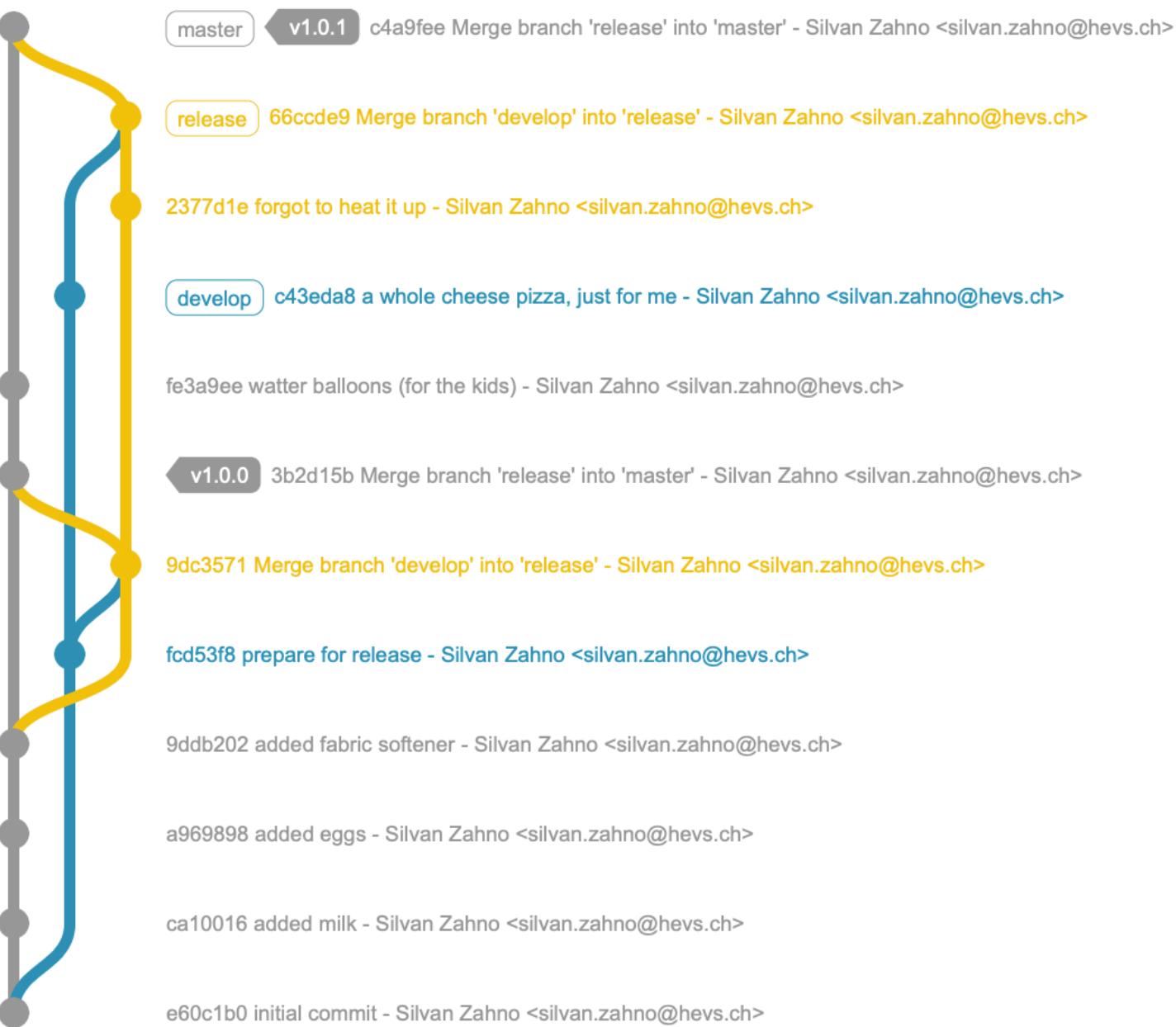
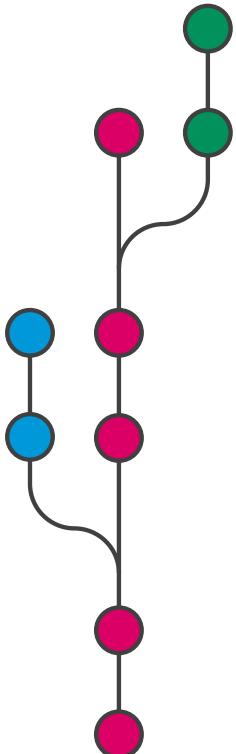
Silvan Zahno silvan.zahno@hevs.ch



Votre système actuel



Version control





Outils possibles

Git (git)



Subversion (svn)



Mercurial (hg)



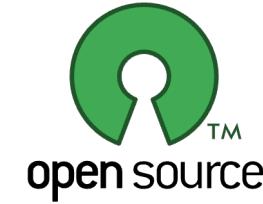
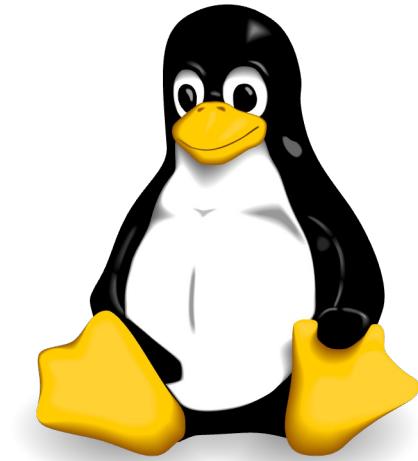
mercurial

SyD Version Control

Linux Torvalds

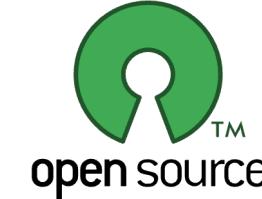
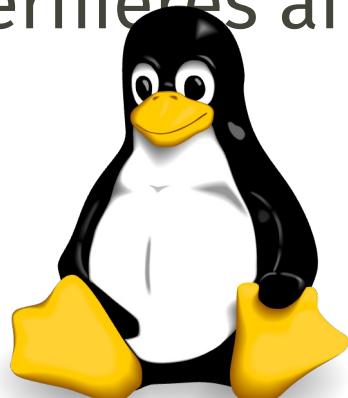
Linux (1991)

Git (2005)



Pourquoi Linux a besoin de git

Linux est devenu le plus grand projet de développement collaboratif de l'histoire de l'informatique au cours des 30 dernières années.



- 600 distributions Linux actives
- 85% de tous les smartphones
- 500 superordinateurs de pointe
- > 27,8 millions de lignes de code
- > 12 000 contributeurs
- > 1 million de commits

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

Pourquoi pour un Power&Control ou Design&Material git est nécessaire



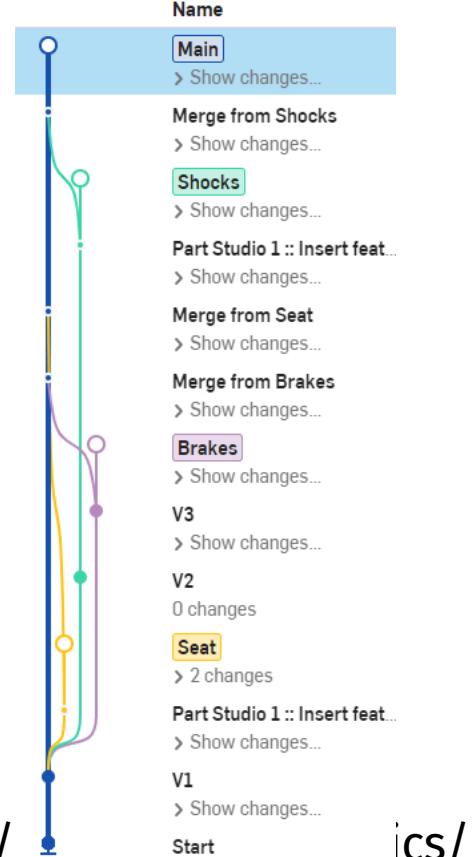
Version

- Suivi des modifications
- Collaboration
- Capacités de retour en arrière
- Documentation
- Déploiement



AUTODESK
Vault

<https://truelist.co/>



Plateformes Git



Gitlab

<https://gitlab.com>

<https://gitlab.hevs.ch>



Github

<https://github.com>



Bitbucket

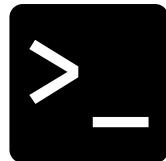
<https://bitbucket.com>



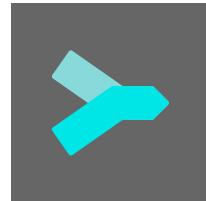
Bitbucket

Outils Git

Ligne de commande



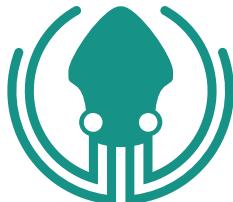
Sublime Merge



Git Cola



Git Kraken



Fork



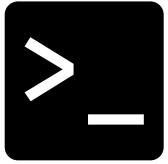
Tower



SmartGit



Git ligne de commande



```
zas - zas@zac - ~ - zsh - 99x52
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

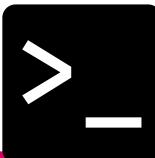
grow, mark and tweak your common history
branch   List, create, or delete branches
commit   Record changes to the repository
merge    Join two or more development histories together
rebase   Reapply commits on top of another base tip
reset   Reset current HEAD to the specified state
switch  Switch branches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

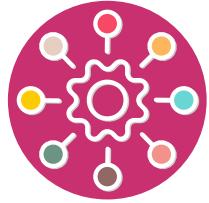
[zas@zac ~] (base)
$
```

git commands



Command	Description	Command	Description
Démarrer une zone de travail		Développez, marquez et modifiez votre histoire commune	
	Cloner un dépôt dans un nouveau répertoire		Lister, créer ou supprimer des branches
	Créer un dépôt Git vide ou réinitialiser un dépôt existant		Changer de branche ou restaurer les fichiers de l'arborescence de travail
Travailler sur la modification en cours		Enregistrer les modifications apportées à la base de données	
	Ajouter le contenu d'un fichier à l'index		Afficher les changements entre les livraisons, les livraisons et l'arbre de travail, etc.
	Déplacer ou renommer un fichier, un répertoire ou un lien symbolique		Joindre deux ou plusieurs historiques de développement
	Réinitialiser le HEAD actuel à l'état spécifié		Réappliquer les commits par-dessus un autre conseil de base
	Supprimer des fichiers de l'arbre de travail et de l'index		Créer, lister, supprimer ou vérifier un objet d'étiquette
Examiner l'historique et l'état		Collaborer	
	Afficher les journaux de livraison		Télécharger des objets et des références à partir d'un autre dépôt
	Afficher différents types d'objets		Récupérer et intégrer des objets d'une autre base de données ou d'une branche locale
	Afficher l'état de l'arbre de travail		Mettre à jour les références distantes ainsi que les objets associés

Sublime Merge



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

LICENSE UPGRADE REQUIRED

master

Commits Files Summary

LOCATIONS

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-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 702c8ff178adbf6639884c48b72e4bc68361d13c 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

Merge remote-tracking branch 'origin/master'

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.

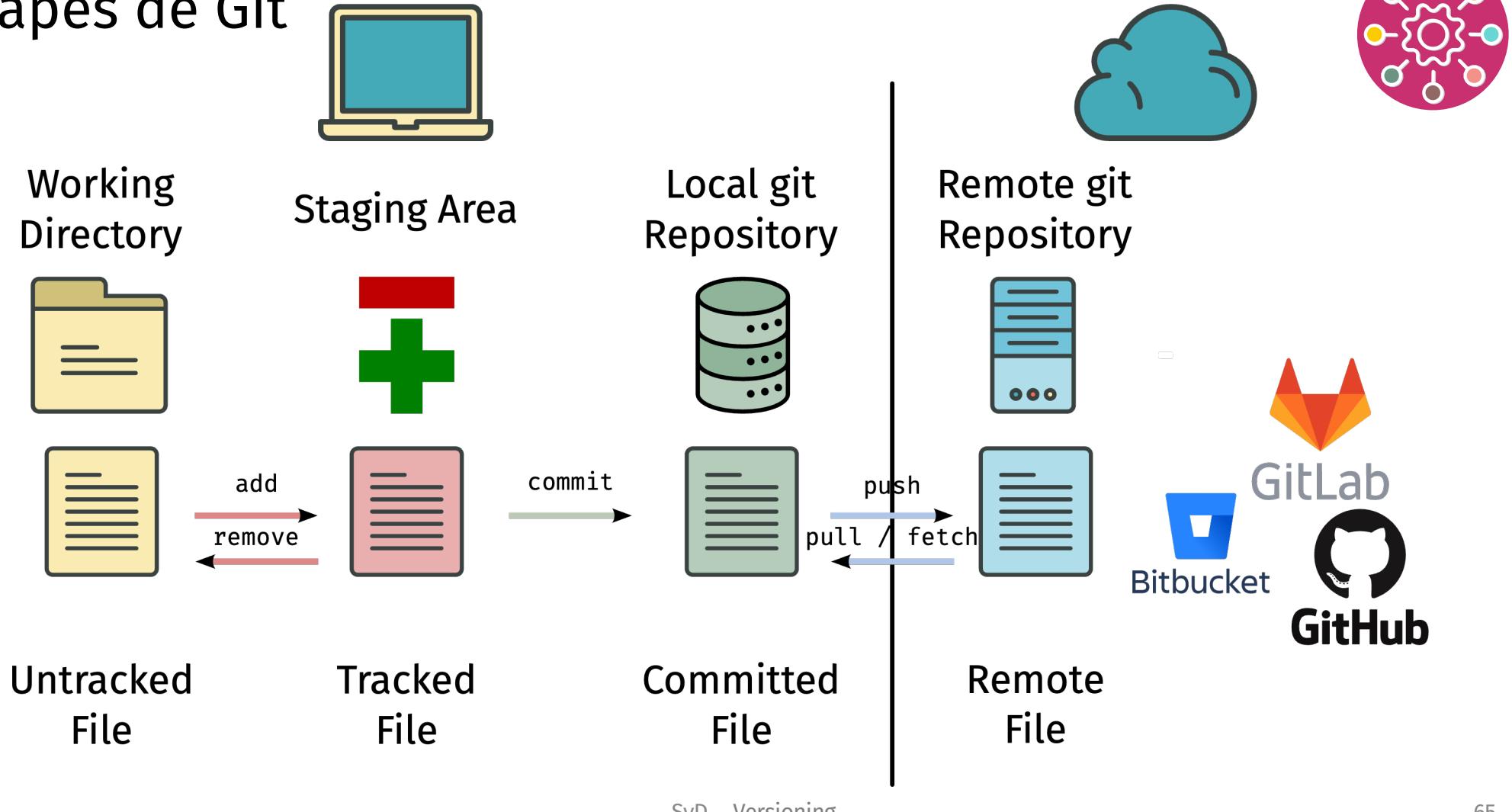
Subsection: Umsetzung:

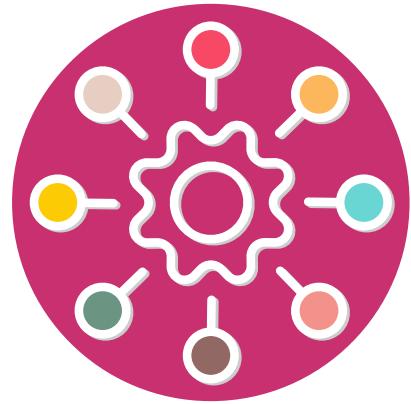
```
164 Le \textbf{mainDecoder} peut être écrit en VHDL. Pour cela, vous pouvez analyser
l'exemple de code \ref{fig:riscv-mainDecoder-code} ci-dessous et l'adapter en
conséquence.
165
166 \textbf{Note:} Dans HDL Designer, lorsque vous sélectionnez le type de contenu d'un
bloc, choisissez \textbf{VHDL File -> Architecture}, et contrôlez que le
langage soit défini sur \textbf{VHDL 2008}. Sur la page suivante, \textbf{Architecture}
correspond au nom de la vue (un bloc peut avoir différents
contenus) et \textbf{Entity} au nom du bloc (\textbf{mainDecoder} par exemple).}
167 }
168 \opt{d}{%
169 Schreiben Sie hierzu für beide Subblöcke, \textbf{mainDecoder} sowie \textbf{ALUDecoder}, eine Wahrheitstabelle für alle benötigten Instruktionen.
170 \opt{f}{\caption{figure}{Exemple de code MainDecoder}}
171 \opt{d}{\caption{figure}{MainDecoder Code-Beispiel}}
172 \label{fig:riscv-mainDecoder-code}
173
174 \subsubsection{ALU}
175 \opt{f}{%
176 L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:
177 }
178 \opt{d}{%
179 Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden
Tabelle:
180 }
181
182 \begin{table}[h]
```

g:riscv-mainDecoder-code:

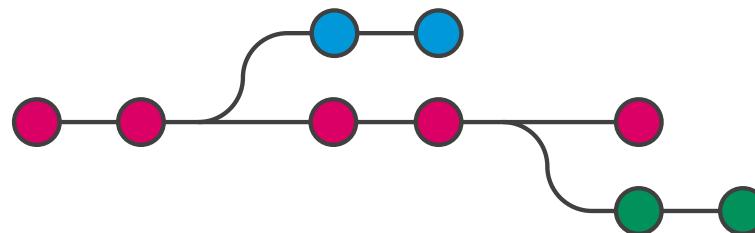
```
110 \opt{f}{\caption{figure}{Exemple de code MainDecoder}}
111 \opt{d}{\caption{figure}{MainDecoder Code-Beispiel}}
112 \label{fig:riscv-mainDecoder-code}
113
114 \subsubsection{ALU}
115 \opt{f}{%
116 L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:
117 }
118 \opt{d}{%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

Étapes de Git





Git Branch et Merge Exemple

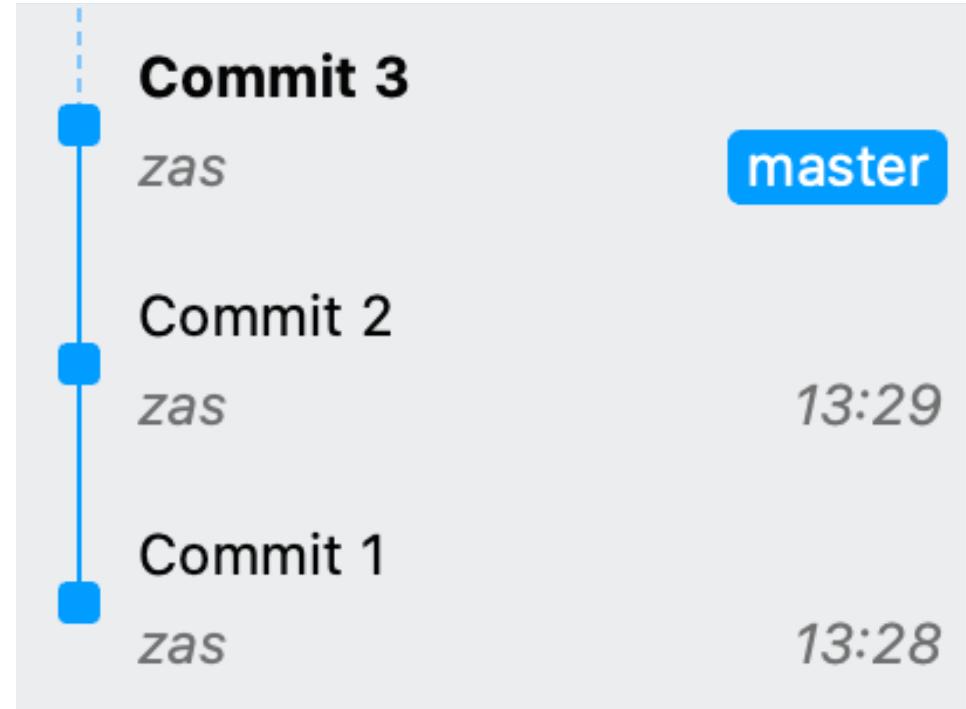




Branch et Merge

Initial repo state

Chaque repo a une branche **main** ou une branche **master** comme branche par défaut.

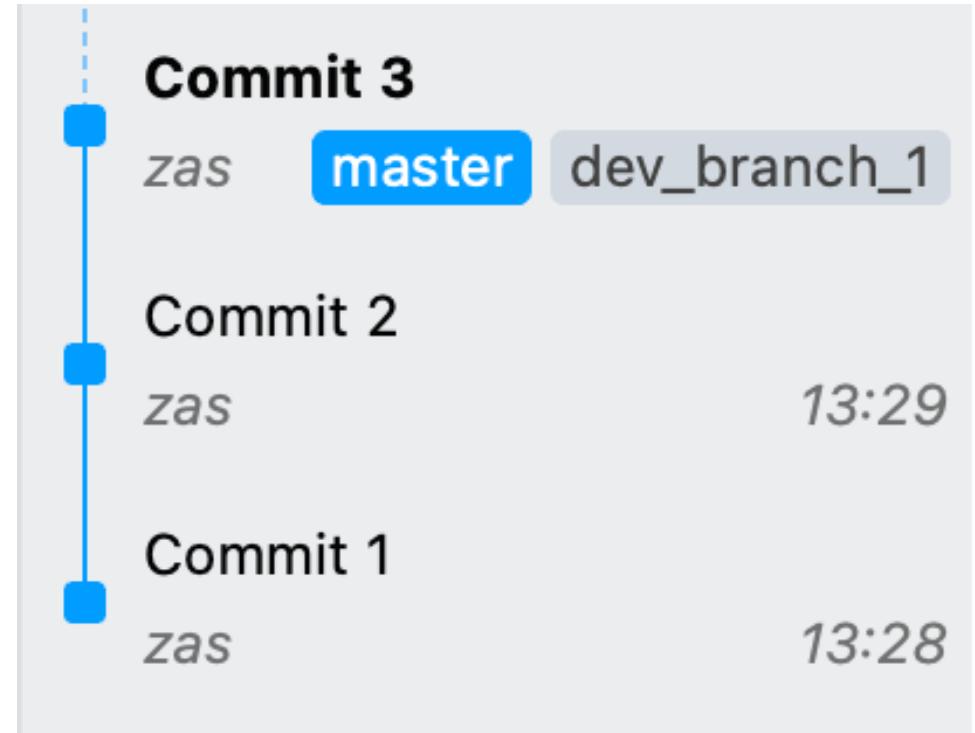




Branch et Merge

Create branch dev_branch_1

Create branch



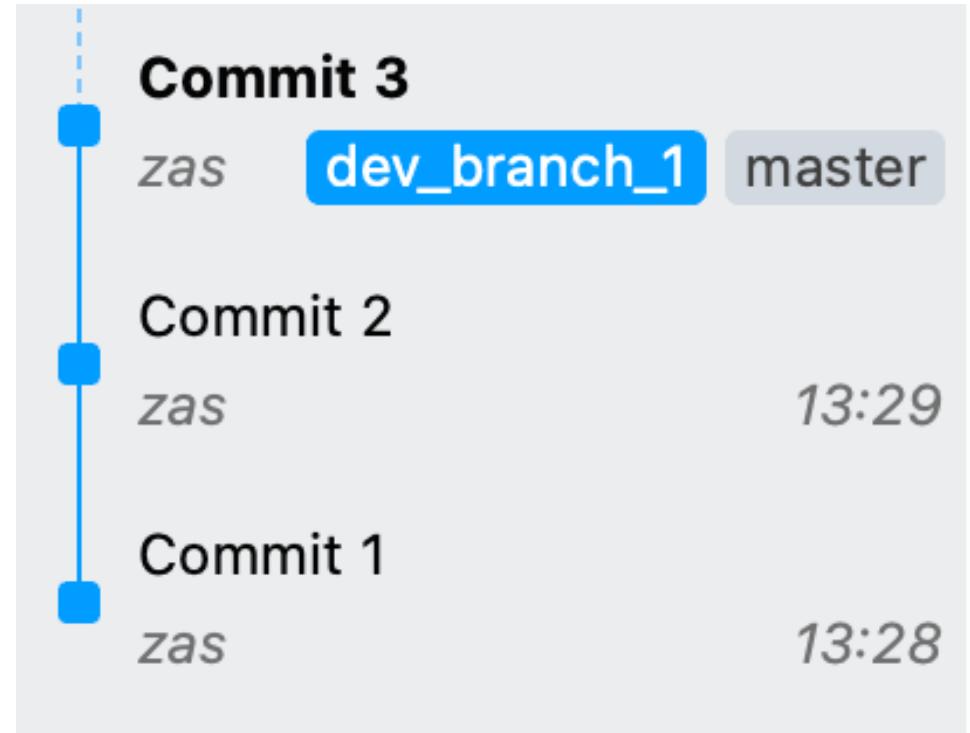


Branch et Merge

Checkout branch dev_branch_1

Vérifier sur quelle branche nous nous trouvons

Checkout branch



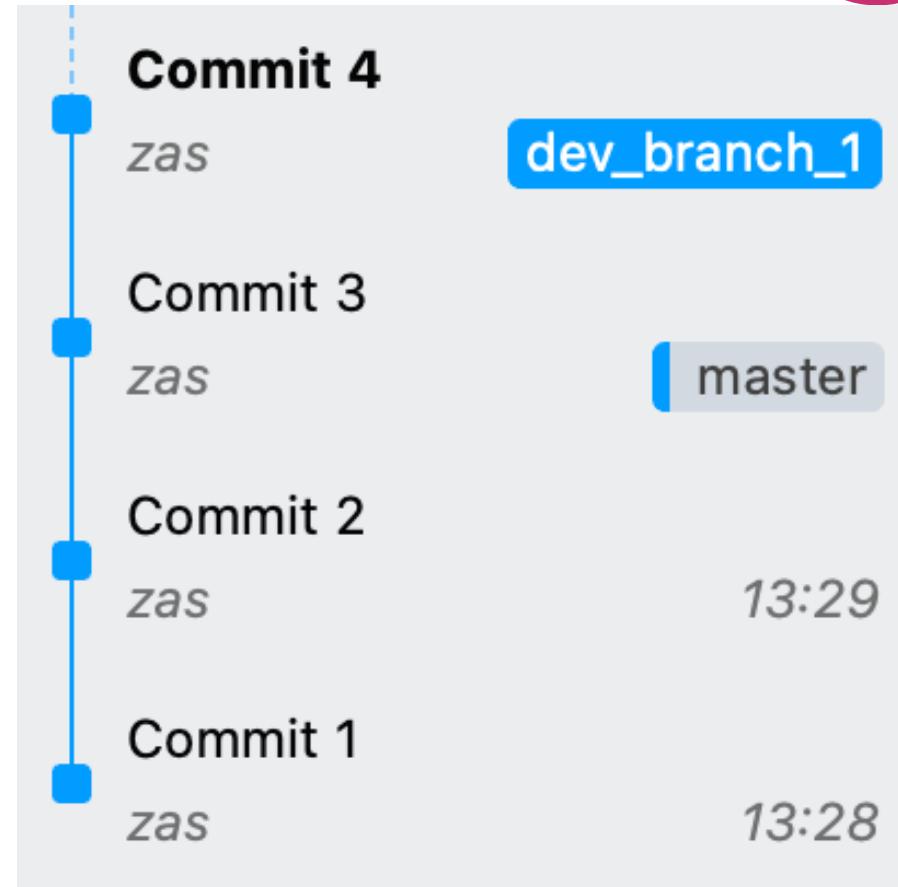


Branch et Merge

Commit on dev_branch_1

Stage new file

Commit stages files



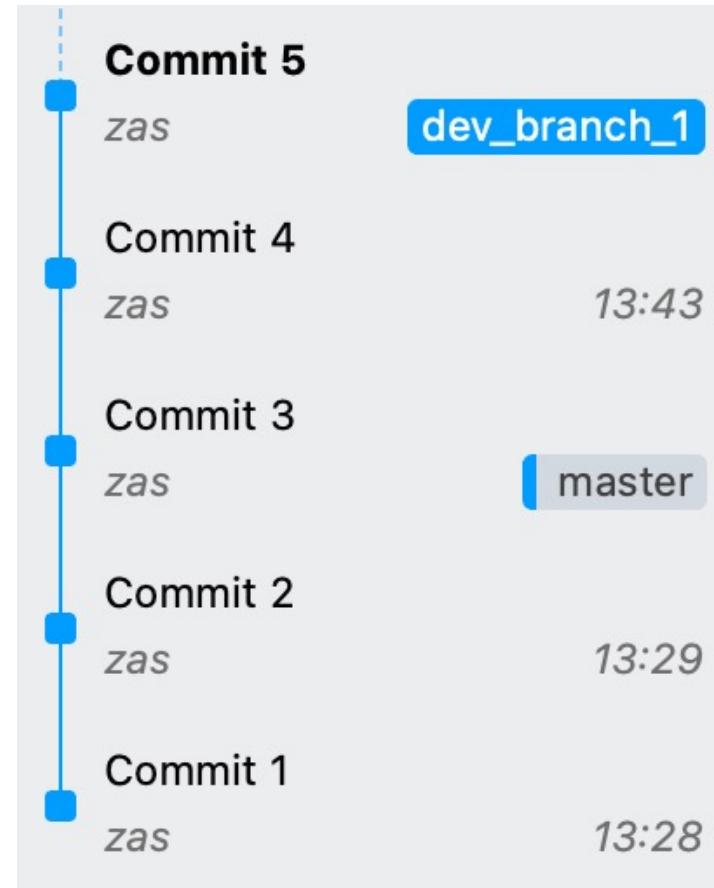


Branch et Merge

Commit on dev_branch_1

Stage new file

Commit stages files



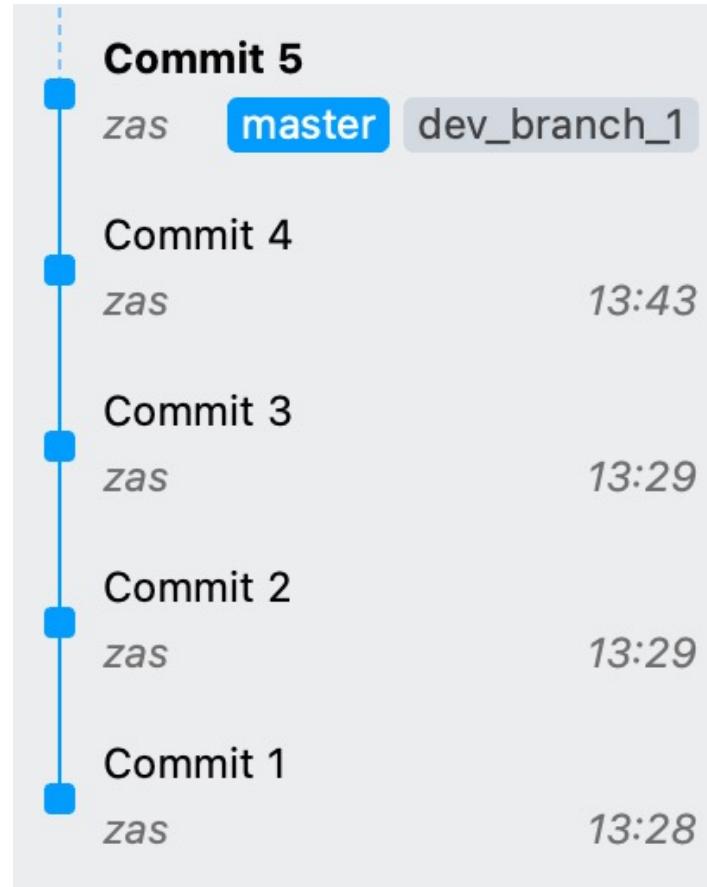


Branch et Merge

Merge master and dev_branch_1

Checkout master branch

Merge dev_branch_1 into master



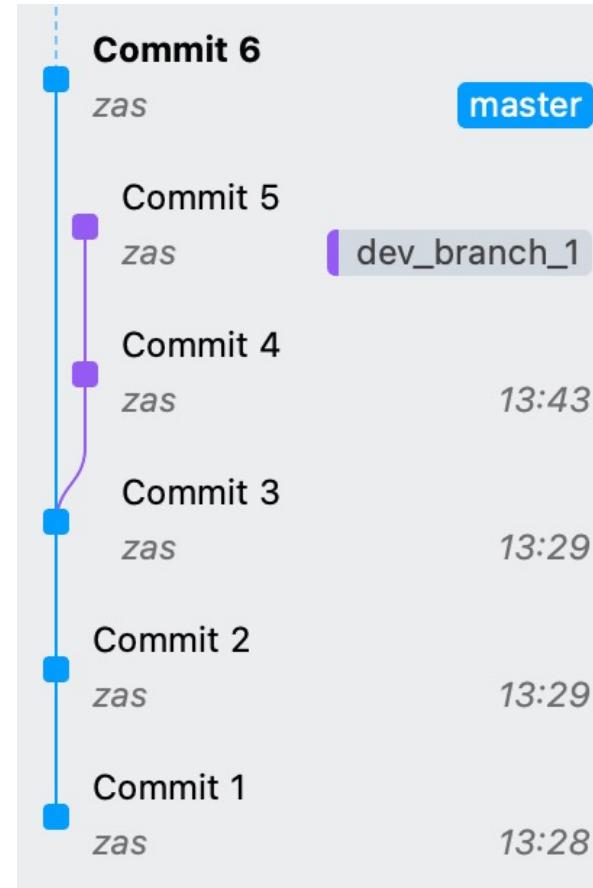


Branch et Merge

Commit on master branch

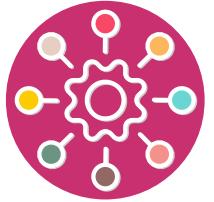
Stage new file

Commit stages files

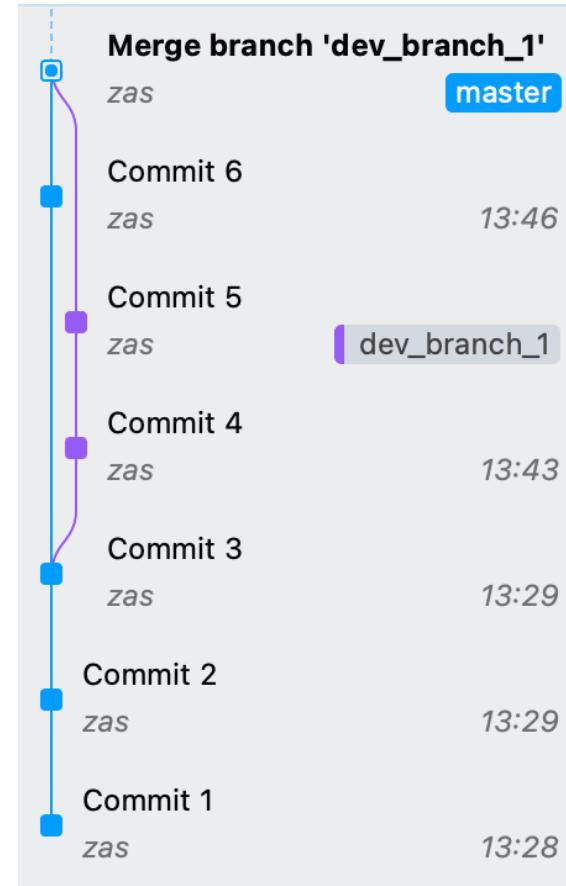


Branch et Merge

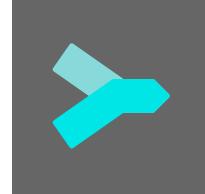
Three way merge master and dev_branch_1



Merge into



Demo



LICENSE UPGRADE REQUIRED

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

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 702c8ff178adbf6639884c48b72e4bc68361d13c f163cea8c5f992b7c7854993694b6870e0da8b

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:

```
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.

...Collapse all

Subsection: Umsetzung:

```
63
64 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.
65
66 \textbf{Note:} Dans HDL Designer, lorsque vous sélectionnez le type de contenu d'un
bloc, choisissez \textbf{VHDL File -> Architecture}, et contrôlez que le
langage soit défini sur \textbf{VHDL 2008}. Sur la page suivante, \textbf{Architecture}
correspond au nom de la vue (un bloc peut avoir différents
contenus) et \textbf{Entity} au nom du bloc (\textbf{mainDecoder} par exemple.)
```

Subsection: Umsetzung:

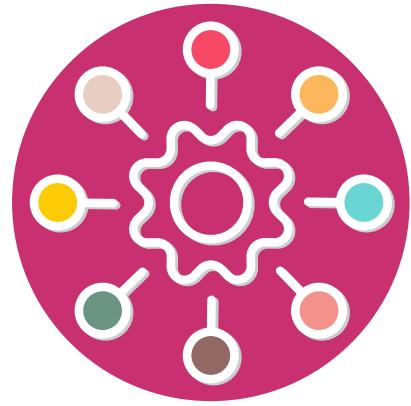
```
63
64 Le \textbf{mainDecoder} peut être écrit en VHDL. Pour cela, vous pouvez analyser
l'exemple de code \ref{fig:riscv-mainDecoder-code} ci-dessous et l'adapter en
conséquence.
65
66 \textbf{Note:} Dans HDL Designer, lorsque vous sélectionnez le type de contenu d'un
bloc, choisissez \textbf{VHDL File -> Architecture}, et contrôlez que le
langage soit défini sur \textbf{VHDL 2008}. Sur la page suivante, \textbf{Architecture}
correspond au nom de la vue (un bloc peut avoir différents
contenus) et \textbf{Entity} au nom du bloc (\textbf{mainDecoder} par exemple.)
```

67 }
68 \opt{d}{%
69 Schreiben Sie hierzu für beide Subblöcke, \textbf{mainDecoder} sowie \textbf{ALUDecoder}, eine Wahrheitstabelle für alle benötigten Instruktionen.

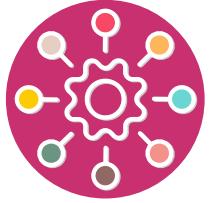
g:riscv-mainDecoder-code:

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

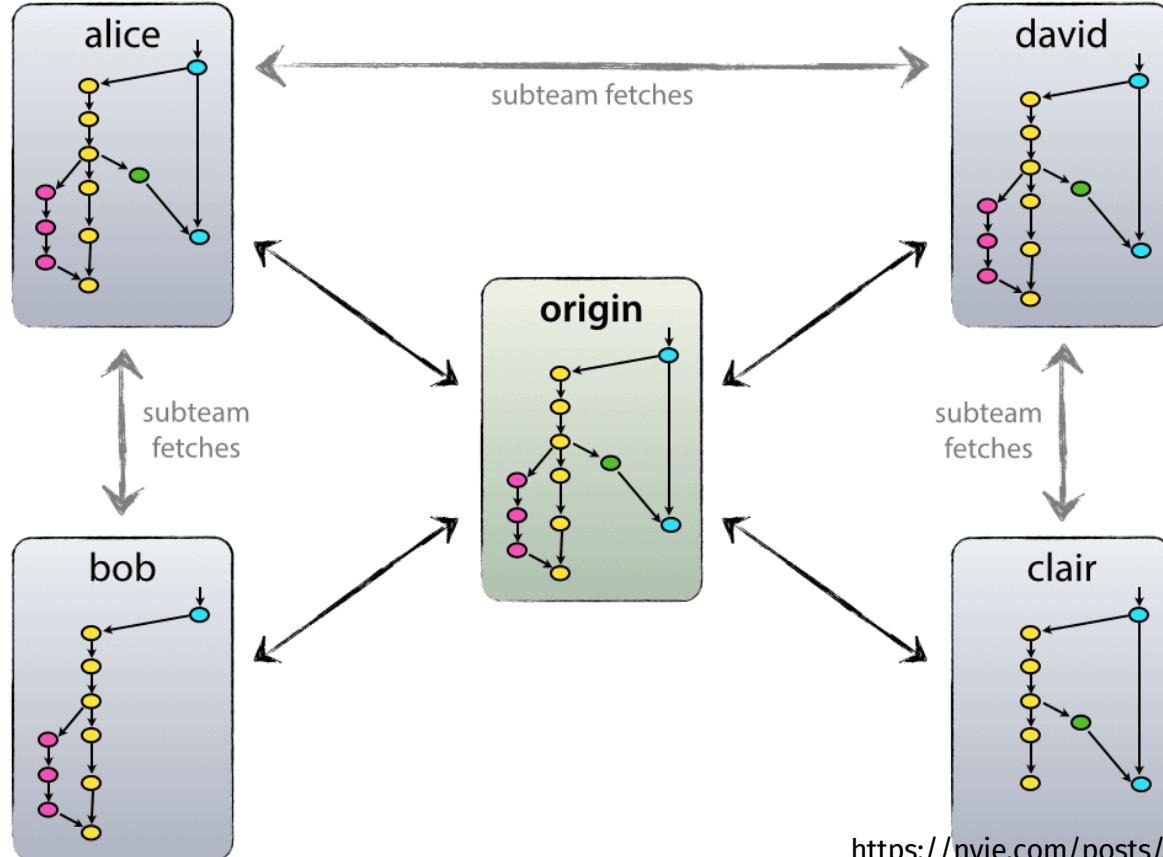
113
114 \subsubsection{ALU}
115 \opt{f}{%
116 L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:
117 }
118 \opt{d}{%
119 Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden
Tabelle:
120 }
121
122 \begin{table}[h]



Gitflow

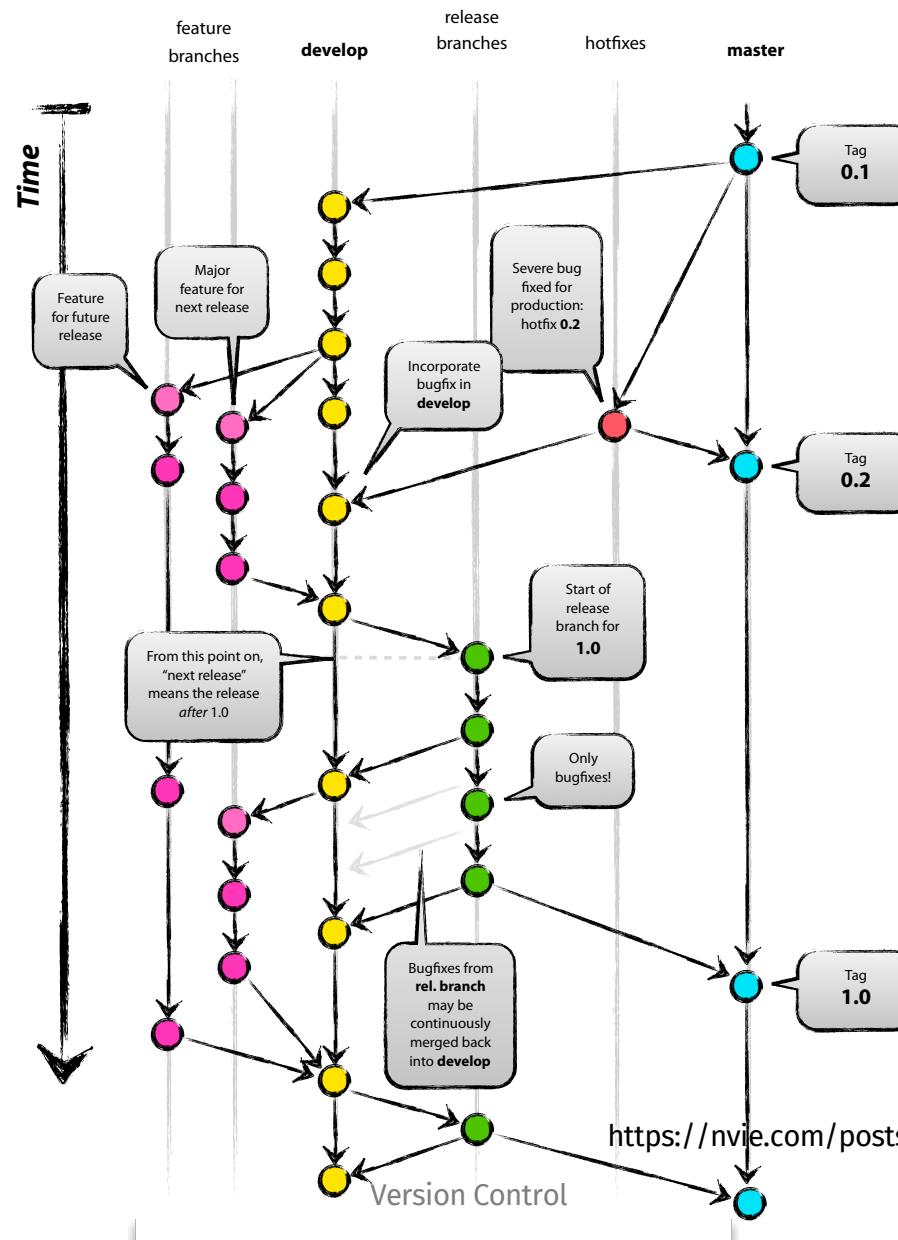


Gitflow Collaboration

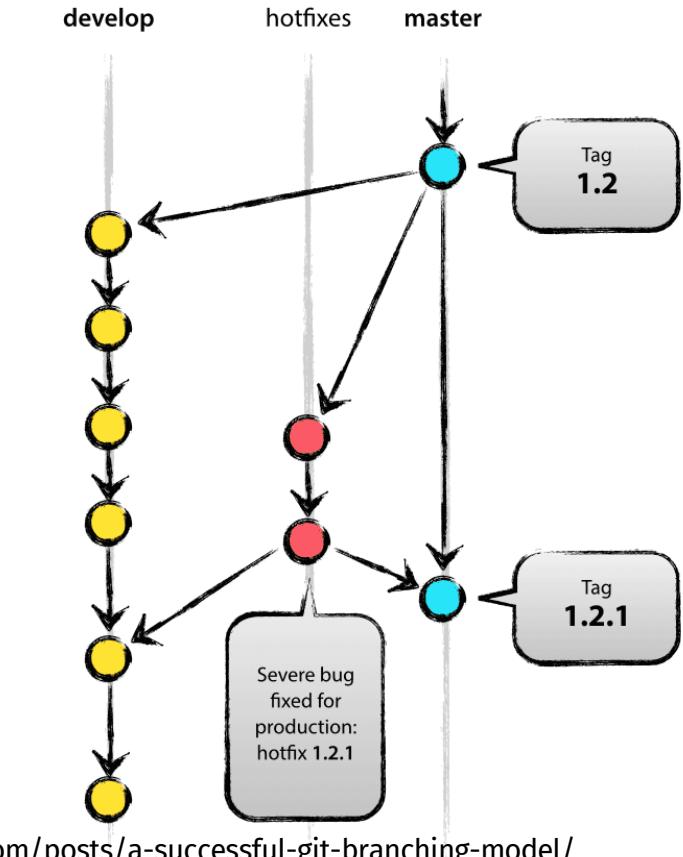
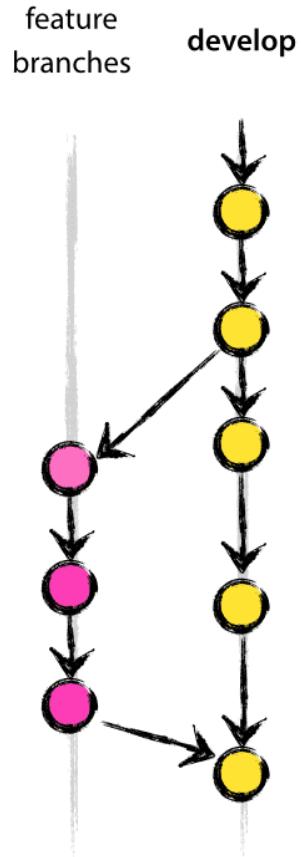
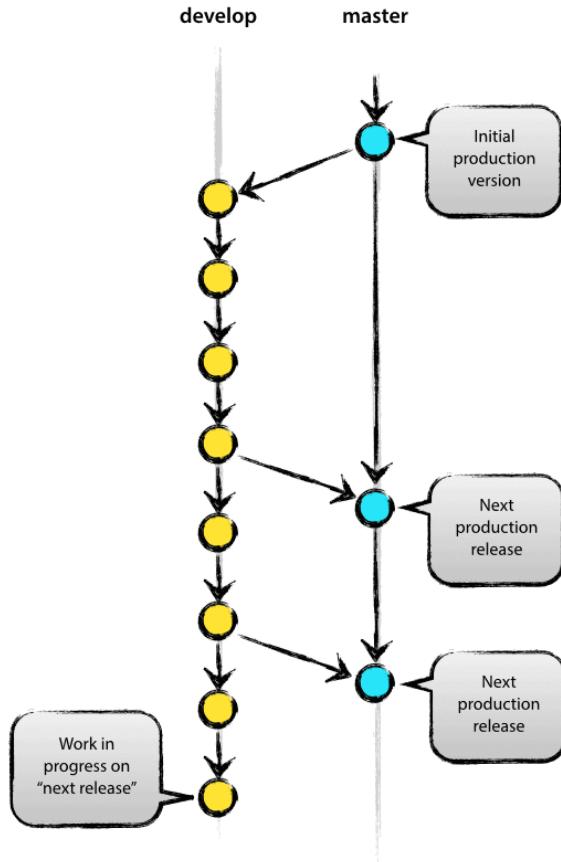


<https://nvie.com/posts/a-successful-git-branching-model/>

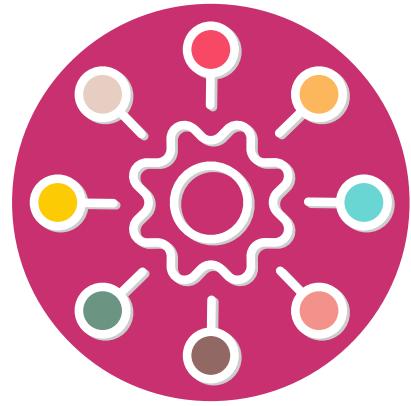
Gitflow



Gitflow Branching



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



Git CI/CD

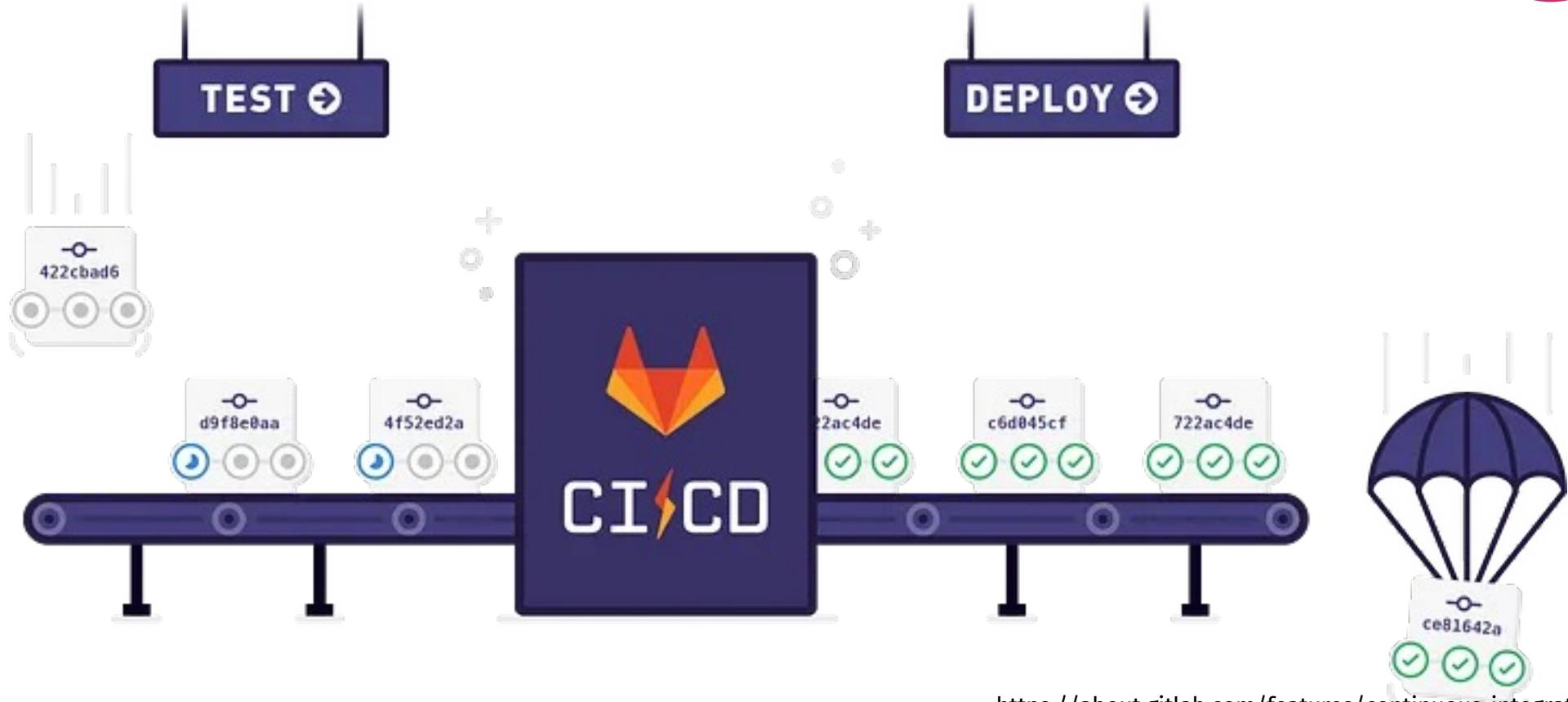
Qu'est-ce que la CI/CD ?



- L'intégration continue (Continuous Integration, CI) est la pratique qui consiste à intégrer fréquemment des modifications de code dans un référentiel partagé, qui est ensuite automatiquement construit et testé.
- La livraison continue (Continuous Delivery, CD) va encore plus loin en déployant automatiquement les modifications du code dans des environnements de type production afin de les tester et de les valider.
- L'automatisation des tests est un élément essentiel de la CI/CD, car elle permet de détecter les bogues et autres problèmes à un stade précoce du processus de développement.
- Les outils les plus courants sont GitLab CI/CD, Github Actions, Jenkins, CircleCI et Travis CI.

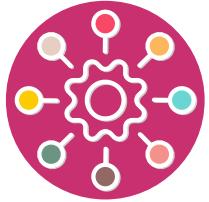


Qu'est-ce que la CI/CD ?



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

Gitlab Workflow



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

