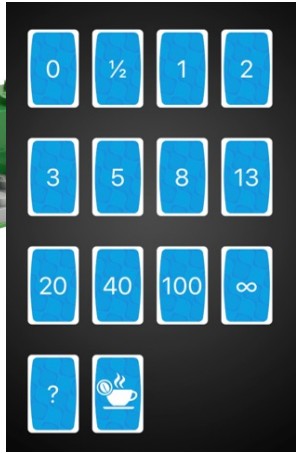
Estimation des tâches

Estimation 2

Créer un petit maison

X Point(s)

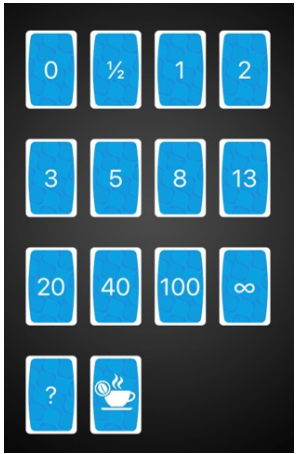


Estimation des tâches

Estimation 3

Créer un programable robot

X Point(s)

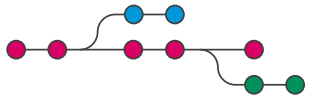


Hes·so VALAIS WALLIS

Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Conception des systèmes

Version Control



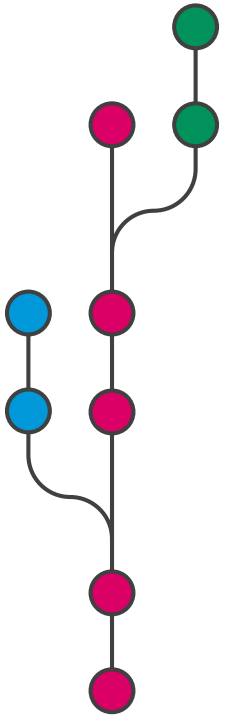
Filière Systèmes industriels

Silvan Zahno silvan.zahno@hevs.ch

Votre système actuel

```
.
├── fichier important copy.txt
├── fichier important v1.txt
├── fichier important v2.1 backup.txt
├── fichier important v2.1 backup.txt.old
├── fichier important v2.1.txt
├── fichier important v2.txt
└── fichier important.txt

1 répertoire, 7 fichiers
```

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>

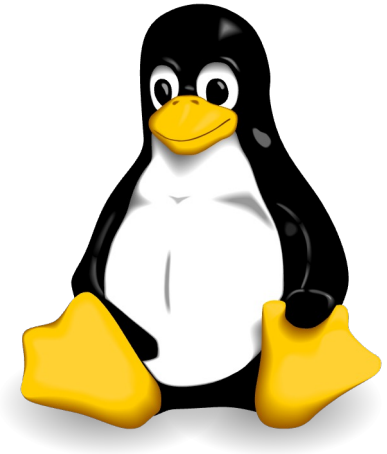
Outils possibles

Git (git)

Subversion (svn)

Mercurial (hg)

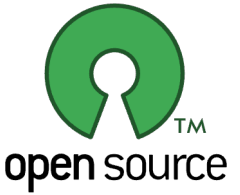
Linux Torvalds



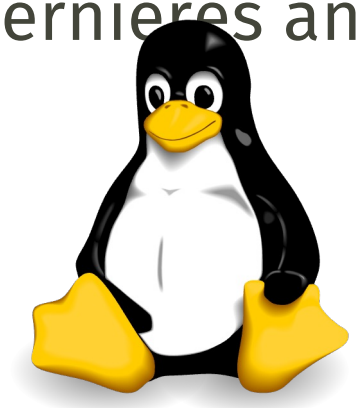
Linux (1991)

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/

Plateformes Git

Gitlab

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

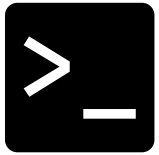
Github

https://github.com

Bitbucket

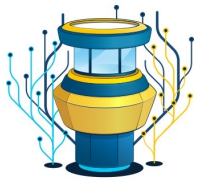
https://bitbucket.com

Outils Git

Ligne de commande 

Fork

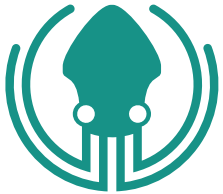
Sublime Merge 



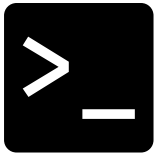
Tower

Git Cola 

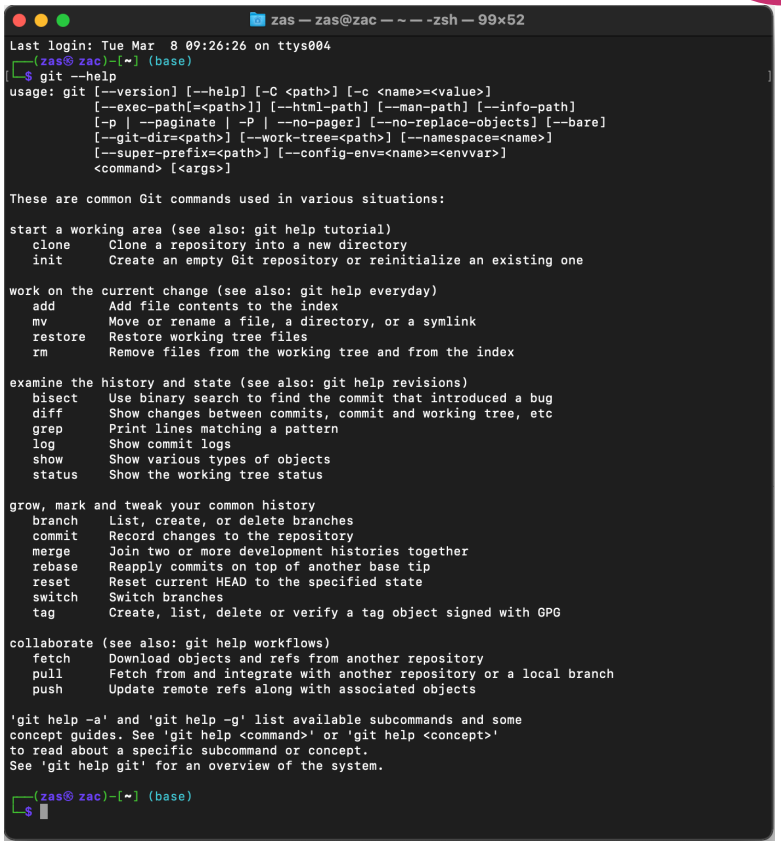
SmartGit 

Git Kraken 

Git ligne de commande



```
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
```

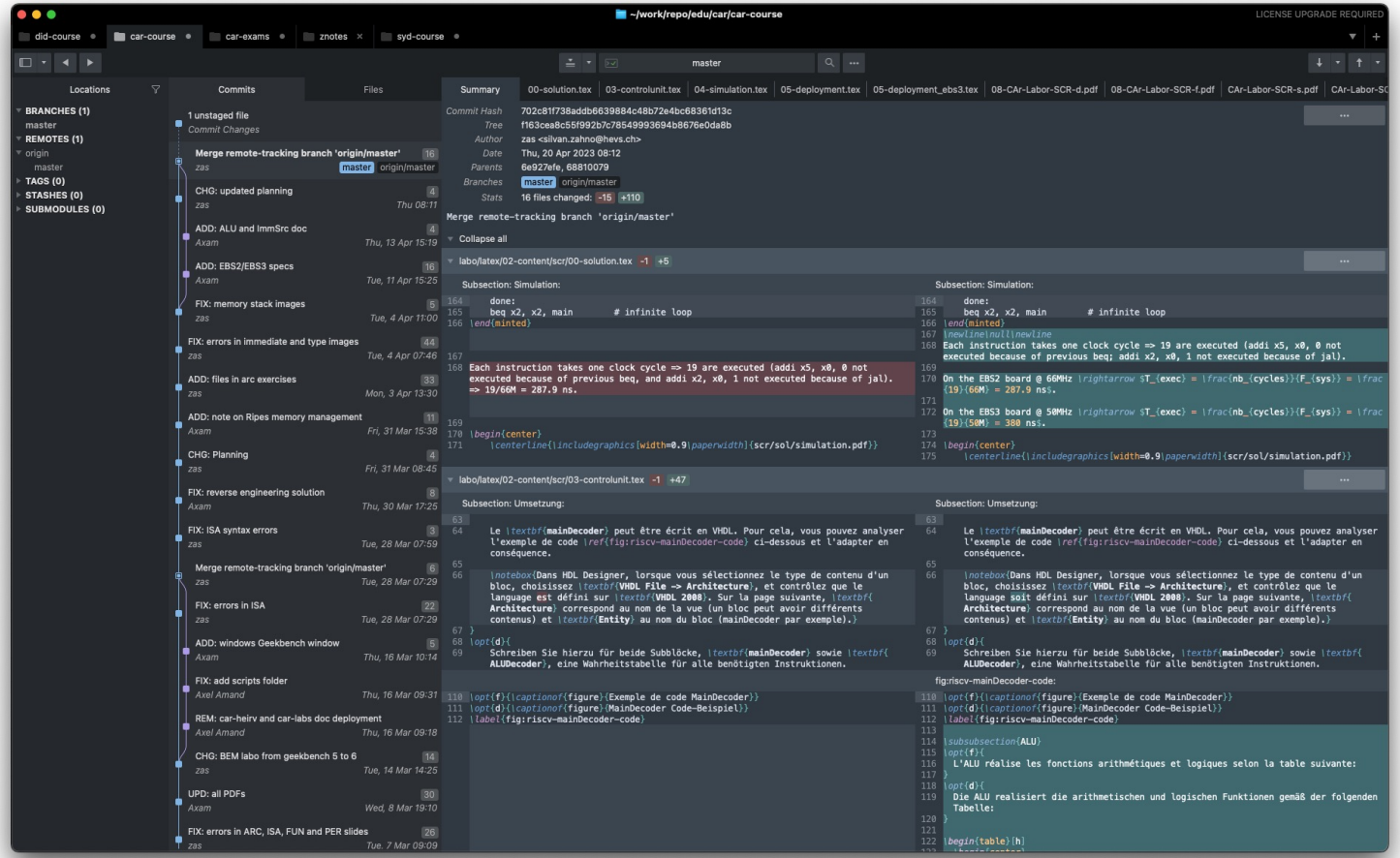


git commands

Command	Description
Démarrer une zone de travail	
`clone`	Cloner un dépôt dans un nouveau répertoire
`init`	Créer un dépôt Git vide ou réinitialiser un dépôt existant
Travailler sur la modification en cours	
`add`	Ajouter le contenu d'un fichier à l'index
`mv`	Déplacer ou renommer un fichier, un répertoire ou un lien symbolique
`reset`	Réinitialiser le HEAD actuel à l'état spécifié
`rm`	Supprimer des fichiers de l'arbre de travail et de l'index
Examiner l'historique et l'état	
`log`	Afficher les journaux de livraison
`show`	Afficher différents types d'objets
`status`	Afficher l'état de l'arbre de travail

Command	Description
Développez, marquez et modifiez votre histoire commune	
`branch`	Lister, créer ou supprimer des branches
`checkout`	Changer de branche ou restaurer les fichiers de l'arborescence de travail
`commit`	Enregistrer les modifications apportées à la base de données
`diff`	Afficher les changements entre les livraisons, les livraisons et l'arbre de travail, etc.
`merge`	Joindre deux ou plusieurs historiques de développement
`rebase`	Réappliquer les commits par-dessus un autre conseil de base
`tag`	Créer, lister, supprimer ou vérifier un objet d'étiquette
Collaborer	
`fetch`	Télécharger des objets et des références à partir d'un autre dépôt
`pull`	Récupérer et intégrer des objets d'une autre base de données ou d'une branche locale
`push`	Mettre à jour les références distantes ainsi que les objets associés

Sublime Merge



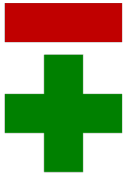
Étapes de Git

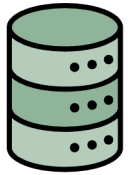
Working Directory

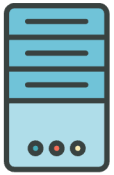
Staging Area

Local git Repository

Remote git Repository







add

remove

commit

push

pull / fetch

GitLab

Bitbucket

GitHub

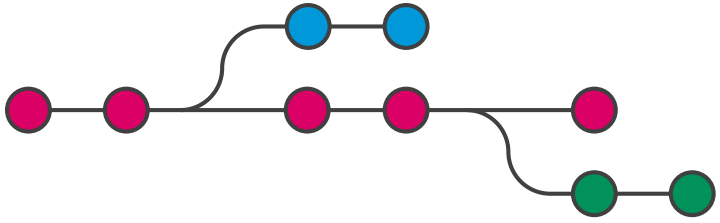
Untracked File

Tracked File

Committed File

Remote File

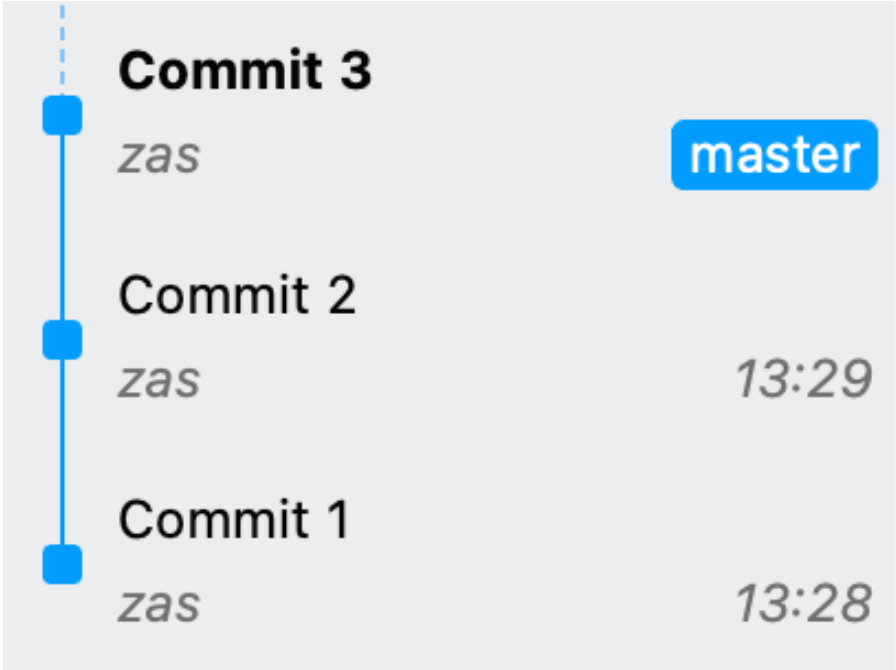
Git Branch et Merge Exemple



Branch et Merge

Initial repo state

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

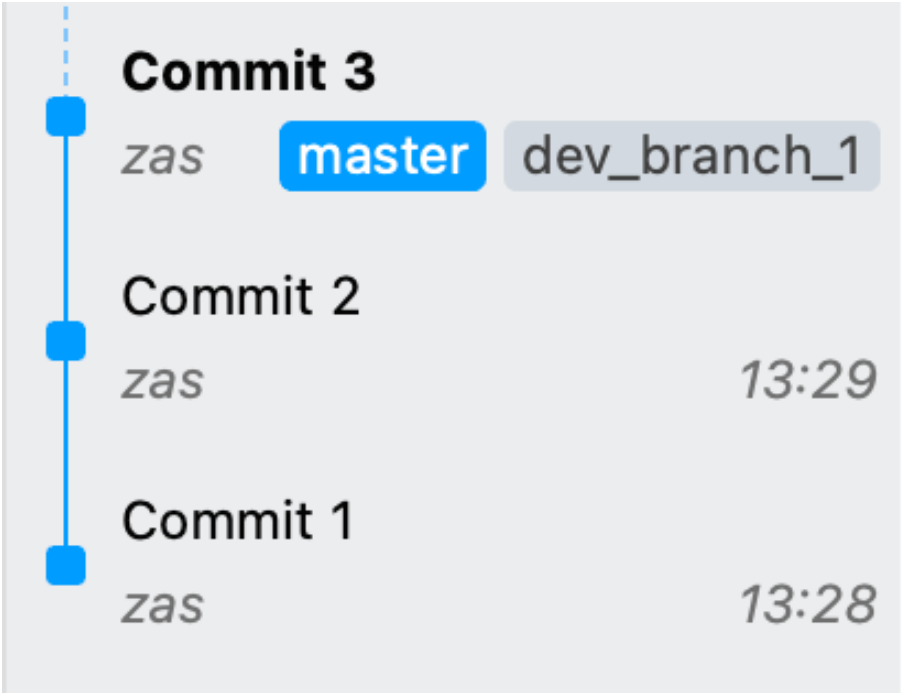


Branch et Merge

Create branch dev_branch_1

Create branch dev_branch_1

```
$ git branch dev_branch_1
```



Branch et Merge

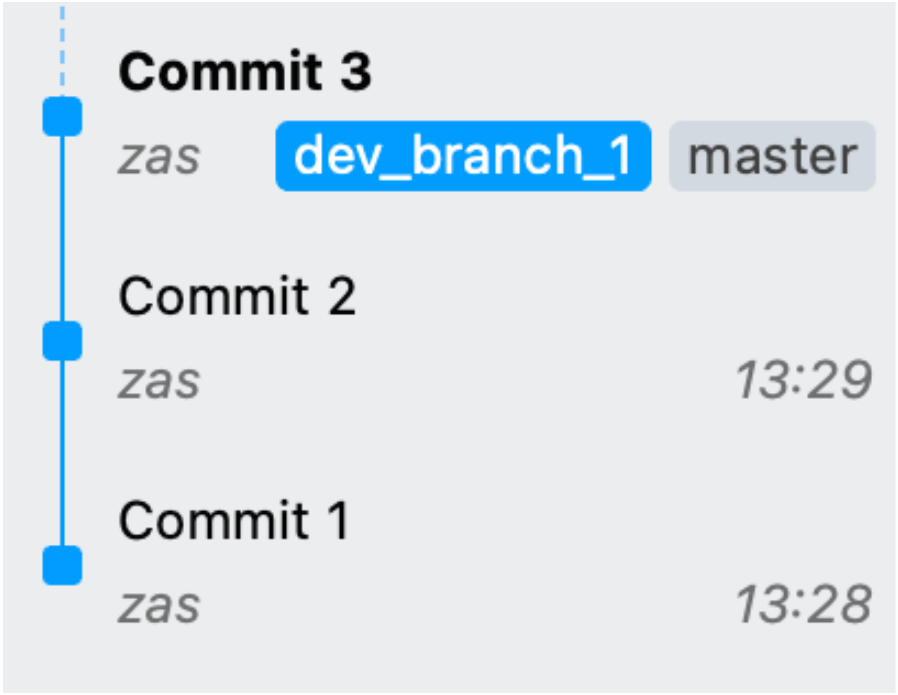
Checkout branch dev_branch_1

Vérifier sur quelle branche nous nous trouvons

```
$ git branch
```

Checkout branch dev_branch_1

```
$ git checkout dev_branch_1
```



Branch et Merge

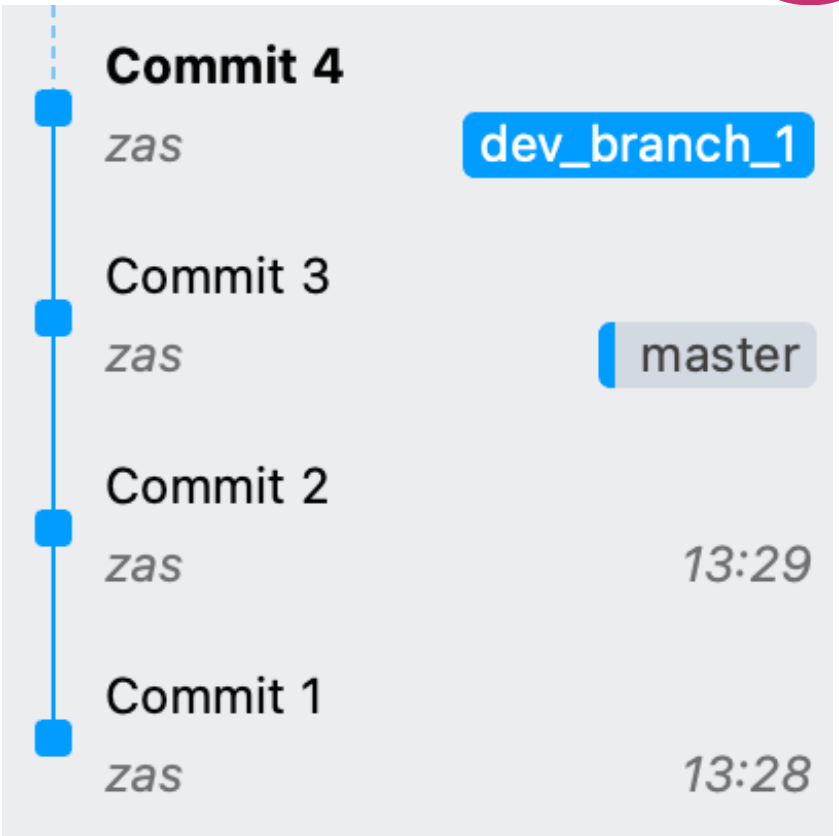
Commit on dev_branch_1

Stage new file

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

Commit stages files

```
$ git commit –m “Commit 4
”
```



Branch et Merge

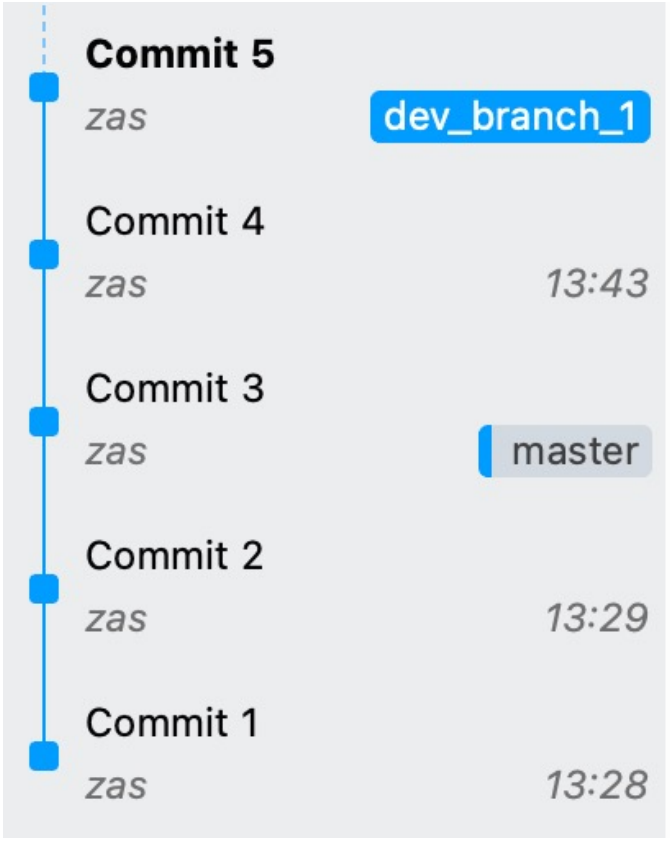
Commit on dev_branch_1

Stage new file

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

Commit stages files

```
$ git commit –m “Commit 5”
```



Branch et Merge

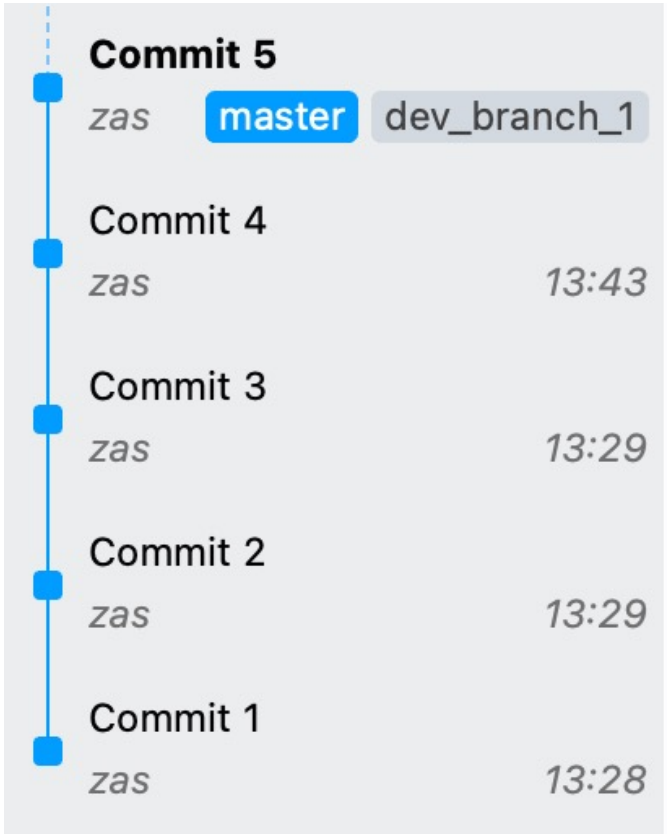
Merge `master` and `dev_branch_1`

Checkout `master` branch

```
$ git checkout master
```

Merge `dev_branch_1` into `master`

```
$ git merge dev_branch_1
```



Branch et Merge

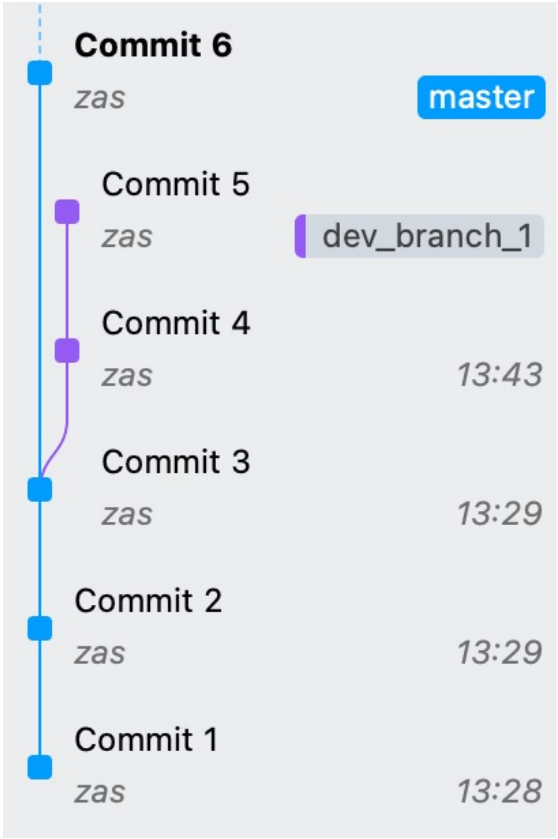
Commit on `master` branch

Stage new file

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

Commit stages files

```
$ git commit –m “Commit 6”
```

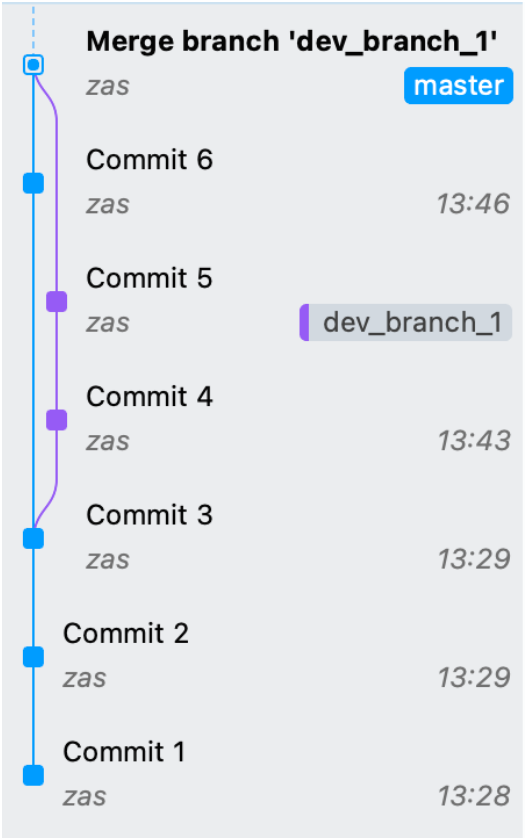


Branch et Merge

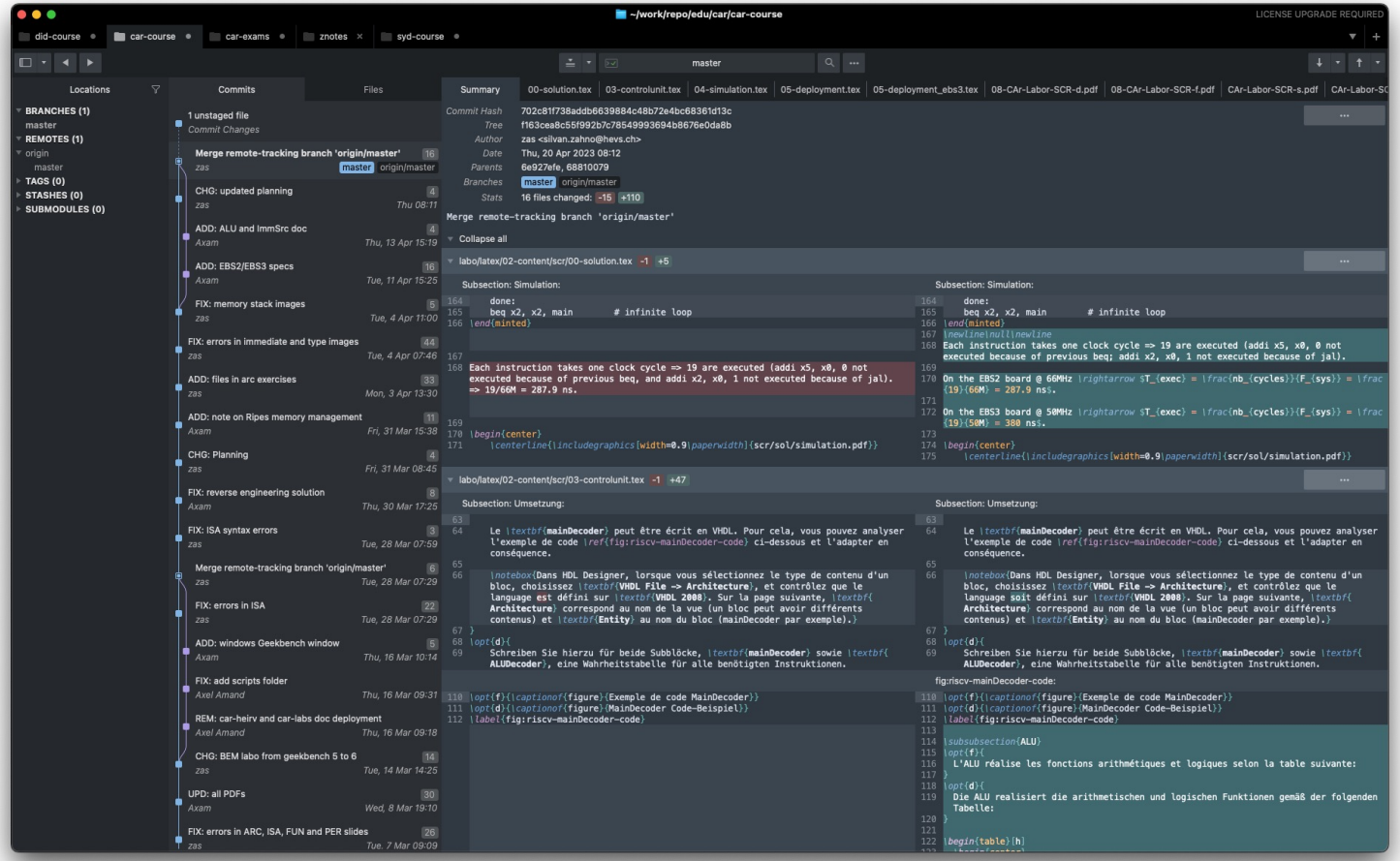
Three way merge `master` and `dev_branch_1`

Merge `dev_branch_1` into `master`

```
$ git merge dev_branch_1
```

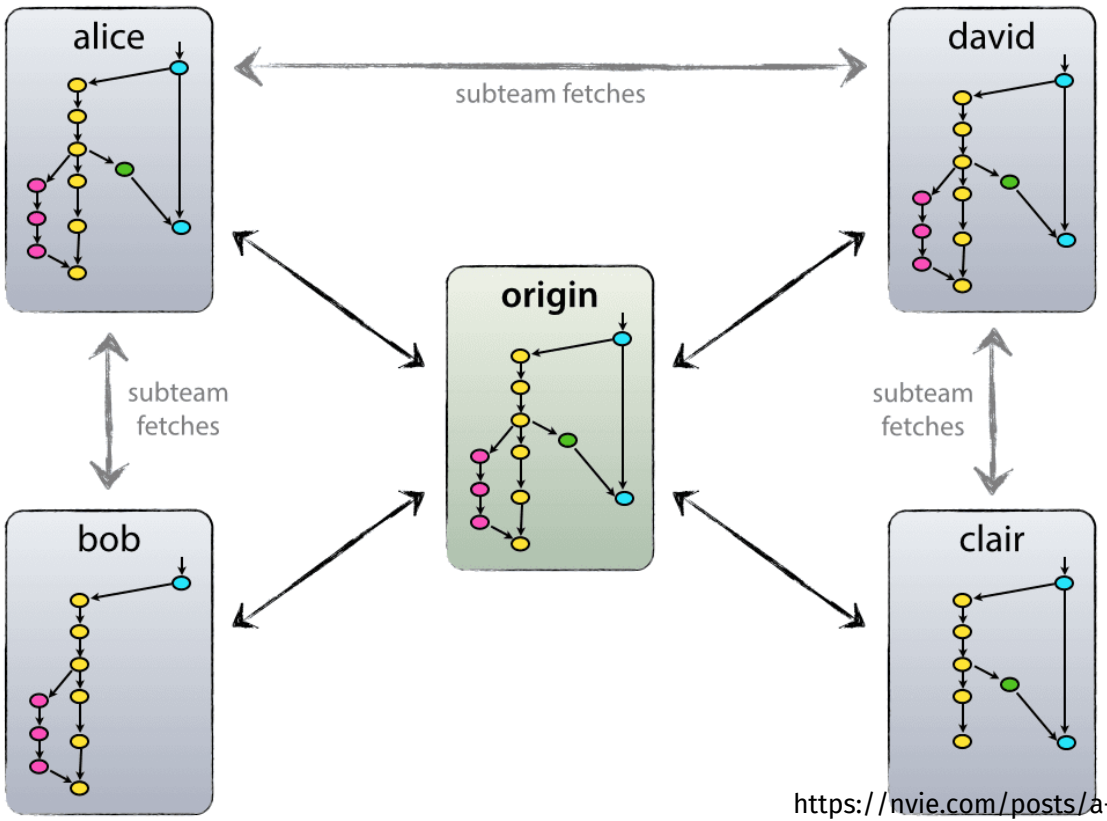


Demo


~/work/repo/edu/car/car-course
LICENSE UPGRADE REQUIRED
did-course
car-course
car-exams
znotes
syd-course
master
Locations
Commits
Files
BRANCHES (1)
master
REMOTES (1)
origin
master
TAGS (0)
STASHES (0)
SUBMODULES (0)
1 unstaged file
Commit Changes
Merge remote-tracking branch 'origin/master'
zas
master
origin/master
CHG: updated planning
zas
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'
zas
Tue, 28 Mar 07:29
FIX: errors in ISA
zas
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-heirv and car-labs doc deployment
Axel Amand
Thu, 16 Mar 09:18
CHG: BEM labo from geekbench 5 to 6
zas
Tue, 14 Mar 14:25
UPD: all PDFs
Axam
Wed, 8 Mar 19:10
FIX: errors in ARC, ISA, FUN and PER slides
zas
Tue, 7 Mar 09:09
Summary
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
Commit Hash 702c81f738addb6639884c48b72e4bc68361d13c
Tree f163cea8c55f992b7c785499936948676e0da8b
Author zas <silvan.zahno@hevs.ch>
Date Thu, 20 Apr 2023 08:12
Parents 6e927efe, 68810079
Branches master origin/master
Stats 16 files changed: -15 +110
Merge remote-tracking branch 'origin/master'
Collapse all
labo/latex/02-content/scr/00-solution.tex -1 +5
Subsection: Simulation:
done:
beq 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, and addi x2, x0, 1 not executed because of jal).
=> 19/66M = 287.9 ns.
\begin{center}
\centerline{\includegraphics[width=0.9\paperwidth]{scr/sol/simulation.pdf}}
Subsection: Simulation:
done:
beq x2, x2, main # infinite loop
\end{minted}
\newline\null\newline
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).
On the EBS2 board @ 66MHz \rightarrow $T_{exec} = \frac{nb_{cycles}}{F_{sys}} = \frac{19}{66M} = 287.9 ns$.
On the EBS3 board @ 50MHz \rightarrow $T_{exec} = \frac{nb_{cycles}}{F_{sys}} = \frac{19}{50M} = 380 ns$.
\begin{center}
\centerline{\includegraphics[width=0.9\paperwidth]{scr/sol/simulation.pdf}}
labo/latex/02-content/scr/03-controlunit.tex -1 +47
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.
\notebox{Dans HDL Designer, lorsque vous sélectionnez le type de contenu d'un bloc, choisissez \textbf{VHDL File -> Architecture}, et contrôlez que le language est 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 (mainDecoder par exemple).}
}
\opt{d}{
Schreiben Sie hierzu für beide Subblöcke, \textbf{mainDecoder} sowie \textbf{ALUDecoder}, eine Wahrheitstabelle für alle benötigten Instruktionen.
\opt{f}{\captionof{figure}{Exemple de code MainDecoder}}
\opt{d}{\captionof{figure}{MainDecoder Code-Beispiel}}
\label{fig:riscv-mainDecoder-code}
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.
\notebox{Dans HDL Designer, lorsque vous sélectionnez le type de contenu d'un bloc, choisissez \textbf{VHDL File -> Architecture}, et contrôlez que le language 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 (mainDecoder par exemple).}
}
\opt{d}{
Schreiben Sie hierzu für beide Subblöcke, \textbf{mainDecoder} sowie \textbf{ALUDecoder}, eine Wahrheitstabelle für alle benötigten Instruktionen.
fig:riscv-mainDecoder-code:
\opt{f}{\captionof{figure}{Exemple de code MainDecoder}}
\opt{d}{\captionof{figure}{MainDecoder Code-Beispiel}}
\label{fig:riscv-mainDecoder-code}
\subsubsection{ALU}
\opt{f}{
L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:
}
\opt{d}{
Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden Tabelle:
}
\begin{table}[h]

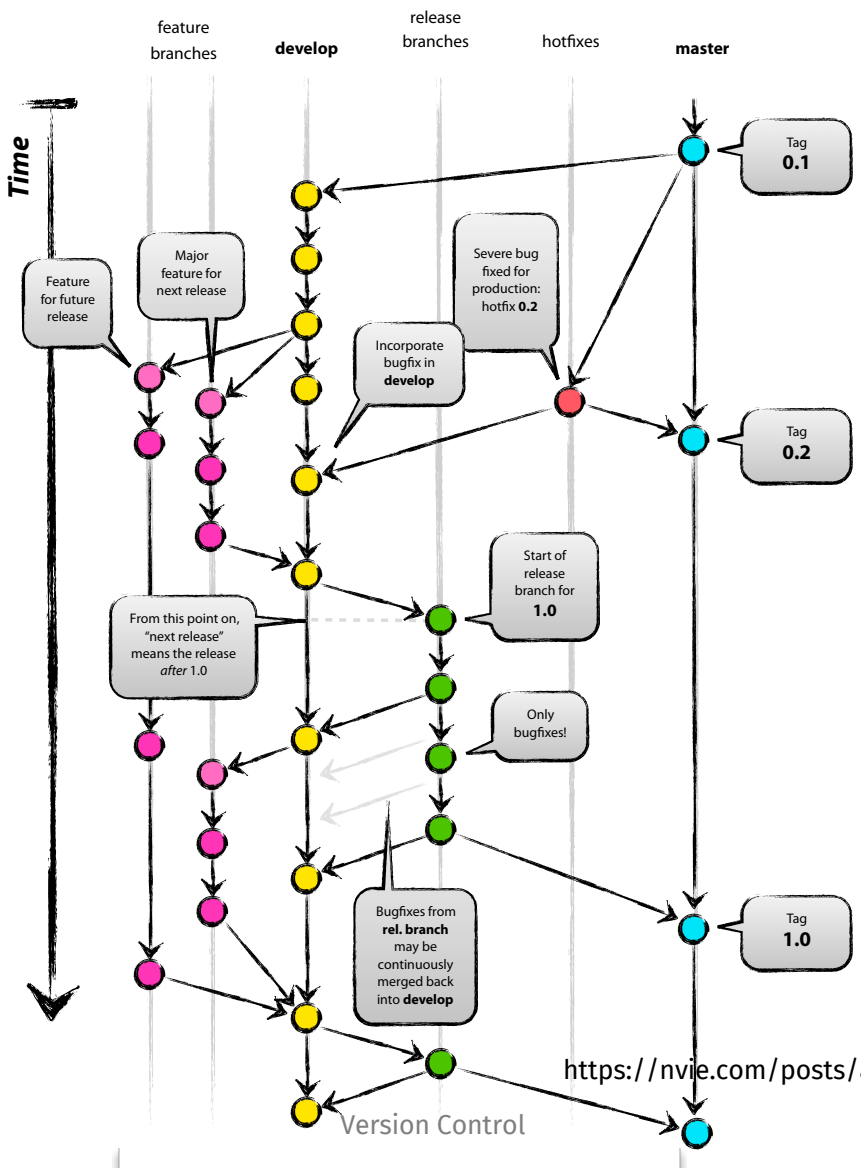
Gitflow

Gitflow Collaboration



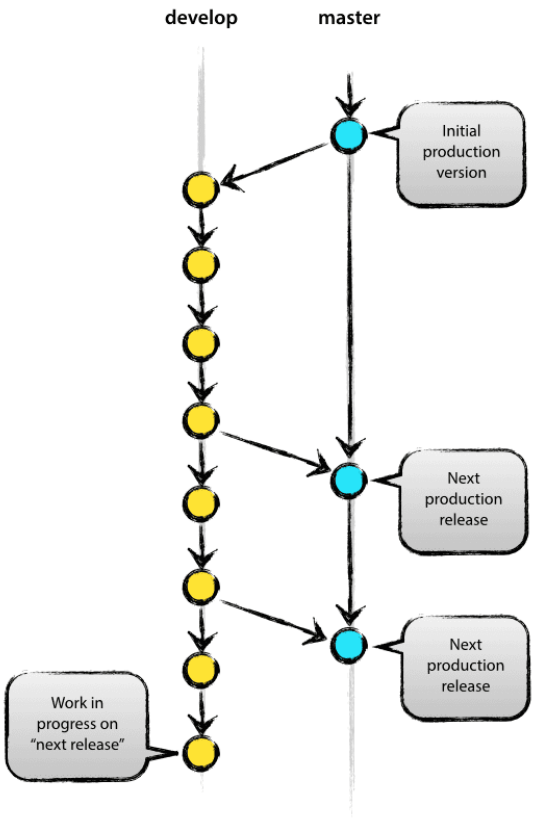
https://nvie.com/posts/a-successful-git-branching-model/

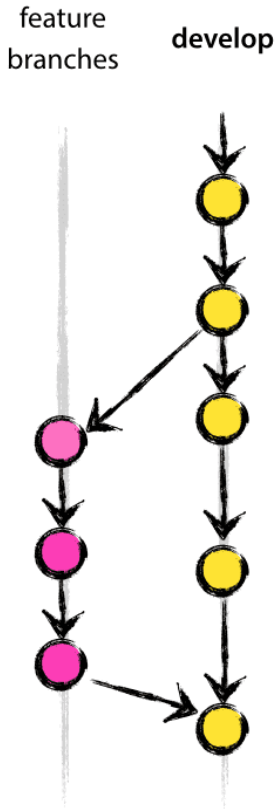
Gitflow

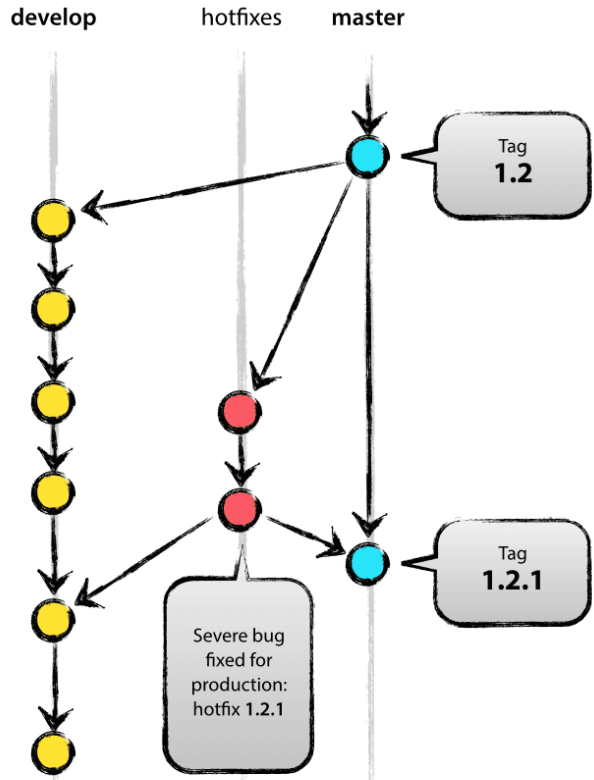


https://nvie.com/posts/a-successful-git-branching-model/

Gitflow Branching







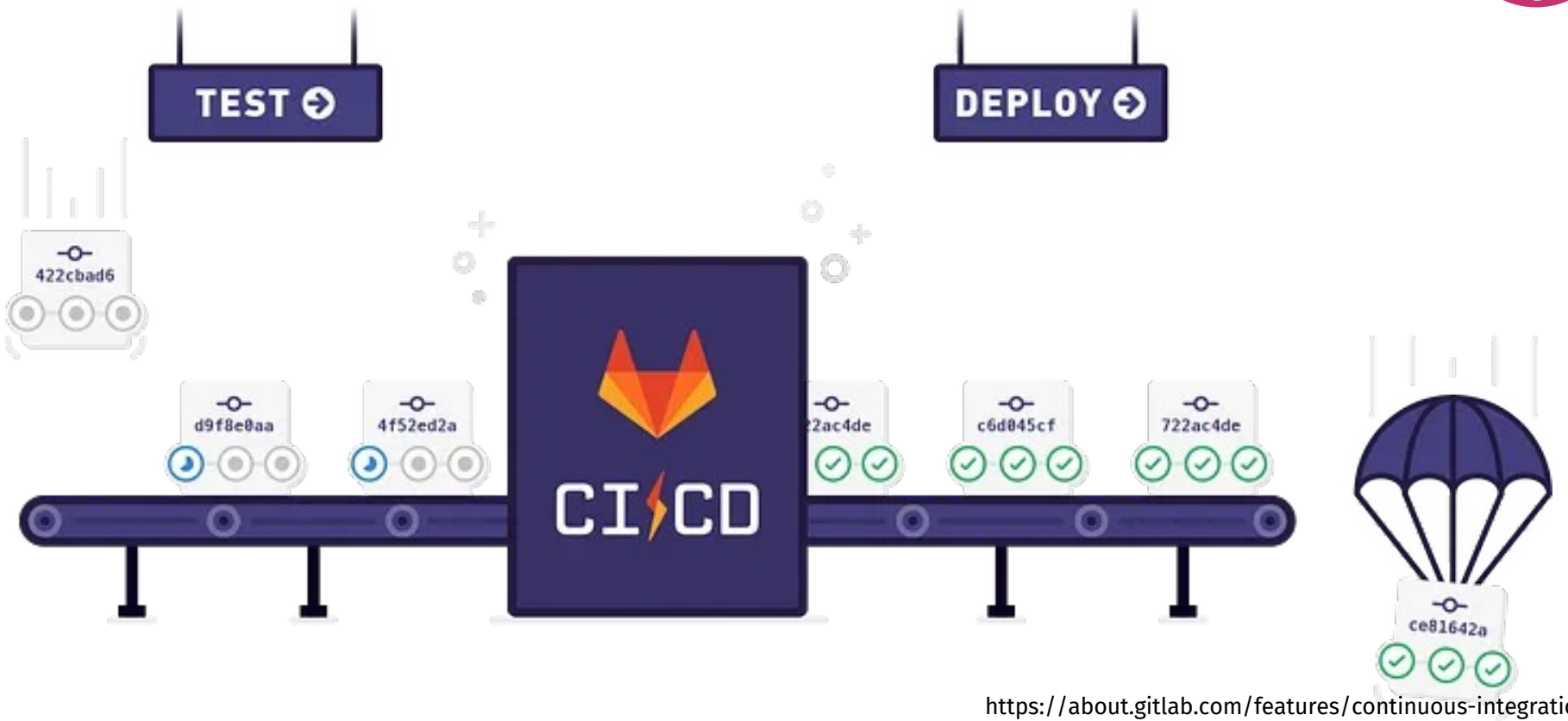
https://nvie.com/posts/a-successful-git-branching-model/

Git CI/CD

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/