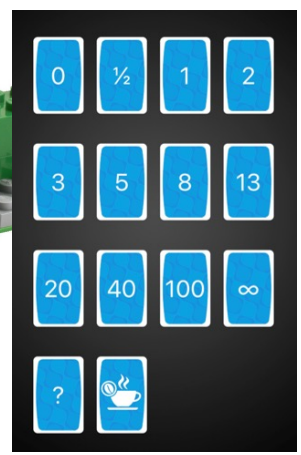
Task Estimation

Estimation 2

Create a small house

X Point(s)

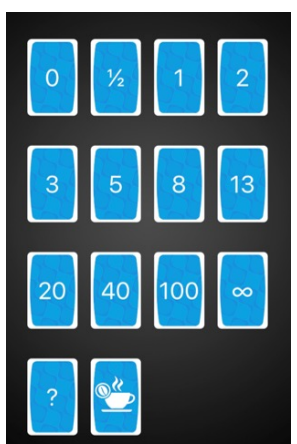


Task Estimation

Estimation 3

Create a programmable Lego robot

X Point(s)



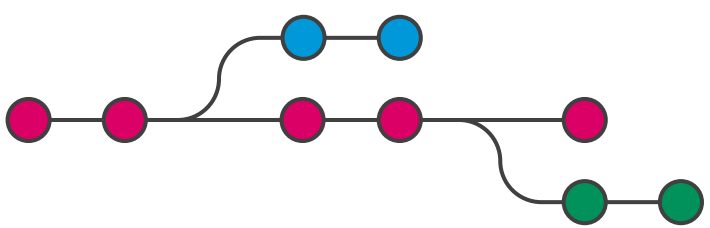
Hes·so VALAIS WALLIS

School of Engineering

System Design

Version Control

Systems Engineering program



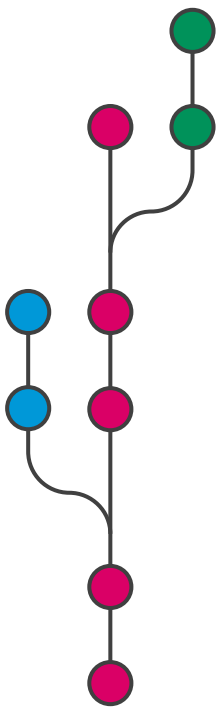
Silvan Zahno silvan.zahno@hevs.ch

Your current system

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

1 directory, 7 files
```

Version control



master v1.0.1 c4a9fee Merge branch 'release' into 'master' - Silvan Zahno <silvan.zahno@hevs.ch>

release 66ccde9 Merge branch 'develop' into 'release' - Silvan Zahno <silvan.zahno@hevs.ch>

2377d1e forgot to heat it up - Silvan Zahno <silvan.zahno@hevs.ch>

develop c43eda8 a whole cheese pizza, just for me - Silvan Zahno <silvan.zahno@hevs.ch>

fe3a9ee watter balloons (for the kids) - Silvan Zahno <silvan.zahno@hevs.ch>

v1.0.0 3b2d15b Merge branch 'release' into 'master' - Silvan Zahno <silvan.zahno@hevs.ch>

9dc3571 Merge branch 'develop' into 'release' - Silvan Zahno <silvan.zahno@hevs.ch>

fcd53f8 prepare for release - Silvan Zahno <silvan.zahno@hevs.ch>

9ddb202 added fabric softener - Silvan Zahno <silvan.zahno@hevs.ch>

a969898 added eggs - Silvan Zahno <silvan.zahno@hevs.ch>

ca10016 added milk - Silvan Zahno <silvan.zahno@hevs.ch>

e60c1b0 initial commit - Silvan Zahno <silvan.zahno@hevs.ch>

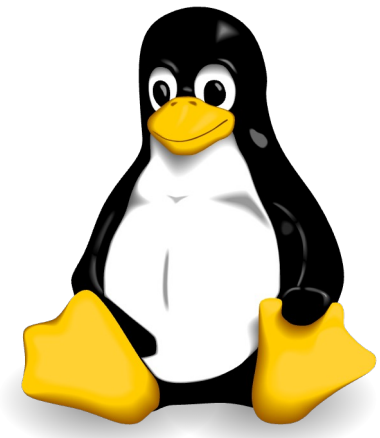
Possible Tools

Git (git)

Subversion (svn)

Mercurial (hg)

Linux Torvalds

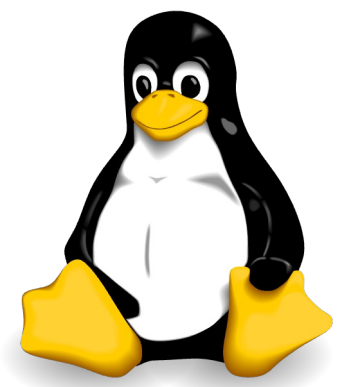


Linux (1991)

Git (2005)

Why Linux needs git

Linux has become the largest collaborative development project in the history of computing over the last 30 years.



- 600 active Linux distribution
- 85% of all Smartphones
- 500 top supercomputers
- >27.8 millon lines of code
- > 12’000 contributors
- > 1 million commits

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

Git Platforms

Gitlab

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

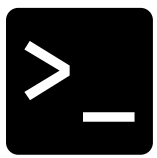
Github

https://github.com

Bitbucket

https://bitbucket.com

Git Tools

Commandline 

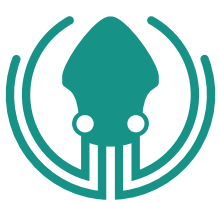
Fork 

Sublime Merge 

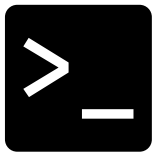
Tower 

Git Cola 

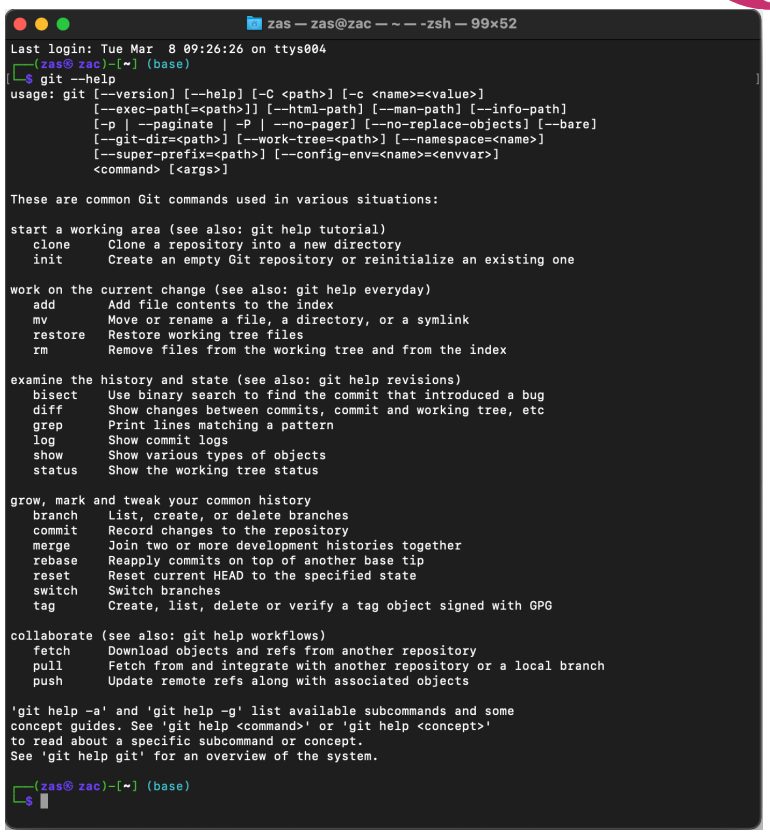
SmartGit 

Git Kraken 

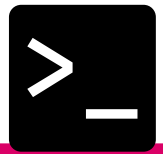
Git Commandline



```
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
```



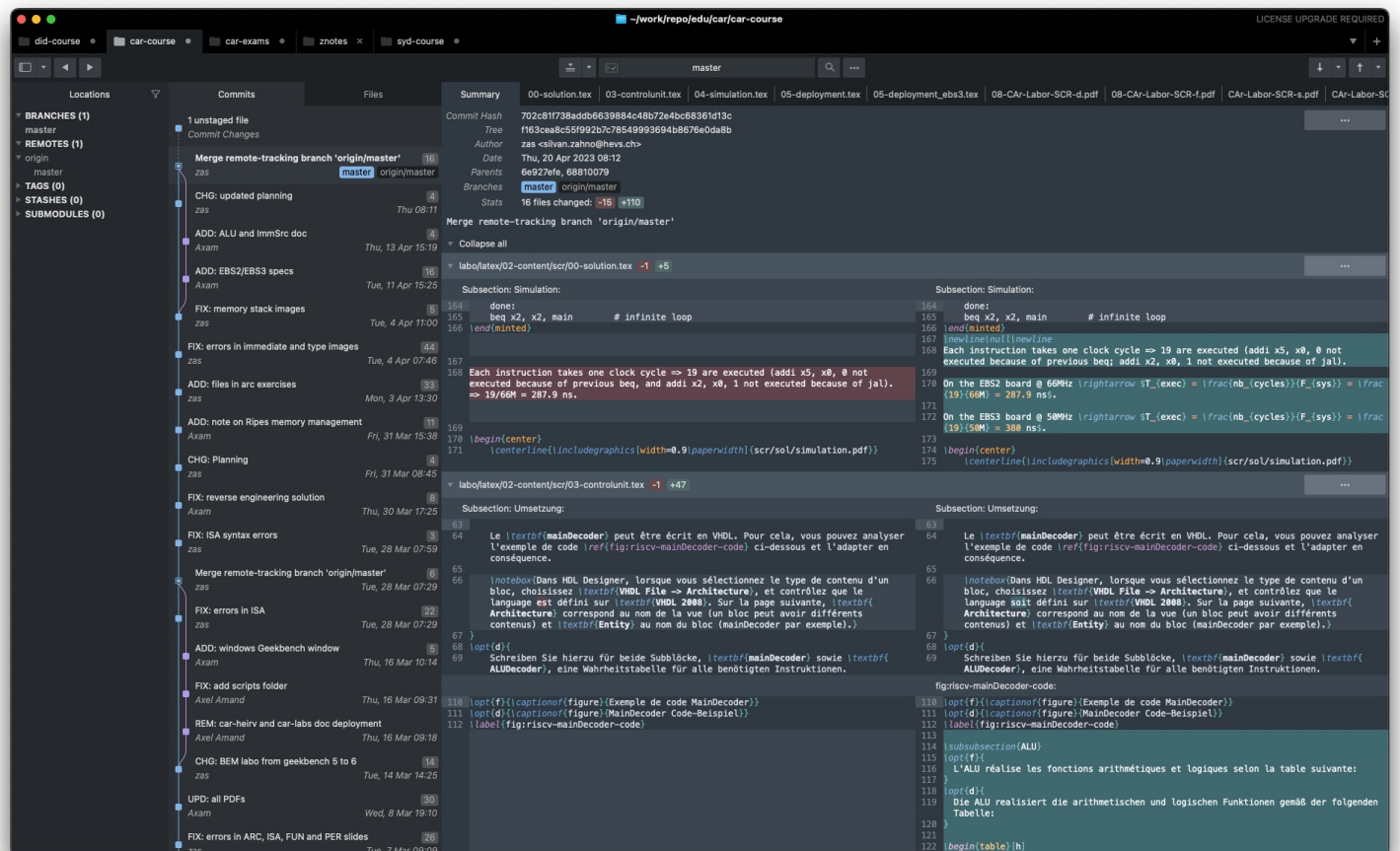
`git` Commands



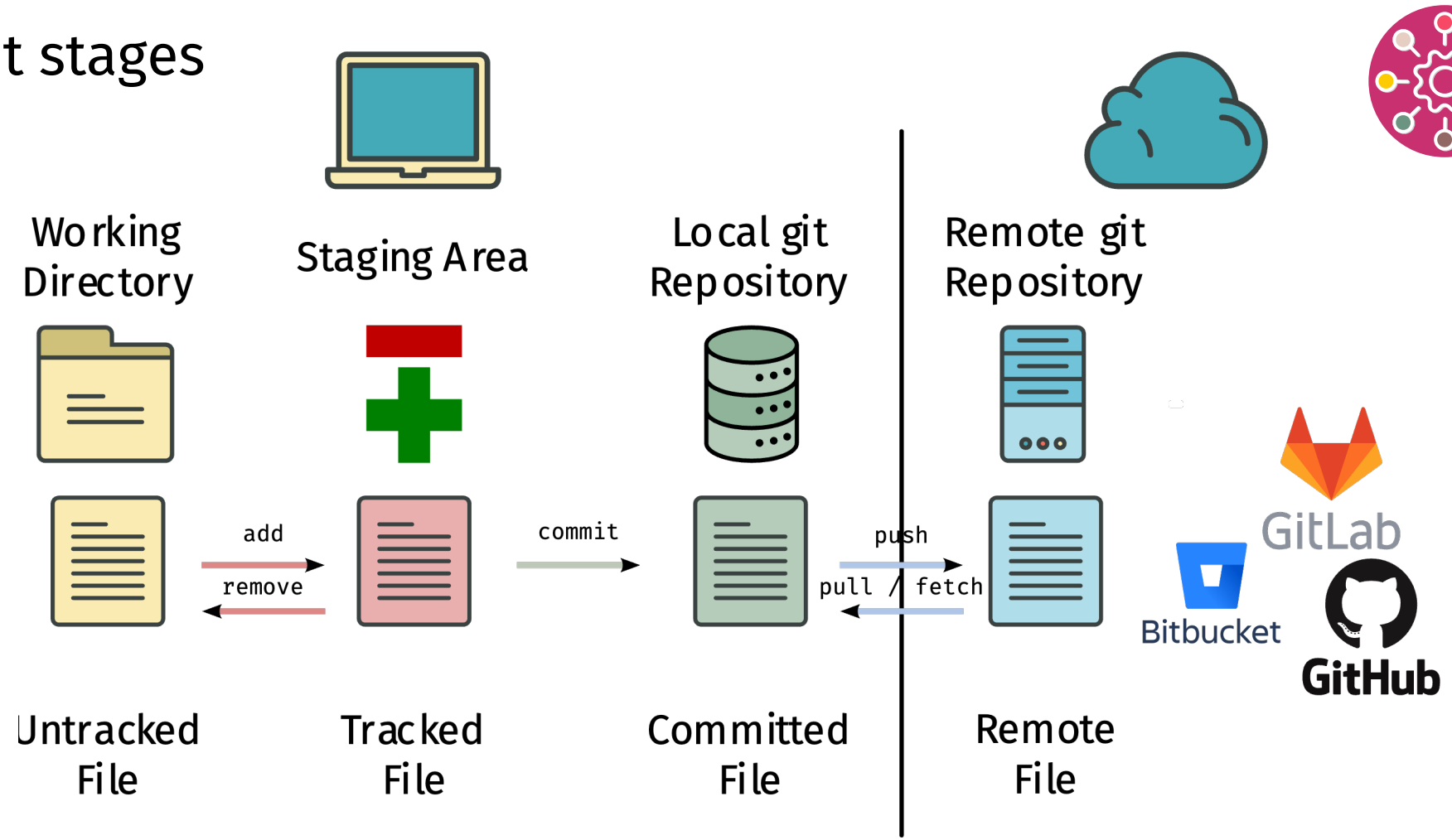
Command	Description
Start a working area	
`clone`	Clone a repository into a new directory
`init`	Create an empty Git repository or reinitialize an existing one
Work on the current change	
`add`	Add file contents to the index
`mv`	Move or rename a file, a directory, or a symlink
`reset`	Reset current HEAD to the specified state
`rm`	Remove files from the working tree and from the index
Examine the history and state	
`log`	Show commit logs
`show`	Show various types of objects
`status`	Show the working tree status

Command	Description
Grow, mark and tweak your common history	
`branch`	List, create, or delete branches
`checkout`	Switch branches or restore working tree files
`commit`	Record changes to the repo
`diff`	Show changes between commits, commit and working tree, etc
`merge`	Join two or more development histories together
`rebase`	Reapply commits on top of another base tip
`tag`	Create, list, delete or verify a tag object
`Collaborate`	
`fetch`	Download objects and refs from another repo
`pull`	Fetch from and integrate with another repo or a local branch
`push`	Update remote refs along with associated objects

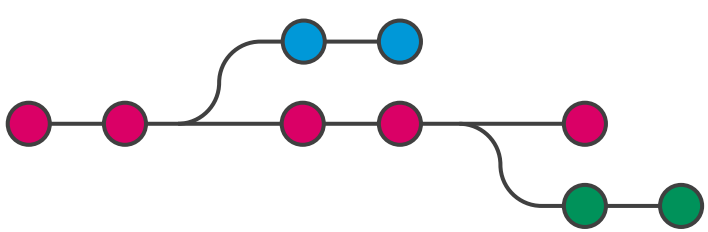
Sublime Merge



Git stages


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

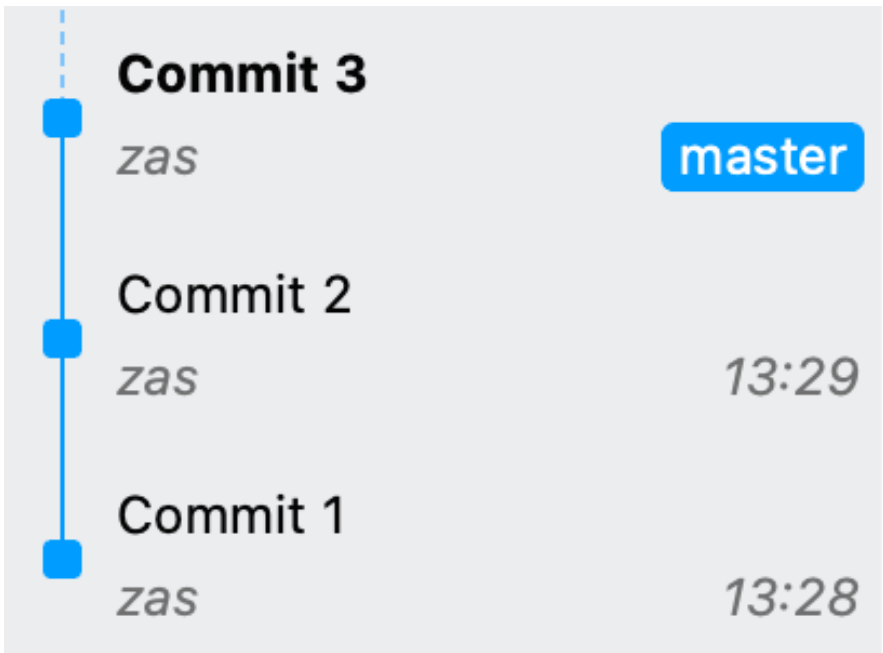
Git Branch and Merge Example



Branch and Merge

Initial repo state

Every repo has either a `main` or `master` branch as default branch

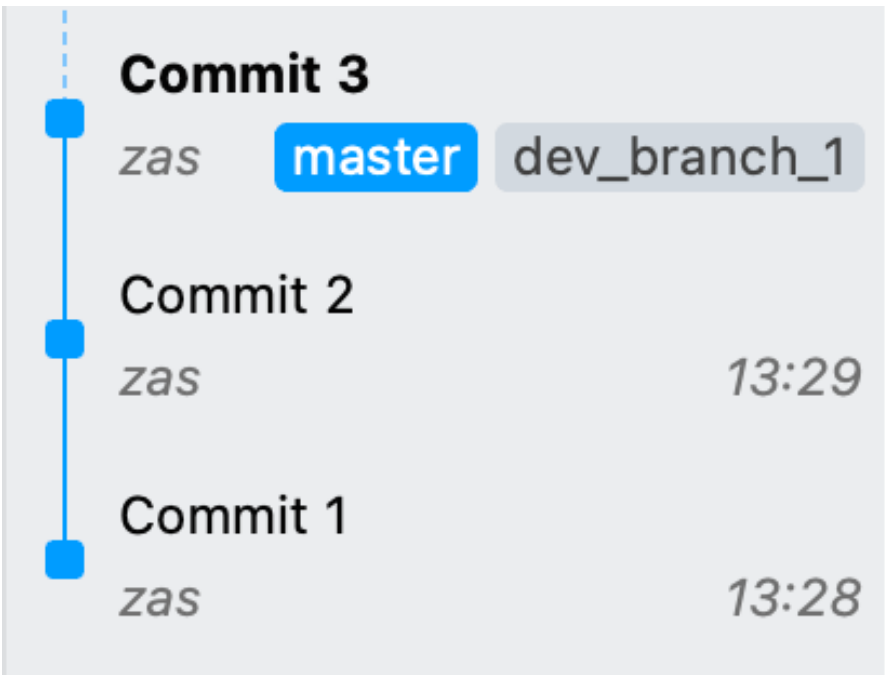


Branch and Merge

Create branch dev_branch_1

Create branch dev_branch_1

```
$ git branch dev_branch_1
```



Branch and Merge

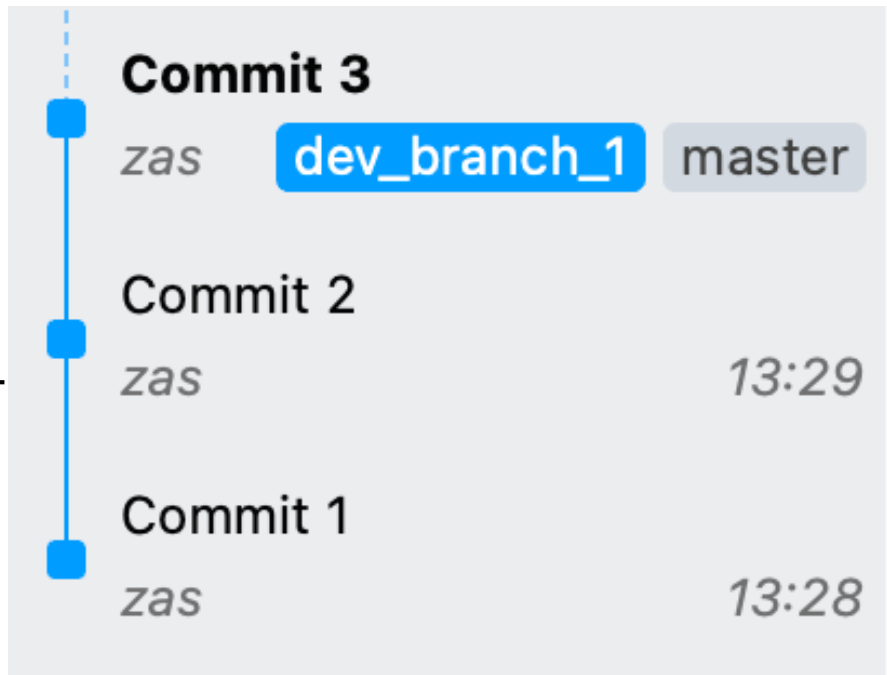
Checkout branch `dev_branch_1`

Check on which branch we are on

```
$ git branch
```

Checkout branch `dev_branch_1`

```
$ git checkout dev_branch_1
```



Branch and Merge

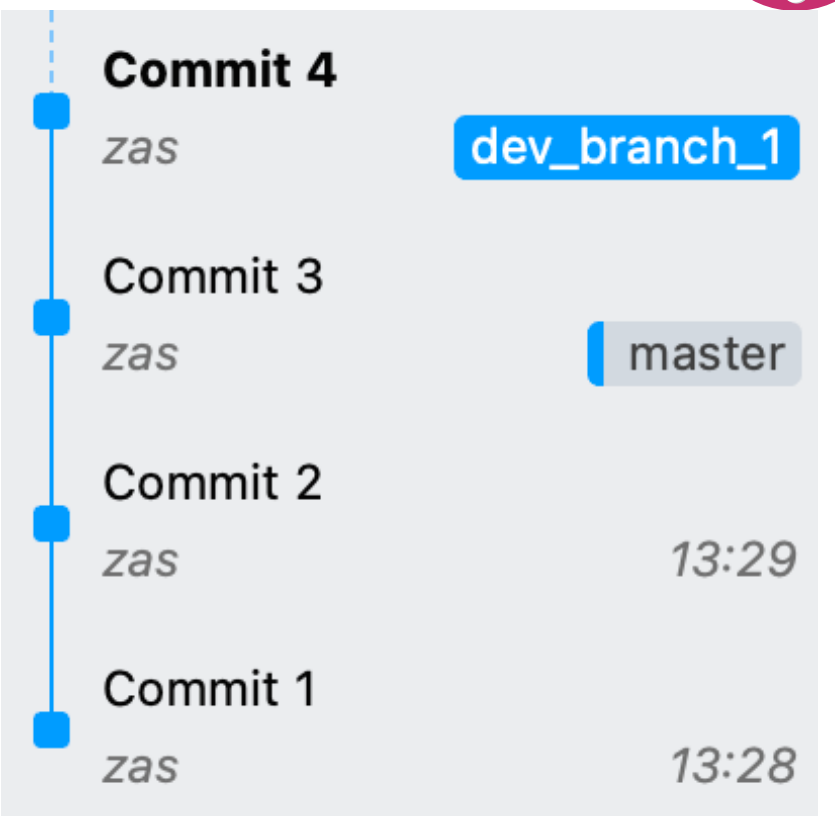
Commit on dev_branch_1

Stage new file

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

Commit stages files

```
$ git commit –m “Commit 4”
```



Branch and Merge

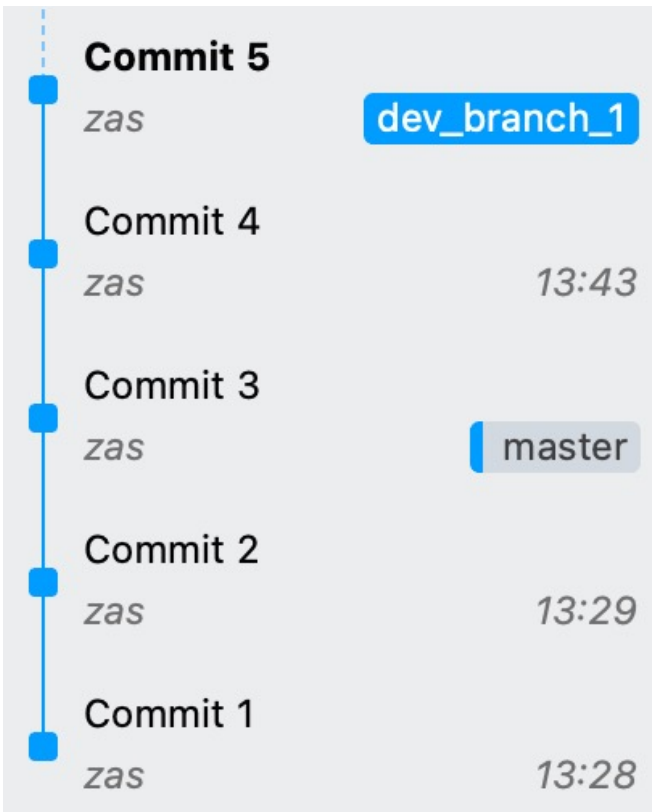
Commit on dev_branch_1

Stage new file

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

Commit stages files

```
$ git commit –m “Commit 5”
```



Branch and Merge

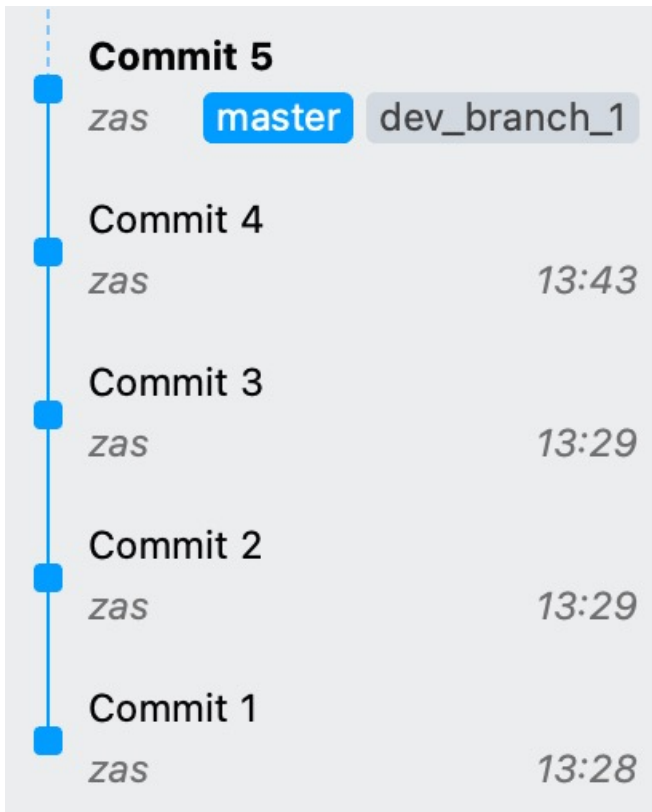
Merge `master` and `dev_branch_1`

Checkout `master` branch

```
$ git checkout master
```

Merge `dev_branch_1` into `master`

```
$ git merge dev_branch_1
```



Branch and Merge

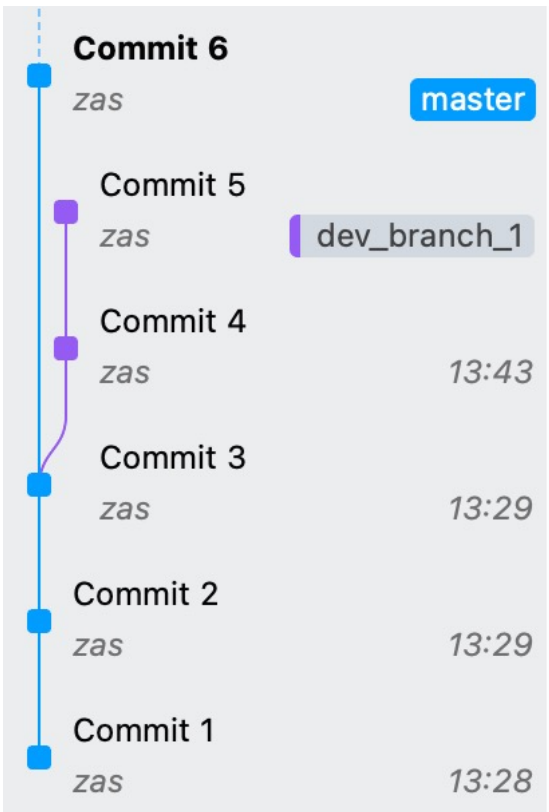
Commit on `master` branch

Stage new file

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

Commit stages files

```
$ git commit –m “Commit 6”
```

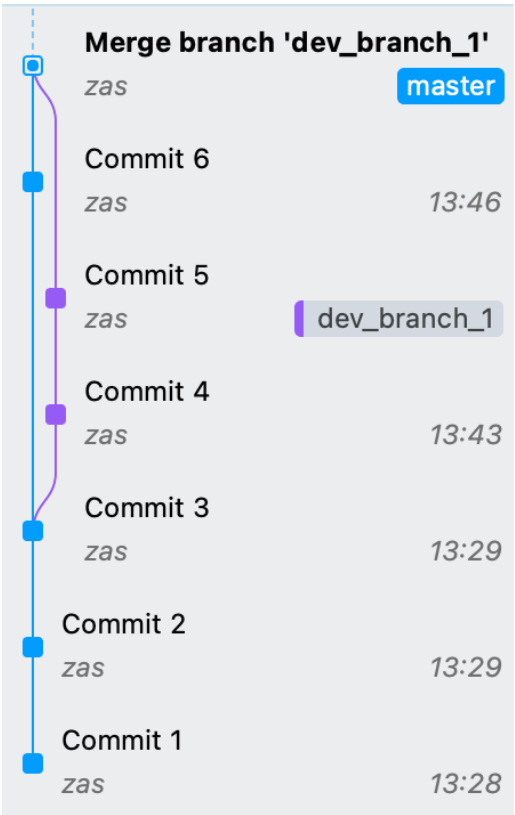


Branch and Merge

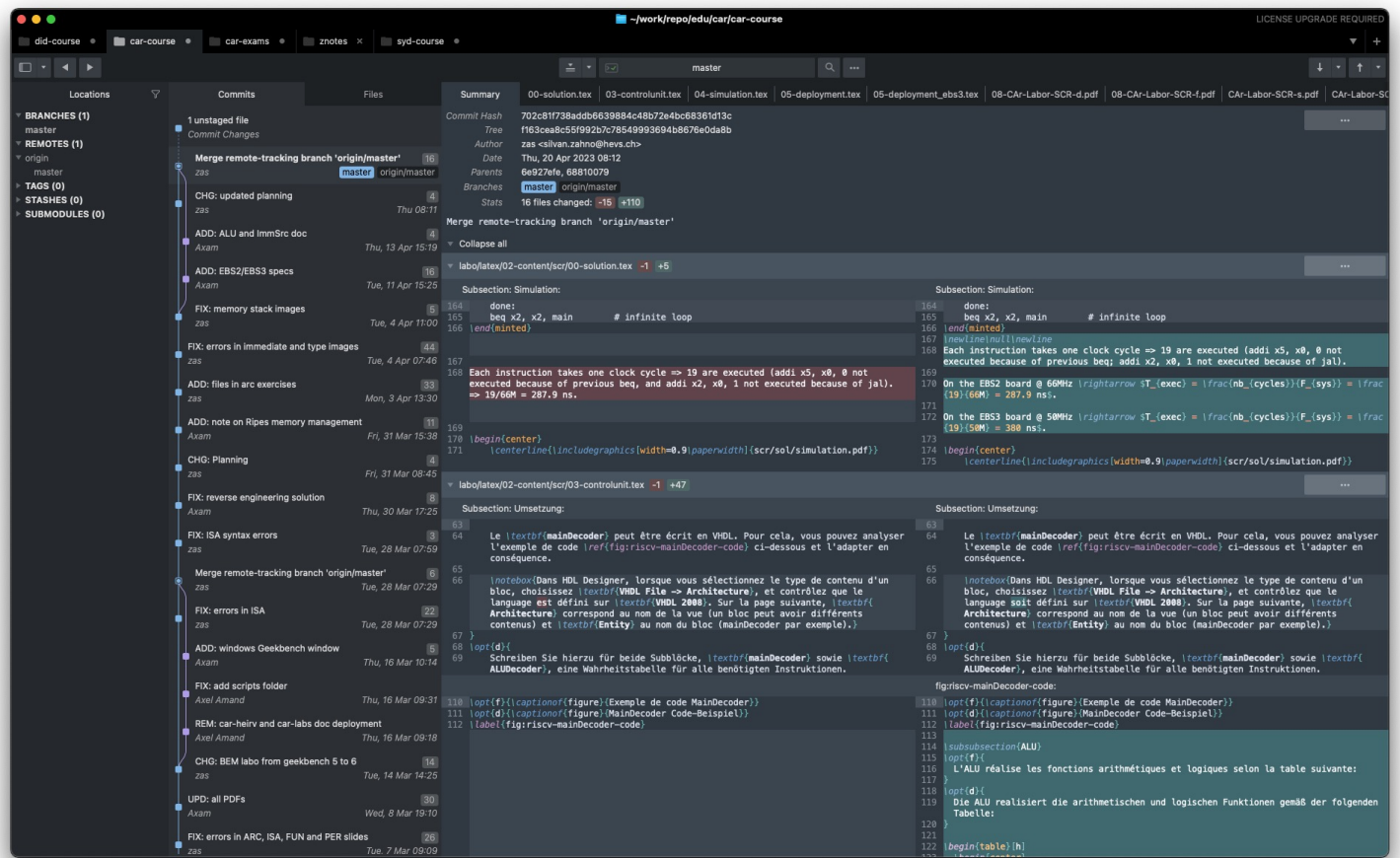
Three way merge `master` and `dev_branch_1`

Merge `dev_branch_1` into `master`

```
$ git merge dev_branch_1
```

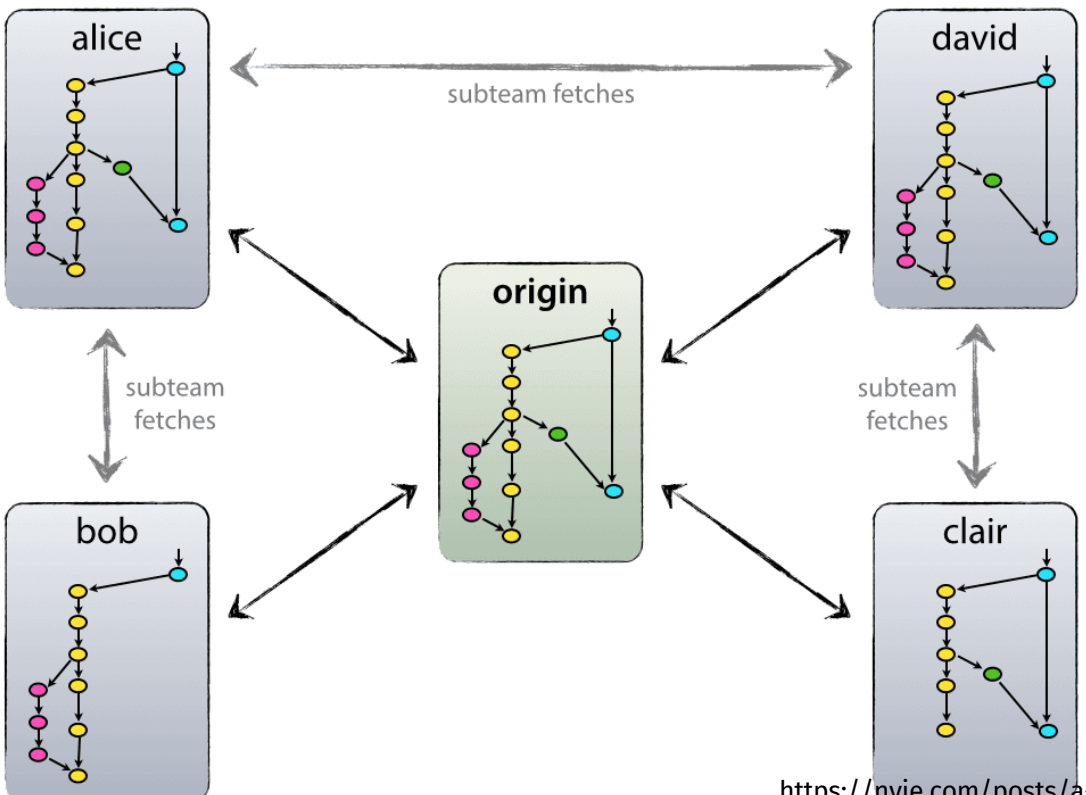


Demo


~/work/repo/edu/car/car-course
LICENSE UPGRADE REQUIRED
did-course
car-course
car-exams
znotes
syd-course
master
Locations
Commits
Files
Summary
BRANCHES (1)
master
REMOTES (1)
origin
master
TAGS (0)
STASHES (0)
SUBMODULES (0)
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
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 f163cea8c55f992b7c78549993694b8676e0da8b
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:
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.
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.
}
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]

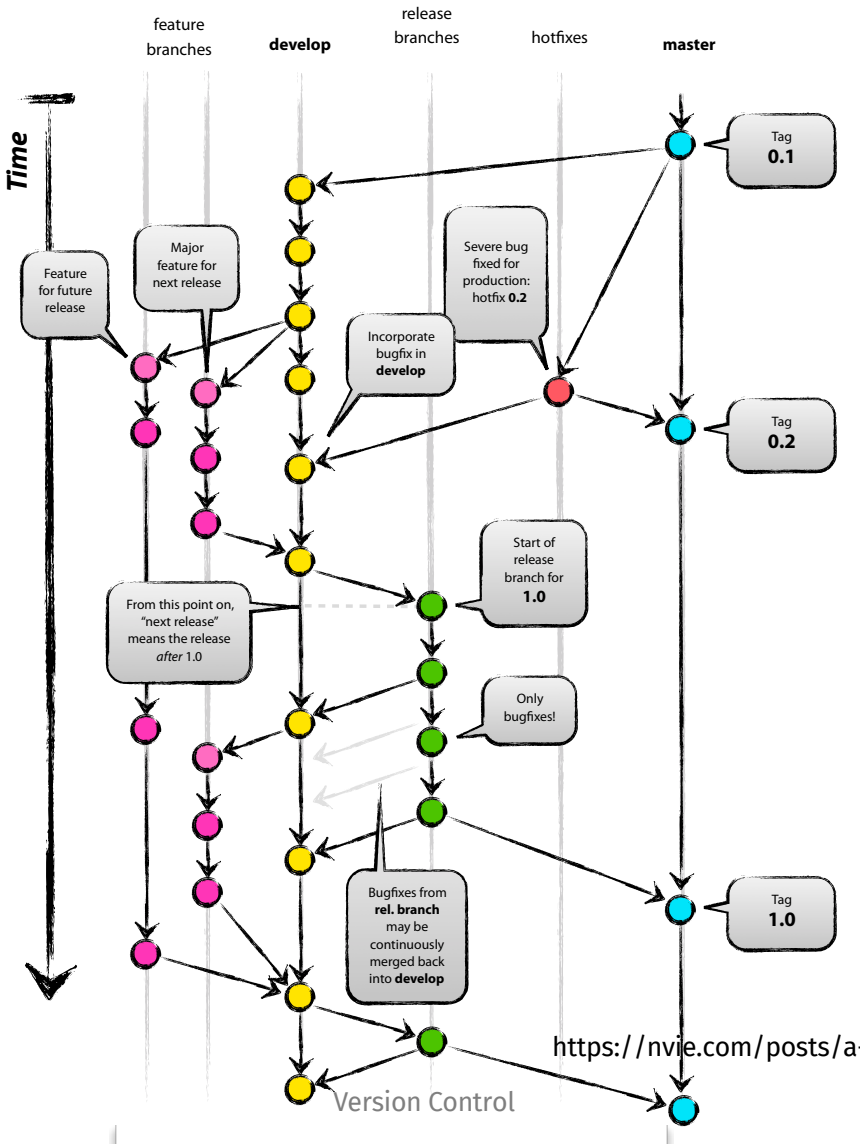
Gitflow

Gitflow Collaboration



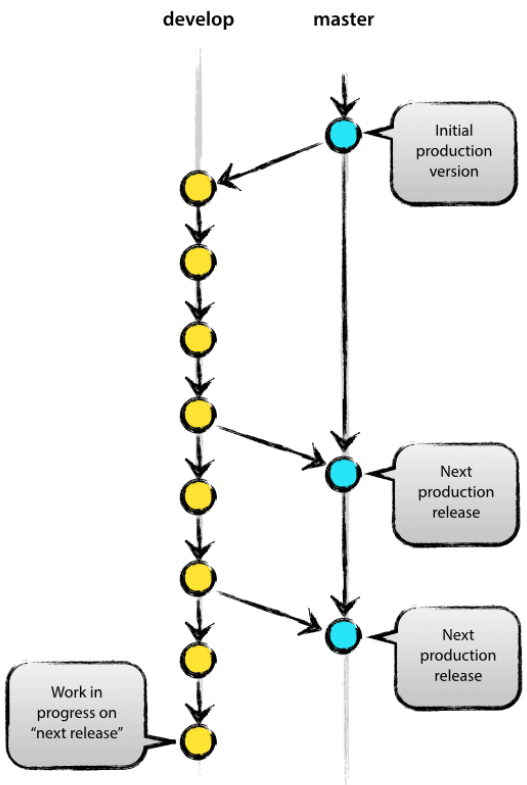
https://nvie.com/posts/a-successful-git-branching-model/

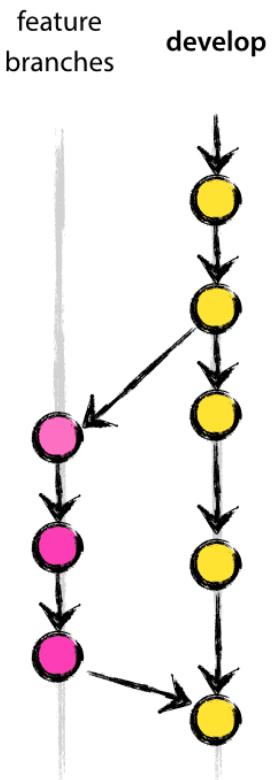
Gitflow

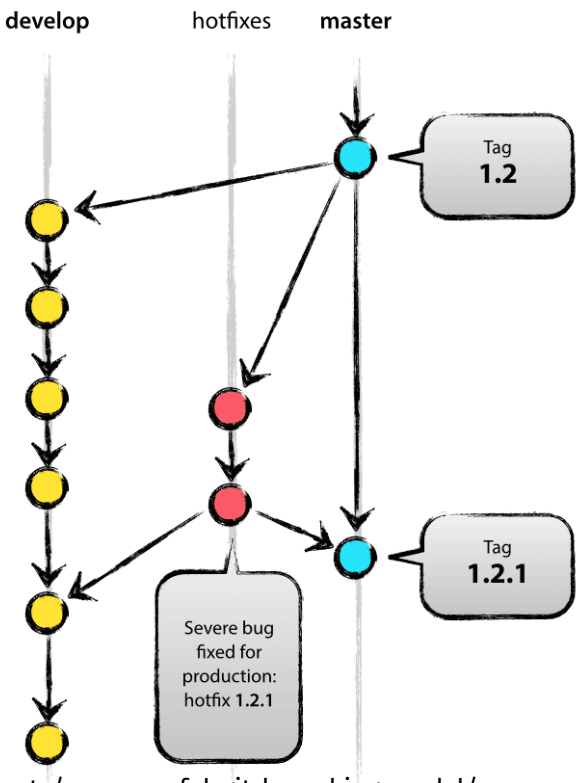


https://nvie.com/posts/a-successful-git-branching-model/

Gitflow Branching







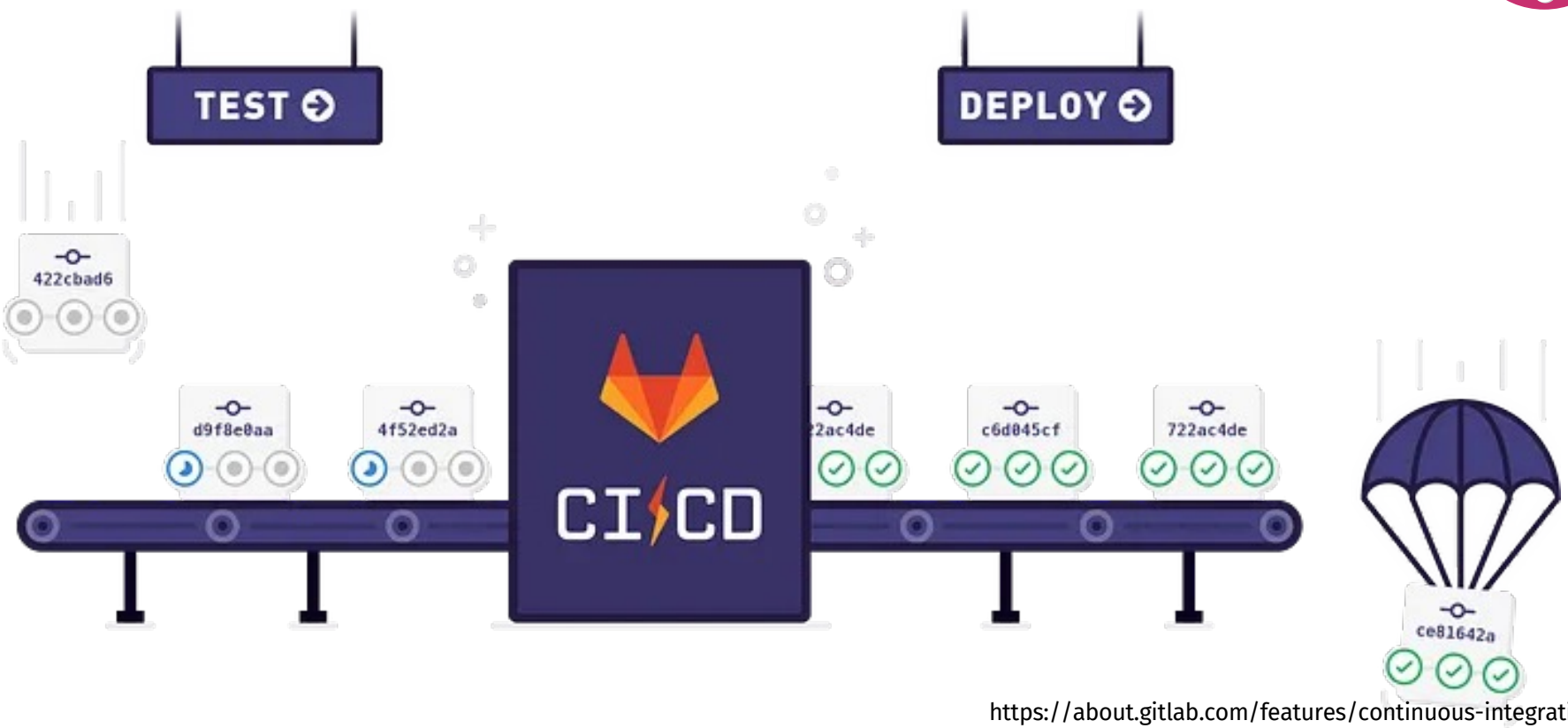
https://nvie.com/posts/a-successful-git-branching-model/

Git CI/CD

What is CI/CD?

- Continuous Integration (CI) is the practice of frequently integrating code changes into a shared repository, which is then automatically built and tested.
- Continuous Delivery (CD) takes CI a step further by automatically deploying code changes to production-like environments for further testing and validation.
- Automated testing is a critical component of CI/CD, as it helps catch bugs and other issues early in the development process.
- Popular tools are GitLab CI/CD, Github Actions, Jenkins, CircleCI, and Travis CI.

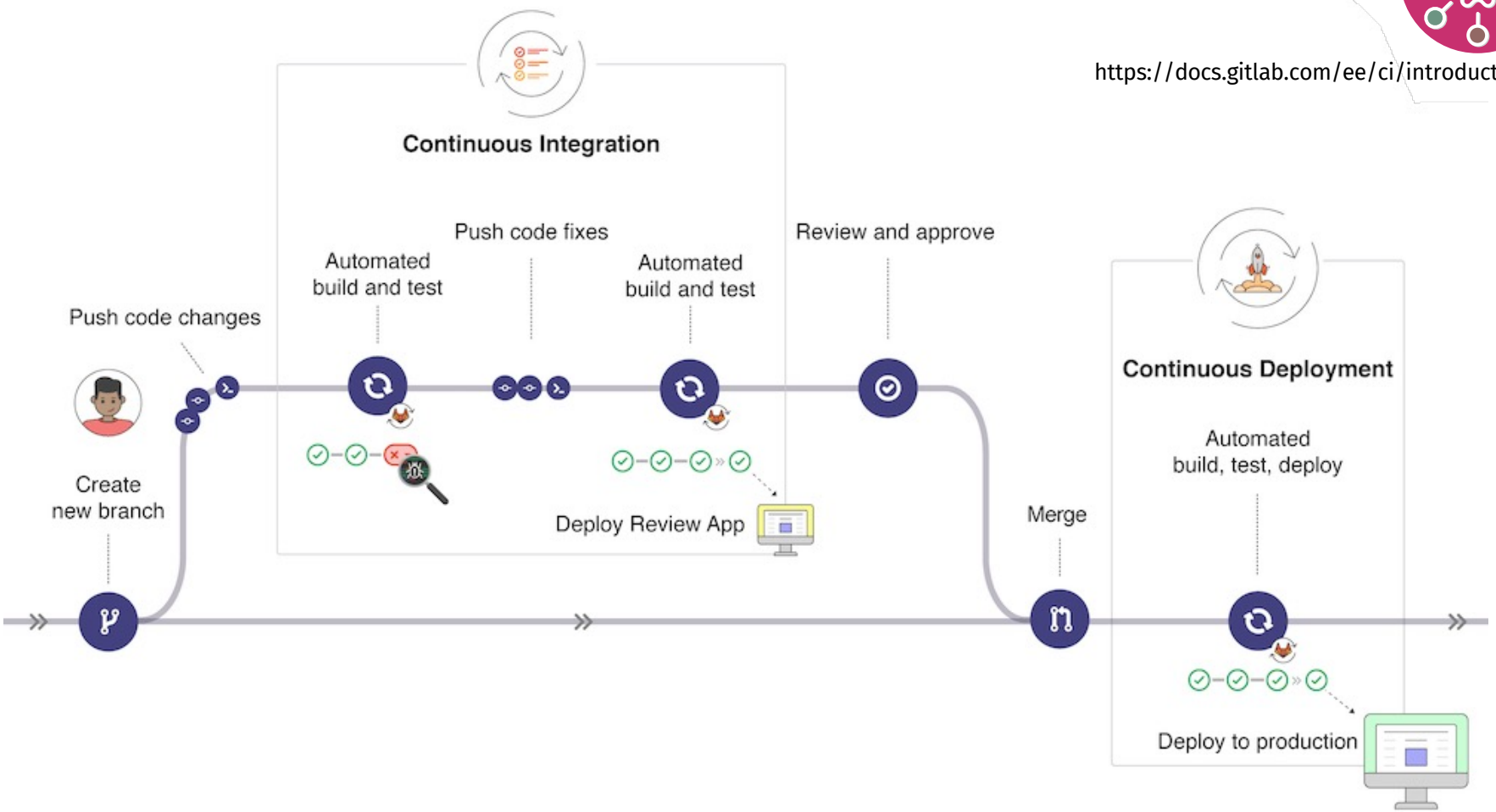
What is CI/CD?



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

Gitlab Workflow

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

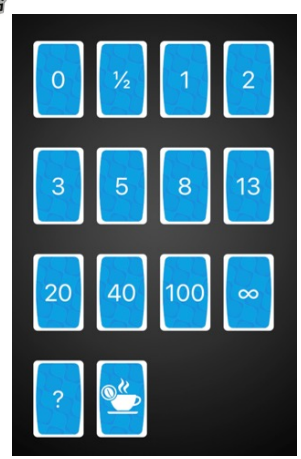


Task Estimation

Estimation 4

Create a Star Wars
Star Destroyer (4784pcs)

X Point(s)

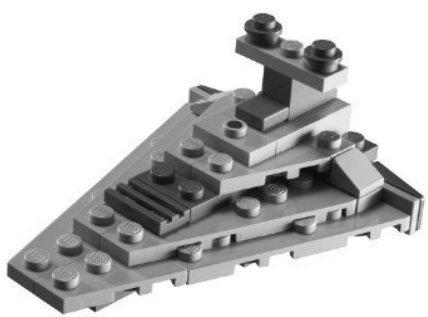


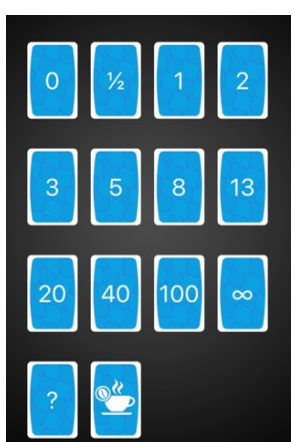
Task Estimation

Estimation 5

Create a Star Wars
Star Destroyer (37pcs)

X Point(s)





Introduction to git file versioning

Contents