

Task Estimation

Estimation 2

Create a small house
X Point(s)



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

Task Estimation

Estimation 3

Create a programmable Lego robot

X Point(s)

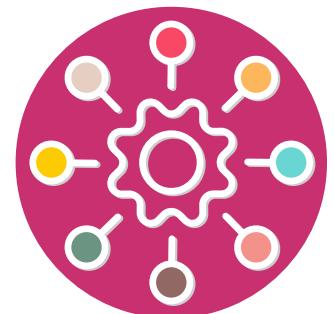
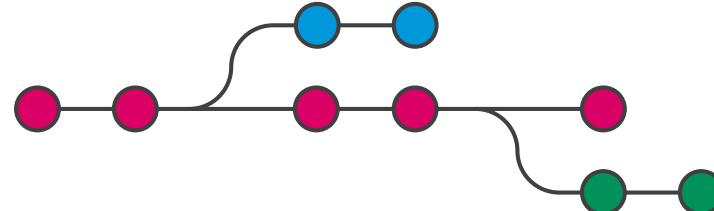


0	$\frac{1}{2}$	1	2
3	5	8	13
20	40	100	∞
?	☕		



System Design Version Control

Systems Engineering program



Silvan Zahno silvan.zahno@hevs.ch

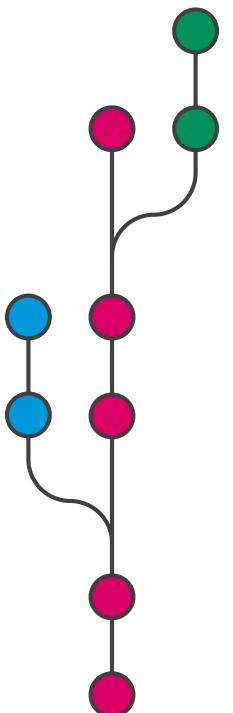


Your current system

```
•
  └── 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





Possible Tools

Git (git)



Subversion (svn)

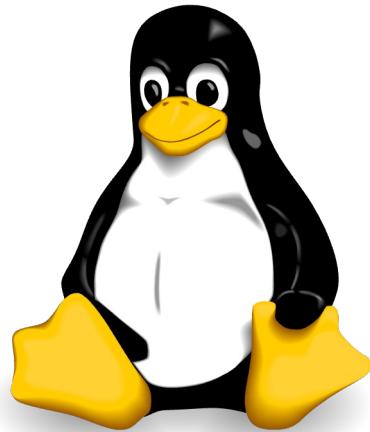


Mercurial (hg)

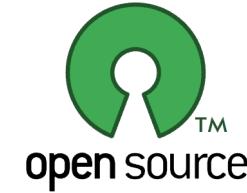


Linus Torvalds

Linux (1991)

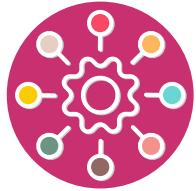
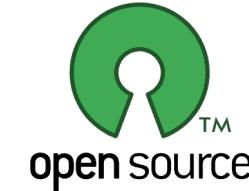
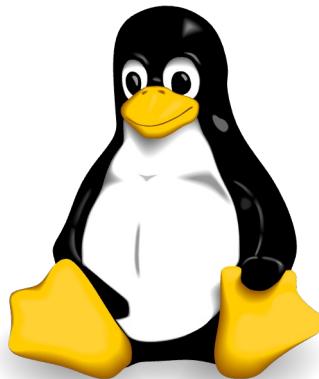


Git (2005)



Why Linux needs git

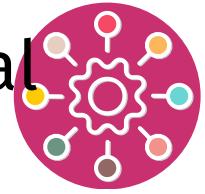
Linux has become the largest collaborative development project in the history of computing over the last 30 years.



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

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

Why you need git in Power&Control or Design&Material



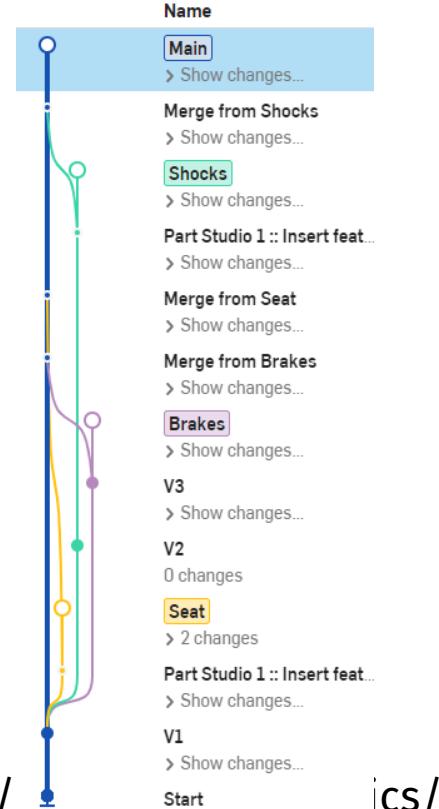
Versioning

- Tracking Changes
- Collaboration
- Rollback Capabilities
- Documentation
- Deployment



AUTODESK
Vault

<https://truelist.co/>



Git Platforms



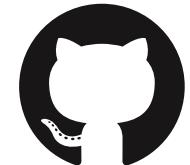
Gitlab

<https://gitlab.com>

<https://gitlab.hevs.ch>



GitLab



GitHub

Github

<https://github.com>

Bitbucket

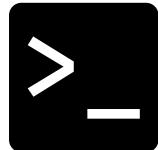
<https://bitbucket.com>



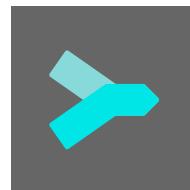
Bitbucket

Git Tools

Commandline



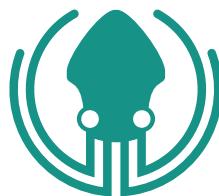
Sublime Merge



Git Cola



Git Kraken

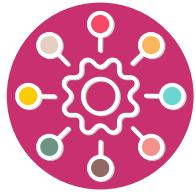


Fork

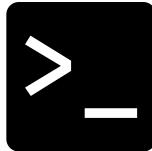


Tower

SmartGit



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

```
zas@zac:~$ git --help
Last login: Tue Mar  8 09:26:26 on ttys004
[zs@zac:~] (base)
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path=<path>] [-h <html-path>] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [-e | --no-replace-objects] [--bare]
           [--git-dir=<path>] [-w | --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

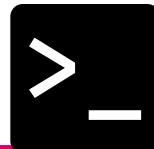
collaborate (see also: git help workflows)
fetch   Download objects and refs from another repository
pull    Fetch from and integrate with another repository or a local branch
push    Update remote refs along with associated objects

'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.

(zs@zac:~) (base)
zs$
```

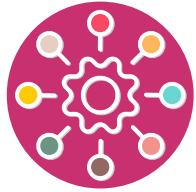
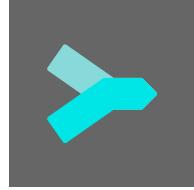


git Commands



Command	Description	Command	Description
Start a working area		Grow, mark and tweak your common history	
clone	Clone a repository into a new directory	branch	List, create, or delete branches
init	Create an empty Git repository or reinitialize an existing one	checkout	Switch branches or restore working tree files
Work on the current change		commit	Record changes to the repo
add	Add file contents to the index	diff	Show changes between commits, commit and working tree, etc
mv	Move or rename a file, a directory, or a symlink	merge	Join two or more development histories together
reset	Reset current HEAD to the specified state	rebase	Reapply commits on top of another base tip
rm	Remove files from the working tree and from the index	tag	Create, list, delete or verify a tag object
Examine the history and state		Collaborate	
log	Show commit logs	fetch	Download objects and refs from another repo
show	Show various types of objects	pull	Fetch from and integrate with another repo or a local branch
status	Show the working tree status	push	Update remote refs along with associated objects

Sublime Merge



The screenshot shows a GitHub repository interface for a project named 'car-course'. The repository has the following structure:

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

Commits (1 untagged file, Commit Changes):

- Merge remote-tracking branch 'origin/master' (16 commits by zas, Thu, 20 Apr 08:11)
 - CHG: updated planning (4 commits by zas, Thu, 13 Apr 15:19)
 - ADD: ALU and immSrc doc (4 commits by Axam, Thu, 13 Apr 15:19)
 - ADD: EBS2/EBS3 specs (4 commits by Axam, Tue, 11 Apr 15:25)
 - FIX: memory stack images (5 commits by zas, Tue, 4 Apr 11:00)
 - FIX: errors in immediate and type images (44 commits by zas, Tue, 4 Apr 07:46)
 - ADD: files in arc exercises (33 commits by zas, Mon, 3 Apr 13:30)
 - ADD: note on Ripes memory management (11 commits by Axam, Fri, 31 Mar 15:38)
 - CHG: Planning (4 commits by zas, Fri, 31 Mar 08:45)
 - FIX: reverse engineering solution (8 commits by Axam, Thu, 30 Mar 17:26)
 - FIX: ISA syntax errors (3 commits by zas, Tue, 28 Mar 07:59)
 - Merge remote-tracking branch 'origin/master' (6 commits by zas, Tue, 28 Mar 07:29)
 - FIX: errors in ISA (22 commits by zas, Tue, 28 Mar 07:29)
 - ADD: windows Geekbench window (5 commits by Axam, Thu, 16 Mar 10:14)
 - FIX: add scripts folder (Axel Amand, Thu, 16 Mar 09:31)
 - REM: car-hdrv and car-labs doc deployment (Axel Amand, Thu, 16 Mar 09:18)
 - CHG: BEM labo from geekbench 5 to 6 (14 commits by zas, Tue, 14 Mar 14:25)
 - UPD: all PDFs (30 commits by Axam, Wed, 8 Mar 19:10)
 - FIX: errors in ARC, ISA, FUN and PER slides (26 commits by zas, Tue, 7 Mar 09:09)

Files (Summary):

- Commit Hash: 702cb1f738add663984a48b72e4bc68361d3c
- Tree: f163ceab55f992b7c7854993994b8676e0da8b
- Author: zas <silvan.zahn@hevs.ch>
- Date: Thu, 20 Apr 2023 08:12
- Parents: 6e927afe, 68810079
- Branches: master origin/master
- Stats: 16 files changed: -15 +10

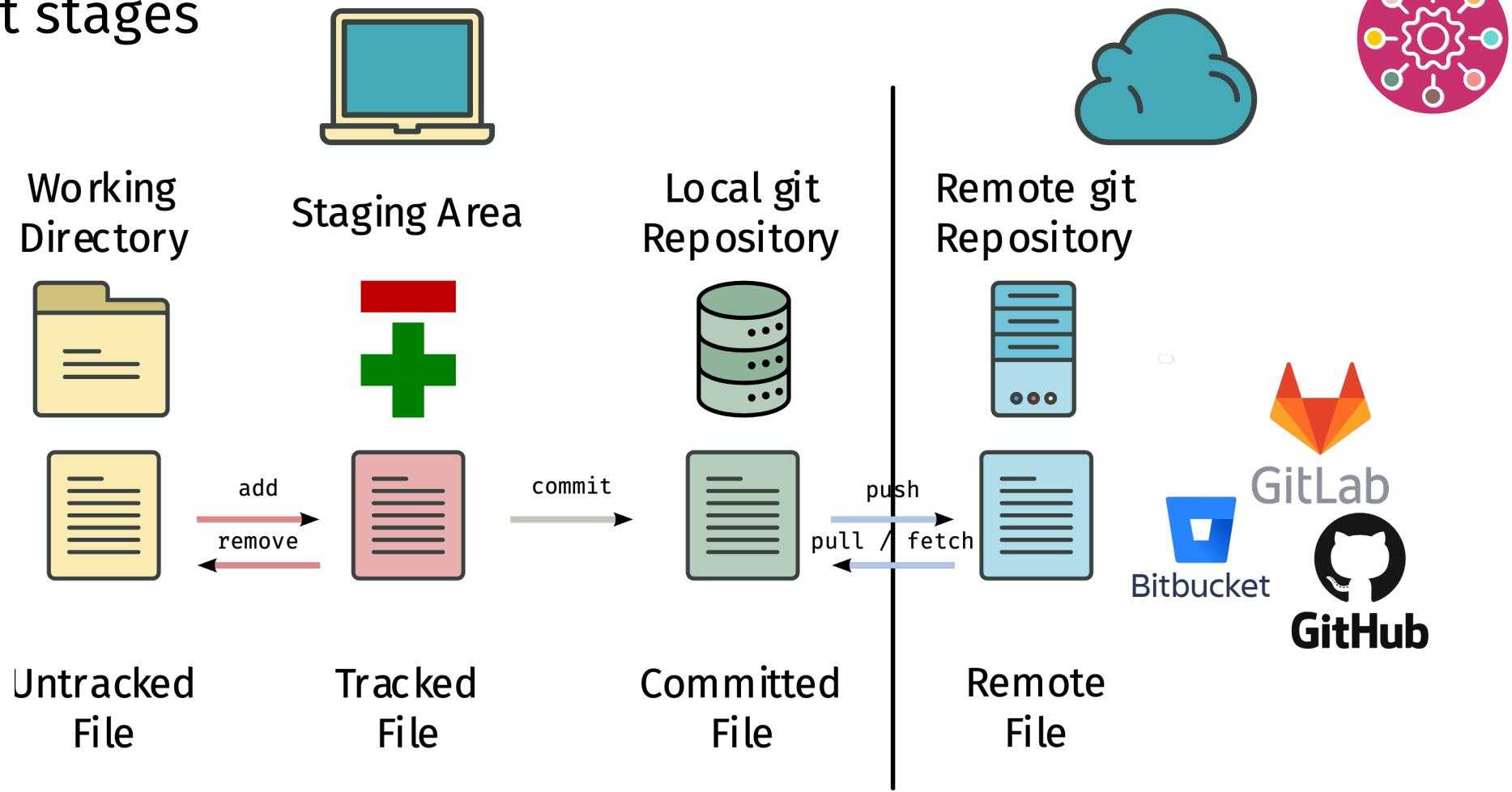
Code Review (00-solution.tex, 03-controlunit.tex, 04-simulation.tex, 05-deployment.tex, 05-deployment_ebs3.tex, 08-CAR-Labor-SCR-d.pdf, 08-CAR-Labor-SCR-f.pdf, CAR-Labor-SCR-s.pdf, CAR-Labor-SCR-t.pdf):

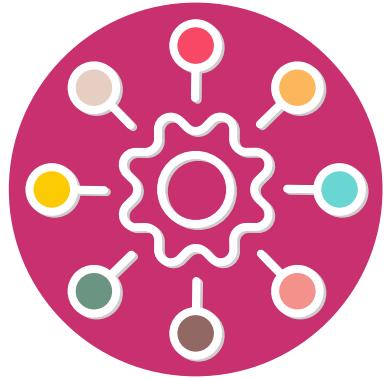
- 00-solution.tex: Merge remote-tracking branch 'origin/master'
- 03-controlunit.tex: Merge remote-tracking branch 'origin/master'
- 04-simulation.tex: Merge remote-tracking branch 'origin/master'
- 05-deployment.tex: Merge remote-tracking branch 'origin/master'
- 05-deployment_ebs3.tex: Merge remote-tracking branch 'origin/master'
- 08-CAR-Labor-SCR-d.pdf: Merge remote-tracking branch 'origin/master'
- 08-CAR-Labor-SCR-f.pdf: Merge remote-tracking branch 'origin/master'
- CAR-Labor-SCR-s.pdf: Merge remote-tracking branch 'origin/master'
- CAR-Labor-SCR-t.pdf: Merge remote-tracking branch 'origin/master'

Code Snippets (00-solution.tex, 03-controlunit.tex, 04-simulation.tex, 05-deployment.tex, 05-deployment_ebs3.tex, 08-CAR-Labor-SCR-d.pdf, 08-CAR-Labor-SCR-f.pdf, CAR-Labor-SCR-s.pdf, CAR-Labor-SCR-t.pdf):

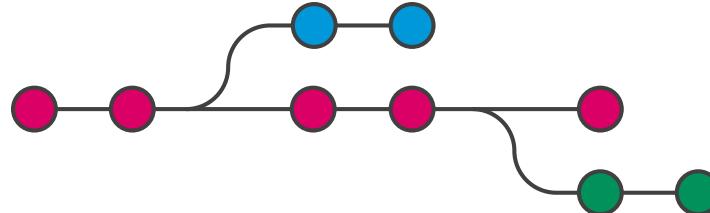
- 00-solution.tex: Subsection: Simulation:
 - 164 done:
165 beq x2, x2, main # infinite loop
166 end;#intended;
 - 167 Each instruction takes one clock cycle => 19 are executed (addi x5, x0, 0 not executed because of previous beq; addi x2, x0, 1 not executed because of jal).
=> 19*60M = 287.9 ns.
- 03-controlunit.tex: Subsection: Umsetzung:
 - 63 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.
 - 64 **inotbox**[Dans HDL Designer, lorsque vous sélectionnez le type de contenu d'un bloc, choisissez [textbf{VHDL File} => Architecture], et contrôlez que le langage 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.)]
 - 65 **inotbox**[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 (mainDecoder par exemple.)]
 - 66 **inotbox**[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 (mainDecoder par exemple.)]
 - 67]
68 **lopt{d}**
69 Schreiben Sie hierzu für beide Subblöcke, [textbf{mainDecoder}] sowie [textbf{ALUDecoder}], eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 110 **lopt{f}**[Caption of figure: Exemple de code MainDecoder]
111 **lopt{d}**[Caption of figure: MainDecoder Code-Beispiel]
112 **label{fig:riscv-mainDecoder-code}**
 - 113 **subsubsection{ALU}**
114 **lopt{f}**
115 L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:
116 **table**
117 **lopt{d}**
118 Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden Tabelle:
119 **begin{table}[h]**
120 **end{table}**
121 **lopt{f}**
- 04-simulation.tex: Subsection: Simulation:
 - 164 done:
165 beq x2, x2, main # infinite loop
166 end;#intended;
 - 167 On the EBS2 board @ 60MHz [rightarrow ST_exec] = $\frac{1}{60 \cdot 60M} = 287.9$ ns.
171 On the EBS3 board @ 50MHz [rightarrow ST_exec] = $\frac{1}{50 \cdot 50M} = 388$ ns.
173 **begin{center}**
174 **centerline{\includegraphics[width=0.9\paperwidth]{scr/sol/simulation.pdf}}**
175 **end{center}**
- 05-deployment.tex: Subsection: Umsetzung:
 - 63 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.
 - 64 **inotbox**[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 (mainDecoder par exemple.)]
 - 65 **inotbox**[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 (mainDecoder par exemple.)]
 - 66 **inotbox**[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 (mainDecoder par exemple.)]
 - 67]
68 **lopt{d}**
69 Schreiben Sie hierzu für beide Subblöcke, [textbf{mainDecoder}] sowie [textbf{ALUDecoder}], eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 110 **lopt{f}**[Caption of figure: Exemple de code MainDecoder]
111 **lopt{d}**[Caption of figure: MainDecoder Code-Beispiel]
112 **label{fig:riscv-mainDecoder-code}**
 - 113 **subsubsection{ALU}**
114 **lopt{f}**
115 L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:
116 **table**
117 **lopt{d}**
118 Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden Tabelle:
119 **begin{table}[h]**
120 **end{table}**
121 **lopt{f}**
- 05-deployment_ebs3.tex: Subsection: Umsetzung:
 - 63 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.
 - 64 **inotbox**[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 (mainDecoder par exemple.)]
 - 65 **inotbox**[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 (mainDecoder par exemple.)]
 - 66 **inotbox**[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) und [textbf{Entity}] au nom des Blöcken (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 110 **lopt{f}**[Caption of figure: Exemple de code MainDecoder]
111 **lopt{d}**[Caption of figure: MainDecoder Code-Beispiel]
112 **label{fig:riscv-mainDecoder-code}**
 - 113 **subsubsection{ALU}**
114 **lopt{f}**
115 L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:
116 **table**
117 **lopt{d}**
118 Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden Tabelle:
119 **begin{table}[h]**
120 **end{table}**
121 **lopt{f}**
- 08-CAR-Labor-SCR-d.pdf: Subsection: Umsetzung:
 - 63 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.
 - 64 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 65 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 66 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 110 **lopt{f}**[Caption of figure: Exemple de code MainDecoder]
111 **lopt{d}**[Caption of figure: MainDecoder Code-Beispiel]
112 **label{fig:riscv-mainDecoder-code}**
 - 113 **subsubsection{ALU}**
114 **lopt{f}**
115 L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:
116 **table**
117 **lopt{d}**
118 Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden Tabelle:
119 **begin{table}[h]**
120 **end{table}**
121 **lopt{f}**
- 08-CAR-Labor-SCR-f.pdf: Subsection: Umsetzung:
 - 63 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.
 - 64 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 65 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 66 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 110 **lopt{f}**[Caption of figure: Exemple de code MainDecoder]
111 **lopt{d}**[Caption of figure: MainDecoder Code-Beispiel]
112 **label{fig:riscv-mainDecoder-code}**
 - 113 **subsubsection{ALU}**
114 **lopt{f}**
115 L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:
116 **table**
117 **lopt{d}**
118 Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden Tabelle:
119 **begin{table}[h]**
120 **end{table}**
121 **lopt{f}**
- CAR-Labor-SCR-s.pdf: Subsection: Umsetzung:
 - 63 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.
 - 64 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 65 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 66 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 110 **lopt{f}**[Caption of figure: Exemple de code MainDecoder]
111 **lopt{d}**[Caption of figure: MainDecoder Code-Beispiel]
112 **label{fig:riscv-mainDecoder-code}**
 - 113 **subsubsection{ALU}**
114 **lopt{f}**
115 L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:
116 **table**
117 **lopt{d}**
118 Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden Tabelle:
119 **begin{table}[h]**
120 **end{table}**
121 **lopt{f}**
- CAR-Labor-SCR-t.pdf: Subsection: Umsetzung:
 - 63 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.
 - 64 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 65 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 66 **inotbox**[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) und [textbf{Entity}] au nom des Blöcke (mainDecoder und ALUDecoder), eine Wahrheitstabelle für alle benötigten Instruktionen.
 - 110 **lopt{f}**[Caption of figure: Exemple de code MainDecoder]
111 **lopt{d}**[Caption of figure: MainDecoder Code-Beispiel]
112 **label{fig:riscv-mainDecoder-code}**
 - 113 **subsubsection{ALU}**
114 **lopt{f}**
115 L'ALU réalise les fonctions arithmétiques et logiques selon la table suivante:
116 **table**
117 **lopt{d}**
118 Die ALU realisiert die arithmetischen und logischen Funktionen gemäß der folgenden Tabelle:
119 **begin{table}[h]**
120 **end{table}**
121 **lopt{f}**

Git stages





Git Branch and Merge Example

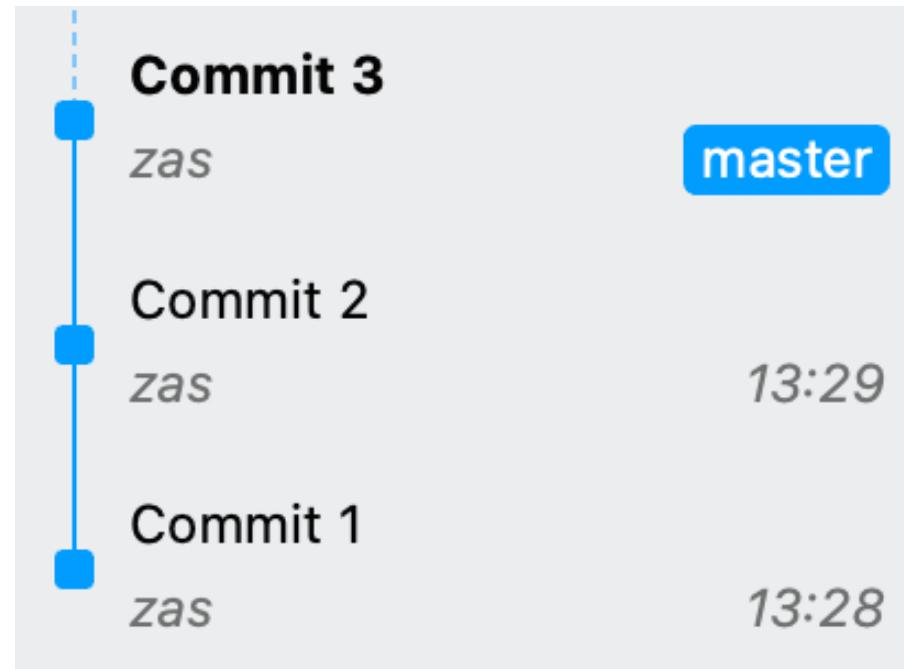




Branch and Merge

Initial repo state

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



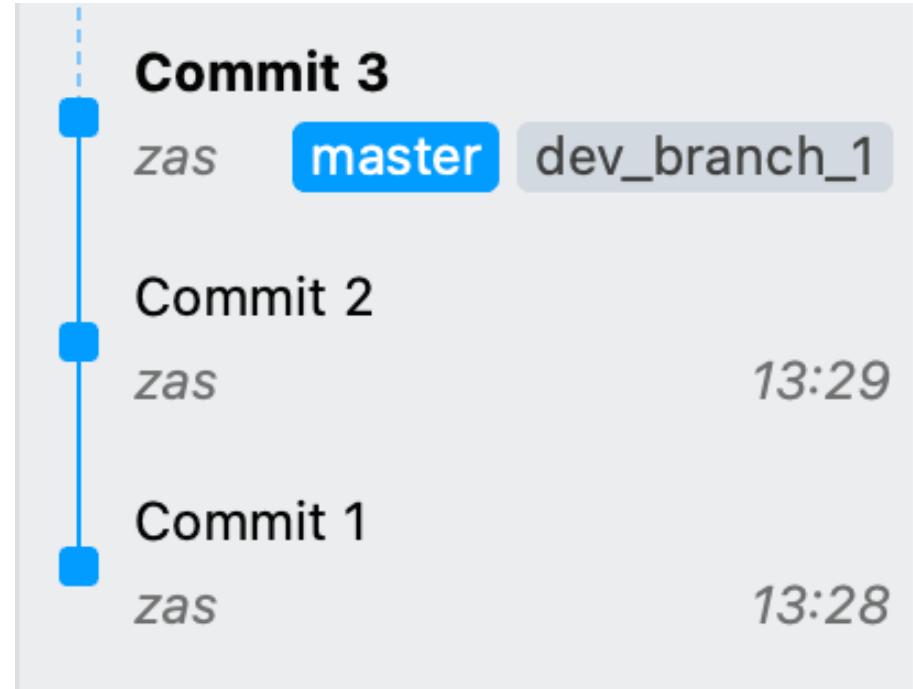


Branch and Merge

Create branch dev_branch_1

Create branch dev_branch_1

```
$ git branch dev_branch_1
```





Branch and Merge

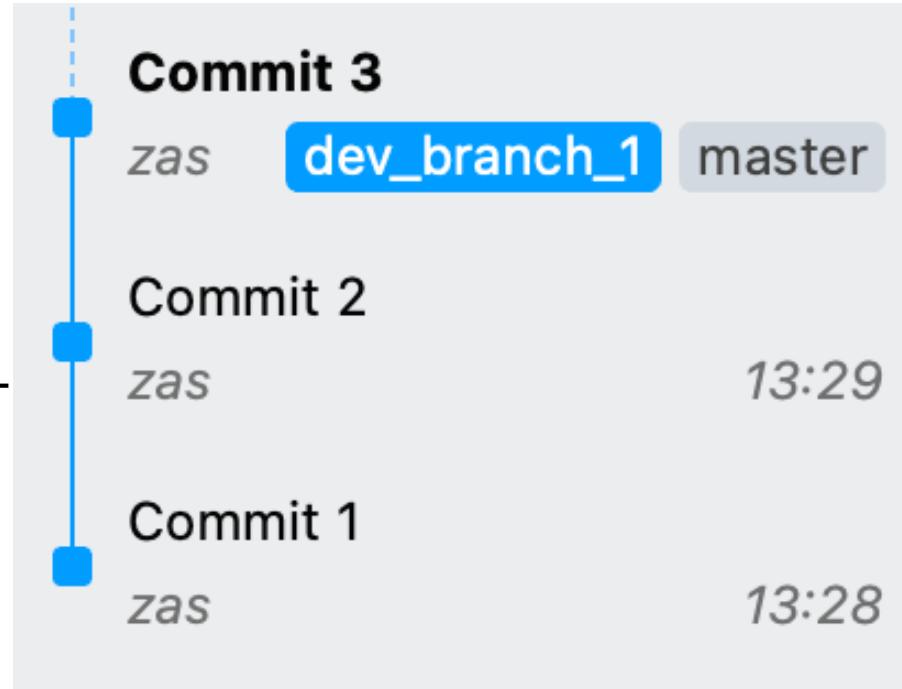
Checkout branch dev_branch_1

Check on which branch we are on

```
$ git branch
```

Checkout branch dev_branch_1

```
$ git checkout dev_branch_1
```





Branch and Merge

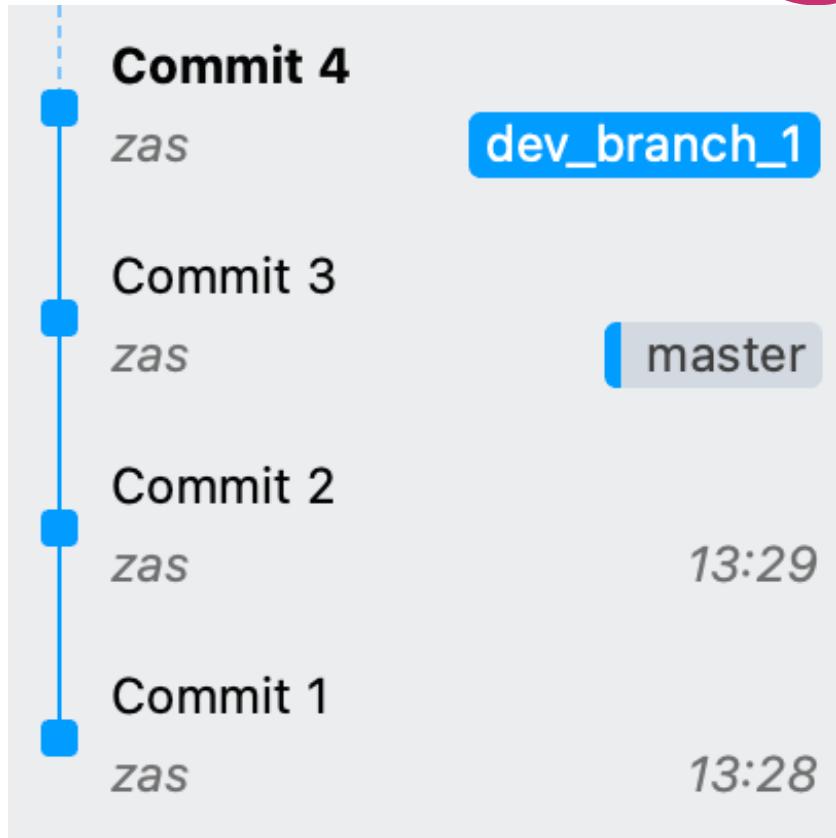
Commit on dev_branch_1

Stage new file

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

Commit stages files

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





Branch and Merge

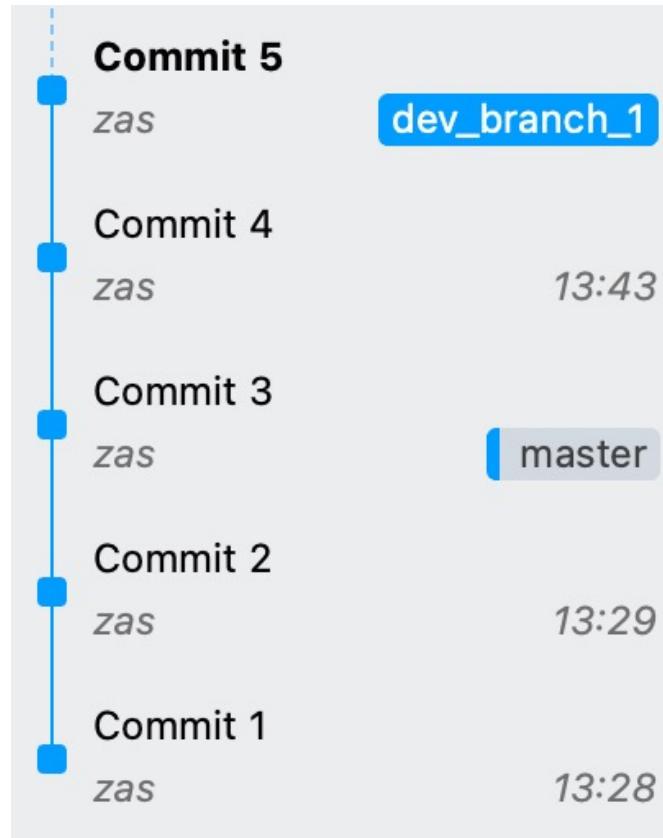
Commit on dev_branch_1

Stage new file

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

Commit stages files

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





Branch and Merge

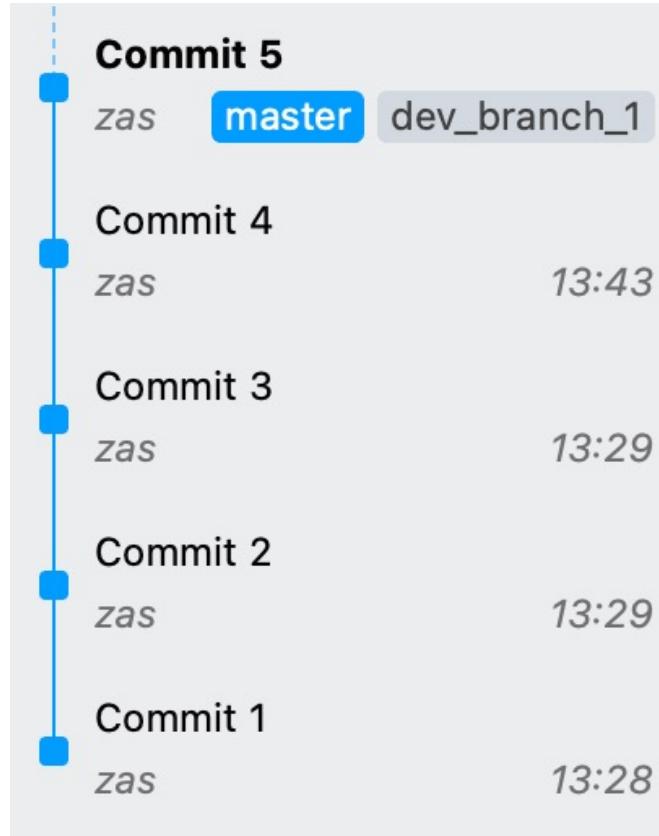
Merge master and dev_branch_1

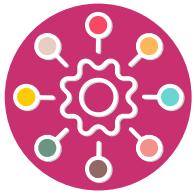
Checkout master branch

```
$ git checkout master
```

Merge dev_branch_1 into master

```
$ git merge dev_branch_1
```





Branch and Merge

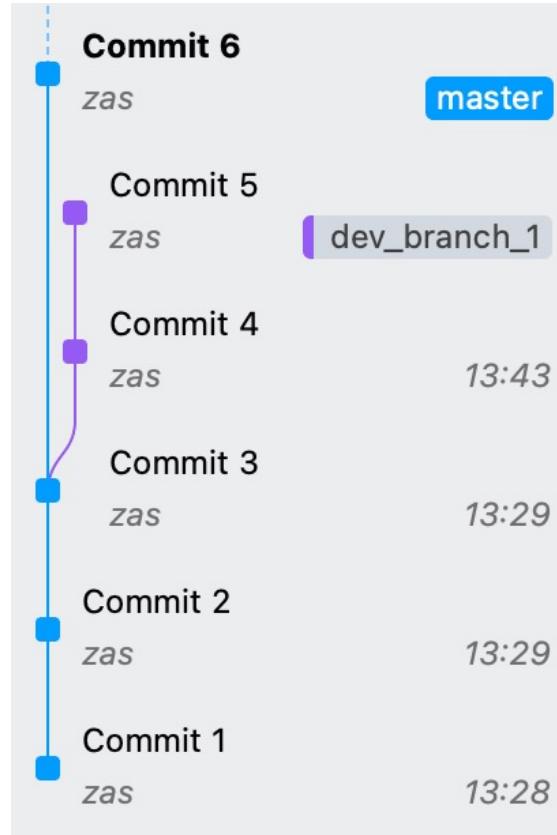
Commit on master branch

Stage new file

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

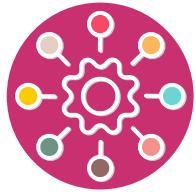
Commit stages files

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



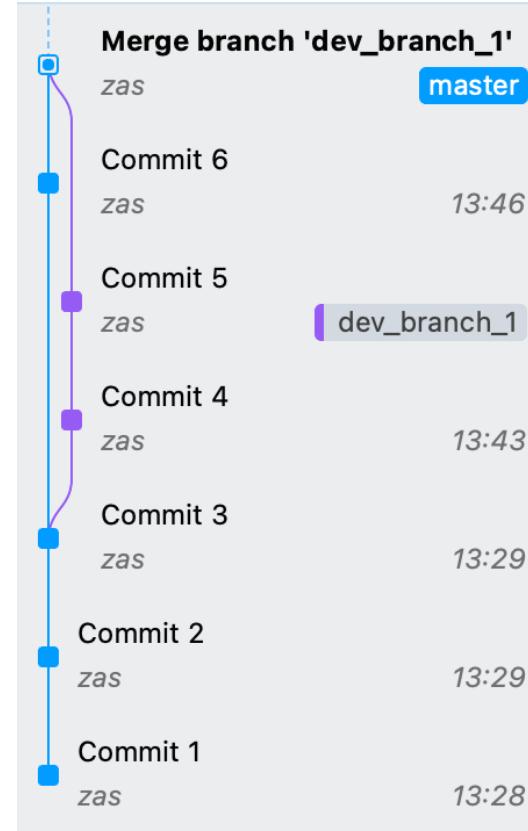
Branch and Merge

Three way merge master and dev_branch_1

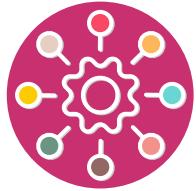
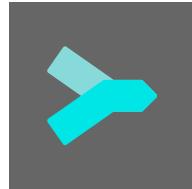


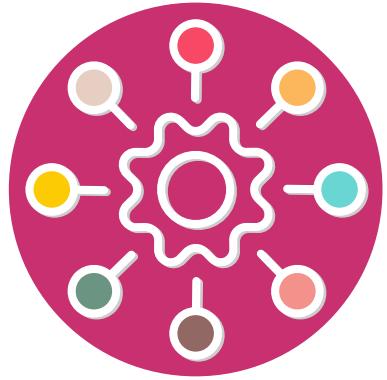
Merge dev_branch_1 into master

```
$ git merge dev_branch_1
```



Demo

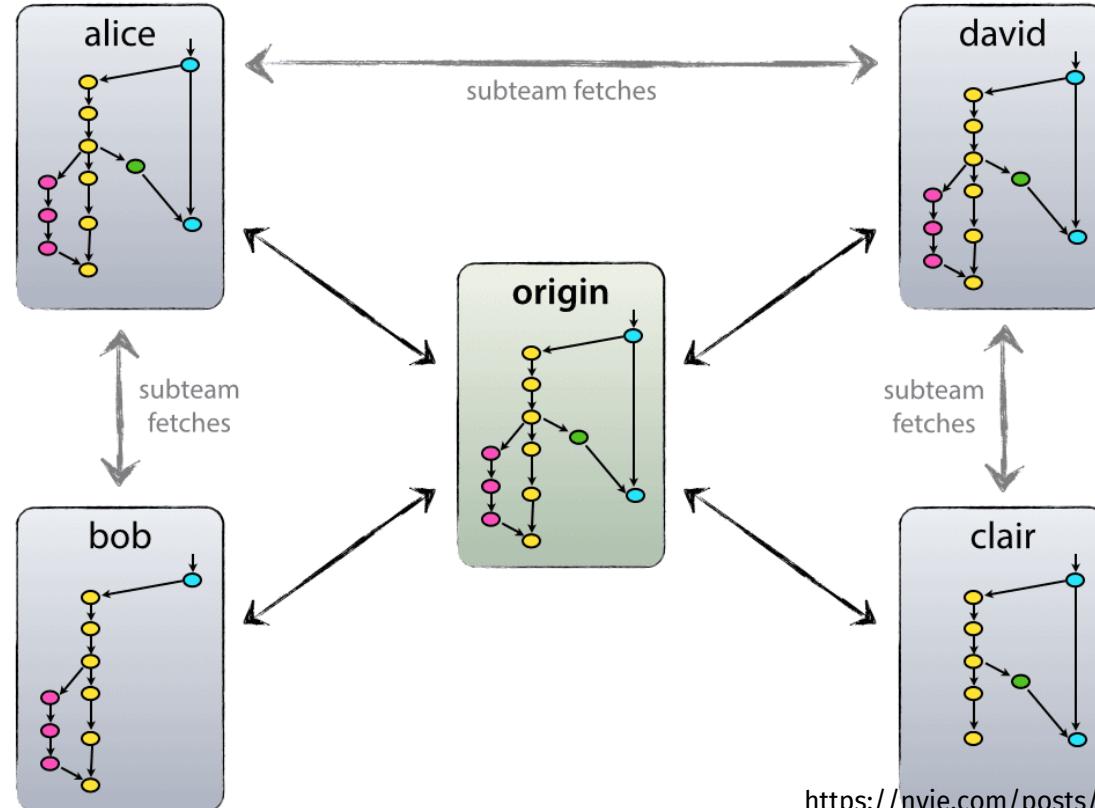




Gitflow

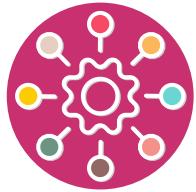
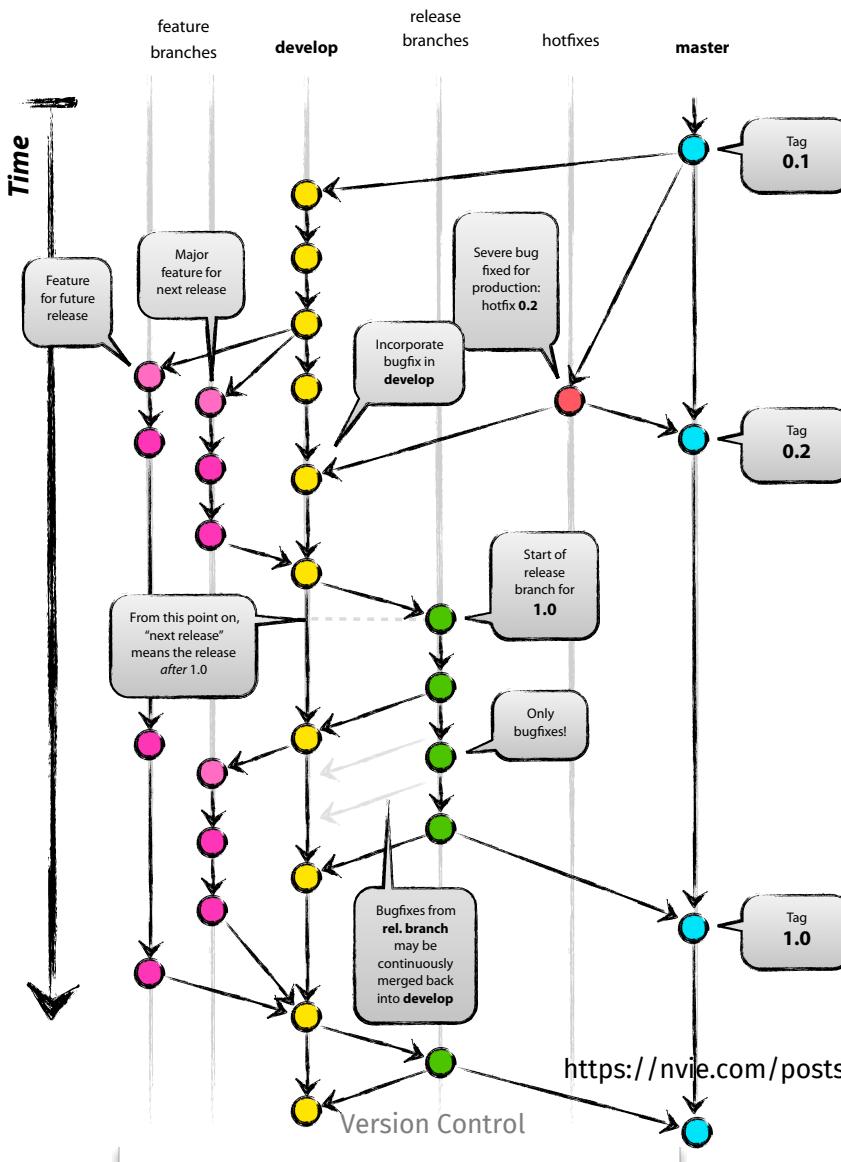


Gitflow Collaboration

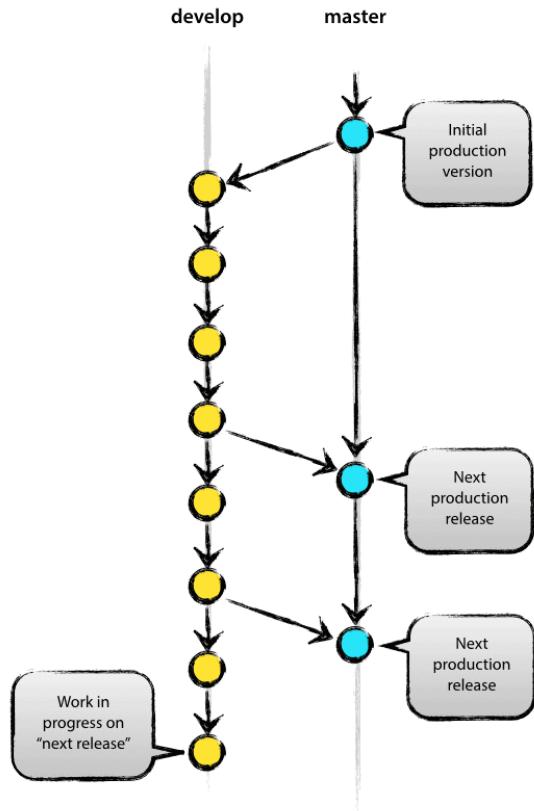


<https://nvie.com/posts/a-successful-git-branching-model/>

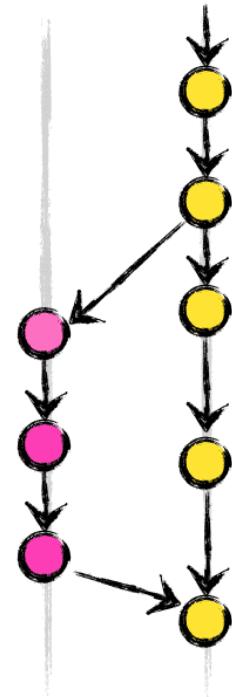
Gitflow



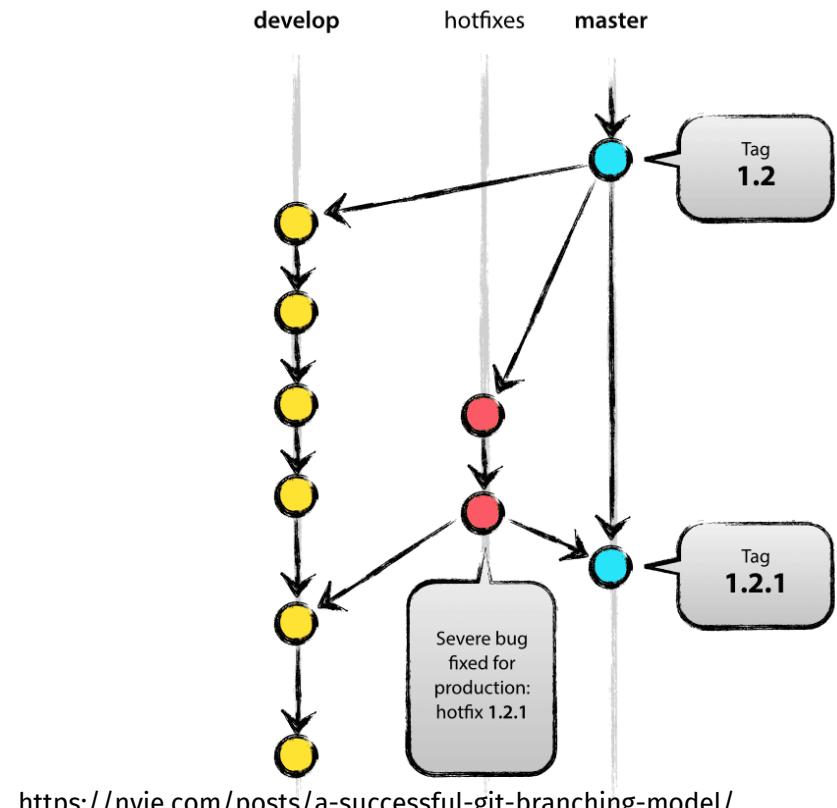
Gitflow Branching



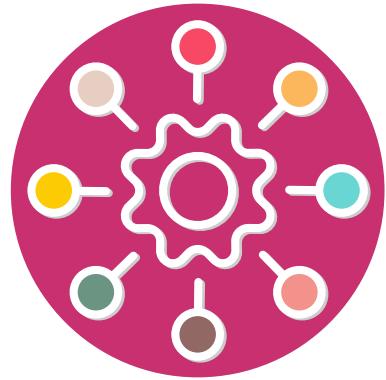
feature
branches



develop

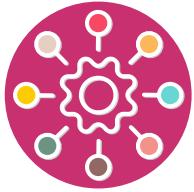


<https://nvie.com/posts/a-successful-git-branching-model/>



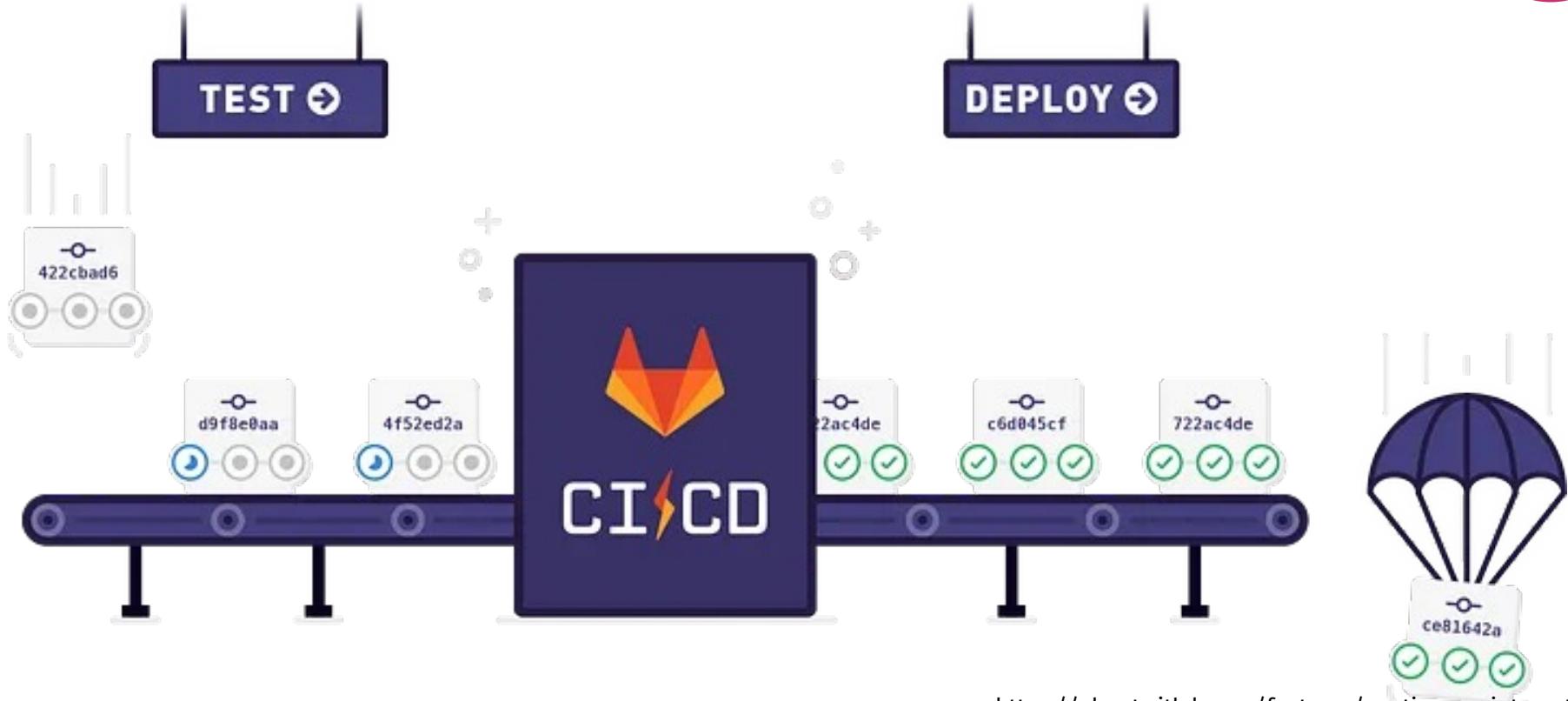
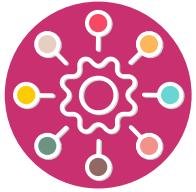
Git CI/CD

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?

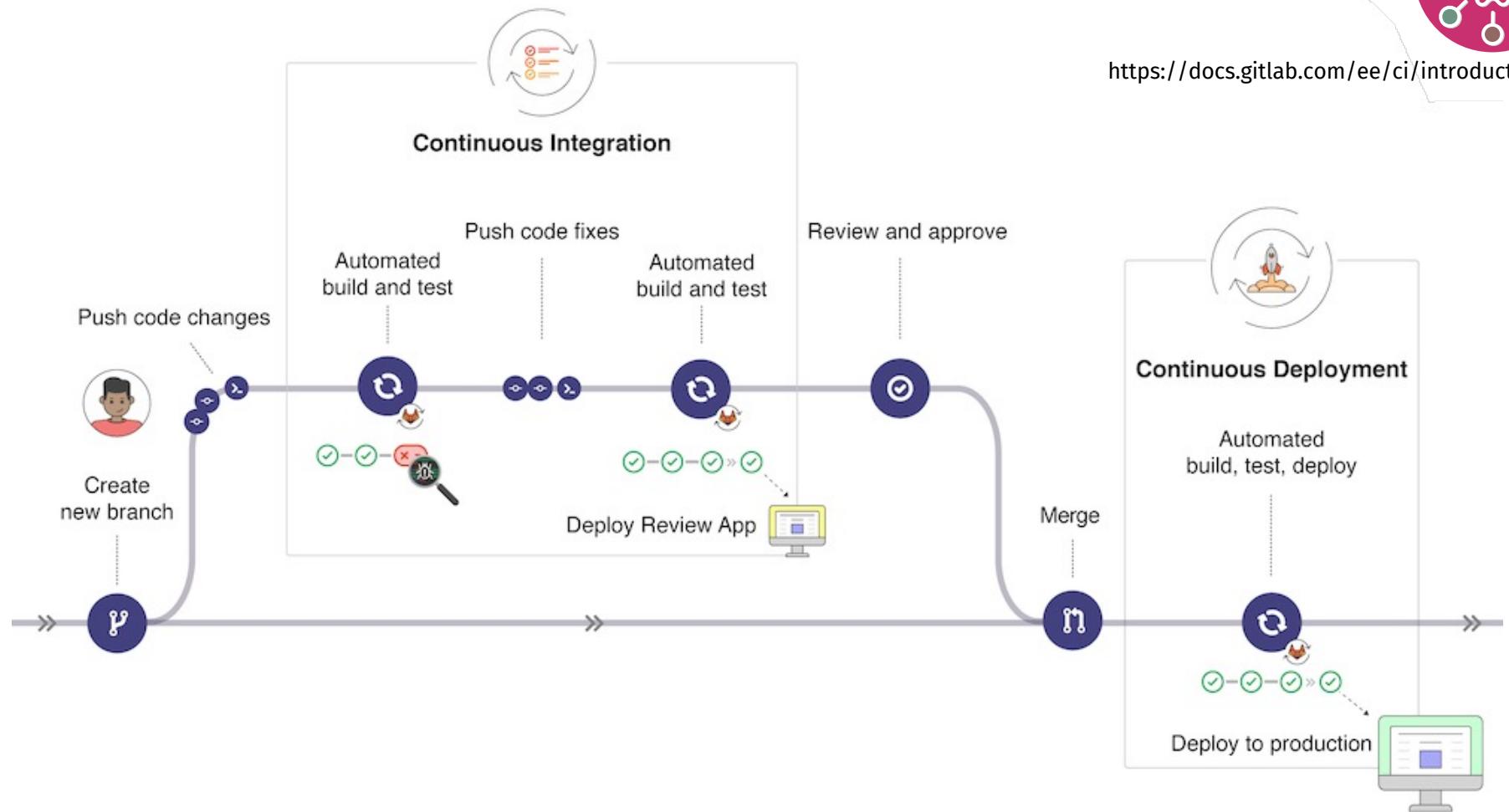


<https://about.gitlab.com/features/continuous-integration/>

Gitlab Workflow

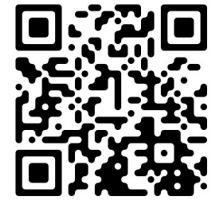


<https://docs.gitlab.com/ee/ci/introduction/>



Task Estimation

Estimation 4



Create a Star Wars
Star Destroyer (4784pcs)

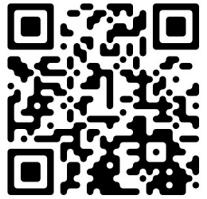
X Point(s)



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

Task Estimation

Estimation 5



Create a Star Wars
Star Destroyer (37pcs)

X Point(s)



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



SYSTEMS DESIGN / INTRODUCTION TO GIT - PART A

Introduction to git - Part A

Installation & Setup



SYSTEMS DESIGN / INTRODUCTION TO GIT - PART B

Introduction to git - Part B



Contents

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

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