

```
    setOnLongClickListener(this)  
}
```

# Git & CI/CD

CPE -2023



# Planning

	Lundi	Mardi	Mercredi	Jeudi
AM	Cours	Cours	Cours	
PM		Guide TD		

Lundi 20/02 15h45-17h45
Examen écrit

# Git





Git

# GIT is a VCS but what is a VCS?

1. Veritas Cluster Server, a commercial high-availability computer clustering system
2. Version Control System, a system for managing multiple revisions
3. Venture CapitalS, specials investment funds
4. Vietnamese Council of State, the highest standing body of the National Assembly of Vietnam



Git

# When is GIT born?

1. 1998, Hugo Chávez is elected President of Venezuela
2. 2002, Salt Lake city winter olympic games
3. 2005, Pope John Paul II dies
4. 2008, Satoshi Nakamoto publish "Bitcoin: A Peer-to-Peer Electronic Cash System"



# Who invented GIT?

1. Linus Torvalds, historical inventor of the linux kernel
2. Rod Johnson, programmer and Spring Framework creator
3. Larry Page, Google co-founder
4. Marie Curie, pioneer researcher in radioactivity, first woman to own a Noble Price and only person to win a Nobel Prize in two different sciences (Physics & chemistry)



# What is git internal checksum algorithm?

1. None, no one needs to check file/system integrity
2. MD5, producing vulnerable 128-bit hash value, published in 1992
3. SHA-1, producing vulnerable 160-bit hash value, published in 1995
4. SHA-256, NSA secured 256-bit hash value, published in 2001

# Pourquoi Git ?



Git

# Pourquoi git?

Avant les gestionnaires de version ...



Avec Git

- Collaborative
- Versionnage de projet avec des commits immutables. (historique de commits avec la date et l'utilisateur qui a fait les changements)
- Résolution de conflits

Sans git

# Gestions des versions

-  TravailCollectif-part1.pdf
-  TravailCollectif-part2&3.pdf
-  TravailCollectif-part2&3&1&4-corrigéeSaufLa4.pdf
-  TravailCollectif-part2&3&1&4-V0.pdf
-  TravailCollectif-part2&3&1-corrigée.pdf
-  TravailCollectif-part2&3&1-v0.pdf
-  TravailCollectif-part4.pdf
-  TravailCollectif-part4-corrigée.pdf
-  Rapport de stage -v0.pdf
-  Rapport de stage -v1.pdf
-  Rapport de stage -vf.pdf
-  Rapport de stage -vf-final.pdf
-  Rapport de stage -vf-final (copy).pdf
-  Rapport de stage -vf-final-final.pdf
-  Rapport de stage -vf-final-final-final-derDesder.pdf

# Basic notions



# Pourquoi git?

A working directory avec la version actuelle

Commit = unit of change

Snapshots pas des diffs !

.git folder contient toutes les git métadonnées:

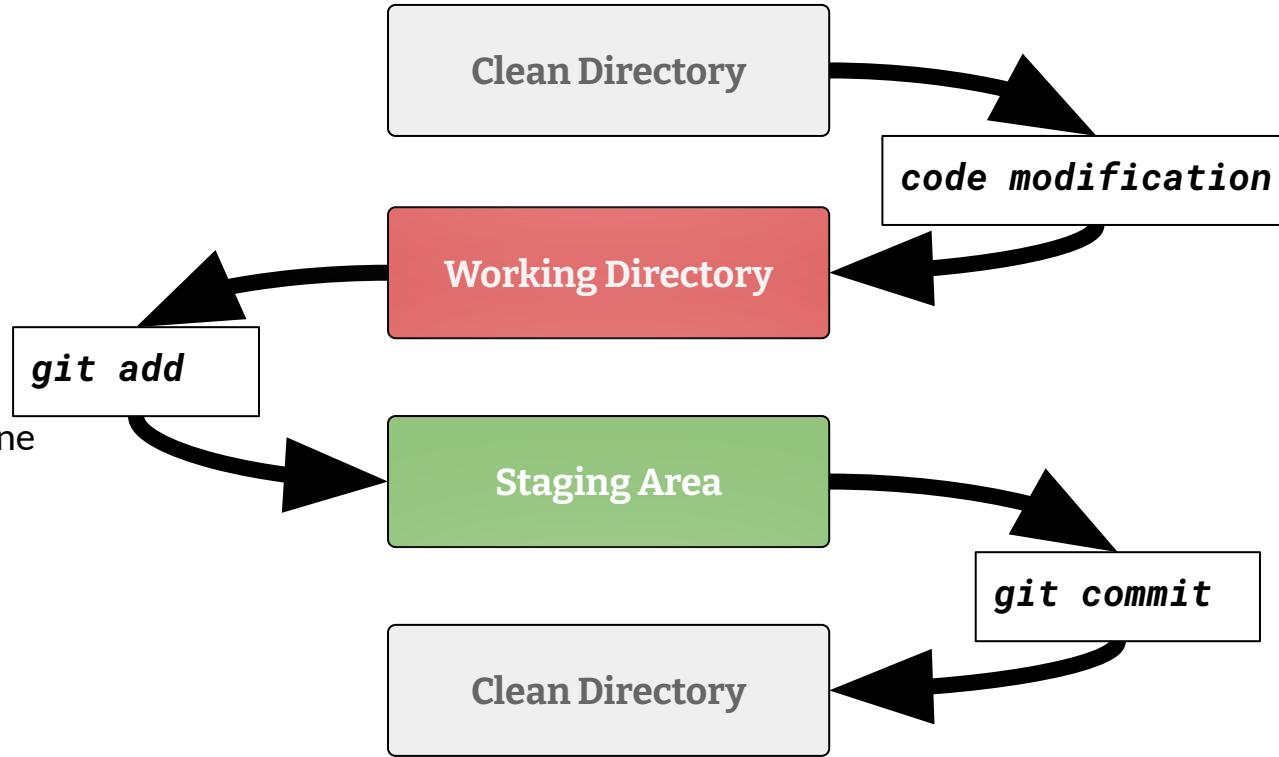
- Snapshots (blob, tree...)
- Commits
- ...



Git

# Pourquoi git?

- Aucun changement depuis le dernier commit
- Commit = unité de changement
- Quelques changements mais git ne s'en préoccupe pas
- Toujours quelques changements mais git s'en préoccupe
- Changements sauvegardés



# Inspect git

Keep an eye open. And the good one!



Git

# Inspection command

Working directory changes details

- `git diff`

Staging area informations:

- `git status`

Commit history:

- `git log`
- `git log --graph --decorate --pretty=oneline --abbrev-commit --all`



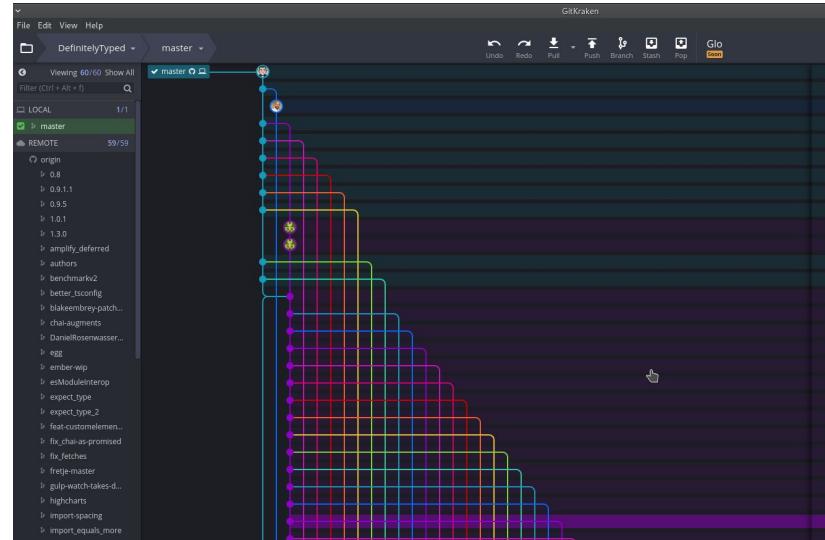
Git

# Useful tools

## Textual interface for git: TIG

```
tig --all 211x58
2018-03-22 16:46 nraymond
o [master] {origin/master} (origin/HEAD) Fix the type of the
2018-03-21 16:03 Armando Aquirre
2018-03-21 22:19 Juan Jimenez-Anca
2018-03-09 11:21 Juan Jimenez-Anca
2018-03-21 14:44 Mine Starks
2018-03-20 15:52 Tiger Oakes
2018-03-20 15:43 Tiger Oakes
2018-03-21 14:16 Mine Starks
2018-03-21 11:37 Pierre-Olivier Bédard
2018-03-21 09:54 Mine Starks
2018-03-19 22:24 Vajkay René
2018-03-18 17:16 Vajkay René
2018-03-18 15:46 Vajkay René
2018-03-18 15:07 Vajkay René
2018-03-21 09:42 Mine Starks
2018-03-21 12:53 Frank Schulz
2018-03-21 12:37 Frank Schulz
2018-03-21 09:42 Mine Starks
2018-03-21 09:42 Mine Starks
2018-03-13 02:30 s0m3n3
2018-03-13 11:11 s0m3n3
2018-03-14 21:58 s0m3n3
2018-03-21 09:30 Mine Starks
2018-03-20 15:13 Kelvin Jin
2018-03-21 09:29 Mine Starks
2018-03-20 21:15 Luke Andrews
2018-03-21 09:28 Mine Starks
2018-03-20 17:32 lincoln
2018-03-20 16:56 lincoln
2018-03-21 09:27 Mine Starks
2018-03-19 10:48 Adam Laycock
2018-03-21 09:26 Mine Starks
2018-03-19 14:34 Igor Polishchuk
2018-03-19 14:26 Igor Polishchuk
2018-03-19 14:17 Igor Polishchuk
2018-03-21 09:19 Mine Starks
2018-03-17 16:02 kobanyan
2018-03-17 16:01 kobanyan
2018-03-17 16:00 kobanyan
2018-03-12 01:26 kobanyan
2018-03-21 09:40 Mine Starks
M Merge pull request #24179 from cortopy/ethereumjs-util
o tslint without extra rules
o create ethereumjs-util
M Merge pull request #24421 from NotWoods/three-0.91
o Updated triangle static methods
o Added migration changes for 0.89 -> 0.91
M Merge pull request #24447 from pobed2/patch-1
o Add 'extensions' to GraphQLError
M Merge pull request #24362 from vajkayrene/master
o Adds recent fields from UglifyJSPlugin
o Sets TS version of the dependency and fixes lint i
o Adds missing TSC option 'strictFunctionTypes'
o Adds type support for webpack's 'uglifyjs-webpack-
M Merge pull request #24440 from frankschulz/maste
o Add tests
o Fix https type
M Merge pull request #24299 from ciferox/adone
o [adone] fs * + openbind, alx
o [adone] fs * + read writeFileAtomic * move is.* t
o [adone] add lodash, benchmark
M Merge pull request #24422 from kjbin/node-ah-
o [node] Bump async_hooks definitions to v8.10
M Merge pull request #24427 from ellipsis-ai/
o Added alternate method signature for adddu
M Merge pull request #24402 from dynamsoft
o change format & test passed
o add interfaces
M Merge pull request #24375 from Arcath/sc
o update screeps to 2.2.2
M Merge pull request #24377 from amadare
o added newline
o fixed compilerOptions.files typo
o Fixed CreditCardTypeChangeHandler typ
M Merge pull request #24228 from koban
o [sinon-chrome] Follow the browser sp
o [sinon-chrome] Make translations par
o [sinon-chrome] Remove Plain interfa
o [sinon-chrome] Support plugin
M Merge pull request #24243 from Fla
```

## Graphical Interface for GIT: GitKraken, gitk, gitg...



# Premières étapes

# S'authentifier

Dans votre working directory :

```
> git config --global user.name jdalton  
> git config --global user.email jdalton@takima.fr
```

Pourquoi ?



# Pas (encore) un projet Git

Dans ton projet :

```
> git init
```

Qu' observes-tu ?



Display hidden files ...

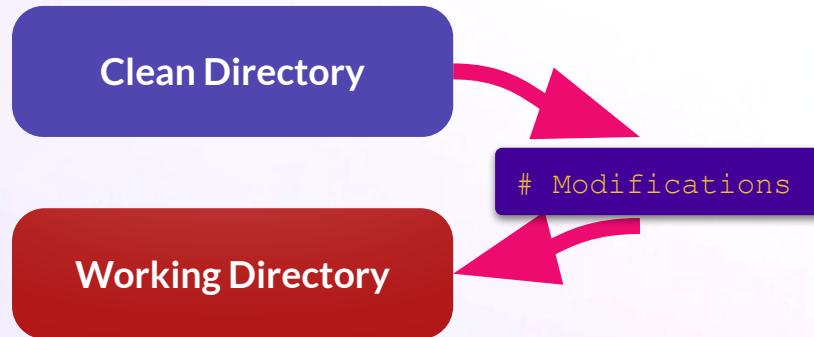
# Initialement

Clean Directory

Pas de modification depuis le dernier commit



# Faire des modifications



Quelques modifications ont été faites, mais git n'en prends pas compte



# git add

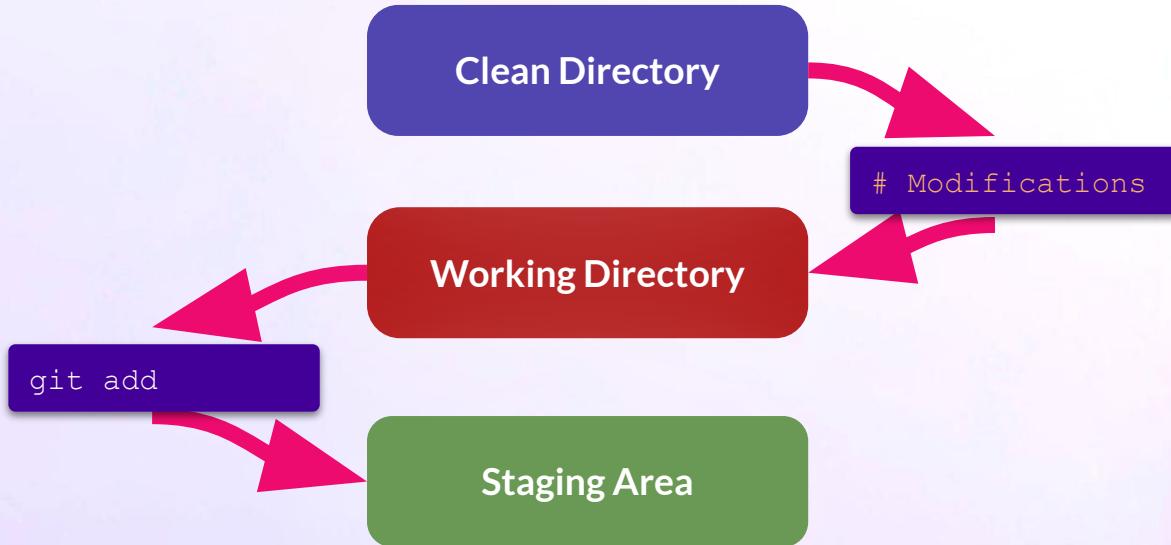
```
> git add myFile.java
```

```
> git add .
```

```
> git add -A
```



# Staging area



Les modifications ne sont toujours enregistrées  
mais Git en prend compte



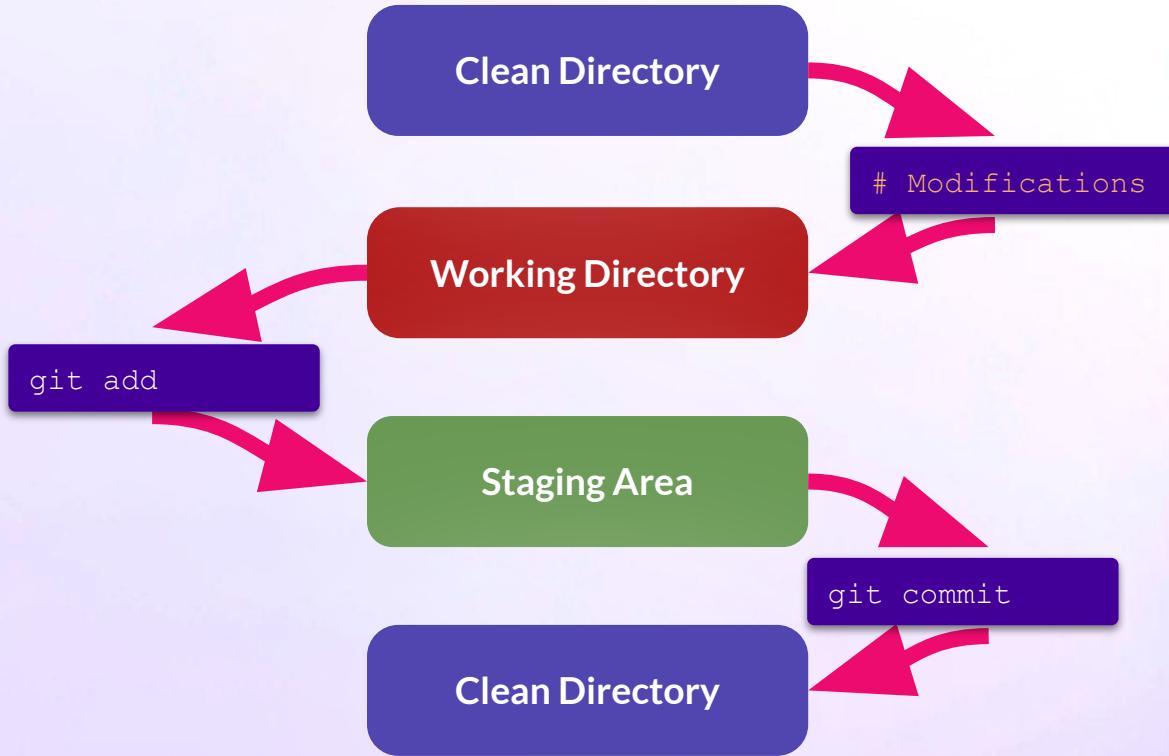
# git commit

```
> git commit -m "feature: Add function that do something"
```

```
> git commit  
# ouverture de l'éditeur pour rédiger le message du commit
```



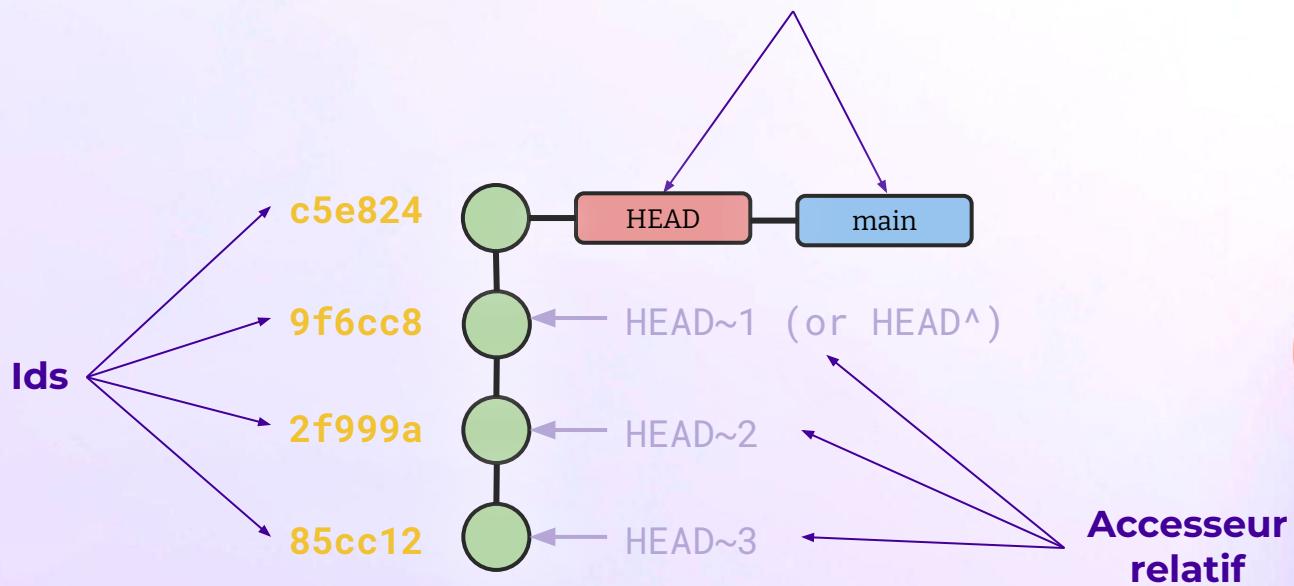
# Modifications enregistrées



# Branches

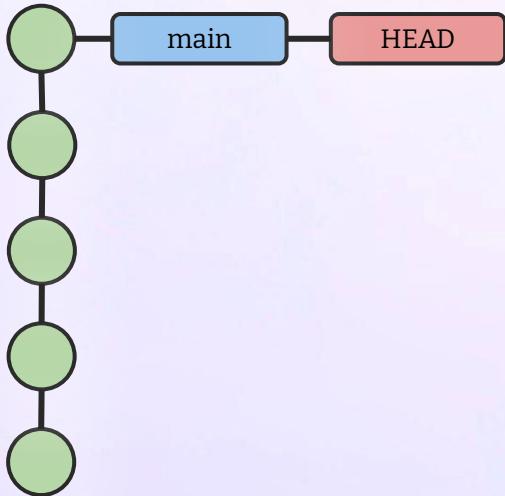
# Désigner un commit

Références qui pointent sur le commit



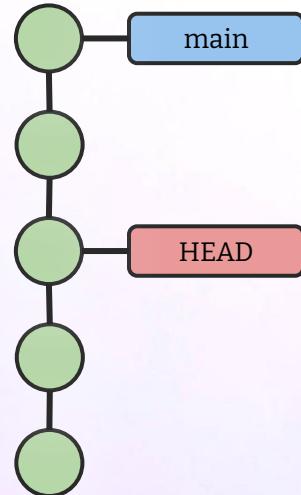
# HEAD

Attachée :



```
> git checkout main
```

Détachée :



```
> git checkout main~2
```



# Forking



# Forking

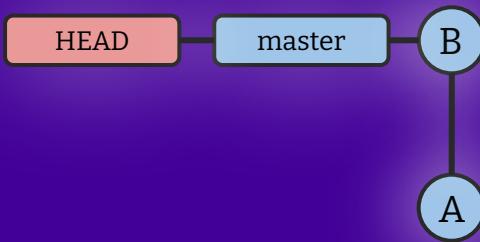
## Command Line

```
# We will reuse the last git log  
# as an example...
```

## Working directory

```
my-project  
└── README.md  
    └── src  
        └── hello.java
```

## Git log





Git

# Forking: Créer une branche

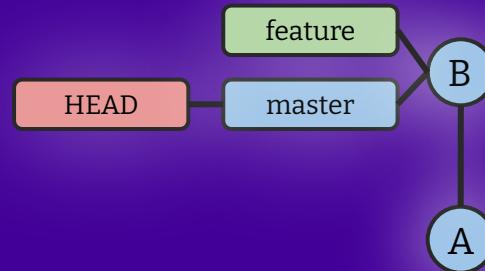
Command Line

```
git branch feature
```

Working directory

```
my-project
├── README.md
└── src
    └── hello.java
```

Git log





Git

# Forking: Checkout une autre branche

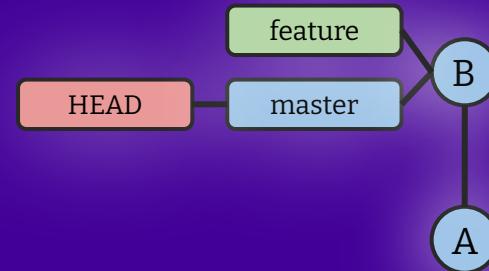
## Command Line

```
git checkout feature  
# or "git checkout -b feature" to also create the  
branch
```

## Working directory

```
my-project  
└── README.md  
    └── src  
        └── hello.java
```

## Git log





Git

# Forking: Creer un commit sur une branche

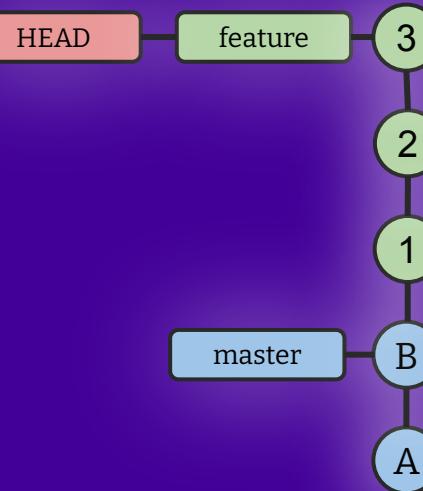
## Command Line

```
# Create feature.java file  
git add src/feature.java  
git commit -m "1"  
# Create 2 more commits...
```

## Working directory

```
my-project  
└── README.md  
└── src  
    └── hello.java  
        └── feature.java
```

## Git log





Git

# Forking: Checkout master

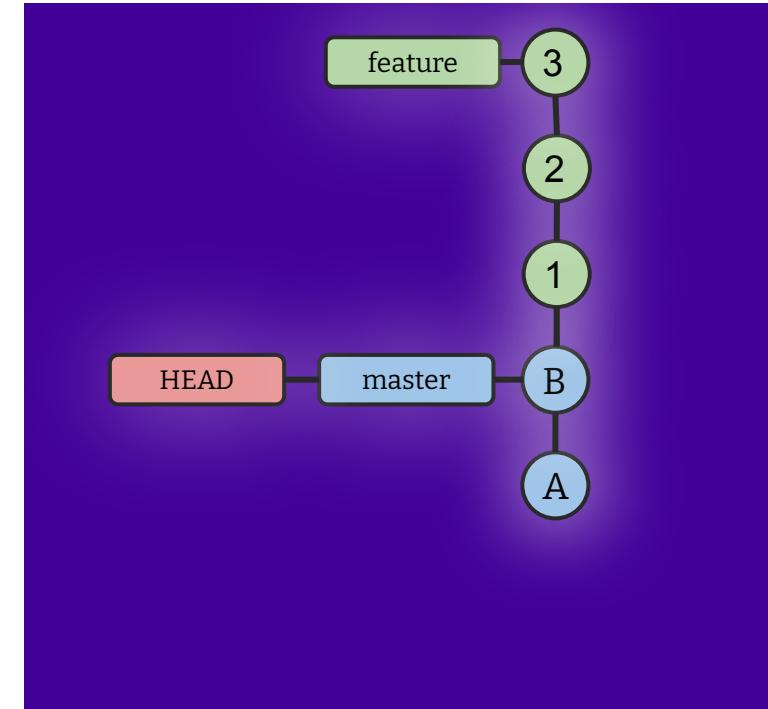
Command Line

```
git checkout master
```

Working directory

```
my-project
├── README.md
└── src
    └── hello.java
```

Git log



# CI/CD

DevOps workflows made easy



 CI/CD  
**CI/CD?**

# Avant CI/CD

Lent cycle de releases

- Des semaines voir des mois ...
- Perte de temps

Conflits le jour du “merge day”

- Des bugs à résoudre
- Tester/deployer manuellement

# Pourquoi?

- Déployer ses versions à la mains
- Tester la qualité de son code en local
- Une release tous les 6 mois
- Des conflits à n'en plus finir

DEVOPS ⇒ Une personne qui **déteste** faire 2 fois la même chose à la main.

# Continuous Integration and Continuous Delivery

Automate your problems away



With Github



Github  
Actions



SonarCloud

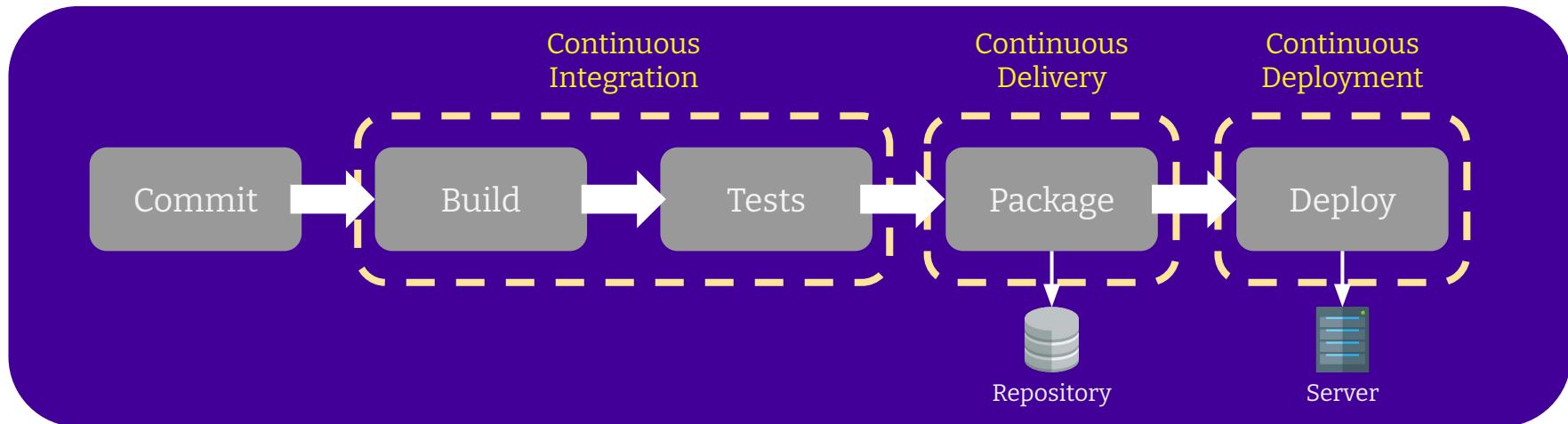
# Maintenant CI/CD

## Continuous Integration (CI)

- Regular integration => PR et merge
- Build/tests automatisés

## Continuous Delivery/Deployment (CD)

- Mettre les artefacts sur un repo
- Automatique déploiement en production



# Pour et Contre



?

# Pour et Contre

- ✓ Releases fréquentes et plus facile
- ✓ Constant feedback
- ✓ Rapide découverte de bugs
- ✓ Plus de temps pour se consacrer aux bugs
  
- ✗ Besoin d'une mentalité DevOps
- ✗ Un temps initial de setup

# Outils

- Github Actions
- Travis CI
- Jenkins
- Gitlab CI
- Circle CI
- ...

# Intro to Github & Github Actions

# Introduction

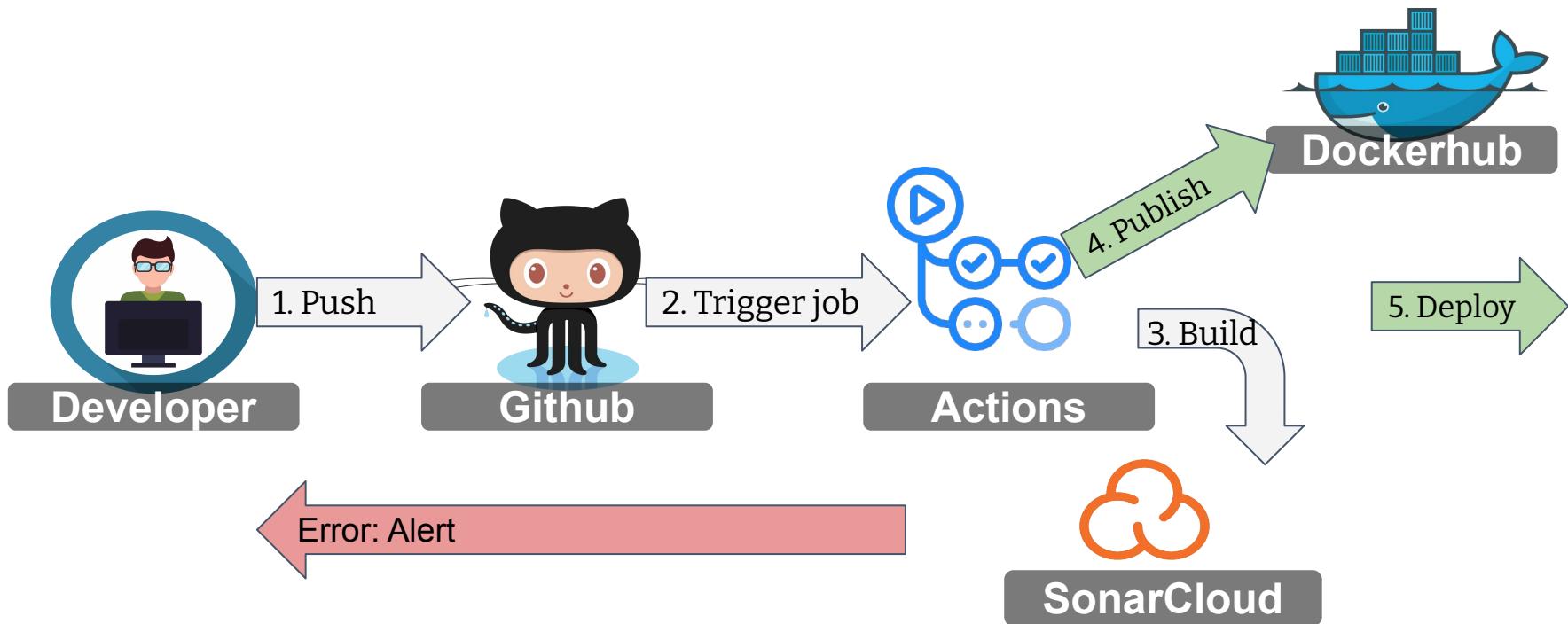
- Github
  - Fondé en 2008 - @Microsoft 2018
  - Git repository manager
  - Github Enterprise
- Github Actions
  - La réponse de github à gitlab-ci
  - (!) Seulement Github
  - Linux, macOS & Windows

# Pourquoi Github Actions ?

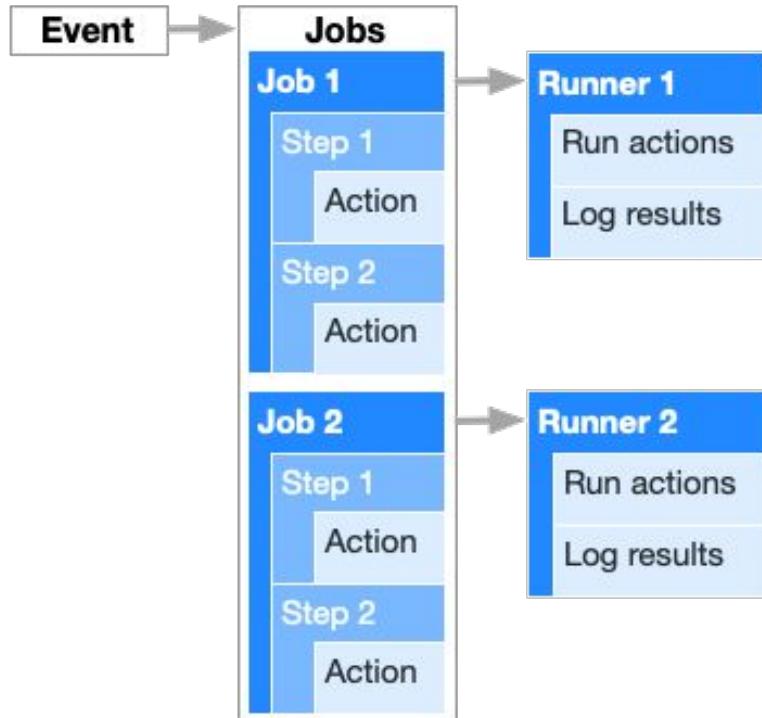
- ✓ Full Github integration
- ✓ Simple installation
- ✓ Declarative configuration
- ✓ Open source
  
- ✗ Full Github integration
- ✗ Des features payants

# Github Actions

# Comment ça marche?



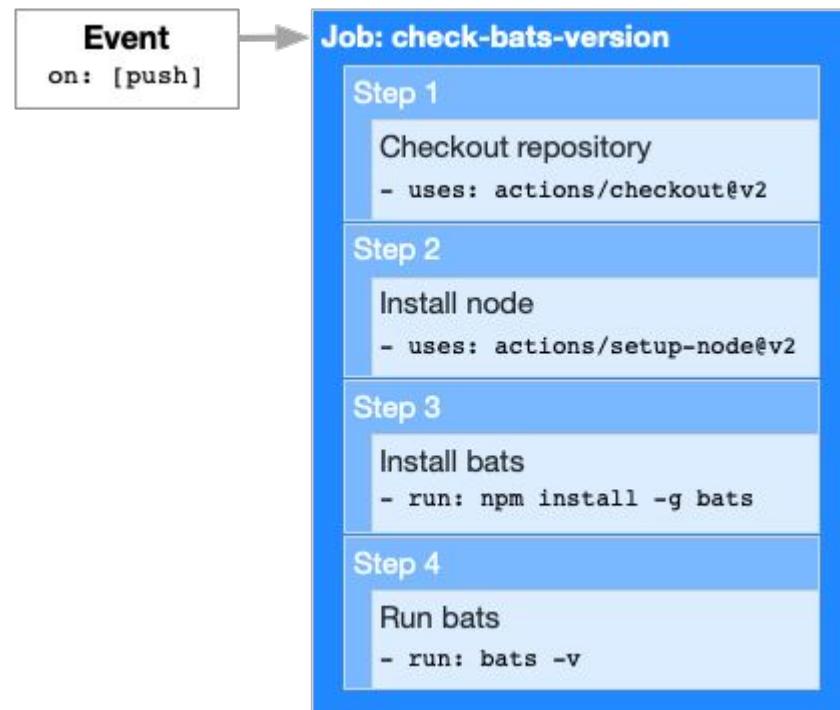
# Comment ça marche?



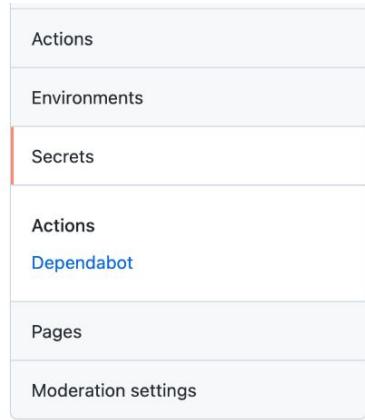
- ❖ Workflow
- ❖ Event
- ❖ Jobs
- ❖ Step
- ❖ Action
- ❖ Template

# .main.yml

```
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v2
        with:
          node-version: '14'
      - run: npm install -g bats
      - run: bats -v
```



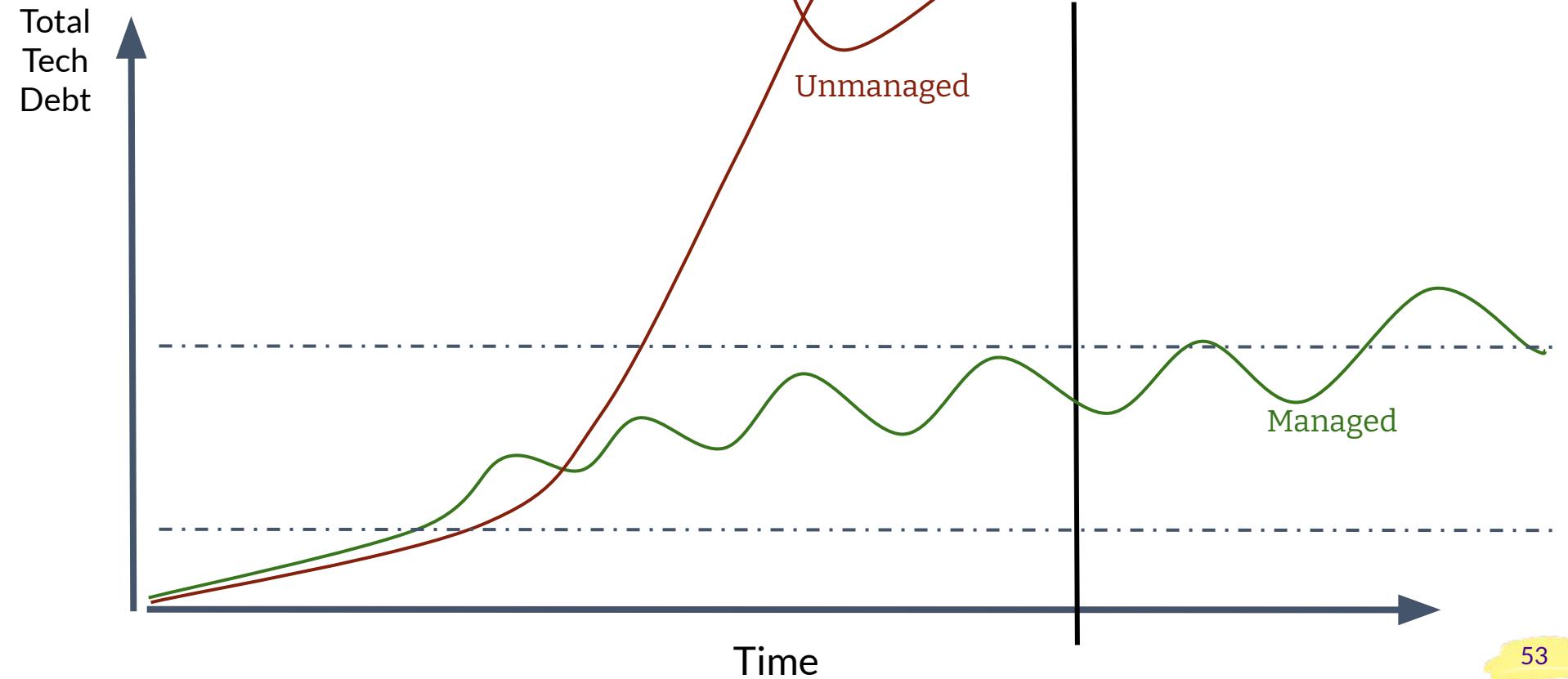
# Secrets et variables



Repository secrets			
	DOCKERHUB_TOKEN	Updated 8 hours ago	<button>Update</button> <button>Remove</button>
	DOCKERHUB_USERNAME	Updated 8 hours ago	<button>Update</button> <button>Remove</button>
	SONAR_TOKEN	Updated 2 hours ago	<button>Update</button> <button>Remove</button>

# Sonar Cloud

# Dette technique



# SonarCloud: qualité

- Analyse de qualité
- 20+ different languages
- Test coverage & execution
- Vérification des bugs & code smells
- Et plus encore

# Take Away

- CI/CD automatise tout du commit à la mise en prod
- Github Actions est une bonne option mais n'est pas la seule
- Ne pas mettre de donnée sensible dans ton `main . yml`
- Utiliser un outil de qualité de code

Enough speaking.  
Take me to the code !

<http://school.pages.takima.io/devops-resources/>

TD02 - git & Github