

L'importance du CI/ CD dans un projet de développement



Présentation



Thomas Boni

CTO @ Go2Scale



@thomasboni



thomas@go2scale.com





01

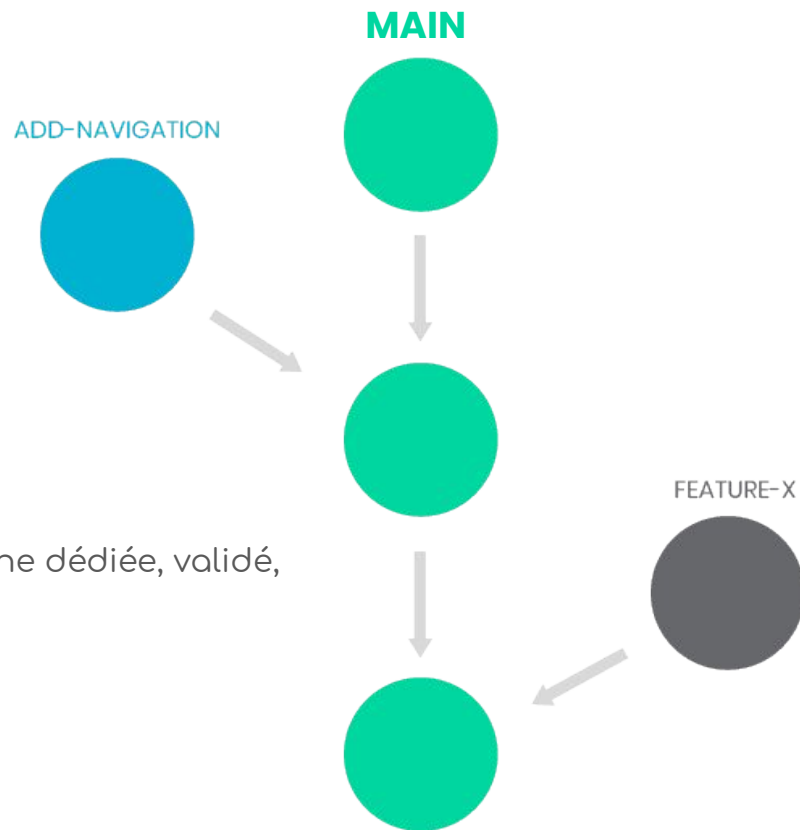
**Qu'est-ce
que le CI/CD ?**

Workflow de développement

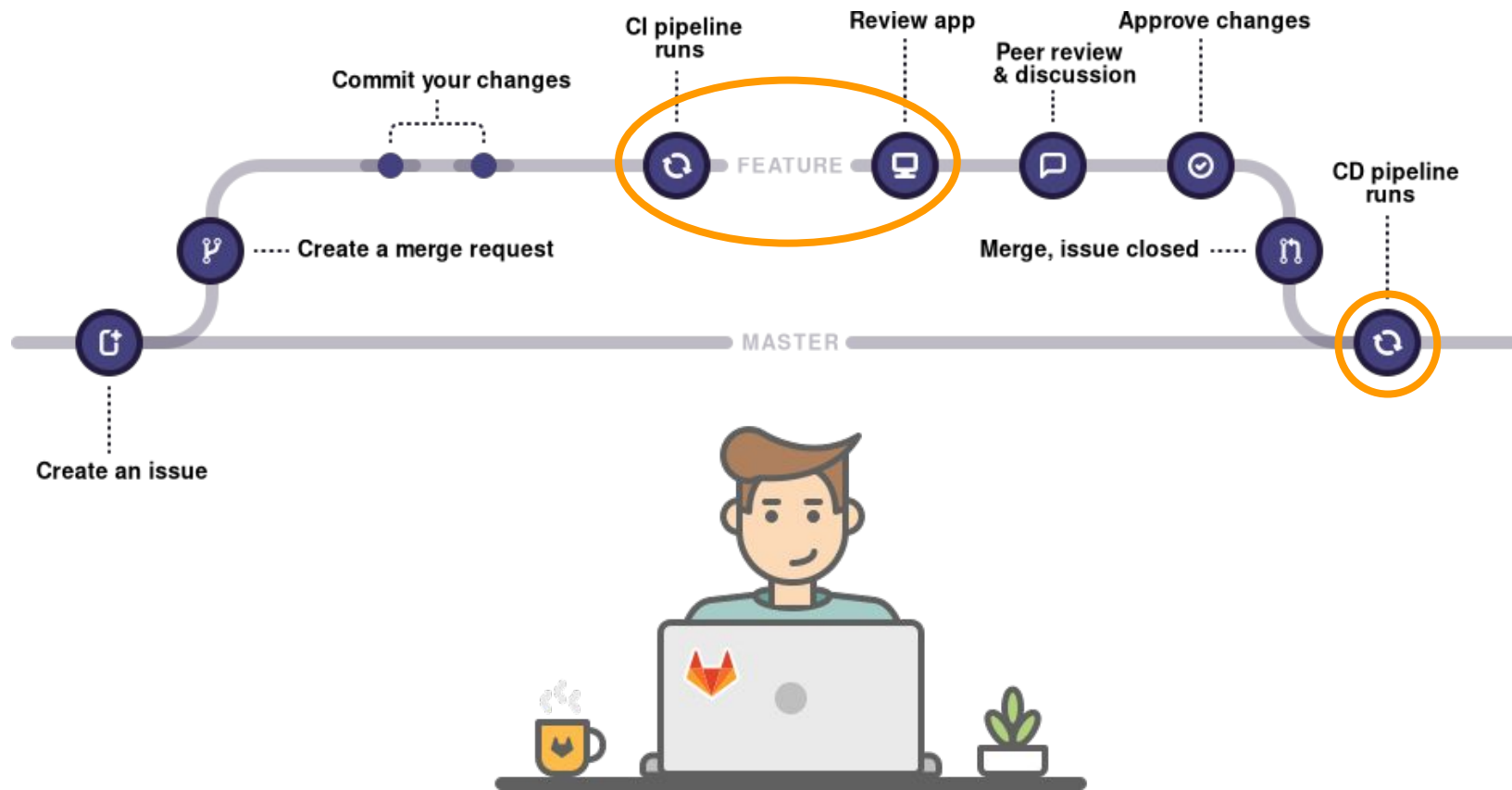
La branche principale - Main

- est la branche par défaut
- devrait toujours être valide

Chaque développement est effectué dans une branche dédiée, validé, puis mergé dans la branche principale (main).



Workflow de développement



Les plateformes de CI/CD



Travis CI



DRONE
by harness



GitLab



GitHub



TEKTON



Jenkins



circleci



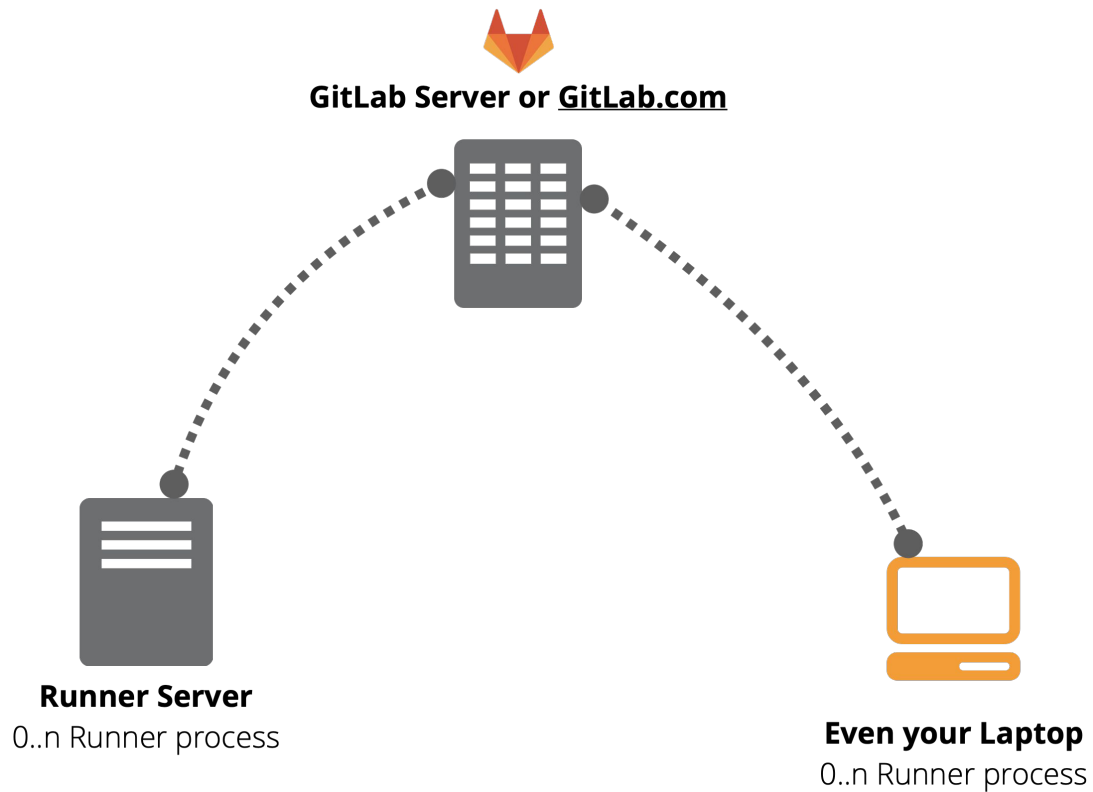


02

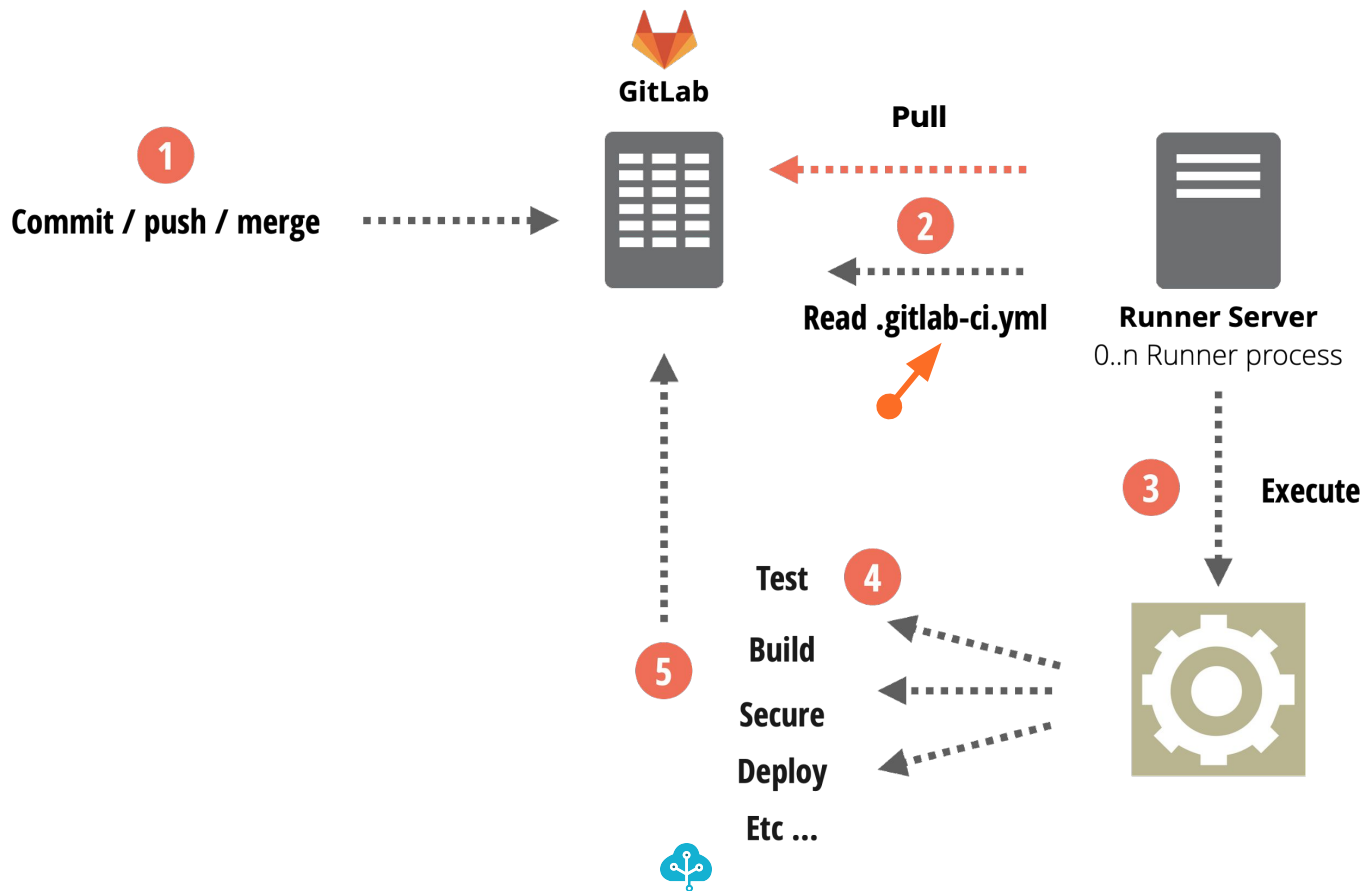


Implémentation dans GitLab

Runners



Runners : le fonctionnement



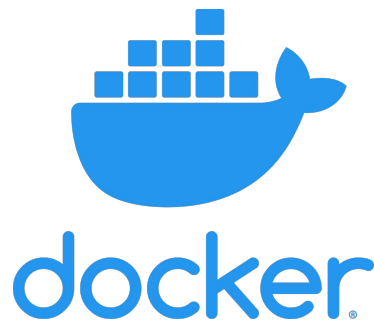
Un runner peut utiliser plusieurs "Executors"

- Shell (exécute cmd directement sur le serveur hébergeant le runner)
- Docker (exécute cmd dans un container)
- Docker Machine
- Kubernetes
- Openshift
- VirtualBox
- Parallels
- SSH (exécute cmd directement sur le serveur via ssh)
- zOS

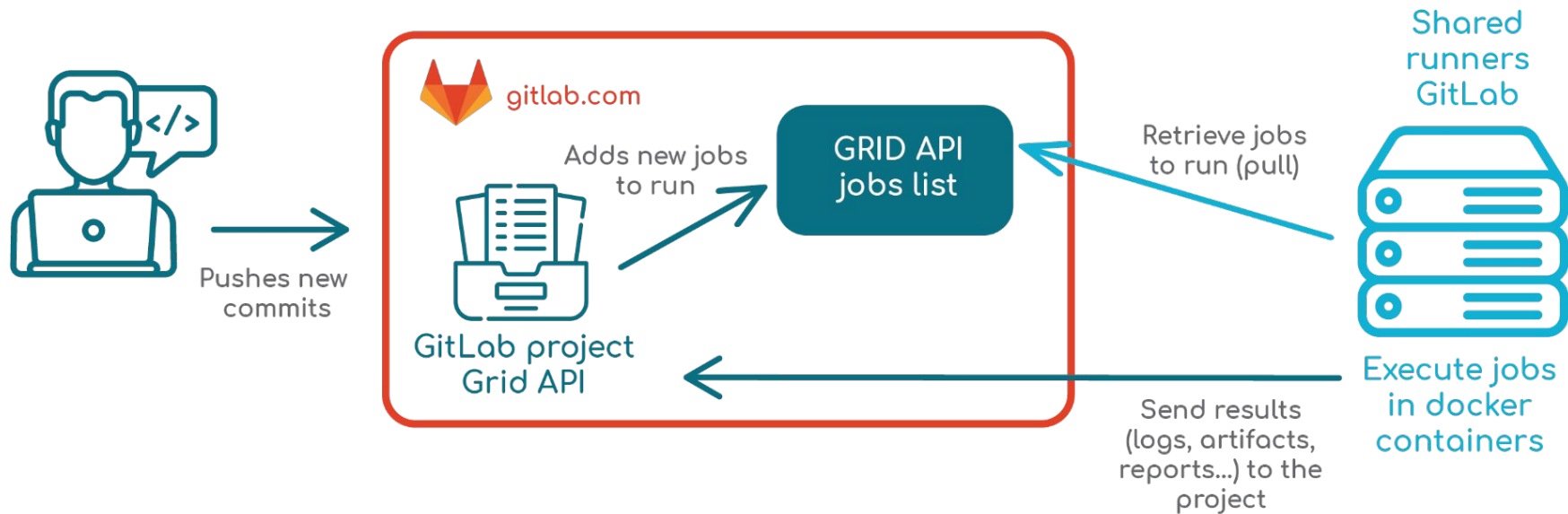


Docker

- Les bases : docs.docker.com/get-started/overview
- Permet la conteneurisation des environnements et applications
- Hub publique d'images docker : hub.docker.com
- Exécuteur le plus utilisé sur GitLab CI/CD



Big picture



Une recette pour les runners 🔍 : `.gitlab-ci.yml`

Comment ça marche ?

- Ajouter un fichier `.gitlab-ci.yml` en racine du projet
- `.gitlab-ci.yml` définit
 - une liste de stages (étapes)
 - un ensemble de jobs (commands) (\in stages)
 - (stages + jobs) == pipeline
- Le pipeline est déclenché à chaque nouveau commit reçu par le serveur et est exécuté dans un runner
- `.gitlab-ci.yml` décrit aux runners ce qu'ils doivent faire



Une recette pour les runners 🔍 : .gitlab-ci.yml

```
stages:
  - ✅ test
  - 📦 build
  - Deploy


python_lint:
  stage: ✅ test
  ...

docker_build:
  stage: 📦 build
  ...

mkdocs:
  stage: 📦 build
  ...
```

Stages

Jobs



Une recette pour les runners 🔍 : `.gitlab-ci.yml`

Jobs options/mots clés:

- [image](#): image Docker utilisée pour exécuter le job
- [script](#): commande shell à appliquer
- [variables](#): variables d'environnement
- [artifacts](#): artifacts produits par le job à garder

La documentation GitLab présentant toutes les options : docs.gitlab.com/ee/ci/yaml/

Quand un job se déclenche, le repo git est cloné dans le répertoire du runner, et le script démarre.



Une recette pour les runners 🔍 : .gitlab-ci.yml

```
stages:
```

- ✅ test
- 📦 build
- Deploy

```
python_lint:
```

```
  stage: ✅ test
```

```
  image: python:3.9-alpine
```

←● Image

```
  script:
```

- pylint *.py

←● Script



Une recette pour les runners 🔍 : .gitlab-ci.yml

```
mkdocs:  
  image: squidfunk/mkdocs-material:6.1.7  
  stage: 📦 build  
  script:  
    - [...]  
    - mkdocs build -d "$MKDOCS_OUTPUT_PATH"
```

variables:

```
  MKDOCS_OUTPUT_PATH: 'website_build/'
```



Variable(s)

artifacts:

```
  when: always  
  expose_as: "Mkdocs build"  
  paths:  
    - "website_build/"
```



Artifact(s)



Une recette pour les runners 🔍 : .gitlab-ci.yml

✅ test



python_lint



📦 build



docker_build



mkdocs



Deploy



helm_deploy:p...



pages



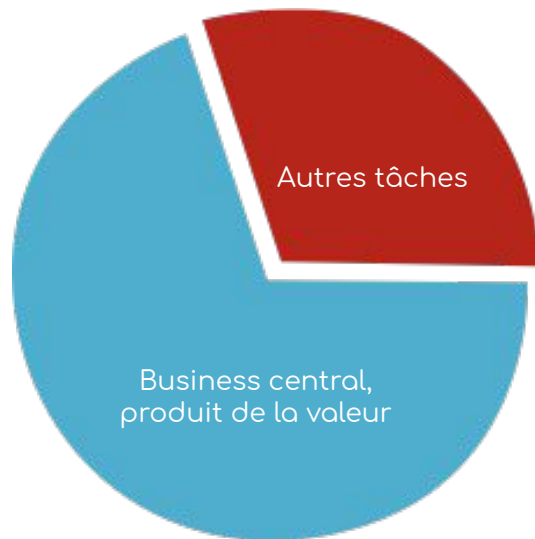


03

R2Devops

La problématique marché

Notre constat



-  Déployer, rédiger des tests, des spécifications, vérifier la qualité et les performances, exécuter les tests...
-  Coder



Notre constat

Problématiques nos clients rencontrent dans leurs projets IT



Ne savent pas par où commencer



Besoin d'un large éventail de compétences



Demande du temps pour la création



Peu de maintenance réalisée



Niveau de confiance faible



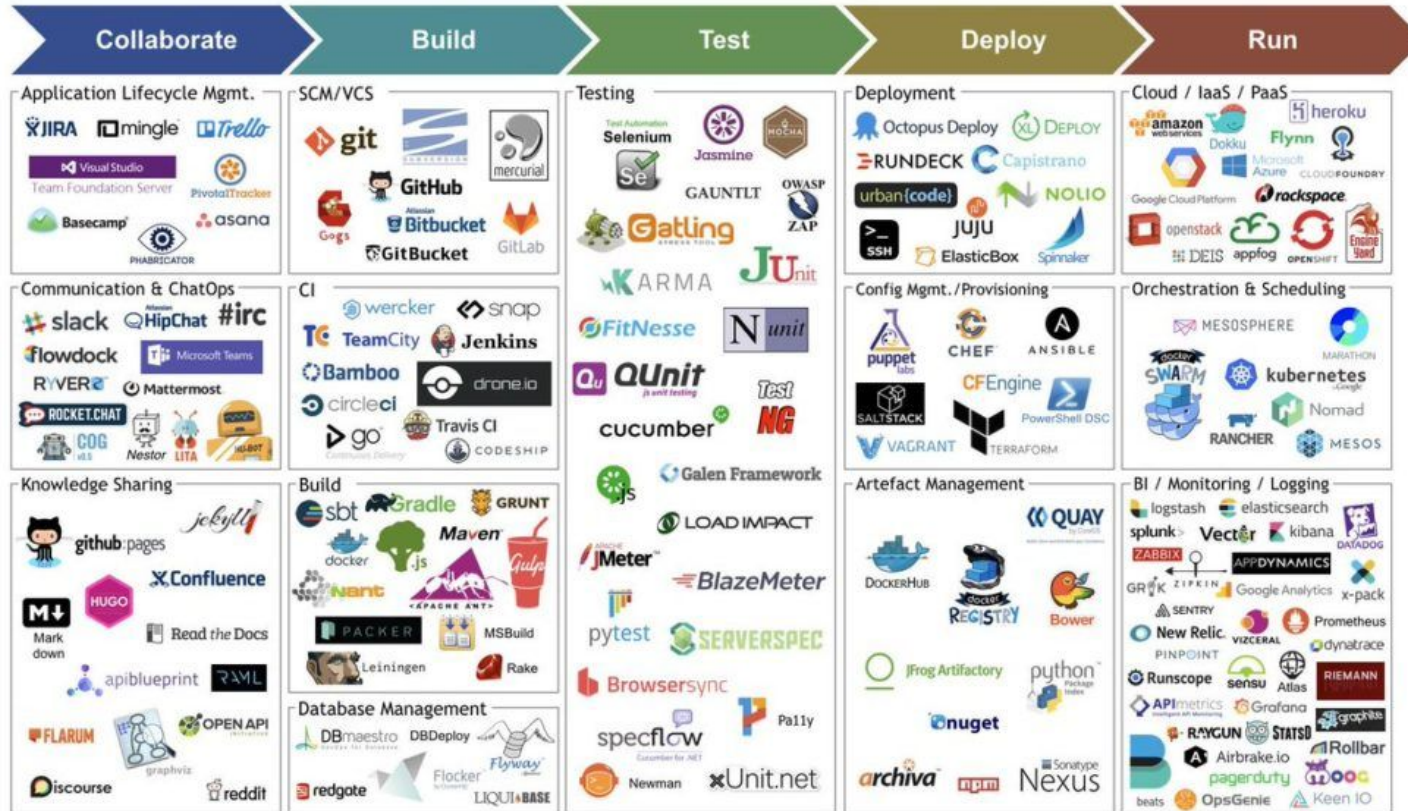
Documentation limitée



Duplication



Tous les outils CI/CD

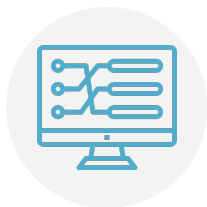


Notre réponse

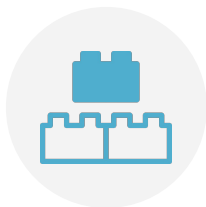
R2Devops



Les développeurs devraient pouvoir se concentrer sur leur activité métier :
amener de la valeur à leurs utilisateurs finaux



Créer des pipelines en
3 clics



Jobs prêts à l'emploi,
fonctionnement en
brique de Lego



Plateforme
collaborative



R2Devops

Métriques entre décembre 2020 et août 2021



👉 gitlab.com/r2devops/hub



59 jobs



62 stars



15 contributeurs



17 forks



258 MR



> 100 000 jobs
récupérés

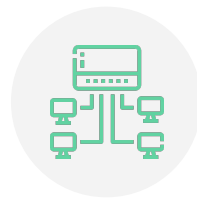


R2Devops

Les avantages



Intégration simplifiée pour
la CI/CD



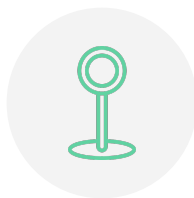
Plateforme centrale de jobs
CI/CD



Pour les débutants et
experts



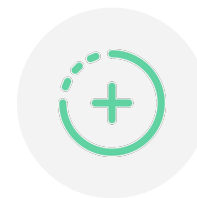
Contribution depuis
votre repo GitLab



Management des
pipelines de vos projets
à un seul endroit



Gestion des jobs
comme ressources
documentées et
versionnées



Accès à d'autres services
depuis R2Devops



04

Pourquoi ?

Intégration et Déploiement Continues

Définition

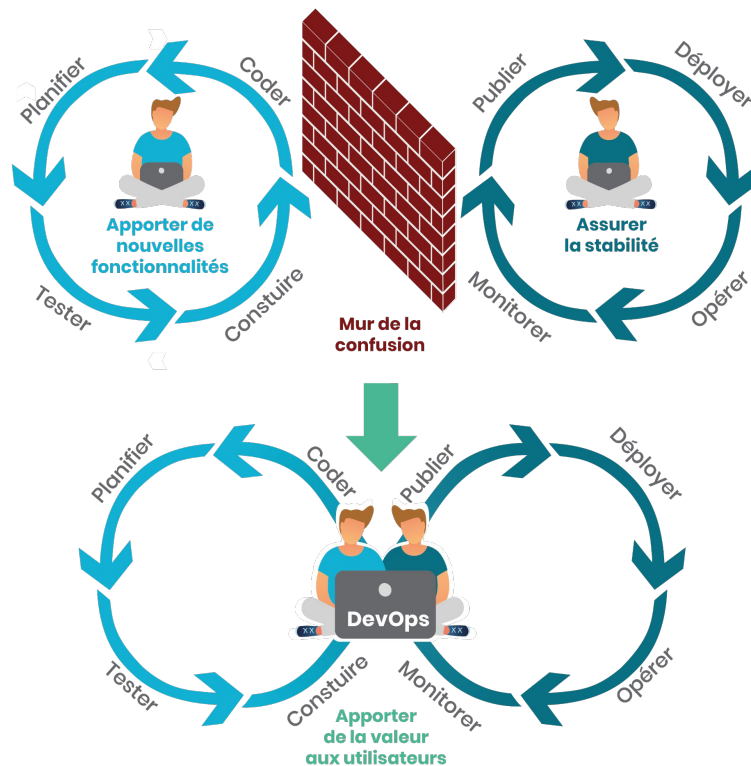
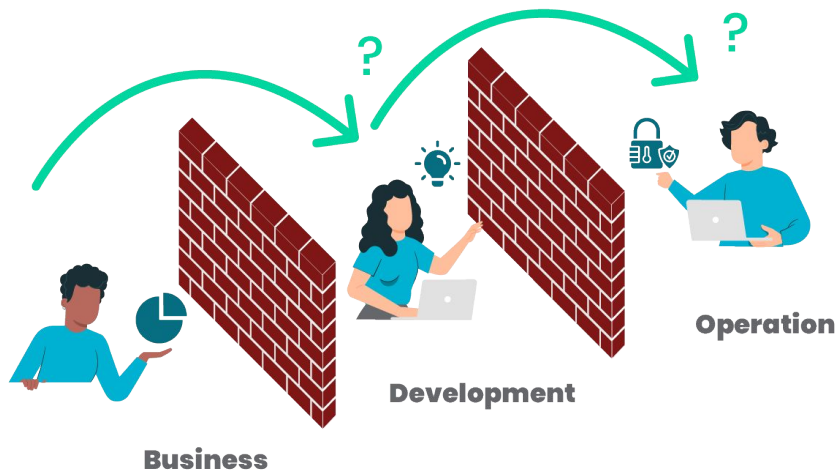
L'objectif :

augmenter la fréquence de distribution des applications grâce à l'automatisation et une surveillance continues tout au long du cycle de vie des applications, des phases d'intégration et de test jusqu'à la distribution et au déploiement.



Méthode Agile et DevOps

Méthode Agile et DevOps



Conclusion

L'importance du CI/CD

Qu'est-ce que c'est ?

Le CI/CD : introduire un haut degré d'automatisation et de surveillance continues dans le processus de développement des applications.

À quoi ça sert ?

Intégrer rapidement tout nouveau segment de code : aujourd'hui nous sommes de plus en plus nombreux à collaborer sur des projets que nous devons livrer de plus en plus vite !

Intégrer fréquemment tout nouveau segment de code : éviter les conflits en mergeant plus souvent des plus petits morceaux de codes (horreur : faire merger 10 dev, le même jour (jour du déploiement) avec chacun des centaines de lignes en commun...).

Surveillance en continu du code, garantir une qualité de service : critère de qualité (performance, taux de couverture, temps de réponse...).

Disposer d'un code toujours prêt à être déployé en production : déploiement depuis n'importe quel endroit, n'importe quand.



Merci



Vous avez des questions ?

contact@go2scale.com

go2scale.com



Go2Scale