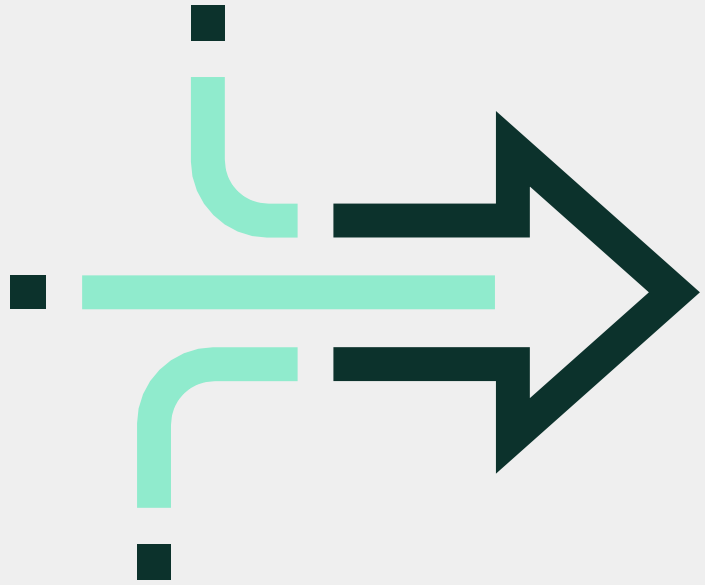
18 JUILLET 2023

SUSE

IT >

Le CI/CD



En 30 minutes

Audience

✅ Toute personne intéressée par l’univers des Technologies de l’Information (IT).

✅ Aucune connaissance préalable n’est requise.

Intervenant #1 - Bertrand

- 20 ans d'expérience professionnelle dans l'IT
- Développeur -> Tech Lead -> Manager -> Architecte
- 30 entreprises de taille et domaine fonctionnel variés
- Consultant (audit, accompagnement) et speaker DevOps
- Salarié, travailleur indépendant et bénévole

linkedin.com/in/berthomas

Agenda

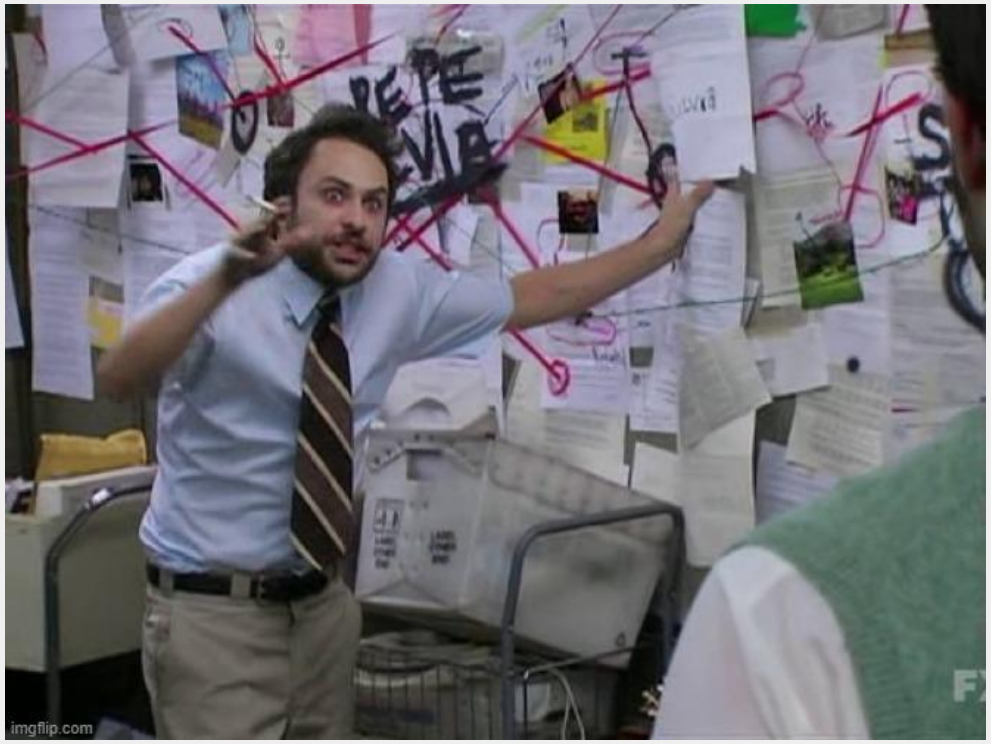
1. **La théorie**
   Définitions et état de l'art

2. **En pratique**
   Guide et exemples

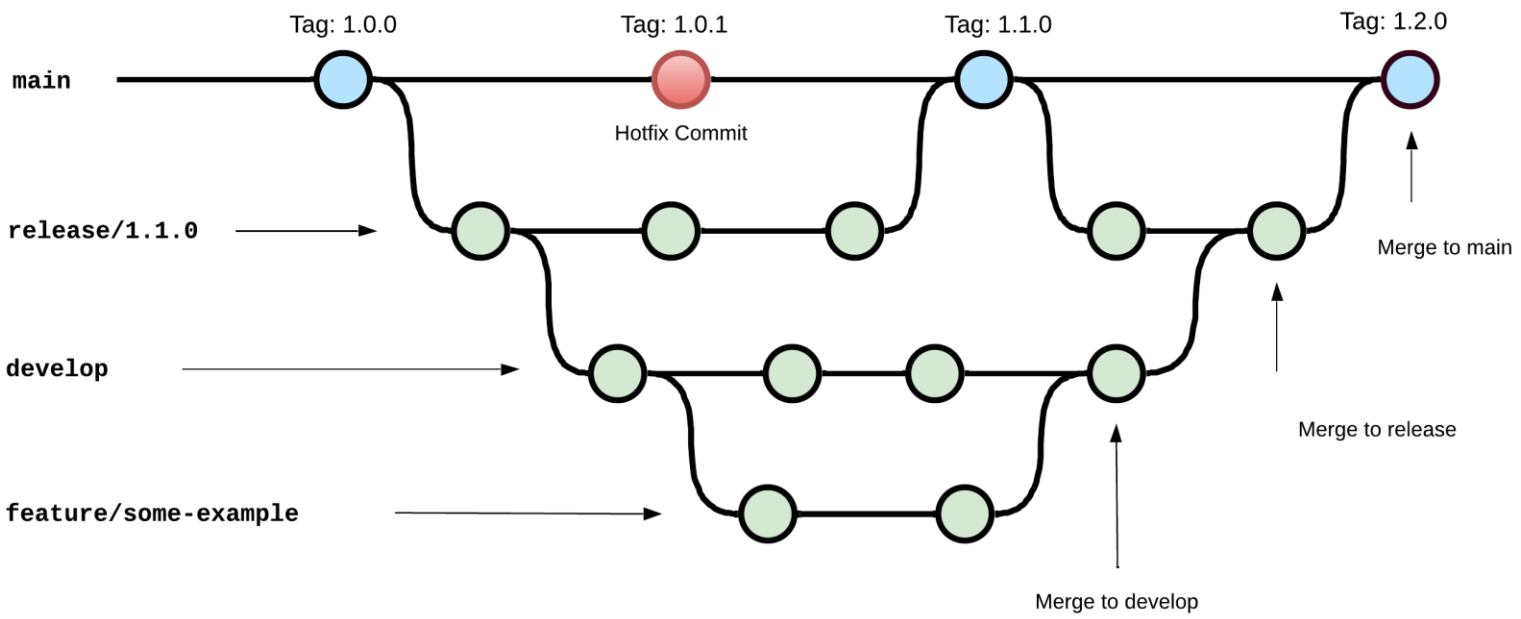
3. **Questions / Réponses**
   A votre tour de participer !

La théorie

Définitions et état de l’art



Challenges de la gestion d'une base de code



Conditions de succès :

- Non-régression
- Maintenabilité
- Conformité
- Sécurité
- Performance
- Suivi de la dette technique
- "Time to market"

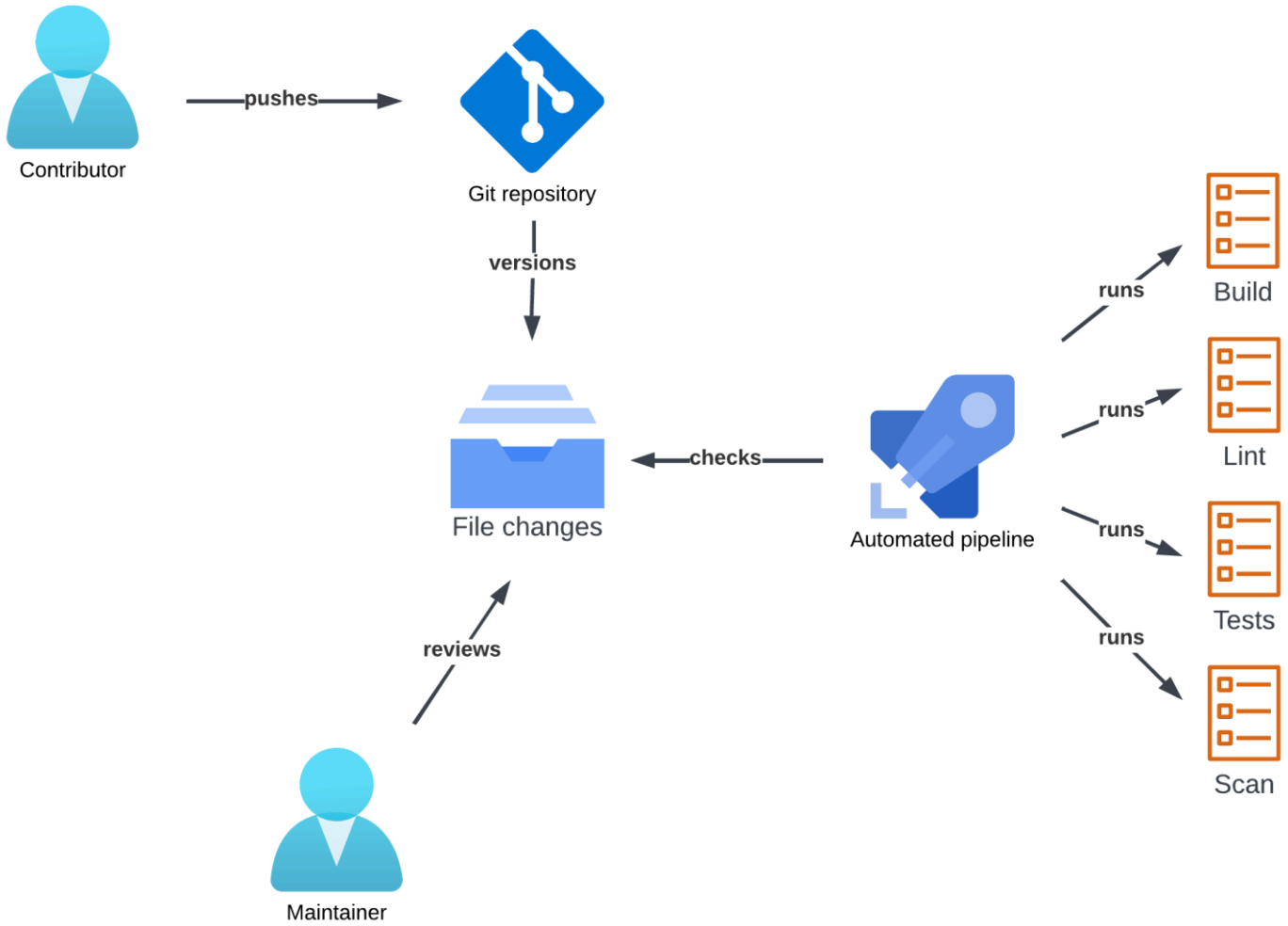
Intégration continue (CI)

« L'Intégration Continue consiste à intégrer les changements apportés au code informatique d'un projet logiciel de façon continuelle, afin de détecter et de corriger immédiatement les éventuelles erreurs. » (lebigdata.fr)

« Les principaux objectifs de l'intégration continue sont de trouver et de corriger plus rapidement les bogues, d'améliorer la qualité des logiciels et de réduire le temps nécessaire pour valider et publier de nouvelles mises à jour de logiciels. » (aws.amazon.com/fr)

Intégration continue = vérification au plus tôt de la qualité du code

Fonctionnement de l'Intégration Continue



Points d'attention sur l'Intégration Continue

- Bonne utilisation de l'outil de gestion de version (git)
- Rapidité d'exécution (durée < 10min)
- Concerne tous les dépôts de code : applicatif, infrastructure, documentation
- Principes DRY, KISS, DRTW pour les pipelines
  - Librairies de tâches et templates (open-source et innersource)
- L'affaire de tous !

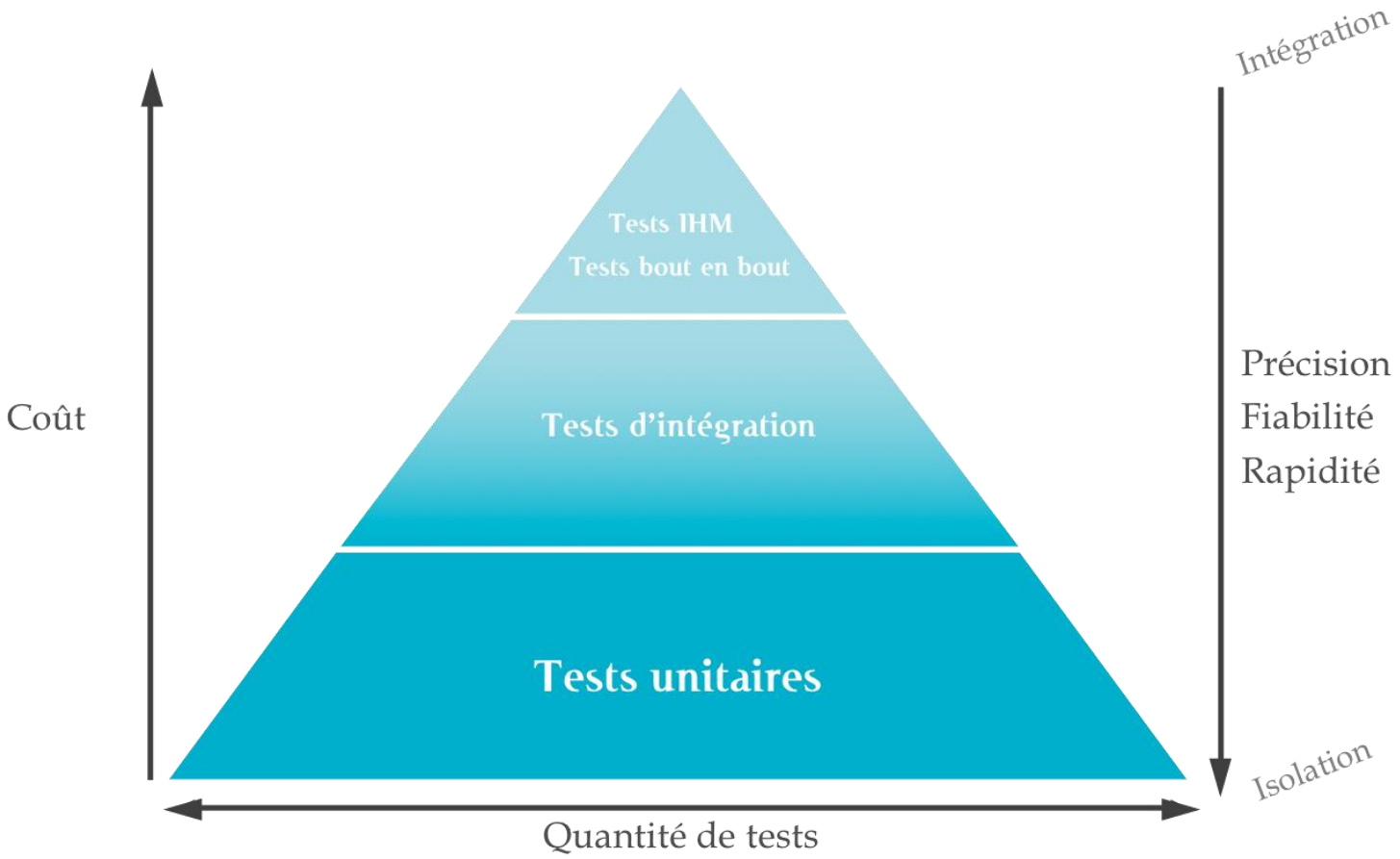
DRY = Don't Repeat Yourself

KISS = Keep It Simple Stupid

DRTW = Don't Reinvent the Wheel

CI/CD et tests automatiques

Exécution en environnement neutre et dynamique



blog.octo.com

Livraison continue (~CD)

« La livraison continue est une méthode de développement de logiciels dans le cadre de laquelle les modifications de code sont automatiquement préparées en vue de leur publication dans un environnement de production. » (aws.amazon.com/fr)

« Avec la livraison continue, une équipe peut décider de lancer le processus de publication n'importe quand. (...) Ensuite le code, sous la forme d'un artefact prêt à être publié, est déployé dans un environnement de test où quelques vérifications manuelles pourront être réalisées en plus avant la validation finale. (blog.acolad.com/fr)

Livraison continue = mise à disposition et validation du paquet prêt à l'emploi

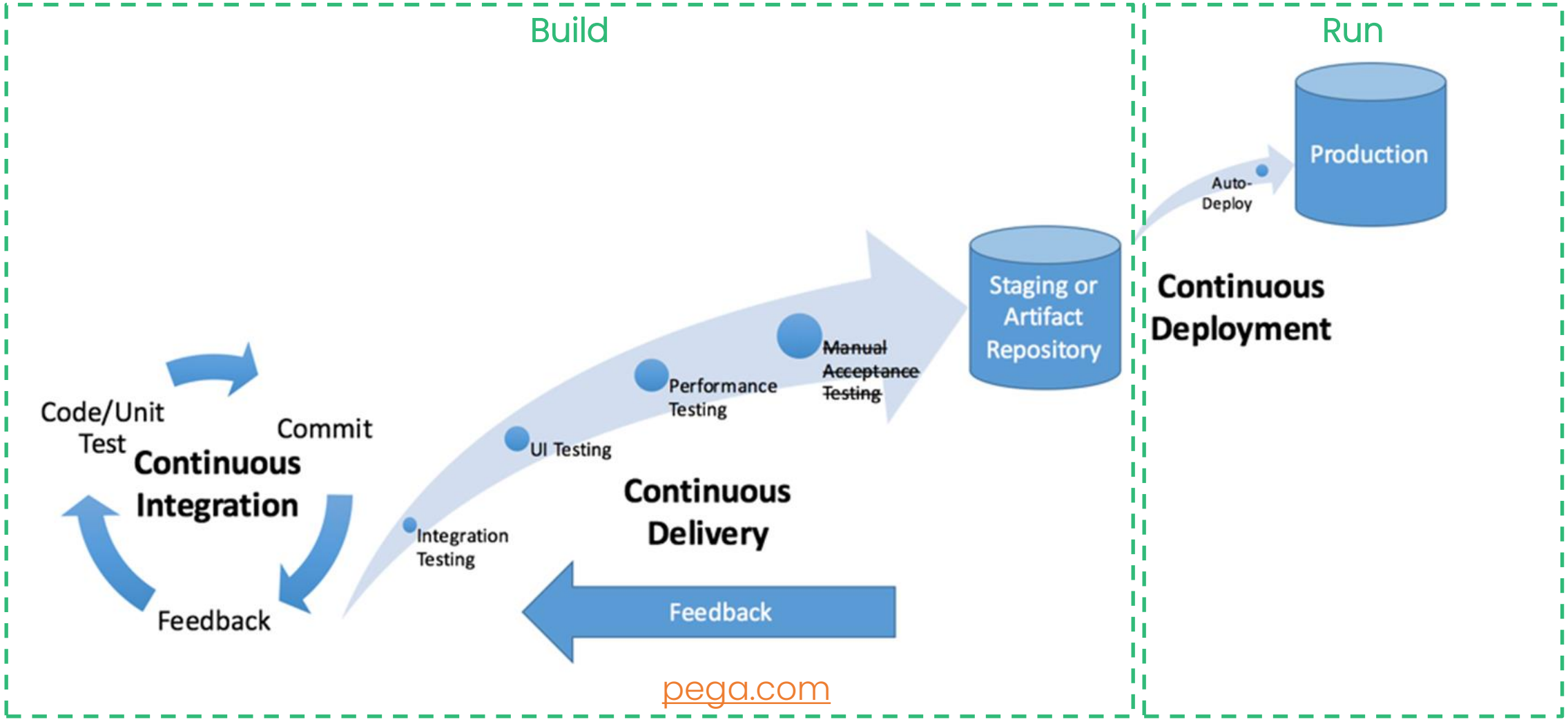
Déploiement continu (~CD)

« Le déploiement continu est une approche d'ingénierie logicielle dans laquelle les fonctionnalités logicielles sont livrées fréquemment par le biais de déploiements automatisés » (fr.wikipedia.org)

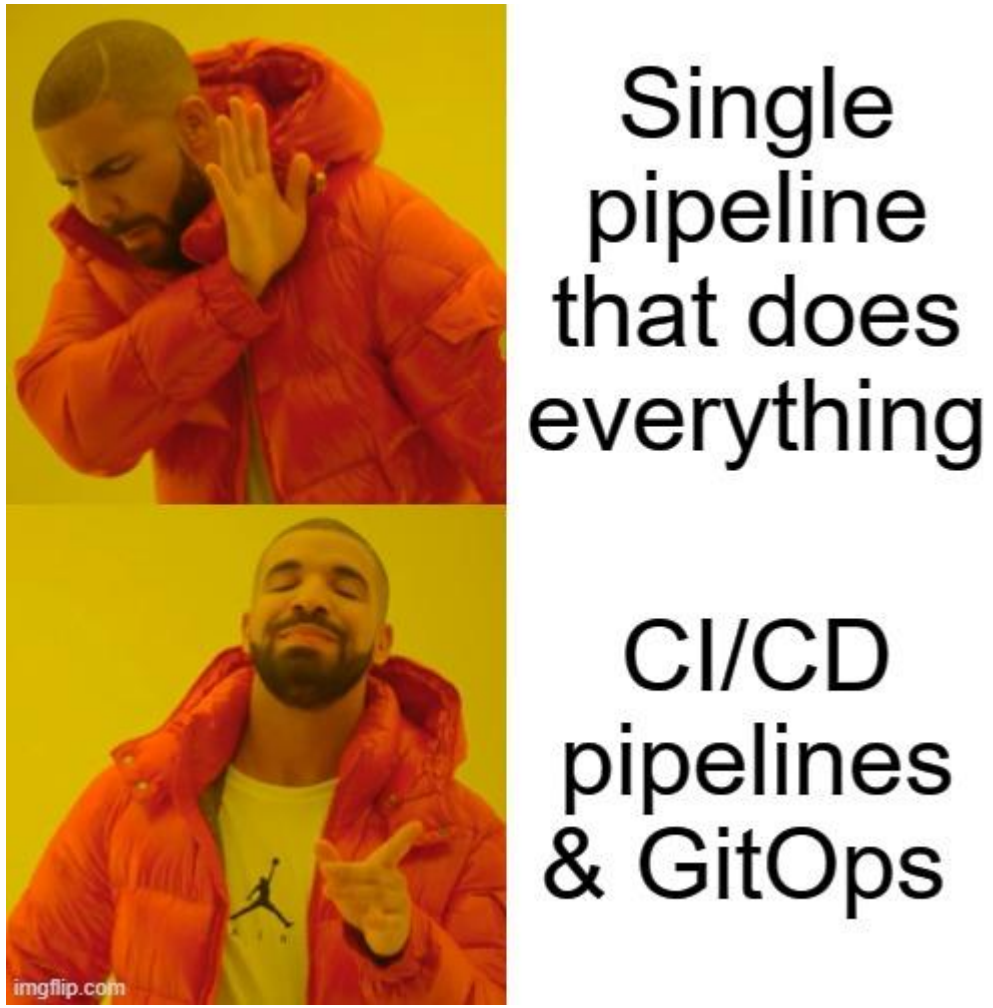
« Le déploiement continu est une stratégie de développement logiciel dans laquelle les modifications apportées au code d'une application sont publiées automatiquement dans l'environnement de production. » (ibm.com/fr)

Déploiment continu = faire d'une installation un non-évènement

Vue d'ensemble

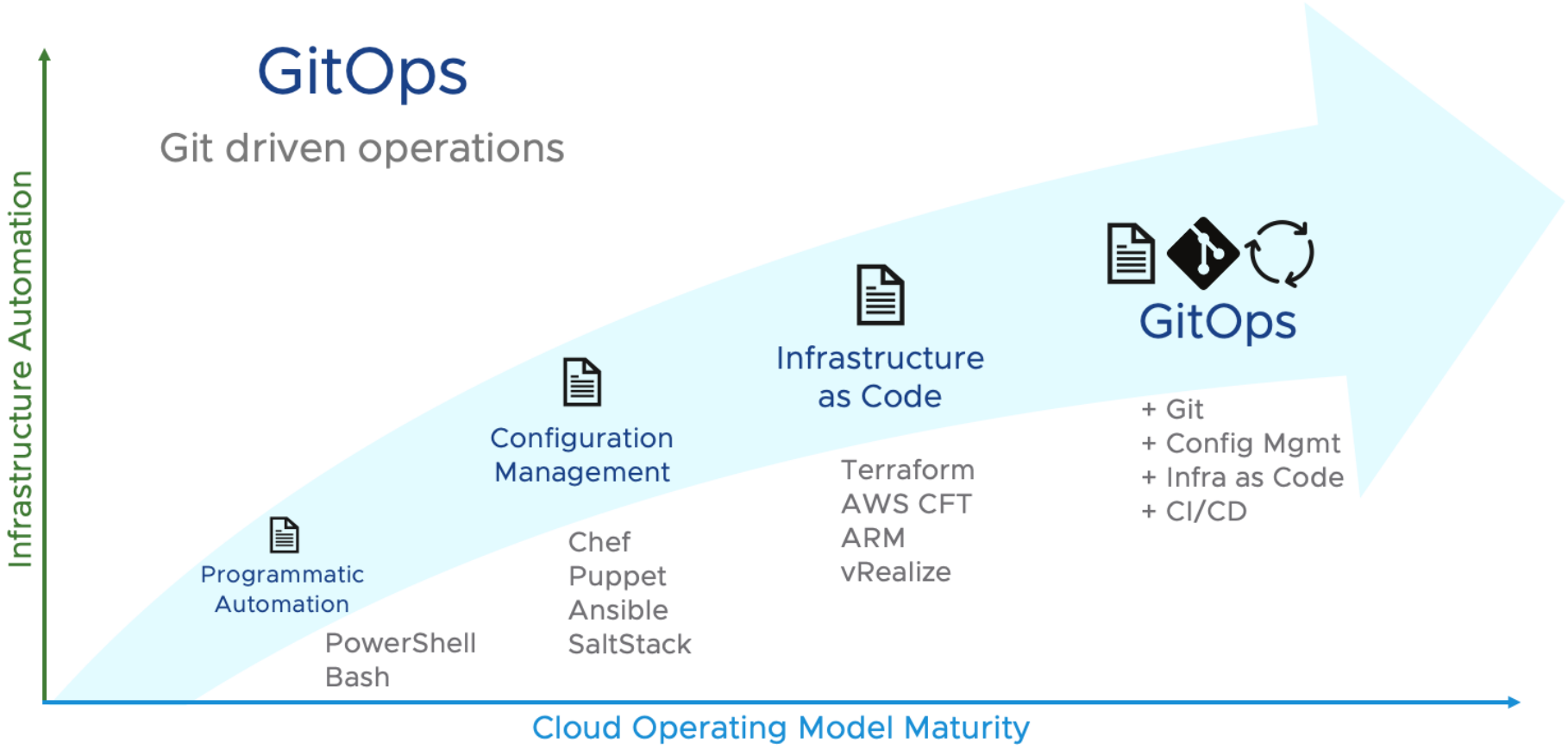


Mise en place du déploiement continu



GitOps

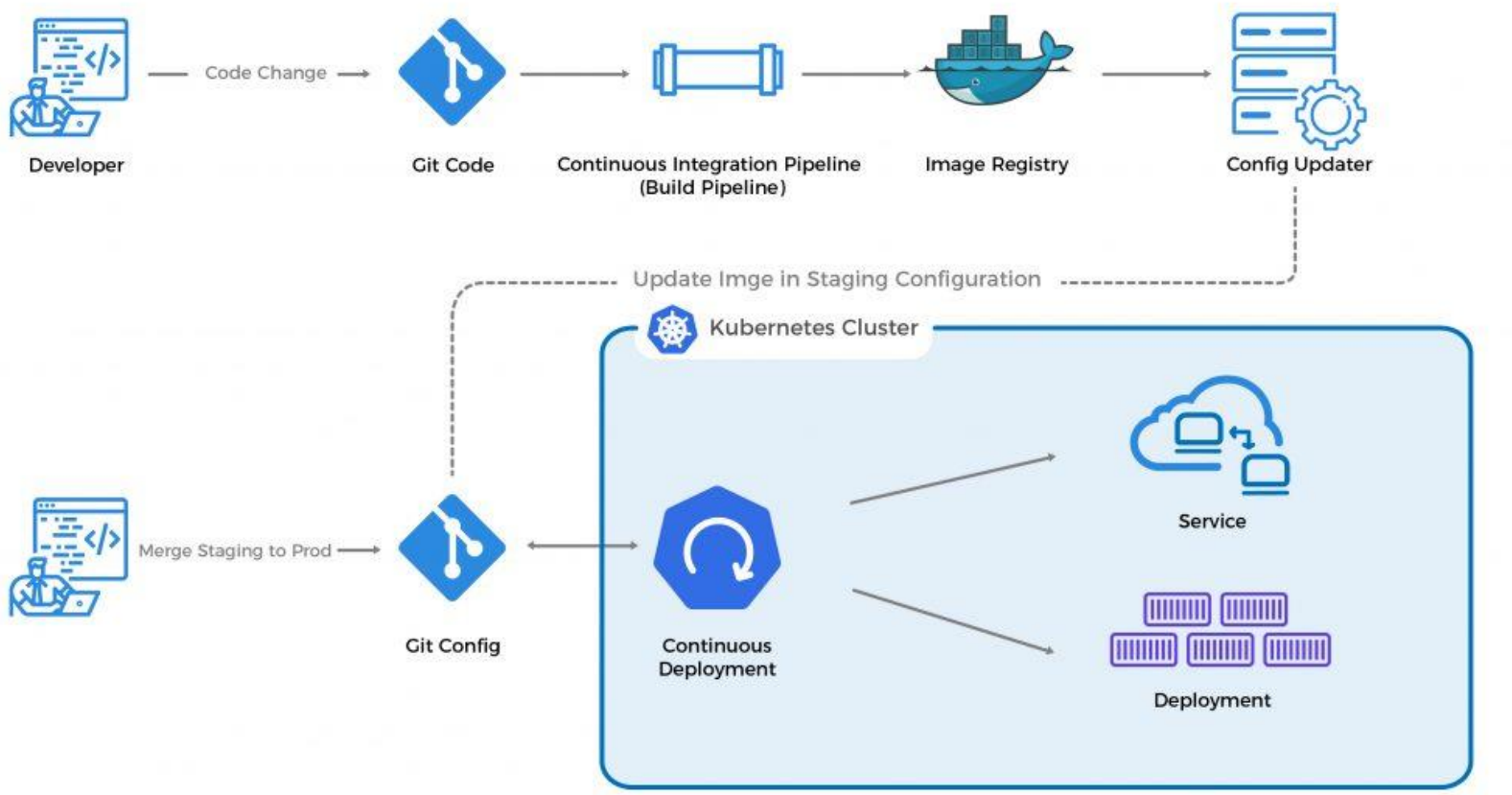
Git comme source de gestion d'infrastructure



blogs.vmware.com/cloud/2021/02/24/gitops-cloud-operating-model/

GitOps et Kubernetes

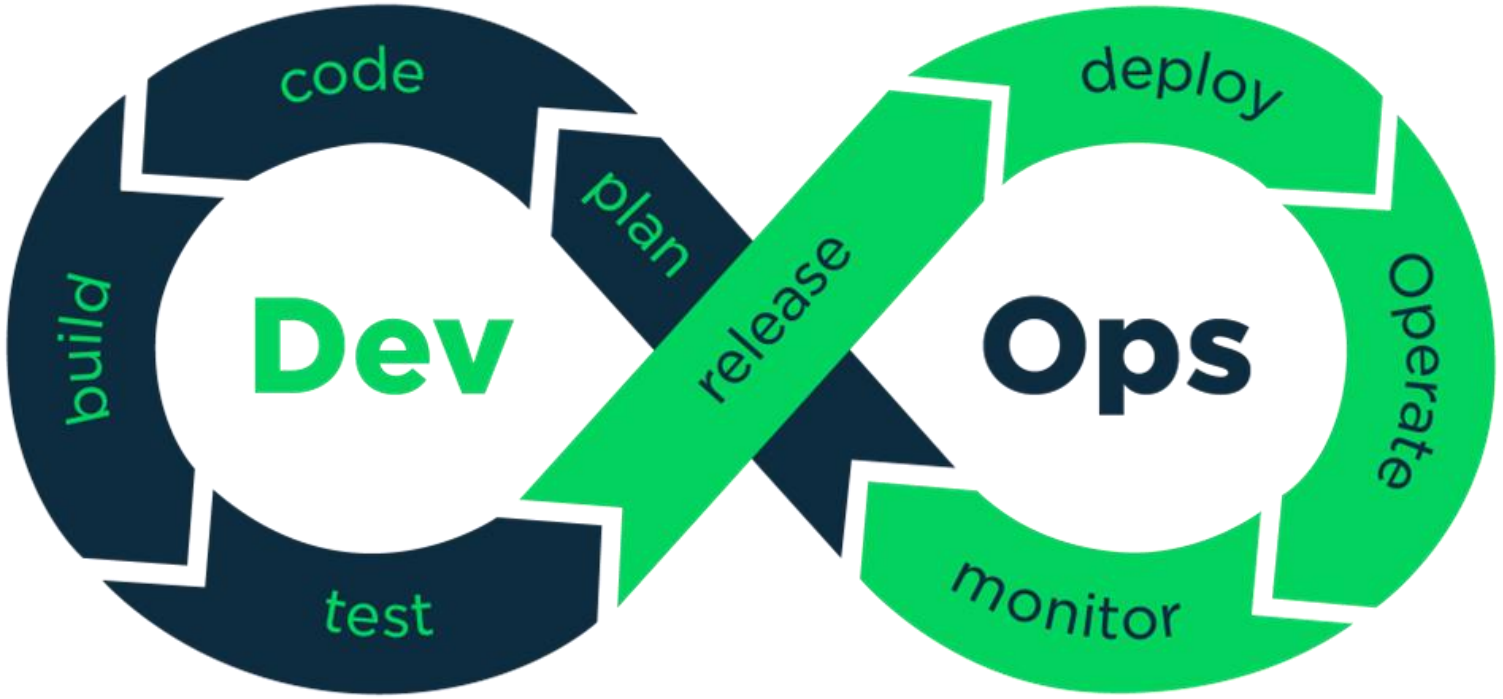
Simplicité et respect des responsabilités (Dev et Ops)



clickittech.com/devops/what-is-gitops

DevOps

Le cadre autour du CI/CD



En pratique

Guide et exemples

Comment progresser

- Evaluer son niveau de maturité
- Créer un plan d'action
- Itérer et partager les résultats
- Evaluer les outils/solutions et rédiger des ADR
- Impliquer en interne le plus de personnes possibles
- Solliciter l'aide externe d'experts
- Revenir au point 1 😁

ADR = Architecture Decision Record

Niveaux de maturité CI/CD

1. Le code est versionné (git)
2. Le pipeline CI valide chaque nouvelle version de code
3. Chaque nouvelle version stable génère automatiquement un nouvel artifact
4. Le déploiement (installation) d'un artifact se fait sans action manuelle
5. Les équipes sont à l'aise avec l'entièreté des outils

Les outils

ALM

- Azure DevOps
- Bamboo, Bitbucket, Confluence, Jira
- GitHub
- GitLab

Code scan

- Sonar

Security scan

- NeuVector
- Trivy

Pipelines

- Circle CI
- Concourse
- Drone
- Jenkins
- Keptn
- Tekton
- Travis CI

Registries

- Artifactory
- Harbor
- Nexus
- Quay

GitOps

- Argo CD
- Fleet
- Flux

ALM = Application Lifecycle Management

Les petits plus

- Badges dans le README
- Editeur et marketplace pour les pipelines (R2Devops)
- GUI pour git (GitKraken)
- Linter universel (MegaLinter)
- Noms explicites (éviter « build » pour un pipeline)
- Plateforme de collaboration (Promyze)
- YAML et Markdown sont vos amis

Exemple 1

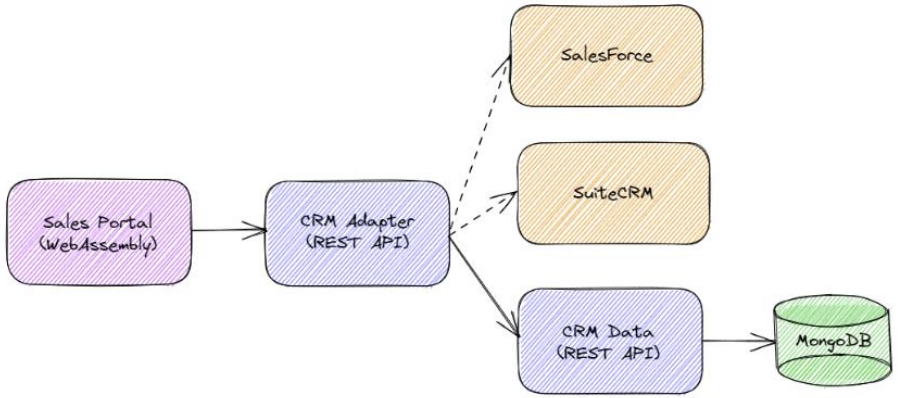
Une base applicative qui va être déployée dans des containers

Sales Portal

CI passing | PKG passing | quality gate failed | Docker v1.0.5332210085

Open source solution for a Sales Portal. It has been created for educational purposes but feel free to use it and contribute!

Design



Installation

To install it on a Kubernetes cluster, use the official Helm chart.

Development

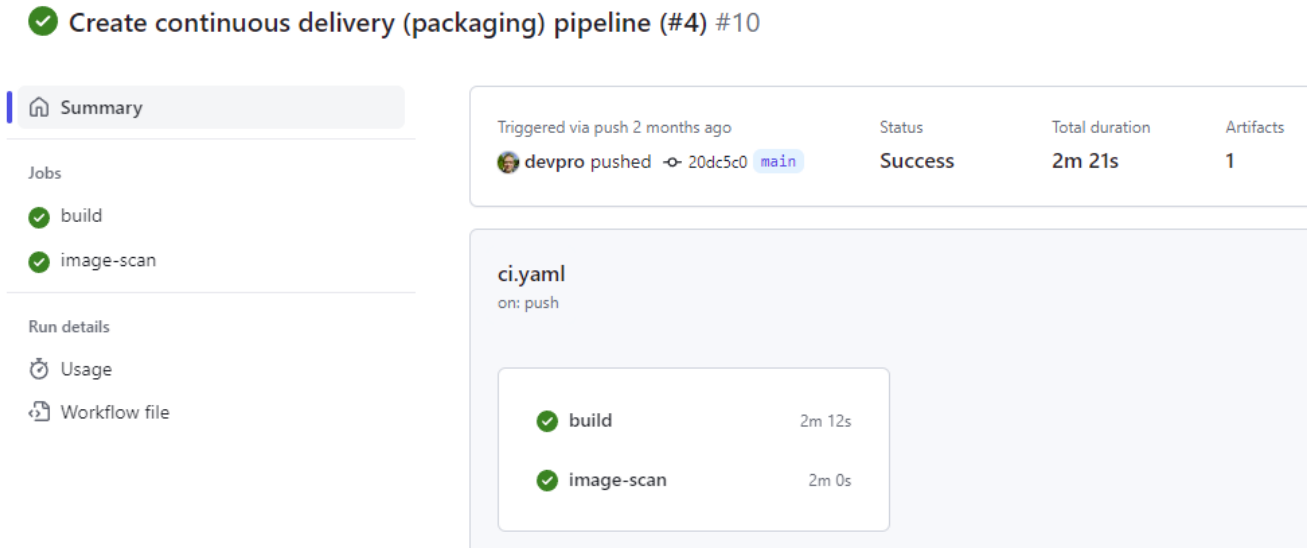
Look at the contributing guide.

```yaml
# ref: https://github.com/devpro/helm-charts/tree/main/charts/devpro-salesportal
defaultNamespace: sales-portal
helm:
  repo: https://devpro.github.io/helm-charts
  chart: devpro-salesportal
  version: 0.1.2
  releaseName: sales-portal
  values:
    front:
      tls:
        secretName: sales-portal-tls
    adapter:
      tls:
        secretName: crm-adapter-tls
    data:
      tls:
        secretName: crm-data-tls
    ingress:
      enabled: true
      className: nginx
```


github.com/devpro/sales-portal

Exemple 2

Réutilisation de code de pipeline




```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3
        with:
          fetch-depth: 0
      - name: Checkout workflow parts
        uses: actions/checkout@v3
        with:
          repository: devpro/github-workflow-parts
          ref: main
          path: workflow-parts
      - name: Start MongoDB
        uses: ./workflow-parts/mongodb/start
      - name: Build, lint & test
        uses: ./workflow-parts/dotnet/build-lint-test
        with:
          sonar_enabled: 'true'
          sonar_organization: ${{ vars.SONAR_ORG }}
          sonar_host_url: ${{ vars.SONAR_HOST_URL }}
          sonar_project_name: Terraform Backend MongoDB
          sonar_project_key: ${{ vars.SONAR_PROJECT_KEY }}
          sonar_token: ${{ secrets.SONAR_TOKEN }}
    env:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```


github.com/devpro/terraform-backend-mongodb

Exemple 3

Qualité de code d'infrastructure (Terraform)

Terraform projects

CI passing

Infrastructure automation projects leveraging Terraform.

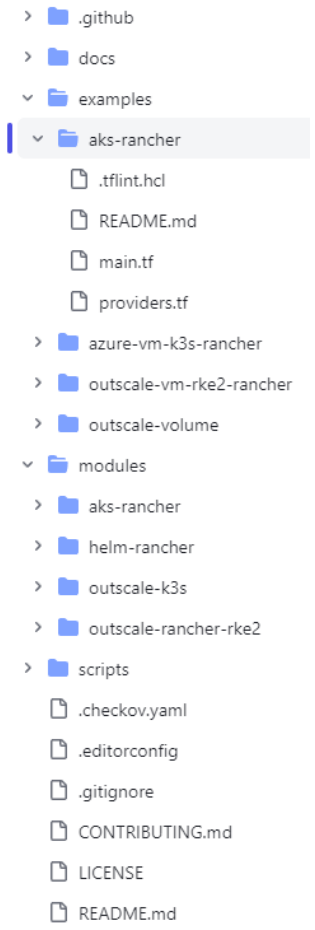
See how you can contribute. Any help will be appreciated!

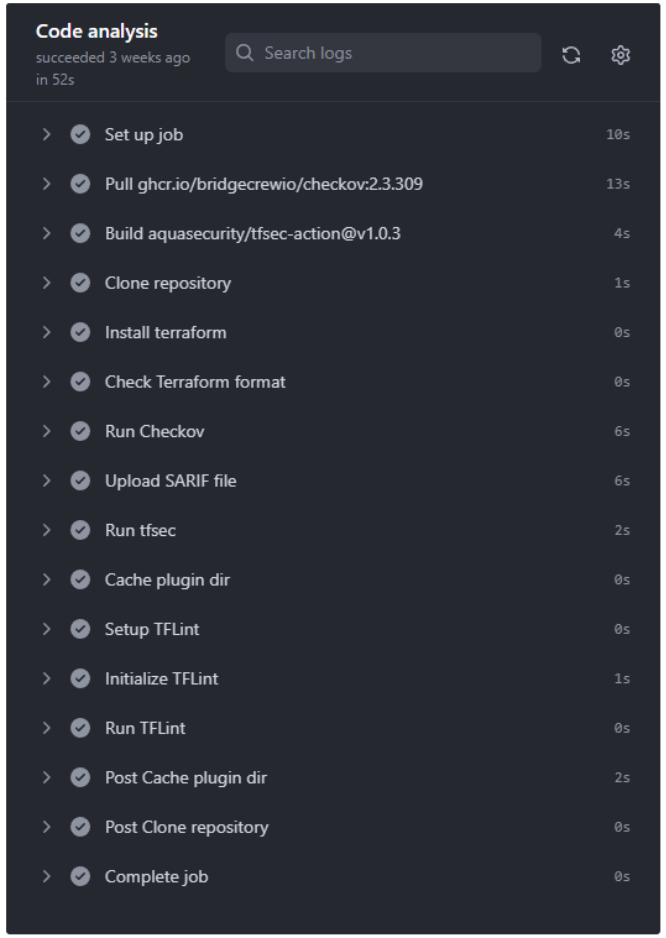
Modules

- aks-rancher
- helm-rancher
- outscale-k3s
- outscale-rancher-rke2

Examples

Name	Reason
aks-rancher	Rancher quickstart on AKS
outscale-rancher-quickstart	Rancher quickstart on Outscale Cloud (management + downstream)
outscale-volume	Volume provisioning on Outscale Cloud

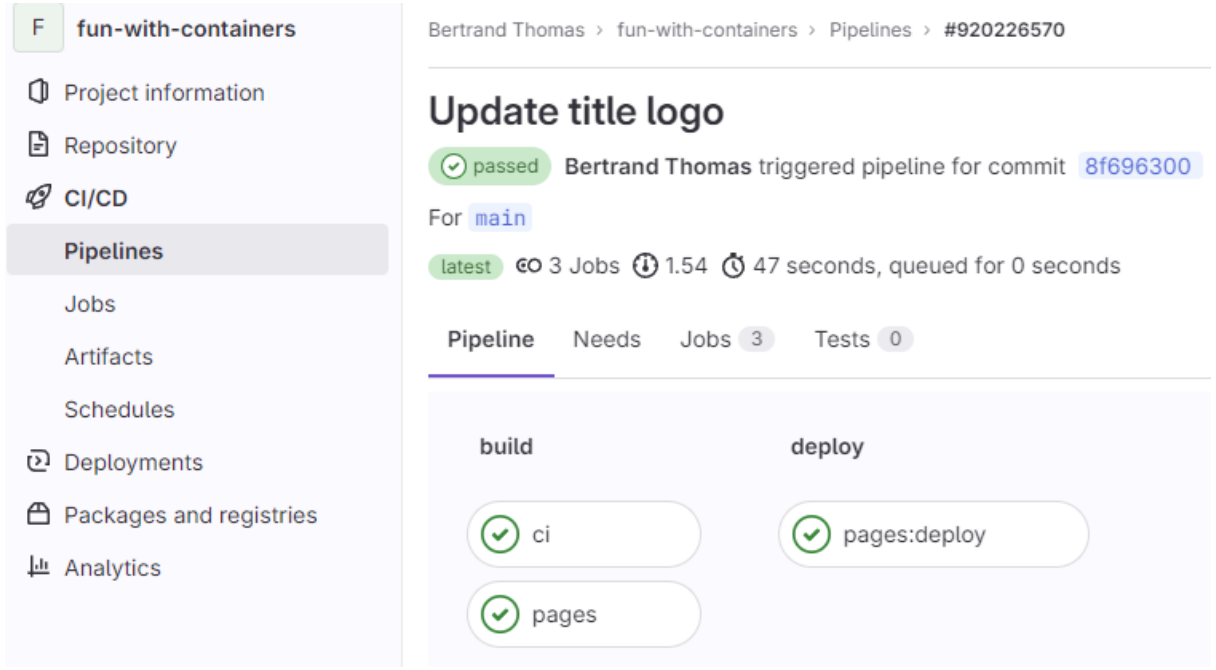




github.com/devpro/terraform-projects

Exemple 4

Documentation-as-code



Fun with containers

Presentations

- GitOps 101 - April 2022
- Observability 101 - April 2022

github.com/devpro/fun-with-containers

Exemple 5

Dépôt de charts Helm

Helm Charts

CI passing PKG passing

Helm charts to ease the deployment of containers on Kubernetes clusters and get information on widely used components.

Quickstart

- Visit devpro.github.io/helm-charts

Devpro Helm Charts

Usage

```
helm repo add devpro https://devpro.github.io/helm-charts
```

This repository contains Helm charts to build clusters with all components running in containers.

Charts

argo-cd (0.1.0@2.5.2)	azure-storage (0.1.0@1.0.0)	cert-manager (0.1.4@1.10.0)
Helm chart for managing ArgoCD	Helm chart for managing Azure Storage	Helm chart for managing cert-manager

github.com/devpro/helm-charts

Q&A

Questions ou remarques ?

Pipelines ?

Modernisation applicative ?

Secrets ?

Artifacts ?

Pipeline-as-code ?

Tests ?

GitOps ?

Kubernetes ?

Merci