

CI/CD : Écosystème et Infrastructure

Emmanuel BRUNO
Université de Toulon, LIS UMR CNRS 7020
emmanuel.bruno@univ-tln.fr

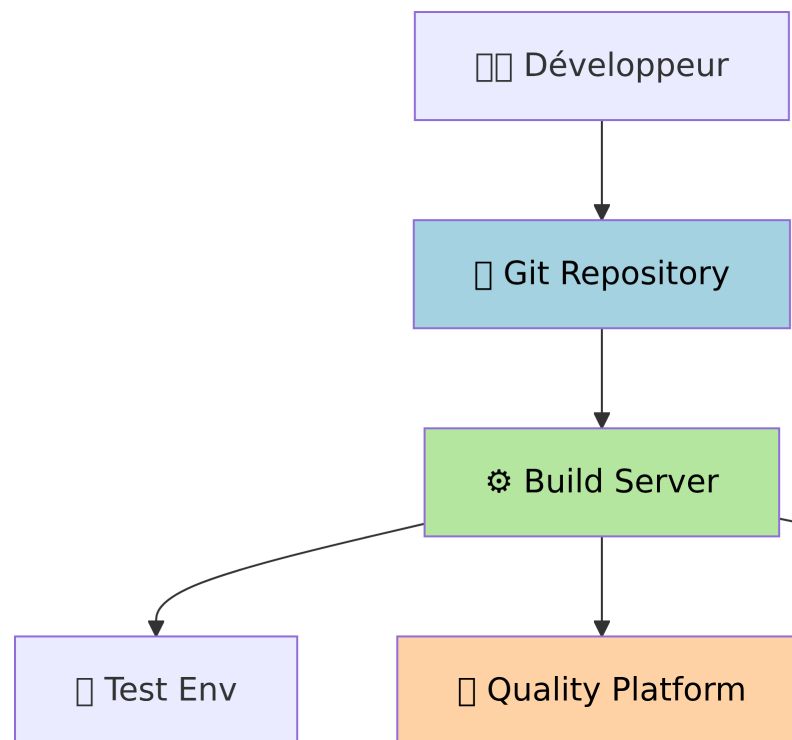
2026-10-14

Table des matières

1 Vue d'ensemble : Les composants d'un système CI/CD	1
2 L'architecture moderne CI/CD	2
3 Les composants essentiels	2
3.1 Serveur de build : Orchestration des pipelines	2
3.2 Environnements de test : Isolation et reproductibilité	3
3.3 Plateforme de qualité : Gardien du code	4
3.4 Registre d'artefacts : Stockage des binaires	4
3.5 Registre d'images : Hub des conteneurs	5
4 Infrastructure de déploiement : Pourquoi Kubernetes ?	6
5 Orchestrateurs : Kubernetes, Docker Swarm, Docker Compose	7
5.1 Solutions intégrées vs Best-of-breed ..	7
5.2 Tout est conteneurisé	9

1 Vue d'ensemble : Les composants d'un système CI/CD

Au-delà du simple pipeline : un écosystème complet



7 composants essentiels

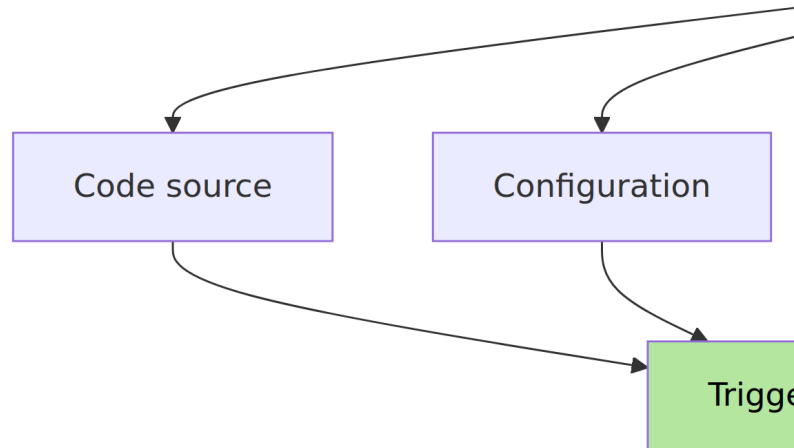
1. **Repository Git** - Source de vérité
2. **Build Server** - Orchestrateur
3. **Test Environments** - Validation
4. **Quality Platform** - Gardien
5. **Artifact Registry** - Binaires
6. **Image Registry** - Conteneurs
7. **Infra Deployment** - Cible

Objectif : Comprendre le rôle et les outils de chaque composant

2 L'architecture moderne CI/CD

La vision complète : Du code à la production

Rôle central



Fonctionnalités essentielles

☑ Gestion des branches & protection ☑ Pull/Merge Requests + code review ☑ Webhooks (déclenchement pipelines) ☑ RBAC & permissions granulaires

🔧 Plateformes Git

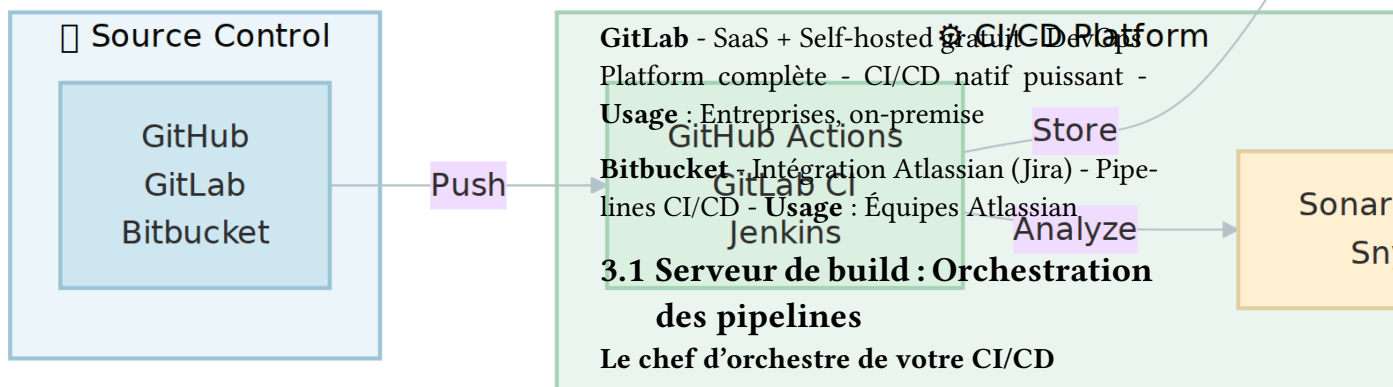
GitHub - SaaS + Enterprise - Actions CI/CD intégré - Communauté énorme - **Usage** : Open source, startups

GitLab - SaaS + Self-hosted **GitLab CI/CD Platform** complète - CI/CD natif puissant - **Usage** : Entreprises, on-premise

Bitbucket - Intégration Atlassian (Jira) - Pipelines CI/CD - **Usage** : Équipes Atlassian

3.1 Serveur de build : Orchestration des pipelines

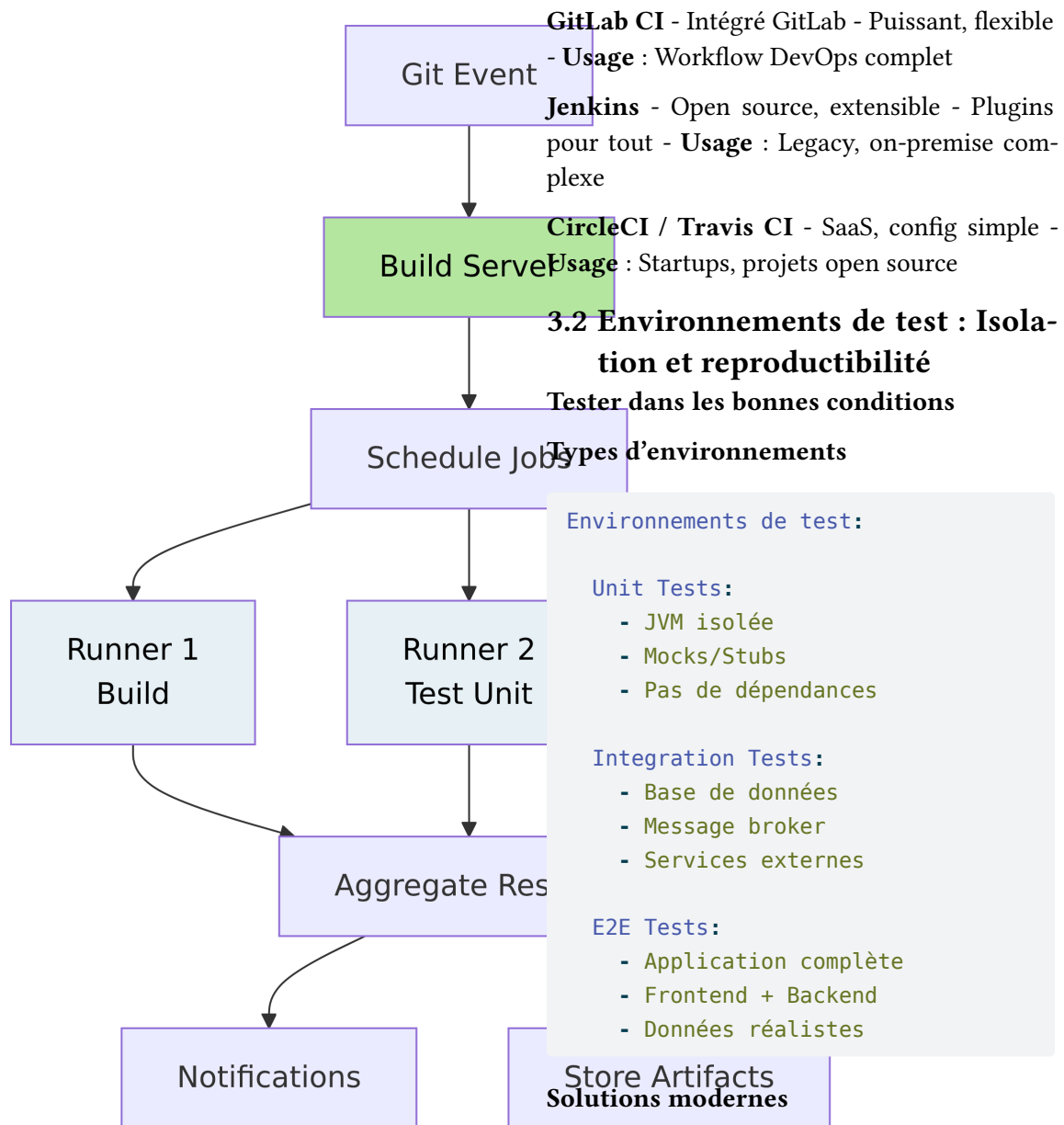
Le chef d'orchestre de votre CI/CD



3 Les composants essentiels

Repository Git : La source de vérité

Single Source of Truth : Tout part du code



Responsabilités

☒ Orchestration (scheduler, dépendances) ⚙
 Exécution (runners, parallélisation) ☒ Reporting (logs, métriques, notifications)

🔧 Plateformes CI/CD

GitHub Actions - Intégré GitHub - YAML simple, marketplace - **Usage** : Projets GitHub

☒ TestContainers

- DB dans Docker
- Éphémère et propre
- Même version que prod

⚙ Preview Environments

- K8s namespace par PR
- URL unique
- Auto-cleanup

🌐 Service Virtualization

- Mock APIs externes
- Contrôle des scénarios
- Pas de dépendances

3.3 Plateforme de qualité : Gardien du code

La qualité comme prérequis, pas comme option

☒ Code Quality (bugs, smells, complexité) 
 Test Coverage (lignes, branches)  Security (CVE, OWASP, secrets) ☒ Trends (dette technique)

Outils de qualité

SonarQube - Leader du marché - Self-hosted ou Cloud - Quality Gates - **Usage** : Toutes tailles d'équipe

Snyk - Focus sécurité (deps, code, containers) - Auto-fix PRs - **Usage** : Security-first teams

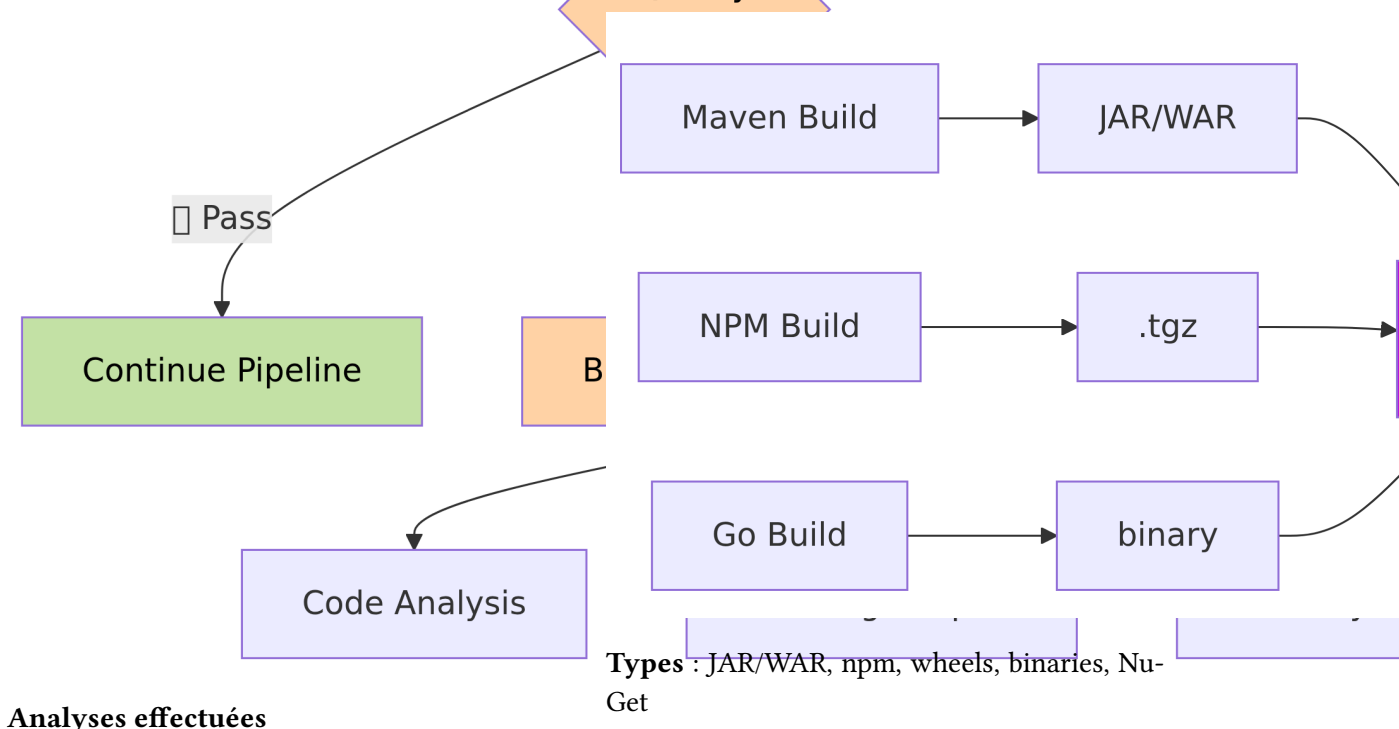
CodeClimate - SaaS simple - Maintenabilité focus - **Usage** : Startups, open source

Trivy - Open source - Scan containers + IaC - **Usage** : CI/CD security scans

3.4 Registre d'artefacts : Stockage des binaires

Stocker les résultats du build de manière persistante

Qu'est-ce qu'un artefact ?



Bénéfices : Traçabilité, reproductibilité, partage, cache

🔧 Registres d'artefacts

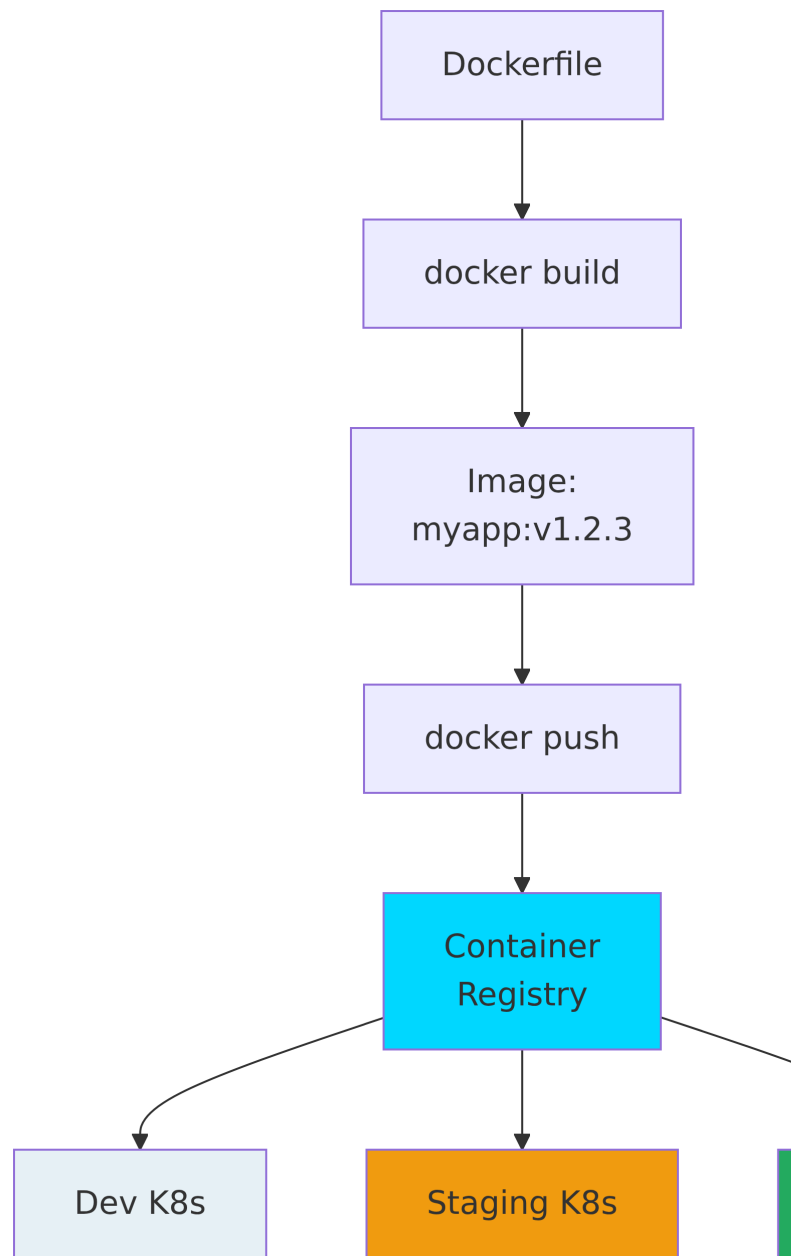
Sonatype Nexus - Repository manager mature - Multi-format (Maven, npm, PyPI...) - Open source disponible - **Usage** : Entreprises, on-premise

JFrog Artifactory - Universal artifact repository - Cloud-native - 30+ formats - **Usage** : Cloud, scaling

GitHub Packages - Intégré GitHub - Zéro configuration - **Usage** : Projets GitHub

3.5 Registre d'images : Hub des conteneurs

Stocker et distribuer les images Docker



Caractéristiques : Tags versionnés, scan vulnérabilités, image signing, layers partagés

🔧 Registres d'images

Docker Hub - Registre public par défaut - Bibliothèque énorme - **Usage** : Images publiques, dev

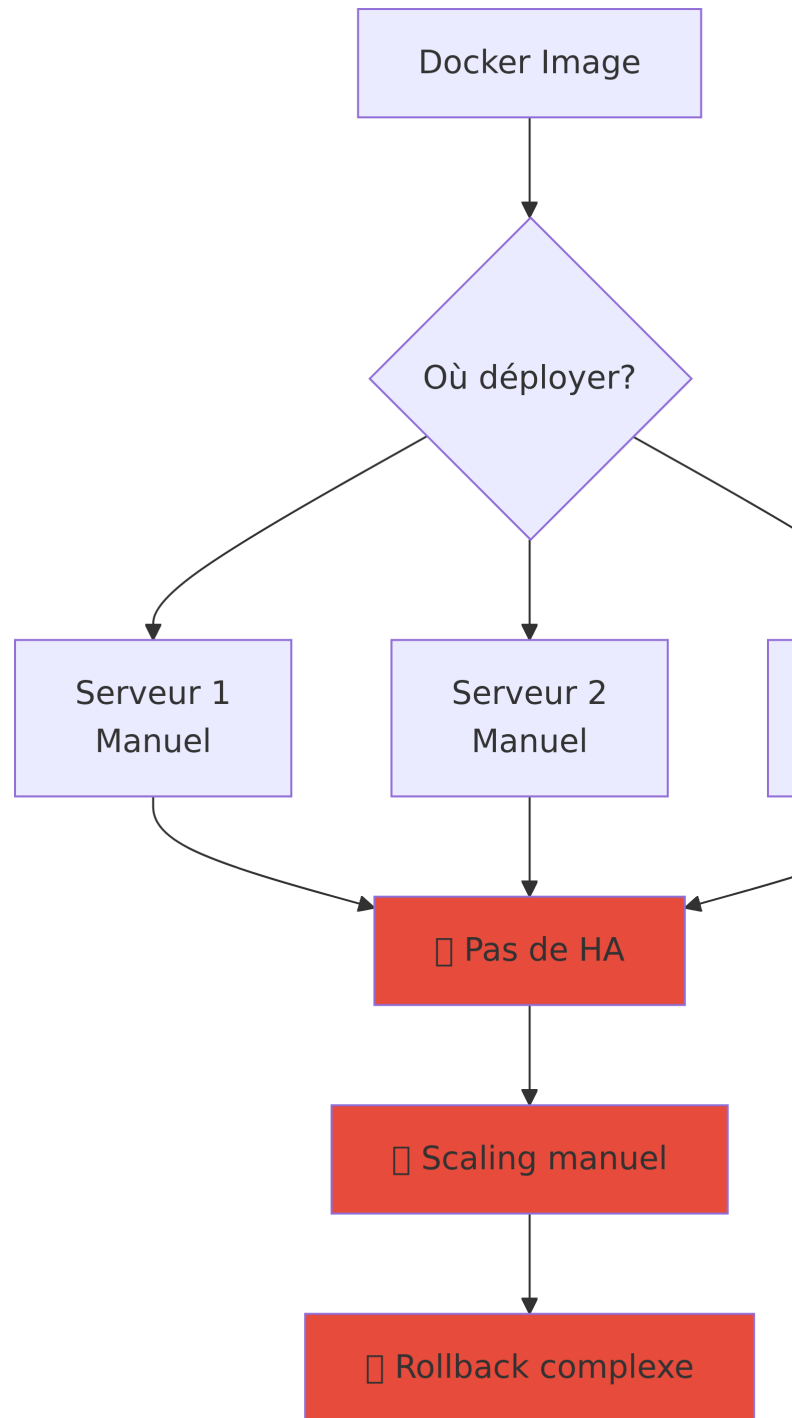
Harbor - Open source self-hosted - Vulnerability scan intégré - RBAC avancé - **Usage** : Entreprises, on-premise

AWS ECR / GCP Artifact Registry - Managed cloud - Intégration native IAM - Scan automatique - **Usage** : Apps cloud (AWS/GCP)

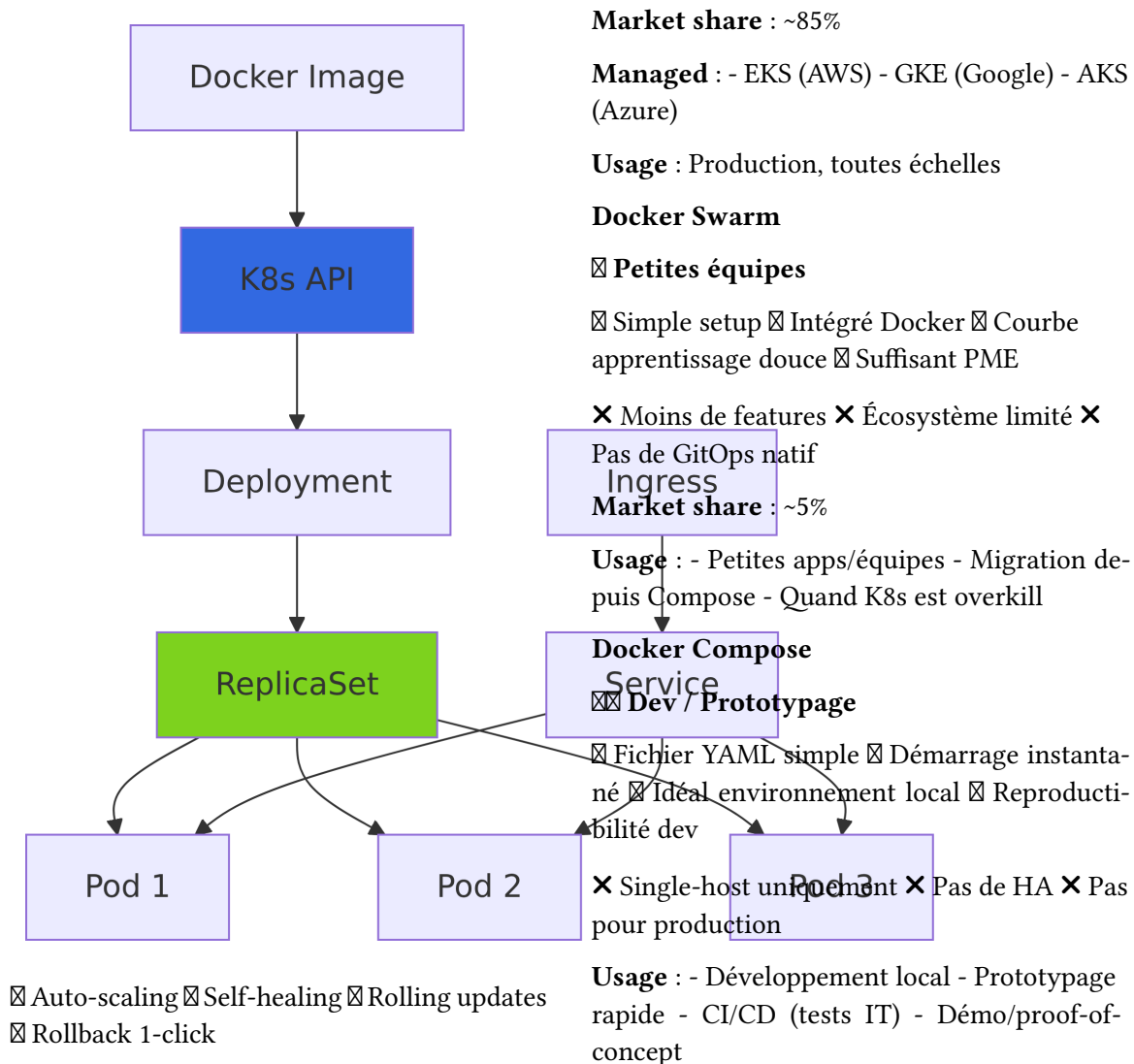
4 Infrastructure de déploiement : Pourquoi Kubernetes ?

L'orchestrateur de conteneurs devenu standard

Avant Kubernetes



Avec Kubernetes



5 Orchestrateurs : Kubernetes, Docker Swarm, Docker Compose

Gérer le déploiement et le scaling des conteneurs

Kubernetes

⚙️ Production

☑ Feature-rich ☑ Écosystème énorme ☑ Multi-cloud ☑ Auto-scaling ☑ Self-healing ☑ GitOps ready

✗ Complexité ✗ Courbe apprentissage

Market share : ~85%

Managed : - EKS (AWS) - GKE (Google) - AKS (Azure)

Usage : Production, toutes échelles

Docker Swarm

☑ Petites équipes

☑ Simple setup ☑ Intégré Docker ☑ Courbe apprentissage douce ☑ Suffisant PME

✗ Moins de features ✗ Écosystème limité ✗

Pas de GitOps natif

Market share : ~5%

Usage : - Petites apps/équipes - Migration depuis Compose - Quand K8s est overkill

Docker Compose

☑ Dev / Prototypage

☑ Fichier YAML simple ☑ Démarrage instantané ☑ Idéal environnement local ☑ Reproductibilité dev

✗ Single-host uniquement ✗ Pas de HA ✗ Pas pour production

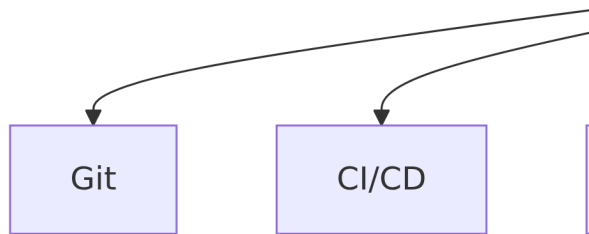
Usage : - Développement local - Prototypage rapide - CI/CD (tests IT) - Démo/proof-of-concept

⚠ Non recommandé en production

5.1 Solutions intégrées vs Best-of-breed

Deux philosophies d'architecture CI/CD

🗄 All-in-One (Integrated)

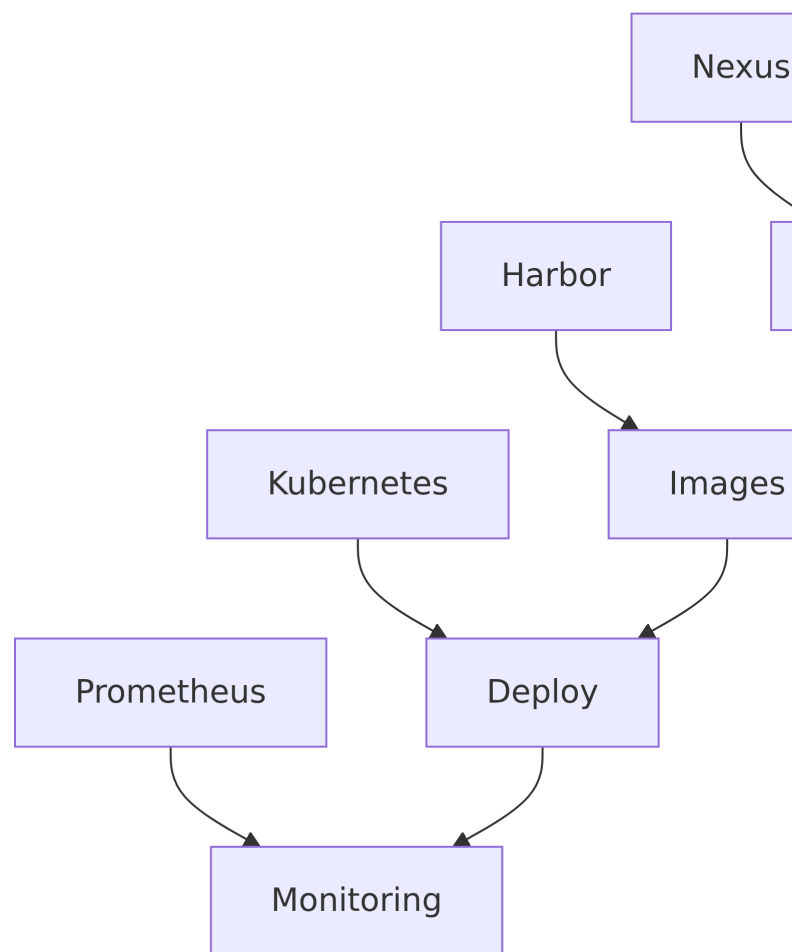


Exemples : - GitLab DevOps Platform - Azure DevOps - JFrog Platform - Atlassian Stack

Avantages : ☒ Intégration native ☒ Single UI/UX ☒ Un seul vendor ☒ Setup rapide ☒ Support unifié

Inconvénients : ✗ Vendor lock-in ✗ Moins flexible ✗ Pas toujours « best tool »

🔧 **Best-of-Breed (Composable)**



Approche : Meilleur outil pour chaque fonction

Avantages : ☒ Meilleur de chaque catégorie ☒ Flexibilité maximale ☒ Portable (multi-cloud) ☒ Éviter vendor lock-in

Inconvénients : ✕ Intégration complexe ✕
Multiple vendors ✕ Maintenance ++ ✕ Courbe
apprentissage

5.2 Tout est conteneurisé

Un écosystème moderne CI/CD s'appuie largement sur des conteneurs : runners, environnements de test, outils de qualité, artefacts et infra déclarative.

Runners / agents CI (conteneurs)

- Scalables et éphémères
- Isolation des builds
- Démarrage rapide

Environnements de test éphémères

- TestContainers, namespaces K8s par PR
- Reproductibilité des scénarios
- Nettoyage automatique

Outils de qualité conteneurisés

- SonarQube, Snyk, Trivy
- Scans CI intégrés (SAST/SCA)
- Rapports et Quality Gates

Applications déployées en conteneurs

- Images versionnées et signées
- Tags par environnement (dev/pr/staging/prod)
- Scan vulnérabilités avant push

Infrastructure as Code (IaC) + conteneurs

- Terraform / Helm / Kustomize
- Déploiement déclaratif et reproductible
- Intégration avec GitOps pour la synchronisation