CI / CD

Architecture de Delivery Moderne : Automatisation, Sécurisation, Pipelines Robustes & GitOps.

CONTINUOUS INTEGRATION • CONTINUOUS DEPLOYMENT • DEVOPS ENGINE

01. Fondations CI/CD

Théorie, cycle de vie logiciel et élimination des tâches manuelles.

CI VS CD : LES DÉFINITIONS

Intégration Continue (CI)

Automatisation de la validation du code dès chaque modification (push).

Build, compilation, tests unitaires et analyse statique instantanés.

Déploiement Continu (CD)

Acheminement automatique et sécurisé de l'artéfact validé vers la production.

Livraison continue (validation manuelle) ou Déploiement continu (100% automatisé).

LES 3 PILIERS DU PIPELINE

Build & Packaging

Compilation du code source et création d'artéfacts immuables (images Docker, packages NPM/JAR) prêts à être déployés.

Validation & Test

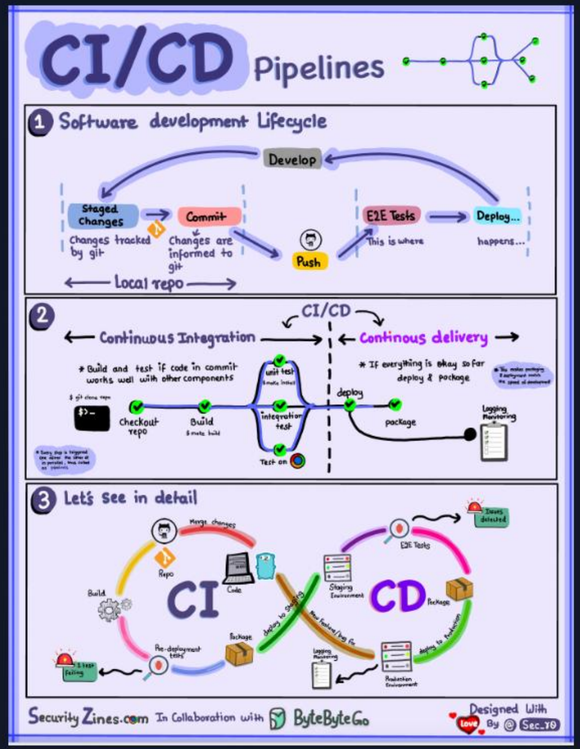
Exécution de la suite de tests (unitaires, intégration, sécurité) garantissant la non-régression applicative.

Delivery & Promotion

Distribution et déploiement progressif à travers les différents environnements (Staging, UAT, Production).

LE PIPELINE DE DELIVERY

- **Commit :** Validation du code par le développeur dans Git.
- **Automated Build :** Déclenchement automatique du build et des tests.
- **Security Scan :** Recherche de vulnérabilités (Snyk, SonarQube).
- **Artifact Registry :** Enregistrement de l'image (Docker Hub, Nexus).
- **Deployment :** Déploiement automatique ou progressif.



LA STRATÉGIE DE VALIDATION

- **Tests Unitaires (70%) :** Valident le comportement isolé de chaque fonction de manière ultra-rapide.
- **Tests d'Intégration (20%) :** Vérifient la bonne communication entre les composants et bases de données.
- **Tests End-to-End (10%) :** Simulent un utilisateur réel sur l'application (Playwright, Cypress).
- **Fail-Fast :** Le pipeline doit s'arrêter immédiatement en cas d'échec pour avertir l'équipe.

02. Outils & Orchestrateurs

Comparer les solutions du marché pour faire le bon choix d'architecture.

COMPARATIF DES ORCHESTRATEURS

Solution	Type	Avantages Clés	Inconvénients Clés
GitHub Actions	SaaS / Hybride	Intégration Git native, écosystème d'actions massif	Dépendance de l'infrastructure GitHub
GitLab CI	SaaS / On-Premise	Pipelines multi-projets puissants, intégré à l'outil	Configuration YAML parfois complexe sur gros projets
Jenkins	Self-Hosted	Personnalisation infinie, plugins abondants	Maintenance lourde ("Jenkins Hell"), scripts complexes
ArgoCD / Flux	Kubernetes Native	GitOps pur, réconciliation automatique d'état	Limité aux déploiements d'infrastructures Cloud-Native

MODÈLE CLOUD VS SELF-HOSTED

Orchestrateurs SaaS (SaaS / Cloud)

Aucune infrastructure à maintenir, scalabilité automatique et mises à jour transparentes.

☁️ Parfait pour se concentrer sur les pipelines plutôt que sur la maintenance système.

Agents Personnalisés (Self-Hosted)

Exécution des runners sur vos propres serveurs (EC2, Kubernetes, Bare Metal).

🖴 Indispensable pour la sécurité des données internes, l'accès au réseau local et la gestion des coûts.

RUNNERS & INFRASTRUCTURE

K8s

AUTO-SCALING

Mise à l'échelle des exécuteurs

L'utilisation de conteneurs éphémères pour exécuter les builds est devenue le standard de l'industrie.

Avec des architectures de type **Kubernetes Event-driven Autoscaling (KEDA)**, vos runners s'activent et s'éteignent instantanément en fonction du nombre de jobs en file d'attente.

Élimination complète des coûts de serveurs inactifs.

03. Syntaxes & Exemples

Conception pratique de pipelines avec GitHub Actions et GitLab CI/CD.

GITHUB ACTIONS WORKFLOW

Le fichier `.github/workflows/ci.yml` structure l'exécution automatisée des tâches sous GitHub :

- ✓ `on: push` déclenche le pipeline à chaque commit.
- ✓ `runs-on: ubuntu-latest` définit l'environnement.

```
name: Node.js CI on: [push, pull_request] jobs: build: runs-on:
ubuntu-latest steps: - uses: actions/checkout@v4 - name: Install
& Test run: | npm ci npm test
```

GITLAB CI/CD PIPELINE

Le fichier `.gitlab-ci.yml` est basé sur le concept clé de **Stages** (étapes séquentielles) et de **Jobs** (tâches parallèles dans chaque étape) :

- `stages` organise l'ordre d'exécution logique.
- `artifacts` permet de persister les fichiers d'un stage à l'autre.

```
stages: - build - test compile: stage: build image: node:20
script: - npm install artifacts: paths: - node_modules/
```

04. Stratégies de Déploiement

Acheminer le code en production sans interruption de service.

PATTERNS DE PRODUCTION

Rolling Update

Mise à jour progressive des instances une à une. Risque mineur de coexistence de deux versions de code en même temps.

Blue-Green

Doublement complet de l'infrastructure. Le trafic est redirigé instantanément vers le nouvel environnement validé.

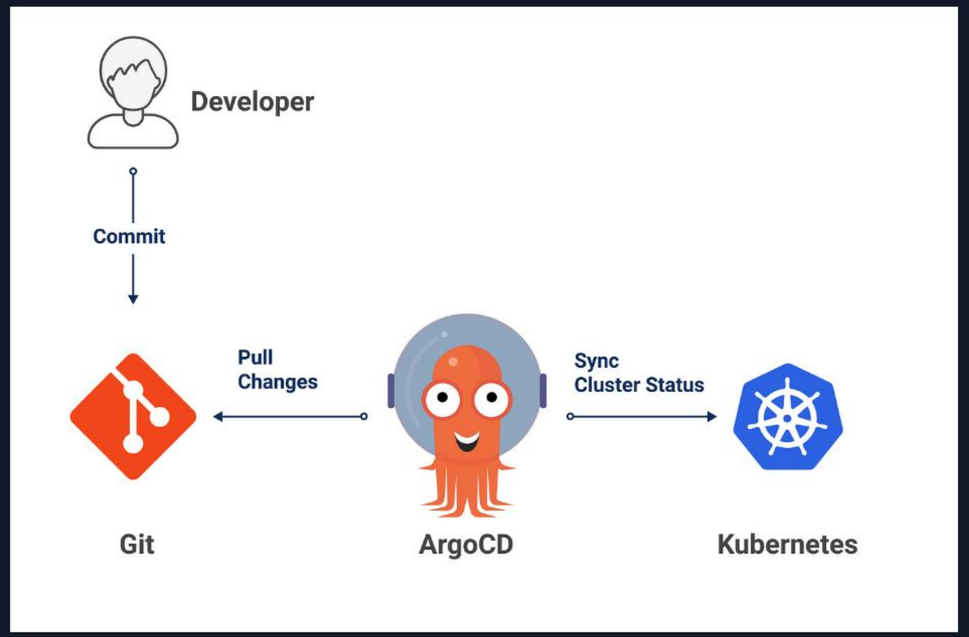
Canary Release

Déploiement initial de la nouvelle version auprès d'un groupe restreint d'utilisateurs (ex: 5%) afin de tester sa stabilité.

L'APPROCHE GITOPS

Le GitOps utilise **Git** comme l'unique source de vérité pour la configuration de l'infrastructure déclarative.

- **Pas d'accès direct :** Les développeurs n'ont plus d'accès direct au cluster (kubectl).
- **Réconciliation :** L'agent GitOps (ex: ArgoCD) aligne l'état du cluster sur celui décrit dans Git.



DEVSECOPS : SÉCURISER LA CI/CD

- **Secrets Management :** Ne stockez jamais d'identifiants en clair. Utilisez des coffres-forts dynamiques (HashiCorp Vault, AWS Secrets Manager).
- **SAST & DAST :** Intégrez des outils d'analyse de code statique (SonarQube) et dynamique pour détecter les failles avant le déploiement.
- **SCA (Software Composition Analysis) :** Analysez en continu les dépendances tierces de vos applications à la recherche de vulnérabilités connues (CVE).

05. Métriques & Performance

Mesurer l'impact de l'automatisation sur les performances de vos équipes.

LES 4 MÉTRIQUES DORA

DORA

DEVOPS RESEARCH

Indicateurs de Performance Elite

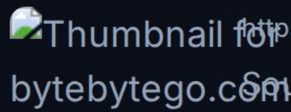
Mesurez l'efficacité de vos pratiques CI/CD à l'aide des quatre standards du marché :

- **Fréquence de déploiement :** Souvent par jour (Elite).
- **Lead Time for Changes :** Moins de 24h du commit à la prod.
- **Taux d'échec des déploiements :** Inférieur à 15%.
- **Temps moyen de restauration (MTTR) :** Moins d'une heure.

Questions ?

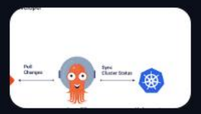
devops-roadmap.sh | pipeline-support@expert-devops.com

IMAGE SOURCES



https://assets.bytebytego.com/diagrams/0140-ci-cd-pipeline.png

Source: bytebytego.com



https://miro.medium.com/v2/resize:fit:1400/0*xlhcKxOp0eh9RHsu

Source: medium.com