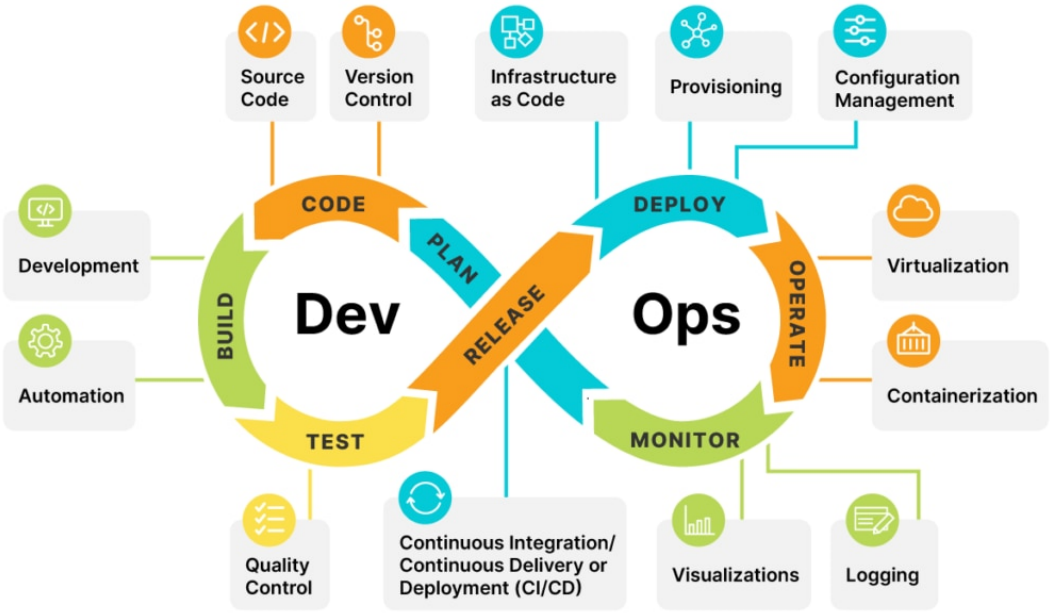
DevOps - CI/CD - les 8 étapes

Qu'est-ce que le DevOps ?

Le DevOps est une combinaison de philosophies culturelles, de pratiques et d'outils qui augmentent la capacité d'une organisation à fournir des applications et des services à haute vélocité. Cette approche permet d'améliorer et de moderniser les produits à un rythme plus rapide que les processus de développement logiciel traditionnels et de gestion de l'infrastructure.



Qu'est-ce que le CI / CD ?

CI (Continuous Integration - Intégration Continue)

L'Intégration Continue (CI), est une pratique de développement logiciel dans laquelle les développeurs intègrent régulièrement leur code dans un répertoire Git. Chaque fois qu'un développeur pousse du code, une série d'automatisations est déclenchée pour compiler, tester et vérifier le code nouvellement ajouté. L'objectif principal de la CI est de détecter les problèmes d'intégration tôt et de garantir que le code nouvellement ajouté ne casse pas l'application existante.

Les étapes typiques incluent la compilation du code (build), l'exécution de tests unitaires et d'autres tests automatisés, la vérification des normes de code, la génération de rapports de test et la fourniture de commentaires rapide aux développeurs sur l'état de leur code. Dans un contexte de conteneurisation, comme avec Kubernetes, c'est également pendant l'intégration continue que sont build les images.

Les outils CI couramment utilisés incluent Jenkins, Travis CI, CircleCI, GitLab CI/CD, Github Actions, et bien d'autres.

CD (Continuous Deployment - Déploiement Continu)

Le CD, ou Déploiement Continu, est une extension de la CI qui vise à automatiser le déploiement des changements de code nouvellement validés vers les environnements de production ou de pré-production.

Contrairement à la CI qui se concentre principalement sur les builds, les tests et les vérifications, le CD s'intéresse à l'automatisation de la mise en production des changements.

Avec le CD, une fois que le code a passé avec succès les étapes de la CI et qu'il a été validé, il est automatiquement déployé dans un environnement cible, généralement à partir d'un pipeline automatisé. Le CD permet de minimiser le temps nécessaire pour mettre en production de nouvelles fonctionnalités ou des correctifs, tout en réduisant les risques liés aux déploiements manuels.

Il existe deux approches principales en matière de CD :

1. **Continuous Deployment (Déploiement Continu)** : dans cette approche, chaque changement de code validé est automatiquement déployé dans l'environnement de production sans intervention humaine. Cela nécessite une confiance élevée dans l'automatisation et une solide suite de tests automatisés.
2. **Continuous Delivery (Livraison Continue)** : ici, les changements de code validés sont automatiquement déployés dans un environnement de pré-production ou de staging. La décision de déployer dans l'environnement de production est prise manuellement, généralement par un responsable technique, après avoir évalué les risques.

Les 8 étapes du Processus DevOps

1. **Plan** : La phase de planification consiste à définir les objectifs du projet, les exigences, et les fonctionnalités attendues. Cela inclut la création de backlogs, la planification des sprints et l'organisation des tâches.

Activités :

- Réunions de planification de sprint
- Établissement des user stories et des tâches
- Définition des critères d'acceptation

---

2. **Code** : La phase de codage est où les développeurs écrivent le code nécessaire pour implémenter les fonctionnalités définies. Le code est généralement versionné à l'aide de systèmes de contrôle de version comme Git.

Activités :

- Écriture et révision du code
- Utilisation de branches pour le développement de fonctionnalités
- Gestion des versions du code source

---

3. **Build** : La phase de build consiste à compiler le code source et à le transformer en artefacts exécutables. Cela peut inclure la compilation du code, la résolution des dépendances et la création de packages.

Activités :

- Compilation du code
- Résolution des dépendances
- Création de builds automatisés avec des outils comme Jenkins ou GitLab CI

---

4. **Test** : La phase de test vise à identifier et corriger les bugs avant le déploiement en production. Cela inclut des tests automatisés comme les tests unitaires, d'intégration, et de performance.

Activités :

- Exécution de tests unitaires et d'intégration
- Tests de performance et de sécurité
- Analyse des résultats de test et correction des bugs

---

5. **Release** : La phase de release consiste à préparer la version finale du logiciel pour le déploiement. Cela inclut la gestion des versions et la préparation des artefacts pour la distribution.

Activités :

- Création des releases
- Documentation des versions
- Validation de la conformité aux critères d'acceptation

---

6. **Deploy** : La phase de déploiement consiste à déployer le logiciel dans des environnements de production. Cette étape peut être automatisée pour minimiser les interruptions de service.

Activités :

- Déploiement des artefacts en production
- Utilisation de techniques comme le déploiement bleu/vert ou les déploiements canari
- Automatisation du déploiement avec des outils comme Kubernetes ou Ansible

---

7. **Operate** : La phase d'exploitation consiste à gérer et à surveiller les applications en production. L'objectif est de garantir leur disponibilité, performance et sécurité.

Activités :

- Surveillance des performances et de la disponibilité
- Gestion des incidents et des résolutions
- Maintenance et mises à jour des systèmes

---

8. **Monitor** : La phase de surveillance vise à collecter et analyser les données de performance et de logs pour détecter les problèmes et améliorer les applications.

Activités :

- Mise en place de systèmes de surveillance et de logging
- Analyse des métriques de performance
- Détection et résolution proactive des problèmes