CI/CD : Présentation et Utilisation

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

CI/CD signifie :
— CI : Continuous Integration (Intégration Continue)
— CD : Continuous Delivery ou Continuous Deployment (Livraison Continue ou Déploiement Continu)

Le CI/CD est un ensemble de pratiques et d'outils permettant d'automatiser le développement, les tests et le déploiement des applications logicielles.

2 Pourquoi utilise-t-on le CI/CD ?

Dans un projet logiciel moderne, plusieurs développeurs travaillent simultanément sur le même code.

Sans CI/CD, on rencontre souvent les problèmes suivants :
— conflits fréquents entre les versions du code,
— erreurs détectées trop tard,
— déploiements manuels et risqués,
— perte de temps dans les tâches répétitives,
— difficultés à maintenir la qualité du projet.

Le CI/CD permet donc de :
— automatiser les tests,
— détecter rapidement les erreurs,
— livrer plus vite les nouvelles fonctionnalités,
— réduire les risques en production,
— améliorer la collaboration entre développeurs.

3 Les deux parties principales du CI/CD

3.1 Continuous Integration (CI)

L'intégration continue consiste à :

— intégrer fréquemment le code dans un dépôt partagé,
— exécuter automatiquement des tests à chaque modification,
— vérifier que le nouveau code ne casse pas l'existant.
Le processus typique du CI est :

1. un développeur modifie le code,
2. il envoie (push) ses changements vers Git,
3. un outil CI exécute automatiquement :
   — les tests unitaires,
   — les tests d'intégration,
   — l'analyse du code,
   — la construction de l'application.

Si une étape échoue, le développeur est immédiatement informé.

3.2 Continuous Delivery / Deployment (CD)

Le CD correspond à la partie déploiement.
Il existe deux niveaux :
— Continuous Delivery : Le code est toujours prêt à être déployé, mais le déploiement final est manuel.
— Continuous Deployment : Le déploiement en production est totalement automatique.
Le CD permet donc :
— d'automatiser la mise en production,
— de réduire les erreurs humaines,
— de livrer plus rapidement aux utilisateurs.

4 Outils utilisés en CI/CD

Plusieurs outils permettent de mettre en place une chaîne CI/CD :
— GitHub Actions
— GitLab CI/CD
— Jenkins
— CircleCI
— Azure DevOps
— Travis CI
Pour les conteneurs et le déploiement :
— Docker
— Kubernetes
— Ansible

— Terraform

5 Exemple de pipeline CI/CD

Un pipeline CI/CD classique contient :

1. Build : compilation du code
2. Test : exécution des tests
3. Analyse : vérification de la qualité du code
4. Packaging : création d'une application exécutable
5. Deployment : mise en production

Exemple simplifié d'un pipeline :

```
1. git push
2. run unit tests
3. build application
4. create docker image
5. deploy to server
```

6 CI/CD dans le Machine Learning (MLOps)

Le CI/CD est aussi très important en Machine Learning.
On parle souvent de :
— CI/CD pour le code
— CI/CD pour les données
— CI/CD pour les modèles
Exemple de pipeline CI/CD ML :

1. entraînement automatique du modèle,
2. évaluation des performances,
3. validation du modèle,
4. déploiement en production.

Des outils comme MLflow ou Kubeflow sont souvent utilisés avec CI/CD.

7 Avantages du CI/CD

Les principaux avantages sont :
— gain de temps,
— meilleure qualité du code,
— déploiements rapides et fiables,
— réduction des bugs en production,
— processus entièrement automatisé.

8 Conclusion

Le CI/CD est aujourd'hui une pratique essentielle du développement logiciel moderne.

Il permet de :
— automatiser le cycle de développement,
— améliorer la collaboration,
— sécuriser les déploiements,
— livrer des applications plus rapidement et avec plus de qualité.

Toute équipe de développement professionnelle utilise aujourd'hui une chaîne CI/CD.