Intégration continue vs livraison continue vs déploiement continu

CI et CD sont deux acronymes fréquemment utilisés dans les pratiques de développement modernes et DevOps. CI signifie intégration continue, une meilleure pratique fondamentale de DevOps où les développeurs fusionnent fréquemment les modifications de code dans un référentiel central où s'exécutent les builds et les tests automatisés. Mais le CD peut signifier soit une livraison continue, soit un déploiement continu.

Devops : Qu'est-ce que DevOps ?

Les meilleures pratiques DevOps incluent la gestion de projet agile, le virage à gauche avec CI/CD, l'automatisation, la surveillance, l'observabilité et la rétroaction continue.

Il est préférable de comprendre DevOps comme un moteur commercial pour améliorer la communication et la collaboration entre les équipes de développement et d'exploitation, afin d'augmenter la vitesse et la qualité du déploiement de logiciels.

C'est une nouvelle façon de travailler qui a de profondes implications pour les équipes et les organisations pour lesquelles elles travaillent.

Quelles sont les différences entre CI, CD et CD ?

Intégration continue

L'intégration continue (CI) est la pratique consistant à automatiser l'intégration des modifications de code de plusieurs contributeurs dans un seul projet logiciel. Il s'agit d'une bonne pratique DevOps principale, permettant aux développeurs de fusionner fréquemment les modifications de code dans un référentiel central où les builds et les tests s'exécutent ensuite. Des outils automatisés sont utilisés pour affirmer l'exactitude du nouveau code avant l'intégration.

Un système de contrôle de version du code source est au cœur du processus CI. Le système de contrôle de version est également complété par d'autres contrôles tels que des tests automatisés de qualité du code, des outils de révision de style de syntaxe, etc.

Les développeurs pratiquant l'intégration continue fusionnent leurs modifications dans la branche principale aussi souvent que possible. Les modifications du développeur sont validées en créant un build et en exécutant des tests automatisés sur le build. Ce faisant, vous évitez les problèmes d'intégration qui peuvent survenir lorsque vous attendez le jour de la publication pour fusionner les modifications dans la branche de publication.

L'intégration continue met l'accent sur l'automatisation des tests pour vérifier que l'application n'est pas interrompue chaque fois que de nouveaux commits sont intégrés dans la branche principale.

Livraison continue

La livraison continue est une approche où les équipes publient des produits de qualité fréquemment et de manière prévisible du référentiel de code source à la production de manière automatisée.

La livraison continue est une extension de l'intégration continue car elle déploie automatiquement toutes les modifications de code dans un environnement de test et/ou de production après l'étape de construction.

Cela signifie qu'en plus des tests automatisés, vous disposez d'un processus de publication automatisé et que vous pouvez déployer votre application à tout moment en cliquant sur un bouton.

En théorie, avec la livraison continue, vous pouvez décider de publier quotidiennement, hebdomadairement, bimensuellement ou selon les besoins de votre entreprise. Cependant, si vous voulez vraiment profiter des avantages de la livraison continue, vous devez déployer en production le plus tôt possible pour vous assurer de libérer de petits lots faciles à dépanner en cas de problème.

Déploiement continu

Le déploiement continu (CD) est un processus de publication de logiciel qui utilise des tests automatisés pour valider si les modifications apportées à une base de code sont correctes et stables pour un déploiement autonome immédiat dans un environnement de production.

Le cycle de publication du logiciel a évolué au fil du temps. L'ancien processus consistant à déplacer le code d'une machine à une autre et à vérifier s'il fonctionne comme prévu était un processus sujet aux erreurs et gourmand en ressources. Désormais, les outils peuvent automatiser l'ensemble de ce processus de déploiement, ce qui permet aux organisations d'ingénierie de se concentrer sur les principaux besoins de l'entreprise plutôt que sur les frais généraux de l'infrastructure.

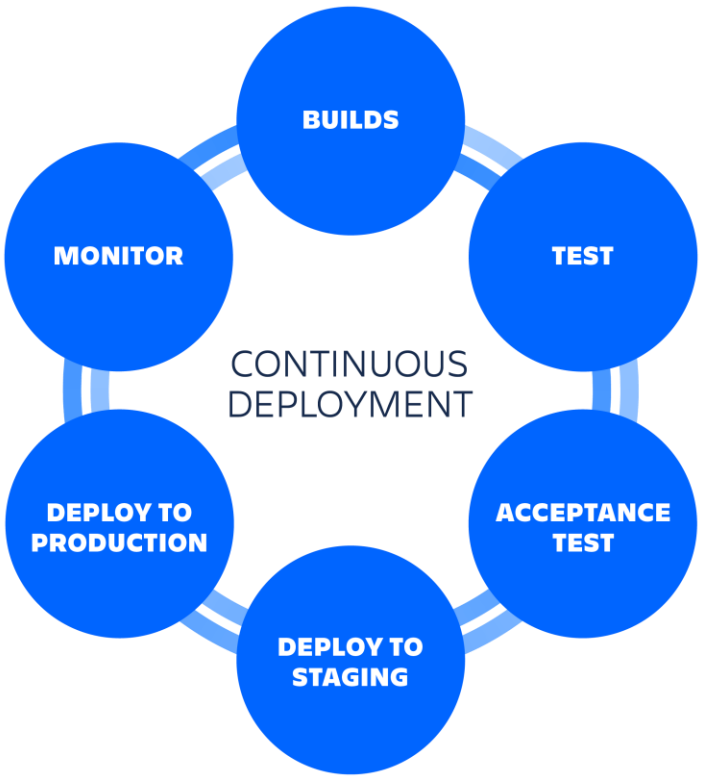
Déploiement continu vs livraison continue

La distinction entre déploiement continu et livraison continue peut prêter à confusion en raison de la nomenclature. Ils sont tous deux abrégés en CD et ont des responsabilités très similaires. La livraison est le précurseur du déploiement. En livraison, il y a une dernière étape d'approbation manuelle avant la mise en production.

Ce qui suit est un exercice mnémotechnique pour aider à se souvenir de la distinction entre les deux. Pensez à recevoir un colis de votre magasin de vente au détail en ligne préféré. En attendant l'arrivée d'un colis, vous coordonnez avec un service de **livraison**. C'est la phase de livraison. Une fois le colis arrivé avec succès, vous ouvrez le colis et examinez son contenu pour vous assurer qu'il correspond aux attentes. Si ce n'est pas le cas, il peut être rejeté et retourné. Si le package est correct, vous êtes prêt à **déployer** et à utiliser le nouvel achat !

Au cours de la phase de livraison, les développeurs examineront et fusionneront les modifications de code qui seront ensuite regroupées dans un artefact. Ce package est ensuite déplacé vers un environnement de production où il attend l'approbation pour être ouvert pour le déploiement. En phase de déploiement, le colis est ouvert et revu avec un système de contrôles automatisés. Si les contrôles échouent, le colis est rejeté.

Lorsque les contrôles réussissent, le package est automatiquement déployé en production. Le déploiement continu est le pipeline complet de déploiement de logiciels automatisé de bout en bout.



Avantages et inconvénients du déploiement continu

Le déploiement continu offre des avantages de productivité incroyables pour les entreprises de logiciels modernes. Il permet aux entreprises de répondre aux demandes changeantes du marché et aux équipes de déployer et de valider rapidement de nouvelles idées et fonctionnalités.

Avec un pipeline de déploiement continu en place, les équipes peuvent réagir aux commentaires des clients en temps réel. Si les clients soumettent des rapports de bogues ou des demandes de fonctionnalités, les équipes peuvent immédiatement répondre et déployer des réponses. Si l'équipe a une idée pour un nouveau produit ou une nouvelle fonctionnalité, elle peut être entre les mains des clients dès que le code a été poussé.

Cependant, les avantages du déploiement continu ont un prix. Bien que le retour sur investissement soit élevé, un pipeline de déploiement continu peut représenter un coût d'ingénierie initial élevé. En plus du coût initial, une maintenance technique continue peut-être nécessaire pour garantir que le pipeline continue de fonctionner rapidement et en douceur.

Outils de déploiement continu

La mise en place d'un déploiement continu nécessite des investissements d'ingénierie substantiels. Voici une liste d'outils nécessaires pour créer un pipeline de déploiement continu.

- Tests automatisés
  La dépendance la plus critique pour le déploiement continu est le test automatisé. En fait, toute la chaîne d'intégration continue, de livraison et de déploiement en dépend. Des tests automatisés sont utilisés pour empêcher toute régression lors de l'introduction d'un nouveau code et peuvent remplacer les révisions manuelles des nouvelles modifications de code.
- Déploiements
  continus La caractéristique distinctive entre le déploiement continu et la livraison est l'étape automatisée d'activation du nouveau code dans un environnement réel. Un pipeline de déploiement continu doit pouvoir annuler un déploiement en cas de déploiement de bogues ou de modifications importantes. Les outils de déploiement automatisés tels que les déploiements vert-bleu sont indispensables à un déploiement continu approprié.

- Surveillance et alertes
  Un pipeline de déploiement continu robuste disposera d'une surveillance et d'alertes en temps réel. Ces outils offrent une visibilité sur la santé de l'ensemble du système et sur l'état avant et après les déploiements de nouveau code. De plus, des alertes peuvent être utilisées pour déclencher un déploiement continu « annuler » pour annuler un déploiement ayant échoué.

Bonnes pratiques de déploiement continu

Une fois qu'un pipeline de déploiement continu est établi, une maintenance et une participation continues de l'équipe d'ingénierie sont requises pour assurer son succès. Les meilleures pratiques et comportements suivants garantiront qu'une équipe d'ingénierie tire le meilleur parti d'un pipeline de déploiement continu.

- Développement piloté par les
  tests Le développement piloté par les tests consiste à définir une spécification de comportement pour les nouvelles fonctionnalités logicielles avant le début du développement. Une fois la spécification définie, les développeurs écriront des tests automatisés correspondant à la spécification. Enfin, le code livrable réel est écrit pour satisfaire les cas de test et correspondre aux spécifications. Ce processus garantit que tout nouveau code est couvert par des tests automatisés en amont. L'alternative à cela est de fournir d'abord le code, puis de produire une couverture de test par la suite. Cela laisse la possibilité d'écarts entre le comportement de spécification attendu et le code produit.

- Méthode unique de déploiement
  Une fois qu'un pipeline de déploiement continu est en place, il est essentiel qu'il soit la seule méthode de déploiement. Les développeurs ne doivent pas copier manuellement le code vers la production ou éditer des éléments en direct. Les modifications manuelles externes au pipeline de CD désynchroniseront l'historique de déploiement, interrompant le flux de CD.

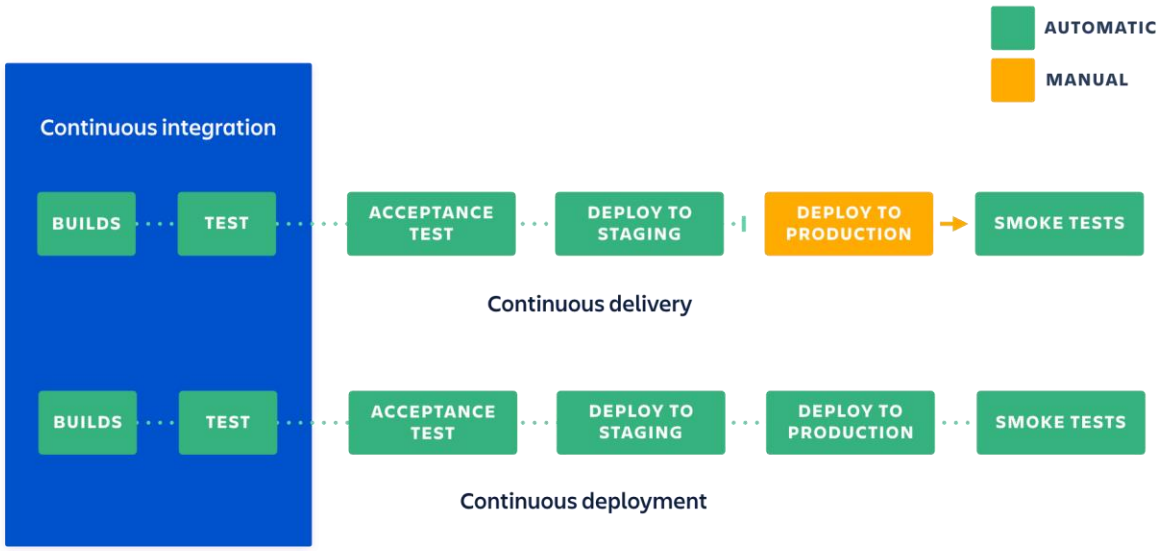
- Conteneurisation
  La conteneurisation d'une application logicielle garantit qu'elle se comporte de la même manière sur n'importe quelle machine sur laquelle elle est déployée. Cela élimine toute une classe de problèmes où le logiciel fonctionne sur une machine mais se comporte différemment sur une autre. Les conteneurs peuvent être intégrés dans le pipeline de CD afin que le code se comporte de la même manière sur la machine d'un développeur que lors des tests automatisés et du déploiement en production.

Le déploiement continu peut être un outil puissant pour les organisations d'ingénierie modernes. Le déploiement est l'étape finale du « pipeline continu » global qui comprend l'intégration, la livraison et le déploiement. La véritable expérience du déploiement continu est l'automatisation au niveau auquel le code est déployé en production, testé pour l'exactitude et automatiquement annulé lorsqu'il est incorrect, ou accepté s'il est correct.

Le déploiement continu va encore plus loin que la livraison continue. Avec cette pratique, chaque changement qui passe toutes les étapes de votre pipeline de production est communiqué à vos clients. Il n'y a pas d'intervention humaine, et seul un test raté empêchera un nouveau changement d'être déployé en production.
Le déploiement continu est un excellent moyen d'accélérer la boucle de rétroaction avec vos clients et de soulager l'équipe car il n'y a plus de Release *Day*. Les développeurs peuvent se concentrer sur la création de logiciels et voient leur travail être mis en ligne quelques minutes après avoir fini de travailler dessus.

Comment les pratiques sont liées les unes aux autres

Pour faire simple, l'intégration continue fait partie à la fois de la livraison continue et du déploiement continu. Et le déploiement continu est comme la livraison continue, sauf que les versions se produisent automatiquement.



Quels sont les bénéfices de chaque pratique ?

Nous avons expliqué la différence entre l'intégration continue, la livraison continue et les déploiements continus, mais nous n'avons pas encore examiné les raisons pour lesquelles vous les adopteriez. La mise en œuvre de chaque pratique a un coût évident, mais il est largement compensé par leurs avantages.

Intégration continue

Ce dont vous avez besoin (coût)

- Votre équipe devra rédiger des tests automatisés pour chaque nouvelle fonctionnalité, amélioration ou correction de bogue.
- Vous avez besoin d'un serveur d'intégration continue capable de surveiller le référentiel principal et d'exécuter les tests automatiquement pour chaque nouveau commit poussé.
- Les développeurs doivent fusionner leurs modifications aussi souvent que possible, au moins une fois par jour.

Ce que vous gagnez

- Moins de bogues sont envoyés à la production car les régressions sont capturées tôt par les tests automatisés.
- La construction de la version est facile car tous les problèmes d'intégration ont été résolus tôt.
- Moins de changement de contexte car les développeurs sont alertés dès qu'ils interrompent la construction et peuvent travailler à la corriger avant de passer à une autre tâche.
- Les coûts de test sont considérablement réduits - votre serveur CI peut exécuter des centaines de tests en quelques secondes.
- Votre équipe d'assurance qualité passe moins de temps à tester et peut se concentrer sur des améliorations significatives de la culture qualité.

Livraison continue

Ce dont vous avez besoin (coût)

- Vous avez besoin d'une base solide en intégration continue et votre suite de tests doit couvrir suffisamment votre base de code.
- Les déploiements doivent être automatisés. Le déclencheur est toujours manuel, mais une fois qu'un déploiement est lancé, aucune intervention humaine ne devrait être nécessaire.
- Votre équipe devra très probablement adopter les indicateurs de fonctionnalité afin que les fonctionnalités incomplètes n'affectent pas les clients en production.

Ce que vous gagnez

- La complexité du déploiement de logiciels a été supprimée. Votre équipe n'a plus besoin de passer des jours à préparer une sortie.
- Vous pouvez libérer plus souvent, accélérant ainsi la boucle de rétroaction avec vos clients.
- Il y a beaucoup moins de pression sur les décisions pour de petits changements, encourageant ainsi une itération plus rapide.

Déploiement continu

Ce dont vous avez besoin (coût)

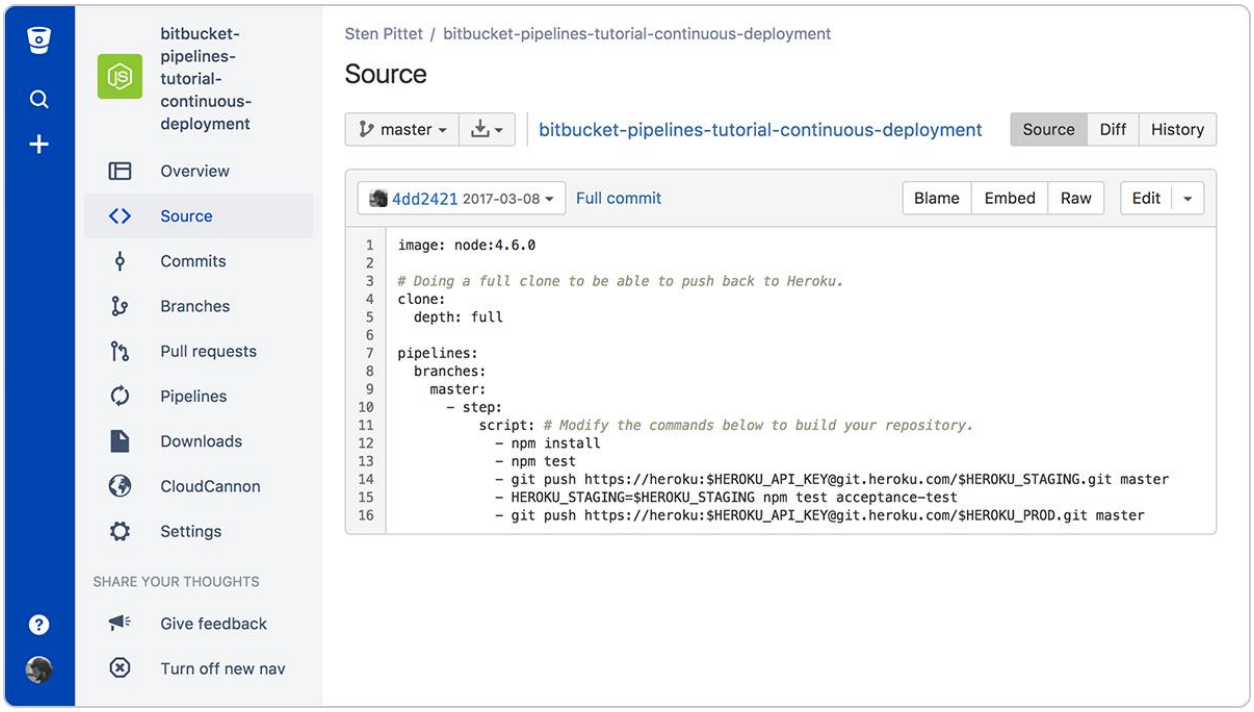
- Votre culture de test doit être optimale. La qualité de votre suite de tests déterminera la qualité de vos versions.
- Votre processus de documentation devra suivre le rythme des déploiements.
- Les indicateurs de fonctionnalité deviennent une partie inhérente du processus de publication des modifications importantes pour vous assurer que vous pouvez vous coordonner avec les autres départements (Support, Marketing, RP...).

Ce que vous gagnez

- Vous pouvez développer plus rapidement car il n'est pas nécessaire d'interrompre le développement pour les versions. Les pipelines de déploiement sont déclenchés automatiquement pour chaque modification.

- Les versions sont moins risquées et plus faciles à corriger en cas de problème lorsque vous déployez de petits lots de modifications.
- Les clients constatent un flux continu d'améliorations et la qualité augmente chaque jour, au lieu de chaque mois, trimestre ou année.

L'un des coûts traditionnels associés à l'intégration continue est l'installation et la maintenance d'un serveur CI. Mais vous pouvez réduire considérablement le coût d'adoption de ces pratiques en utilisant un service cloud comme Bitbucket Pipelines qui ajoute une automatisation à chaque référentiel Bitbucket. En ajoutant simplement un fichier de configuration à la racine de votre référentiel, vous pourrez créer un pipeline de déploiement continu qui sera exécuté pour chaque nouvelle modification transmise à la branche principale.



Passer de l'intégration continue au déploiement continu

Si vous venez de démarrer un nouveau projet sans utilisateur pour le moment, il peut être facile pour vous de déployer chaque commit en production. Vous pouvez même commencer par automatiser vos déploiements et publier votre version alpha en production sans clients. Ensuite, vous augmenteriez votre culture de test et vous vous assureriez d'augmenter la couverture du code au fur et à mesure que vous construisez votre application. Au moment où vous êtes prêt à intégrer les utilisateurs, vous disposerez d'un excellent processus de déploiement continu où toutes les

nouvelles modifications sont testées avant d'être automatiquement mises en production.

Mais si vous avez déjà une application existante avec des clients, vous devriez ralentir les choses et commencer par une intégration continue et une livraison continue. Commencez par implémenter des tests unitaires de base qui s'exécutent automatiquement, pas besoin de se concentrer encore sur l'exécution de tests complexes de bout en bout. Au lieu de cela, vous devriez essayer d'automatiser vos déploiements dès que possible et atteindre un stade où les déploiements dans vos environnements de transfert sont effectués automatiquement. La raison en est qu'en ayant des déploiements automatiques, vous pourrez concentrer votre énergie sur l'amélioration de vos tests plutôt que d'avoir à arrêter périodiquement les choses pour coordonner une version.

Une fois que vous pouvez commencer à publier des logiciels quotidiennement, vous pouvez envisager un déploiement continu, mais assurez-vous que le reste de votre organisation est également prêt. Documentation, accompagnement, commercialisation. Ces fonctions devront s'adapter à la nouvelle cadence des versions, et il est important qu'elles ne passent pas à côté de changements importants pouvant avoir un impact sur les clients.

Lire nos guides

Vous pouvez trouver des guides qui iront plus en profondeur pour vous aider à démarrer avec ces pratiques.

- Premiers pas avec l'intégration continue
- Premiers pas avec la livraison continue
- Premiers pas avec le déploiement continu

Références :

https://mobidev.biz/blog/