Engineer

Livraison continue

maintenir le logiciel prêt à être livré sur demande

Ce que c’est

La livraison continue permet à une équipe de publier une modification logicielle en toute sécurité au moment voulu. Chaque changement suit un parcours reproductible, avec des résultats qui renseignent sur sa capacité à franchir l’étape suivante. Une intervention en production peut toujours dépendre d’une personne qui l’autorise, d’un créneau de maintenance ou d’une contrainte de lancement. Au moment de décider, l’équipe dispose déjà d’une version éprouvée et d’une procédure pour l’installer.

L’intégration continue consiste à intégrer fréquemment de petites modifications et à vérifier le code ainsi réuni. Elle indique rapidement si le logiciel se construit et si les tests passent. La livraison continue ajoute l’installation de la version produite et la vérification de son fonctionnement dans des environnements représentatifs. En déploiement continu, les changements qui remplissent les conditions passent automatiquement en production. Une approbation avant cette étape reste compatible avec la livraison continue si elle porte sur une version déjà prête.

Installer du code et ouvrir une fonctionnalité peuvent être deux événements distincts. La nouvelle version peut fonctionner avec la fonctionnalité désactivée. Plus tard, l’équipe peut donner l’accès à quelques utilisateurs avant de l’ouvrir plus largement. Les deux décisions ont intérêt à avoir un responsable et des vérifications propres. On peut ainsi confirmer que l’installation elle-même est sûre, puis observer les effets du changement une fois qu’il devient visible.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que la livraison continue ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des changements testés, publiés et réversibles, pour livrer aux clients un mardi après-midi plutôt qu'un vendredi soir tard.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Déploiement SaaS GIS automatisé : GitHub Actions — déploiement, données et reporting automatisés

Déploiement SaaS GIS automatisé : GitHub Actions Contactez-nous →

Pourquoi c’est important

Une sortie qui repose sur une séquence retenue par une seule personne est difficile à confier à un collègue. Un parcours commun conserve le résultat de chaque étape. Il permet de distinguer un test qui échoue, un environnement indisponible et une approbation qui n’est pas encore donnée. Consignez l’identifiant de la version réellement installée. Lorsque le diagnostic commence, une étiquette comme `latest` ne suffit pas à dire quels fichiers étaient en service au moment du problème.

Si chaque sortie mobilise beaucoup de temps, plusieurs modifications attendent le prochain créneau. Lorsqu’une régression apparaît, l’équipe doit examiner toutes les causes que ce lot a pu introduire. Une procédure routinière facilite les sorties plus petites et permet de suivre l’effet de chacune. Les mêmes preuves permettent aussi de justifier un report si, par exemple, un service voisin traverse une intervention programmée.

Un changement peut traverser plusieurs dépôts. Les migrations de base de données, les règles réseau et les paramètres d’authentification peuvent relever de responsables différents. Il faut donc recenser les combinaisons qui doivent fonctionner ensemble. Un environnement de test représentatif permet de les vérifier avant le passage en production. Documentez la procédure de façon à ce qu’une autre personne autorisée puisse reprendre l’intervention avec les mêmes éléments.

À quoi cela ressemble en pratique

Le cas sur les chaînes CI/CD construites avec GitHub Actions décrit des flux séparés pour les contrôles de code, les tests et l’installation dans différents environnements. L’interface, l’API et la base de données ont chacune reçu une image de base adaptée à leur rôle. Le cas indique qu’un parcours défini et des vérifications visibles ont fait des mises en production une activité courante. Les choix de paquets et d’approbation dépendent du projet.

Imaginons une application qui ajoute l'export d'un rapport. Le flux enregistre la révision du code, lance les tests puis construit un paquet doté d'un identifiant unique. Ce paquet est installé en préproduction. Un compte de test autorisé télécharge un rapport issu de données représentatives. La vérification porte sur les colonnes produites et les droits d'accès. La personne qui approuve la publication voit que le test a porté sur le même paquet. Celui-ci rejoint alors la production avec ses paramètres propres. Un contrôle rapide confirme que l'export fonctionne et que le parcours existant répond toujours.

Un échec en préproduction doit laisser la version visible avec la cause de l'arrêt. En production, on doit pouvoir retrouver la version précédente et le premier indicateur qui a changé. Répétez le parcours dans un environnement maîtrisé, y compris avec un service dépendant volontairement arrêté. Si la base de données change, vérifiez aussi que l'ancienne version de l'application peut lire l'état intermédiaire pendant une éventuelle récupération.

Où les problèmes surviennent

Reconstruire le paquet dans chaque environnement peut rompre la continuité des preuves. Une dépendance ou la version d'un compilateur peut évoluer entre deux installations fondées sur le même code. Faites progresser le paquet identifié qui a passé les contrôles lorsque les environnements sont compatibles. Si une plateforme exige son propre paquet, conservez son identifiant et les résultats qui lui correspondent. Le nom commercial de la version ne précise pas les octets qui ont été testés.

Une approbation peut devenir une procédure de publication parallèle. Donnez à la personne responsable le paquet, le résultat des contrôles, l'environnement cible et les risques qui restent ouverts. Elle peut alors décider. Si le passage repose ensuite sur une longue série de commandes propres à cette personne, le dernier trajet reste difficile à déléguer. Réservez les décisions humaines aux cas qui font appel au jugement et rendez les opérations habituelles reproductibles.

Un voyant vert perd sa valeur si un test ne passe qu'à l'occasion d'une relance. Examinez ces tests instables et vérifiez que les contrôles de santé détectent ce qui compte pour le service. Pour une modification risquée, prévoyez une période d'observation avant de l'étendre. Décidez qui peut l'arrêter et à quel signal. Après une interruption, reprenez avec un paquet identifié et un état dont l'équipe a vérifié la compatibilité.

Pour aller plus loin

- DORA sur la livraison continue définit la capacité de publier une version à la demande.
- DORA sur l'intégration continue explique l'intégration fréquente et le retour rapide des tests.
- DORA sur l'automatisation du déploiement décrit des opérations reproductibles entre environnements.
- Commandes de déploiement dans GitHub Actions documente les environnements et leurs règles de protection.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que la livraison continue ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/continuous-delivery/