

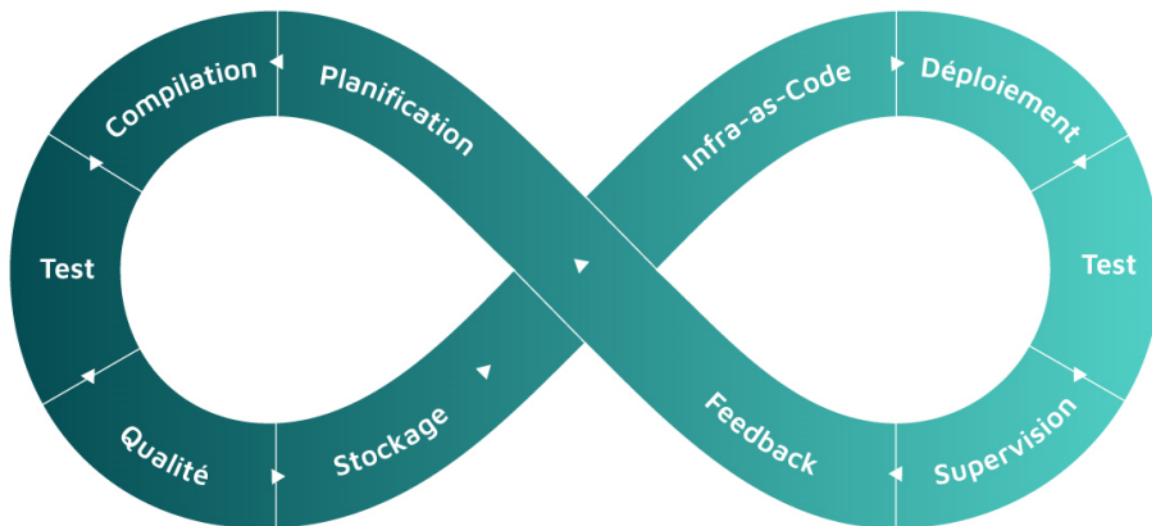
La chaîne d'intégration et de livraisons continues

Table des matières

I. Introduction	3
II. Intégration continue pour gérer la qualité du code	3
III. Livraison continue ou Continuous Delivery (CD)	4
IV. Étendre l'agilité jusqu'aux DevOps	5
V. Auto-évaluation	6
Solutions des exercices	7

I. Introduction

Quels sont les « 4 C » de la chaîne continue ?



En suivant la logique de gestion de flux qui vise à augmenter le rythme et fluidifier le travail, la méthode Agile utilise des outils permettant d'automatiser certaines étapes et de réduire les interventions des membres de l'équipe Scrum.

Globalement, on identifie 4 niveaux « d'écoulement » du flux continu : l'exploration continue (CE), la livraison continue (CD), l'intégration continue (CI) et le déploiement continu (continuous exploration, continuous delivery, continuous integration et continuous deployment).

L'exploration continue se situe du point de vue « métier » et vise à alimenter en continu l'équipe Scrum en opportunités d'amélioration apportées par des user stories. Elle concerne les outils mentionnés plus haut comme Jira, Trello, Confluence, Miro, etc.

II. Intégration continue pour gérer la qualité du code

Définition Qu'est-ce que l'intégration continue ?

L'intégration continue vise à intégrer les développements au fil de l'eau, et non plus l'ensemble des développements en fin de lot. En cela, elle contribue à l'apprentissage qui est, lui aussi, continu dans un processus Agile. Elle permet de prendre connaissance du code développé et des éventuelles régressions ou anomalies apportées par le nouveau code, au plus tôt.

Organiser l'automatisation avec l'orchestrateur

L'orchestrateur permet, en étant couplé à la gestion de configuration, d'automatiser les fusions de code (opérations de « merge »), la production d'un exécutable (opération de « build ») et le déploiement sur un serveur d'intégration dédié aux tests du même nom.

En fait, il va organiser l'ensemble de la continuité, pas seulement l'intégration.

Concrètement, ce sont des triggers (ou déclencheurs) qui lancent la construction d'un exécutable et, en cas de succès, lancent les tests automatiques qui auront été définis ainsi que les analyses de codes. Ces déclencheurs scrutent toute modification du code.

Quels peuvent être les outils d'orchestration et d'analyse de code ?

Du côté du marché des orchestrateurs

Parmi les références du marché, on retrouve la solution Open Source *Jenkins*, ou encore *Bamboo* de l'éditeur Atlassian, *Visual Studio Team Services* de Microsoft, *Gitlab CI* (1 version libre), *Travis CI*, *Concourse CI* et *Azure DevOps* de Microsoft en mode SaaS.

Les tests automatiques permettent de décharger l'équipe de la charge de test visant à vérifier la possibilité de régression. Les analyses de code donnent des indications sur la qualité du code et très souvent, proposent des suggestions d'amélioration.

L'analyse de code travaille, quant à elle, à réduire la dette technique qui peut vite polluer une équipe, en permettant dès la première intégration de réaliser des quick wins par une série de corrections simples, ou en identifiant une mauvaise pratique à ne pas reproduire.

Du côté du marché d'analyse de code

Également Open source, *SonarQube* accompagne souvent *Jenkins*. On retrouve aussi *Cast* de Cast Software, ou *GitLab Code Quality*, etc. de GitLab. Pour l'analyse de code, les solutions sont parfois plus nombreuses et spécifiques du fait de l'existence de différents langages. *GitLab Code Quality*, notamment, est plus générique, car il s'appuie sur des plug-ins Open Source pour gérer les différents langages, en particulier pour les parties front et back d'un même projet.

III. Livraison continue ou Continuous Delivery (CD)

En quoi consiste la livraison continue ?

La livraison est l'étape qui permet d'installer la solution sur un environnement intermédiaire (hors production, donc) qui implique la prise en compte de la définition et de la structure de l'environnement. La livraison continue est ainsi sa réalisation de manière automatisée. L'automatisation implique non seulement moins de ressources humaines, mais aussi moins de risques d'erreurs : elle oblige à définir des règles suffisantes et nécessaires qui, au fil des incidents rencontrés, sont de plus en plus fiables.

L'automatisation de la livraison s'appuie sur un concept fondamental qu'on nomme infrastructure as code, dans la lignée de l'IaaS pour infrastructure as a service.

Qu'est-ce que l'infrastructure as code ?

L'infrastructure as code signifie littéralement que l'infrastructure, c'est-à-dire toute la partie physique de l'informatique, devient du logiciel. Cela est rendu possible par l'utilisation omniprésente, aujourd'hui, de la virtualisation des ressources dites machine et réseau. Désormais, on installe manuellement dans de grandes armoires des boîtiers contenant de la mémoire vive, des disques, des processeurs et des équipements réseau. Ces ressources sont ensuite déclarées dans un outil qui permet de gérer toutes les opérations de mise en œuvre des environnements sans intervenir physiquement. Dès lors, il devient possible d'automatiser l'installation.

Pour la livraison continue, une partie du travail consiste à mettre en œuvre le concept d'infrastructure as code. Il reste ensuite à réaliser l'installation et le paramétrage sur l'environnement cible, donc à définir des règles, des critères et des actions pour le faire de manière automatique. Des garde-fous sont cependant mis en place pour que les étapes ne s'enchaînent pas sans vérification. Utiliser des outils comme *Docker* permet, par exemple, d'obtenir des images des serveurs. On peut aussi repartir d'une case « propre » comportant tous les composants nécessaires et sur laquelle on installe uniquement ce qui a été développé.

Attention À noter !

On n'installe pas la solution si, par exemple, des warnings sont remontés par l'opération de build, ou s'il n'y a pas d'exécutable produit.

Qu'est-ce que le déploiement continu ?

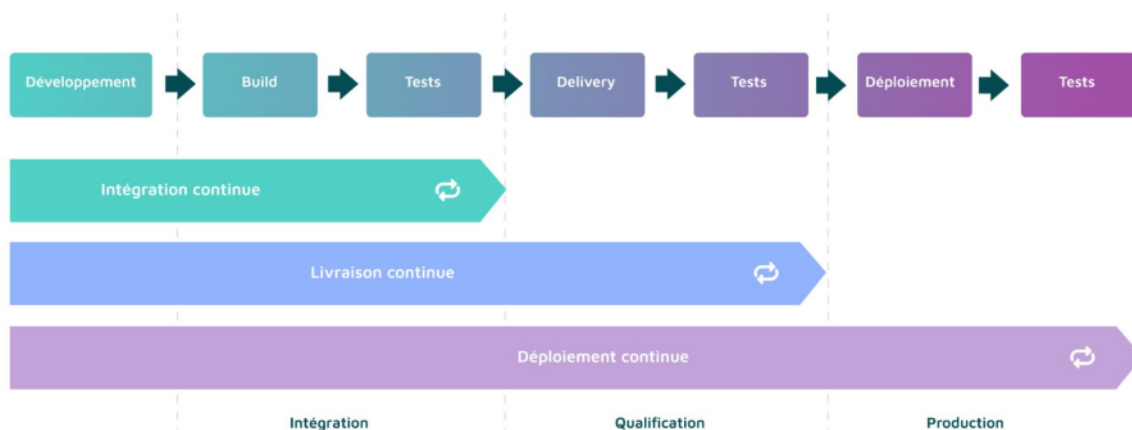
Après la livraison continue, le déploiement continu peut venir compléter la chaîne automatisée en réalisant la mise en production.

Le déploiement se distingue de la livraison sur la plateforme cible, car le déploiement met en production, donc à disposition des clients et/ou utilisateurs finaux.

Ce déploiement doit tenir compte :

- De la continuité du service, c'est-à-dire qu'il doit continuer à utiliser le produit lorsque l'installation en production est effectuée.
- De l'intégrité des données de production, c'est-à-dire qu'il doit veiller à ce que l'installation ne dégrade pas les données des utilisateurs finaux.
- De l'interruption de service si nécessaire, c'est-à-dire qu'il doit identifier les sessions actives, informer de la coupure et de la réouverture du service, etc.

Les environnements/étapes d'un projet et les différentes approches :



IV. Étendre l'agilité jusqu'aux DevOps

Définition Qu'est-ce que DevOps ?

L'origine du nom DevOps vient de la contraction des mots « *development* » ou « *developers* », et « *operational* ». Les premiers développent la solution, les seconds gèrent la partie opérationnelle, c'est-à-dire l'exécution de la solution et donc l'infrastructure qui l'héberge et les composants technologiques : serveur applicatif, OS, base de données (hors contenu), etc.

Quel est l'impact de DevOps ?

Dans cette section consacrée à l'approche DevOps, nous retrouverons les sujets déjà abordés sur la chaîne continue de production du produit, en matière d'objectifs et de pratiques. Ce qu'on peut souligner ici, c'est la valeur ajoutée du nom DevOps apportée par la jonction entre le développement et l'opérationnel et dont l'impact est à la fois culturel et organisationnel.

Les 2 parties entretiennent traditionnellement une relation client-fournisseur. Du côté « *développement* », le mot d'ordre est de tout oser pour innover et augmenter la valeur produite. Du côté « *opérations* », on cherche à fiabiliser, garantir le bon fonctionnement et éviter les incidents et contrariétés pour les utilisateurs. Et toute livraison est un changement qui peut apporter son lot de problèmes et mettre à mal la fiabilité.

La révolution culturelle consiste ici à mettre en œuvre la même dynamique que celle créée par la partie « *développement* » afin de produire une valeur plus grande et meilleure, mais sur l'ensemble de l'équipe en incluant la partie « *opérations* ». On se soucie de la partie opérationnelle quand on développe, et on s'intéresse aux possibilités d'amélioration quand on supporte la solution.

Sur le plan organisationnel, cela se traduit par la sollicitation d'une personne « *opérationnelle* » dédiée au projet et qui, dans la mesure du possible, est intégrée à la Scrum team. On appelle cette personne le « *technicien DevOps* ». Cette personne devient l'interlocuteur privilégié des équipes support et infrastructure.

Au sein de l'équipe Scrum, le technicien DevOps a la responsabilité des outils d'intégration et de livraison continue. Il coach également les autres membres de l'équipe sur l'utilisation de ces outils. Pour une meilleure productivité, il faut, sur chaque projet, bien positionner le curseur entre « *Dev* » et « *Ops* » : qui fait quoi avec les outils ?

Pour des raisons d'efficacité, les développeurs :

- Ne confient pas tout au technicien DevOps, car cela engendrerait un risque de créer un goulot d'étranglement et de devoir attendre.
- Ne font pas tout eux-mêmes, car ce serait aller au-delà de leurs compétences. Cela pourrait aussi dégrader un environnement dont le rétablissement prendra un temps certain, ou les rendre excessivement lents dans l'accomplissement des actions.

Enfin, comme le technicien DevOps fait idéalement partie de l'équipe, cette personne assiste aux événements Scrum de l'équipe et partage le même niveau d'information (sprint planning, rétrospectives, daily, etc.).

V. Auto-évaluation

Exercice 1 : Quiz

[solution n°1 p.9]

Question 1

Parmi les propositions suivantes, lesquelles ne représentent pas une partie de la chaîne continue de production ?

- ☐ Continuous development
- ☐ Continuous delivery
- ☐ Continuous testing
- ☐ Continuous integration

Question 2

Le technicien DevOps est l'orchestrateur de la chaîne continue.

- ☐ Vrai
- ☐ Faux

Question 3

L'Infrastructure as Code est :

- ☐ Impossible sans la virtualisation des serveurs
- ☐ Une théorie plus qu'une réalité
- ☐ Un paramétrage pour indiquer à la solution les caractéristiques matérielles
- ☐ Un moyen de limiter les problèmes d'installation

Question 4

Le déploiement continu permet une installation automatisée et la livraison continue permet la constitution du package nécessaire pour l'installation.

- ☐ Vrai
- ☐ Faux

Question 5

L'application de DevOps est :


- ☐ Un changement organisationnel
- ☐ Un changement culturel
- ☐ Un changement d'outils
- ☐ Les 3 à la fois

Solutions des exercices

Exercice p. 6 Solution n°1**Question 1**

Parmi les propositions suivantes, lesquelles ne représentent pas une partie de la chaîne continue de production ?


- ☐ Continuous development
- ☐ Continuous delivery
- ☒ Continuous testing
- ☐ Continuous integration

 Le développement se fait en continu, mais ne constitue pas une approche automatisée. Les tests sont, quant à eux, répartis sur la chaîne de production (dans l'intégration continue, dans la livraison continue, etc.) sans pour autant incarner l'un des 4 niveaux d'écoulement du flux continu. Pour rappel, ceux-ci sont l'exploration continue (CE), la livraison continue (CD), l'intégration continue (CI) et le déploiement continu (continuous exploration, continuous delivery, continuous integration et continuous deployment).

Question 2

Le technicien DevOps est l'orchestrateur de la chaîne continue.


- ☐ Vrai
- ☒ Faux

 Même si le technicien DevOps est un leader technique sur les aspects « *intégration* » et « *livraison* » par sa connaissance des procédures et des outils, les décisions sont prises au niveau de l'équipe. L'orchestrateur est le nom de l'outil qui vérifie les conditions de lancement d'une opération puis lance et vérifie le résultat de l'opération, etc.

Question 3

L'Infrastructure as Code est :

- ☒ Impossible sans la virtualisation des serveurs
- ☐ Une théorie plus qu'une réalité
- ☐ Un paramétrage pour indiquer à la solution les caractéristiques matérielles
- ☒ Un moyen de limiter les problèmes d'installation

 L'infrastructure as code a pour objectif de gérer les ressources physiques et du logiciel, il est donc indispensable de virtualiser. Comme l'installation passe d'une opération manuelle à une programmation, cela rend possible l'automatisation de l'installation, ce qui implique moins de risques d'erreurs. Les actions ne sont plus éphémères : il s'agit de code qui peut être amélioré, chaque incident faisant l'objet d'une correction pour que ça ne se reproduise plus.

Question 4

Le déploiement continu permet une installation automatisée et la livraison continue permet la constitution du package nécessaire pour l'installation.

- ☐ Vrai
- ☒ Faux

- Q Les deux visent une installation automatisée. Mais le déploiement continu concerne l'environnement de production et toutes les conditions et contraintes dues à son utilisation permanente par les utilisateurs et/ou les clients. Il faut soit se débrouiller pour installer sans couper le service, soit cadrer l'intervention pour qu'elle se réalise dans une période d'interruption de service (c'est l'exemple du fameux serveur en maintenance que l'on cherche désormais à éviter).

Question 5

L'application de DevOps est :

- ☐ Un changement organisationnel
- ☐ Un changement culturel
- ☐ Un changement d'outils
- ☒ Les 3 à la fois

- Q Les outils ne sont pas les mêmes, car on n'installe plus manuellement sur des serveurs physiques, et ils doivent permettre de définir des règles de déclenchement et d'exécution. On change l'organisation de manière à casser les barrières entre développement et support, ce qui est aussi un changement culturel. D'autant plus culturel que le support s'*« agile »* par ce biais.