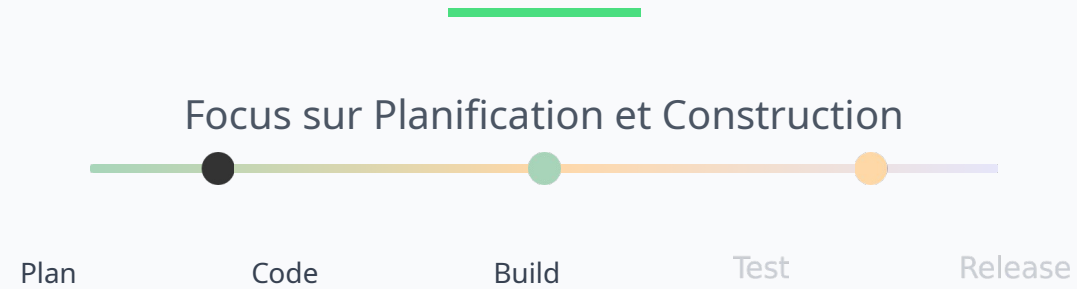


Continuous Development

Pilier de l'Agilité DevOps



Introduction



Qu'est-ce que le Développement Continu ?



Objectifs du Développement Continu



Principes Fondamentaux

Synthèse



Bénéfices du Développement Continu



Lien avec l'Intégration Continue



Phase 1: Planification (Plan)



Gestion du Backlog



User Stories



Planification de Sprint



Outils de Planification



Phase 2: Construction (Build)



Gestion de Version avec Git



Stratégies de Branches



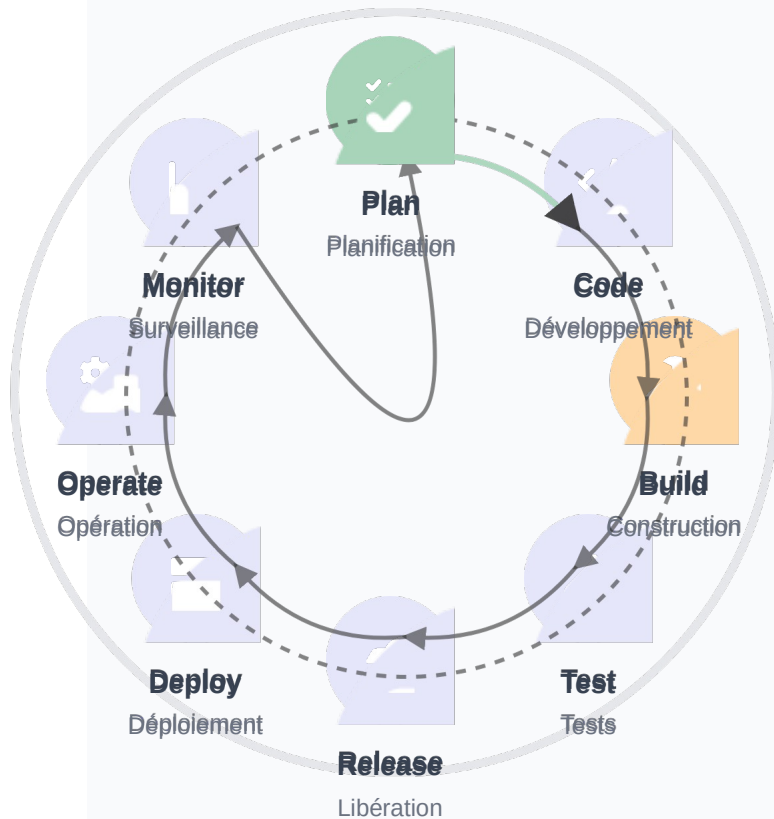
Bonnes Pratiques d'Écriture de Code



Automatisation du Build



Outils de Build



Le Développement Continu

Est la première étape **cruciale** du cycle DevOps, initiant l'ensemble du processus.

Il s'inscrit dans la séquence **Plan -> Code -> Build -> Test -> Release -> Deploy -> Operate -> Monitor**, posant les bases pour les phases ultérieures d'intégration, de déploiement et de surveillance continues.



Focus de la Présentation

Cette formation se concentre particulièrement sur les deux premières phases du cycle : **Plan** et **Build**, qui constituent les fondations du Développement Continu.

Définition

Le Développement Continu est une pratique fondamentale de l'ingénierie logicielle au sein de l'approche DevOps. Il implique que les membres d'une équipe collaborent pour planifier, concevoir et écrire du code de manière itérative et constante.

Caractéristiques



Itératif et continu

Le développement se fait par petites itérations gérables plutôt que par grands sauts.



Intégration constante

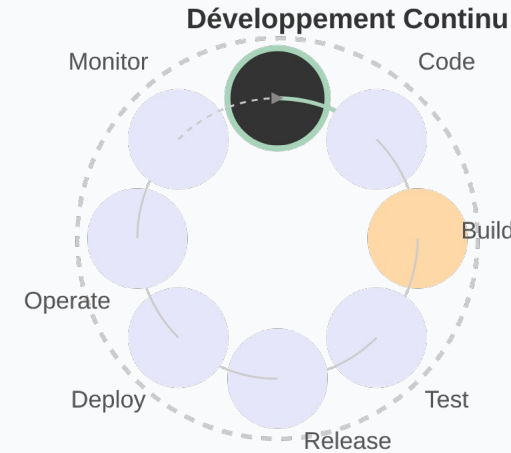
Le code est maintenu dans un état toujours prêt pour l'intégration.



Collaboration

Encourage la communication et la coopération entre les développeurs, les Product Owners et les parties prenantes.

Le Développement Continu dans le cycle DevOps



Key Insight: Le Développement Continu est la première étape cruciale du cycle DevOps, initiant l'ensemble du processus.



Accélérer le développement

En découpant le travail en petites itérations gérables, les équipes peuvent livrer des fonctionnalités plus rapidement et de manière plus prévisible.



Favoriser une culture de collaboration

Le Développement Continu encourage la communication et la coopération entre les développeurs, les Product Owners et les autres parties prenantes.



Améliorer la qualité du code

L'accent est mis sur des révisions de code régulières, l'écriture de tests unitaires et le respect des bonnes pratiques de codage, ce qui réduit les défauts et la dette technique.



Automatisation

L'automatisation des tâches répétitives, comme le build et les tests unitaires, est un principe central pour garantir l'efficacité et la fiabilité.



Réduire les risques liés aux changements

Les modifications sont intégrées fréquemment et en petites quantités, ce qui facilite l'identification et la correction rapide des problèmes.



Cycles de feedback courts

Obtenir un retour rapide sur le code développé permet d'ajuster le tir rapidement et d'améliorer continuellement le produit.



Automatisation

Automatiser les tâches répétitives comme le build et les tests unitaires pour garantir l'efficacité et la fiabilité du processus de développement.



Cycles de Feedback Courts

Obtenir un retour rapide sur le code développé permet d'ajuster le tir rapidement et d'améliorer continuellement le produit.



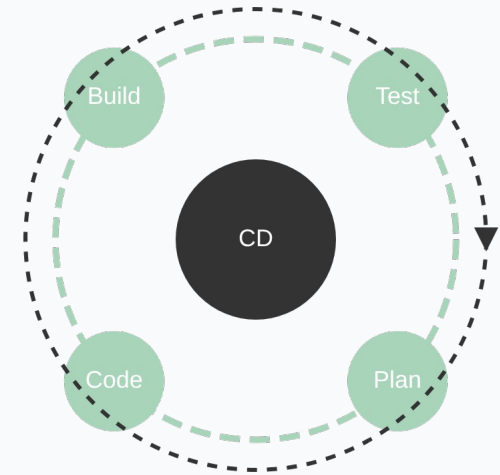
Itérations Rapides

Découper le travail en petites itérations gérables pour livrer des fonctionnalités plus rapidement et de manière plus prévisible.



Collaboration

Favoriser la communication et la coopération entre les développeurs, les Product Owners et les autres parties prenantes pour créer une culture collaborative.



Intégration Fréquente

Les modifications sont intégrées fréquemment et en petites quantités, ce qui facilite l'identification et la correction rapide des problèmes.

Définition

La phase de Planification est le point de départ fondamental de tout cycle de développement logiciel. C'est à cette étape que les idées initiales et les besoins des utilisateurs sont recueillis, analysés et transformés en tâches concrètes et réalisables.

Elle pose les bases de ce qui sera construit, en définissant la vision, les objectifs et les fonctionnalités du produit.

Importance Stratégique



Direction Claire

Fournit une vision et des objectifs partagés pour guider l'équipe



Priorisation

Permet de se concentrer sur les fonctionnalités à plus forte valeur ajoutée



Alignement

Assure que toutes les parties prenantes partagent la même compréhension

Le Processus de Planification



Recueil des Idées

Collecte des besoins et des attentes des utilisateurs



Création du Backlog

Organisation et priorisation des fonctionnalités



Planification de Sprint

Définition des objectifs à court terme et engagement de l'équipe



Transition vers le Build

Début de la phase de construction avec des tâches claires





Product Backlog

Une liste ordonnée de toutes les fonctionnalités, améliorations, corrections de bugs et tâches techniques nécessaires pour un produit.

Il s'agit du référentiel unique des exigences du produit.



Gestion

 Le Product Owner est responsable de la gestion

 Collaboration avec l'équipe de développement

 Interaction avec les parties prenantes



Contenu du Backlog



Epics (grandes fonctionnalités)



User Stories (petites fonctionnalités orientées utilisateur)



Tâches techniques



Collaboration



Product Owner



Développeurs



Parties prenantes



Qu'est-ce qu'une User Story ?

Une User Story est une description simple et concise d'une fonctionnalité du point de vue de l'utilisateur final.

Elle facilite la compréhension des besoins et se concentre sur la valeur ajoutée pour l'utilisateur.

Importance

 Capture les besoins utilisateur

 Fait pivoter sur la valeur

 Facilite la collaboration



Structure



En tant que **rôle**

Identifie le type d'utilisateur



Je veux **action**

Définit la fonctionnalité



Afin de **bénéfice**

Explique la valeur



Exemple

CLIENT

En tant que client,

Je veux pouvoir réinitialiser mon mot de passe,

Afin de retrouver l'accès à mon compte si je l'oublie.



Qu'est-ce que la Planification de Sprint ?

Une cérémonie clé dans les méthodologies agiles, notamment Scrum, où l'équipe de développement sélectionne les éléments les plus prioritaires du Product Backlog pour le prochain sprint.

Processus de Planification

Équipe de développement et Product Owner se réunissent

Sélection des éléments prioritaires du Product Backlog

Estimation de l'effort nécessaire (story points)

Décomposition des user stories en tâches plus petites

Création du Sprint Backlog

Le Sprint

Une période de temps fixe, généralement de 1 à 4 semaines, durant laquelle l'équipe se concentre sur la livraison des fonctionnalités planifiées.

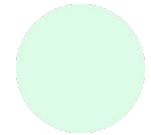
Transformation du Product Backlog en Sprint Backlog



Product Backlog
Toutes les
fonctionnalités ordonnées
par priorité



**Sélection &
Estimation**
Choix des éléments
à réaliser et évaluation



Sprint Backlog
Tâches à accomplir durant
le sprint

Avantages de la Planification de Sprint

- ✓ Alignement sur les objectifs du produit
- ✓ Visibilité sur le travail à accomplir
- ✓ Engagement de l'équipe
- ✓ Base pour la gestion des versions

Les outils de planification agiles facilitent la gestion de projet en visualisant le backlog, en suivant l'avancement des tâches et en favorisant la collaboration.



Jira

Un outil complet de gestion de projet agile, largement utilisé pour le suivi des bugs, la gestion des tâches et la planification de sprints.

Fonctionnalités clés:

- ✓ Suivi des bugs et des tâches
- ✓ Planification détaillée de sprints
- ✓ Tableaux Kanban et backlogs



Trello

Un outil de gestion de projet visuel basé sur des tableaux Kanban, idéal pour organiser des tâches et des idées de manière simple et intuitive.

Fonctionnalités clés:

- ✓ Interface visuelle et intuitive
- ✓ Tableaux Kanban personnalisables
- ✓ Intégrations tierces



Azure DevOps Boards

Fait partie de la suite Azure DevOps, il offre des fonctionnalités de planification, de suivi des tâches et de gestion de backlog pour les équipes de développement.

Fonctionnalités clés:

- ✓ Intégré avec Azure DevOps
- ✓ Tableaux Kanban et backlogs
- ✓ Suivi des tâches et des itérations



Le choix de l'outil de planification dépend des besoins spécifiques de l'équipe, du projet et de l'organisation. Tous ces outils aident à mettre en pratique les phases de planification du Développement Continu.

Qu'est-ce que la phase de Construction (Build) ?



Transformation de l'idée en réalité tangible

L'étape cruciale où les idées et les plans se transforment en réalité tangible.



Écriture et assemblage du code

Le code est écrit, assemblé et préparé pour devenir un artefact logiciel fonctionnel.



Préparation pour le déploiement

Le code est prêt à être testé et potentiellement déployé dans l'environnement de production.

"La phase de Construction est l'étape cruciale où les idées et les plans se transforment en réalité tangible."



Écriture du Code

Rédaction du code source



Gestion de Version

Git, branches et commits



Automatisation

Build et tests automatisés



Artefact

Code empaqueté et prêt

Concepts Fondamentaux de Git



Repository (Dépôt)

L'endroit où le code source et son historique sont stockés.



Commit

Un instantané des modifications apportées au code, accompagné d'un message descriptif.



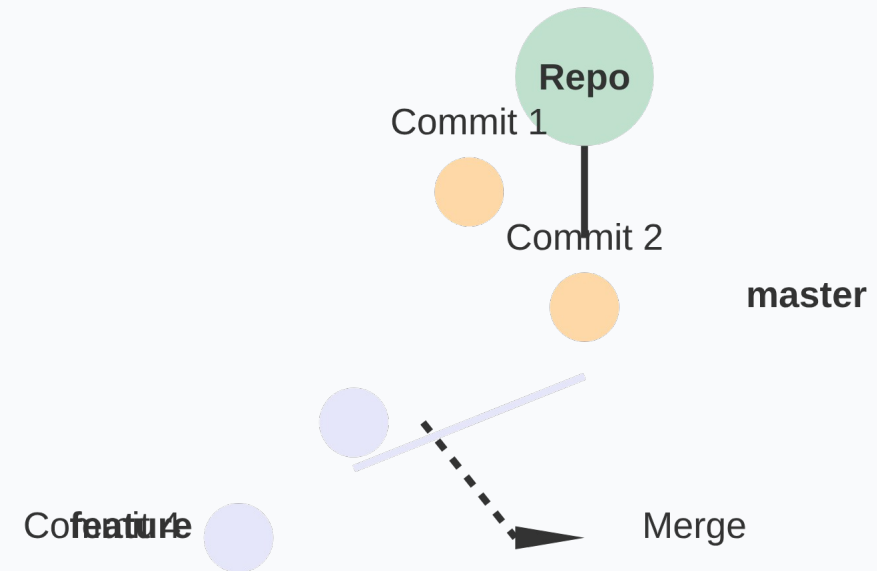
Branch (Branche)

Une ligne de développement indépendante qui permet de travailler sur de nouvelles fonctionnalités ou corrections sans affecter la ligne principale.



Merge (Fusion)

L'action de combiner les modifications d'une branche dans une autre.





GitFlow

Stratégie structurée avec multiples branches spécialisées



master Code de production stable



develop Intégration des nouvelles fonctionnalités



feature/* Développement de nouvelles fonctionnalités



release/* Préparation des versions



hotfix/* Corrections urgentes en production



Avantages

Structure claire, rôles de branches définis, gestion des versions et hotfixes séparés



GitHub Flow

Stratégie plus simple et orientée déploiement continu



main (ou master) Seule branche principale, toujours déployable



feature/* Toutes les nouvelles fonctionnalités et corrections



Pull Requests Intégration via revue et tests



Avantages

Simple, plus léger, axé sur des cycles de livraison rapides, intégration via Pull Requests

L'écriture de code de qualité est cruciale pour garantir la maintenabilité et la robustesse du logiciel. Trois pratiques fondamentales doivent être adoptées:



Code Propre

- ✓ Lisible et compréhensible
- ✓ Facile à maintenir
- ✓ Nommage des variables explicite
- ✓ Structure adéquate

💡 Code lisible = moins de bugs



Conventions de Style

- ✓ Consistance à travers l'équipe
- ✓ Outils de linting
- ✓ Formatage standard
- ✓ Règles claires

💡 améliorer la collaboration



Tests Unitaires

- ✓ Écrits en parallèle du code
- ✓ Valident le comportement
- ✓ Détectent les régressions
- ✓ Facilitent la refonte

💡 confiance pour refactorer

i **Avantage:** Lorsqu'ils sont adoptés ensemble, ces pratiques créent un code base solide, maintenable et évolutif.

L'automatisation du build est un pilier du Développement Continu, transformant automatiquement le code source en artefact déployable.



💡 **Note:** Cette automatisation est indispensable à l'Intégration Continue, car chaque modification du code doit déclencher un build et des tests automatiques.

Les outils de build automatisent le processus de transformation du code source en artefacts exécutables ou déployables, en incluant la compilation, l'exécution des tests et la création de packages.



Java



Maven

Outil de gestion de projet et de build basé sur le modèle d'objet de projet (POM). Gère les dépendances et les configurations du projet.



Gradle

Système de build flexible et performant, utilisant un DSL basé sur Groovy ou Kotlin. Idéal pour les projets Java et Android avec des configurations complexes.



JavaScript



npm

Gestionnaire de paquets Node.js standard. Utilisé pour installer, gérer et exécuter des scripts pour les projets JavaScript et Node.js.



yarn

Alternative à npm, offrant une expérience similaire pour la gestion des paquets et des scripts de build des projets JavaScript.



.NET



MSBuild

Système de build de Microsoft pour les applications .NET. Utilisé pour compiler et construire des applications .NET en automatisant les tâches de build.



MSBuild est intégré à Visual Studio et peut être utilisé via la ligne de commande pour automatiser le processus de build.



Chaque outil de build est spécifique à son écosystème et s'intègre bien avec les pratiques de Développement Continu pour automatiser le processus de build et faciliter l'Intégration Continue.



Visibilité accrue sur le projet

Grâce à une planification transparente et des cycles courts, toutes les parties prenantes ont une meilleure compréhension de l'avancement et des défis.



Meilleure qualité de code dès le départ

L'accent mis sur les bonnes pratiques de codage, les revues de code et les tests unitaires précoces garantit un code plus robuste et moins sujet aux erreurs.



Réduction du "time-to-market"

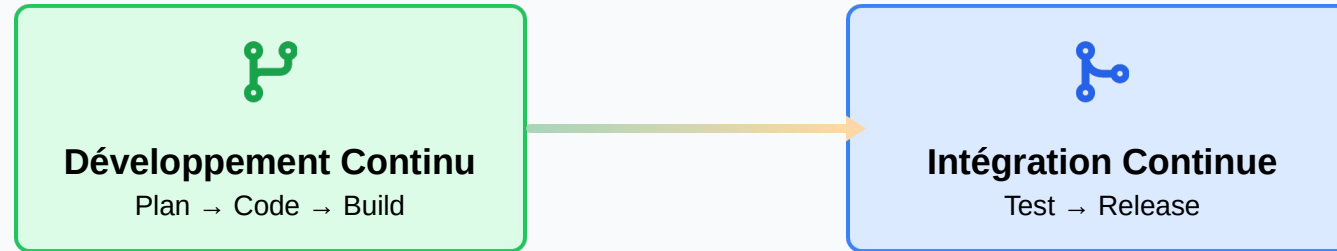
En découpant le travail en petites itérations et en automatisant les processus, les fonctionnalités peuvent être livrées plus rapidement aux utilisateurs.



Équipe plus alignée et agile

La collaboration constante, les boucles de feedback rapides et la flexibilité inhérente au processus favorisent une équipe plus cohérente et réactive aux changements.





Passerelle Naturelle

La phase de "Build" du Développement Continu est la passerelle naturelle et indispensable vers l'Intégration Continue (CI).



Déclenchement Automatique

Chaque modification de code (chaque "commit") déclenche automatiquement un processus de build et l'exécution des tests.



Code Fonctionnel

Le processus garantit que le code reste toujours dans un état fonctionnel et intégrable.



Boucle Continue

Le Build se transforme en CI, créant une boucle continue de développement, d'intégration et de livraison.

"Le Développement Continu sans l'Intégration Continue serait comme un codeur sans compilateur - la théorie sans la pratique."



Questions

Merci de votre attention. Avez-vous des questions sur le Développement Continu ou ses phases Plan et Build ?

Prochaines Étapes

Suite à cette présentation sur le Développement Continu, vous pouvez explorer les autres phases du cycle DevOps :



Intégration Continue

Automatiser l'intégration du code et des tests



Release

Stratégies de livraison et de versioning



Déploiement

Automatisation et orchestration des déploiements



Monitoring

Surveillance et feedback en production



À retenir: Le Développement Continu est la base du cycle DevOps, initialisant l'ensemble du processus.