جامعة الحسن الثاني بالدار البيضاء
UNIVERSITÉ HASSAN II DE CASABLANCA

FSBM
FACULTE DES SCIENCES BEN M'SICK
UNIVERSITE HASSAN II DE CASABLANCA

MODULE :

Le Cloud Computing & DevOps

Pr. F. Benabbou
Master DSBD
Faculté des Sciences Ben M'Sik Casablanca

TABLE OF CONTENTS

01 CLOUD COMPUTING

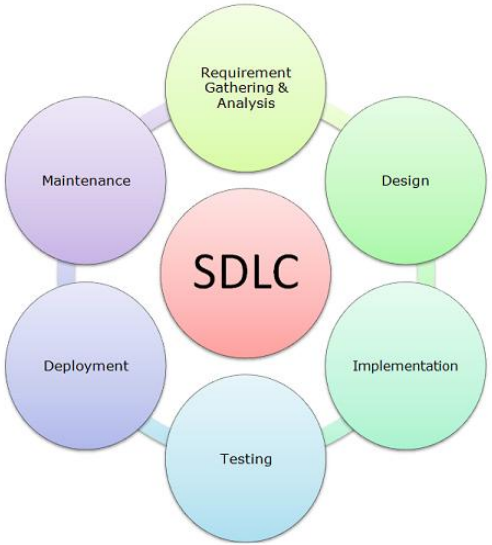
- Introduction générale
- La Virtualisation
- Les concepts de base du Cloud Computing
- Technologies émergentes du CC : Edge, Fog, …
- Étude de cas et projet pratique

02 DevOps & Cloud

- La philosophie DeVops
- Version control systems (git)
- Integration Continuous
- Tests automatisés
- Deploiment Continu CD
- Infrastructure en tant que Code (IaC)
- Surveillance et Journalisation
- Étude de cas et projet pratique

Traditional software development model (TSDL)

- Avant DevOps, le développement de logiciels suivait une approche plus traditionnelle, souvent désignée sous le nom de **Waterfall** or **Siloed Model**.
- Déroulent générale
  - **Analyse** des besoins
  - **Conception** : Production d'une spécification de l'artefact logiciel
  - **Implémentation**: Développement du logiciel
  - **Test** : Examen de la qualité du produit ou du service, recherche et correction des défauts
  - **Déploiement** en production.
  - **Maintenance**



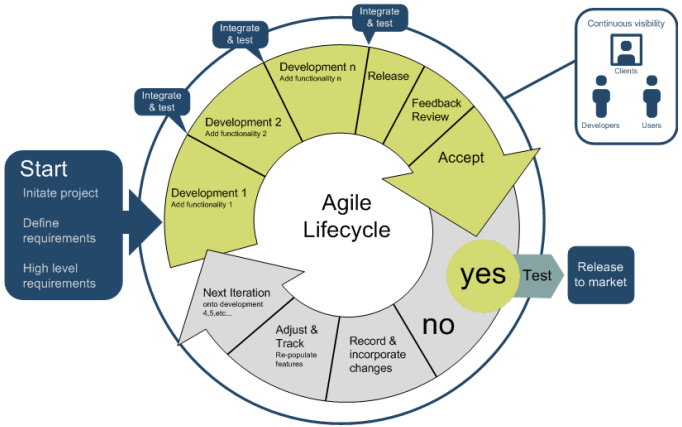
Waterfall model of software development Life Cycle

Limitation de TSDL

- **Silos de développement et d'exploitation :**
  - les équipes de développement et d'exploitation étaient séparées, avec une interaction limitée.
  - Les développeurs écrivaient le code et les services opérationnels le déployaient et le maintenaient
  - Problèmes de communication, des retards et des frictions entre les équipes.
- **Cycles de développement longs :**
  - Le développement suivait un modèle en cascade, dans lequel les projets progressaient linéairement à travers des phases telles que la collecte des besoins, la conception, le développement, les tests et le déploiement.
  - Chaque phase devait être achevée avant de passer à la suivante, ce qui entraînait de longs cycles entre les versions.
- **Automatisation limitée :**
  - Les tests étaient généralement effectués manuellement et le déploiement était souvent pris en charge par les opérations, avec peu de participation des développeurs.
  - Cela augmentait le risque d'erreurs lors du déploiement et rendait le processus plus lent.
- **Retour d'information tardif :**
  - Les feedback interviennent tardivement dans le processus, parfois seulement après la sortie du produit.
  - Il était donc difficile de répondre rapidement aux besoins des clients ou de procéder à des ajustements rapides.

Agile software development Model (ASDL)

- Les méthodes de développement agiles désignent un ensemble de principes et de pratiques visant à améliorer le processus de développement de logiciel en se concentrant sur la **flexibilité**, la **collaboration**, et **l'adaptabilité** aux changements.
- Le développement de logiciels agiles repose sur une approche **itérative** et incrémentale où les projets sont divisés en cycles courts appelés sprints (généralement de 1 à 4 semaines).
- Certains cycles sont répétés plusieurs fois et aboutissent à un **produit** intermédiaire fonctionnel.
- Le but est de fournir une application dans un délai court



Caractéristiques des méthodes agiles

- **Livraison rapide et continue** : Les produits ou fonctionnalités sont livrés régulièrement et de manière incrémentale.
- **Flexibilité face aux changements** : Les équipes peuvent ajuster le développement en fonction des retours des utilisateurs ou des besoins changeants du marché.
- **Collaboration constante** : Une forte interaction entre les développeurs, les utilisateurs et les parties prenantes tout au long du projet.
- **Amélioration continue** : Les processus sont régulièrement réévalués et optimisés, avec une attention particulière à l'amélioration des performances et de la qualité.
- **Communication ouverte** : Les équipes sont encouragées à échanger de manière transparente et proactive.

Manifeste des méthodes agiles

- Les méthodes agiles sont souvent associées au Manifeste Agile, rédigé en 2001, qui énonce **quatre** valeurs fondamentales :
  - Les individus et leurs interactions plutôt que les processus et les outils (Des équipes engagées et passionnées sauront créer un produit de valeur)
  - Des logiciels fonctionnels plutôt que de la documentation exhaustive (le développeur devrait se concentrer sur la production de fonctionnalités et non sur la rédaction de documents).
  - La collaboration avec les clients plutôt que la négociation de contrats (se concentrer sur les besoins des utilisateurs que sur des accords contractuels rigides).
  - L'adaptation au changement plutôt que le suivi d'un plan rigide (accepter des demandes changements même tard dans le développement du produit).

Comparaison ASDLC vs TSDLC

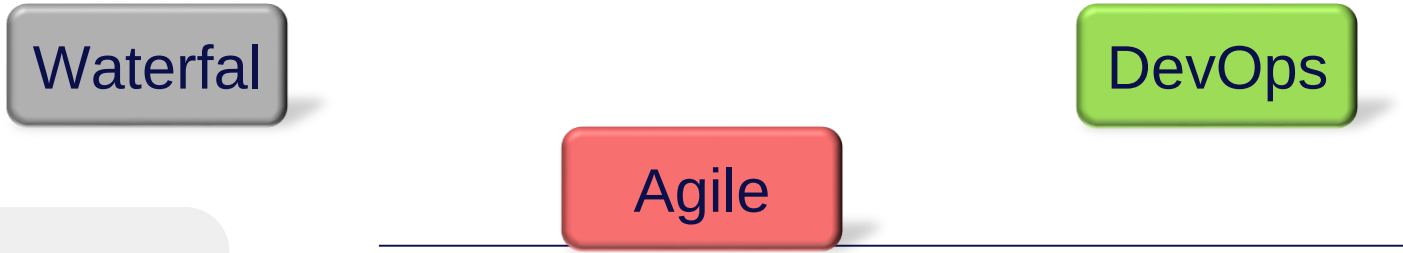
Critère	Agile SDLC	Traditional SDLC (Waterfall)
Méthodologie	Itérative et incrémentale, avec des sprints courts (1-4 semaines)	Linéaire et séquentielle, chaque phase dépend de la précédente
Flexibilité et Adaptation	Très flexible, les exigences peuvent évoluer au fil du projet	Moins flexible, les exigences sont définies en début et ne changent pas facilement
Planification et Documentation	Documentation légère, planification continue	Documentation détaillée et planification complète dès le début
Livraison et Feedback	Livraisons fréquentes et incrémentales, feedback continu	Livraison finale à la fin du projet, feedback tardif
Gestion des Risques	Risques identifiés tôt et ajustés rapidement	Risques identifiés tardivement, souvent après les tests
Rôle des Parties Prenantes	Participation active et continue des parties prenantes	Implication au début pour les exigences et à la fin pour la validation
Équipe de Développement	Autonomie des équipes, collaboration transversale	Équipes spécialisées par phase (analystes, développeurs, testeurs)
Adaptation à la Taille du Projet	Adapté aux projets de taille petite à moyenne avec exigences changeantes	Adapté aux projets de grande envergure avec exigences stables
Coûts et Temps	Peut avoir des coûts plus élevés si les priorités changent souvent, mais peut livrer plus rapidement	Moins coûteux au début, mais peut devenir plus cher si des erreurs sont découvertes tardivement

Exemple

- Parmi les frameworks agiles les plus populaires figurent Scrum, Kanban, Extreme Programming (XP) et Lean Software Development.

Histoire de DevOps

- Le mouvement DevOps a vu le jour entre 2007 et 2008 et a été inventé par **Patrick Debois**, qui a organisé la première conférence DevOpsDays en Belgique en 2009
- Les communautés des opérations informatiques et du développement de logiciels ont exprimé leurs inquiétudes quant au modèle traditionnel de développement de logiciels.
- L'objectif principal était de combler le fossé entre ces deux équipes traditionnellement cloisonnées, d'améliorer la collaboration et d'accélérer la livraison des logiciels sans compromettre la qualité.



Facteurs clés qui ont influencé sur le devOps

Facteurs clés

01

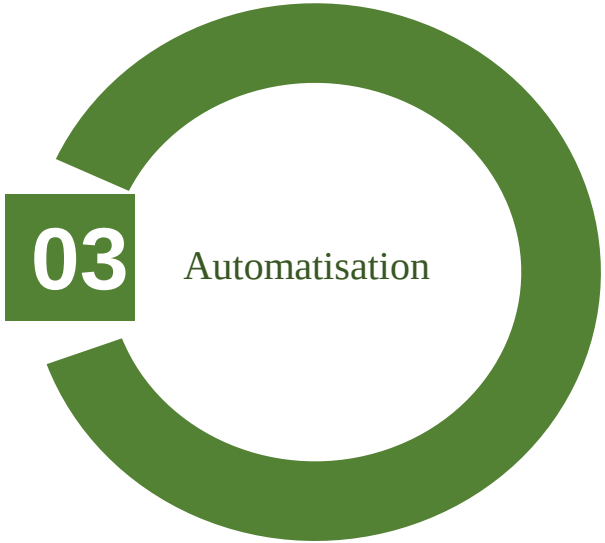
- Les **méthodologies agiles** favorisent la collaboration, le développement itératif et le retour d'information des clients.
- Le Dev. Agile a laissé un fossé entre les équipes de développement et les équipes opérationnelles.
- DevOps a étendu les principes Agile aux équipes opérationnelles.

Facteurs clés

02

- DevOps a également emprunté des concepts à la fabrication sans gaspillage, en se concentrant sur l'efficacité, la réduction du gaspillage et l'amélioration continue.

Facteurs clés

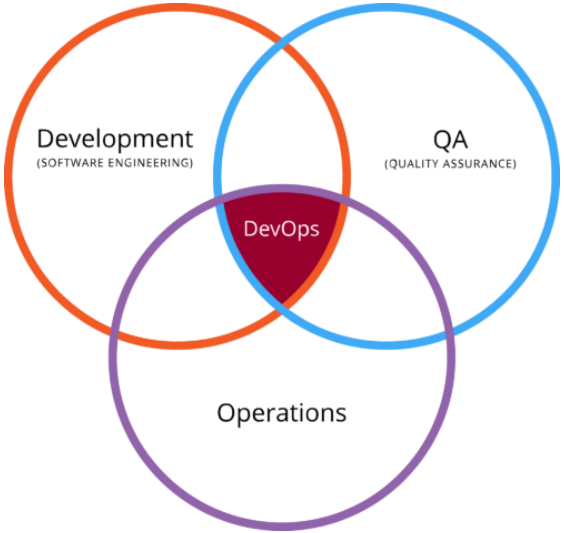


03

- L'automatisation permet à DevOps de réaliser son objectif principal : accélérer la livraison de logiciels tout en maintenant une haute qualité et une collaboration étroite entre les équipes de développement (Dev) et d'exploitation (Ops).

C’est quoi DevOps?

- Le DevOps est une méthode de développement de logiciels qui combine le développement de logiciels (Dev) et les opérations informatiques (Ops).
- DevOps intègre toutes les fonctions de développement de logiciels, du développement à l'exploitation, au sein d'un même cycle.
- Il nécessite un niveau de coordination plus élevé entre les différentes parties prenantes du processus de développement de logiciels (à savoir le développement, l'assurance qualité et les opérations).



le DevOps Adopte les principes du Lean

Pour économiser des coûts, des efforts et du temps, les développeurs devraient suivre les principes directeurs suivants : YAGNI, JIT

« Commencer toujours par ajouter quelques méthodes obligatoires dans la classe ». Cette approche présente plusieurs avantages:

Minimisation du gaspillage : YAGNI réduit le gaspillage des ressources sur les fonctionnalités inutilisées en se concentrant uniquement sur les fonctionnalités essentielles.

Accélère la livraison : Elle permet des cycles de publication plus rapides, s'alignant parfaitement avec l'objectif DevOps de déploiement rapide.

Simplifie la maintenance : Une base de code allégée est plus facile à gérer et à dépanner, ce qui améliore la fiabilité globale du système.

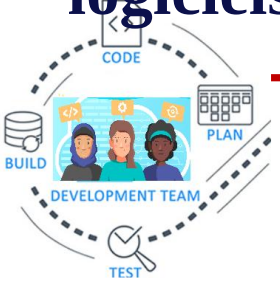
Le JIT (Just-In-Time) est dérivé de la production allégée et se concentre sur la production en temps voulu et l'alignement de l'utilisation des ressources sur la demande. Les avantages du JIT dans le cadre de DevOps sont les suivants :

Efficacité accrue : La méthodologie JIT réduit le temps entre le développement et le déploiement.

Optimisation des ressources : Elle optimise les ressources, en évitant la surutilisation ou la sous-utilisation.

Développement adaptatif : La méthode JIT permet aux équipes de s'adapter rapidement à l'évolution des besoins, un principe fondamental des méthodologies agiles.

Parties prenantes du processus de développement de logiciels



Le **développeur** de logiciels est le créateur du produit, il est responsable de l'écriture du code et, dans un contexte DevOps, le développeur effectue également les tests unitaires et le déploiement, ainsi que la surveillance continue

Le **Ops** (Opérations) est responsable de *l'exploitation* à la fois des **applications** et de **l'infrastructure**. Cela inclut le déploiement, la surveillance et la maintenance des systèmes pour s'assurer qu'ils fonctionnent de manière efficace, tout en gérant également les besoins en matière de sécurité et de scalabilité.

Le QA (Quality Assurance) assure la qualité des logiciels en testant les défauts, en vérifiant les fonctionnalités et en garantissant la conformité aux exigences. Ils automatisent les tests, favorisent la détection précoce des bugs et collaborent avec Dev et Ops pour garantir des versions fiables et de haute qualité..

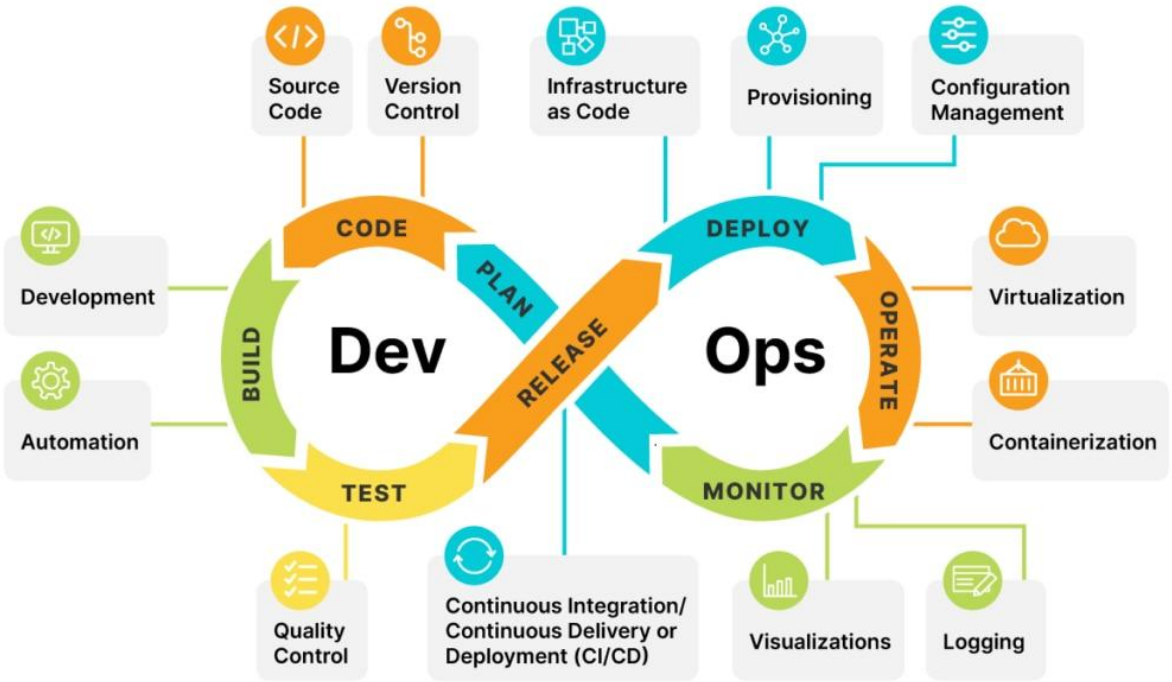
Principes clés de DevOps

- **Collaboration**
  - Les équipes dev. Et ops. (développement et d'exploitation) communiquent, partagent les informations et collabore tout au long du cycle de développement et de déploiement..
- **Automatisation**
  - L'une des pratiques essentielles de DevOps consiste à automatiser autant que possible le cycle de développement des logiciels.
  - Les développeurs se concentre plus sur le code pour développer de nouvelles fonctionnalités.
  - Elle permet de réduire les erreurs humaines et d'augmenter la productivité de l'équipe
  - L'automatisation est un élément clé d'un pipeline Continous Integration et Continous deployment (CI/CD) .
- **Amélioration Continue**
  - Ce principe met l'accent sur l'expérimentation, la réduction du gaspillage et l'optimisation de la vitesse, des coûts et de l'efficacité dans le cycle de développement.
  - Il est lié à la livraison continue, il permet aux équipes DevOps de diffuser régulièrement des mises à jour, d'améliorer les systèmes logiciels, d'accroître l'efficacité du développement et d'apporter une plus grande valeur ajoutée aux clients.

Principes clés de DevOps

- **Feedback continu**
  - Les équipes DevOps de se concentrer sur les besoins réels des utilisateurs pour développer des produits adaptés à leurs besoins.
  - **Réactivité aux commentaires** : Les pratiques DevOps permettent de collecter rapidement les retours des utilisateurs et d'y répondre via un suivi en temps réel et un déploiement rapide.
  - **Visibilité immédiate** : Les équipes ont une visibilité en temps réel sur la façon dont les utilisateurs interagissent avec le système, ce qui permet de développer de nouvelles améliorations basées sur ces informations.

Cycle DevOps



Cycle DevOps

- **Plan** : Organiser le travail à effectuer, le classer par ordre de priorité et en suivre l'exécution.
- **Code** : Écrire, concevoir, développer et gérer en toute sécurité le code et les données du projet avec les équipe.
- **Build** : consiste à :
  - *Test***:** S'assurer que votre code fonctionne correctement et respecte les normes de qualité - idéalement à l'aide de tests automatisés.
  - *Package***:** Regrouper les applications et les dépendances, gérer les conteneurs et les artefacts de construction.
  - *Sécuriser* : Rechercher les vulnérabilités à l'aide de tests statiques, dynamiques, ou *fuzz* (aléatoire) et d'analyse des dépendances.
- **Release**: La gestion des versions de logiciel et la planification des déploiements. Chaque version du logiciel doit être clairement identifiée.
- **Deploy**: Le déploiement fait référence à la mise en production des applications dans les environnements de production, ce qui implique de gérer et configurer l'infrastructure nécessaire pour soutenir les applications.
- **Monitor**: consiste à:
  - *Surveillance* de la performance des applications (temps de réponse, charge du serveur, disponibilité, etc.). et de l'infrastructure (serveurs, bases de données, réseaux, etc.)
  - *Détecter* les problèmes potentiels (pannes, erreurs, régressions de performance, consommation anormale de ressources, etc.).
  - *Feedback* utilisateur et des logs peuvent être collectés pour évaluer si la version déployée est stable et si des corrections ou des améliorations sont nécessaires.

Fonctionnalités DevOps

- Les composantes Développement Continu, Intégration Continue, Tests Continus, Déploiement Continu, Surveillance Continue, Feedback Continu, et Opérations Continues; vise à accélérer les processus tout en maintenant la qualité du logiciel, grâce à l'automatisation et à la collaboration étroite entre les équipes de développement et d'opérations.