Chapitre 1
Introduction à Devops

Enseignante: Dr-Ing. Amina JARRAYA
Email : amina.jarraya@ensi-uma.tn
Niveau: II3- GL

Plan

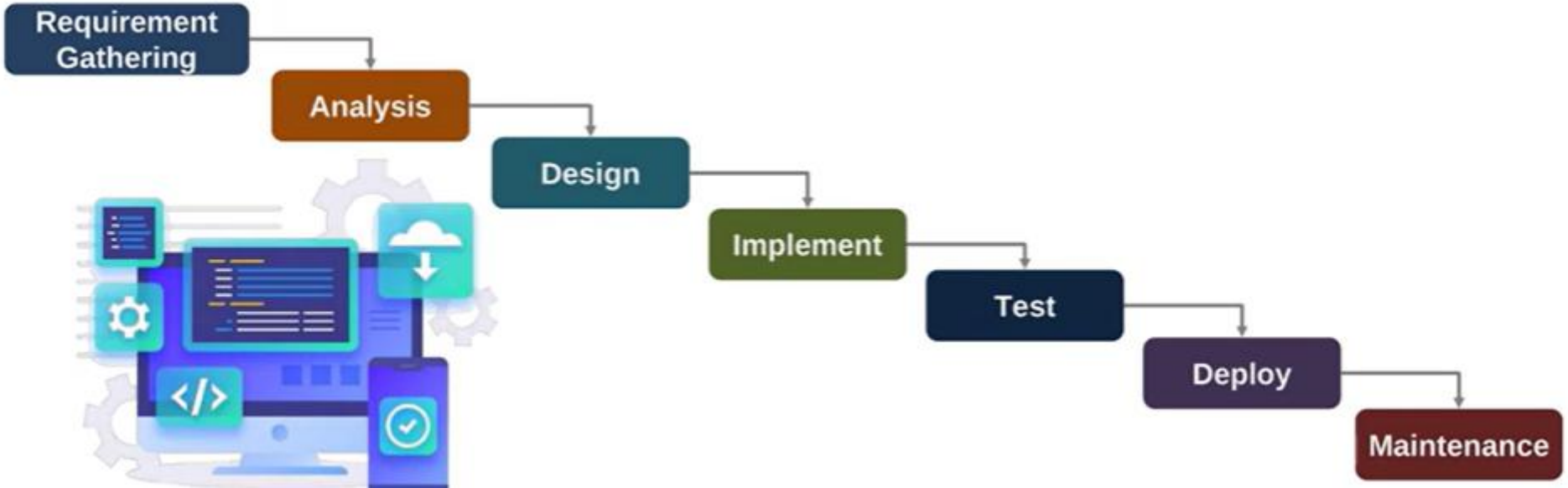
1. Le modèle en cascade et ses limites
2. Le modèle agile et ses limites
3. Présentation de DevOps ?
4. Cycle de vie de DevOps
5. Les phases de DevOps
    - Le développement continu - Contrôle de version
    - L'intégration continue (CI)
    - Les tests continus
    - Le déploiement continu (CD)
    - La conteneurisation
    - Les opérations continues - La gestion de configuration - IaC
    - Les feedbacks continus
6. DevOps pipeline

1. Le modèle en cascade et ses limites

Le modèle en cascade (waterfall model)

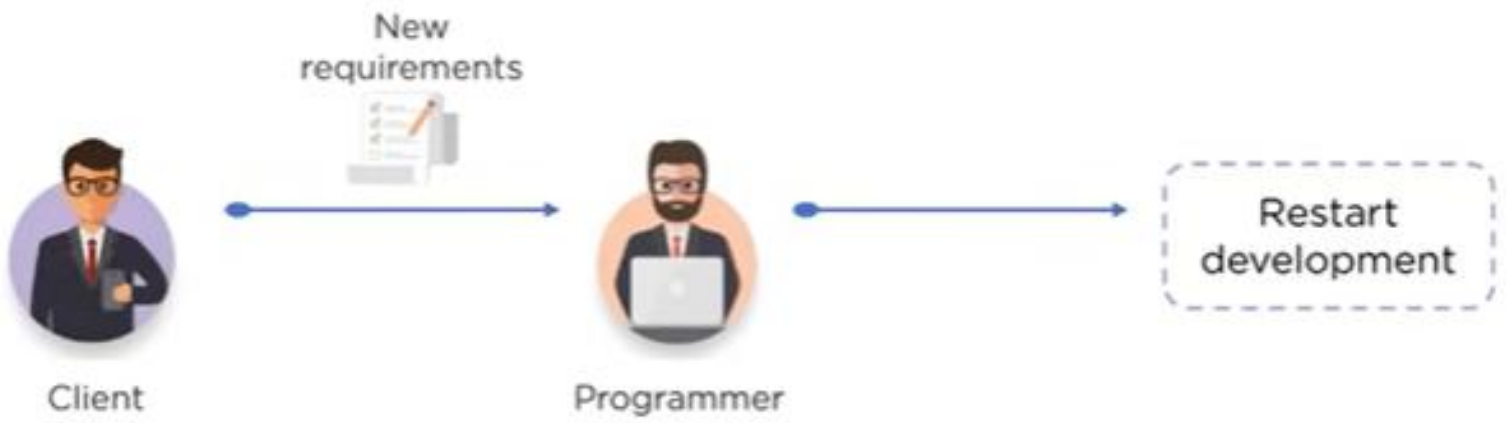
- Le modèle en cascade est une méthode de gestion de projet linéaire et séquentielle où le projet est divisé en phases distinctes qui s'enchaînent, la phase suivante ne débutant qu'après l'achèvement de la précédente.

- Ce modèle, qui fut l'une des premières méthodologies de développement, se caractérise par une planification détaillée et une documentation exhaustive dès le début, et est bien adapté aux projets dont les besoins sont stables et claires car il est difficile et coûteux de revenir en arrière une fois une phase terminée.



Limites du modèle en cascade (waterfall model)

- Un nouveau besoin du client nécessite de refaire tout le cycle de développement.



- Si le client est insatisfait du produit livré, tout le cycle doit être redémarré.

Limites du modèle en cascade (waterfall model)

- Les besoins du client ne sont pas clairs au départ.

- Il est couteux et une perte de temps d’appliquer des changements à la fin du projet

- Le produit doit être livré à temps avec moins de ressources

- L'effet tunnel est le manque de visibilité sur l'avancement du projet et l'attente de la livraison finale pour évaluer les résultats concrets

2. Le modèle agile et ses limites

La méthodologie agile

Agile est une plateforme divisant le travail en des sprints dynamiques

Agile divise le projet en des itérations de longueur équitable de 2 à 8 semaines

Chaque itération (sprint) délivre un produit livrable

Les défis de la méthodologie agile

- Une non-collaboration entre les équipes de développement (Dev) et les équipes d'opérations (Ops) dans un contexte Agile peut entraîner des inefficacités, des retards et des erreurs, sapant les bénéfices de l'approche Agile.

Limites de l’approche agile

- **Complexité à grande échelle:**

L'agilité, bien que souvent efficace pour de petites équipes, peut devenir complexe à gérer dans des organisations vastes ou des projets de grande envergure.

- **Difficultés de planification et de prévisibilité:**

L'approche agile, par sa nature itérative, peut rendre difficile la prévision des coûts, du temps et des ressources nécessaires dès le départ, surtout dans les projets longs et complexes.

- **Besoin de documentation:**

Bien que l'agilité privilégie les échanges et les livraisons fréquentes, certains projets nécessitent une documentation plus rigoureuse, notamment pour la maintenance, la conformité réglementaire, ou la transmission d'informations à long terme.

- **Difficulté à maintenir la collaboration:**

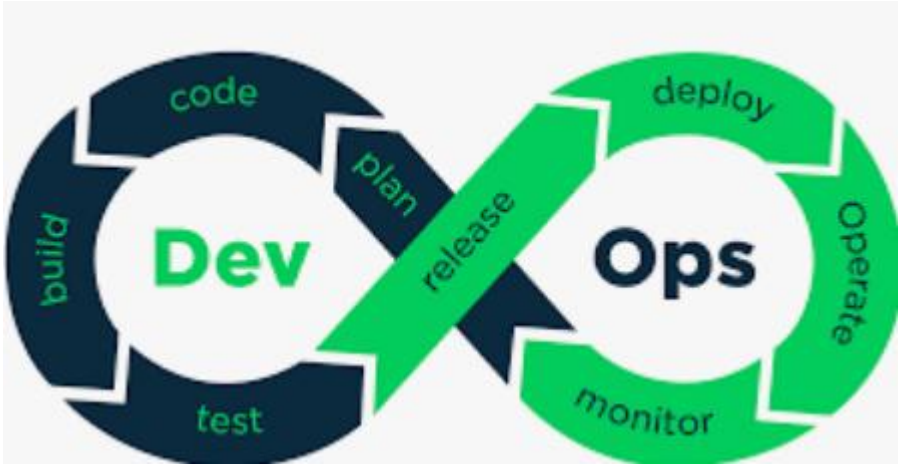
Le maintien d'un niveau élevé de collaboration peut être un défi, surtout dans les équipes distribuées ou lors de projets de grande envergure.

3. Présentation de DevOps

C’est quoi DevOps ?

- DevOps est une approche collaborative qui intègre le développement et les opérations pour rationaliser l'ensemble du cycle de vie du logiciel.

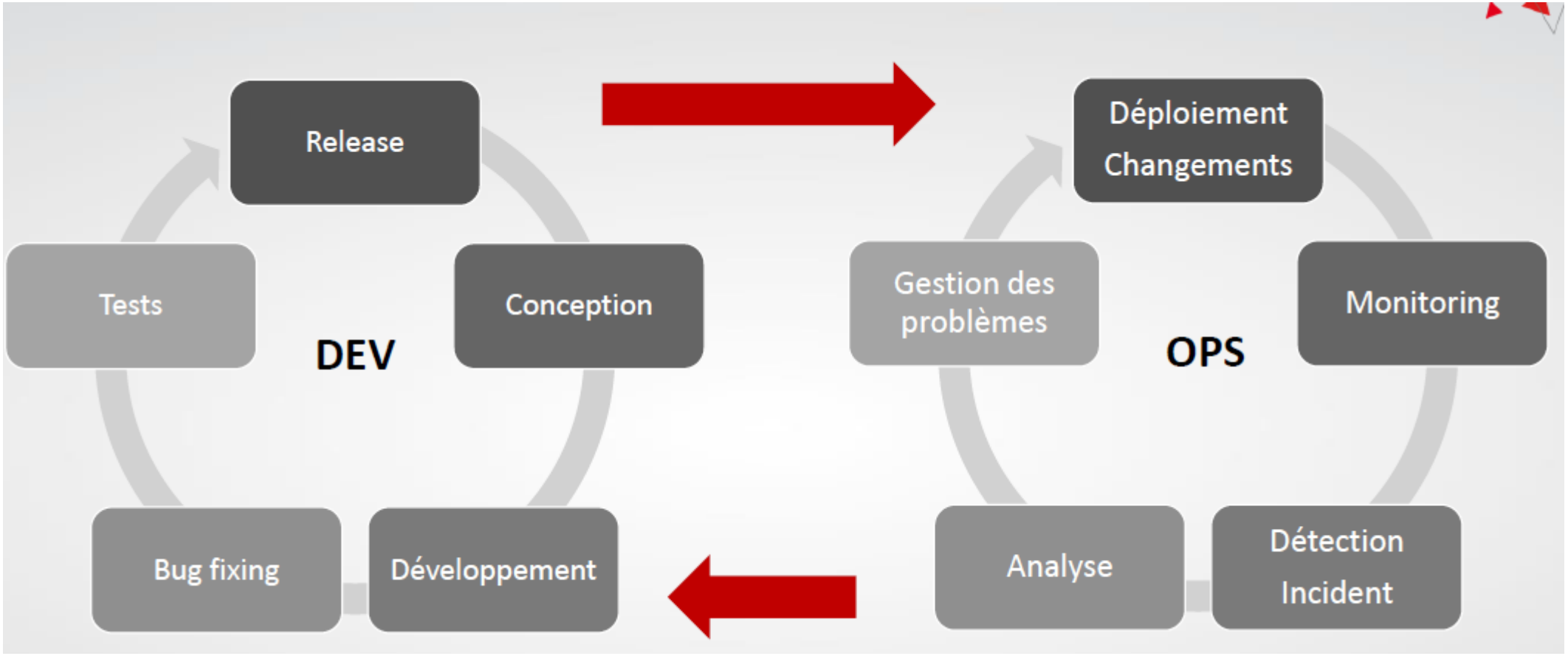
- DevOps favorise la collaboration et l'automatisation pour des déploiements plus rapides et une qualité améliorée.



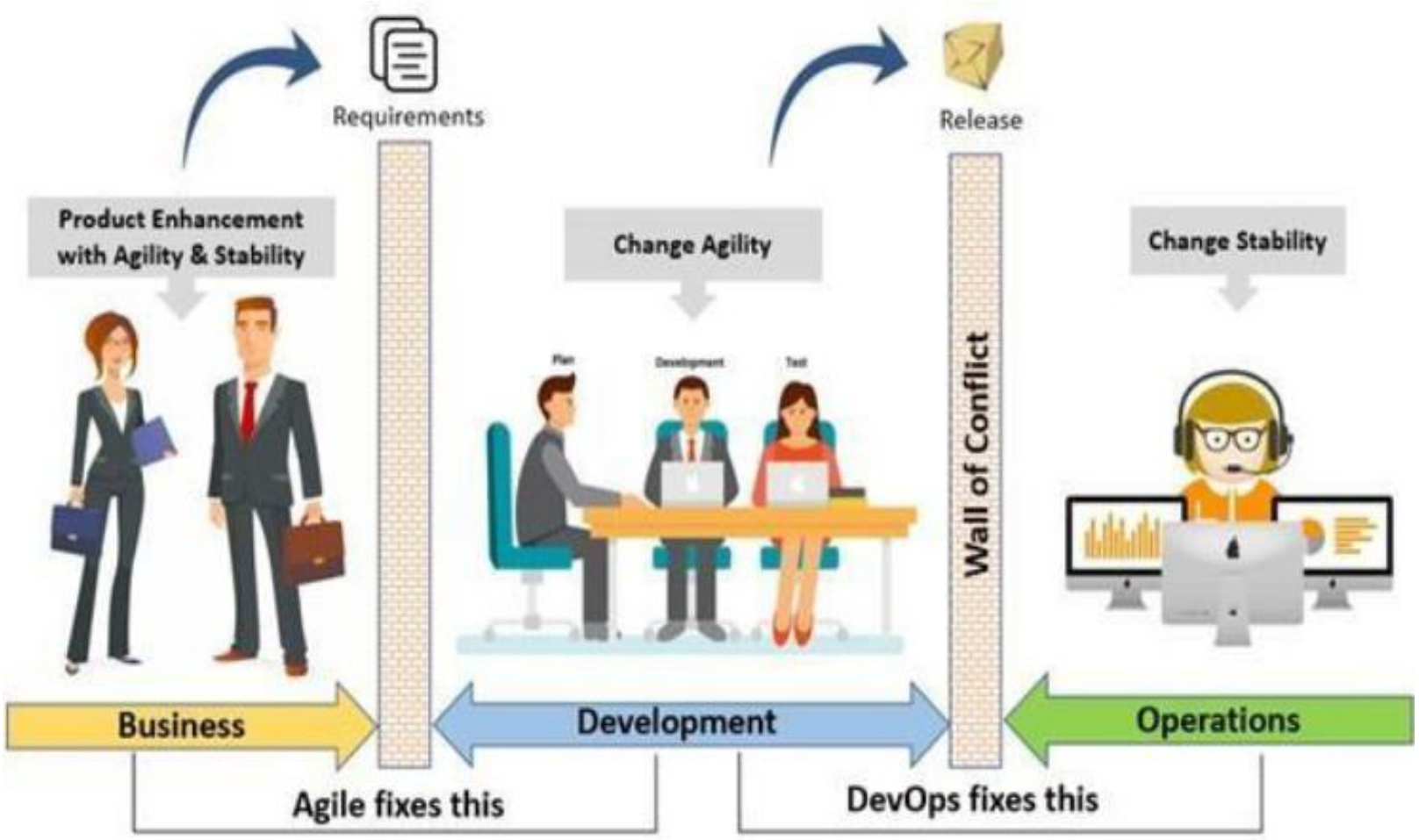
DevOps – vision différentes

Développement	Opérations
Planning et la date de livraison	Qualité de service et disponibilité
Couts de développement	Cout d'exploitation
Releases planning	Changements, Incidents
Dernières technologies	Technologies standards
Environnement de **développement**	Environnements de **production**
Fréquents et importants changements	Minimise le changement en production
Méthodes agiles	Organisation et processus structurés (ITIL)
Pensent à ce qui va faire marcher les choses	Pensent à tout ce qui ne va pas marcher

Avant DevOps – Cycles différents

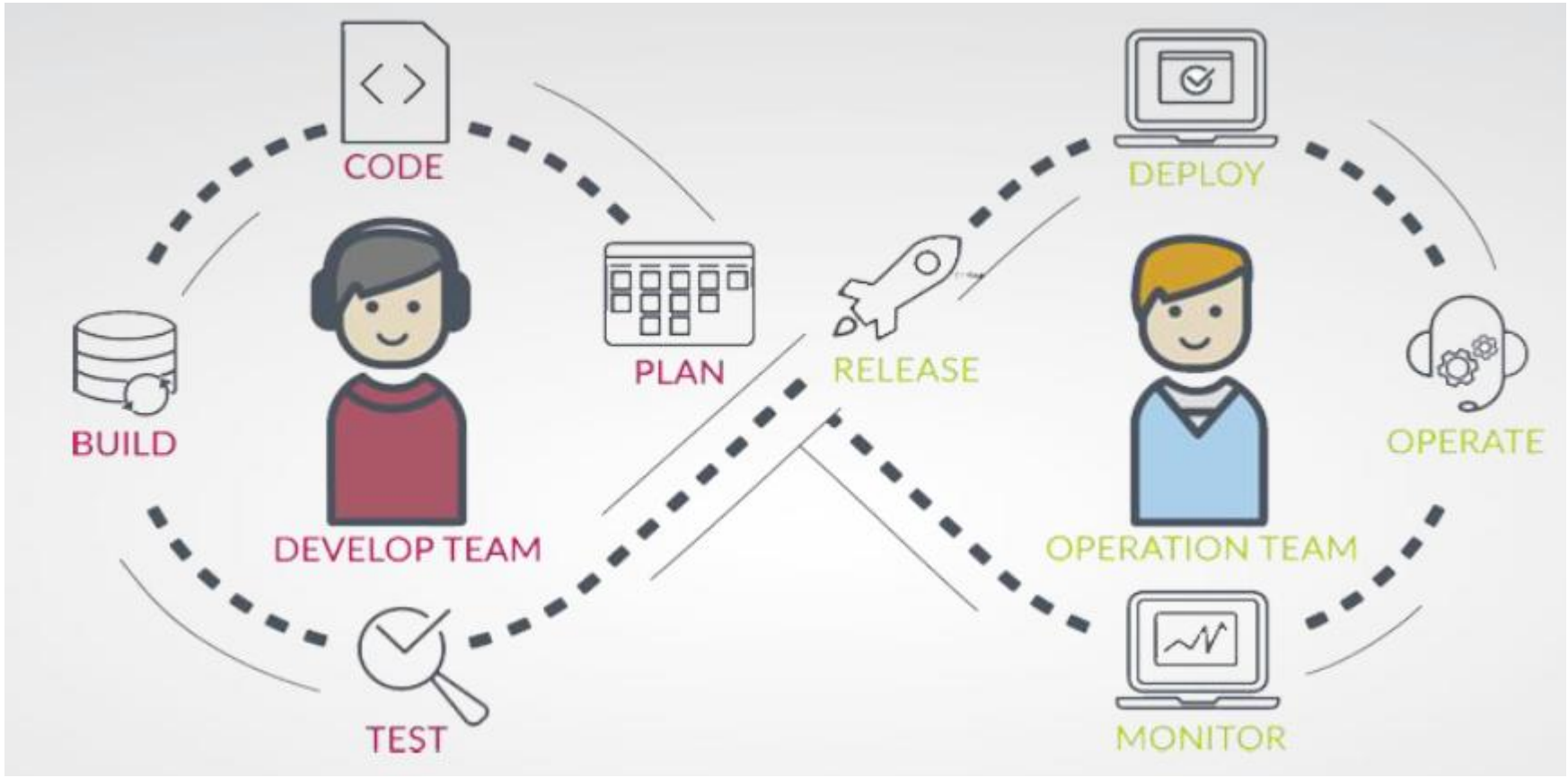


DevOps - intéractions



- L’agilité et les pratiques DevOps interviennent pour briser les frontières entre les différents collaborateurs. **C’est complémentaire**.

DevOps – cycle de vie



- Création d'un **pipeline** automatisé entre les deux équipes appelées **CI/CD** (Continuous Integration / Continious Deployment (ou Delivery)

4. Cycle de vie de DevOps

Cycle de vie de DevOps

- **Planification** : Cette étape est cruciale pour définir les objectifs, identifier les besoins, et établir une feuille de route pour la transformation DevOps.

Cycle de vie de DevOps

- **Codage** : Cette phase implique l'écriture du code par les développeurs.

Cycle de vie de DevOps

- **Construction** : Cette phase permet de détecter les erreurs au niveau du code.

Cycle de vie de DevOps

- **Tests** : sont intégrés dans le pipeline de développement de manière continue. Cela permet de détecter les erreurs dès leur apparition et de les corriger rapidement.

Cycle de vie de DevOps

- **Diffusion** : est une étape cruciale où l'on prépare un produit logiciel prêt à être déployé en production. Elle marque le moment où un ensemble de code a passé avec succès une série de tests et est jugé stable et fiable

Cycle de vie de DevOps

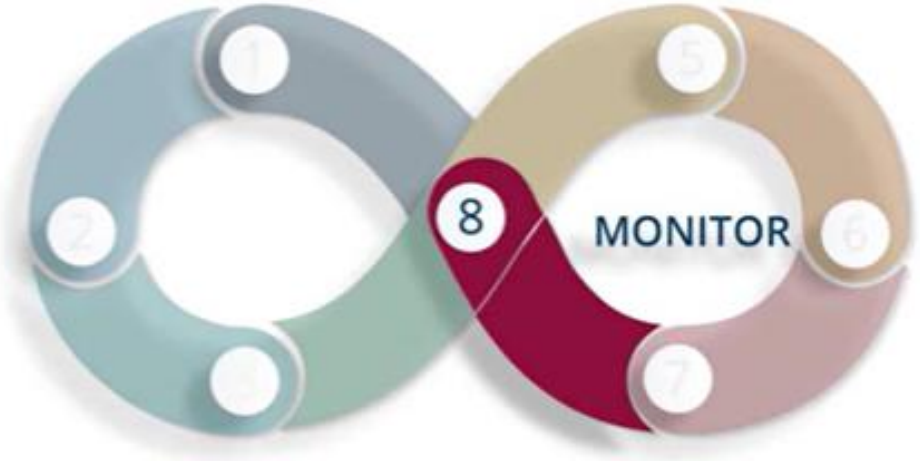
- **Déploiement** : consiste à rendre l'application développée disponible dans l'environnement de production, après avoir passé avec succès les étapes précédentes du pipeline.

Cycle de vie de DevOps

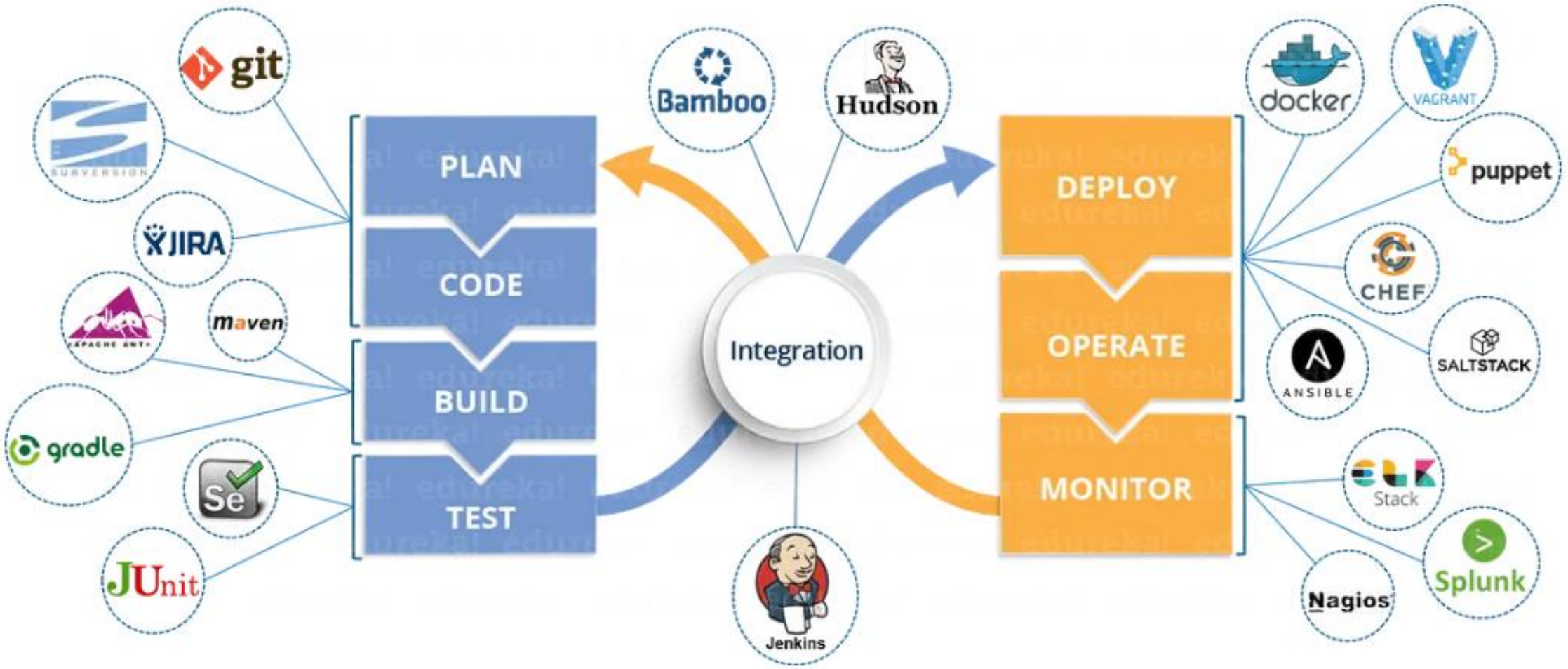
- **Opération** : est la phase où une application est maintenue en production, incluant la maintenance, le dépannage et la supervision pour garantir la fiabilité et la disponibilité. Cette phase vise à garantir la stabilité, la performance et la sécurité de l'application, ainsi qu'à collecter des retours d'expérience pour améliorer les versions futures.

Cycle de vie de DevOps

- **Surveillance** : c'est la pratique consistant à observer et à analyser en permanence les performances et la santé des systèmes, des applications et de l'infrastructure au sein d'un pipeline DevOps.

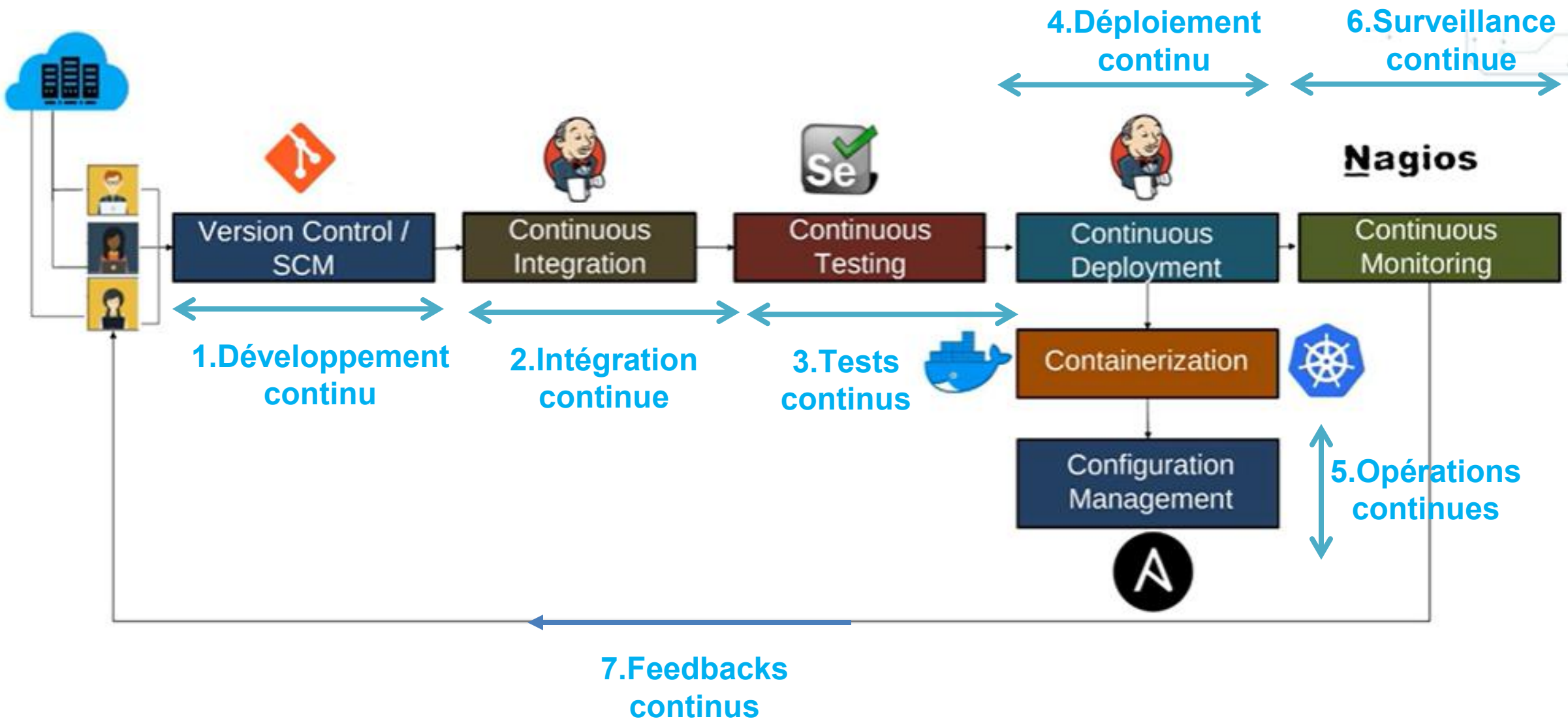


Outils de DevOps



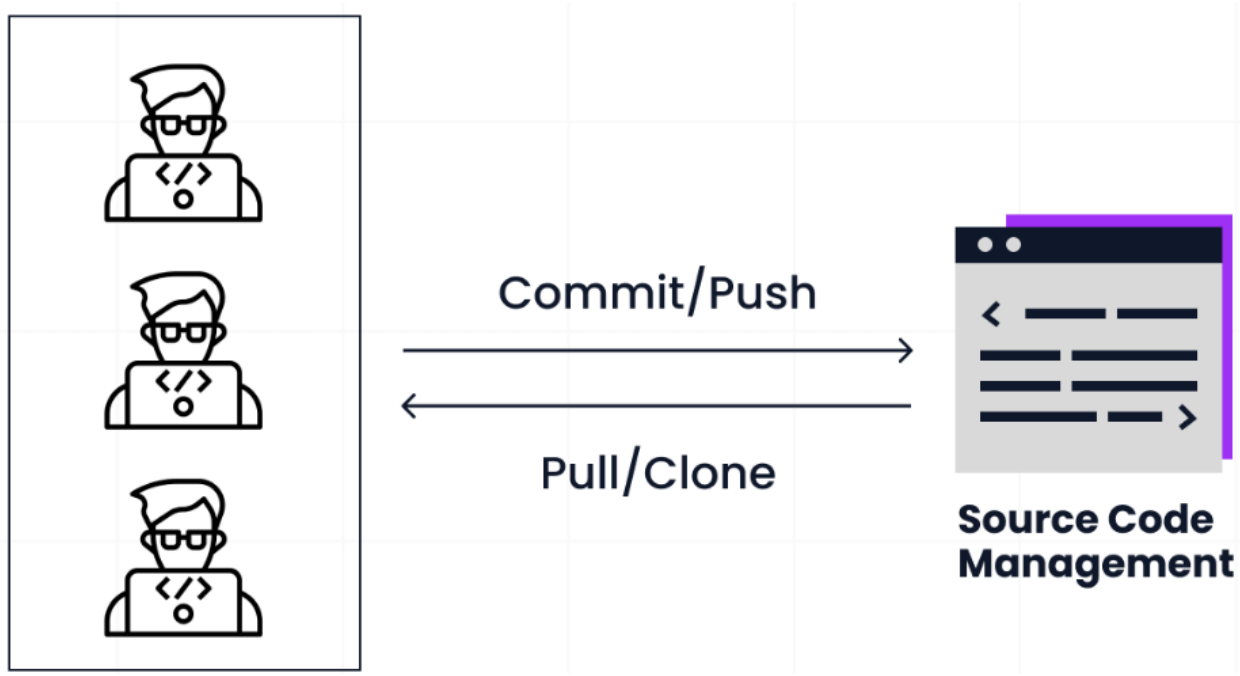
5. Les phases de DevOps

Les 7 phases de DevOps



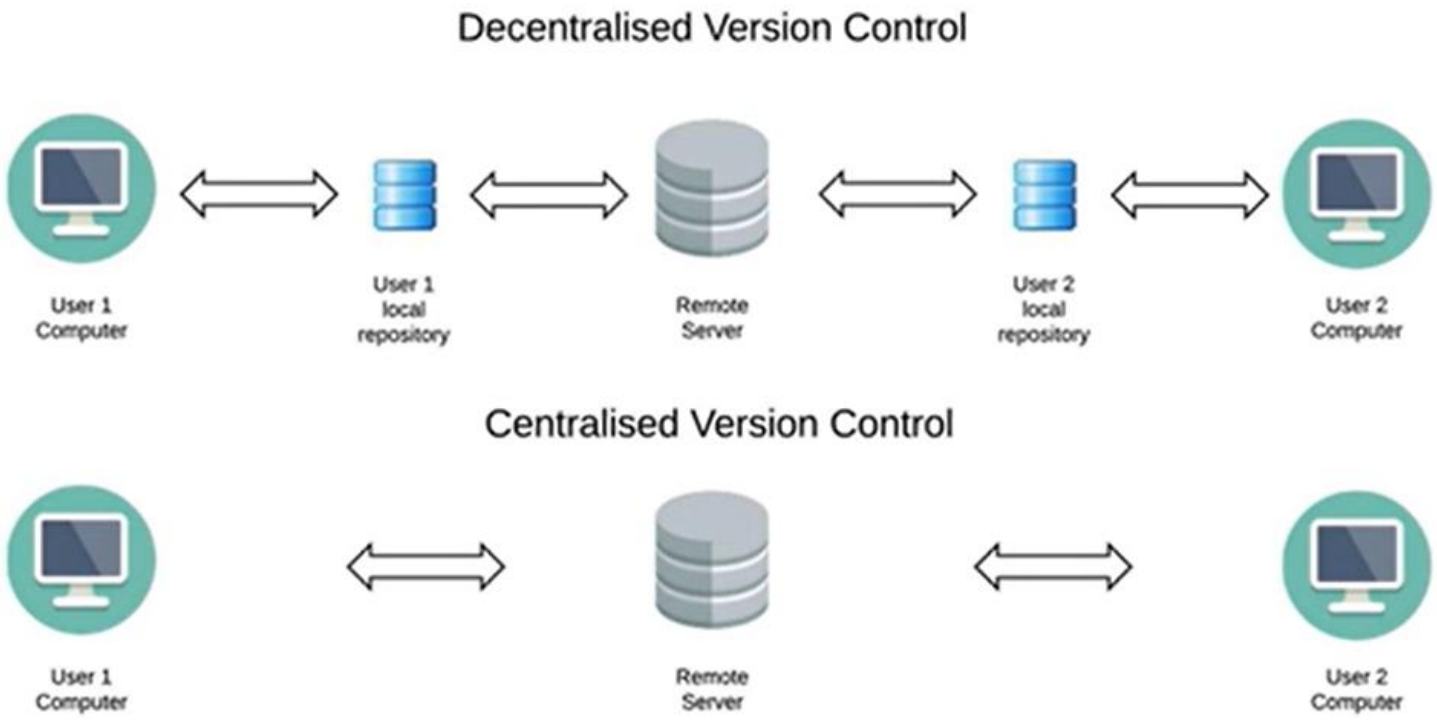
1. Le développement continu

- Le développement de logiciels commence par la planification et le codage.
- Après la phase de planification du cycle de vie DevOps, la création de code source et de ressources commence avec l'objectif de faire progresser la production.
- Quel que soit le langage de codage utilisé, la gestion de la base de code à l'aide d'outils de gestion du code source est une priorité.



1. Le développement continu - Contrôle de version/ Gestion de code source

- ✓ Suivre les changements au cours du temps
- ✓ Supporte différents types de fichiers
- ✓ Facilite la collaboration
- ✓ Restaurer les versions précédentes
- ✓ Le branchement et la fusion



1. Le développement continu - Contrôle de version/ Gestion de code source

- Les outils de contrôle de version

GIT

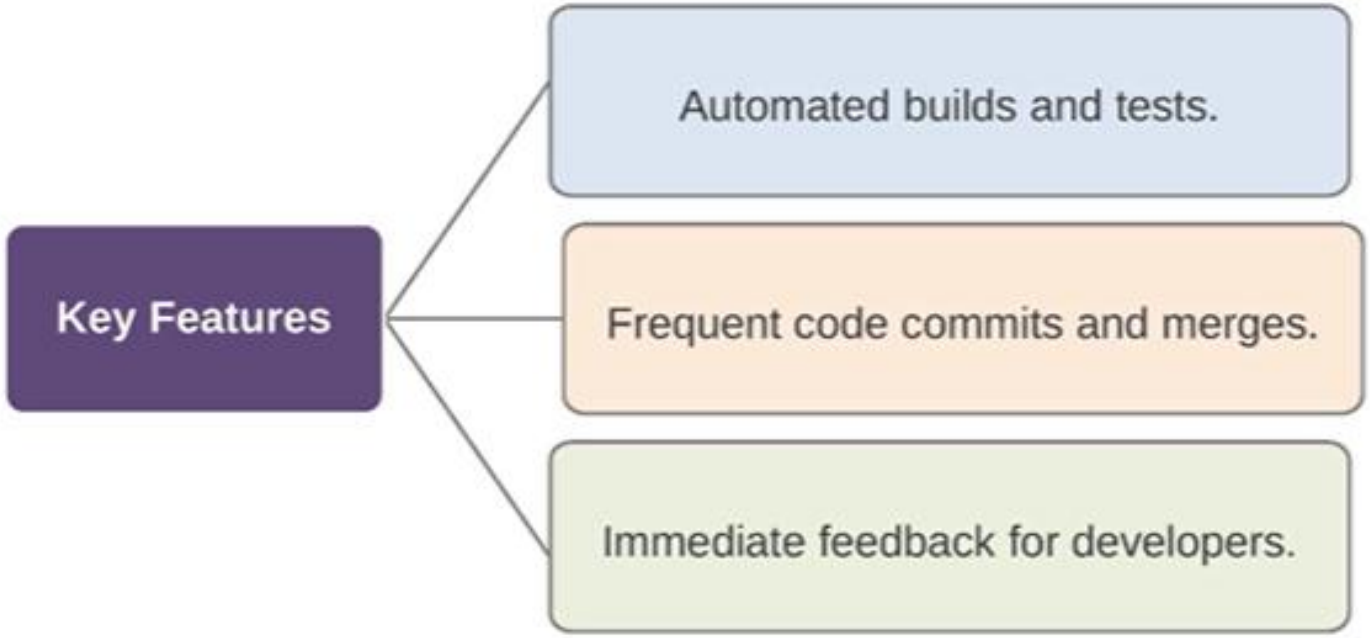
Subversion

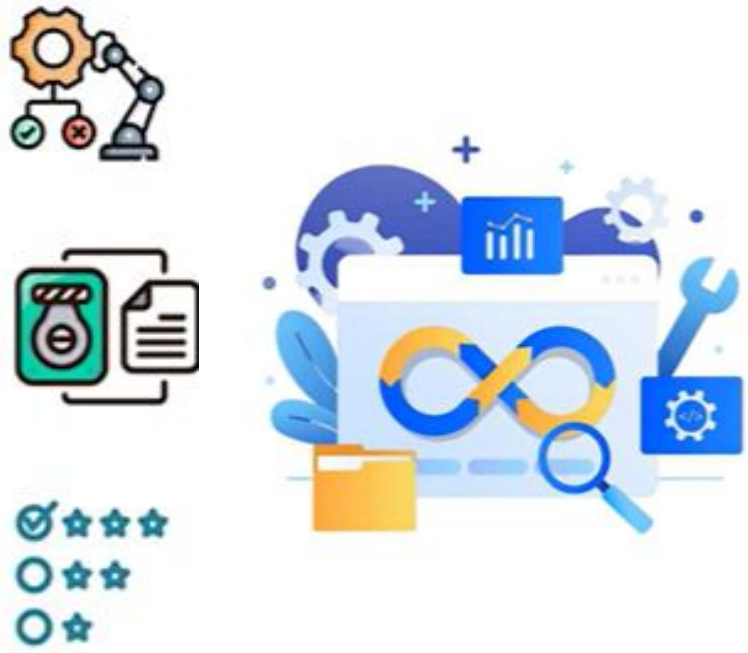
Mercurial

Perforce Helix Core

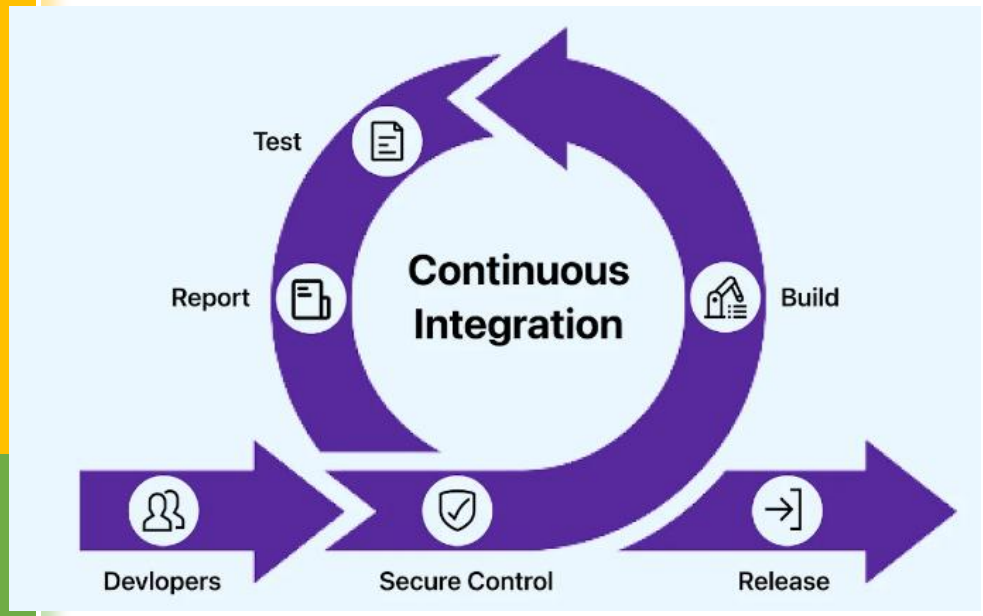
2. L’intégration continue (CI)

- L'intégration continue (CI) est une approche qui consiste à intégrer fréquemment les modifications de code dans un référentiel centralisé, suivies de processus automatisés de construction et de tests. Cela permet de détecter rapidement les problèmes d'intégration et de réduire le temps nécessaire pour valider et publier de nouvelles versions logicielles.





2. L’intégration continue (CI) - outils



GitLab

Jenkins

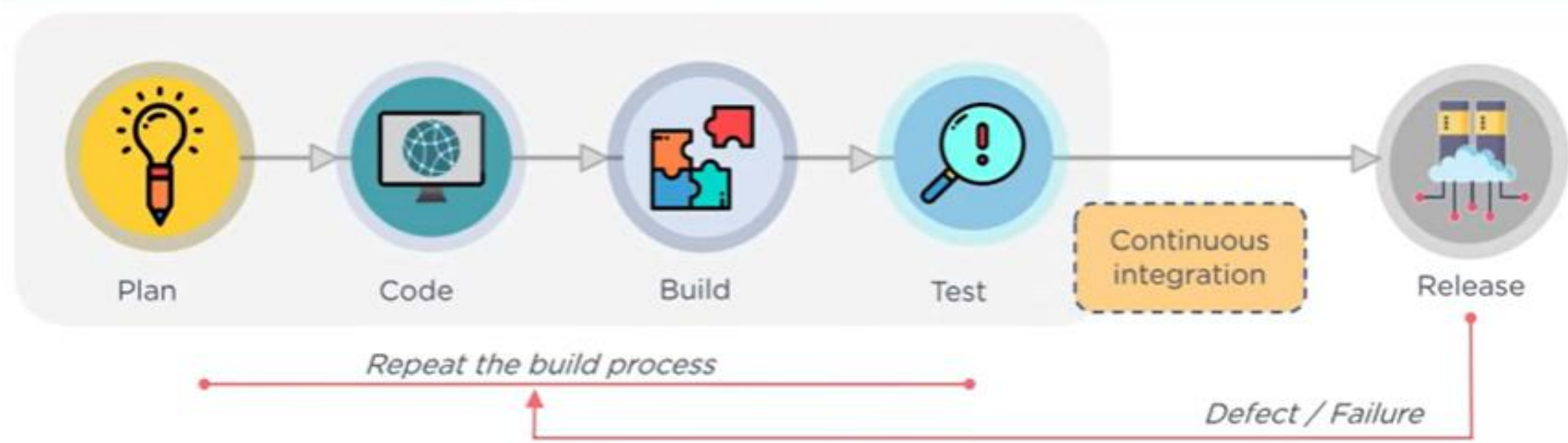
TeamCity

CodeShip

Bamboo

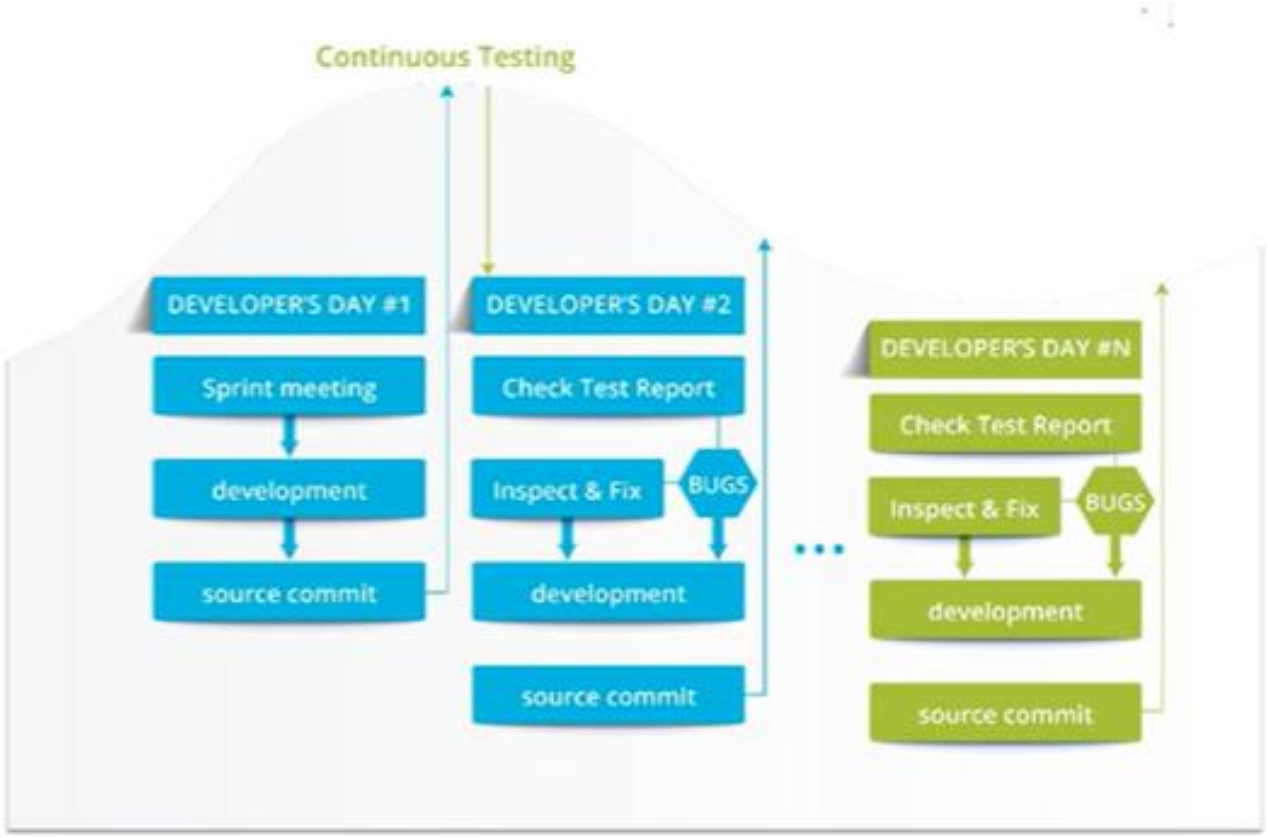
Travis CI

2. L’intégration continue (CI)



3. Les tests continus (CT)

- Les tests continus sont étroitement liés à l'intégration continue, où le code est intégré et testé régulièrement. Les tests automatisés sont exécutés à chaque intégration pour détecter rapidement les erreurs et les problèmes de compatibilité.



3. Les tests continus (CT) - outils

Selenium

JUnit

Postman

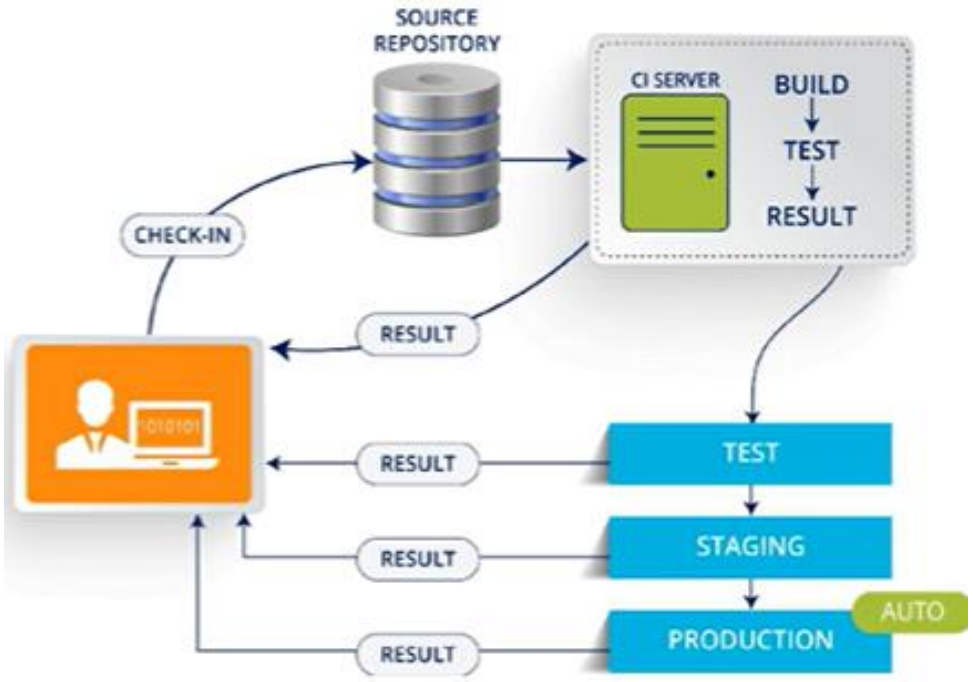
Appium

Cucumber

4. Le déploiement continu (CD)

- Le déploiement continu est une approche qui vise à automatiser le processus de déploiement des modifications logicielles en production, de manière régulière et fiable. Il s'inscrit dans une démarche plus large d'intégration et de livraison continues (CI/CD).

- Le déploiement se fait directement dans l'environnement de production, après des étapes de validation.



4. Le déploiement continu (CD) - outils

Jenkins

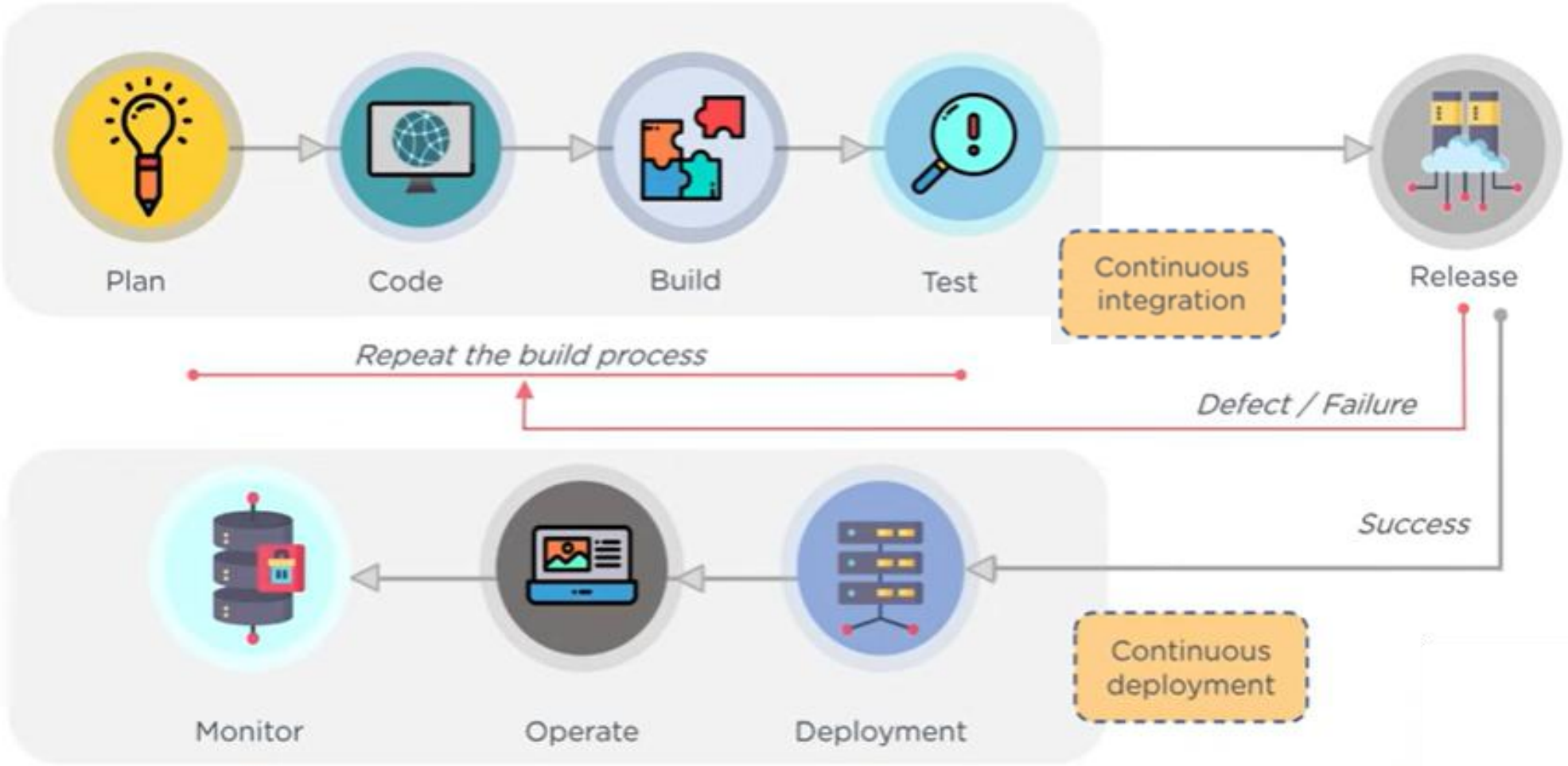
AWS CodeDeploy

GitLab

Azure DevOps

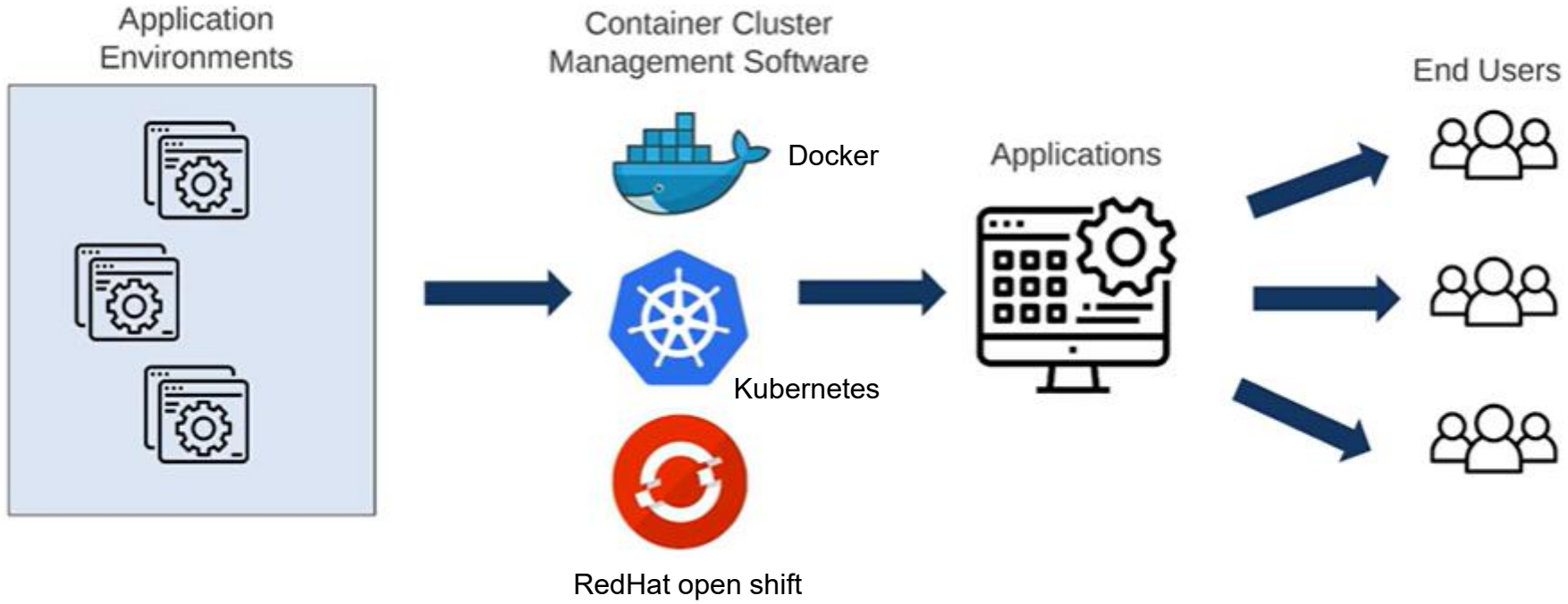
Octopus Deploy

4. Le déploiement continu (CD)



4. Le déploiement continu - la conteneurisation

- La conteneurisation consiste à regrouper dans une même entité toutes les ressources nécessaires au fonctionnement d'une application : le code, les fichiers de configuration, l'environnement d'exécution… Ainsi, cette entité est l'élément unique et nécessaire pour le déploiement d'une application.



5. Les opérations continues - la gestion de configuration - IaC

- La gestion de configuration est un processus par lequel les modifications apportées aux composants d'un produit sont systématiquement identifiées, organisées, contrôlées et maintenues tout au long du cycle de vie .
- Infrastructure-As-Code : elle fait référence à l'existence d'un code qui prépare automatiquement l'environnement nécessaire.

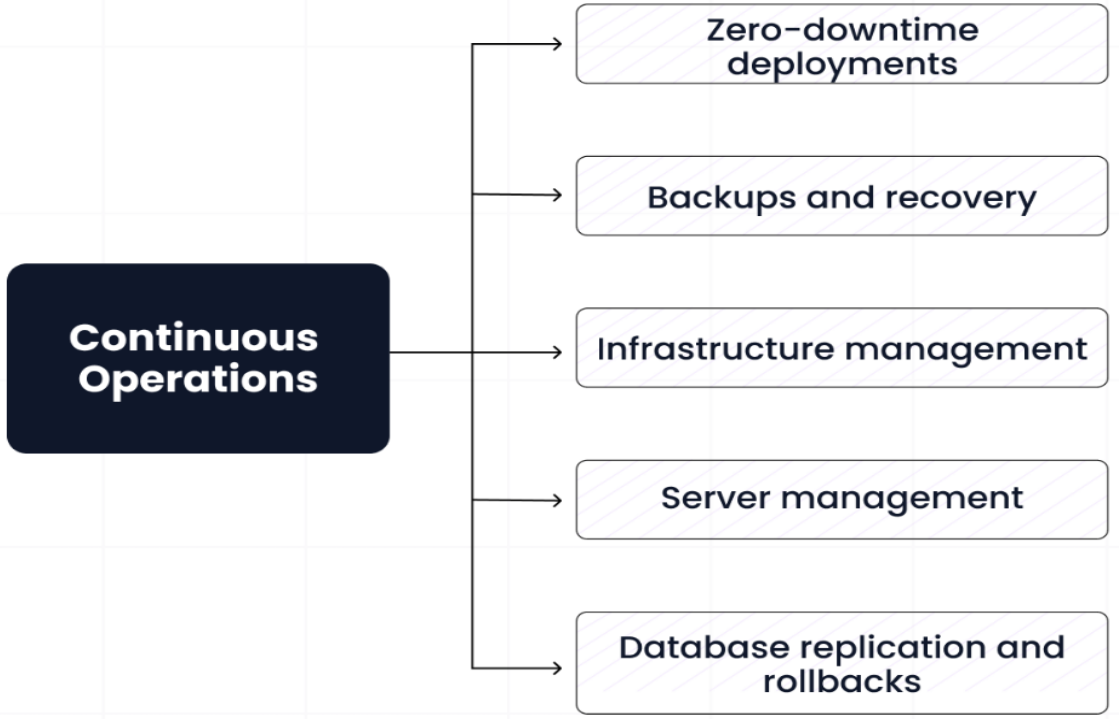
Ansible

Puppet

SaltStack

Terraform

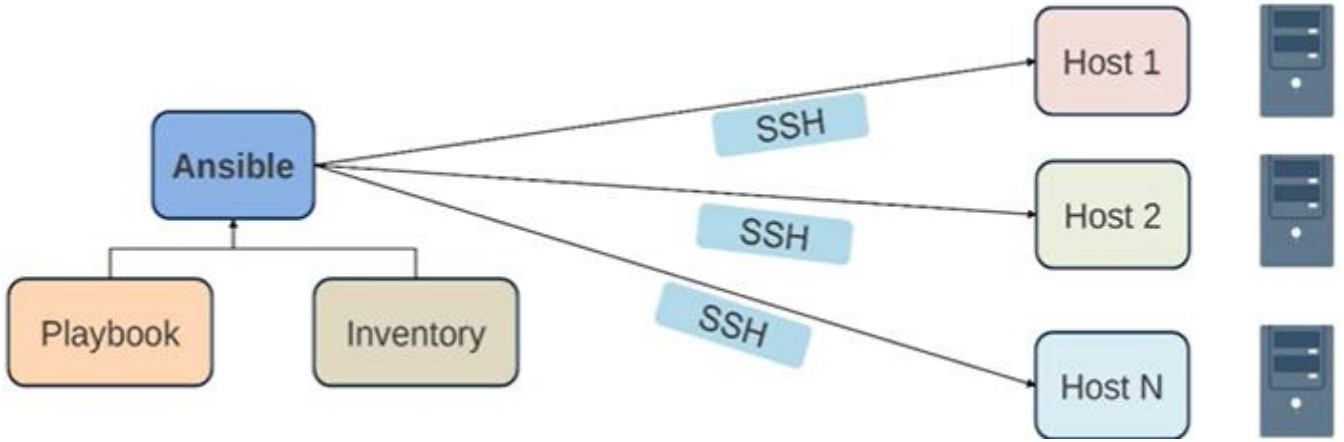
Chef



5. La gestion de configuration utilisant Ansible

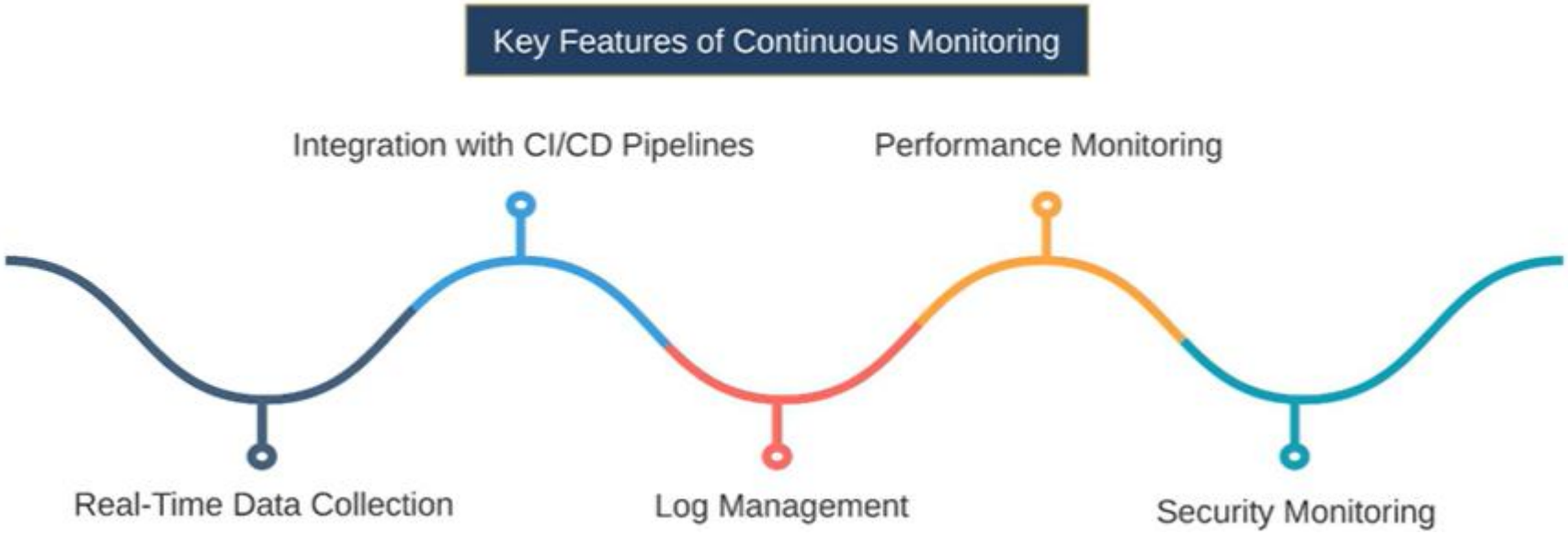
- Ansible est un outil clé pour la gestion de configuration et l'automatisation, permettant de définir et de déployer des configurations de systèmes de manière simple et reproductible.
- Il utilise des playbooks en YAML pour automatiser des tâches telles que le provisionnement, la configuration des serveurs et le déploiement d'applications.

- Playbooks: sont des fichiers YAML qui décrivent les tâches à exécuter. Ils spécifient les modules à utiliser, les cibles (serveurs) et les actions à réaliser.
- Modules: pour effectuer des actions spécifiques sur les serveurs (par exemple, installer un paquet, configurer un service, copier un fichier).
- Inventaire: Un fichier d'inventaire liste les serveurs gérés par Ansible.
- Exécution: Ansible se connecte aux serveurs via SSH (ou WinRM pour Windows) et exécute les tâches définies dans les playbooks.



6. La surveillance continue

- La surveillance continue (Continuous Monitoring ou CM) est un processus automatisé qui permet de suivre et d'analyser en temps réel les performances des systèmes, applications et infrastructures tout au long du pipeline DevOps.



6. La surveillance continue – pourquoi ?

- Surveillance en temps réel: La CM permet de suivre les métriques clés, les journaux et les événements du système en temps réel, fournissant une visibilité immédiate sur son état.
- Détection précoce des problèmes: En analysant les données collectées, la CM peut identifier les anomalies, les erreurs et les performances dégradées, permettant une intervention rapide avant que les problèmes ne s'aggravent.
- Automatisation des actions: La surveillance continue peut être intégrée à des systèmes d'alerte et d'automatisation, permettant une réponse rapide et automatisée aux problèmes détectés.
- Amélioration continue: En analysant les données collectées, la CM fournit des informations précieuses pour l'amélioration continue des systèmes et des processus.
- Sécurité renforcée: La surveillance continue est un élément clé de la sécurité DevOps, permettant de détecter et de répondre aux menaces de sécurité de manière proactive.

6. La surveillance continue – outils

Nagios

Prometheus

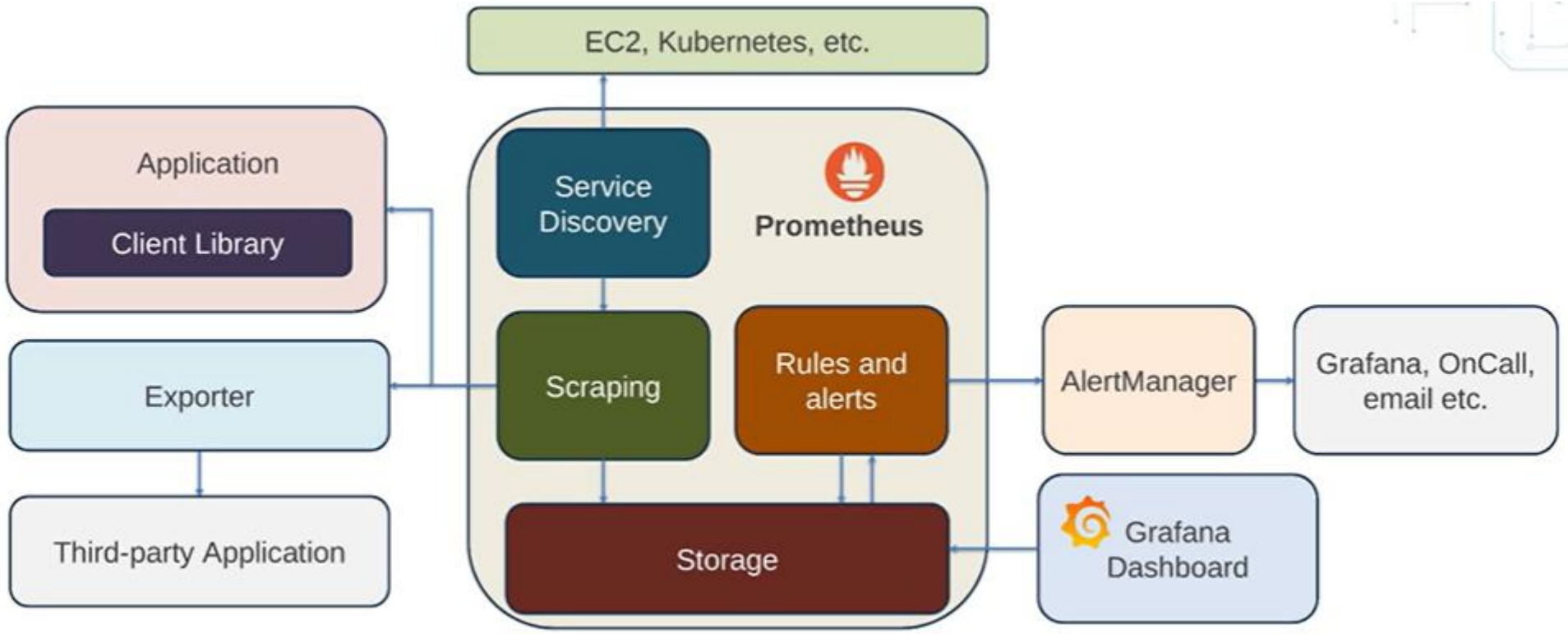
Grafana

New Relic

Splunk

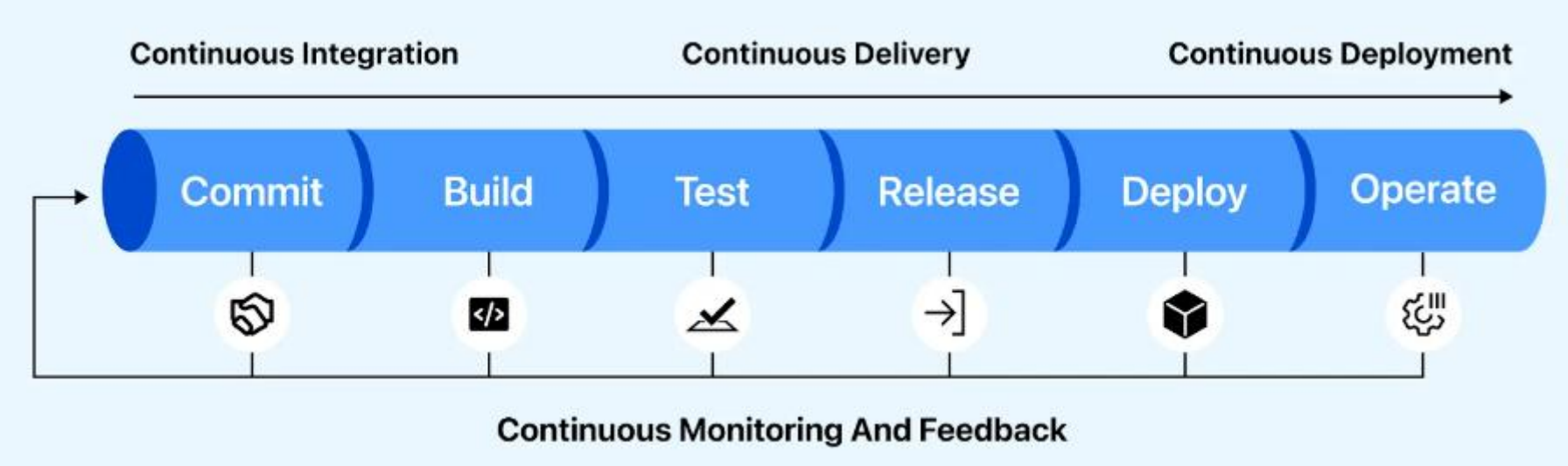
6. La surveillance continue - Prometheus

- Prometheus est un outil essentiel pour la surveillance continue. Il permet de collecter, stocker et interroger des métriques en temps réel, offrant une vue détaillée de l'état des systèmes et des applications. Cela aide à identifier rapidement les problèmes, à améliorer la stabilité et à optimiser les performances dans un environnement DevOps.



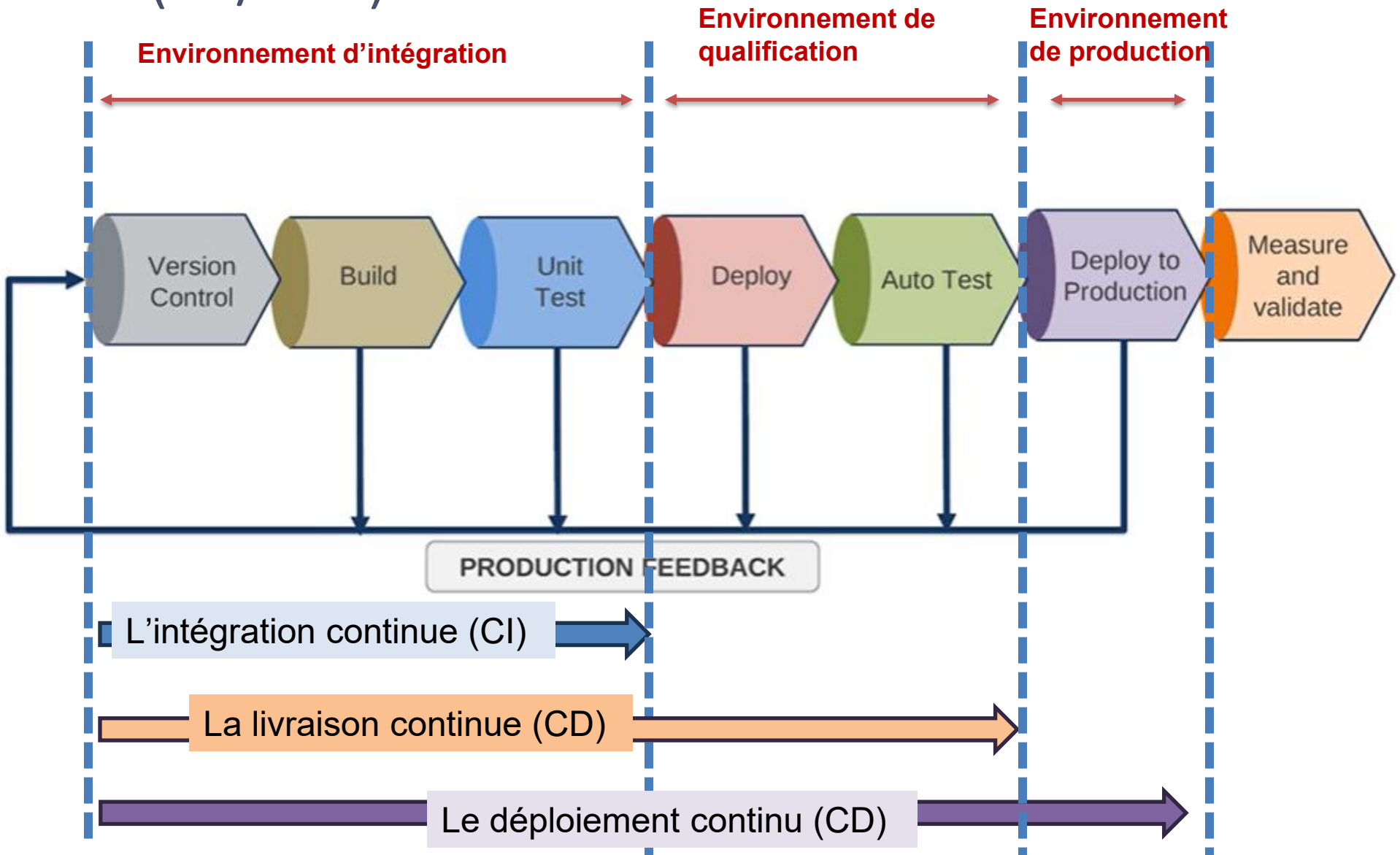
7. Feedbacks continus

- Un feedback continu nécessite la mise en place d'une boucle de feedback afin de recueillir des informations sur les performances logicielles auprès de votre équipe interne et de vos utilisateurs. Les commentaires sont ensuite partagés avec l'équipe DevOps pour vous aider à guider l'itération du produit. Les sources peuvent inclure des enquêtes, des questionnaires, des groupes de discussion, des médias sociaux, des forums, etc.
- Il ne s'agit pas seulement de déterminer si votre logiciel fonctionne correctement, mais aussi de mesurer la satisfaction globale des clients pour orienter la stratégie commerciale et garantir les meilleurs résultats possibles.



6. DevOps pipeline CI/CD

DevOps pipeline – intégration et déploiement continue (CI/CD)



DevOps - environnements

- **L’environnement d’intégration :** un dépôt centralisé partagé par les développeurs où ils intègrent leurs modifications de code (par exemple, un système de gestion de versions comme Git).

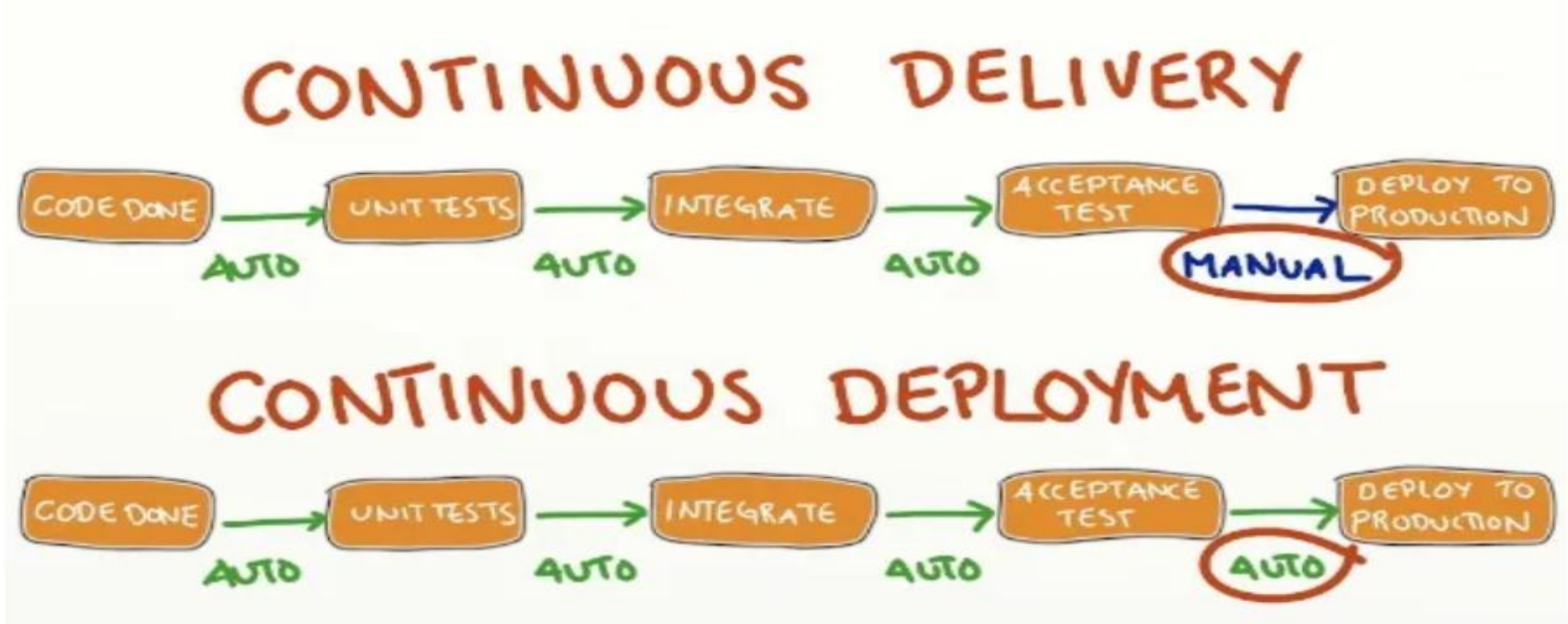
- **L’environnement de qualification :** est un environnement isolé utilisé pour tester et valider les changements avant leur déploiement en production.

- **L’environnement de production :** l'environnement final où l'application ou le service est mis à disposition des utilisateurs finaux. C'est là que le code et les fonctionnalités sont utilisés dans un cadre réel, après avoir été développés et testés. Le déploiement en production est l'aboutissement du processus DevOps, qui vise à assurer une livraison rapide et fiable des logiciels tout en maintenant la qualité et la stabilité.

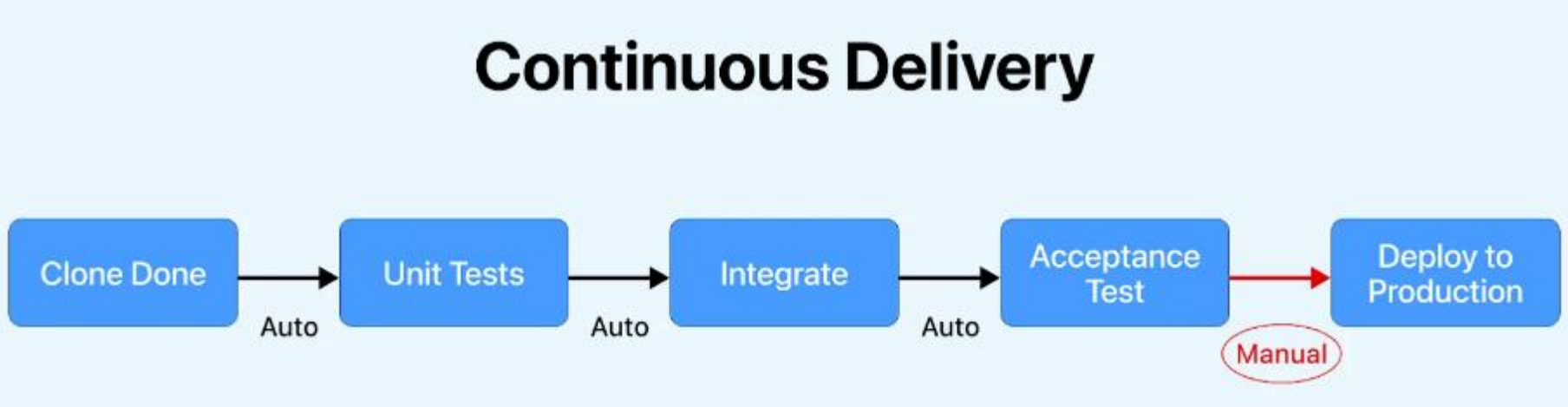
Déploiement continu vs Livraison continue (Continuous Deployment vs continuous delivery)

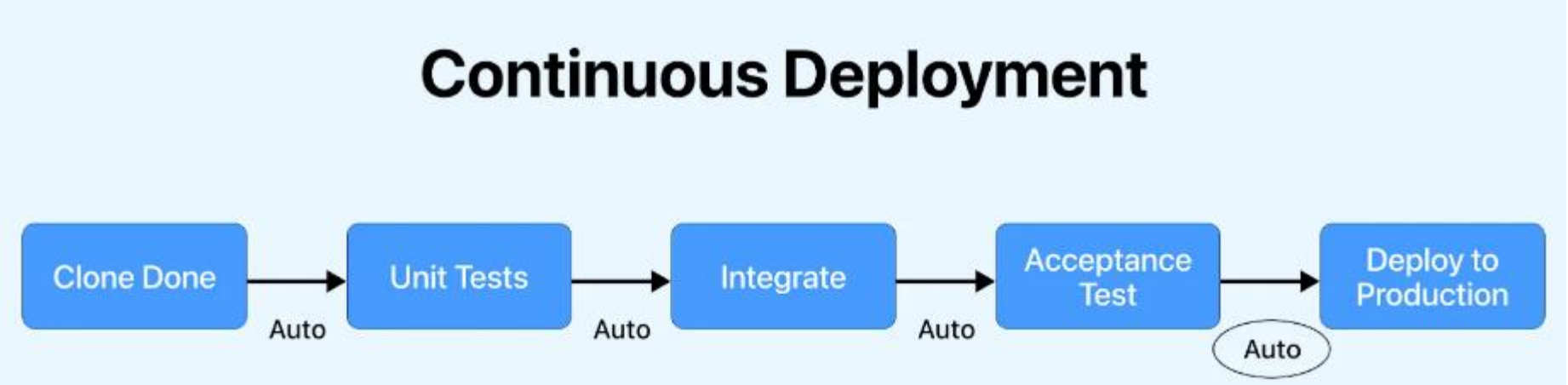
- La livraison continue automatise le processus de création, de test et de préparation des modifications de code pour la publication, mais nécessite une étape manuelle pour le déploiement en production.

- Le déploiement continu, quant à lui, automatise l'ensemble du processus, y compris le déploiement en production, une fois les tests automatisés réussis.



Déploiement continu vs Livraison continue (Continuous Deployment vs continuous delivery)





Pour information…

Top DEVOPS certifications 2025

- *"Apprendre par le projet, c'est découvrir par l'action, créer par la compréhension, et réussir par la persévérance."*

amina.jarraya@ensi-uma.tn

ENSI Manouba