

Introduction DevOps



**UP ASI (Ex JavaEE .Net)
Bureau E204**

Plan du Cours

- Introduction (plan module DevOps, horaire, évaluation, définition DevOps)
- Dev vs Ops
- Dev && Ops
- Continuos Integration / Continuos Deployment (CI/CD)
- Technologies DevOps
- Pratiques DevOps
- Culture DevOps
- Avantages DevOps
- Outils
- Solution finale

Contenu du Module DevOps

- Introduction DevOps
- Jenkins (Serveur d'intégration continue)
- Introduction à Docker
- Git
- Nexus (Gestion des livrables)
- Test unitaire (JUnit)
- Sonar (Qualité de code)
- Docker avancé (Docker compose + Docker volume)
- Grafana et Prometheus
- Validation projet

Horaires

- Durée Totale : **30 heures**
- Séances : **10 séances**
 - Cours : **09 heures**
 - TP : **15 heures**
 - Examen : **06 heures**
- Durée de chaque Séance : **3 heures**
(02h00 synchrone + 01h00 asynchrone)
- * Vendredi : **13h45**

Evaluation

- L'évaluation se fait tout au long du module, et non pas uniquement à la fin.
- La moyenne du module est calculée comme suit :
Moyenne = Note Examen pratique * 100%
- La note de l'examen tient en considération l'Assiduité, la Participation, les **TP** à faire en cours (avancement sur votre projet chaque semaine)
- L'**Examen** sera pratique au cours des deux dernières séances (S9 et S10).
- L'examen de rattrapage est **pratique**.

Introduction

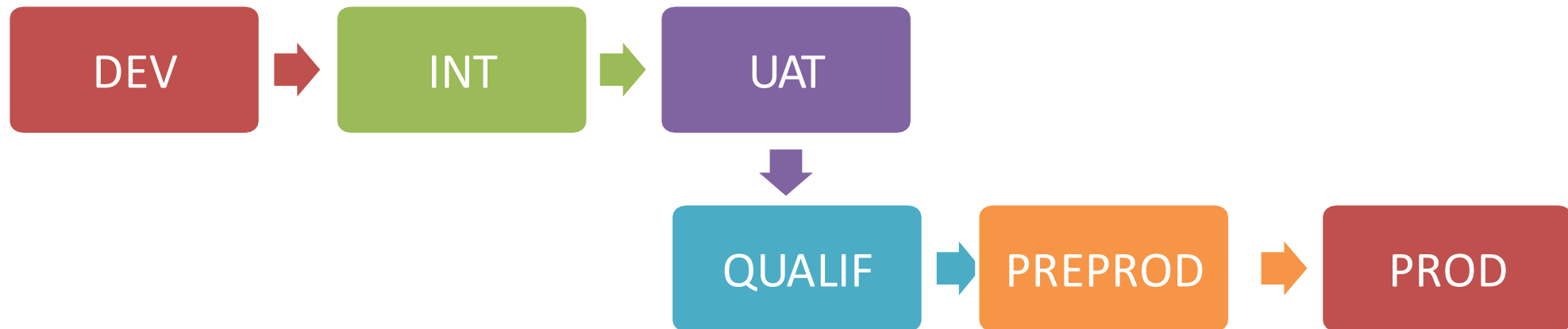


C'est quoi DevOps ?

Introduction – Cycle de vie d'un projet

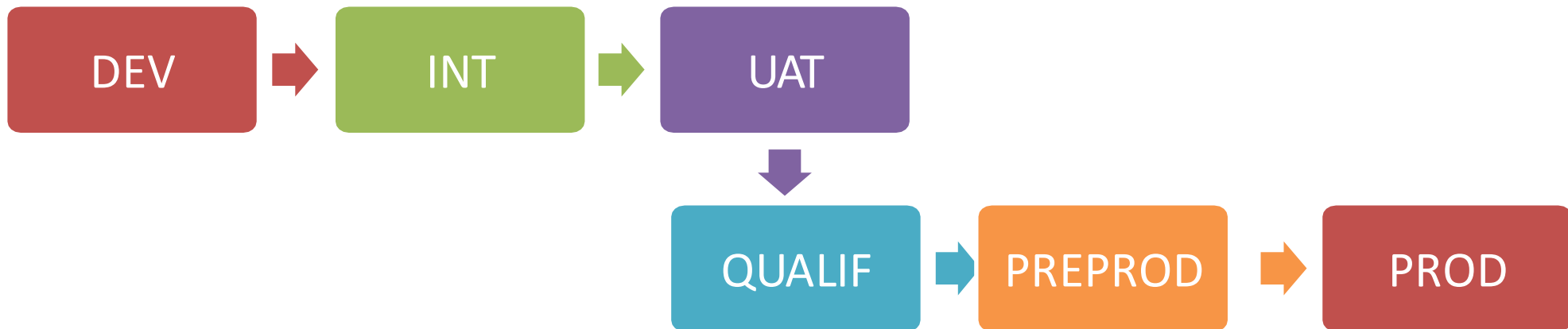


Introduction – Environnements



- 1. Environnement de Développement (DEV):** C'est l'environnement où les développeurs créent et modifient le code source de l'application. Il s'agit d'un environnement isolé où les développeurs travaillent sur leurs propres machines pour développer et tester le code.
- 2. Environnement d'Intégration (INT):** L'environnement d'intégration est l'endroit où les différents composants logiciels développés par les membres de l'équipe sont combinés pour vérifier leur compatibilité et leur fonctionnement ensemble. Cela permet de détecter et de résoudre les problèmes d'intégration.

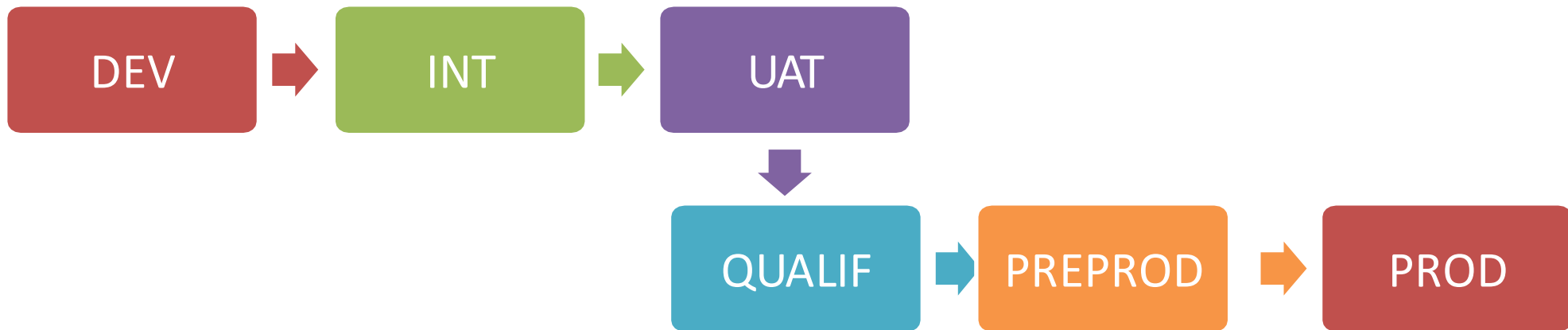
Introduction – Environnements



3. Environnement de Test (UAT - User Acceptance Testing): L'environnement de test est utilisé pour effectuer des tests approfondis de l'application, généralement par des testeurs. L'objectif est de s'assurer que l'application répond aux exigences spécifiées par les utilisateurs et de corriger les erreurs avant la mise en production.

4. Environnement de Qualification (QUALIF): L'environnement de qualification est principalement utilisé pour effectuer des tests de qualification. L'objectif est de vérifier que l'application répond à un ensemble spécifique de normes, de réglementations ou de critères de qualité définis. Ces normes peuvent varier en fonction de l'industrie ou du domaine d'application.

Introduction – Environnements



5. Environnement de Pré-Production (PREPROD): L'environnement de pré-production est davantage axé sur les tests finaux de l'application avant son déploiement en production. L'objectif est de s'assurer que l'application fonctionne correctement dans un environnement similaire à celui de production.

6. Environnement de Production (PROD): C'est l'environnement dans lequel l'application est déployée et utilisée par les utilisateurs finaux. Il s'agit de l'environnement "en direct" où l'application est opérationnelle et accessible aux utilisateurs.

Introduction

Quels sont les problèmes qui ont donné naissance à DevOps ?

- Ca marche chez moi et pas chez toi!
- Les déploiements risqués
- Le peur du changement
- Quel est le problème ?
- Retardons la livraison à la semaine prochaine (corrigeons nos bugs!)



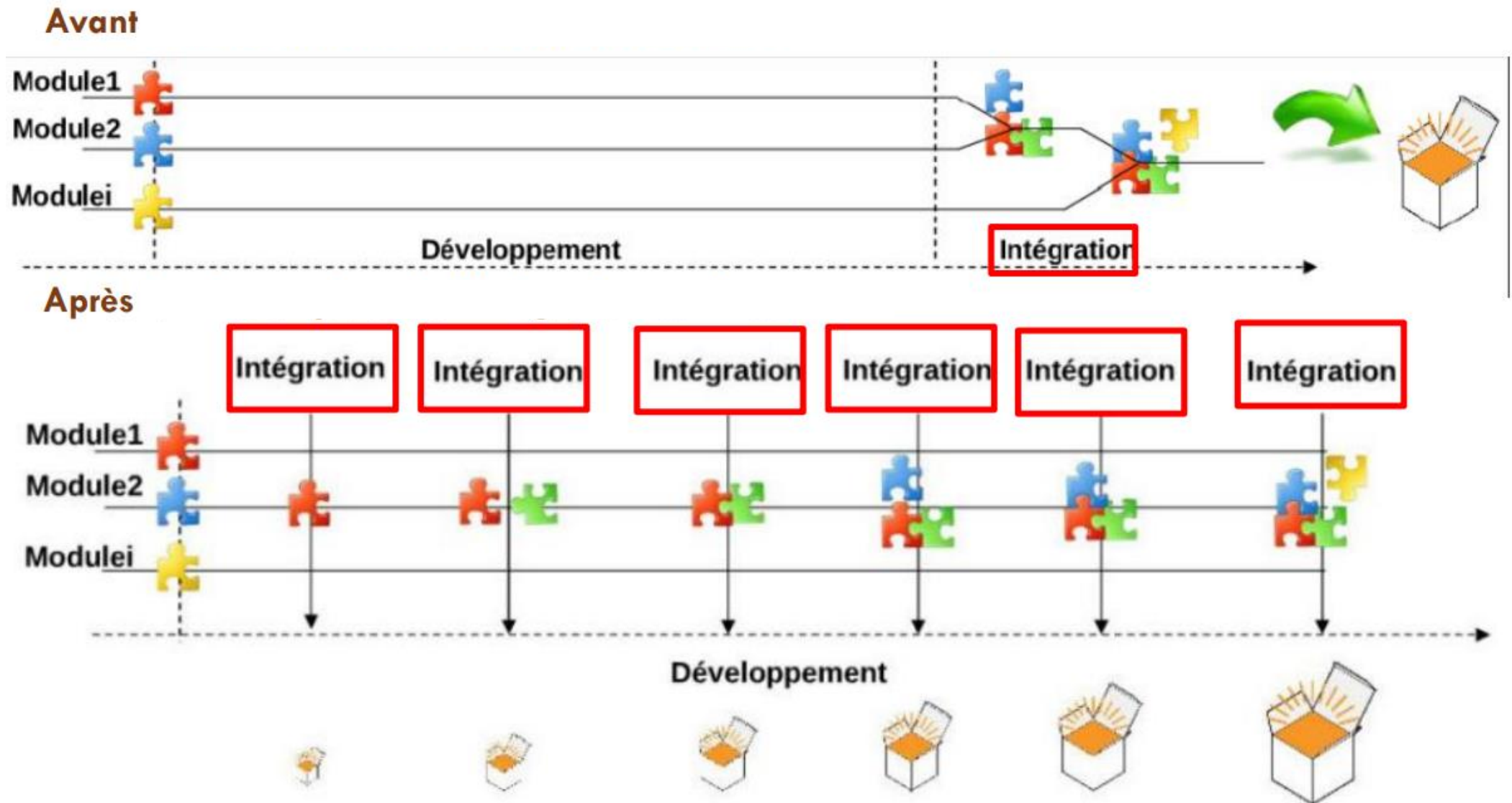
Introduction

DevOps est la contraction des deux termes anglais **development** (développement) et **operations** IT (exploitation)

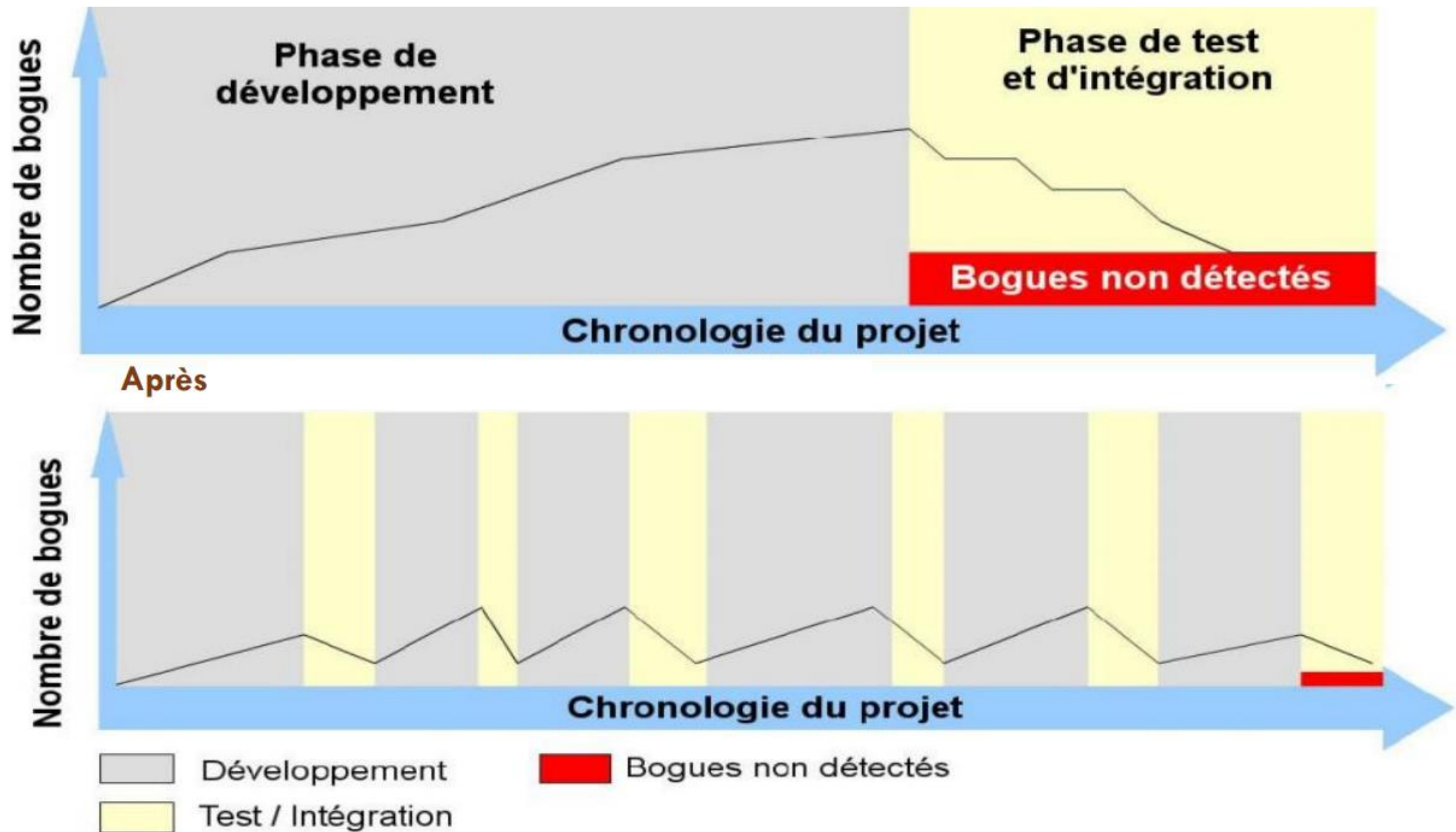
DevOps vise à **réduire la friction** organisationnelle entre les développeurs et l'équipe opérationnelle

L'approche DevOps vise une **meilleure communication** entre les deux équipes en **automatisant les fonctions souvent distinctes de ces équipes en un seul processus intégré et continu afin d'optimiser la production des livrables.**

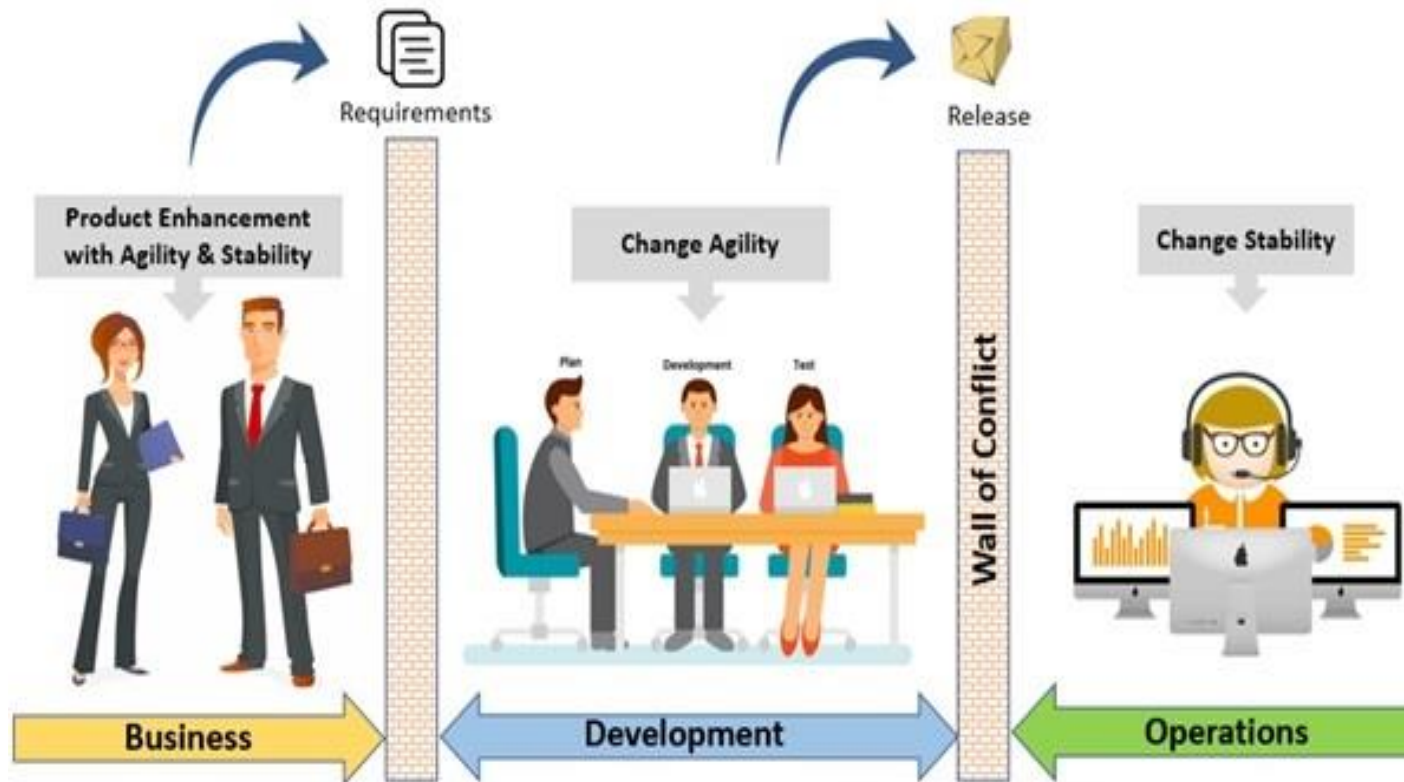
Avant DevOps



Avant DevOps



Dev vs Ops



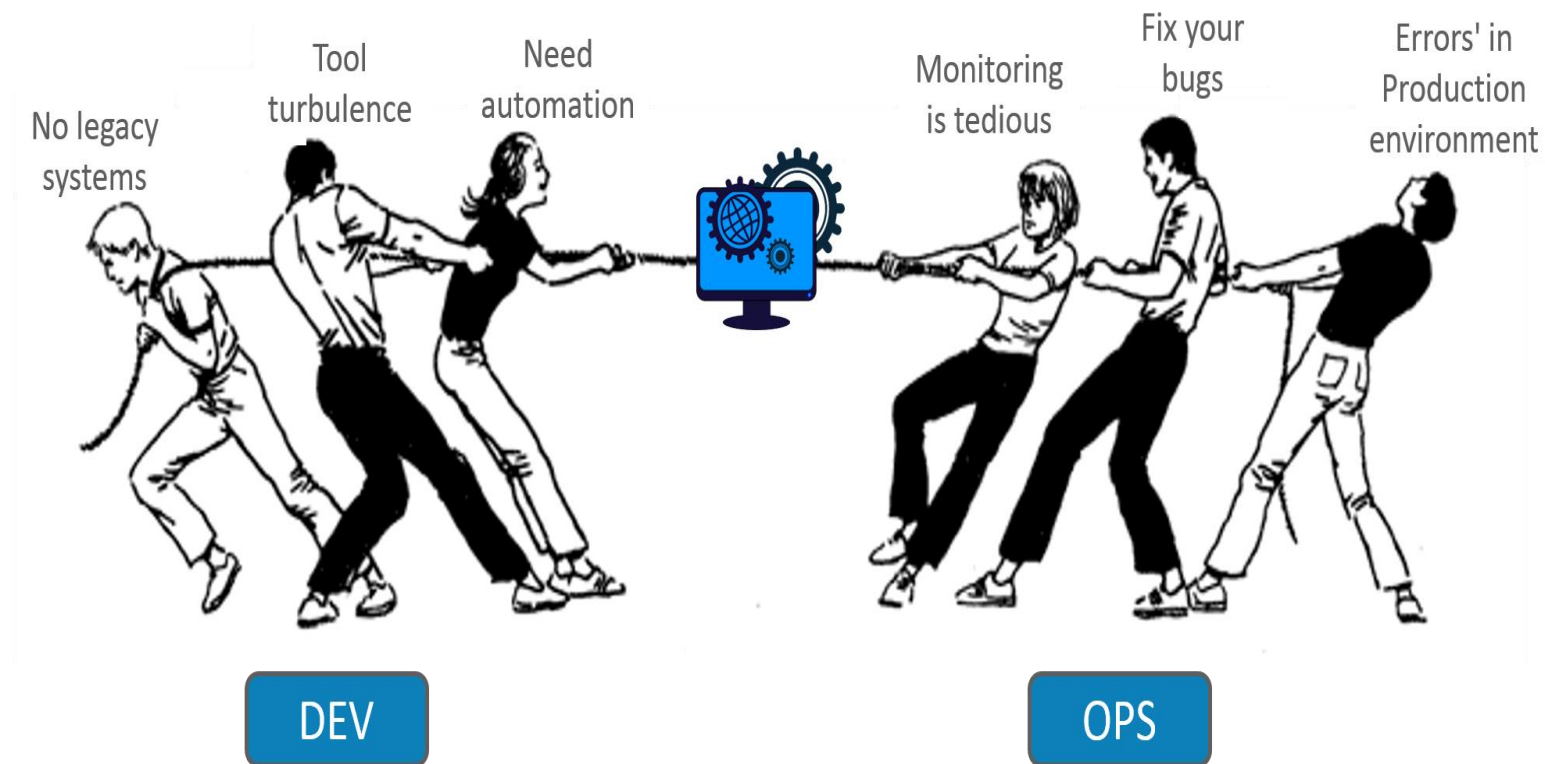
- 3 équipes sont indispensables pour la bonne conduite d'un projet : équipe business, équipe développement et équipe opérationnelle.
- L'interaction entre ces 3 équipes est loin d'être évidente

Dev vs Ops

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

Les objectifs et les visions des deux équipes développement et opérations sont différentes.

Dev vs Ops



Dev: Equipes de développeurs logiciels

=> Modification aux moindres coûts, le plus rapidement possible

Ops: Equipes en charge de la mise en production des produits

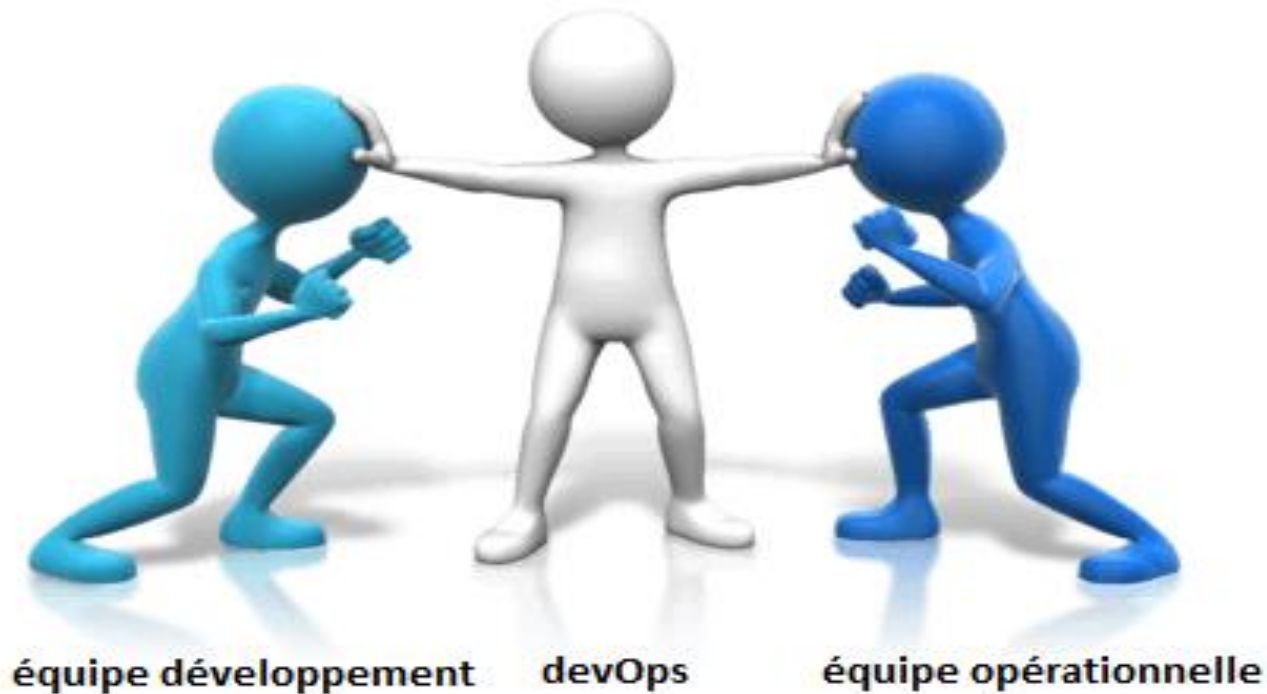
=> Stabilité du système, qualité

L'automatisation est au cœur de l'approche DevOps

Dev vs Ops

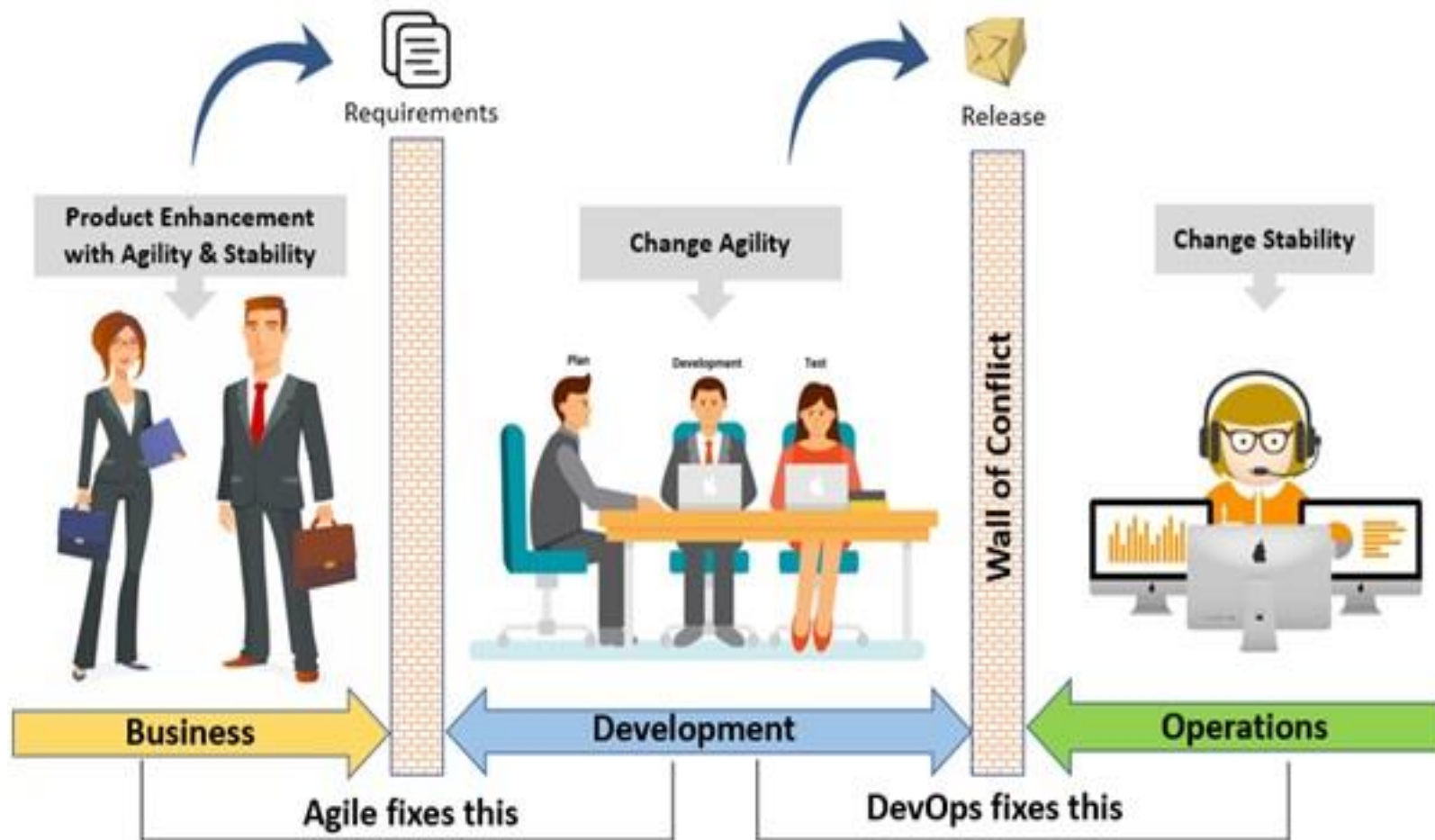


Dev vs Ops



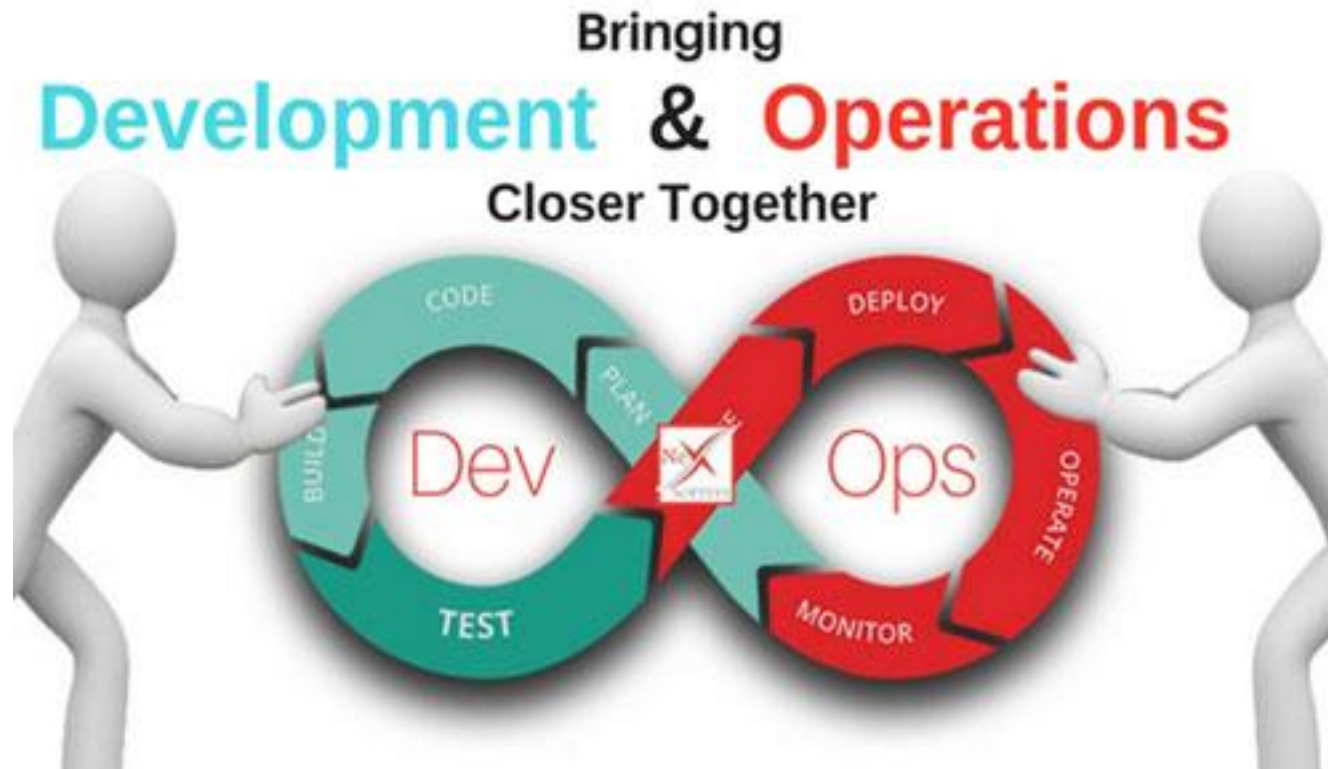
- La culture DevOps joue le rôle de **médiateur** entre les deux équipes dev et ops.
- L'objectif est de transformer la tension entre les deux équipes en une collaboration saine

Dev vs Ops



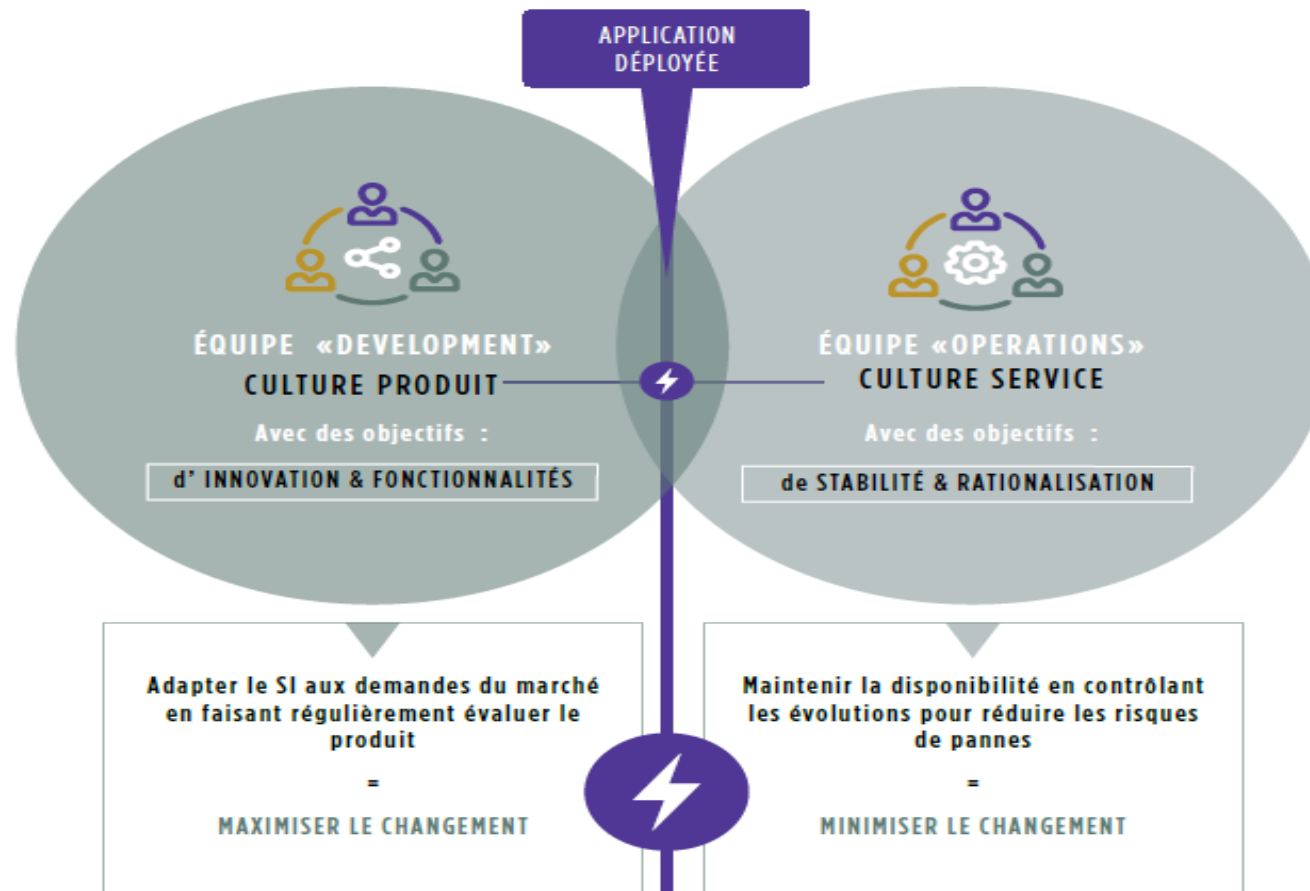
L'agilité et les pratiques DevOps interviennent pour briser les frontières entre les différents collaborateurs.

Dev & Ops



La solution parfaite à intérêt mutuel entre les deux parties est de coordonner les efforts pour que tous les objectifs des équipes soient réalisés avec moins de coût et dans le plus bref des délais

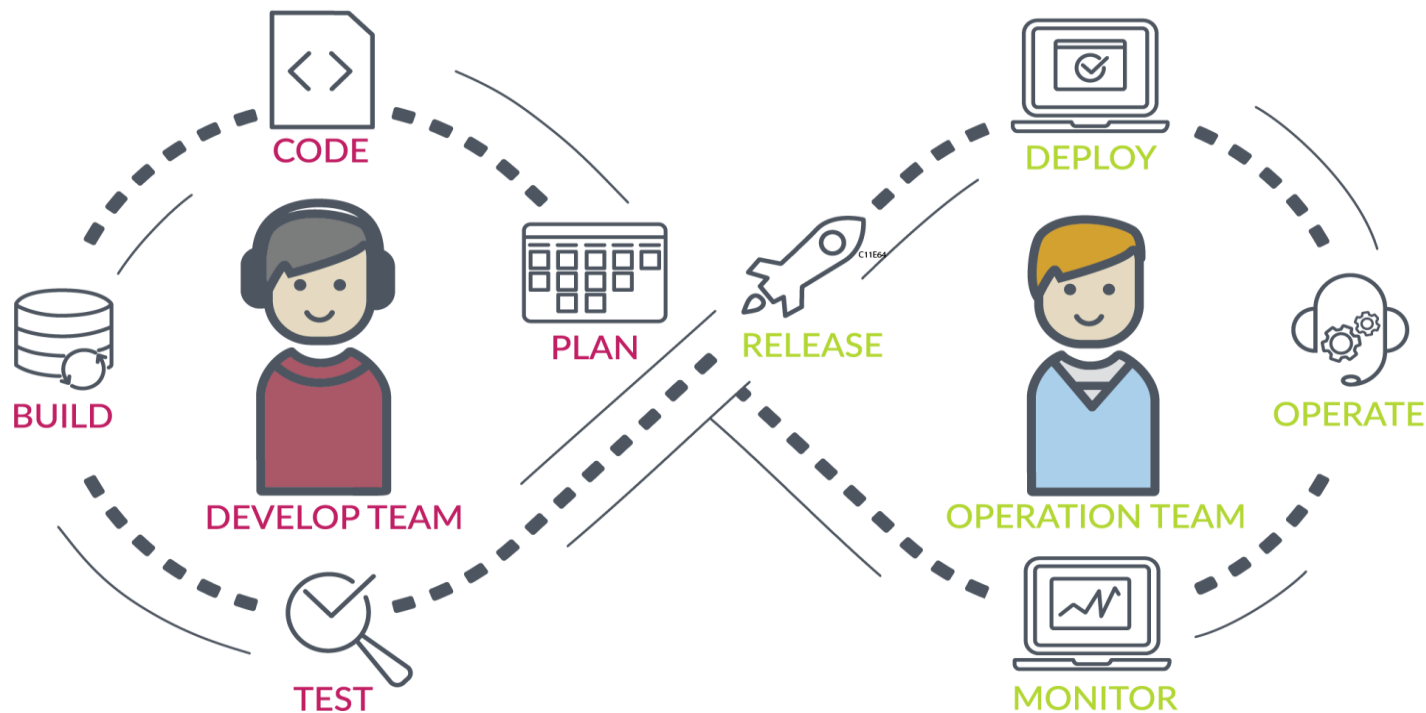
Dev & Ops



Le changement est le point de désaccord entre les deux équipes. DevOps intervient pour assurer à la fois la possibilité d'innover les fonctionnalités tout en étant sûr que la solution déployée est stable sans erreurs

Dev & Ops

DEVOPS PROCESS



Création d'un pipeline automatisé entre les deux équipes appelées CI/CD (continuous integration/ continuous delivery (ou deployment))

Avantages de DevOps



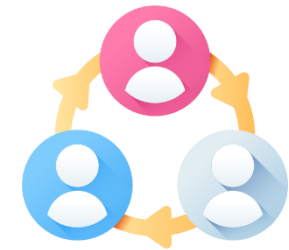
RÉDUCTION DU
CYCLE DES
LIVRAISONS



OPTIMISATION
DES RESSOURCES



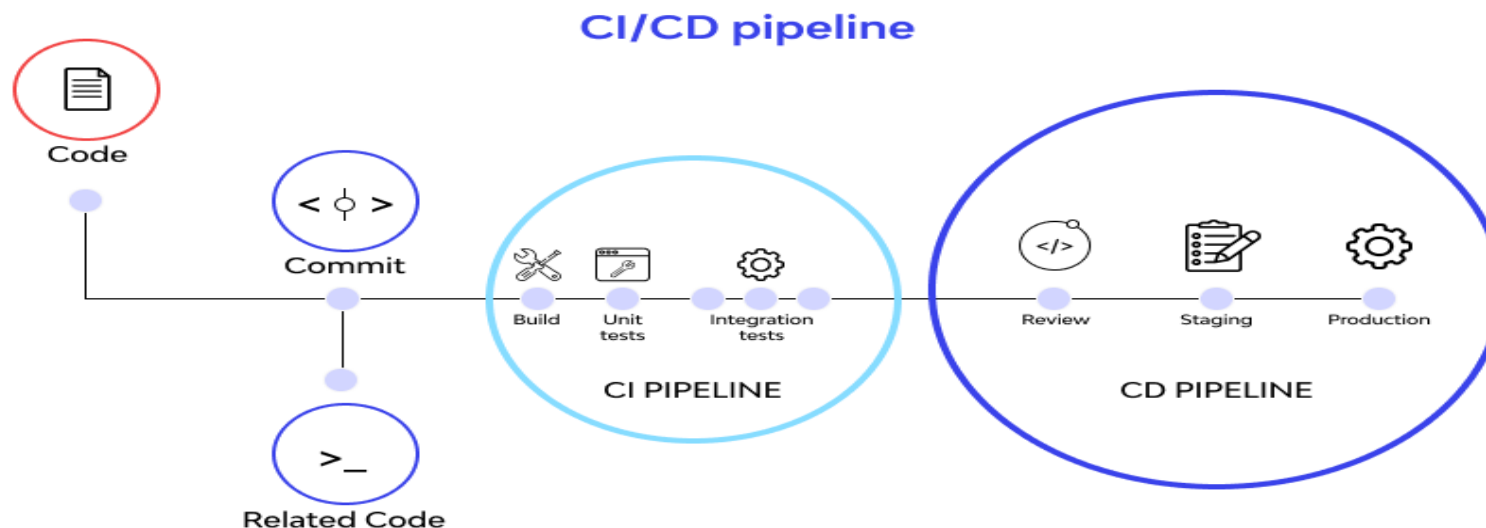
AMÉLIORATION
DE LA QUALITÉ



REPLACER L'HUMAIN
AU CŒUR DU
DISPOSITIF

CI/CD

- La mise en œuvre du modèle DevOps a pour résultat majeur la création d'un pipeline d'intégration et de déploiement continu (CI/CD).
- L'approche CI/CD vous aide **à fournir régulièrement des applications** aux clients et à valider **la qualité des logiciels** en réduisant au maximum les interventions humaines.



CI/CD

- L'approche CI/CD garantit une automatisation et une surveillance continues tout au long du cycle de vie des applications, des phases d'intégration et de test jusqu'à la phase de déploiement.
- Les problèmes et les bugs peuvent être identifiés rapidement en optimisant le coût de la modification
- La réussite de cette approche repose sur le succès de la collaboration agile entre les deux équipes dev et ops.

Continuous Integration/ Continuous Deployment (CI/CD)

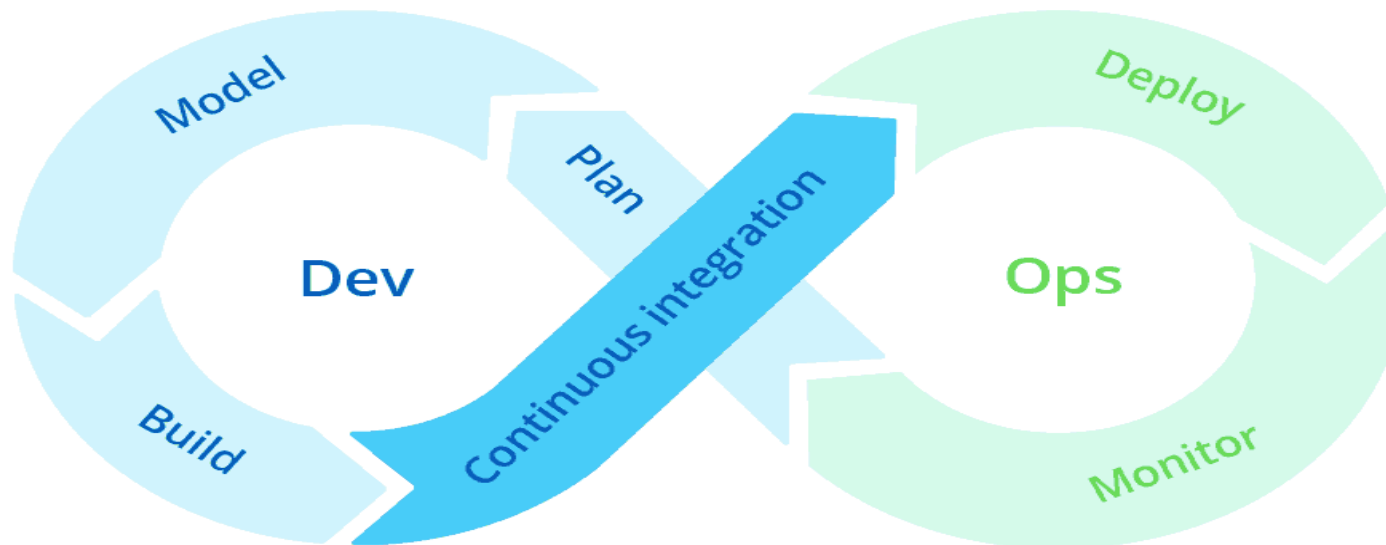
Il existe 3 processus DevOps :

- **Intégration continue**
- **Livraison continue**
- **Déploiement continu**

Continuous Integration/ Continuous Deployment (CI/CD)

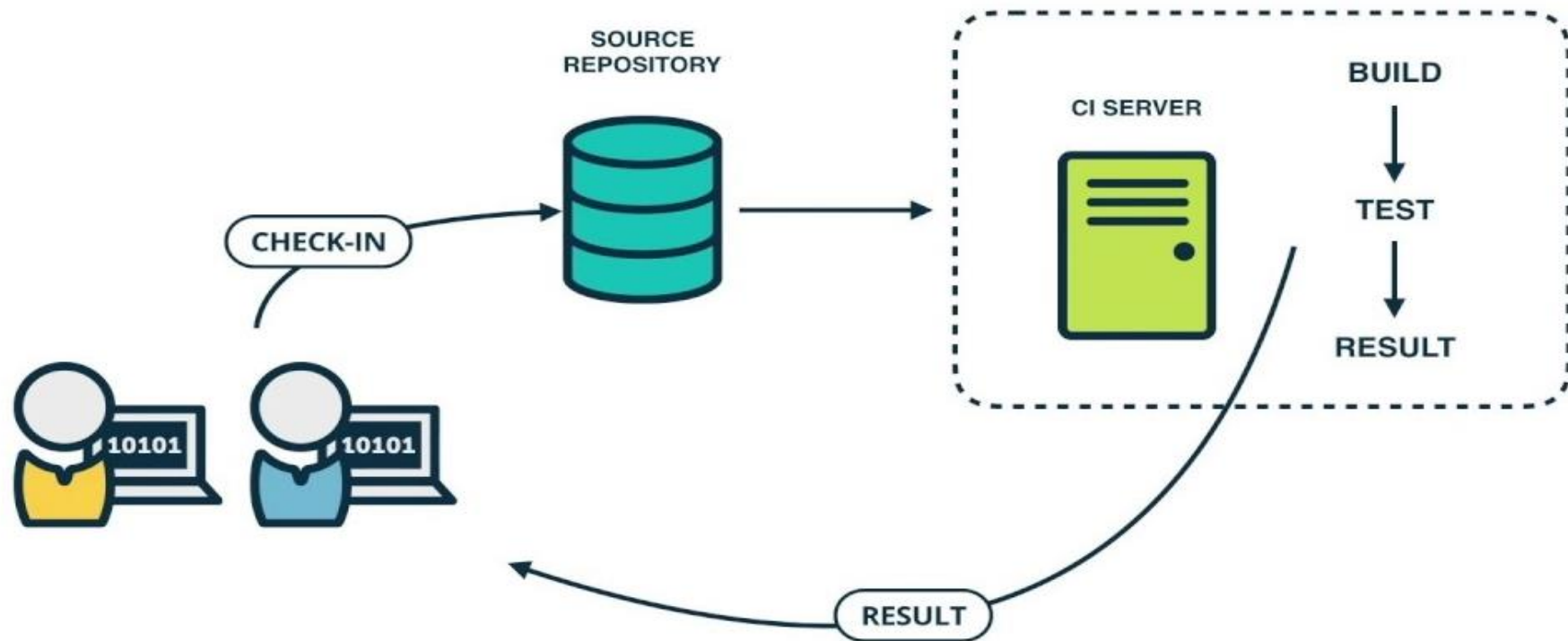
Intégration continue :

- L'intégration continue est un processus consistant à compiler, tester et **déployer** sur un environnement d'intégration (Environnement de dev).
- L'objectif est de détecter les régressions (bugs) du livrable à l'avance pour les fixer en s'appuyant sur des outils.



Continuous Integration/ Continuous Deployment (CI/CD)

Intégration continue :



A chaque modification du code, le processus de build, lancement des tests unitaires et vérification de la qualité du code est lancé. Les développeurs sont avertis en temps réel en cas d'erreur

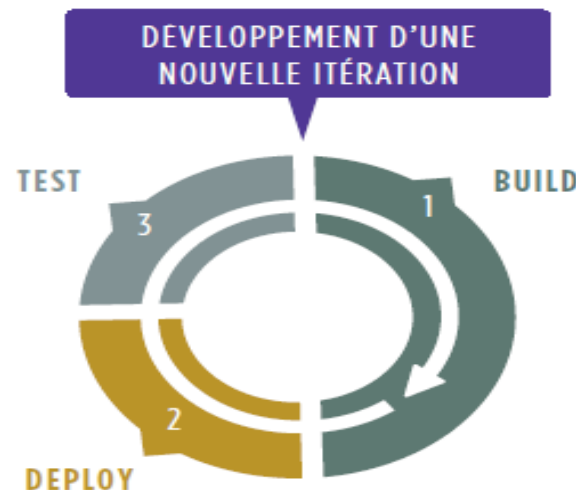
Continuous Integration/ Continuous Deployment (CI/CD)

Livraison continue

C'est la suite logique à l'étape d'intégration continue

La **livraison continue** est une discipline où l'application est construite de manière à [pouvoir être mise en production à n'importe quel moment](#).

C'est un processus automatisé visant à compiler, tester et **livrer (déployer) une application** à chaque modification apportée par un programmeur



Continuous Integration/ Continuous Deployment (CI/CD)

Déploiement continu :

Le déploiement continu est un processus de production, L'objectif est de compiler et déployer une application en production. Ce processus exige que les deux process Intégration continu et Livraison continu aient été réalisés avec succès.

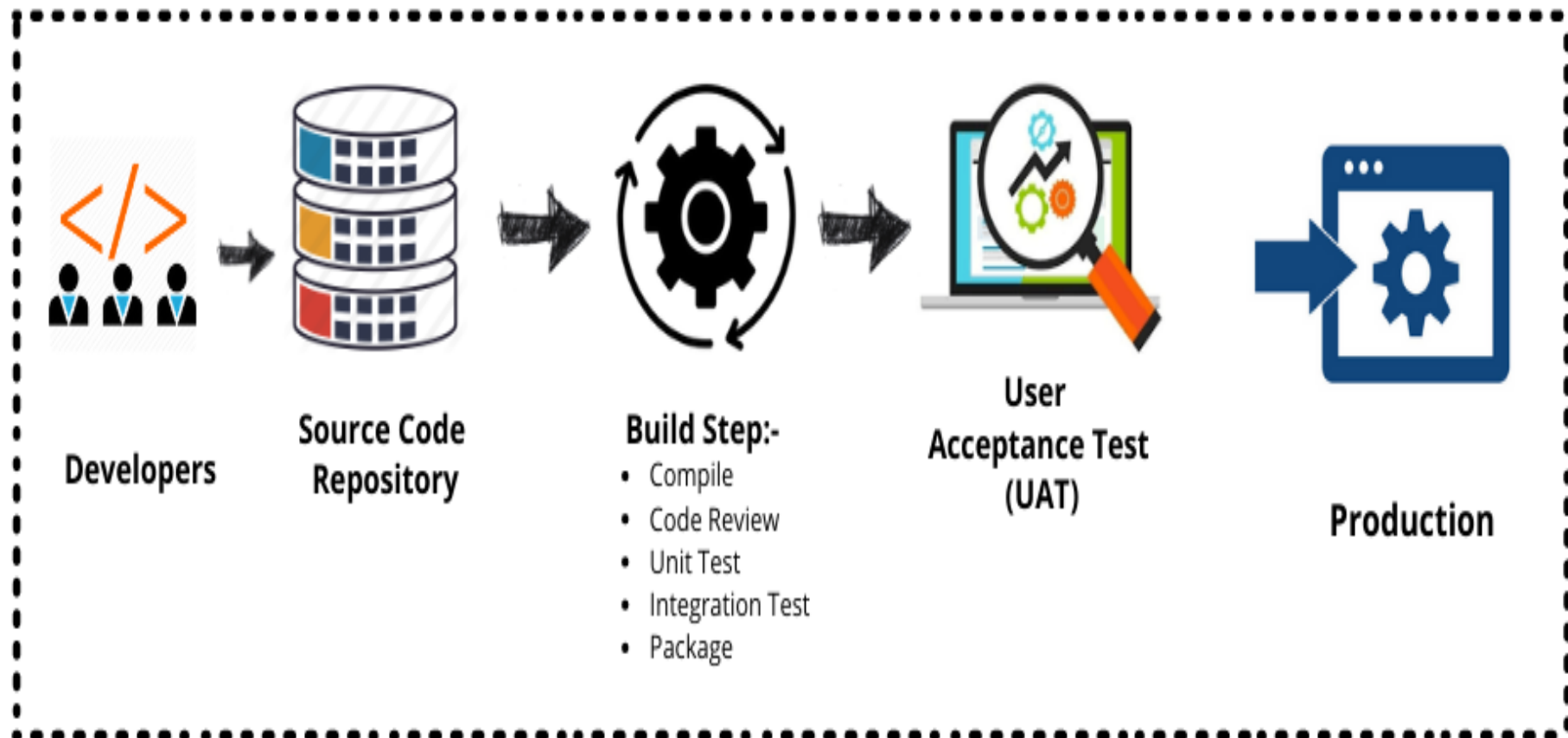
Il est primordiale de mesurer les impacts suite au déploiement grâce à des outils de supervision. En cas de détection de problèmes, un processus de retour en arrière doit être exécuté



Continuous Integration/ Continuous Deployment (CI/CD)

Déploiement continu :

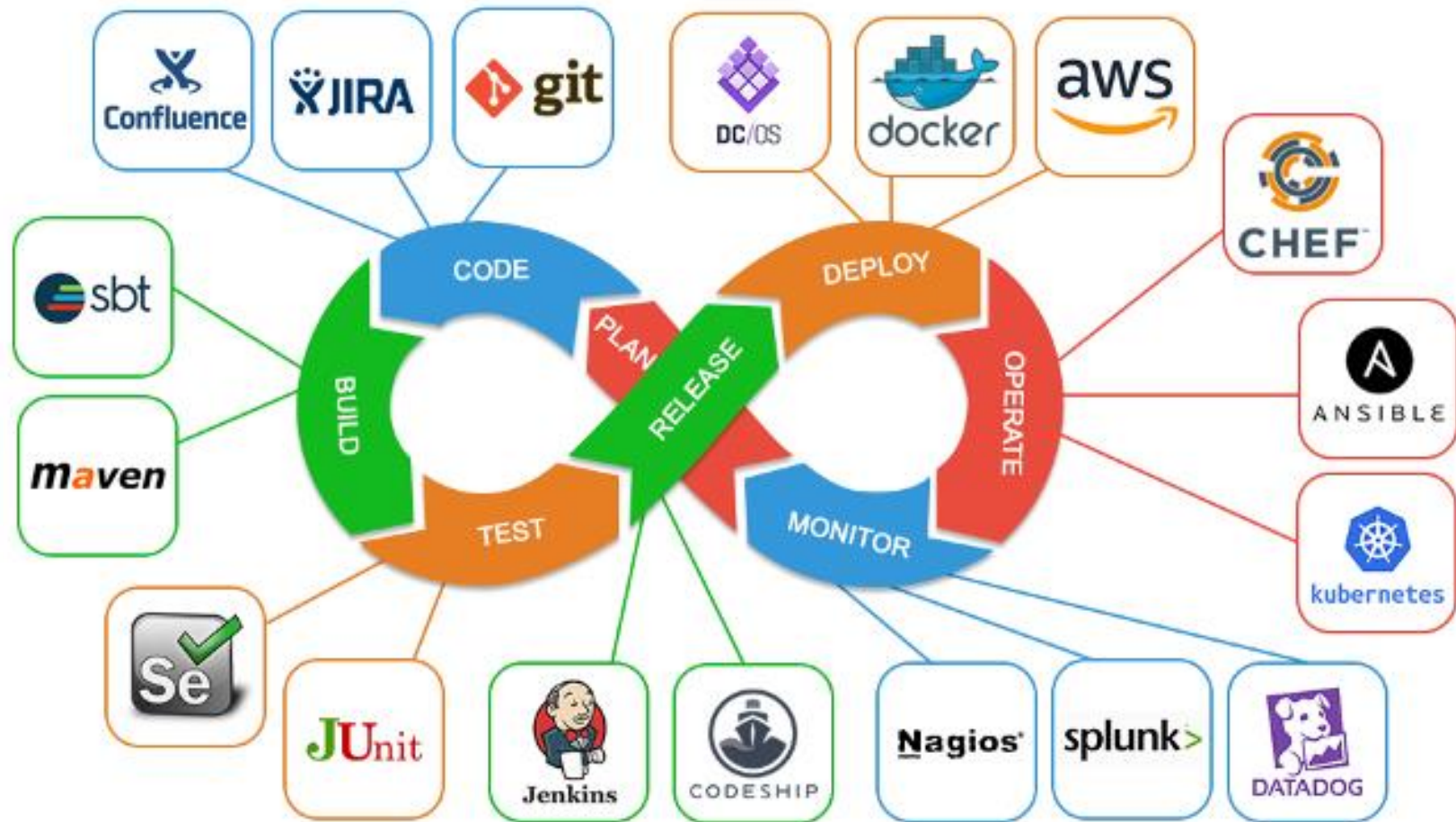
CONTINUOUS DEPLOYMENT



Livraison continue vs. déploiement continu

- On parle de livraison continue lorsque les développeurs livrent souvent et de manière régulière du nouveau code à tester aux équipes de l'assurance qualité (QA) et de l'exploitation. Cette démarche suppose généralement un environnement de test semblable à l'environnement de production et implique souvent un laps de temps entre la publication et le moment où une version est testée, où les modifications sont manuellement acceptées et où le nouveau code est mis en production.
- À l'inverse, le déploiement continu ne nécessite aucune évaluation ni vérification manuelles des changements de code dans l'environnement de test, puisque les tests automatisés sont intégrés très tôt dans le processus de développement et se poursuivent tout au long des phases du lancement. Le déploiement continu présente un avantage certain : l'absence d'attente entre la réussite des tests d'un changement de code aux niveaux de l'application et de la plateforme et sa mise en production.

Technologies DevOps



DevOps : pipeline automatisé d'outils

Outils DevOps

- Lien important qui montre les outils les plus utilisés en DevOps, suivant les technologies utilisées :

<https://www.devopsschool.com/path/>

- Il suffit de choisir la technologie : Java, Python, .NET ... pour avoir les outils DevOps les plus utilisés pour ces technologies.

Outils DevOps

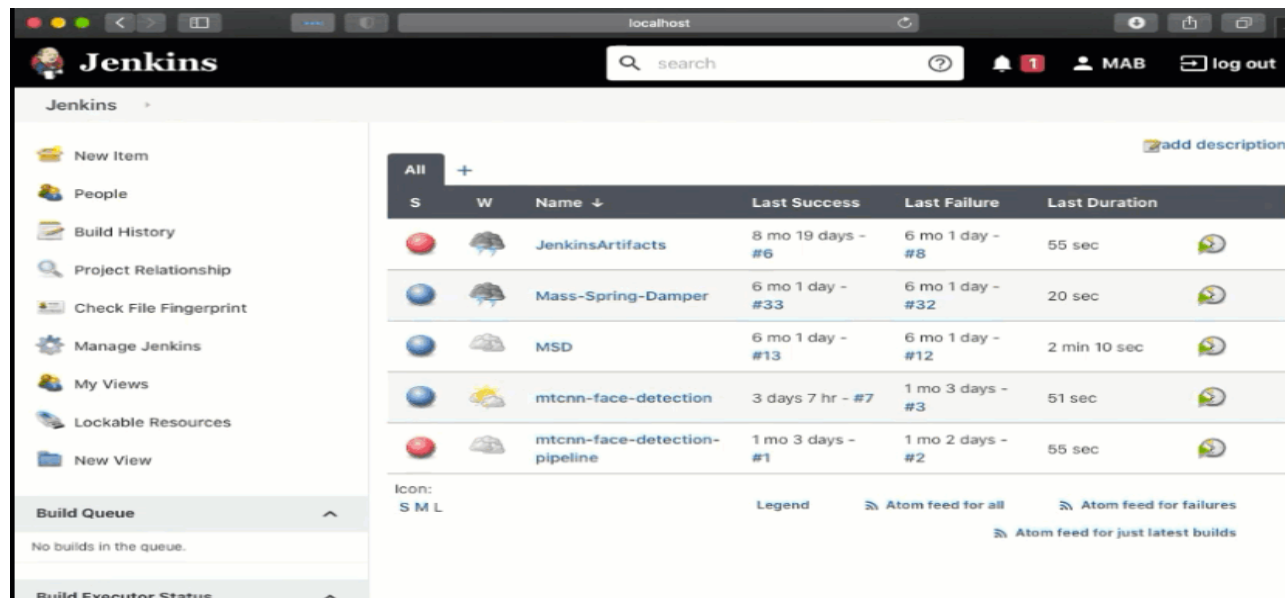
- Dans ce cours, nous allons nous intéresser à :
 - ✓ Virtual Box / Vagrant / Ubuntu
 - ✓ Jenkins
 - ✓ Docker
 - ✓ Maven
 - ✓ JUnit
 - ✓ Git (Github/GitLab)
 - ✓ Sonar
 - ✓ Nexus
 - ✓ Grafana / Prometheus
- Ces outils seront appliqués à deux projets Spring Boot et Angular déjà existants, que nous allons enrichir.

Outils

Jenkins



- Jenkins est un outil logiciel open source d'intégration continue
- A chaque modification de code d'une application dans le gestionnaire de configuration, Jenkins se charge automatiquement de la recompiler, et de la tester

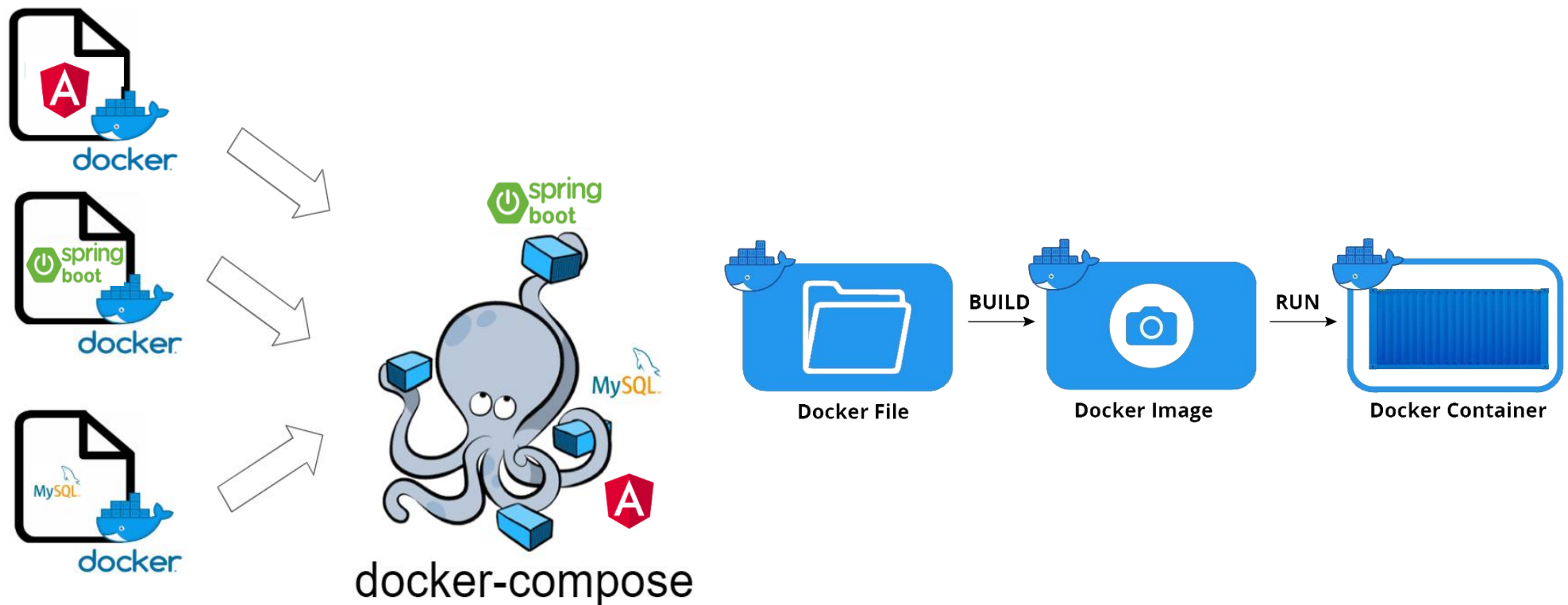


Outils

Docker



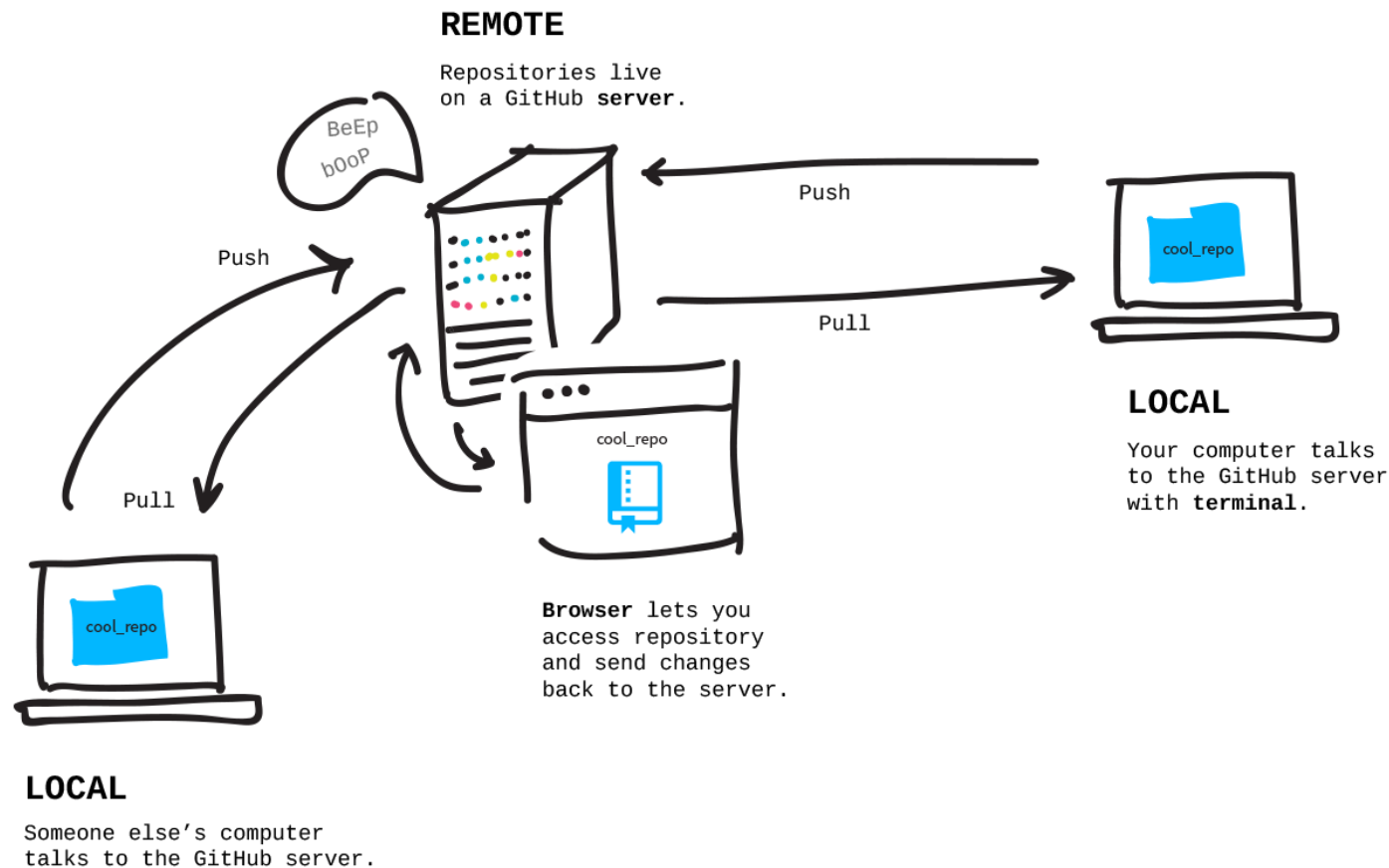
Docker est un outil qui peut emballer une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur



Outils

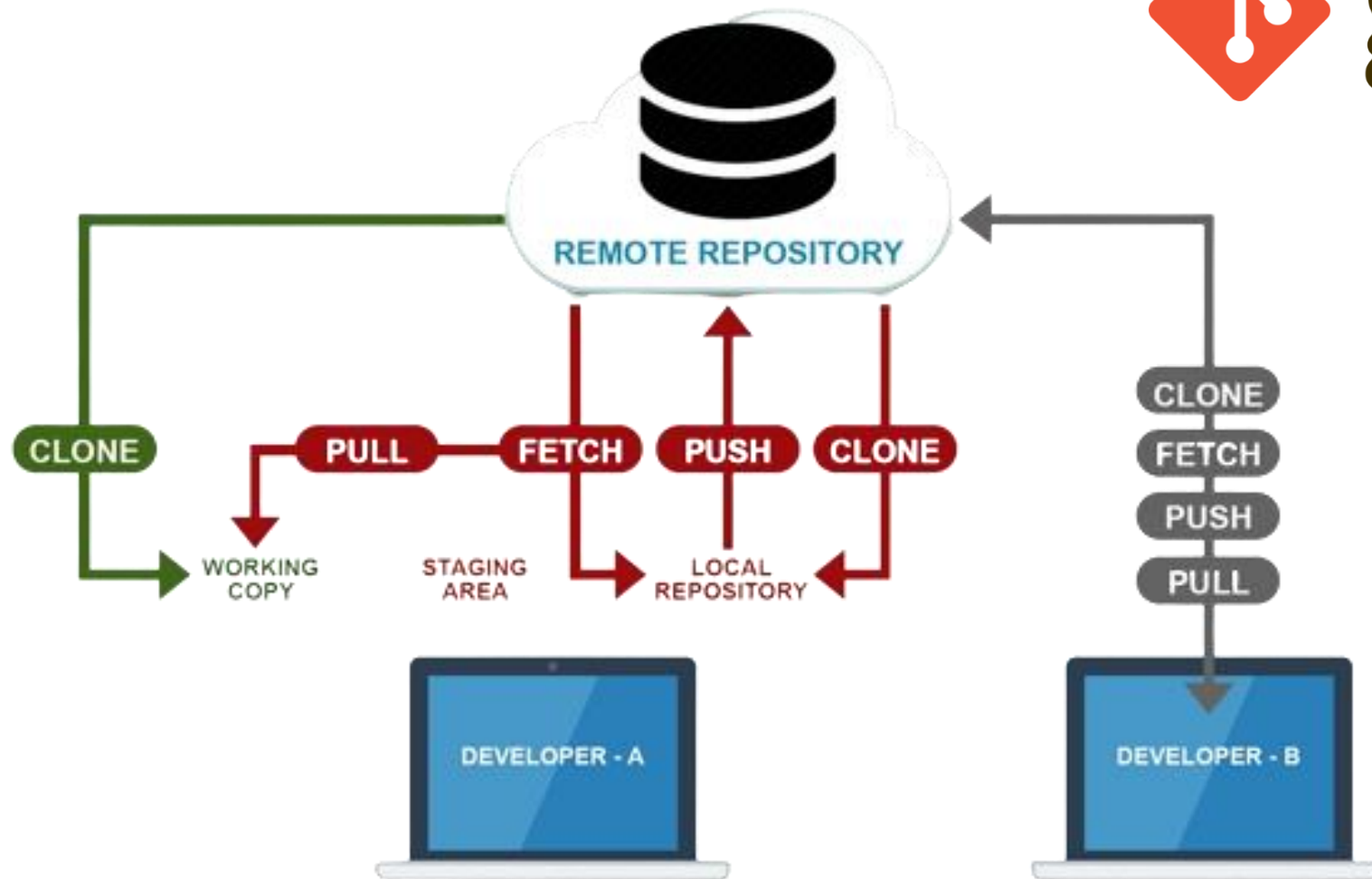
GIT

Git est un logiciel de gestion de versions décentralisé.



Outils

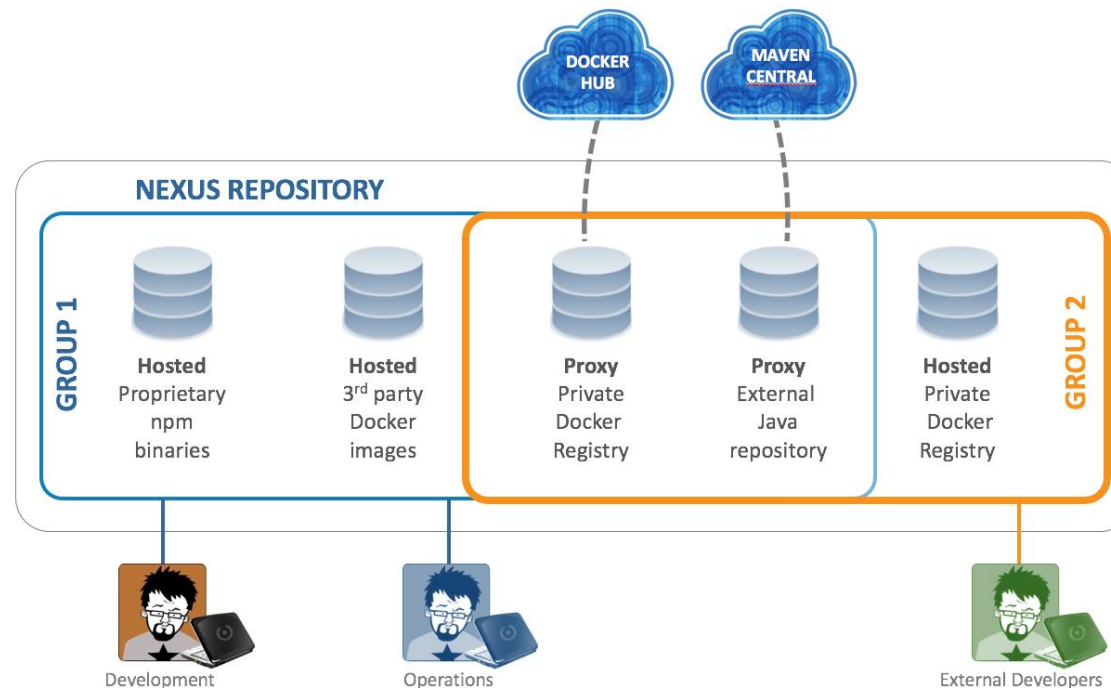
GIT



Outils

Nexus

Nexus est un gestionnaire de référentiel qui organise, stocke et distribue les artefacts nécessaires au développement



Outils

JUnit



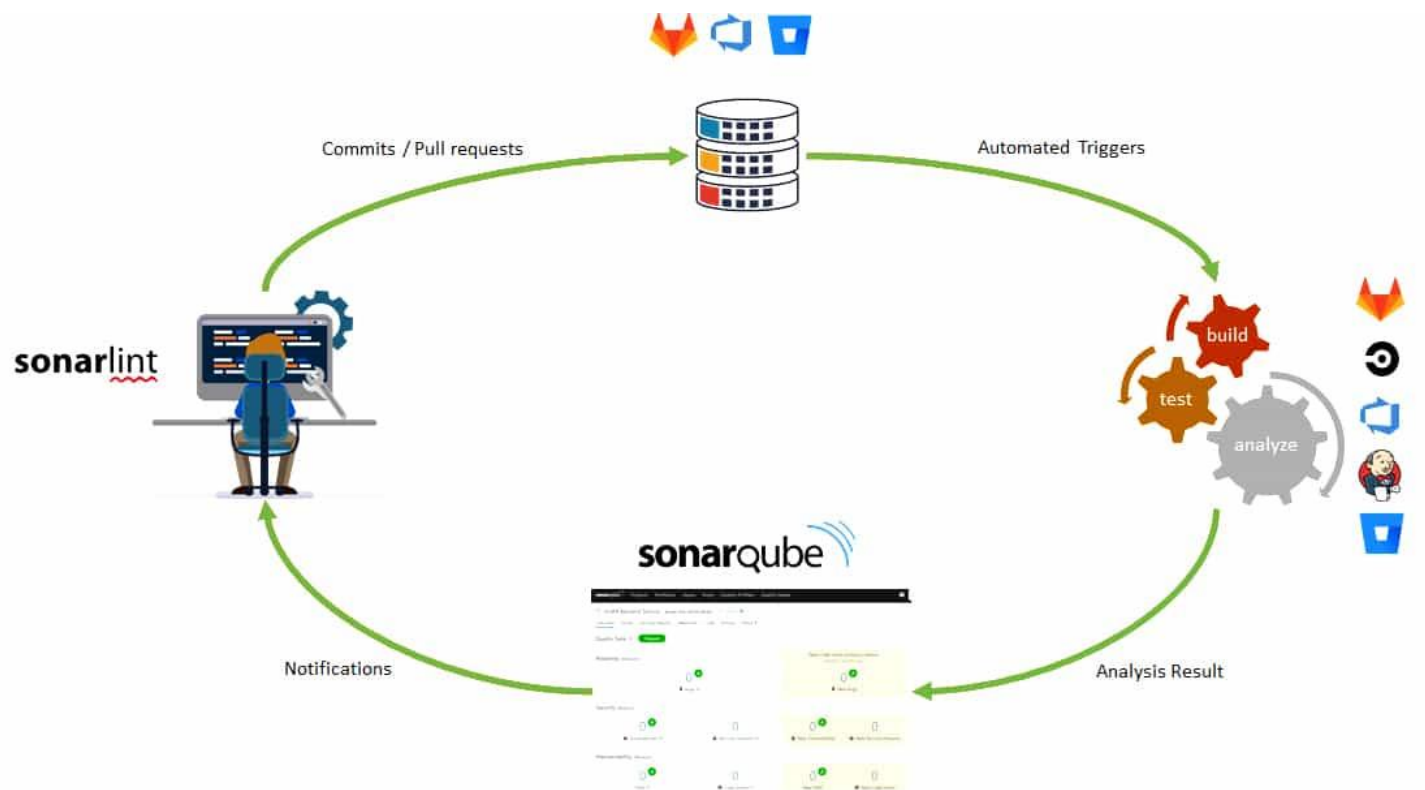
JUnit est un framework de test unitaire pour le langage de programmation java.



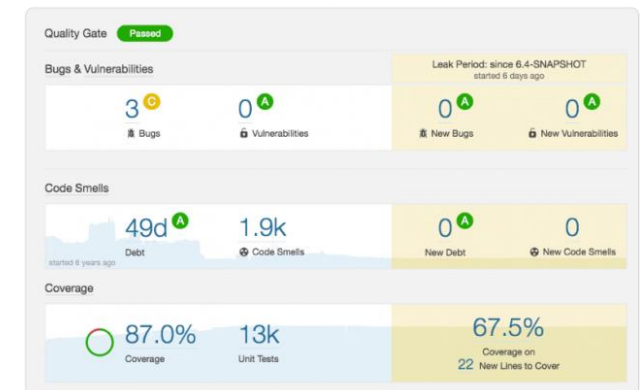
Outils

Sonar

SonarQube est un logiciel libre permettant de mesurer la qualité du code source en continu (Revue de code automatique).



sonarqube



Outils

Maven

C'est un outil de **construction de projets (build)** open source développé par la fondation Apache.



```
[INFO] --- maven-surefire-plugin:3.0.0-M7:test (default-test) @ springboot ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running me.fabriciorby.springboot.SpringBootDemoApplicationTests
[INFO] Running me.fabriciorby.springboot.SpringBootDemoApplicationTests$InnerTest
[WARNING] Tests run: 6, Failures: 0, Errors: 0, Skipped: 1, Time elapsed: 0.02 s - in me.fabriciorby.springboot.SpringBootDemoApplicationTests$InnerTest
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.336 s - in me.fabriciorby.springboot.SpringBootDemoApplicationTests
[INFO]
[INFO] Results:
[INFO]
[WARNING] Tests run: 6, Failures: 0, Errors: 0, Skipped: 1
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.814 s
[INFO] Finished at: 2022-07-14T11:35:15+02:00
[INFO] -----
```

Outils

Grafana / Prometheus

- Grafana est un logiciel libre sous licence GNU Affero General Public License qui permet la visualisation de données. Il permet de réaliser des tableaux de bord et des graphiques depuis plusieurs sources dont des bases de données temporelles comme Graphite, InfluxDB et OpenTSDB.



- Prometheus est un logiciel libre de surveillance informatique et générateur d'alertes. Il enregistre des métriques en temps réel dans une base de données de séries temporelles en se basant sur le contenu de point d'entrée exposé à l'aide du protocole HTTP.



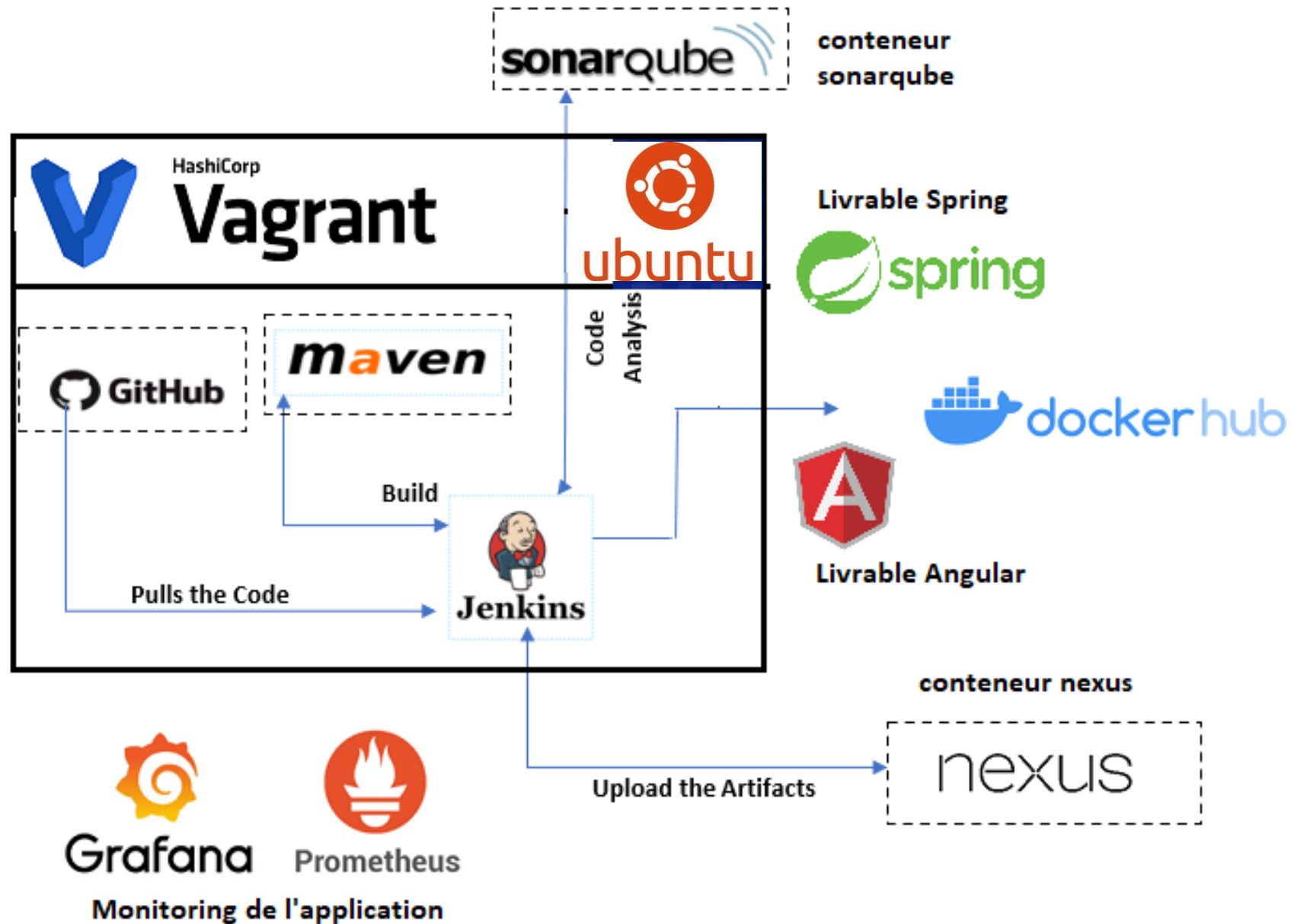
Outils

Grafana / Prometheus

- L'exemple suivant montre un tableau de bord Grafana qui interroge Prometheus pour obtenir des données :





Solution finale

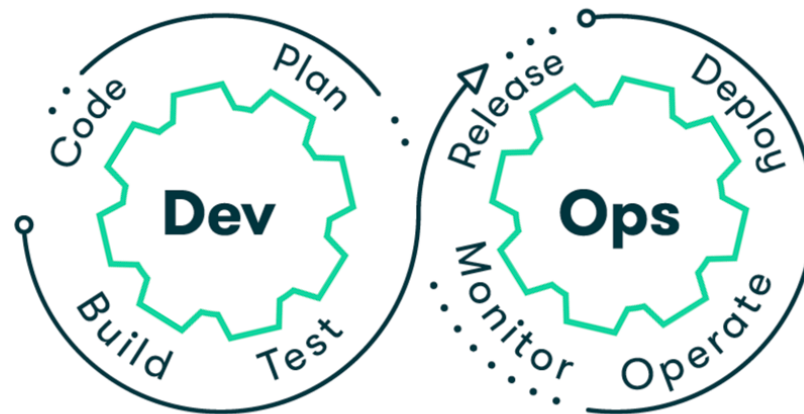
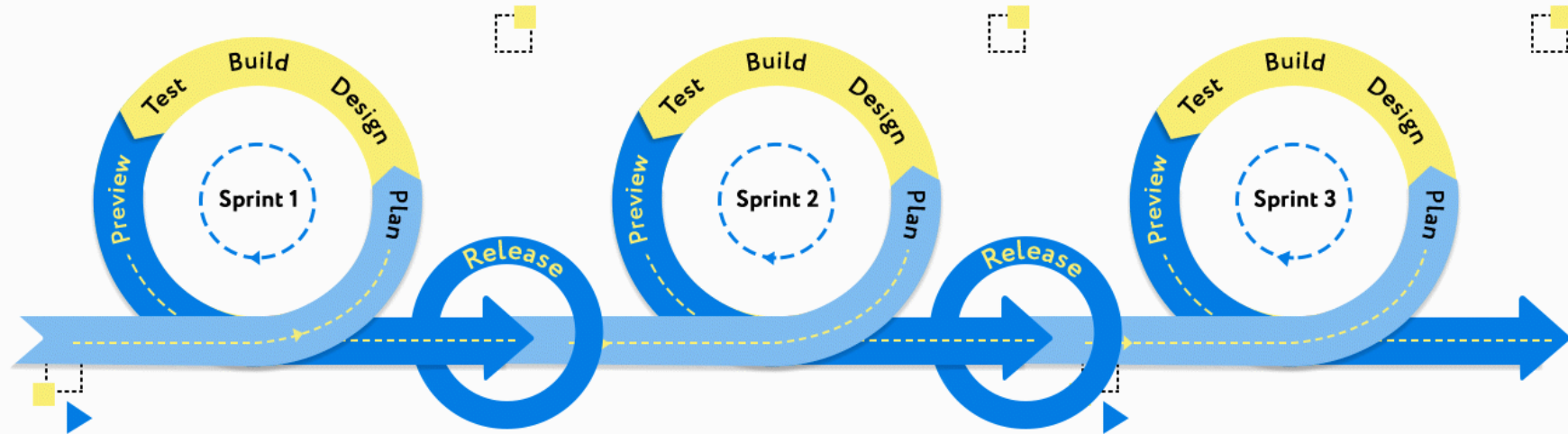


Installation des outils

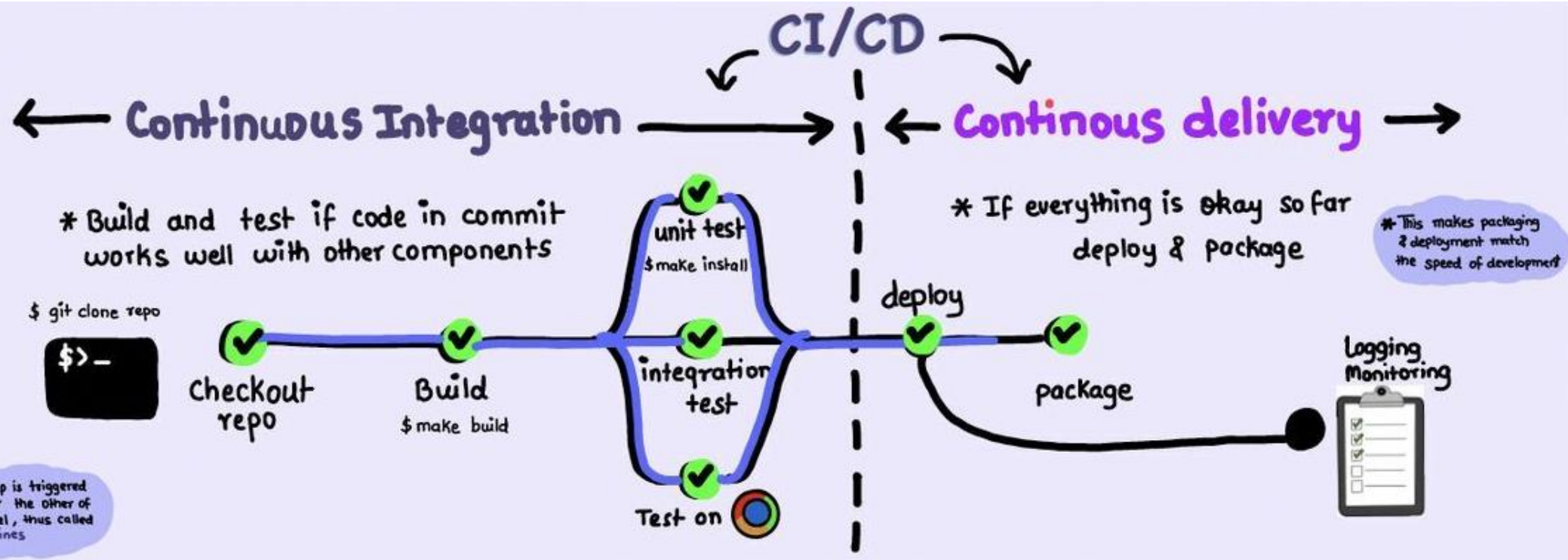
Pour la prochaine séance:

- ✓ Installer en suivant le tuto que nous vous enverrons :
 - Virtual box  **VirtualBox**
 - Vagrant  **VAGRANT**
 - Une machine virtuelle Ubuntu dans Vagrant

Résumé



Résumé



Introduction DevOps

Si vous avez des questions, n'hésitez pas à nous contacter :

Département Informatique
UP Architectures des Systèmes d'Information

Bureau E204

Introduction DevOps

