

esprit
Se former autrement

HONORIS
UNITED UNIVERSITIES



Introduction DevOps

Ghassen Hammouda

Enseignant formateur à Esprit



Plan module DevOps

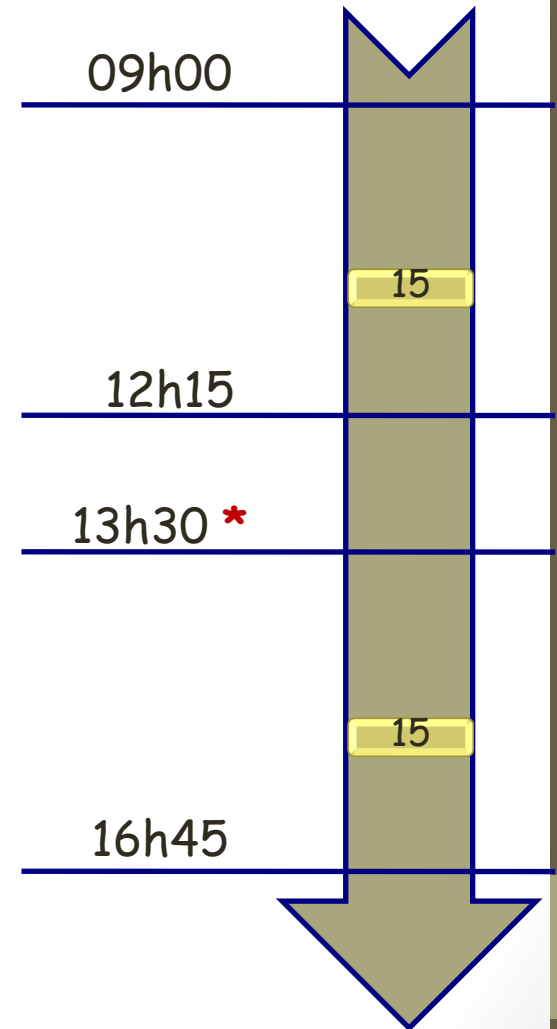
- Introduction DevOps (installation Virtual Box / Vagrant / CentOS)
- GIT (gestionnaire du code source)
- Maven (construction du projet)
- Jenkins (serveur d'intégration continu)
- Docker (plateforme de lancement de l'application sur container)
- Nexus (gestion des livrables)
- JUnit (Les tests unitaires mockData)
- Sonar (qualité de code)
- Docker Avancé (docker compose, docker volume)
- Grafana/ Promethuis
- Validation projet

Plan du cours

- Introduction (plan module devOps, horaire, évaluation, définition devOps)
- Evolution des Méthodologies
- Evolution des architectures
- Dev vs. Ops
- Dev && Ops
- Continuous Integration/ Continuous Deployment (or Delivery) (CI/CD)
- Technologies DevOps
- Pratiques DevOps
- Culture DevOps
- Avantages DevOps
- Outils
- Solution Finale

Horaires

- Durée Totale : **30 heures**
- Séances : **10 séances**
- Cours : **9 heures**
- TP : **15 heures**
- Examen : **6 heures**
- Durée de chaque Séance : **3 heures**
(2h synchrone + 1h 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 pratique tiens 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).
- Projet de 5/6 personnes par groupe.

Introduction



DevOps?

Introduction



Qui utilise DevOps?

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!)
- Nous devons refaire les taches de ces 3 derniers jours !

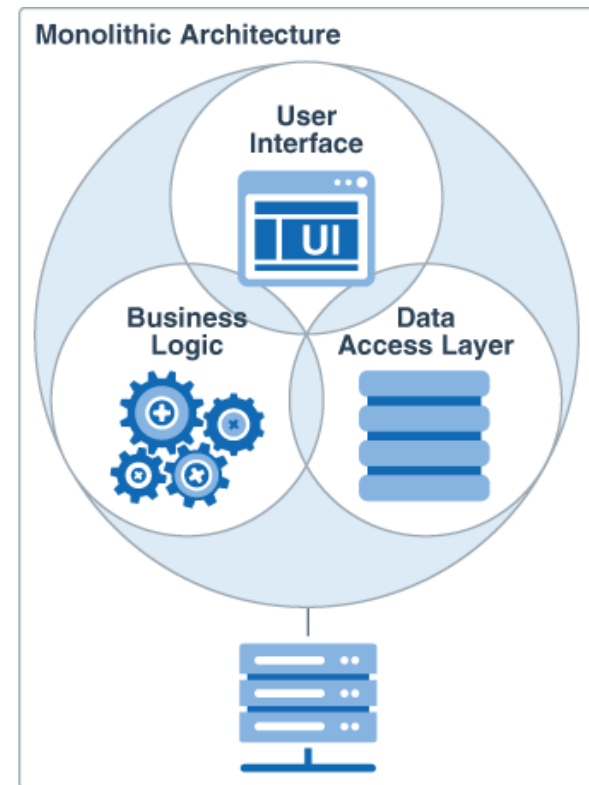
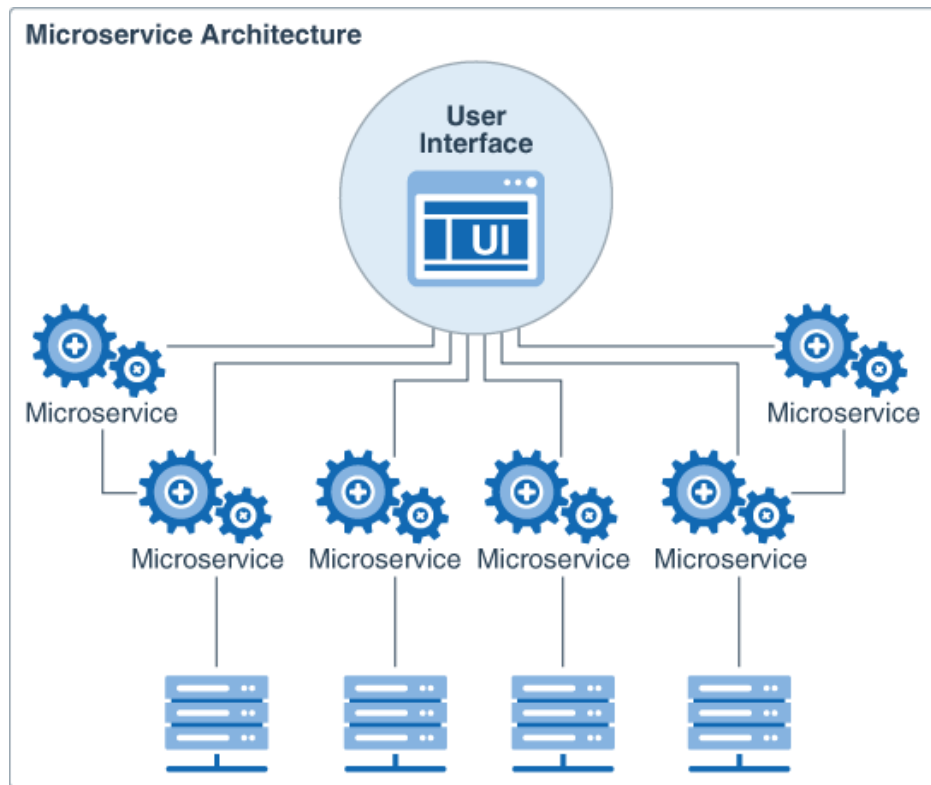


Evolution des Méthodologies

- Méthodologie Classique / Lourde (Méthode RUP, 2TUP, ...) => Méthodologie **Agile** (Méthode Scrum, XP, ...)
- Une méthode agile est un ensemble de pratiques de pilotage et de réalisation de projets, qui met en avant la communication entre les membres de l'équipe de Développement, la communication avec le client, l'adaptation au changement, et s'affranchit (se libère) des outils et des process lourds.
- Il s'agit de travailler en mode itératif (Sprint). Chaque Sprint peut être considéré comme un projet dont le Cycle de vie est en V.

Evolution des Architectures

- Architecture Monolithique => Architecture en **Microservices**



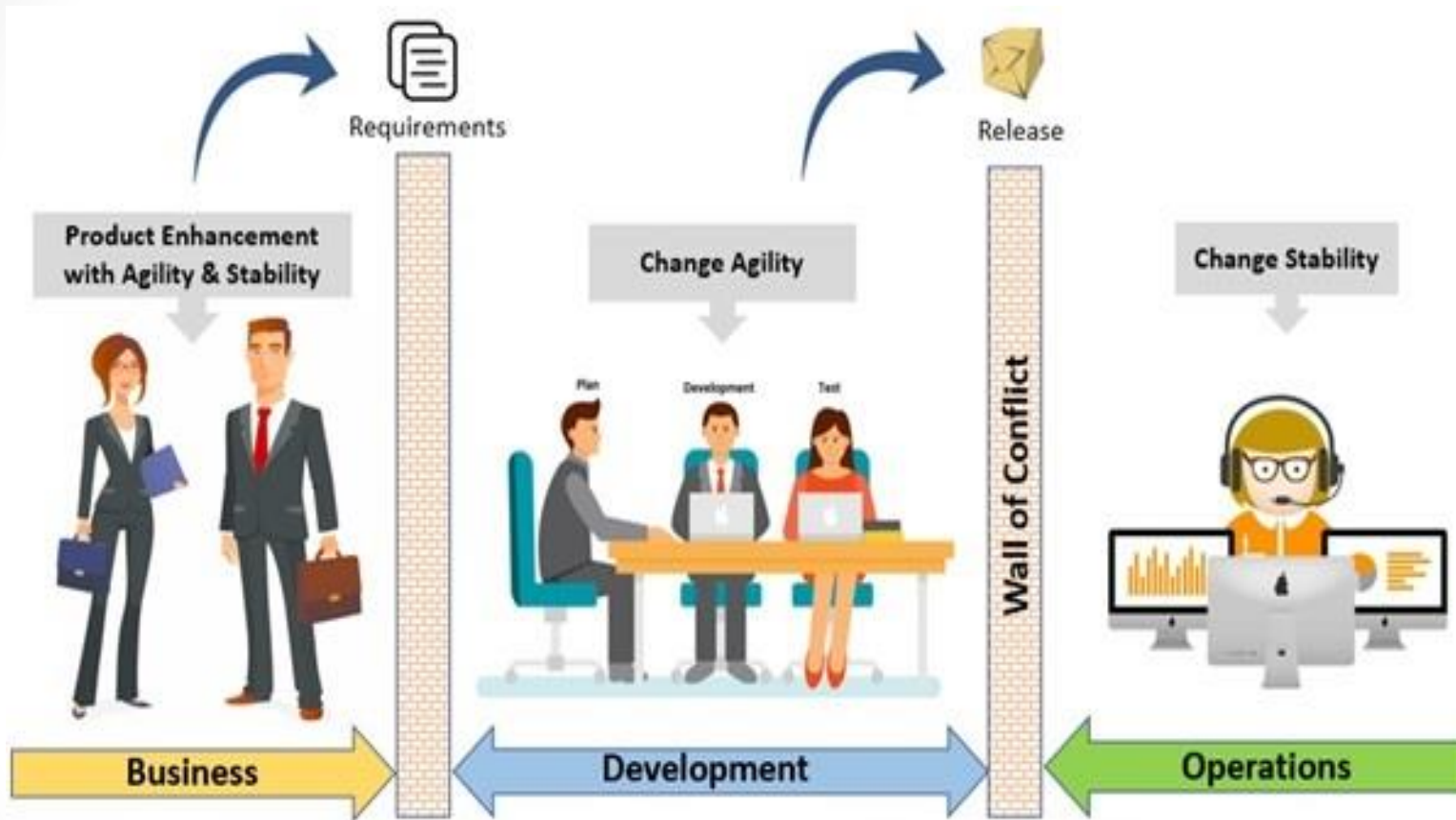
Apport du DevOps

- Utiliser une méthode **Agile**, et une architecture en **Microservice** résout énormément de problèmes. **Mais ...**
- Comment faire travailler étroitement les équipes de production avec les équipes de développement?
- Comment automatiser au maximum les différentes phases du Projet ?
- Comment pouvoir livrer régulièrement et fréquemment (comment éviter les retards et les risques liés au déploiement) ?
- Comment diminuer la peur du changement (comment augmenter la confiance de l'équipe de Production en l'équipe de Développement) ?

Définition DevOps

- 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.

Dev vs. Ops



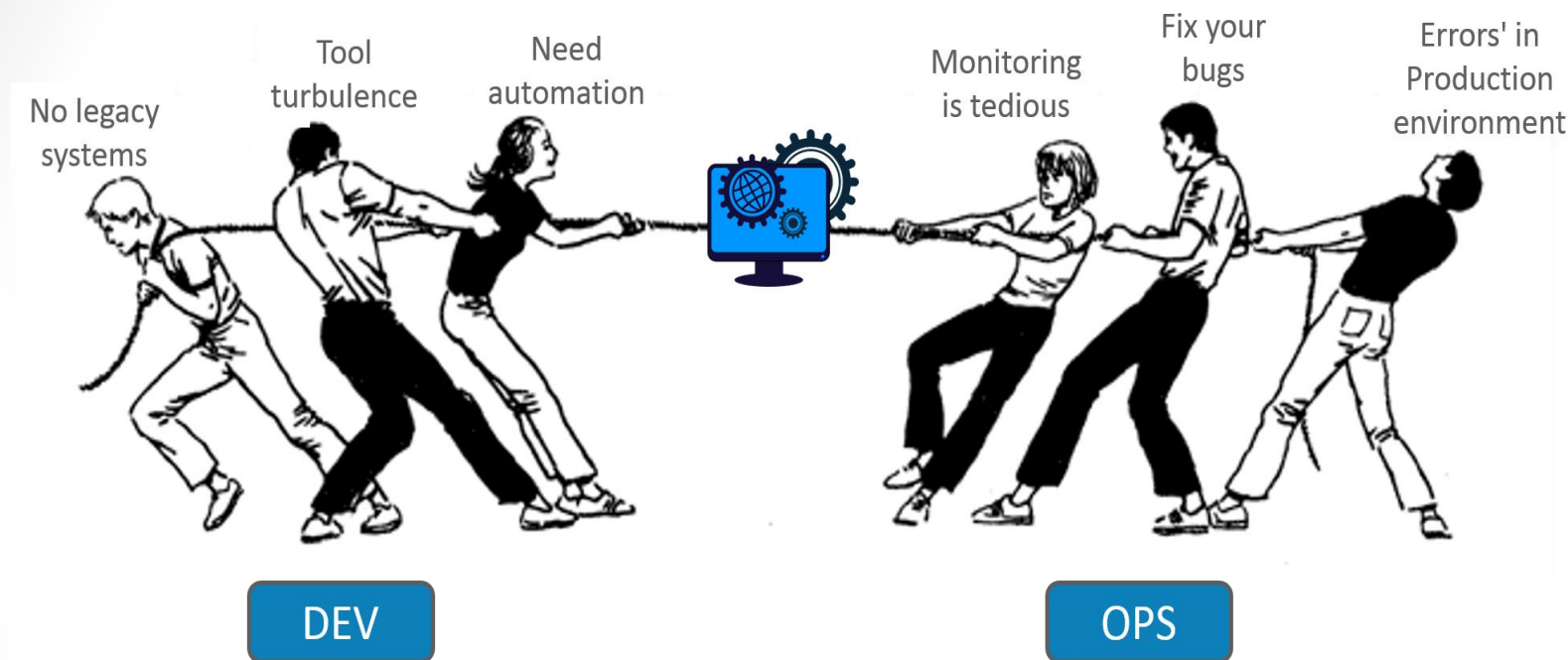
3 équipes sont indispensables pour la bonne conduite d'un projet :
équipe business, équipe développement et équipe opérationnel.
L'interaction entre ces 3 équipes est **loin d'être évidente**

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



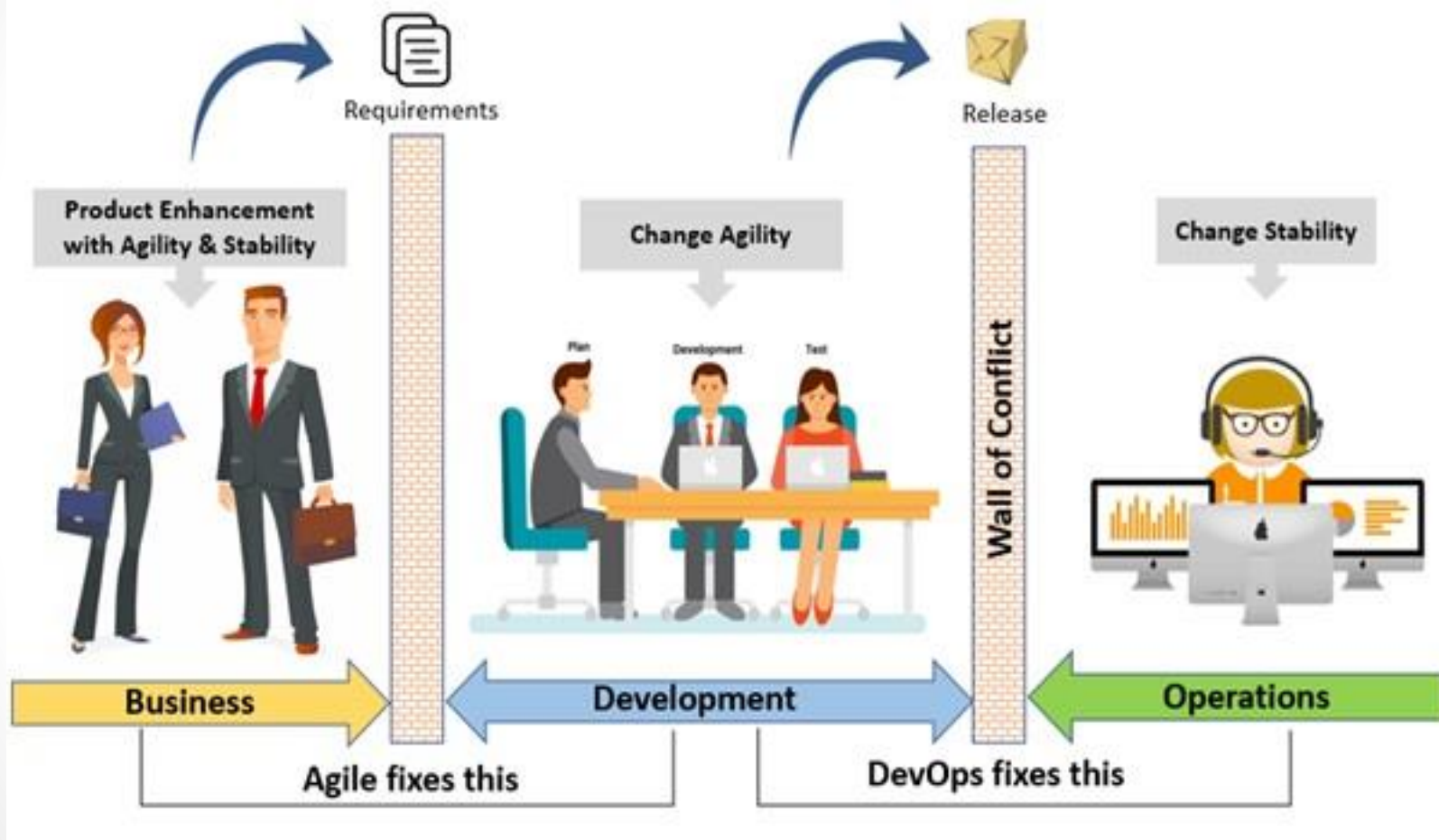
Dev vs. Ops



La culture DevOps joue le rôle de **médiateur** entre les deux équipes dev et ops.

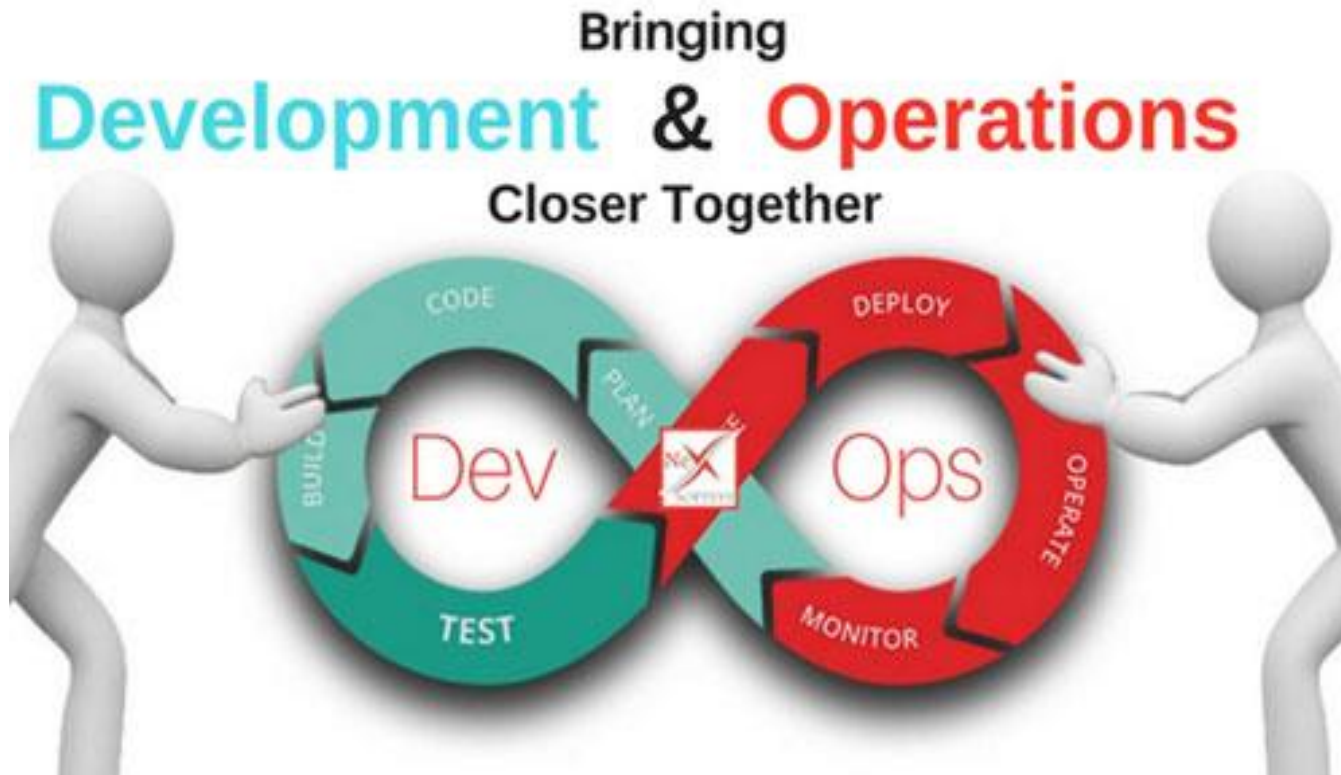
L'objectif est de transformer la tension entre les deux équipes en une **collaboration saine**.

Dev vs. Ops



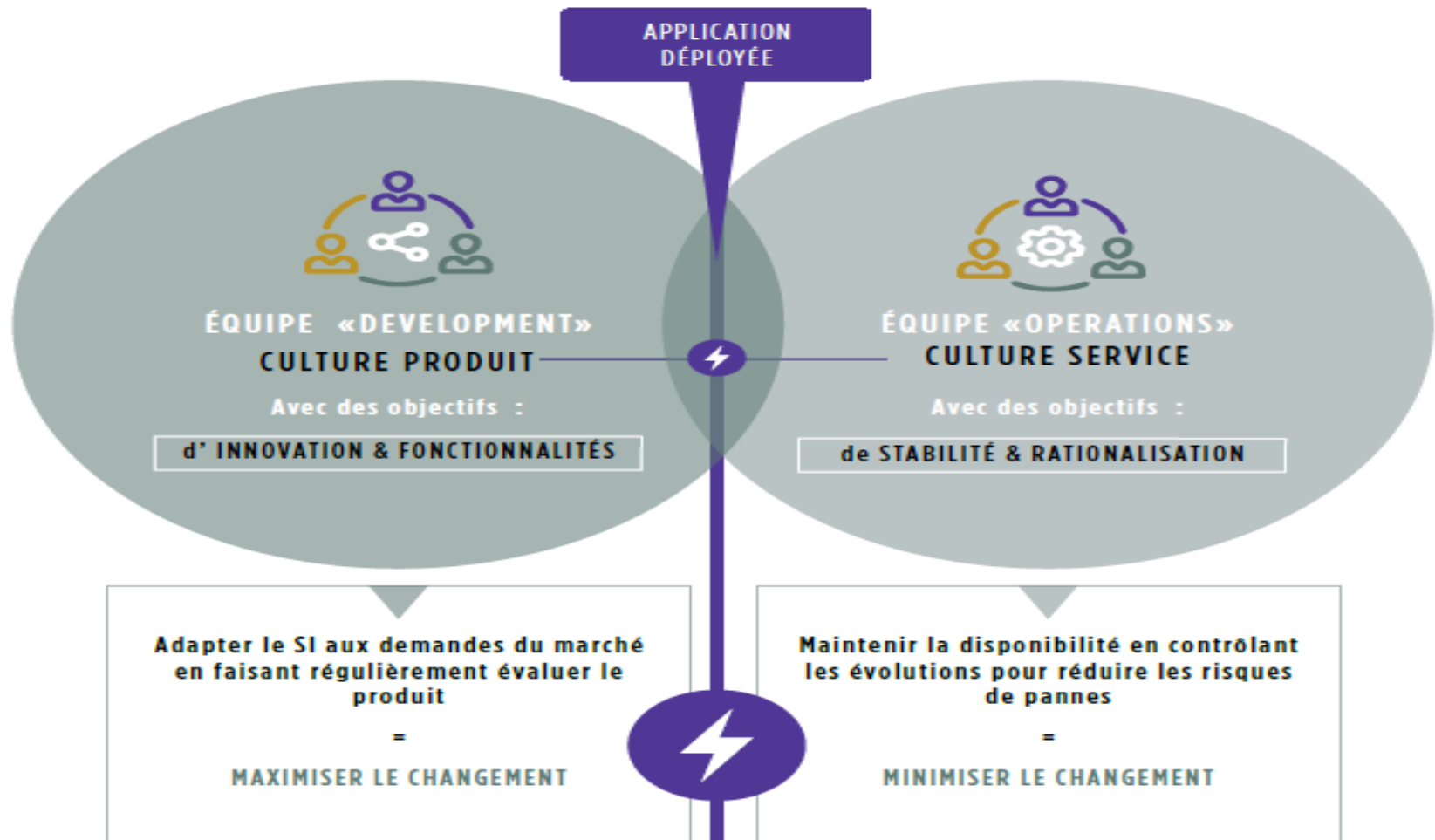
L'agilité et les pratiques DevOps interviennent pour **briser les frontières** entre les différents collaborateurs (business, dev, ops).

Dev & Ops



La solution parfaite à intérêt mutuelle entre les deux parties (dev et ops) est de coordonner les efforts pour que tous les objectifs des équipes soient réalisés avec moindre cout et dans le plus bref des délais.

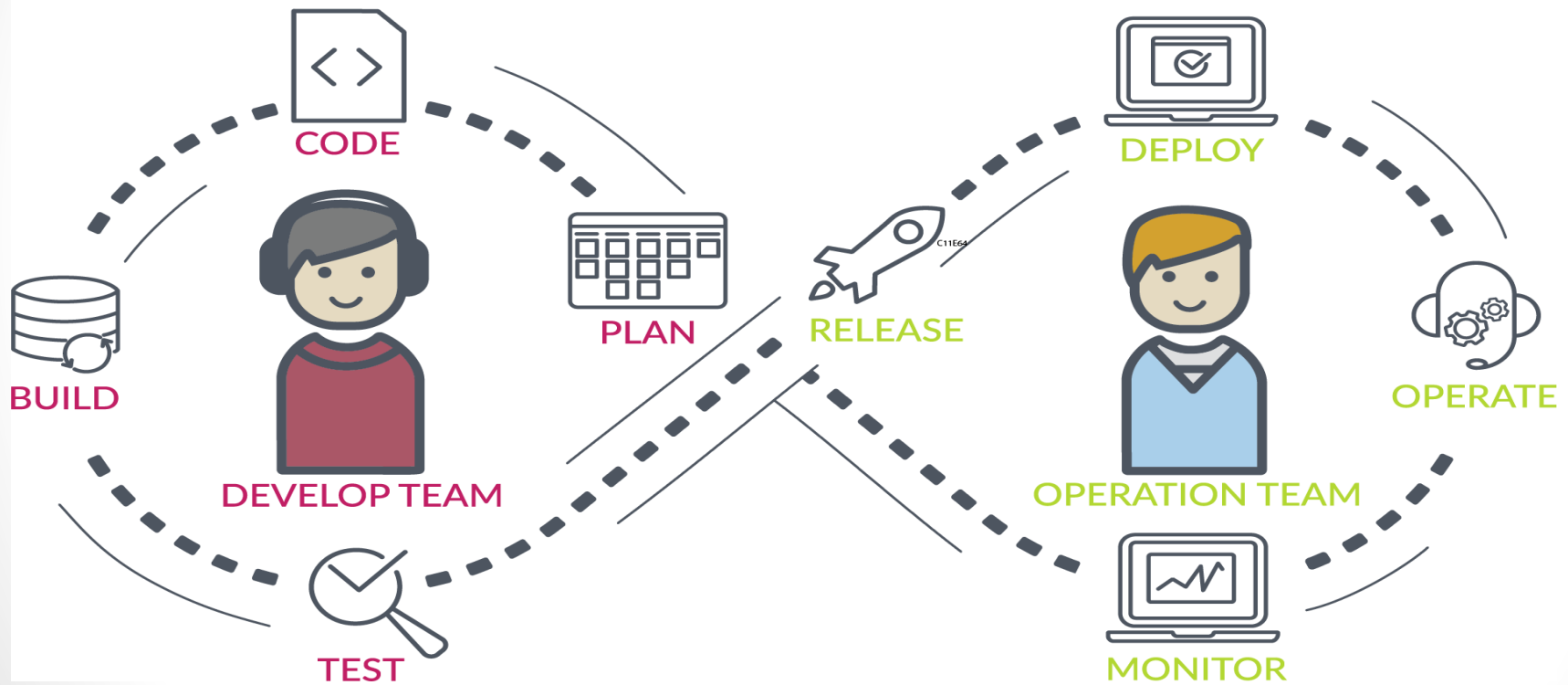
Dev & Ops



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

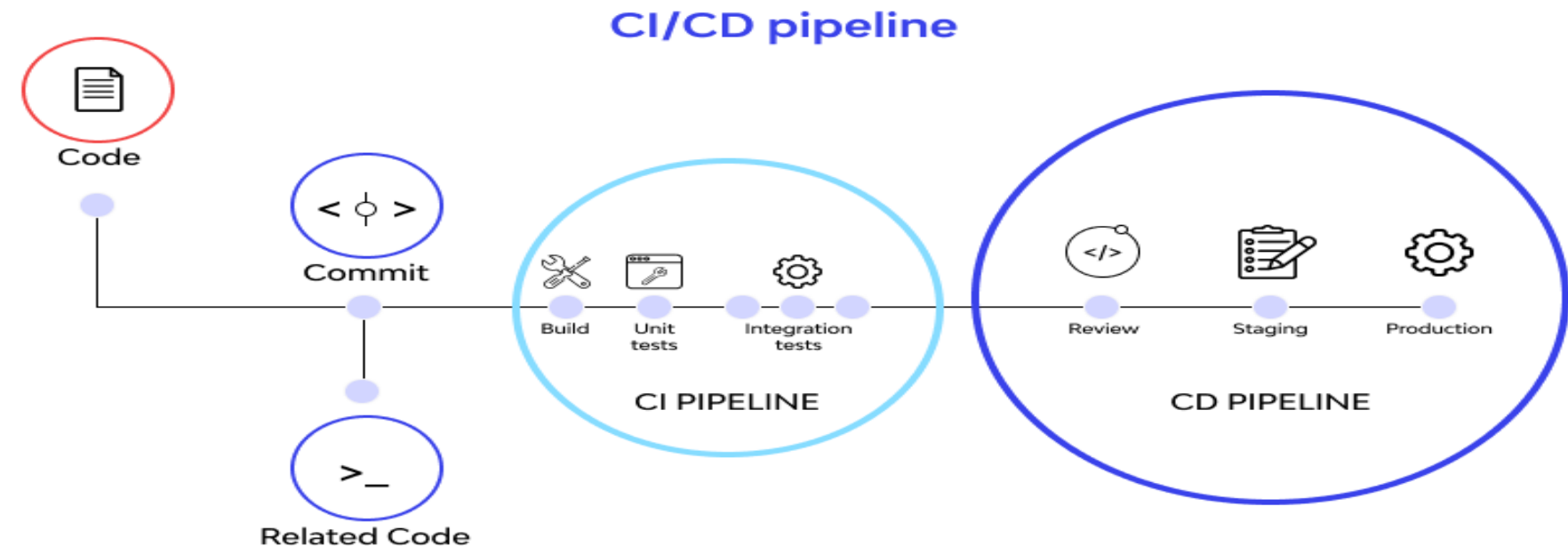


Création d'un pipeline automatisé entre les deux équipes appelé CI/CD (continuous integration/ continuous delivery (ou deployment)).

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 continue **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** 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.

CI/CD

Il existe 3 processus DevOps :

Intégration continu

Livraison continu

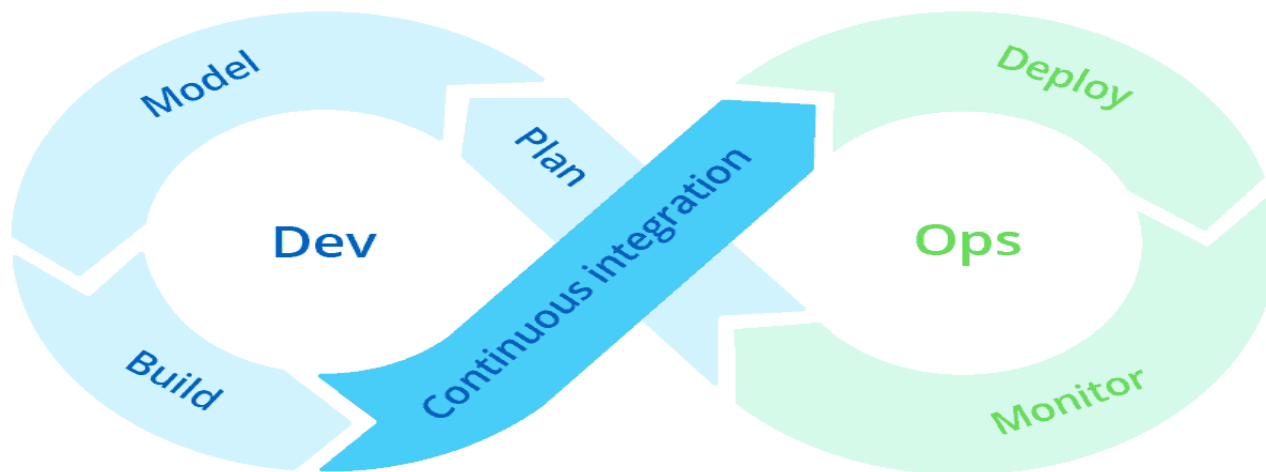
Déploiement continu

CI/CD

Intégration continu

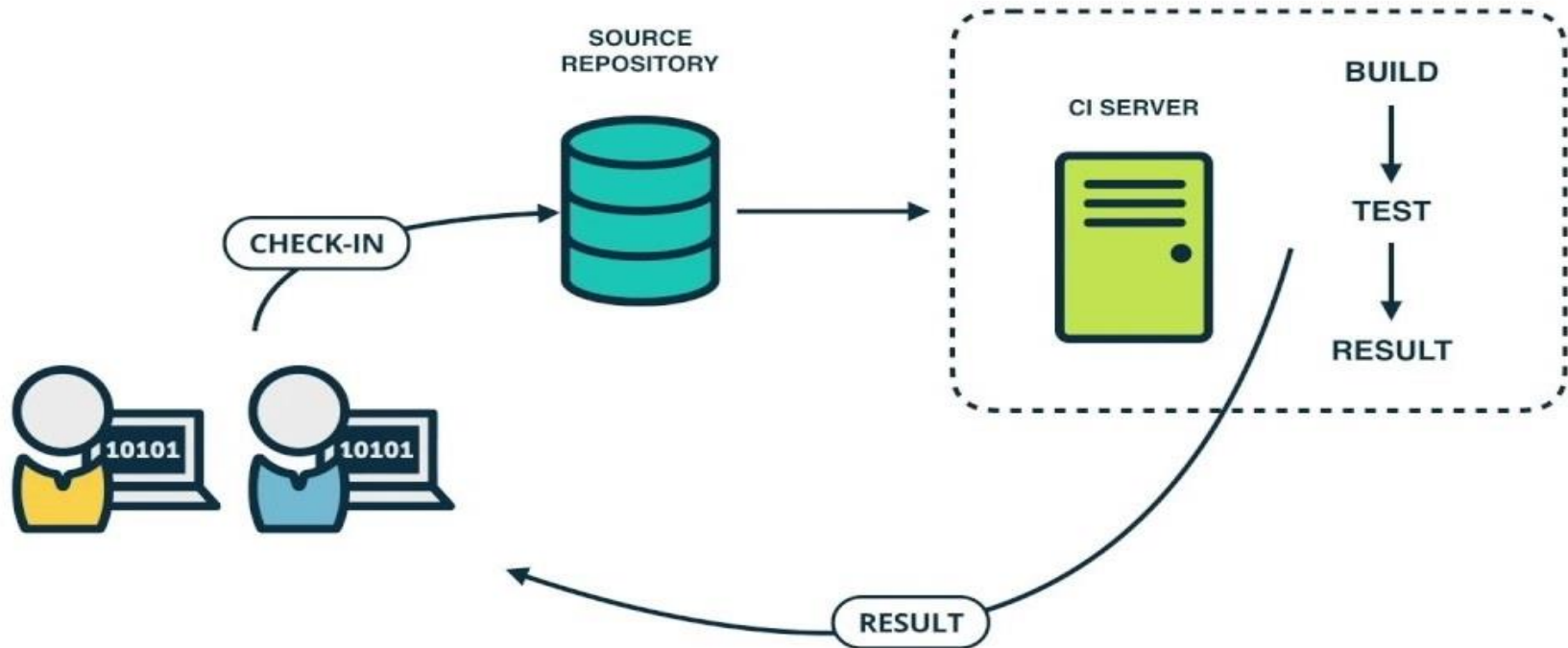
L'intégration continu est un processus consistant à compiler, tester et **déployer sur un environnement d'intégration**.

L'objectif est de détecter les régressions (bugs) du livrable à l'avance pour les fixer en s'appuyant sur des outils.



CI/CD

Intégration continu



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.

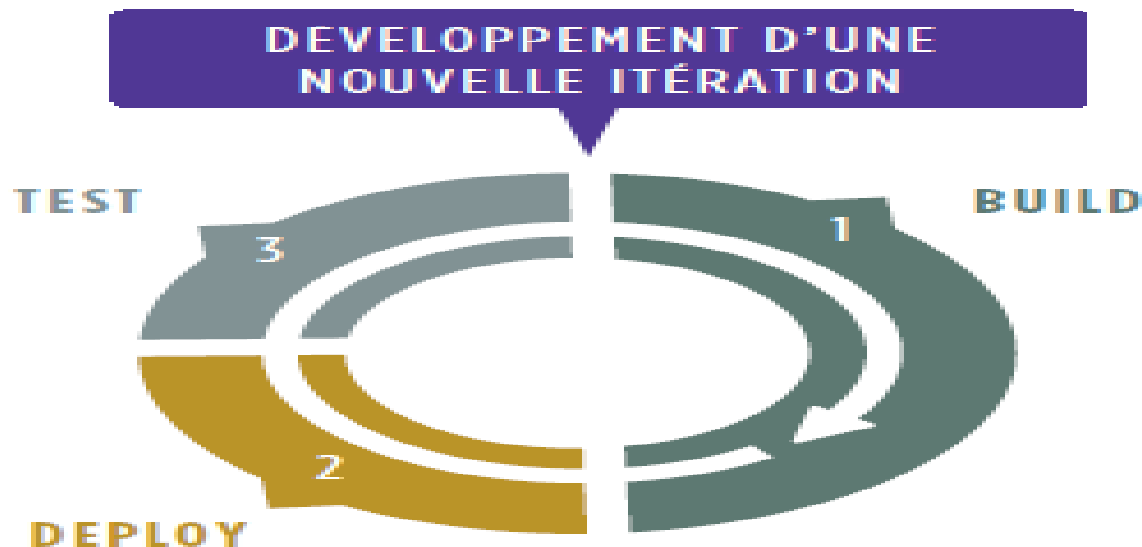
CI/CD

Livraison continu

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

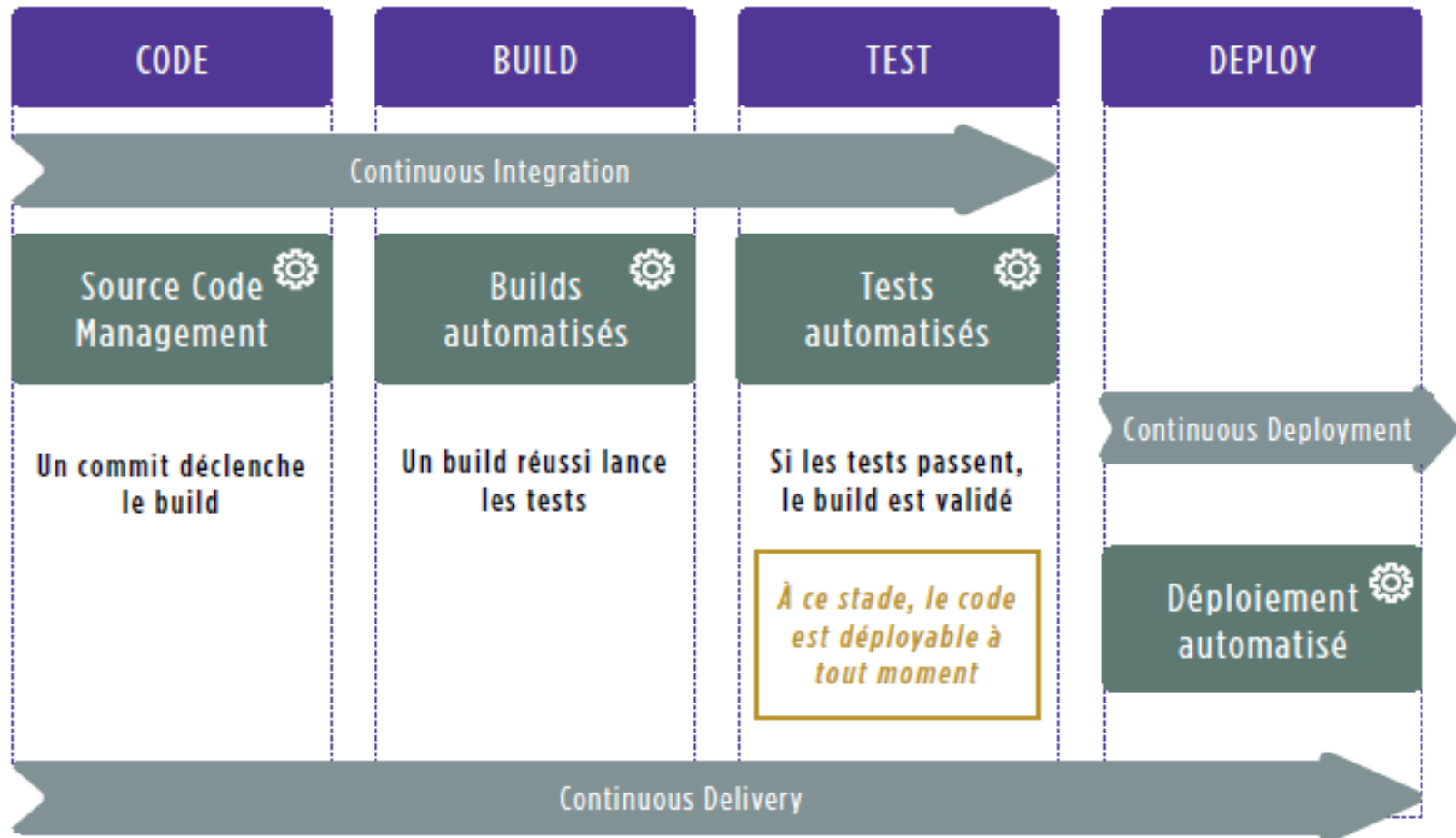
La **livraison continue** est une discipline où l'application est construite de manière à **pouvoir être mise en production à n'importe quel moment**.

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



CI/CD

Livraison continu



CI/CD

Déploiement continu

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

Le déploiement est réalisé par un simple **click** (dans la majorité des cas) ou automatique (avec un niveau de fiabilité maximal).

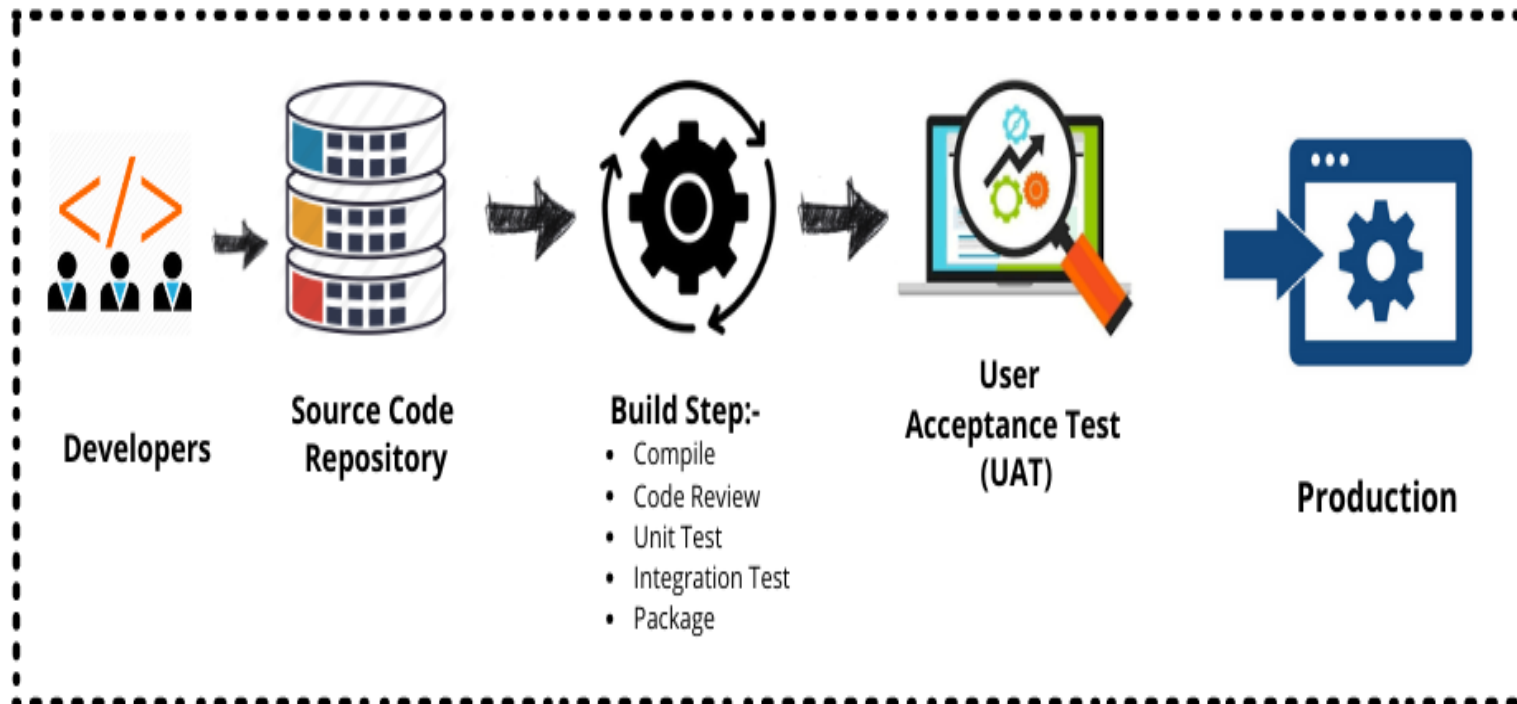
Il est primordiale de mesurer les impacts suite au déploiement grâce à des outils de supervision. En cas de détection de problèmes, un processus de retour en arrière doit être exécuté.



CI/CD

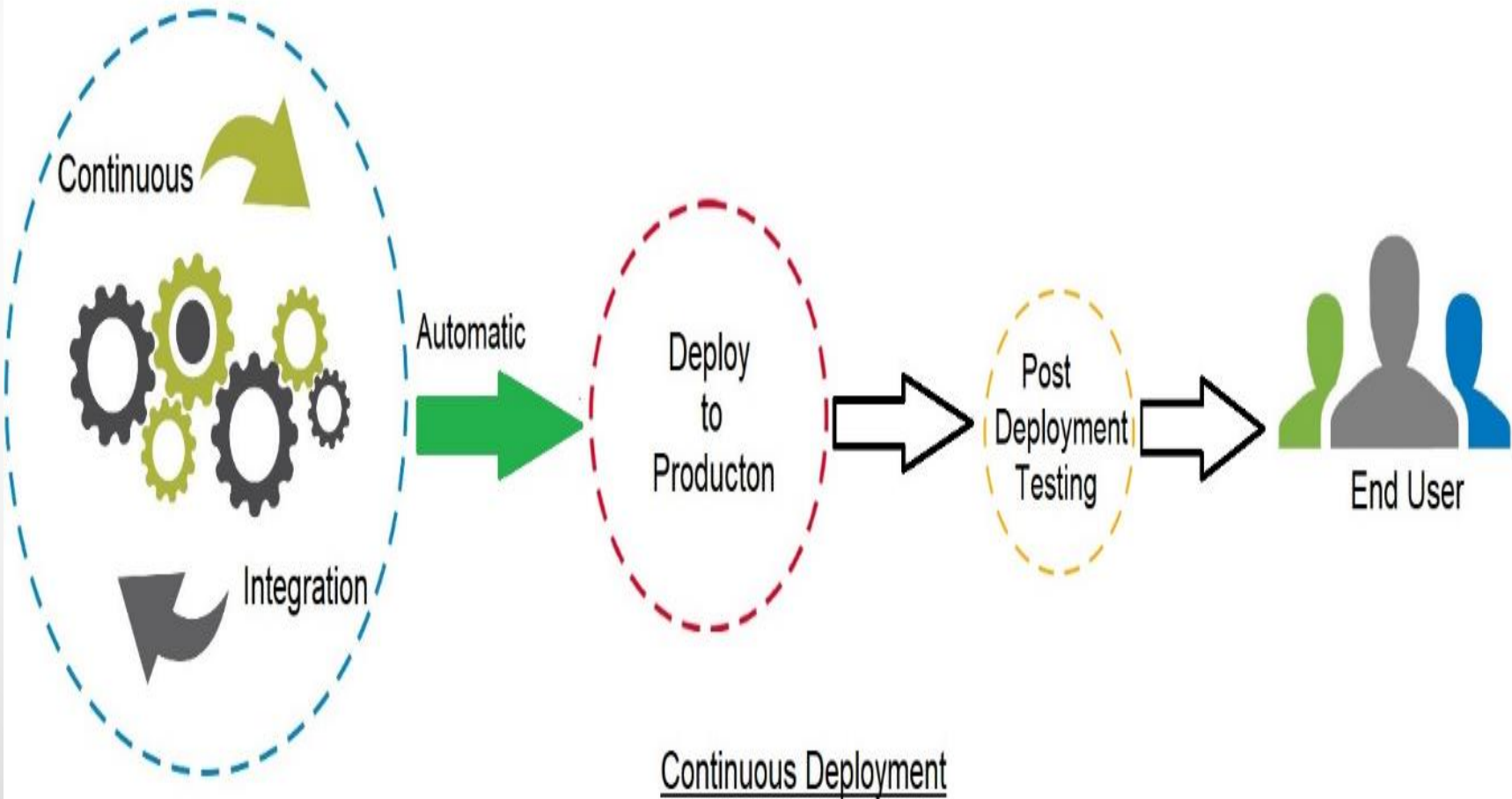
Déploiement continu

CONTINUOUS DEPLOYMENT



CI/CD

Déploiement continu



CI/CD

INTÉGRATION CONTINUE

DÉPLOIEMENT AUTOMATISÉ

TESTS D'ACCEPTATION AUTOMATISÉS

DÉPLOIEMENT CONTINU EN PRODUCTION

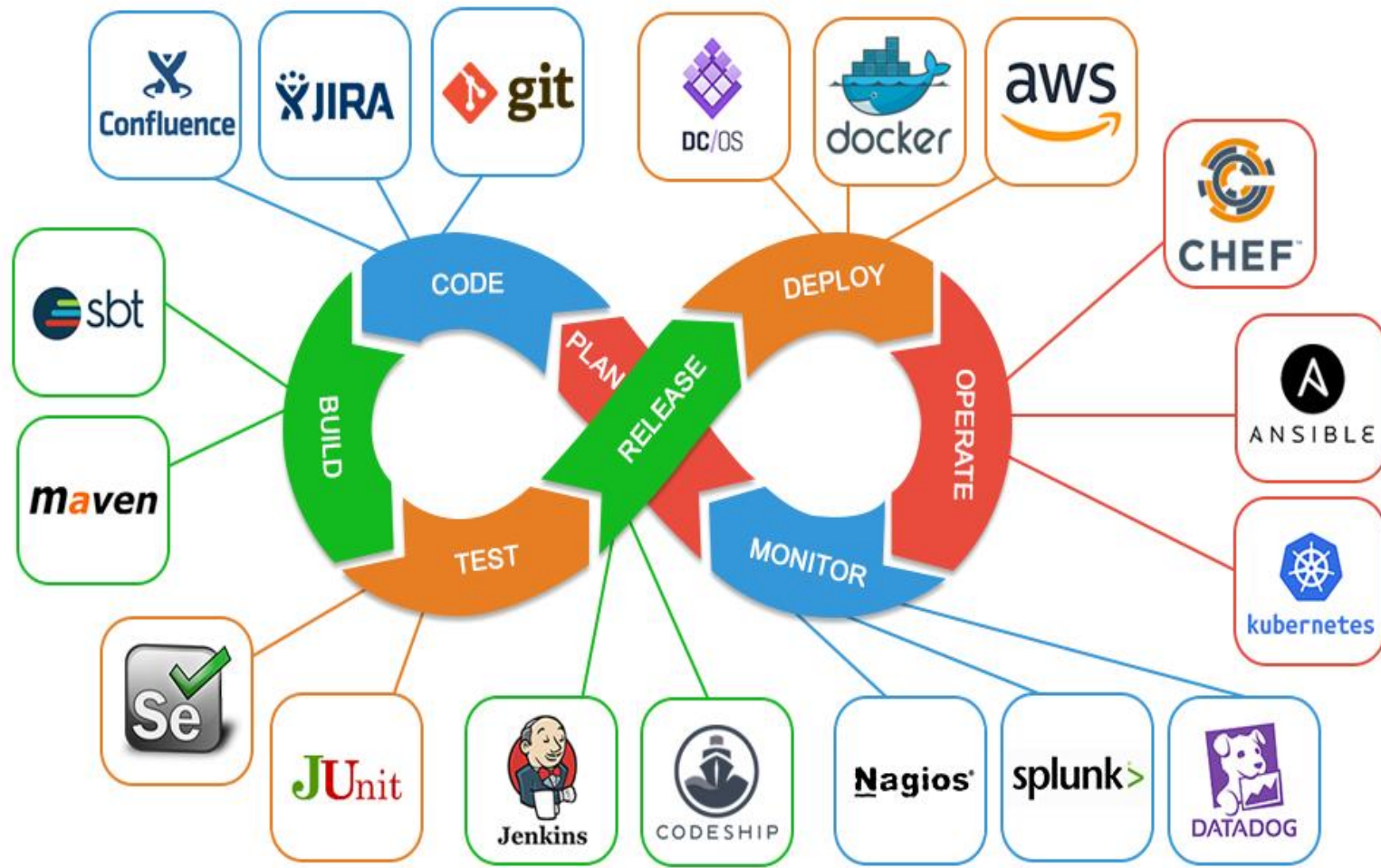
Développement

Intégration

Qualification

Production

Technologies DevOps



DevOps : pipeline automatisé d'outils

Pratiques devOps

Pratique 1: Travailler en mode collaboratif

Pratique 2: L'intégration continue

Pratique 3: L'automatisation des tests

Pratique 4: La livraison continue

Pratique 5: Gestion de la configuration automatisée

Pratique 6: La planification continu

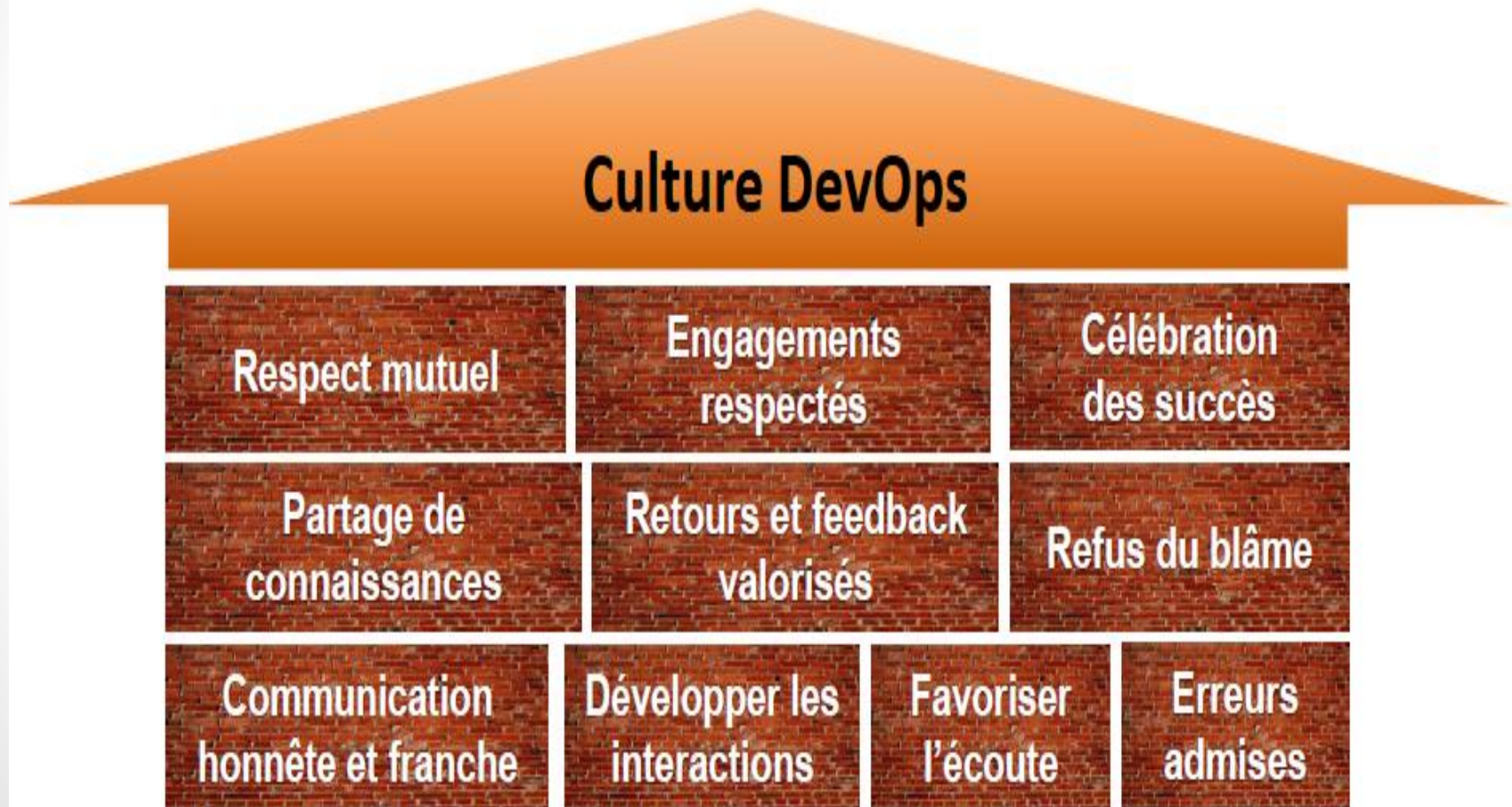
Pratique 7: La gestion des demandes et des RFC (request for change)

Pratique 8: Le cycle de vie d'une application

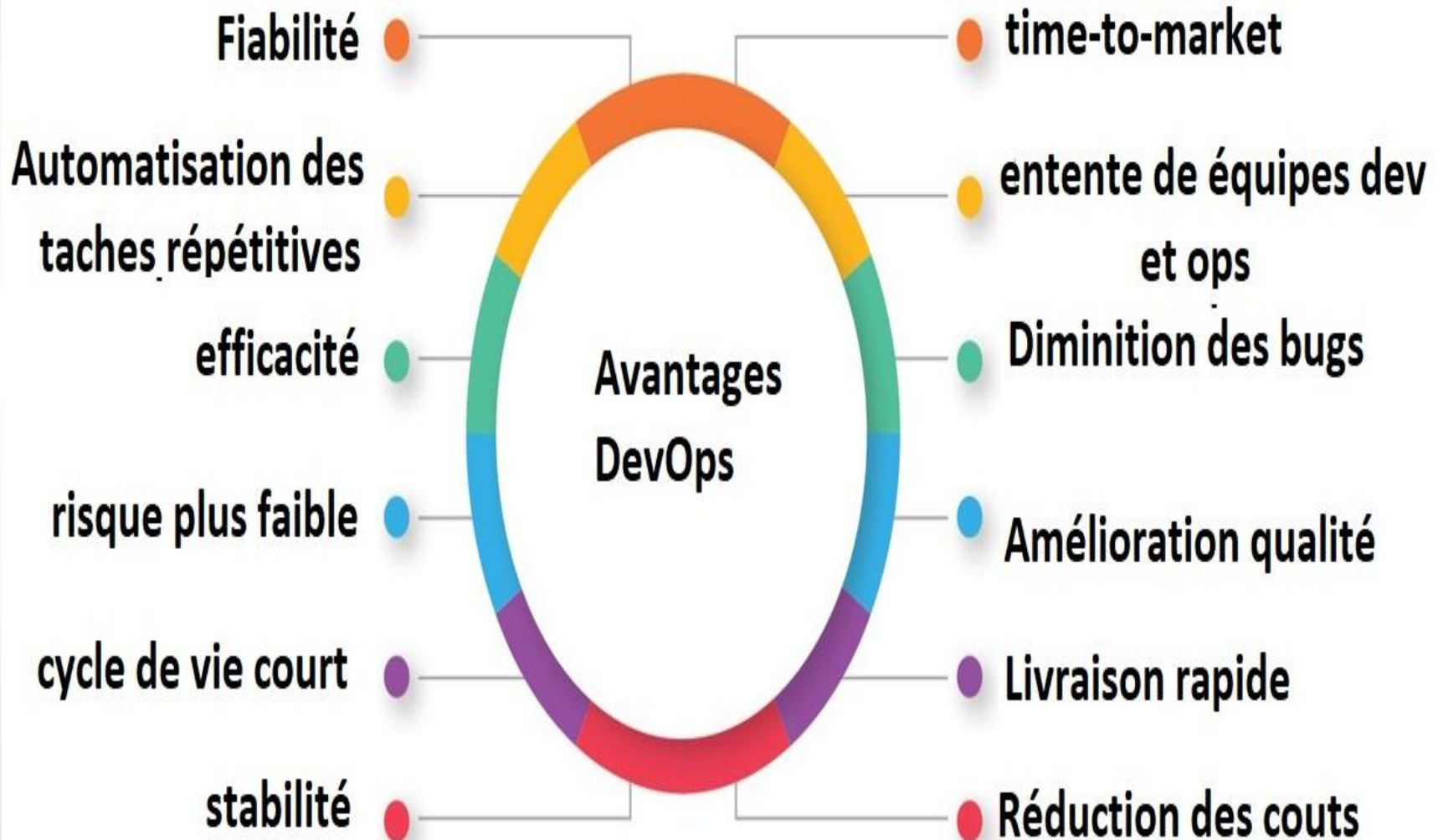
Pratique 9: La gestion des domaines applicatifs

Pratique 10: Tableaux de bord automatisés

Culture devOps



Avantages devOps



Outils



Maven

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

```
[INFO] -----  
[INFO] Building tp2Maven 1.0  
[INFO] -----  
[INFO]  
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ tp2Maven ---  
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e.  
[INFO] Copying 0 resource  
[INFO]  
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ tp2Maven ---  
[INFO] Nothing to compile - all classes are up to date  
[INFO]  
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ tp2Mav  
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e.  
[INFO] Copying 0 resource
```

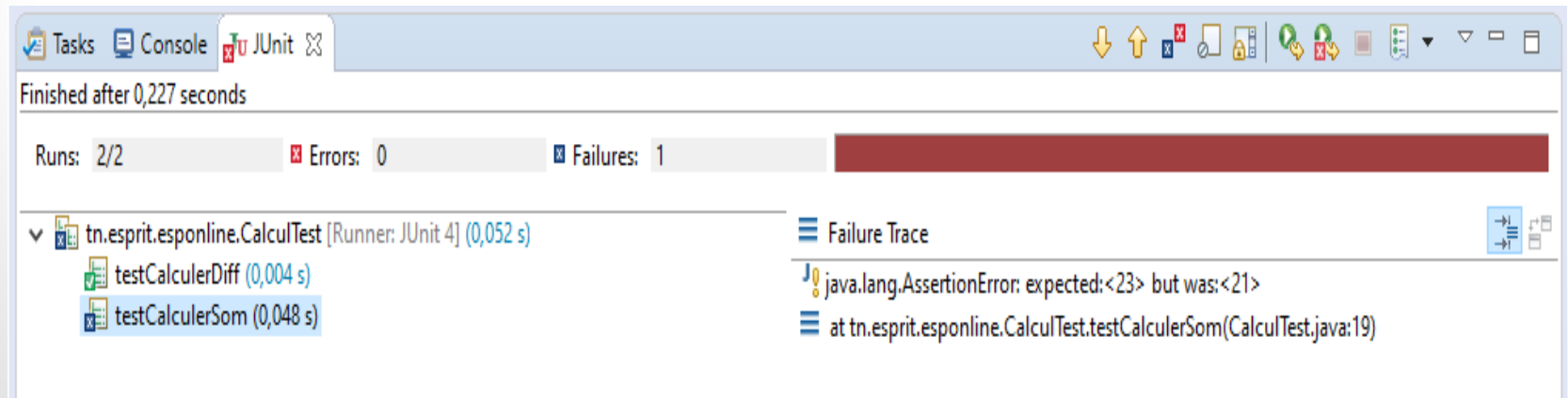
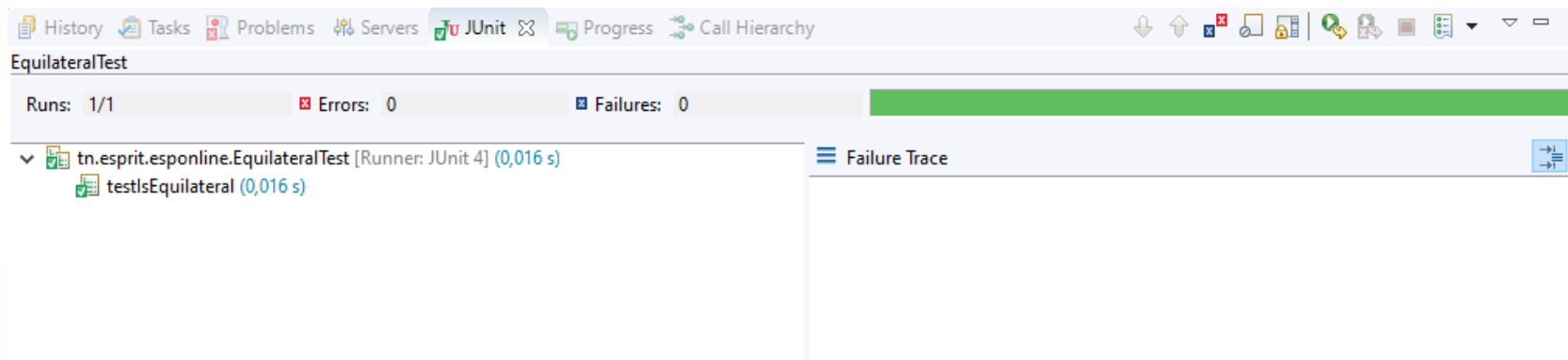
Active Windows

Outils



JUnit

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

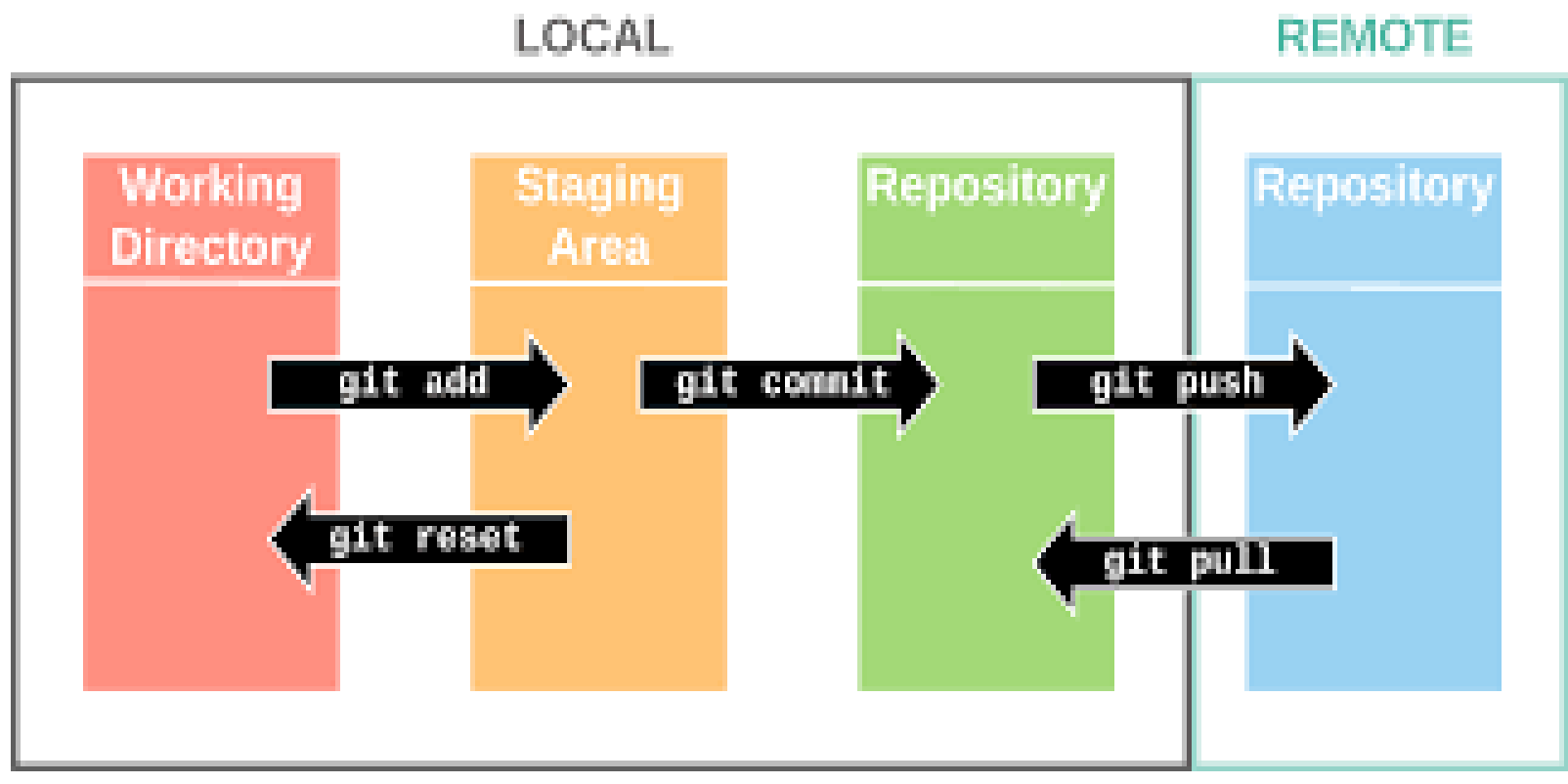


Outils



GIT

Git est un logiciel de gestion de versions décentralisé

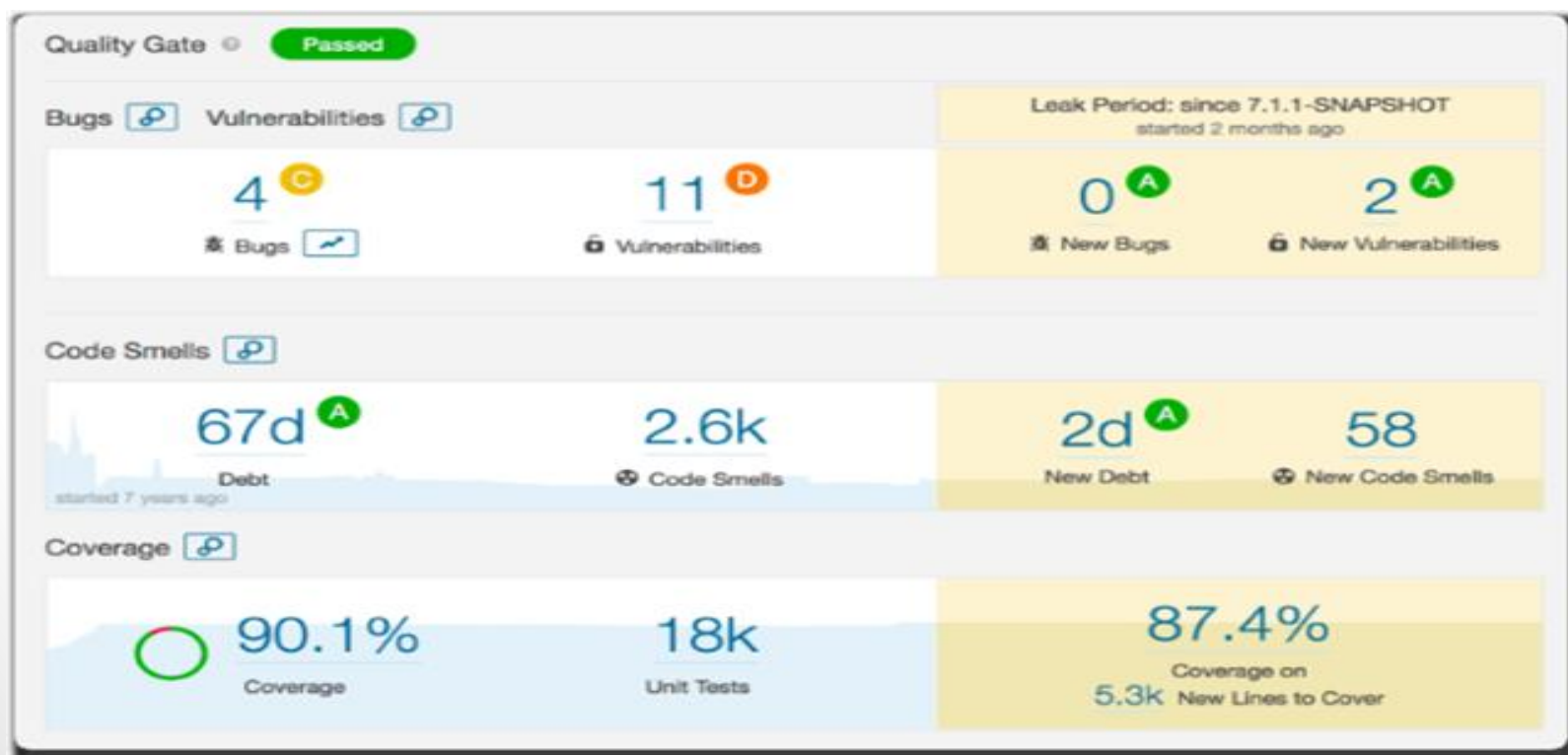


Outils



Sonar

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






Outils




Nexus


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


 **Sonatype Nexus Repository Manager**
OSS 3.22.0-02


 


Browse








 Welcome

 Search

 **Browse**

 Upload

 **Browse** Browse assets and components

	Name ↑	Type	Format	Status
	maven-central	proxy	maven2	Online - Ready to Connect
	maven-public	group	maven2	Online
	maven-releases	hosted	maven2	Online
	maven-snapshots	hosted	maven2	Online
	nuget-group	group	nuget	Online
	nuget-hosted	hosted	nuget	Online
	nuget.org-proxy	proxy	nuget	Online - Ready to Connect

Outils

Jenkins



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.

The screenshot shows the Jenkins web interface in a browser window. The top navigation bar includes the Jenkins logo, a search bar, a notification bell with 1 alert, the user 'MAB', and a 'log out' button. The left sidebar contains links for 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. The main content area displays a table of builds with columns for status (S), warnings (W), name, last success, last failure, and last duration. Below the table, there are links for 'Icon: S M L', 'Legend', and 'Atom feed' options.

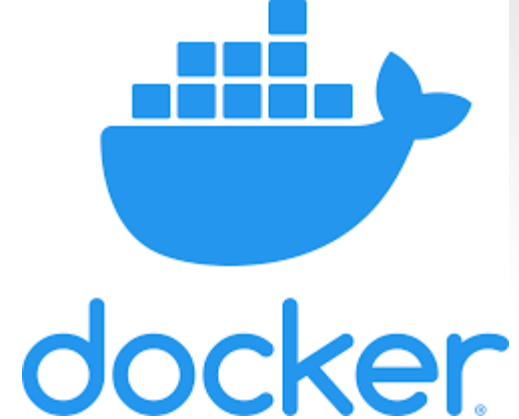
S	W	Name ↓	Last Success	Last Failure	Last Duration
		JenkinsArtifacts	8 mo 19 days - #6	6 mo 1 day - #8	55 sec
		Mass-Spring-Damper	6 mo 1 day - #33	6 mo 1 day - #32	20 sec
		MSD	6 mo 1 day - #13	6 mo 1 day - #12	2 min 10 sec
		mtcnn-face-detection	3 days 7 hr - #7	1 mo 3 days - #3	51 sec
		mtcnn-face-detection-pipeline	1 mo 3 days - #1	1 mo 2 days - #2	55 sec

Icon: S M L Legend Atom feed for all Atom feed for failures Atom feed for just latest builds

Outils

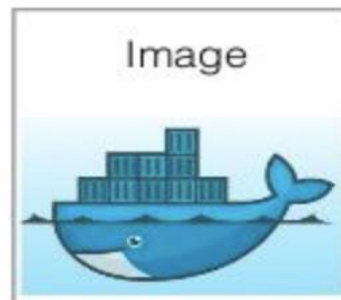
- **Docker**

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.



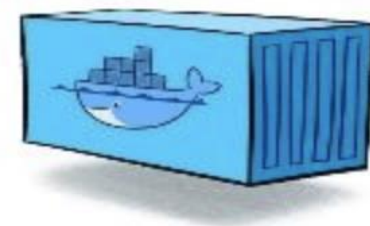
Dockerfile

build



Docker Image

run



Docker Container

Outils



Grafana / Prometheus

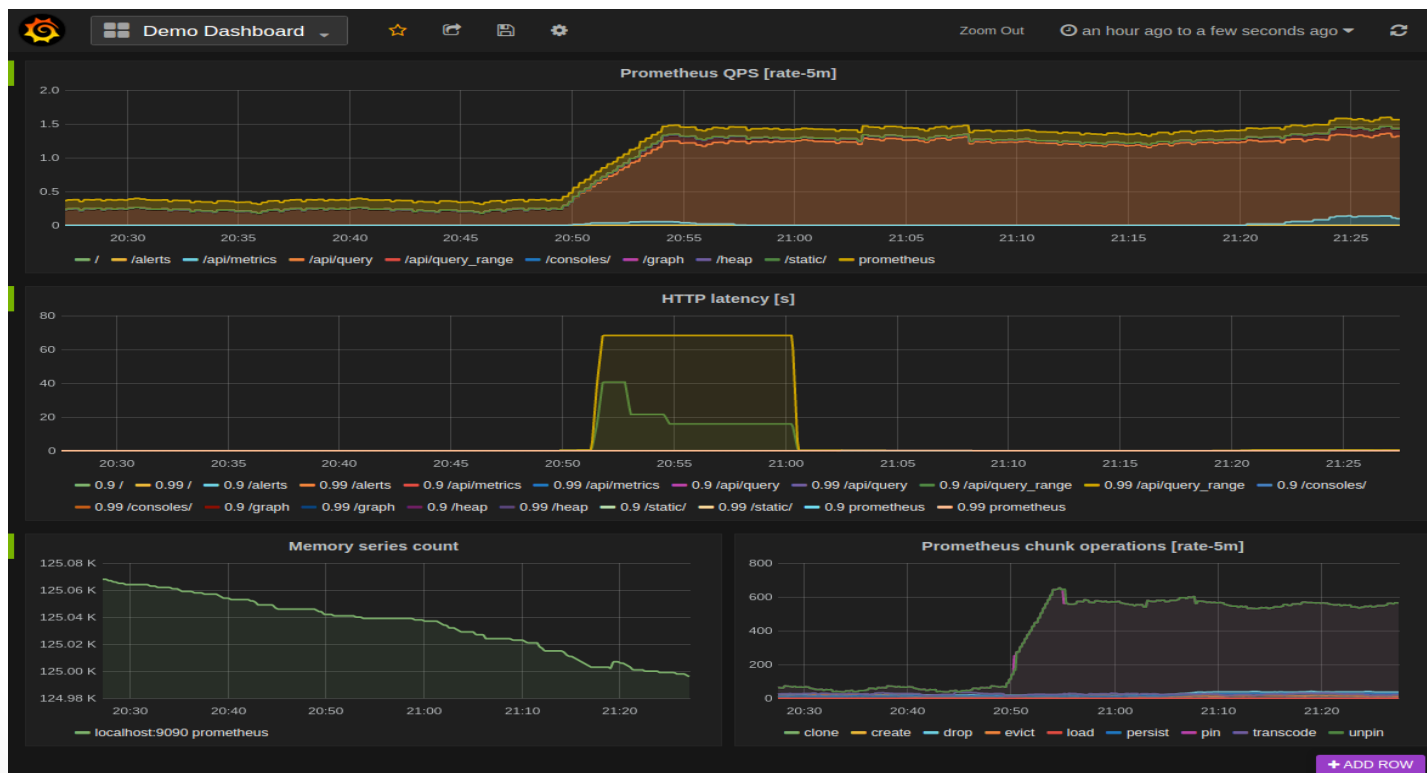
- Grafana est un logiciel libre sous licence GNU Affero General Public License qui permet la visualisation de données. Il permet de réaliser des tableaux de bord et des graphiques depuis plusieurs sources dont des bases de données temporelles comme Graphite, InfluxDB et OpenTSDB.
- Prometheus est un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



Installation des outils

Pour la prochaine séance, il faut installer :

- STS ou intellij
- JDK 8
- MySQL

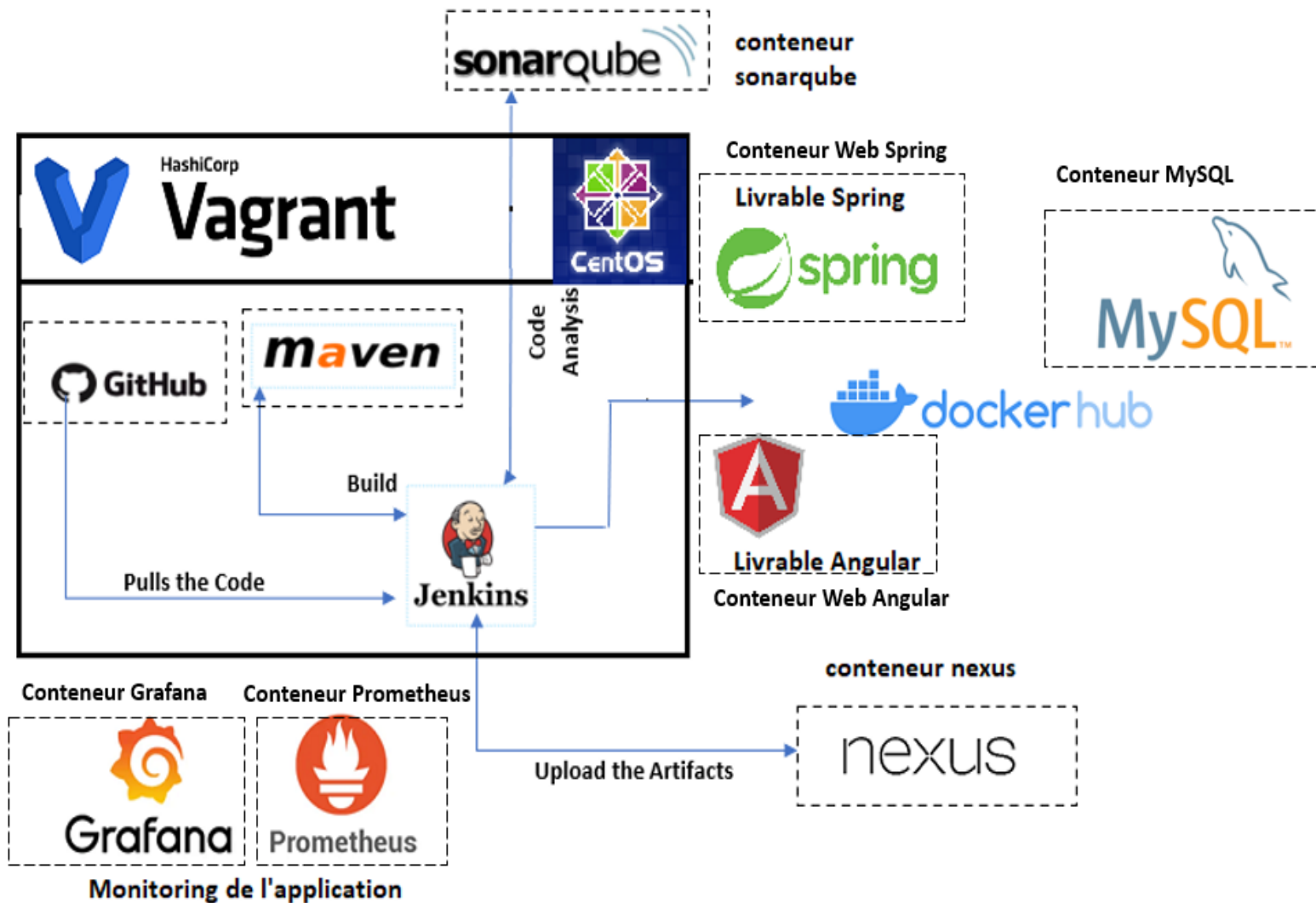
- Virtual box  **VirtualBox**

- Vagrant  **VAGRANT**

- Une machine virtuelle Centos7 dans Vagrant



Solution finale



Solution finale

Stage View



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

**Département Informatique
UP Architectures des Systèmes
d'Information**

Bureau E204