



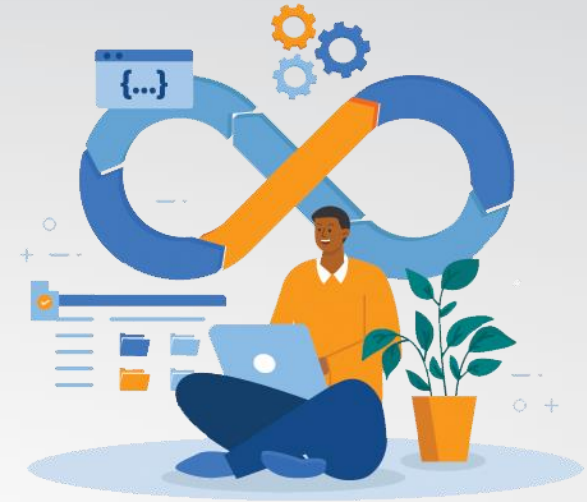
# DevOps

## Chapitre 1 : Introduction

ESPRIT – UP ASI (Architecture des Systèmes d'Information)  
Bureau E204

# ► Plan du cours

- Horaires, Evaluation
- **Contenu du Module DevOps**
- Evolution des **Méthodologies** de Développement
- Evolution des **Architectures** de Projets Informatiques
- **Apport de DevOps**
- **Définitions** : DevOps, Intégration Continue, Déploiement Continue, Livraison Continue
- **Environnement** : Spring Boot, Angular, Virtual Box, Vagrant, Ubuntu, Maven, JUnit, Git, Sonar, Nexus, Jenkins, Docker, Docker Compose, Docker Volume, Prometheus, Grafana
- Solution Finale



# ▶ Horaires

- Durée Totale : **30 heures**
  - Séances : **10 séances**
  - Cours : **9 heures**
  - TP : **15 heures**
  - Evaluation : **6 heures**
- 
- Durée de chaque Séance : **3 heures**
  - 2 heures synchrone + 1 heure asynchrone

\* Vendredi : **13h45**



# Evaluation



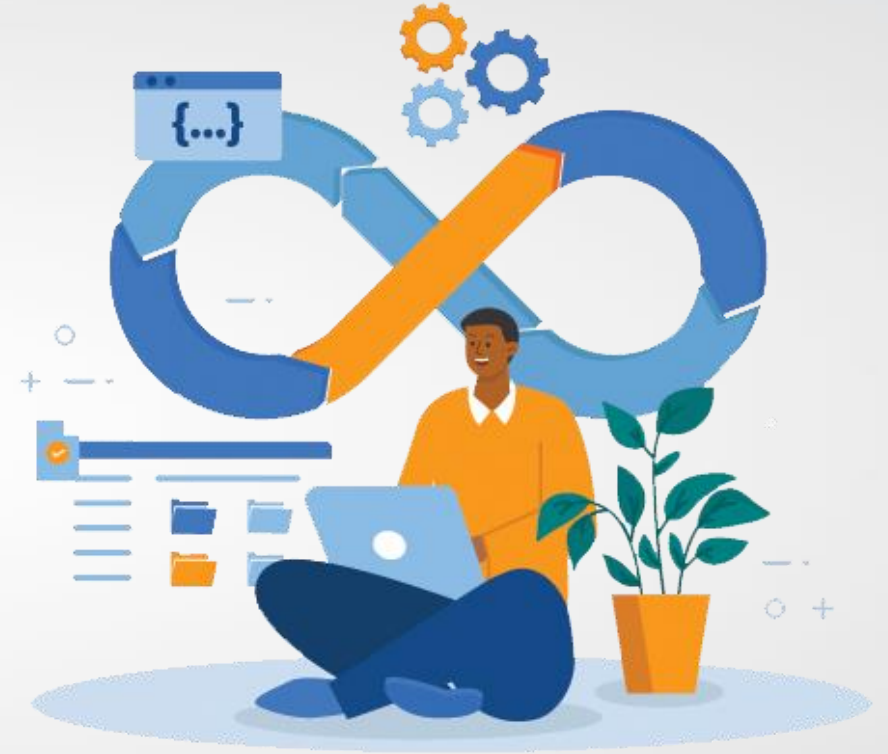
- L'évaluation se fait tout au long du module, et non pas uniquement à la fin
- La moyenne du module est calculée comme suit :  
**Moyenne = Note de la Validation du Projet Final**
- La validation du projet se fait lors des deux dernières séances (S9 et S10)
- La note du projet tiendra compte aussi de l'avancement du projet tout au long du cours (avancement sur votre projet chaque semaine)
- Travail en équipe (5 à 6 par équipe)
- L'examen de rattrapage est **pratique**



# ► Contenu du module DevOps



- Introduction **DevOps** (installation **Virtual Box** / **Vagrant** / **Ubuntu**)
- **Jenkins** (Serveur d'intégration continue)
- Introduction à **Docker**
- **Git** (Projet Spring Boot et Projet Angular)
- **Nexus** (Gestion des livrables)
- Test unitaire (**JUnit**)
- **Sonar** (Qualité de code)
- Docker avancé (**Docker compose** + **Docker volume**)
- **Grafana** et **Prometheus**
- Validation projet final



# ► Evolution des méthodologies



- **Méthodologie Classique / Lourde (ex. RUP, 2TUP)**
  - Orientée vers les processus et les outils
  - Documentation exhaustive
  - Moins de communication
  - Moins d'adaptation au changement
- **Méthodologie Agile (ex. Scrum, XP)**
  - Priorise la communication entre l'équipe de développement et le client
  - Favorise l'adaptation au changement
  - Se libère des outils et processus lourds
- **Travail Itératif (Sprint)**
  - Chaque Sprint est un projet avec un cycle de vie en V



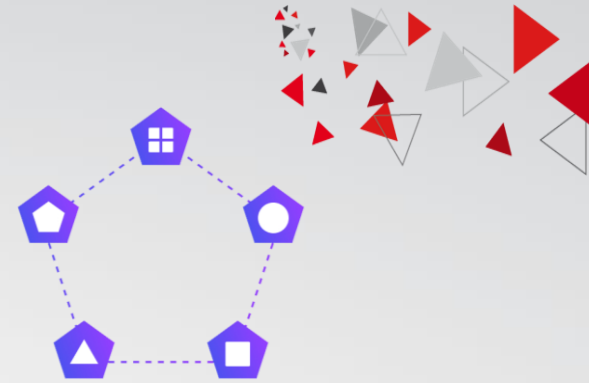
# ► Evolution des architectures

- **Architecture Monolithique**



- Une seule application regroupant toutes les fonctionnalités
- Difficulté à faire évoluer des parties spécifiques de l'application
- Contrainte d'utiliser les mêmes technologies pour l'ensemble de l'application

- **Architecture Micro-services**



- Décomposition de l'application en petits services indépendants, chacun responsable d'une fonction spécifique
- Facilité d'ajout, de mise à jour et de mise à l'échelle des services individuels
- Permet l'utilisation de technologies différentes pour chaque service, en fonction de leurs besoins spécifiques





# ► Cycle de vie d'un projet



## Vie en production

Corrections de bugs  
Maintenance

## Développement

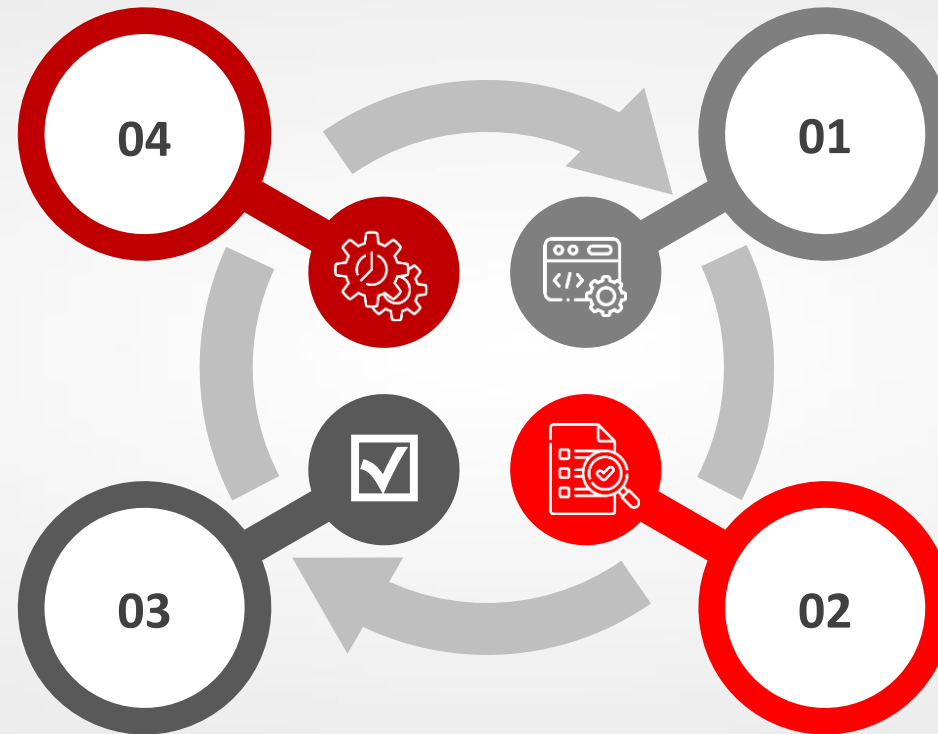
Création de fonctionnalités

## Mise en production

Accessibilité du projet par le client  
Satisfaire les contraintes clients  
(temps de réponse, uptime...)

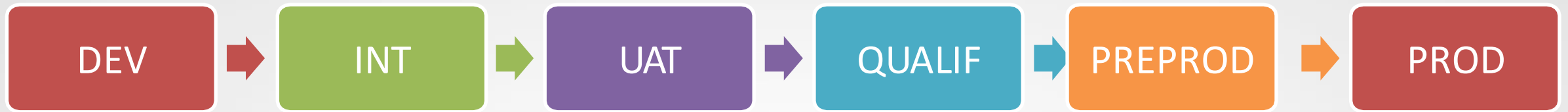
## Recette

Intégration des différents travaux  
Vérification du bon fonctionnement  
Identification des bugs





# ► Environnements



1. Environnement de Développement (**DEV**)
2. Environnement d'Intégration (**INT**)
3. Environnement de Test (**UAT** - User Acceptance Testing)
4. Environnement de Qualification (**QUALIF**)
5. Environnement de Pré-Production (**PREPROD**)
6. Environnement de Production (**PROD**)

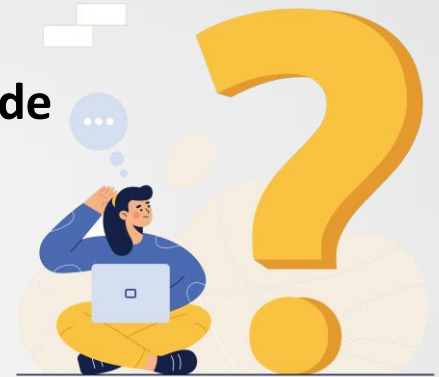


# ► Apport du DevOps



Utiliser une méthode **Agile**, et une architecture en **Micro-service** 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) ?



# ► DevOps - Définition



Le nom **DevOps** est la contraction des deux termes anglais **Development** (développement) et **Operations** IT (exploitation)

DevOps, c'est un **mouvement**, c'est une **culture** qui vise à améliorer la communication entre les études (développeurs) et l'exploitation afin de réduire le temps de mise sur le marché d'un produit.



C'est aussi un ensemble de **bonnes pratiques** :

- › Automatisation des différentes phases du projet (test, monitoring, déploiement, ...),
- › Réduction du Time To Market (**TTM**)



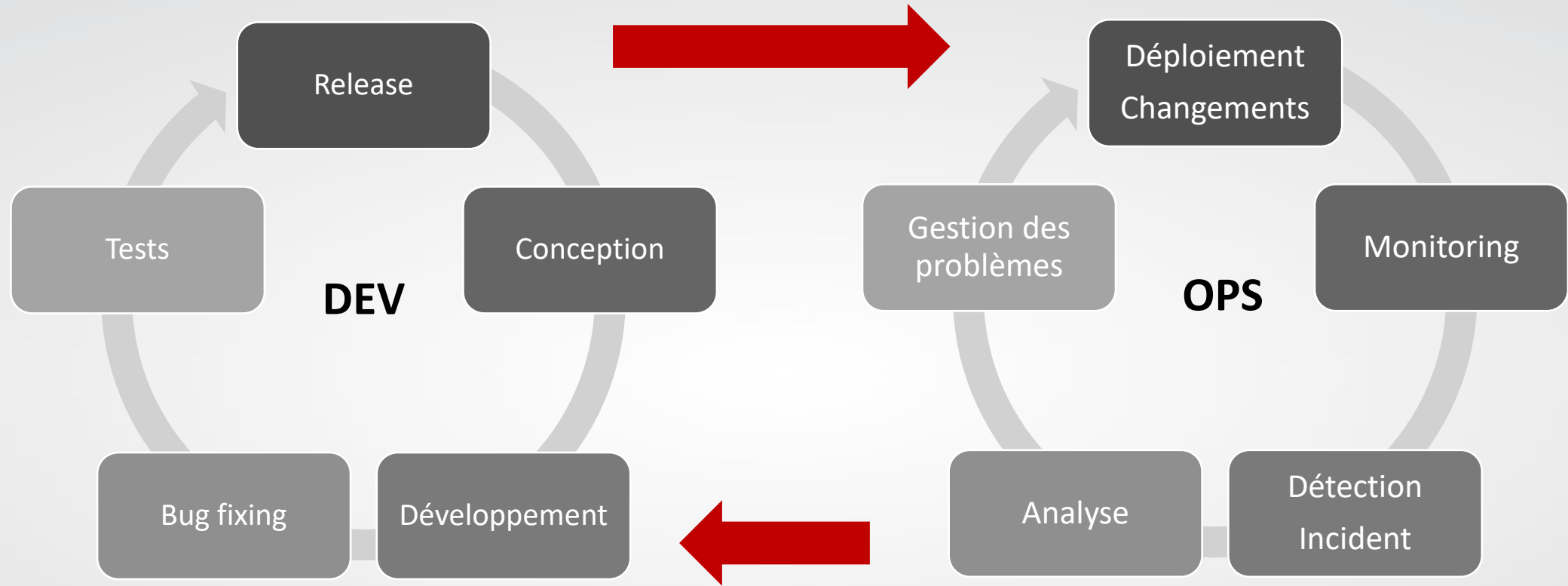
# ► DevOps – Visions différentes



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 <b>développement</b>	Environnements de <b>production</b>
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



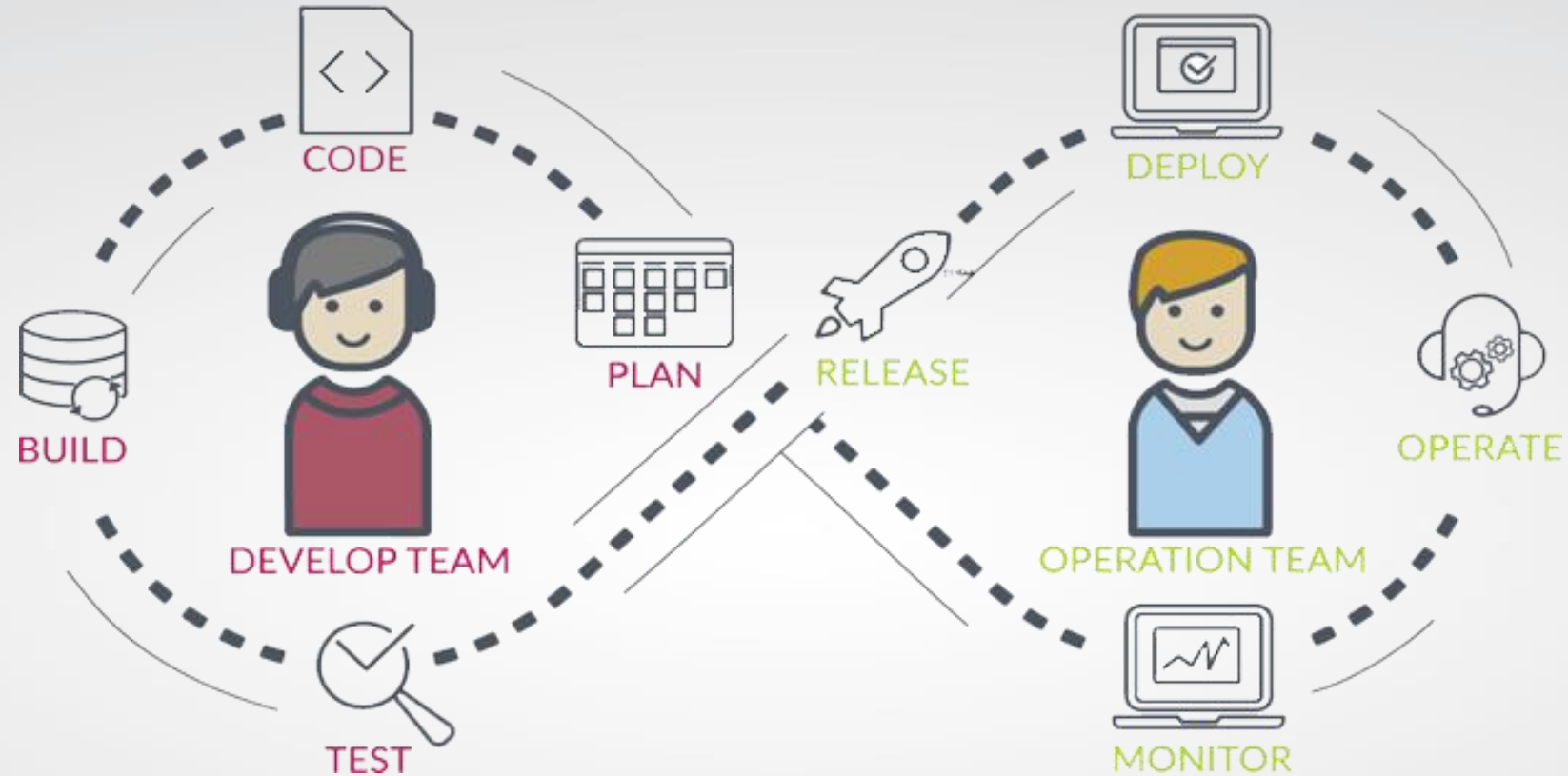
# ► DevOps – Cycles différents





© 2024-2024 – Module DevOps – Introduction

# ► DevOps - processus



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



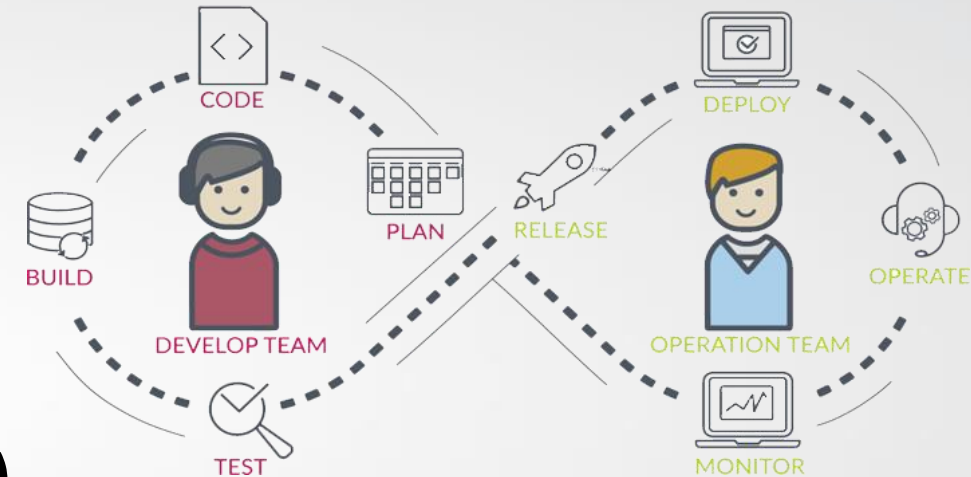


# ► DevOps - processus



Il existe 3 processus DevOps :

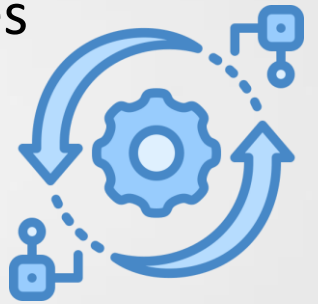
- › Intégration continue : **Continuous Integration (CI)**
- › Livraison continue : **Continuous Delivery (CD)**
- › Déploiement continu : **Continuous Deployment (CD)**



# ► Intégration Continue - CI



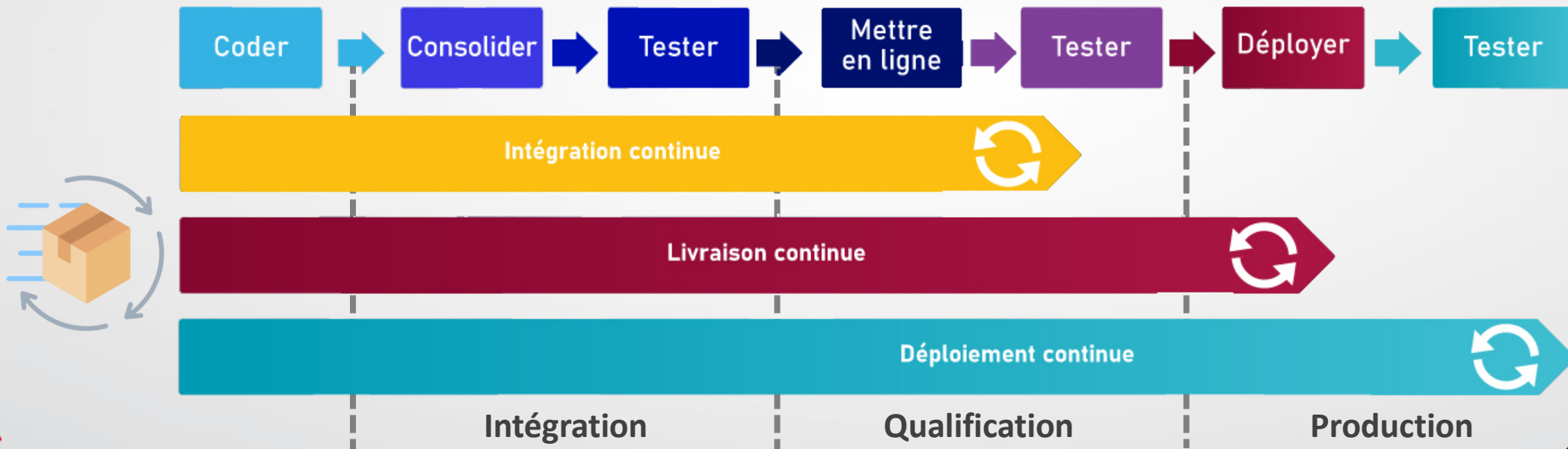
- L'Intégration Continue ou **Continuous Integration (CI)** est un processus axé sur les étapes consistant à compiler, tester et déployer dans un environnement d'intégration.
- Le but est de **tester** aussi **fréquemment** que possible pour détecter les régressions du livrable et les bugs le plus tôt possible.
- La majeure partie du **travail** est **automatisée** à l'aide d'outils de test.
- Le **déploiement** sur la plateforme d'intégration est **simplifié** et peut être effectué par les équipes de développement **sans** nécessiter **l'intervention de l'équipe d'exploitation**.



# ► Livraison Continue - CD



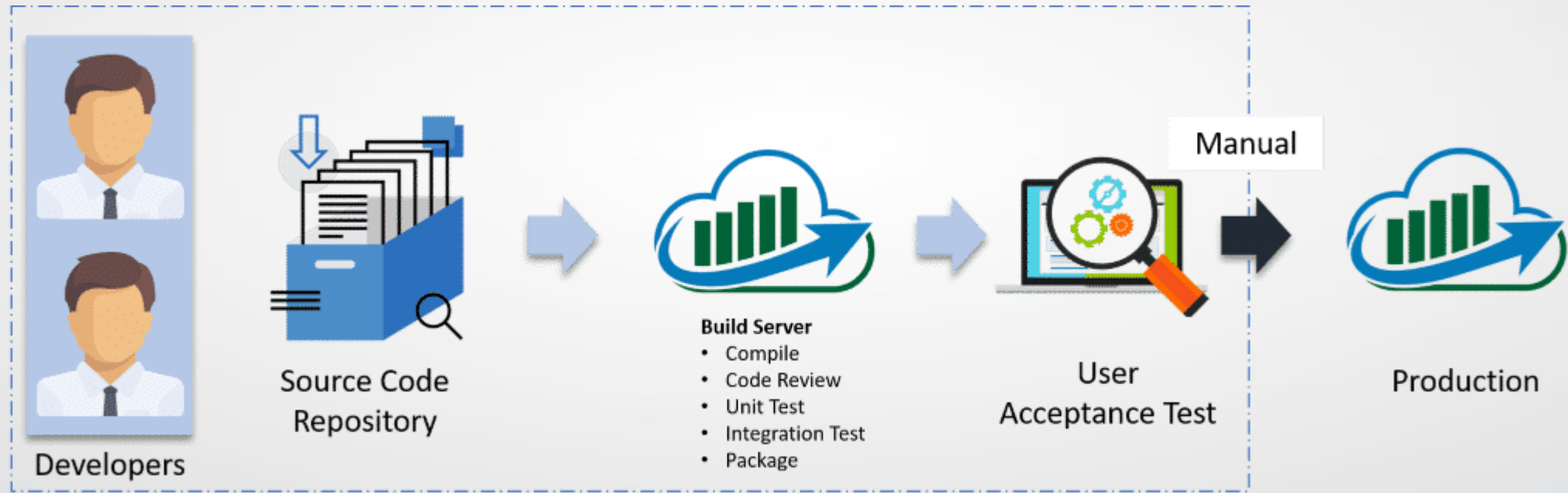
- La Livraison Continue ou **Continuous Delivery (CD)** est un processus orienté production consistant à *déployer* automatiquement sur *un environnement donné* (UAT, QUALIF, PREPROD), à l'exception de la Production où la livraison reste manuelle.



# ► Déploiement Continu - CD



- Le Déploiement Continu ou **Continuous Déploiement (CD)** est un processus orienté production consistant à *déployer* automatiquement sur tous les environnements y compris sur l'*environnement de production*.



# ► DevOps – Les fondamentaux



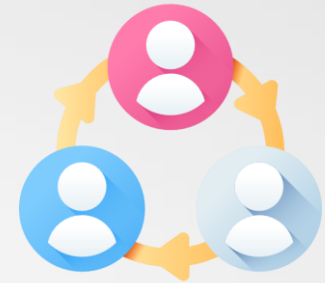
RÉDUCTION DU  
CYCLE DES  
LIVRAISONS



OPTIMISATION  
DES RESSOURCES



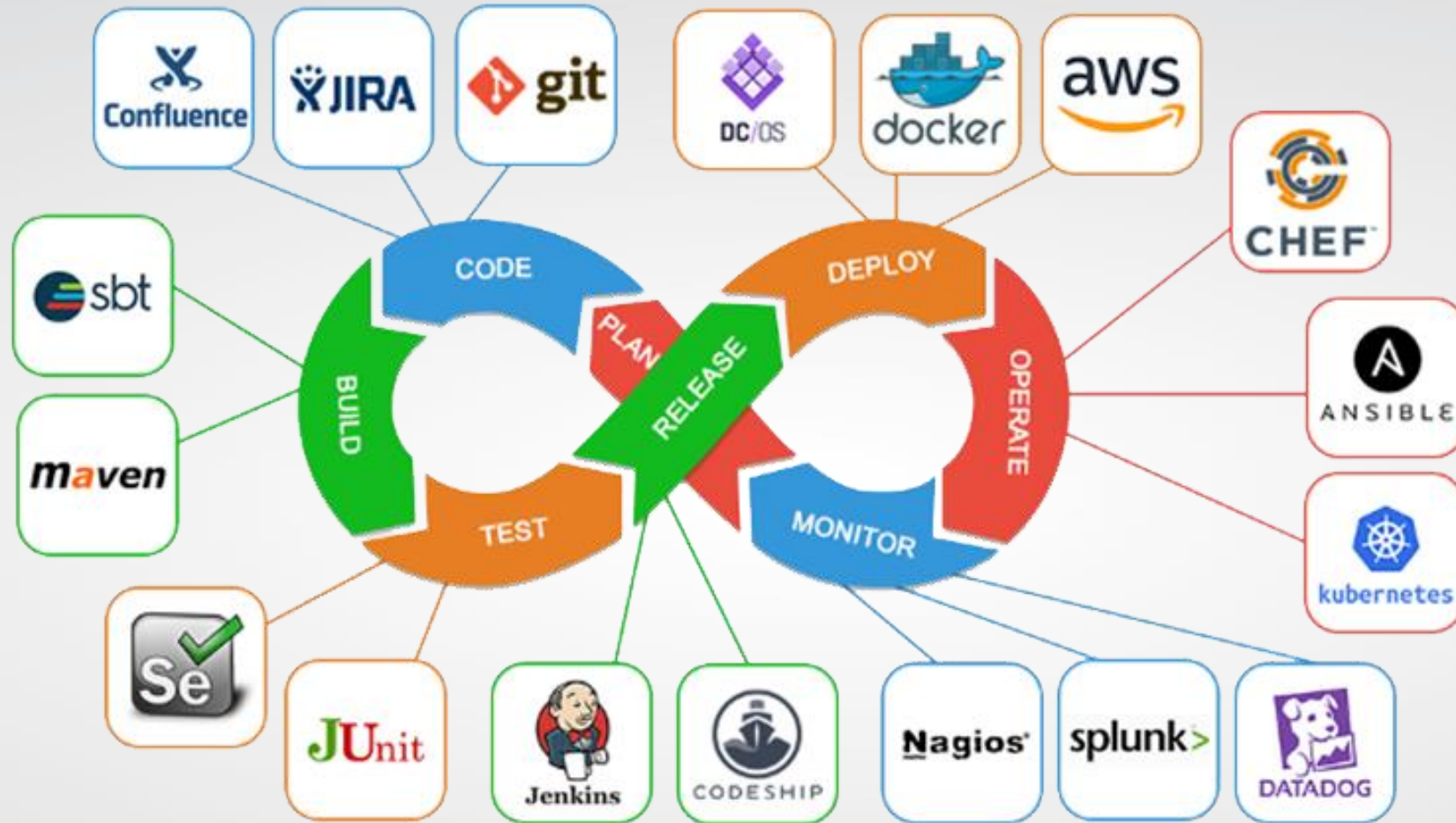
AMÉLIORATION  
DE LA QUALITÉ



REPLACER L'HUMAIN  
AU CŒUR DU  
DISPOSITIF



# ► DevOps - Outils



DevOps en 2023 : LES DERNIÈRES TENDANCES



# ► DevOps - Outils



- 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.





# ► DevOps – Quelques Outils



Dans ce cours, nous allons nous intéresser à :

- › **Virtual Box / Vagrant / Ubuntu**
- › **Jenkins**
- › **Docker**
- › **Maven**
- › **JUnit**
- › **Git**
- › **Sonar**
- › **Nexus**
- › **Docker Compose, Docker Volume**
- › **Grafana / Prometheus**



Ces outils seront appliqués à deux projets **Spring Boot** et **Angular** déjà existants, que nous allons enrichir.



# ► Outils – Jenkins



# Jenkins



- › **Jenkins** est un outil open source d'intégration continue.
- › A chaque modification du code d'une application dans le gestionnaire de version, 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', a user profile 'MAB', and a 'log out' button. The left sidebar contains a menu with options: '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 build jobs under the 'All' tab. The table has columns for 'S' (status), 'W' (workspace icon), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. There are five rows of build data.

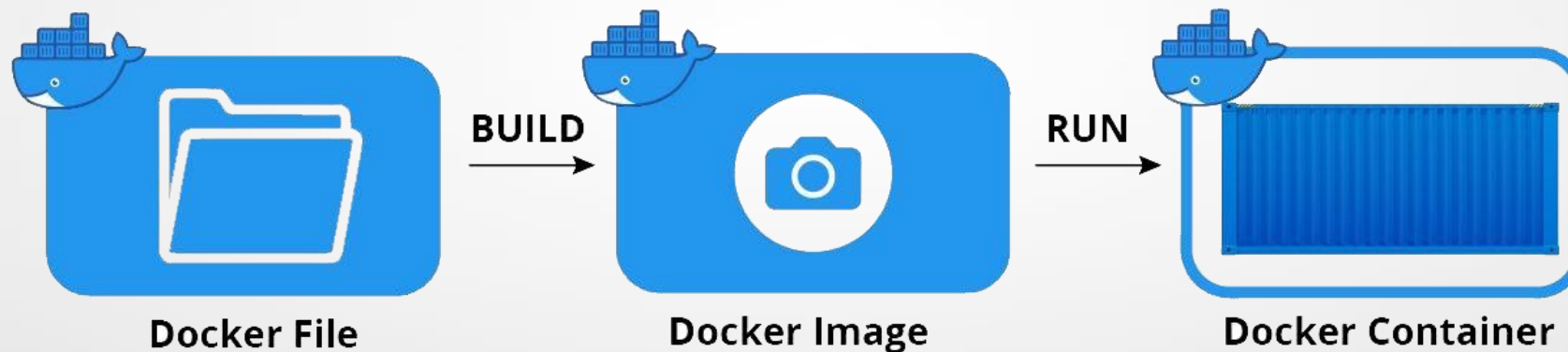
S	W	Name ↓	Last Success	Last Failure	Last Duration
		JenkinsArtifacts	8 mo 19 days - #6	6 mo 1 day - #8	55 sec
		Mass-Spring-Damper	6 mo 1 day - #33	6 mo 1 day - #32	20 sec
		MSD	6 mo 1 day - #13	6 mo 1 day - #12	2 min 10 sec
		mtcnn-face-detection	3 days 7 hr - #7	1 mo 3 days - #3	51 sec
		mtcnn-face-detection-pipeline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



# ► Outils – Docker



- › **Docker** est un outil qui permet de **packager une application et ses dépendances** dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur.



# ► Outils – Maven



- › **Maven** est un outil de **build** et de gestion de projet par construction automatisé. Développé par la fondation Apache.
- › Repose sur le paradigme « **Convention over configuration** » Convention plutôt que configuration.

```
[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
```

Activer Windows



# ► Outils – JUnit

JUnit



- › **JUnit** est un Framework de test unitaire pour le langage de programmation **Java**.



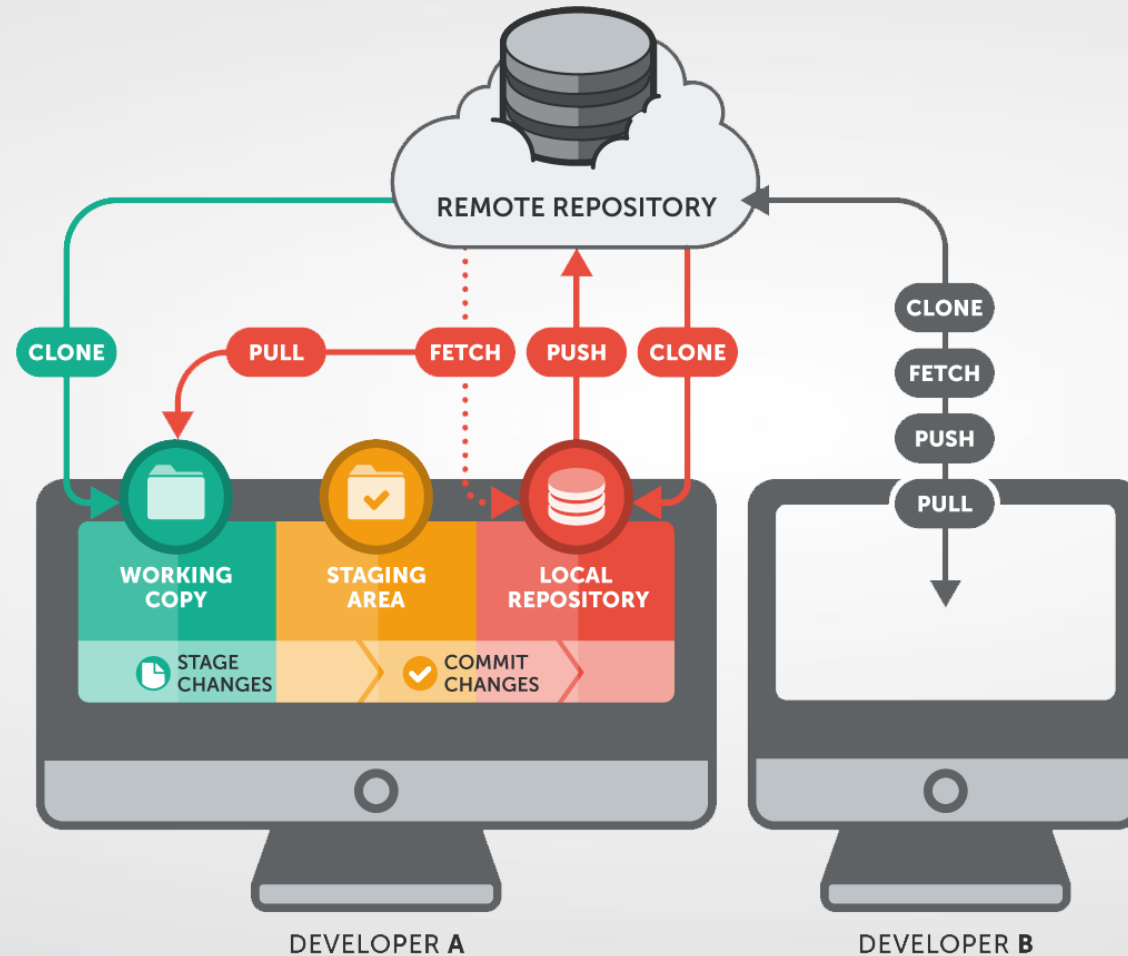
# ► Outils – Git



git



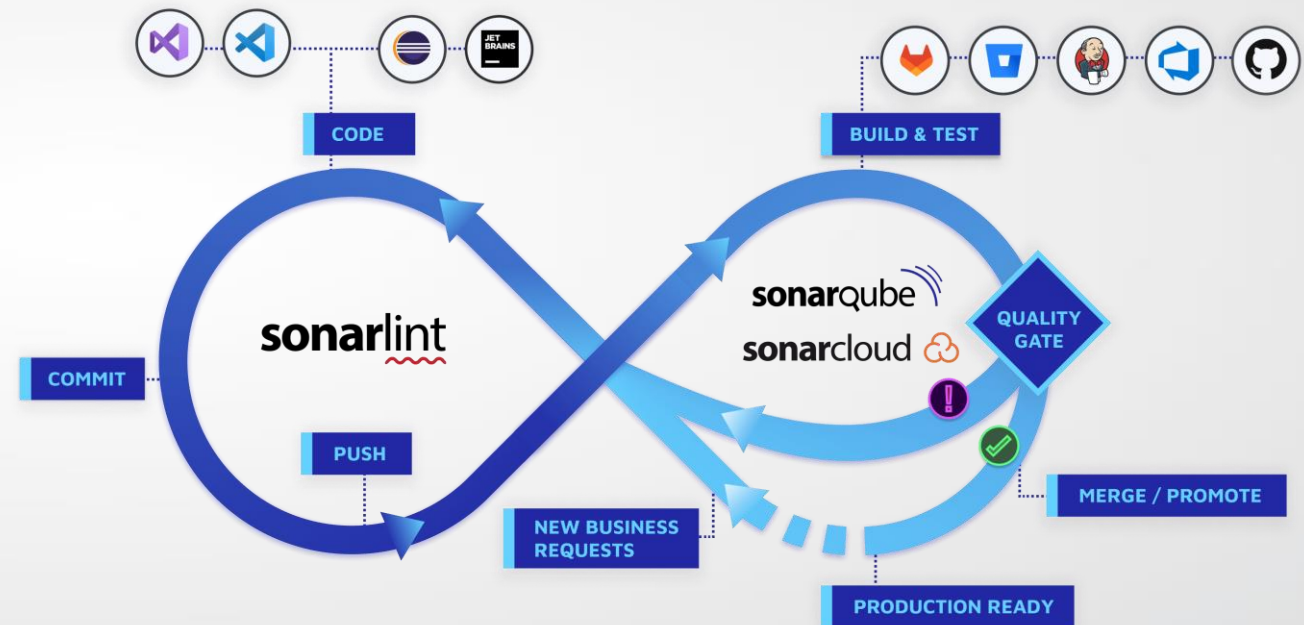
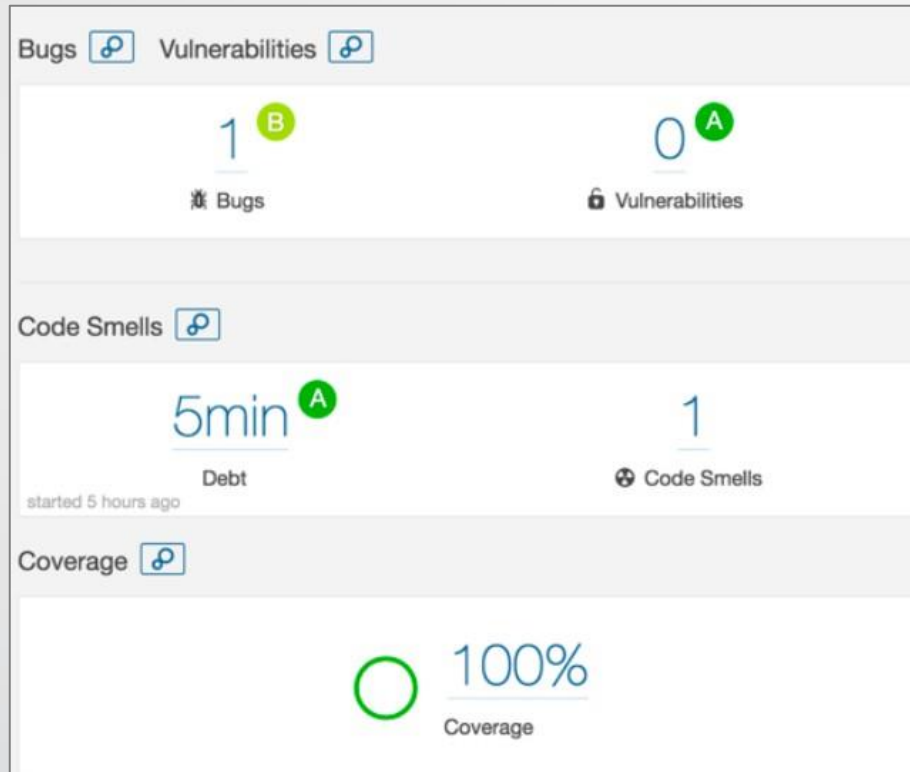
- › **Git** est un logiciel de gestion de versions décentralisé.



# ► Outils – Sonar



- › **SonarQube** est un logiciel libre permettant de mesurer la qualité du code source en continu (revue de code automatique).





# ► Outils – Nexus



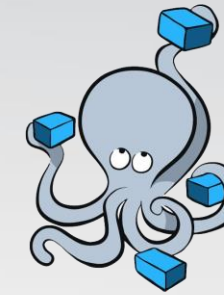
- › **Nexus** est un gestionnaire de référentiel qui organise, stocke et distribue les artefacts nécessaires au développement.

The screenshot shows the Sonatype Nexus Repository Manager web interface. The top navigation bar includes the Sonatype logo, the text 'Sonatype Nexus Repository Manager OSS 3.22.0-02', a search bar, and icons for home, settings, and help. A left sidebar contains navigation links: 'Browse' (selected), 'Welcome', 'Search', and 'Upload'. The main content area is titled 'Browse' and 'Browse assets and components'. It displays a table of repositories with columns for Name, Type, Format, and Status.

	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



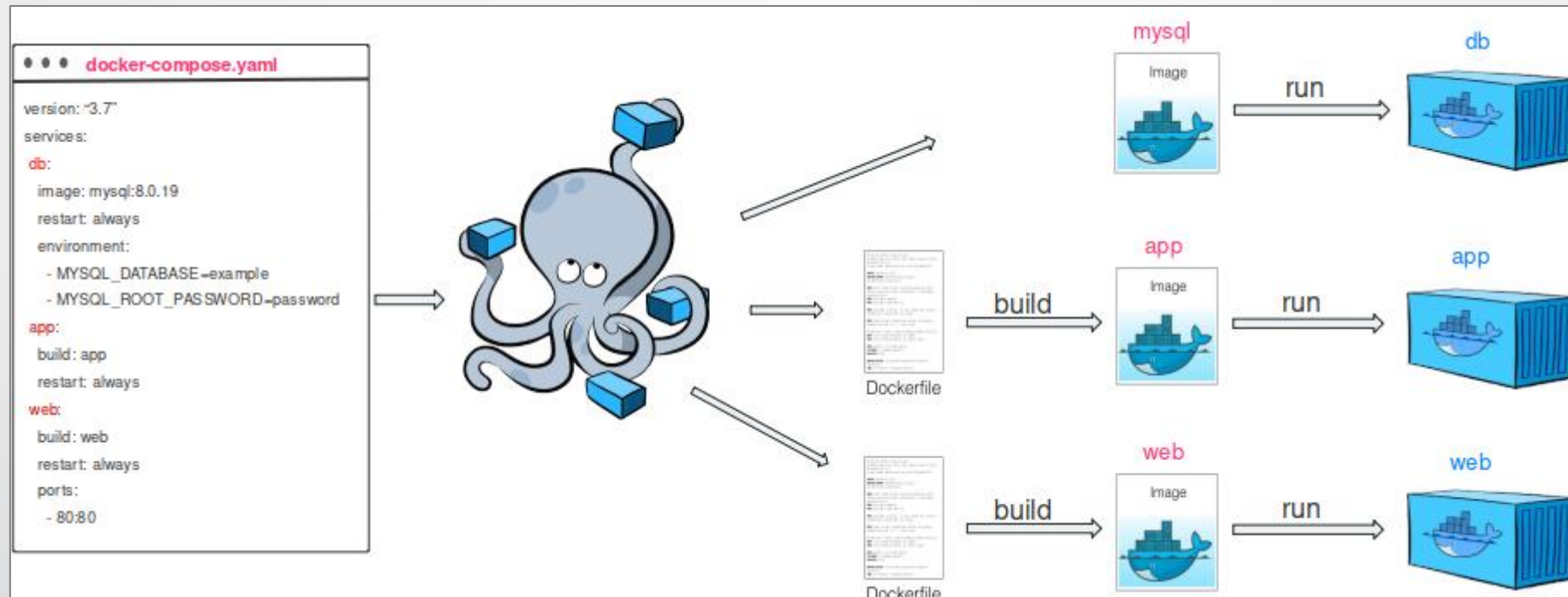
# ► Outils – Docker compose



docker  
Compose



- › **Docker Compose** est un outil qui permet de **définir** et de **gérer** des applications **multi-conteneurs avec Docker** en utilisant un fichier de configuration **YAML** pour spécifier les services, les réseaux et les volumes nécessaires pour l'application.



# ► Outils – Grafana / Prometheus



- › **Grafana** est un logiciel libre qui permet de créer des tableaux de bord et des graphiques à partir de diverses sources de données, y compris des bases de données temporelles telles que Graphite, InfluxDB et OpenTSDB, pour la visualisation de données.



- › **Prometheus** est un logiciel gratuit qui surveille et génère des alertes en enregistrant des métriques en temps réel dans une base de données de séries temporelles via le protocole HTTP.



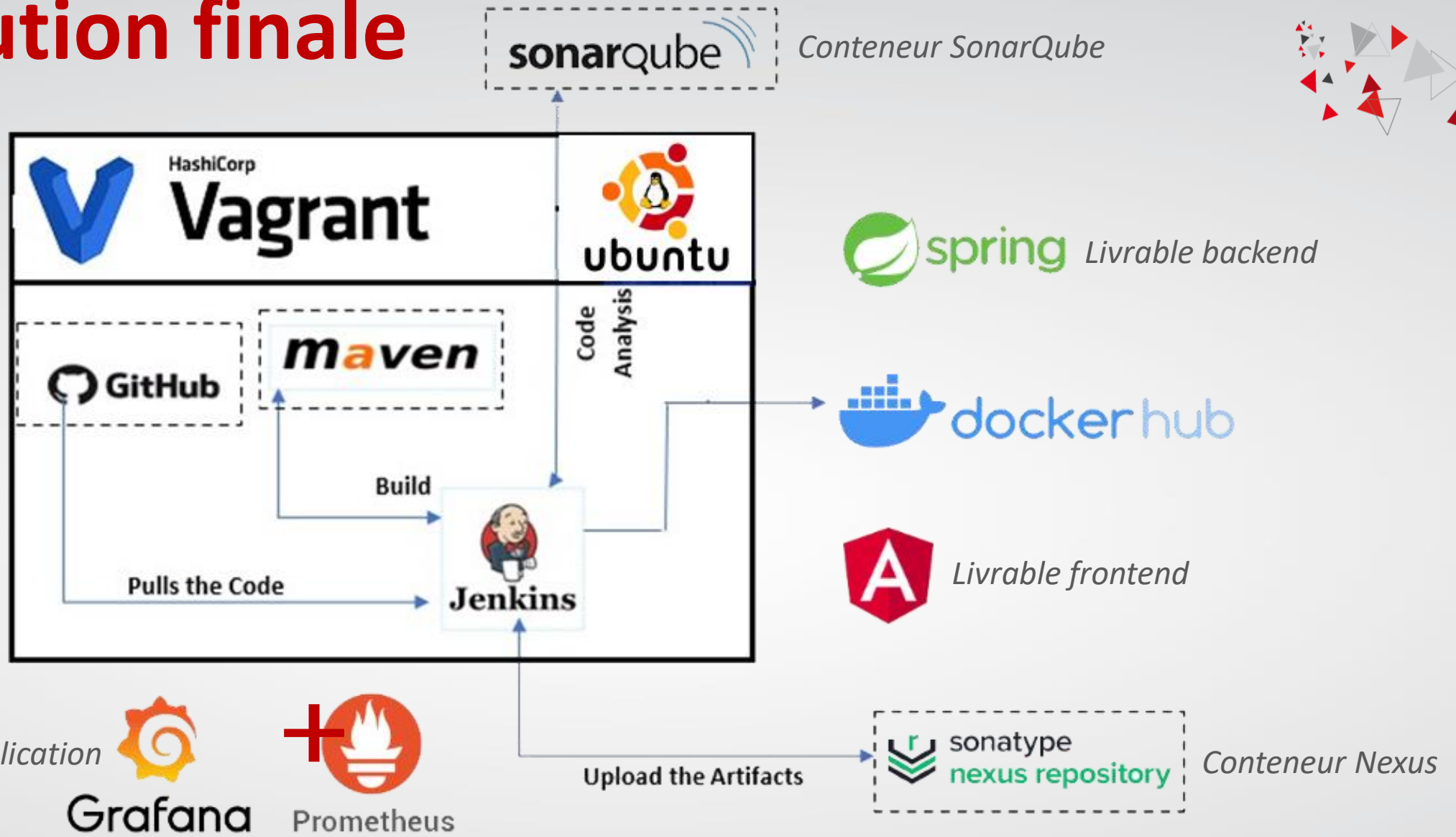
# ► Outils – Grafana / Prometheus



- › 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, suivre le tuto «1- Installation Vagrant-Ubunto.pdf» (voir Drive du cours), pour installer :

› Virtual box



**VirtualBox**


› Vagrant



**VAGRANT**

› Une machine virtuelle Ubuntu dans Vagrant





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