

# Introduction DevOps

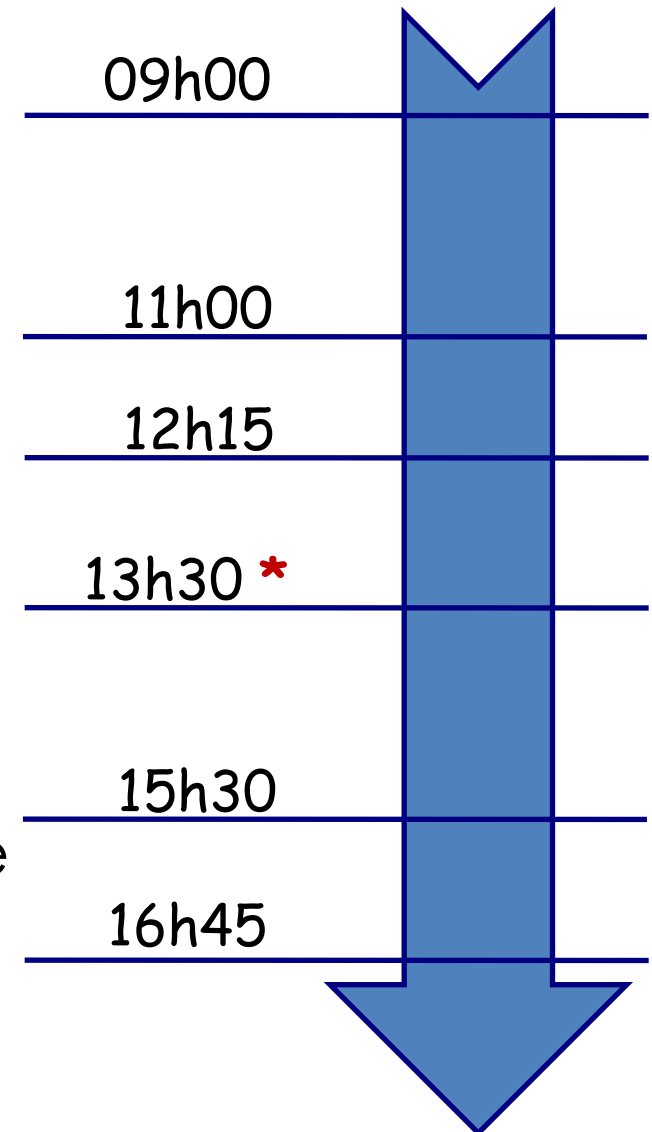


# Plan du Cours

- Horaires, Evaluation
- Contenu du Module DevOps
- Evolution des Méthodologies de Développement
- Apport de DevOps
- Définitions : DevOps, Intégration Continue, Déploiement / 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

- La moyenne du module est calculée comme suit :  
**Moyenne = Note de la Validation du Projet Final.**
- La validation se fait lors des deux dernières séances.
- L'enseignant tiendra compte aussi de l'avancement du projet tout au long du cours.
- Travail en équipe (5 à 6 personnes par équipe)

# Contenu du Module DevOps

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

# 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.

# Apport du DevOps

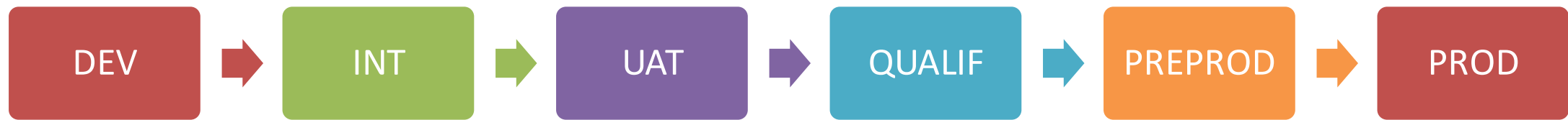
- Utiliser une méthode **Agile**, et une architecture en **Microservice** résout énormément de problèmes. **Mais ...**
- Comment faire travailler étroitement les équipes de production avec les équipes de développement?
- Comment automatiser au maximum les différentes phases du Projet?
- Comment pouvoir 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

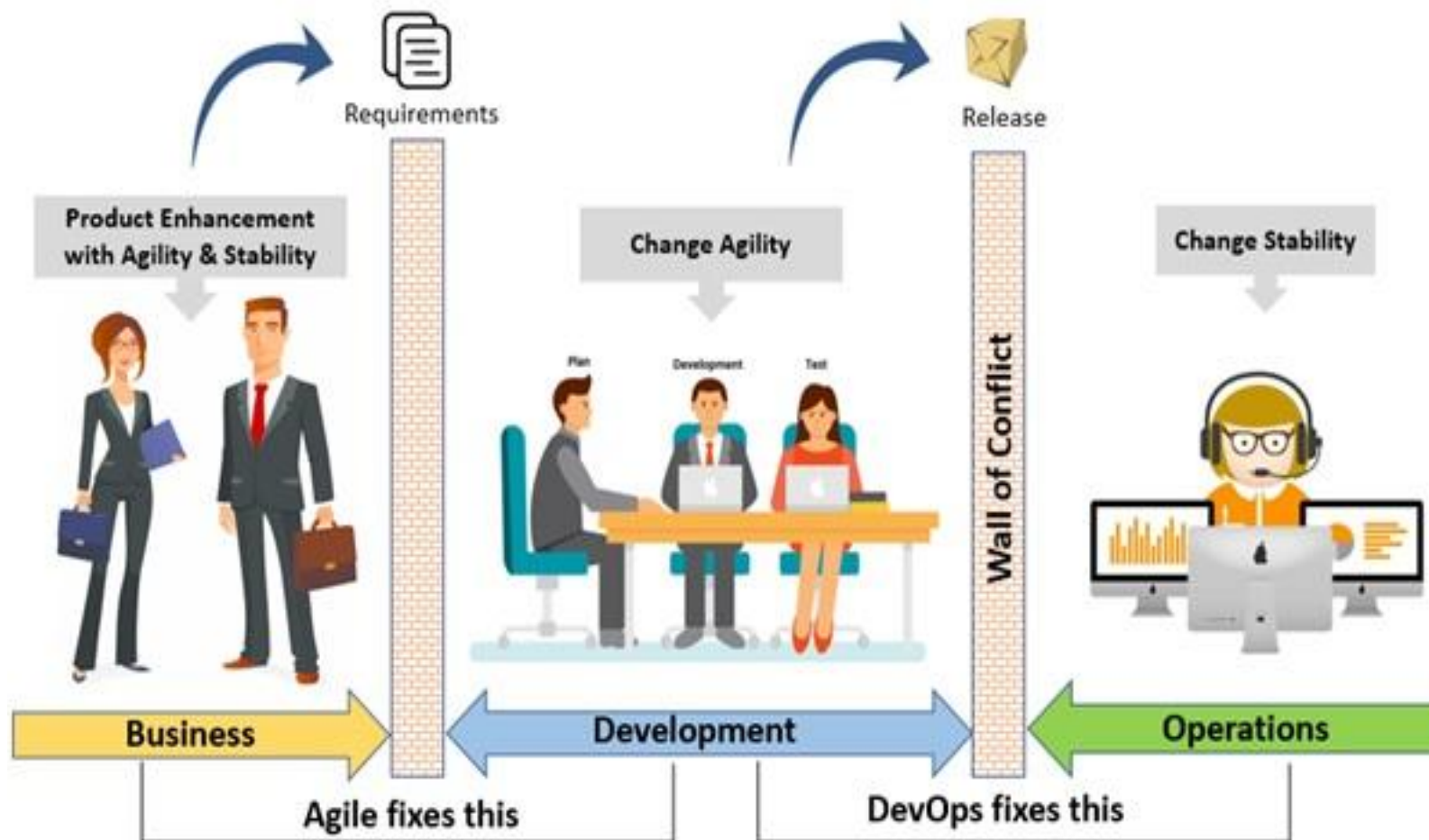
- Le nom « **DevOps** » vient de la contraction du mot « development » (développement) et « operation » (exploitation).
- DevOps est une **culture** qui vise à améliorer la **communication** entre les **développeurs** et l'équipe **d'exploitation**.
- C'est aussi un ensemble de **bonnes pratiques** pour **automatiser** les différentes phases du projet (test, monitoring, déploiement, ..)
- => Réduction du Time To Market (**TTM**)



# Environnements

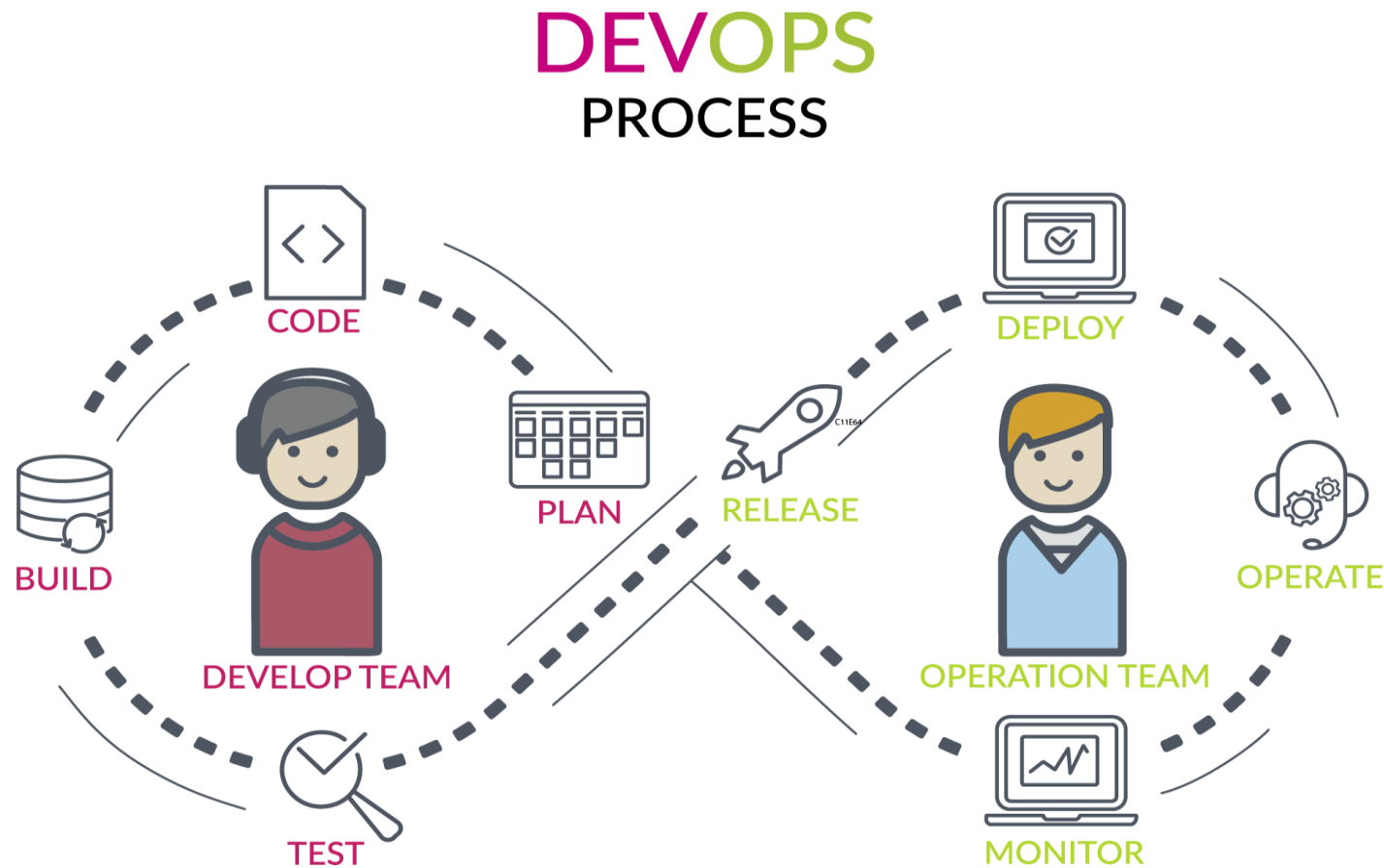


# Définition DevOps



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

# Définition DevOps



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

# Définition Intégration Continue

- L'**Intégration Continue** ou Continuous Integration est un processus orienté études consistant à compiler, tester et déployer sur un environnement d'intégration.
- Le but est de tester aussi souvent et autant que possible les non-régressions du livrable pour détecter les **bugs** plus tôt possible.
- La plupart du travail est réalisé par des outils de test. Le déploiement sur la plateforme d'intégration devient simple et peut être réalisé par les études sans faire intervenir l'exploitation.

# Définition Livraison Continue / Déploiement Continu

- La **Livraison Continue** ou Continuous Delivery est un processus orienté production consistant à déployer automatiquement sur un environnement donné (UAT, Qualification, Pré-Préproduction), à l'exception de la Production où la livraison reste **manuelle**.
- Le **Déploiement Continu** ou Continuous Deployment est un processus orienté production consistant à déployer automatiquement sur tous les environnements y compris sur l'environnement de **production**.

# Cycle de Vie Projet DevOps



# 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.

# Quelques Outils de DevOps

- 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



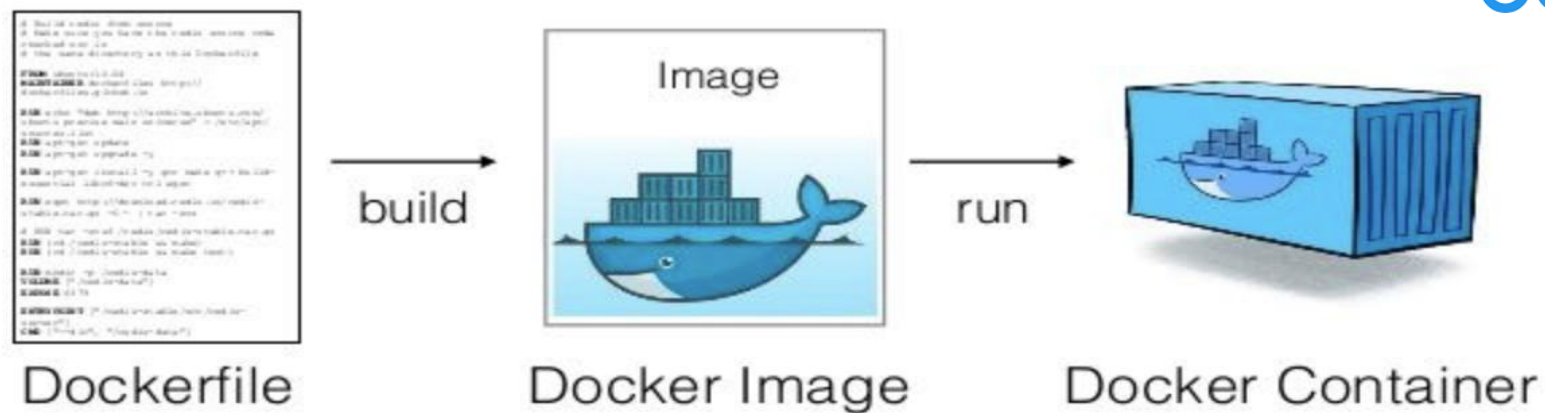
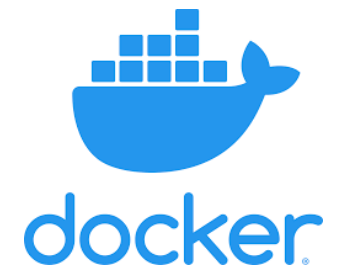
- **Jenkins** est un outil open source d'intégration continue
- A chaque modification de code d'une application dans le gestionnaire de version, Jenkins se charge automatiquement de la recompiler, et de la tester

The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, a notification bell with '1', the user 'MAB', and a 'log out' button. The left sidebar contains a menu with items like '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), weather icon (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

# Outils : Docker

- **Docker** est un outil qui peut 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 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

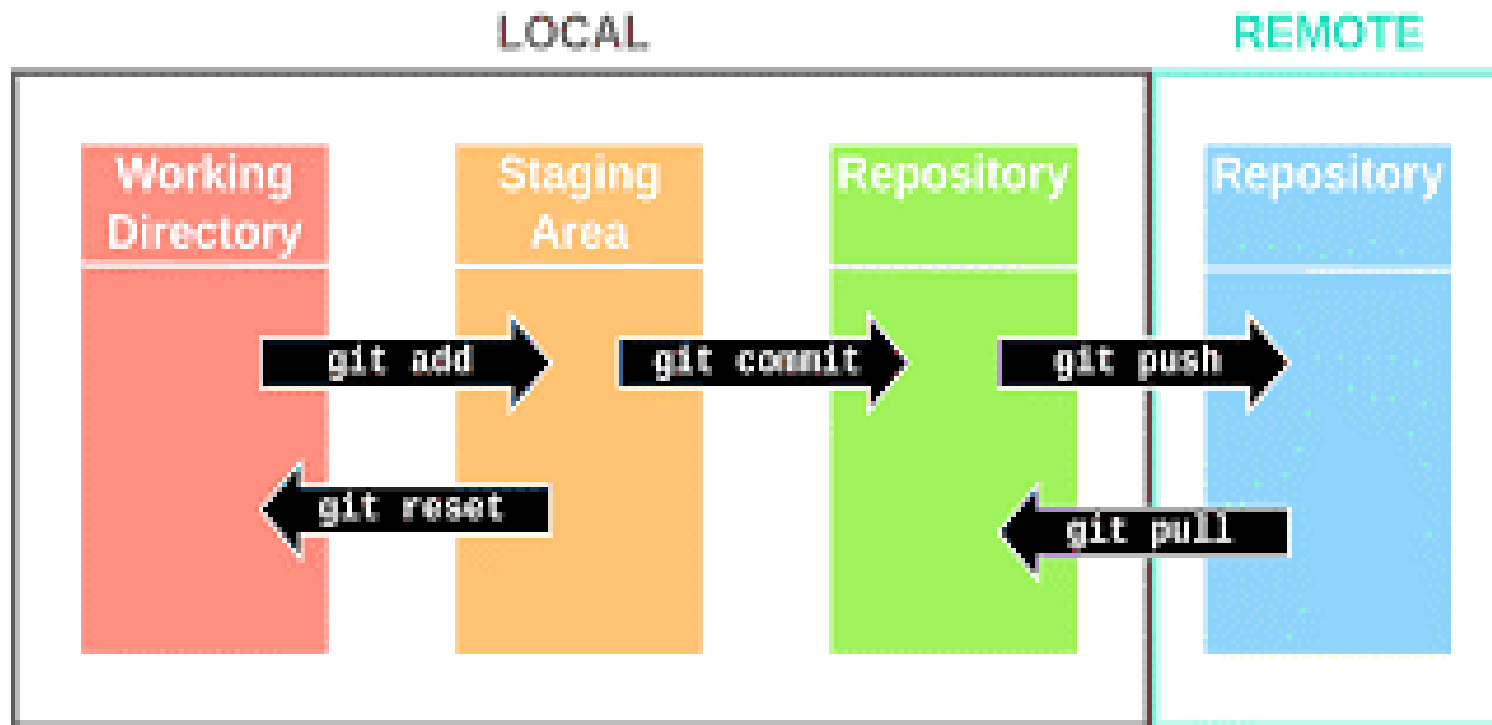
- **JUnit** est un framework de test unitaire pour le langage de programmation Java.

JUnit



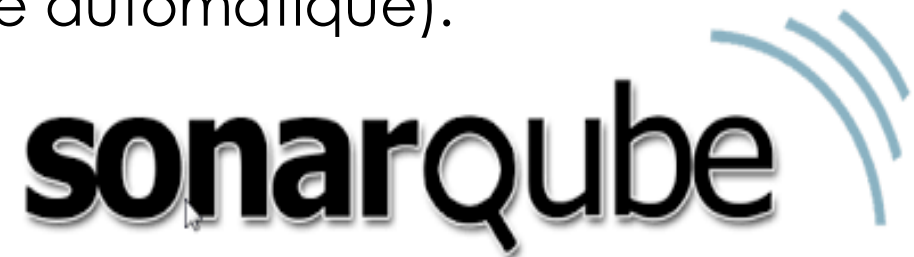
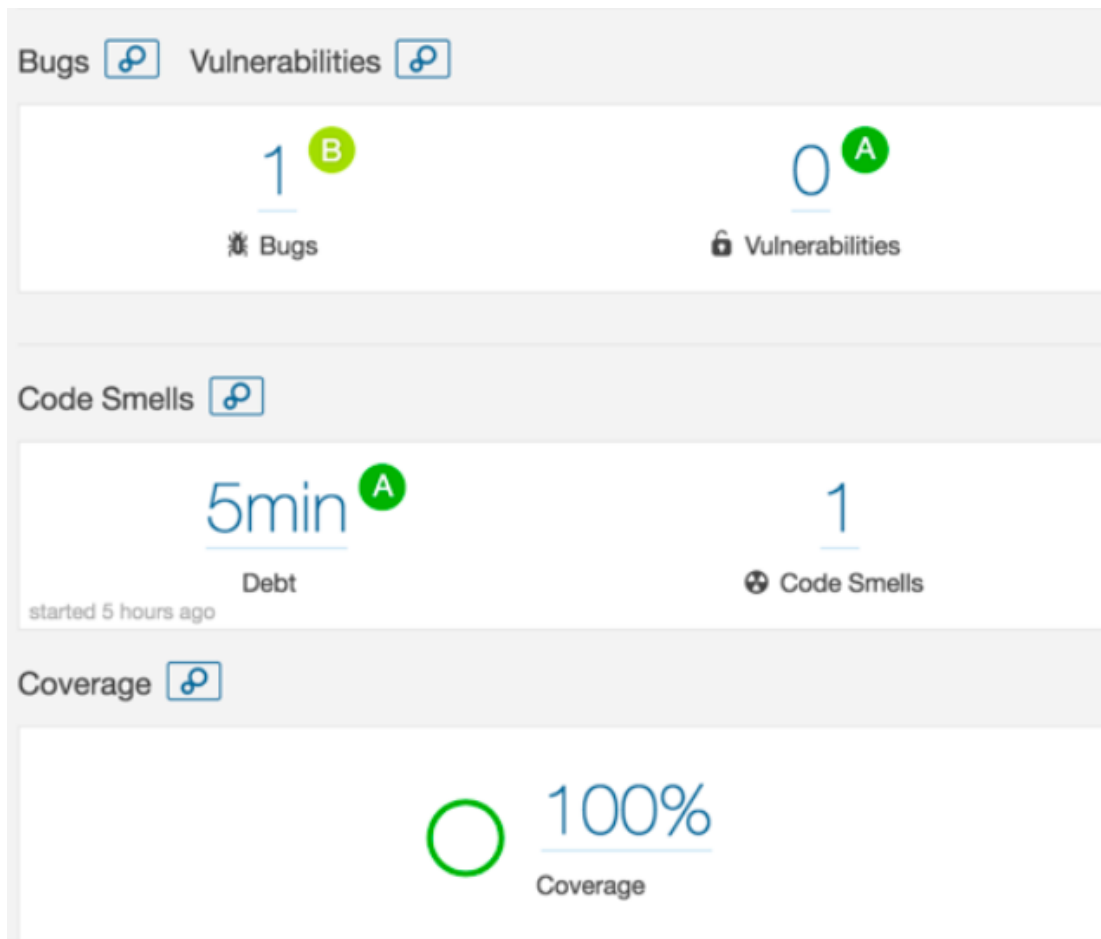
# Outils : Git

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



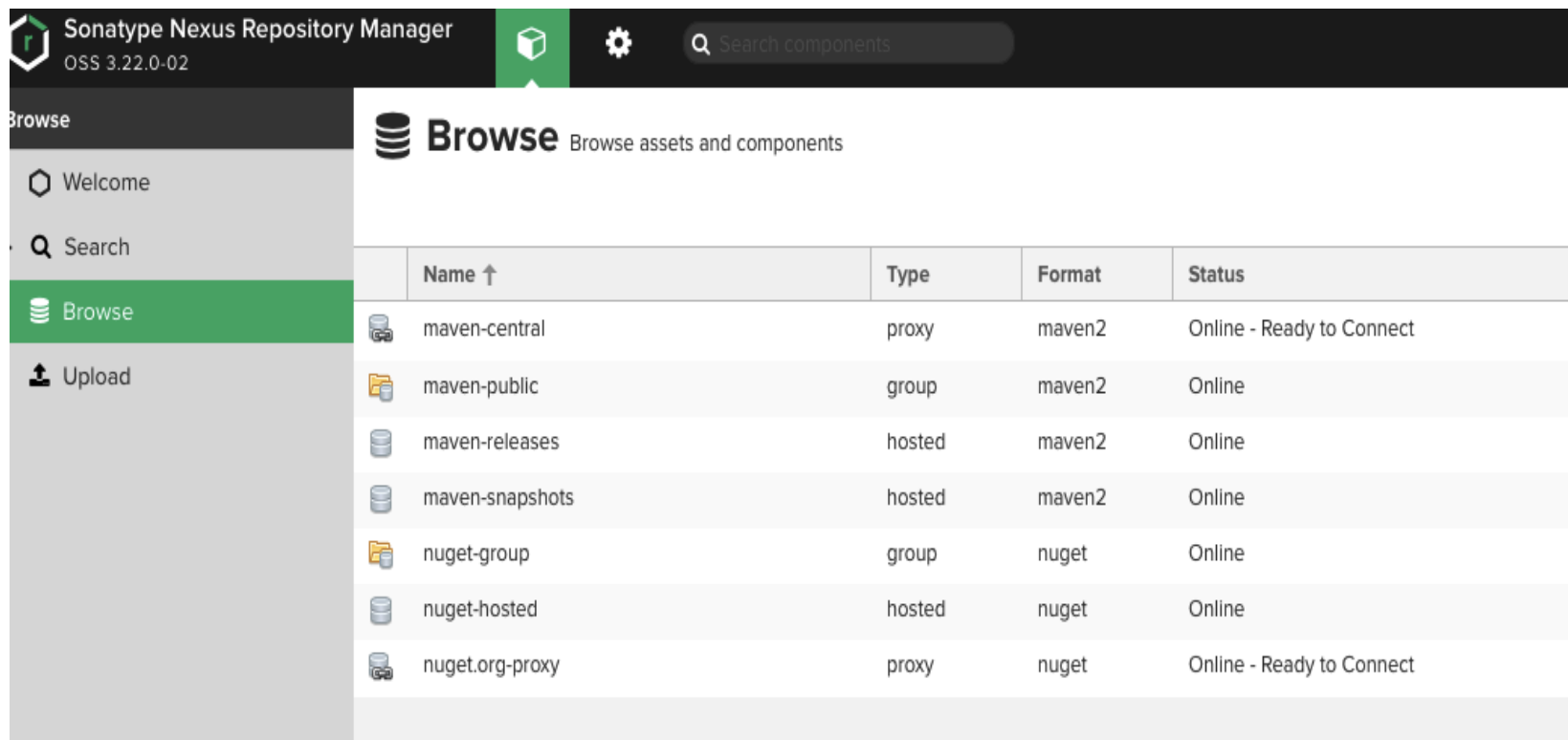
# Outils : Sonar








- **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

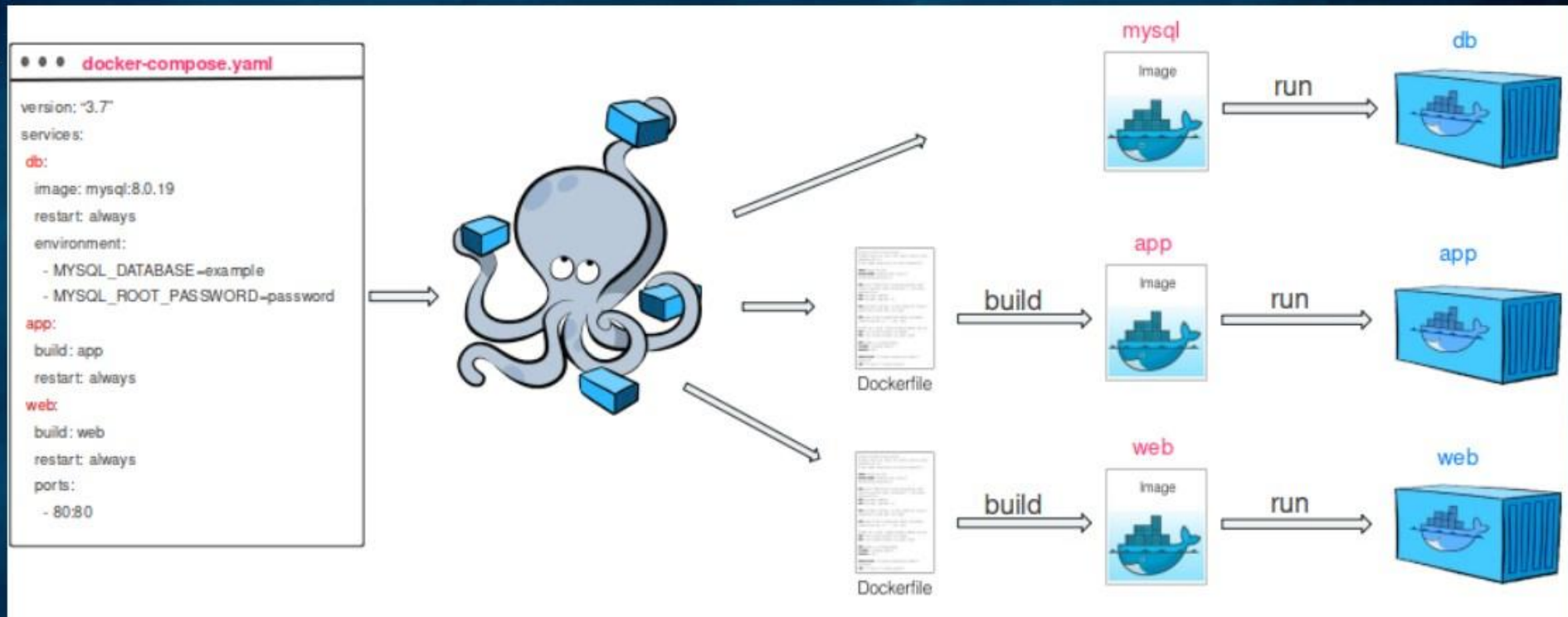
A screenshot of the Sonatype Nexus Repository Manager web interface. The top header shows the Sonatype logo, the text "Sonatype Nexus Repository Manager OSS 3.22.0-02", a green cube icon, a gear icon, and a search bar labeled "Search components". On the left, a sidebar menu has options: "Browse" (selected), "Welcome", "Search", and "Upload". The main content area is titled "Browse" with the subtitle "Browse assets and components". It contains a table with the following data:

	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



Run multiple containers as a service from 1 .yaml file



# Outils : Grafana / Prometheus

## Grafana / Prometheus

- Grafana est un logiciel libre 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.



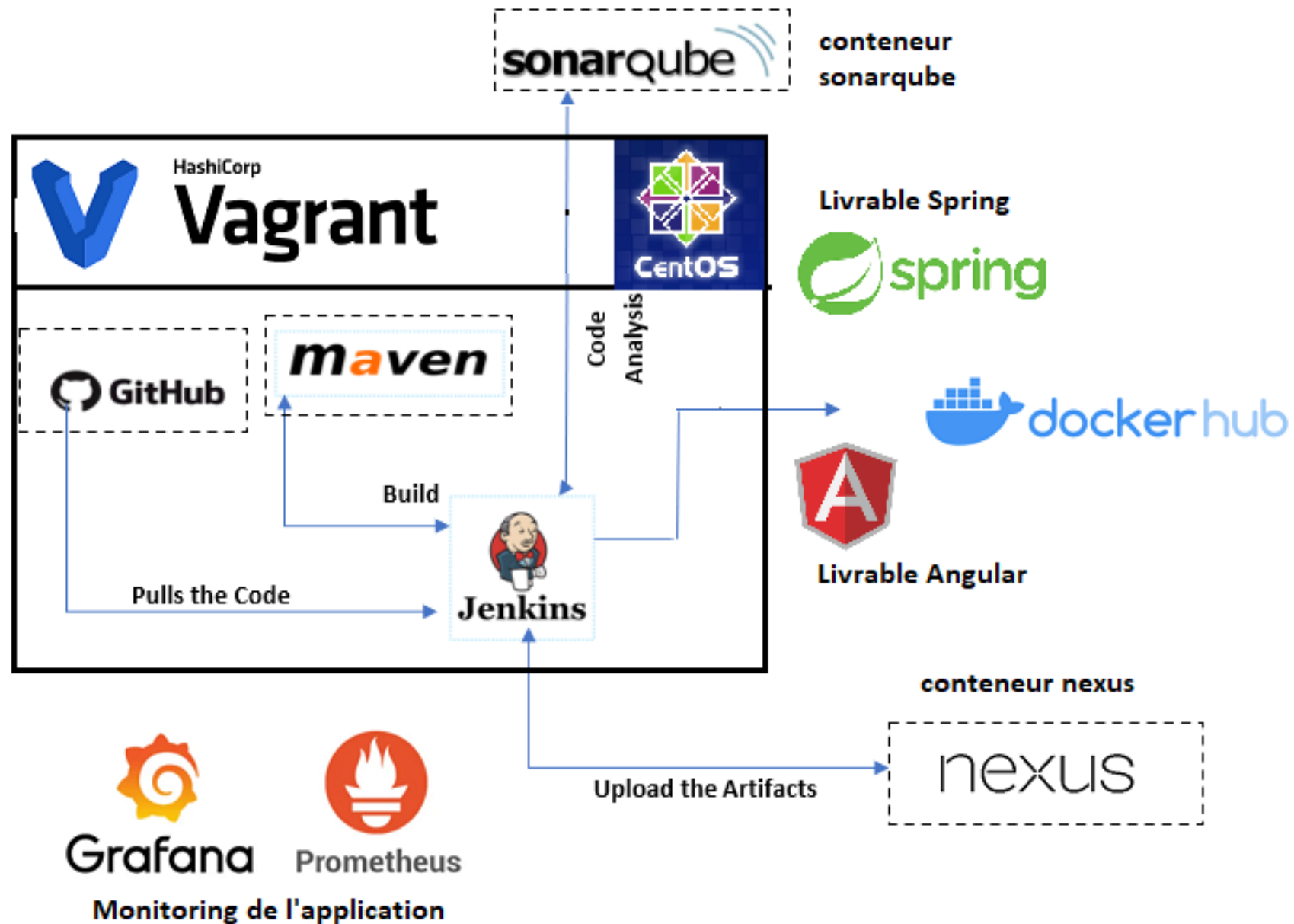
# Outils : Grafana / Prometheus

## 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.pdf» (voir [Drive du cours](#)), pour installer :

- Virtual box  **VirtualBox**
- Vagrant  **VAGRANT**
- Une machine virtuelle Ubuntu dans Vagrant 

# Introduction DevOps

