

A construction worker wearing a yellow hard hat and safety harness is working on a large, complex steel reinforcement grid. The grid consists of numerous intersecting bars of different sizes, forming a dense mesh. The worker is positioned in the upper right quadrant of the frame, looking down at the work. The background shows more of the same reinforcement grid extending across the entire image.

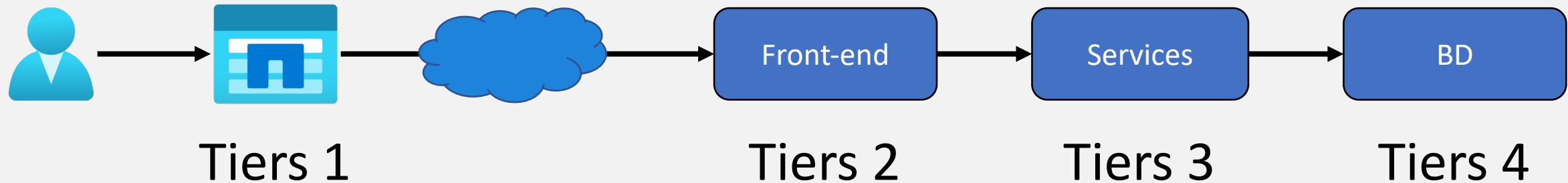
Module 01 - Introduction

Plan de la présentation

- Les infrastructures
- Modèles de déploiement
- DevOps
- Les outils requis pour les cours

Modèles de déploiement applicatif

Modèle d'application utilisé en exemple (4-tiers)

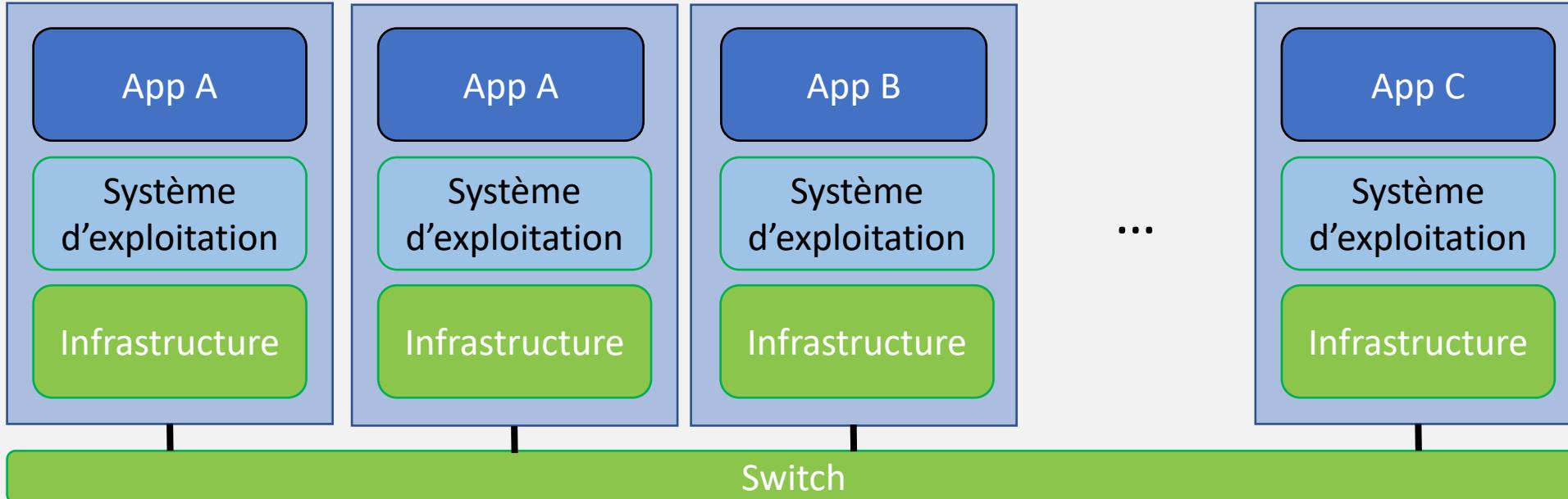


Modèle 1 : Physique les applications s'exécutent sur une ou des machines physiques.

Modèle 2 : Virtualisation, les applications s'exécutent sur des machines virtuelles

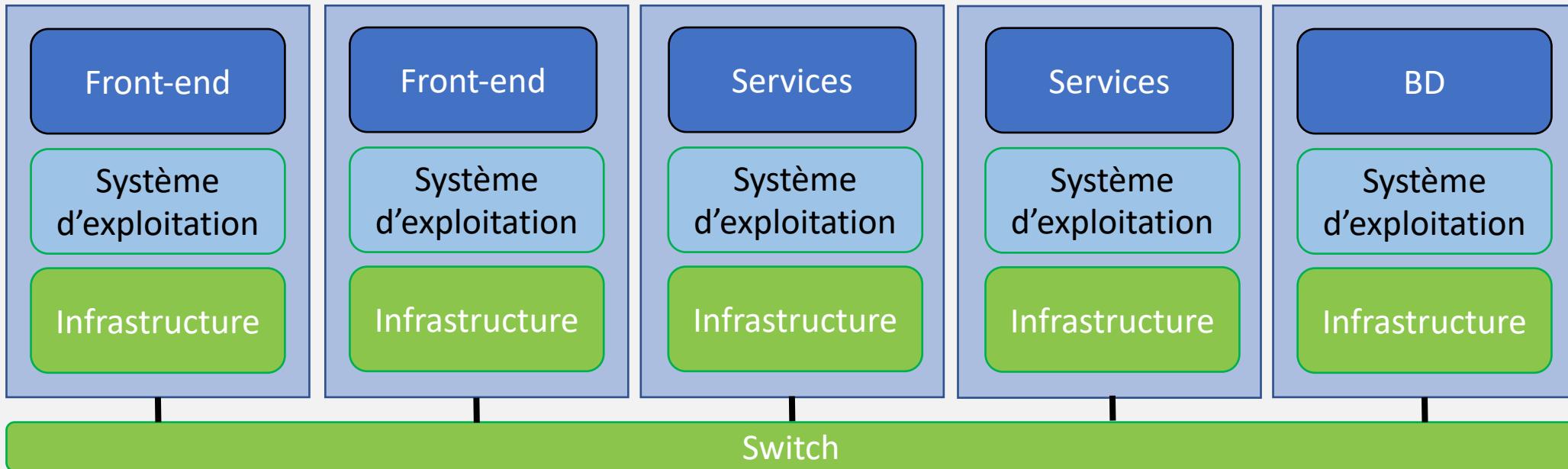
Modèle 3: Conteneurs, les applications s'exécutent sur un ou des conteneurs.

Modèle 1 – Physique



- Une application est installée par machine physique
- Les applications sont isolées
- Coût élevé / perte de ressource
- Complexité de gestion des dépendances
- Long à déployer

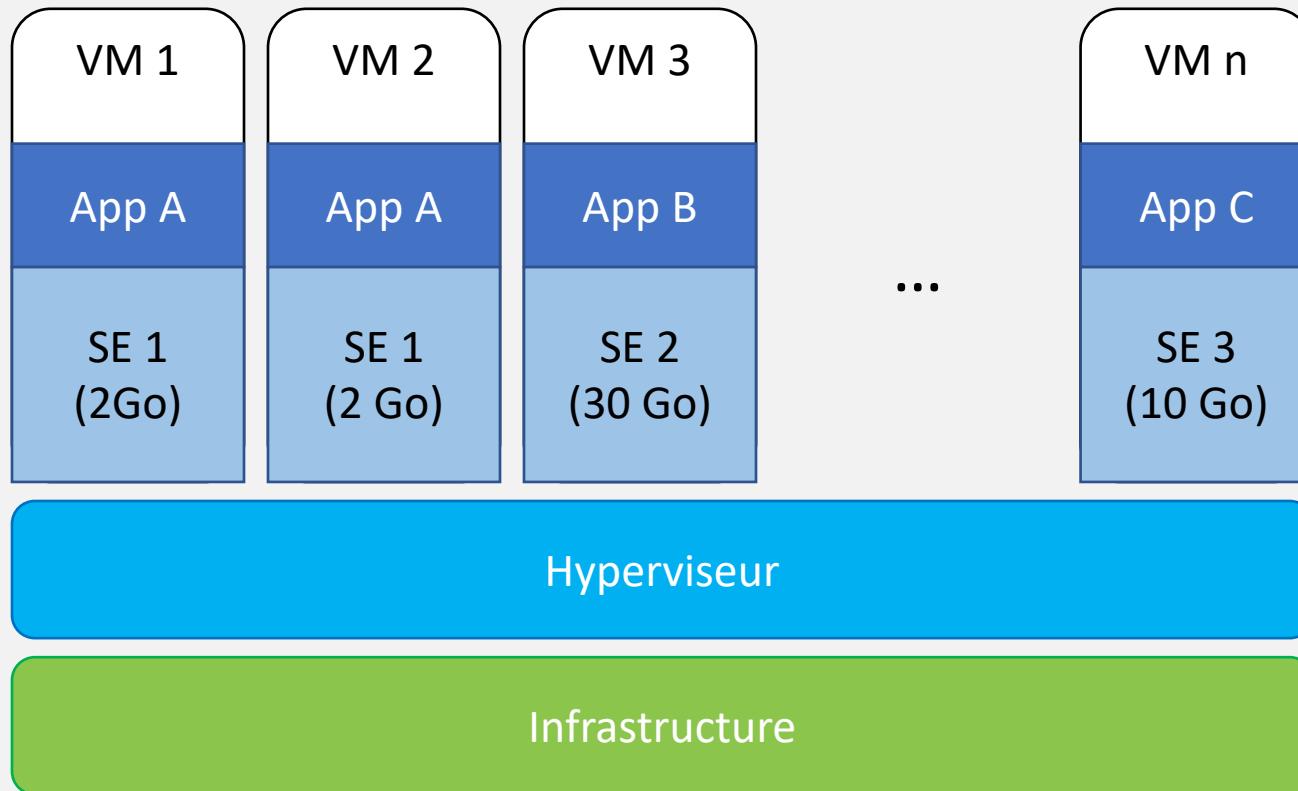
Modèle 1 – Physique



Automatisation :

- Scripts (Puppet, chef, ansible)
- Packages
- Etc.

Modèle 2 – Virtualisation des machines physiques



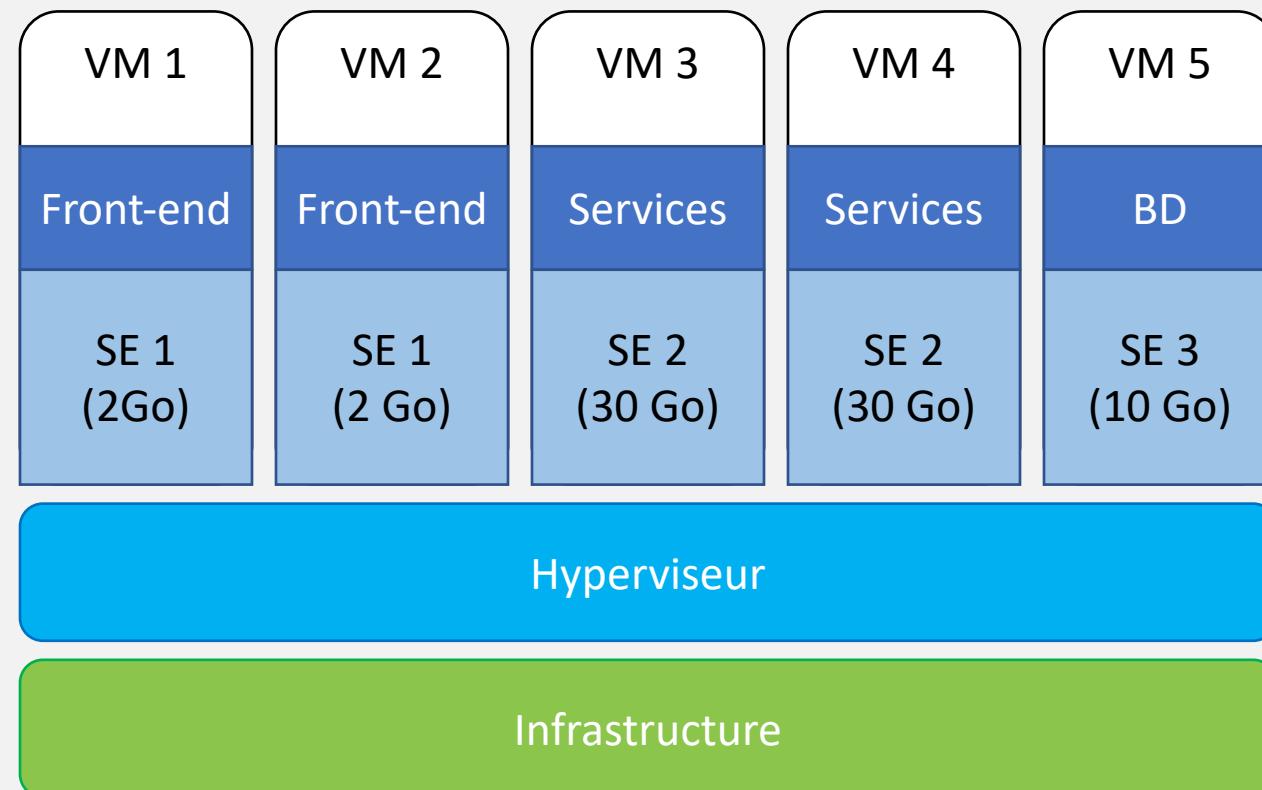
1 hyperviseur par hôte :

- n machines virtuelles (VM) avec des systèmes d'exploitation différents
- Par VM :
 - Un système d'exploitation complet (x Go)
 - Une instance de l'application / service

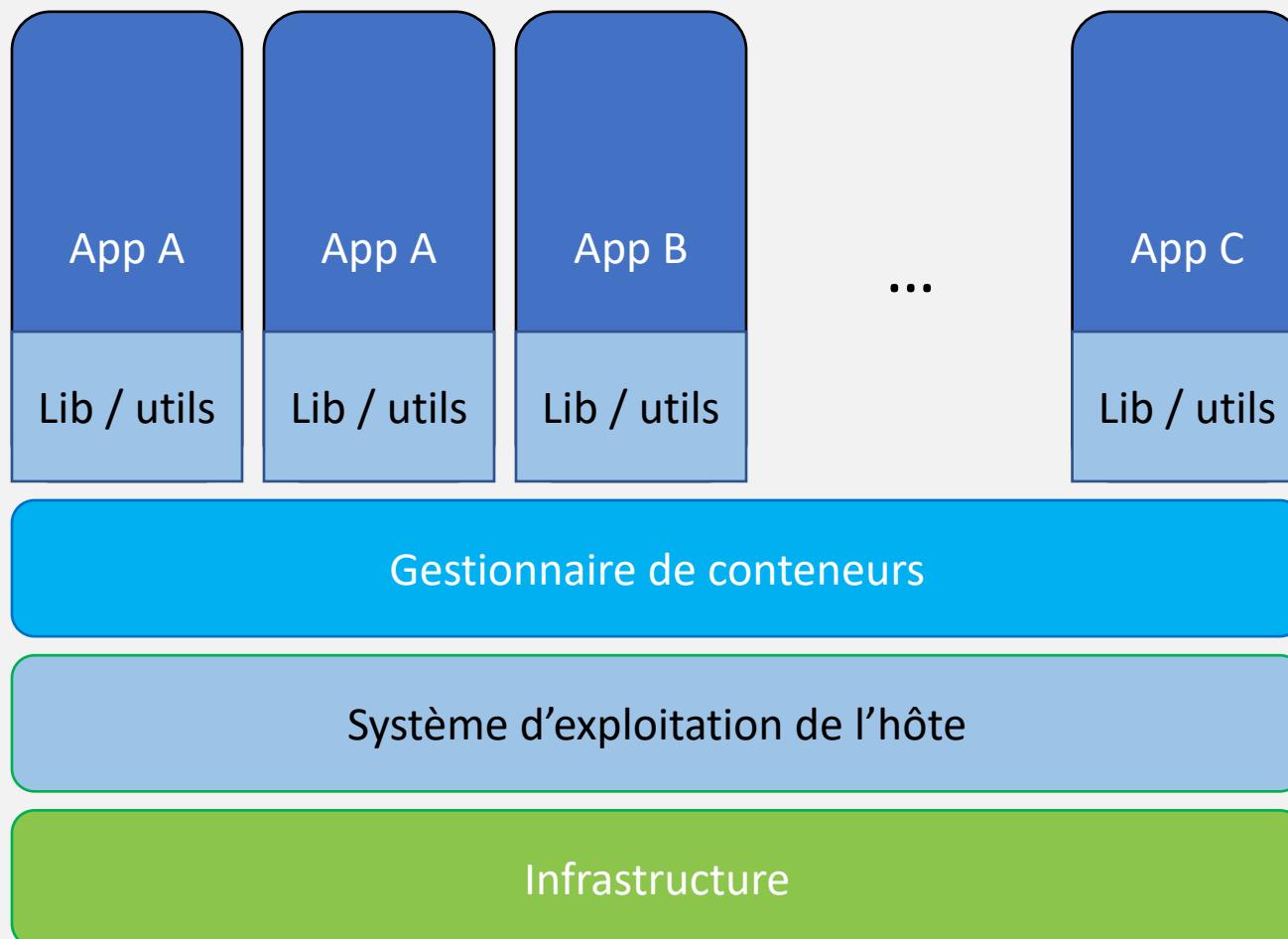
Idées :

- Déplacer les VMs d'un hôte à un autre
 - Répartir la charge
 - Reprise en cas de défaillances matérielles
- Isoler les applications
 - Si dépassement de ressource : peu d'impacts sur les autres applications
 - Si piratage : reste consigné à la VM
- Mise à l'échelle verticale et horizontale facile

Modèle 2 – Virtualisation des machines physiques



Modèle 3 – Virtualisation des applications / services



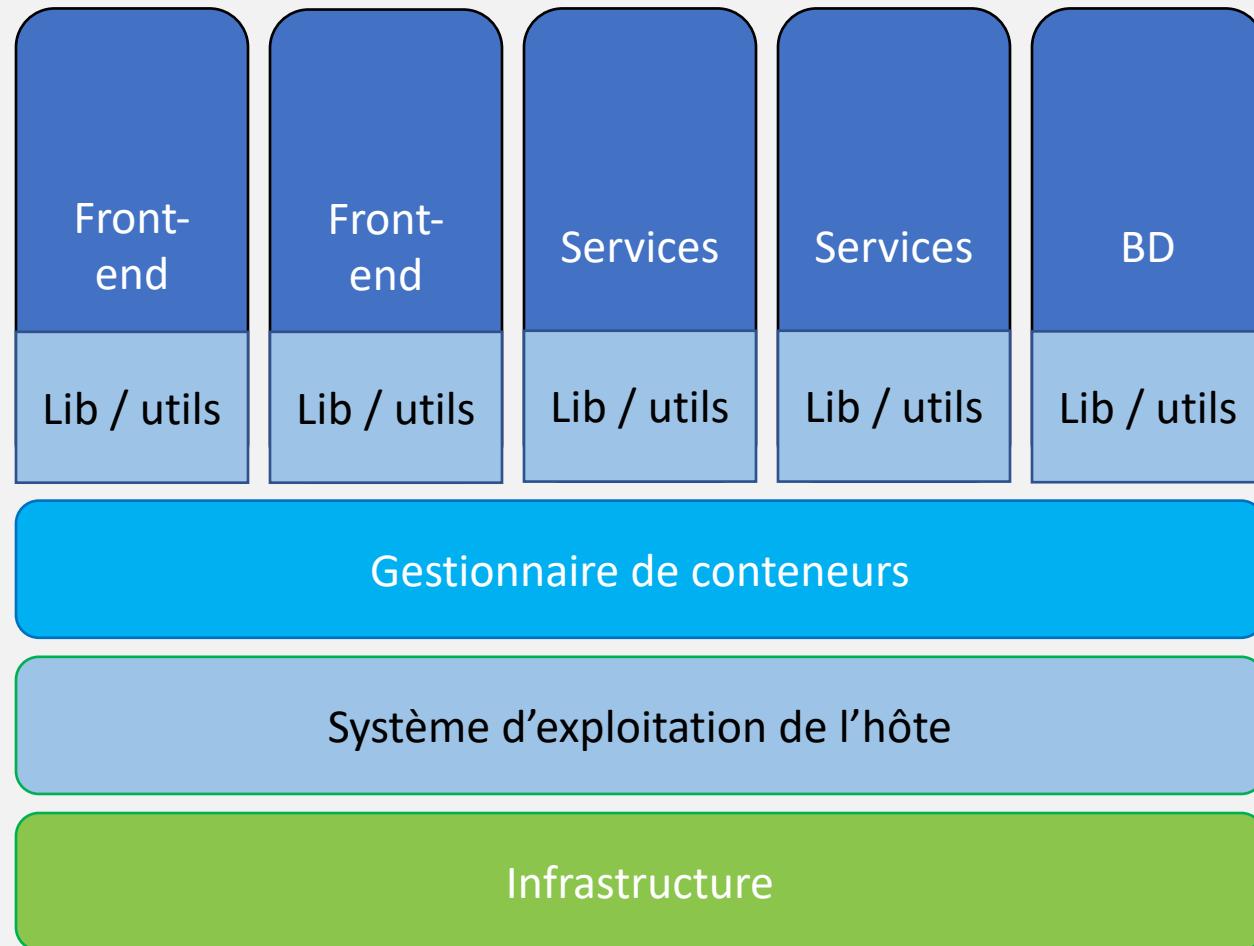
1 hôte :

- 1 système d'exploitation
- Plusieurs conteneurs / hôte
- 1 application / conteneur (= instance d'une image)

Idées :

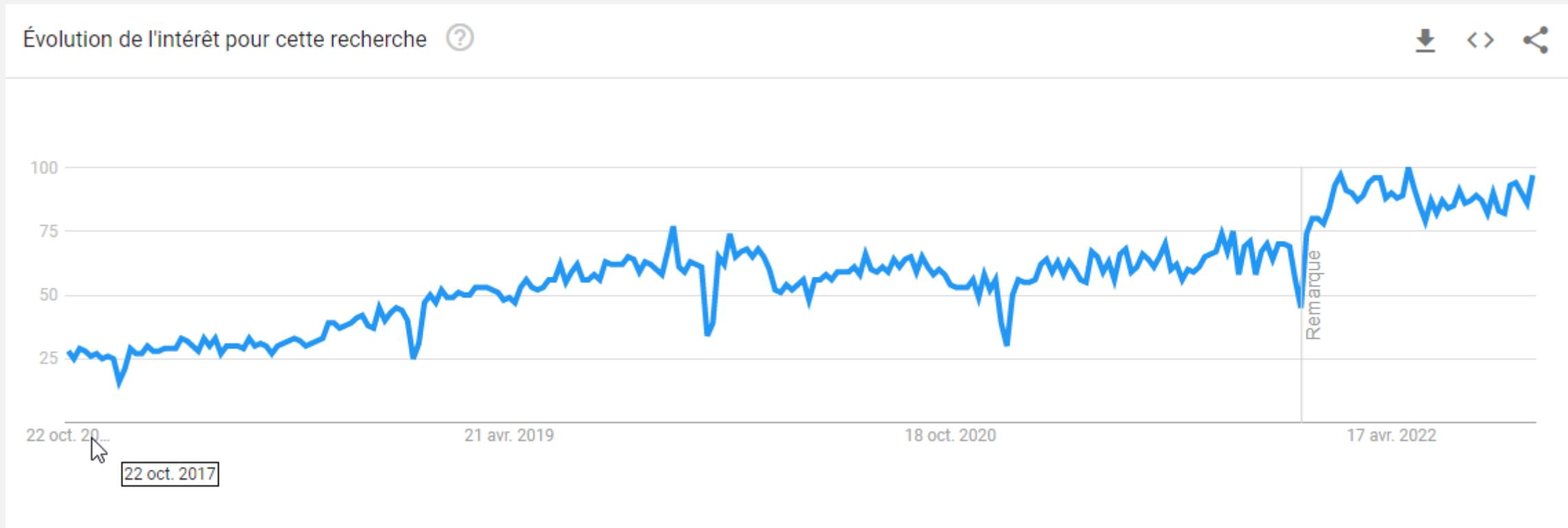
- Déplacer les conteneurs d'un hôte à un autre
 - Répartir la charge
 - Reprise en cas de défaillances matérielles
- Isoler les applications
 - Si dépassement de ressource : peu d'impacts sur les autres applications
 - Si piratage : reste consigné au conteneur
- Mise à l'échelle verticale et horizontale facile
- Léger : le conteneur ne contient que les dépendances de l'application et l'application (quelques Mo), le conteneur ne contient pas le système d'exploitation

Modèle 3 – Virtualisation des applications / services



Qu'est-ce que le DevOps

Contraction de
DEVelopement + OPperationS



Évolution des recherches sur le mot clé [DevOps](#), voire les recherches associées.

Historique

- Premières applications :
 - Taille limitée
 - Autonomes : n'avaient que peu de besoins d'interagir avec d'autres applications
 - L'équipe qui développait l'application devait aussi la maintenir

Les équipes traditionnelles

Développement

Objectifs :

- Évolutions
- Nouvelles fonctionnalités

Opérations

Objectifs :

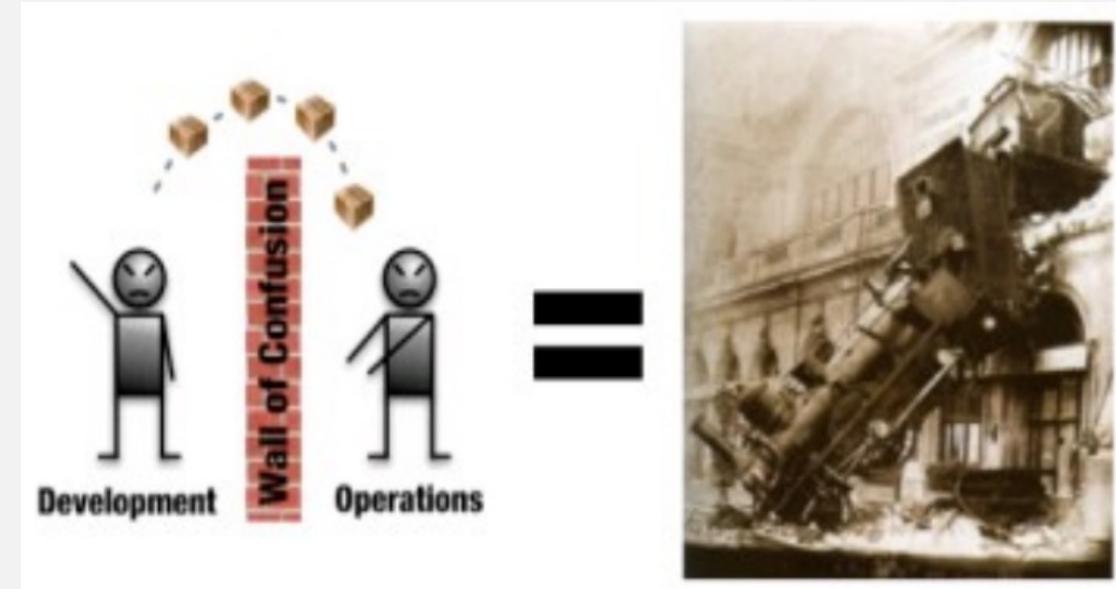
- Stabilité
- Qualité de service

Historique

- Des systèmes de plus en plus gros :
 - Apparition des progiciels de gestion intégré (PGI) ou Enterprise Ressources Planning (ERP)
 - Applications ont besoins de communiquer avec d'autres applications / systèmes
- ⇒ Besoin d'être efficace et séparation des tâches plus développement et plus infrastructure

Historique

- Trois enjeux en gestion de projets :
 - Coût
 - Objectifs
 - Délais
- Opérateurs : Stabilité
 - Augmenter la qualité
 - Au détriment du coût et du temps
- Développeurs : Livrer
 - Ajouter / modifier des fonctionnalités vite et à faible coût
 - Au détriment de la qualité



<https://fr.slideshare.net/rtang03/dev-ops-storyboardv1>

Les besoins

Développement

Objectifs :

- IDE
- Contrôle de version
- Générer des livrables
- Tester

Opérations

Objectifs :

- Publier
- Tester
- Surveiller

Planification et suivi :

- Méthodologie Agiles : Scrum / Kanban
- Tableau de bord et rapports

Ce qu'est le DevOps

Ensemble de **pratique** et **moyens** mis en œuvre afin **d'unifier et optimiser** le travail entre les équipes de **développement** et de **production**

- Une philosophie
- Un nouveau mode de pensée basé sur les méthodologies agiles



DevOps

- Contraction de « développement » et d'« opération »
- Idée, c'est de faire collaborer les deux mondes, de rapprocher les équipes
⇒ ce n'est donc pas un outil !
- Pratiques DevOps
 - Utilisation des méthodes **agiles** pour le développement mais aussi pour l'exploitation
 - Automatiser le plus possible :
 - Création des artefacts
 - Tests (qualité ↗, coût des erreurs ↘)
 - Infrastructure
 - Installation
 - Boucle de rétroaction :
 - Livraison en continue : fréquence ↗, maîtrise du processus de déploiement ↗
 - Livrer de petits incrément : difficulté ↘
 - Livrer souvent : pratique ↗
 - Avoir une rétroaction rapide : ajustement ↗, qualité ↗

En ingénierie logicielle, les *pratiques agiles* mettent en avant la collaboration entre des équipes auto-organisées et pluridisciplinaires et leurs clients

DevOps

Build an Open-Source DevOps Toolchain

Repository:	Build/Integration:
<ul style="list-style-type: none">▪ JFrog▪ Artifactory▪ Sonatype Nexus OSS▪ Git (on-premises)▪ GitLab CE▪ Bit Component Repo	<ul style="list-style-type: none">▪ Jenkins▪ CloudBees (Jenkins)▪ Apache Maven▪ Gradle▪ CruiseControl▪ Travis CI▪ XebiaLabs (Hudson)
IDE:	Visual Studio Code
<ul style="list-style-type: none">▪ Eclipse▪ CircleCI▪ Electric Cloud	<ul style="list-style-type: none">▪ JetBrains IntelliJ
Testing:	Bug Tracking:
<ul style="list-style-type: none">▪ Appium▪ Jmeter▪ Protractor▪ Selenium▪ Visual Studio Test Platform▪ Watir	<ul style="list-style-type: none">▪ Mantis▪ Bugzilla▪ Redmine▪ Trac
Security:	Code Review:
<ul style="list-style-type: none">▪ Chaoslinger▪ Hashicorp Vault▪ OWASP ZAP▪ OSSEC	<ul style="list-style-type: none">▪ Gerrit▪ Review Board▪ Sonarqube
	Compliance Verification:
	<ul style="list-style-type: none">▪ Inspec▪ Compliance Masonry

ID: 378544

The Jenkins interface shows a build log titled "Console Output". The log includes entries like "Skipping 1.196 KB.. Full Log" and "TASK [smtp-server : include] **** skipping: [teamwiesn.com]".

The Visual Studio Team Services interface displays a "Test plan" for the "FabrikamFiber" project. It shows a list of test cases under "Sprint 3: Web Team (Id: 876)". One test case is highlighted: "867 : [Login improvements] As a service engineer I can schedule a login session". Below the list, there is a table of results:

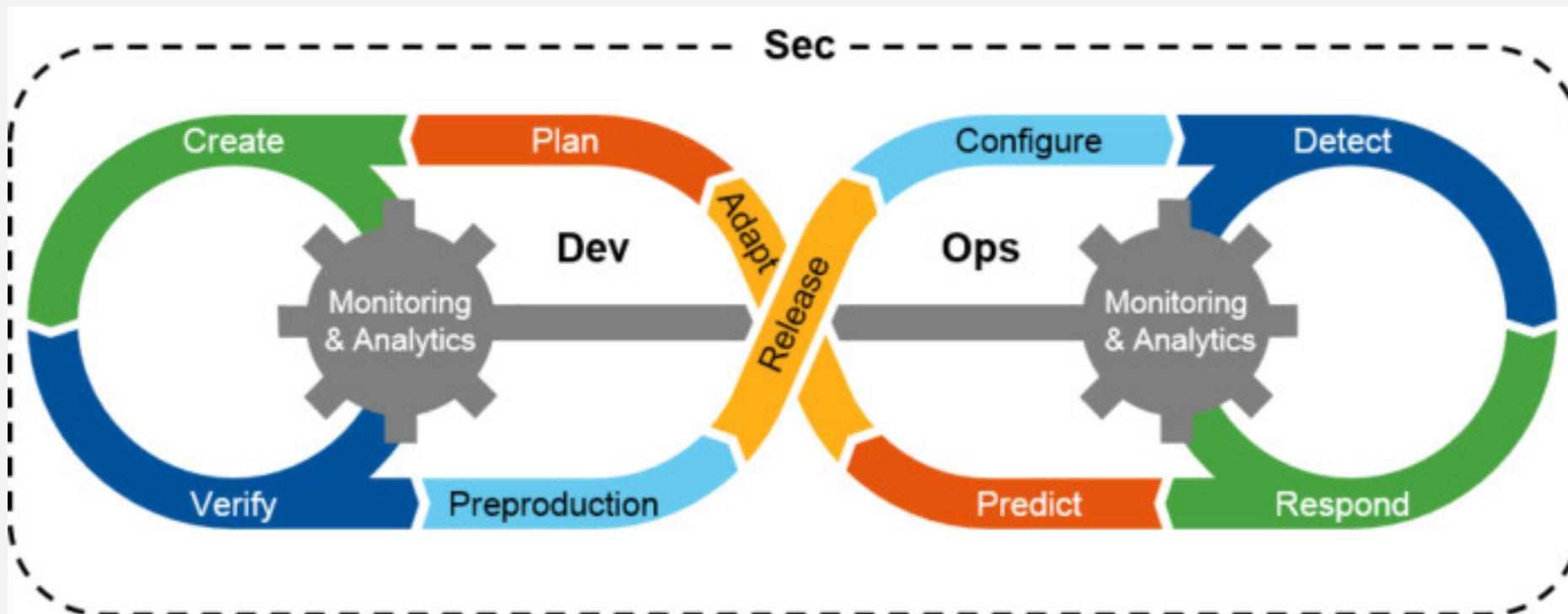
Outcome	ID	Title
Passed	999	Auto fill must not work after clearing
Failed	1000	Sign in with auto-filled credentials in
Passed	997	First login with 'Remember password'
Failed	998	Repeat login must auto fill user name

Gestion de cas de test

Rationalisez les livraisons de qualité

Accédez à Azure Test Plans, qui fait partie d'Azure DevOps, disponible en tant que service cloud managé ou en local. Coordonnez toutes les activités de gestion de test, notamment la planification, la création, l'exécution et le suivi, le tout à partir d'un emplacement central, ou à partir de tableaux Kanban avec des fonctionnalités de qualité incluses. Le concentrateur de test offre aux propriétaires de produits et aux analystes d'entreprise des données approfondies de la progression par rapport aux critères d'acceptation définis et aux mesures de qualité.

DevSecOps



Intégration continue / déploiement continu

- L'intégration continue / Continuous integration (CI) :
 - Générer des artefacts à partir d'un gestionnaire de source / source control management (SCM)
 - Exécuter les tests
- Le déploiement continu / continuous delivery (CD) :
 - Prendre les artefacts
 - Les déploiements dans les environnements ciblés
 - Souvent, il y a un flux de travail pour migrer sur plusieurs environnements avec des validations à chaque niveau

On parle souvent de CI/CD

Intégration continue - principes

L'intégration continue consiste à avoir un ensemble de processus automatisés qui s'enclenchent lorsque l'équipe de développement pousse de nouvelles choses.

Dans le jargon du développement, on dira qu'on « **build et test lors des commits** »

Parmi ces processus, on va également automatiquement mettre à jour l'avancée dans le sprint courant.

Automatisme ne veut pas dire Perte de contrôle, il est possible de débrayer certains comportements ou d'avoir une ou plusieurs validations manuelles.

Les processus d'intégration continue doivent être effectués sur des environnements neutres

Déf. : L'intégration continue ou *Continuous Integration (CI)* en anglais, est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée.

Intégration continue - objectifs

S'assurer que la production actuelle ne comporte pas de régression.

S'assurer que la version actuelle est dans un état livrable et compatible

Maintenir à jour l'avancée du sprint et vérifier si les délais et objectifs sont tenus.

Avoir toujours à disposition une version pour tester, faire une démo ou livrer.

Intégration continue - avantages

D'être réactif vis-à-vis du client (démonstration, livraison, test)

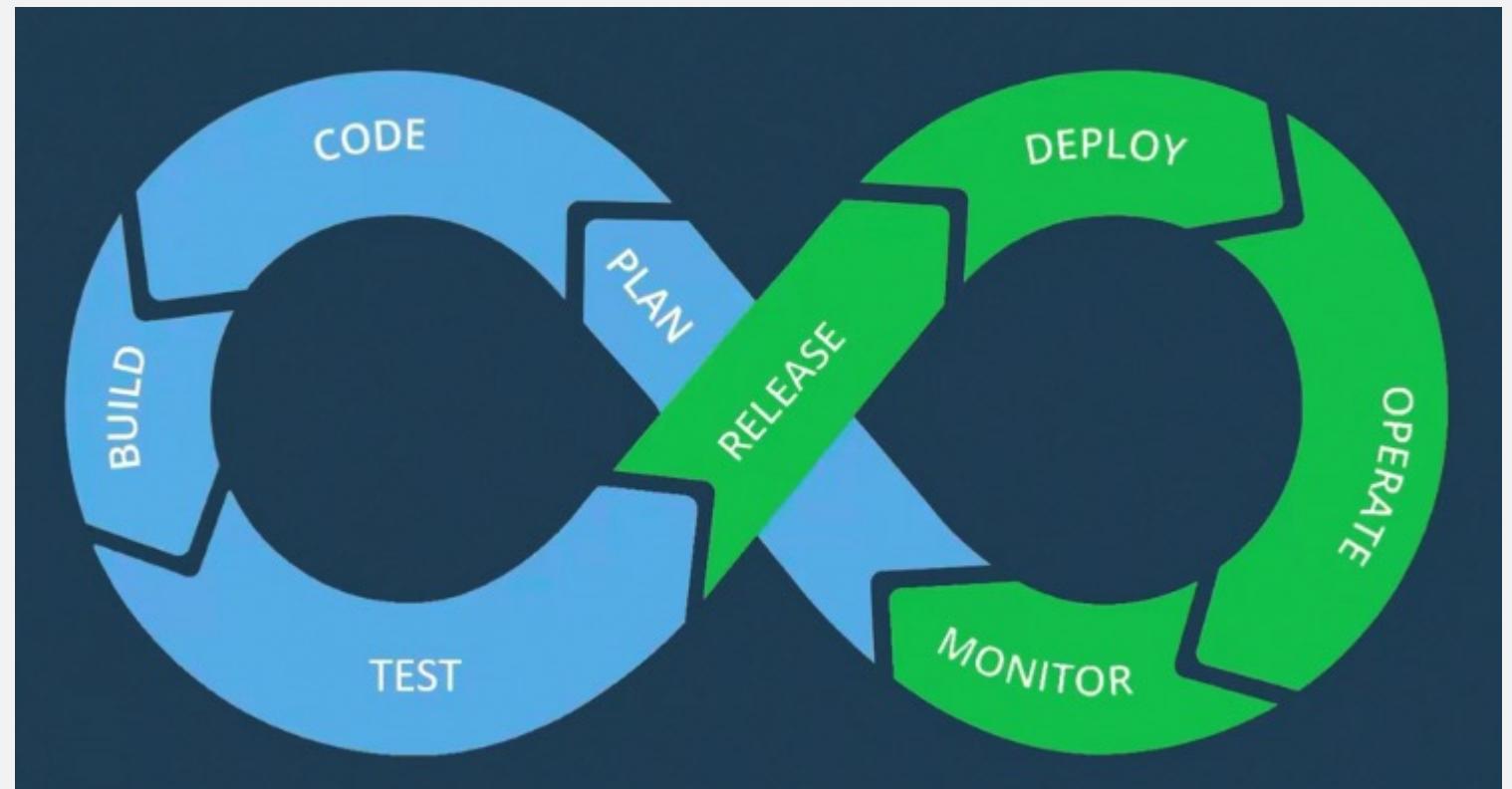
D'avoir un feedback immédiat sur le code effectué, grâce aux tests automatisés et à la mise à jour automatique du travail.

Éviter les problèmes de déploiement ou l'intégration de nouvelles ressources au projet.

Être en capacité de livrer une correction ou de relivrer n'importe quelle version de l'application car tout est historisé.

Les piliers de DevOps

- Planification Agile
- Automatisation de Builds et Déploiement (CI/CD)
- Tests automatisés
- Analyse statique et revues de code
- Monitoring



Ce que DevOps n'est pas :

- **Combattre les idées reçues :**
 - **Une méthodologie**
 - **Une solution logicielle**
 - **Une solution magique (quick-fix) pour régler tous les problèmes de votre organisation.**
 - **Un poste ou un rôle dans l'entreprise**

Les outils du DevOps

Plan	Code	Build	Test	Release	Deploy	Operate	Monitor
Gestion du backlog et planning Indicateurs et rapports Réunions d'équipe	Contrôle de source Stratégie de branches Revues de code	Automatisation de la Build Analyse statique de code Rapports de build	Gestion des campagnes de test Exécution tests unitaires, d'intégration et de charge Rapports de tests	Packaging et configuration Stratégie de version	Gestion des environnements Gestion de la configuration et des déploiements	Infrastructure et services cloud ou on-premise	Service de supervision Récolte et exploitation des logs Rapports et alertes



Les outils requis pour les cours

- Votre PC avec :
 - Git
 - Environnement de développement Visual Studio Code
 - Docker Desktop et Kubernetes
- GitHub
- Microsoft Azure Portal
- Azure DevOps Platform avec Azure Repos

Références :

- « Management Information systems.Chapter 6: IT Infrastructure and Platforms ». Consulté le 19 mai 2020.
<https://paginas.fe.up.pt/~acbrito/laudon/ch6/chpt6-1fulltext.htm#defineinfra>.
- (2020) IT Infrastructure, Wikipédia.
- DIDIERGEORGES, Philippe. *DevOps - Les clés pour le comprendre et le mettre en œuvre*. Web. Editions ENI, 2020.

Droit d'auteur

- Personne ayant contribué à la rédaction de ce document :
 - Jean-Pierre Duchesneau
 - Pierre-François Léon
 - Dernière version : 1.1.2 mars 2023.

Cette oeuvre, création, site ou texte est sous licence Creative Commons Attribution - Pas d'Utilisation commerciale - Partage dans les Mêmes Conditions 4.0 International. Pour accéder à une copie de cette licence, merci de vous rendre à l'adresse suivante \url{http://creativecommons.org/licenses/by-nc-sa/4.0/} \\ ou envoyez un courrier à Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.jEA