

# Rapport de projet de fin d'études

Administration Système, Réseaux et Cybersécurité

Licence Professionnelle Universitaire (LPU-ASRCS)

Effectué au sein de la société Capgemini

---

## Mise en place d'un système de supervision à base des solutions CI/CD en mode DevOps

---

Élaboré par :

Mr. Achraf AZOUAGH

Encadrant technique :

Mr. Amine AAKIL

Encadrant Pédagogique :

Mr. Khalid BOURAGBA

---

*Dédicaces*

---

Je dédie ce modeste travail :

À Mes chers parents, source de vie, d'amour et affection. Que Dieu leur procure bonne santé et longue vie.

À Mes professeurs pour ce qu'ils m'ont appris et pour la patience dont ils ont fait.

À Mes amis pour les moments inoubliables qu'on a partagés, et pour leurs aides et supports dans les moments difficiles.

---

## *Remerciements*

---

Ce chapitre peut être l'occasion d'exprimer une gratitude sincère envers les personnes qui ont apporté une aide, une écoute ou simplement une chaleur gratuite et généreuse.

Je tiens à exprimer ma profonde gratitude et reconnaissance envers **M. Anass ZAILAF** pour ces efforts déployés afin que je puisse aborder un stage chez l'équipe Cloud.

Je tiens à remercier tout particulièrement **M. Amine AAKIL**, mon maître de stage, qui m'a recommandé comme stagiaire et qui a su me laisser une réelle autonomie, tout en me guidant et en m'apportant l'aide et les moyens nécessaires au bon déroulement de mon stage.

Je tiens également à adresser mes vifs remerciements à **M. Redouane LAABOULI** pour m'avoir accordé l'opportunité de découvrir et réaliser un projet aussi innovant au sein de l'équipe « CloudBox ».

Mes remerciements sincères vont également à tout le personnel de Capgemini, pour leur accueil chaleureux, leur bonne humeur quotidienne, leur capacité d'équipe exemplaire ainsi que leur présence quand j'avais besoin de leur aide

Je ne peux pas oublier le grand support de mon respectueux professeur et encadrant interne **M. Khalid BOURAGBA** qui m'a apporté beaucoup de connaissances pas précisément dans la durée du stage mais aussi en ces dernières 3 années.

Bien évidemment, je remercie ma famille et mes amis qui tout au long de ce stage mais aussi tout au long de mes études ont su me soutenir et m'accompagner dans tous ces moments de doute mais aussi dans tous ces moments de joies. Merci à tous !

---

## Préambule

---

Ce document synthétise les enjeux, la démarche et les résultats du travail effectué à la société Capgemini TS au sein de l'équipe CloudBox dans le cadre du projet de fin d'études à l'École Supérieure de Technologie de Casablanca pour l'obtention du diplôme de Licence Professionnelle en Administration Système, Réseaux et Cybersécurité.

Mon projet s'intitule « Mise en place d'un système de supervision et alerte à base des solutions d'intégration et de déploiement continues en mode DevOps ». Le projet est composé de deux volets. Le premier concerne l'automatisation des tests pour des portails web. Ensuite, le deuxième englobe la création d'une solution de monitoring de ces tests et la gestion des anomalies.

Ce travail a pour objectifs : diminuer le coût, détecter plus rapidement les dysfonctionnements de l'application et contribuer à l'amélioration continue. Et finalement, intégrer un outil de monitoring des tests effectués afin d'alerter en cas d'anomalie, ce qui favorise la haute disponibilité.

Pour ce faire, l'équipe CloudBox a décidé de s'outiller de Robot Framework pour l'automatisation des tests. Ensuite, Jenkins pour l'intégration continue. Et finalement, Grafana comme outil de monitoring & alerting.

Pour y parvenir et répondre aux différentes exigences, on a opté pour la méthode DevOps. En premier lieu une formation générale et une étude a été faite afin de se familiariser avec l'environnement technique et d'appréhender les outils requis.

**Mots-clés :** DevOps, CI/CD, Automatisation, Robot Framework, Jenkins, Docker, Grafana.

---

## *Abstract*

---

This document summarizes the issues, the approach and the results of the work done at Capgemini Technology Services within the CloudBox team as part of the end-of-studies project at the Superior School of Technology of Casablanca for the diploma of Professional Degree in Systems Administration, Networks and Cybersecurity.

My project is entitled "Implementation of a monitoring and alert system based on CI/CD solutions whilst taking a DevOps approach". The project is composed of two parts. The first one concerns the automation of tests for web portals. Then, the second one includes the creation of a monitoring solution for these tests and the management of anomalies.

The objectives of this work are to reduce the cost, detect malfunctions in the application more quickly and contribute to the continuous improvement plan. And finally, to integrate a monitoring tool for the tests carried out in order to alert in case of anomaly, which favors the stability of the solution.

To do this, the CloudBox team decided to use the Robot Framework for test automation. Then, Jenkins for continuous integration. And finally, Grafana as a monitoring and alerting tool.

To achieve this and meet the various requirements, we opted for the DevOps approach. First of all, a general training and a study was done in order to get familiar with the technical environment and to understand the required tools.

**Keywords :** DevOps, CI/CD, Automatisation, Robot Framework, Jenkins, Docker, Grafana.

---

## *Table des matières*

---

<b><u>DEDICACES</u></b>	<b>2</b>
<b><u>REMERCIEMENTS</u></b>	<b>3</b>
<b><u>PREAMBULE</u></b>	<b>4</b>
<b><u>ABSTRACT</u></b>	<b>5</b>
<b><u>TABLE DES MATIERES</u></b>	<b>6</b>
<b><u>INTRODUCTION</u></b>	<b>10</b>
<b><u>CHAPITRE I : CONTEXTE</u></b>	<b>11</b>
<b>1. PRESENTATION GENERALE DE L'ORGANISME D'ACCUEIL :</b>	<b>12</b>
A. HISTORIQUE :	13
B. DOMAINES D'EXPERTISE :	14
C. SECTEURS D'ACTIVITE :	14
D. VALEURS :	15
E. METIERS :	17
F. CHIFFRES :	18
G. CAPGEMINI MAROC :	21
<b>2. ENVIRONNEMENT DE TRAVAIL :</b>	<b>22</b>
A. CLOUDBOX :	22
B. CERTIFICATIONS :	24
<b><u>CHAPITRE II : ÉTUDE THEORIQUE AUTOUR DU CLOUD &amp; DEVOPS</u></b>	<b>26</b>
<b>1. CLOUD COMPUTING :</b>	<b>27</b>
A. DEFINITION :	27
B. HISTOIRE :	27
C. TYPES DES SERVICES CLOUD :	28
D. TYPES DE DEPLOIEMENTS CLOUD :	30
E. AVANTAGES DU CLOUD COMPUTING :	32
F. INCONVENIENTS DU CLOUD COMPUTING :	33
G. LES FOURNISSEURS CLOUD :	34

H.	CLOUD ET VIRTUALISATION :	35
<b>2.</b>	<b>DEVOPS :</b>	<b>36</b>
A.	CONTEXTE ET HISTOIRE :	36
B.	PRESENTATION :	37
C.	FONCTIONNEMENT :	37
D.	CI/CD PIPELINE :	38
E.	AVANTAGES :	40
F.	DEVOPS & CLOUD :	41
<b>3.</b>	<b>TESTING :</b>	<b>41</b>
A.	TYPE DE TESTS :	41
B.	TESTS MANUELS VS TESTS AUTOMATISES :	42
<b>4.</b>	<b>DESCRIPTION DE L'AUTOMATISATION :</b>	<b>44</b>
A.	OBJECTIF DE L'AUTOMATISATION :	45
B.	L'APPORT DE L'AUTOMATISATION :	45
<b>5.</b>	<b>CONTENEURISATION :</b>	<b>46</b>
A.	CONTEXTE ET DEFINITION :	46
B.	FONCTIONNEMENT :	46
C.	AVANTAGES :	47
D.	CONTENEURS ET MICROSERVICES :	47
E.	CONTENEURISATION VS VIRTUALISATION :	48
<b>6.</b>	<b>MONITORING ET ALERTING :</b>	<b>49</b>
A.	DEFINITION ET CONTEXTE :	49
B.	PROCESSUS ET FONCTIONNEMENT :	49
C.	AVANTAGES DE LA SUPERVISION :	52
<b>7.</b>	<b>CONCLUSION :</b>	<b>52</b>
<b>CHAPITRE III : ANALYSE DES BESOINS ET JUSTIFICATION DES CHOIX</b>		<b>53</b>

<b>1.</b>	<b>ANALYSE DES BESOINS :</b>	<b>54</b>
A.	BESOINS FONCTIONNELS :	54
B.	BESOINS NON FONCTIONNELS :	54
<b>2.</b>	<b>CONCEPTION DES TESTS :</b>	<b>55</b>
A.	DIAGRAMMES DE SEQUENCE :	55
<b>3.</b>	<b>CONDUITE DU PROJET :</b>	<b>56</b>
A.	METHODE AGILE :	56
<b>4.</b>	<b>PRESENTATION DE LA SOLUTION :</b>	<b>58</b>
A.	ARCHITECTURE GLOBALE :	58
B.	ETAPES DE LA REALISATION :	59
<b>5.</b>	<b>Outils, approches, concepts utilisés et justifications :</b>	<b>59</b>
A.	GESTION DE PROJET :	59
B.	LES FOURNISSEURS CLOUD :	59
C.	EDITEUR DE CODE :	60
D.	AUTOMATISATION DES TESTS :	61
E.	GESTION DE CODE SOURCE :	63
F.	CONTENEURISATION :	64
G.	INTEGRATION CONTINUE :	65
H.	OUTIL DE VISUALISATION ET ALERTING :	67

**CHAPITRE IV : REALISATION DU PROJET**

70

<b>1. INTRODUCTION :</b>	<b>71</b>
<b>2. PROCESSUS DES CAS DE TESTS AUTOMATISES :</b>	<b>71</b>
A. TEST SUR L'INTERFACE LOGIN :	72
B. TEST SUR FORMULAIRE DU DOSSIER AVEC PRIME :	73
C. TEST SUR VISUALISER MES DOSSIERS :	74
D. TEST SUR AIDE & FAQ :	74
E. LA DECONNEXION :	75
<b>3. CONTENEURISATION :</b>	<b>75</b>
A. ARCHITECTURE :	75
B. DOCKERFILE :	77
C. SCRIPT BASH :	79
<b>4. MONITORING :</b>	<b>79</b>
<b>5. INTEGRATION DE JENKINS :</b>	<b>84</b>
A. CONFIGURATION DE L'ENVIRONNEMENT DE JENKINS :	84
B. PREPARATION A L'INTEGRATION DES TESTS AVEC JENKINS :	86
C. RAPPORT DES TESTS AU NIVEAU DE JENKINS :	90
<b>6. ALERTING :</b>	<b>90</b>
A. CREER UNE ALERTE GRAFANA :	91
B. ASSOCIER L'ALERTE A UN SERVICE DE MESSAGERIE :	92
<b>7. GESTION DE L'INCIDENT :</b>	<b>94</b>
<b>8. CONCLUSION :</b>	<b>94</b>
<b><u>CONCLUSION &amp; PERSPECTIVES</u></b>	<b>95</b>

---

## Liste des acronymes

---

Acronymes	Valeur
DevOps	Développement & Opérations
IaaS	Infrastructure en tant que Service
PaaS	Plateforme en tant que Service
SaaS	Logiciel en tant que Service
CI/CD	Intégration continue et déploiement continu
API	Interface de programmation d'application
CLI	Interface de ligne de commande
DevSecOps	Développement, Sécurité & Opérations
AWS	Amazon Services Web

---

## Introduction

---

Dans un monde où les technologies informatiques et logicielles évoluent de plus en plus vite, la livraison de tels projets pose un problème notamment dans un contexte Agile. Ceci est dû au fossé existant entre les équipes de développement et les équipes opérationnelles.

DevOps est un ensemble de pratiques qui met l'accent sur la collaboration et la communication entre les développeurs de logiciels et les professionnels des opérations informatiques, en automatisant le processus de livraison de logiciels et les changements d'infrastructure.

Ainsi, grâce à des études bien ciblées, les précurseurs de DevOps ont pensé à introduire une nouvelle manière de gérer les projets informatiques.

Le DevOps est devenu une culture que les entreprises essaient d'instaurer en interne, en raison de son efficacité et de sa capacité à réduire les délais et les coûts. Cette méthode repose essentiellement sur le développement, les tests, l'intégration, la mise en œuvre et la surveillance constante.

Mon entreprise d'accueil, Capgemini, se devait d'être à la pointe de la technologie, donc il fallait absolument incorporer cette méthode à sa politique d'entreprise. C'est dans ce contexte que s'inscrit mon projet de fin d'études qui consiste à instaurer les pratiques DevOps dans le cadre de mise en place d'un pipeline permettant le CI/CD et le déploiement d'un test fonctionnel automatisé muni par un système de supervision et d'alerte pour le client ENGIE. Ainsi, le rapport a pour objectif de détailler le travail réalisé dans le cadre de mon projet de fin d'études. Il est divisé en cinq chapitres mettant en évidence l'entreprise, l'approche choisie et le travail réalisé tous à la fois.

---

## *Chapitre I : Contexte*

---

**Ce chapitre est consacré à la présentation de Capgemini, en tant qu'organisme d'accueil de mon stage, le cadre général du projet et la conduite de ce dernier.**

## 1. Présentation générale de l'organisme d'accueil :

Capgemini est l'un des leaders mondiaux de l'industrie du conseil et des services informatiques, conduisant une stratégie de développement et de diversification qui a donné naissance à une croissance interne et externe. Elle a été créée par Serge KAMPF le 1er Octobre 1967 à Grenoble, en France, sous le nom de « Sogeti » (Société pour la gestion de l'entreprise et traitement de l'information).

Le groupe, comme beaucoup de ses concurrents, s'est constitué à travers de multiples acquisitions dans tous les secteurs d'activités liés aux services informatiques : consulting, intégration de systèmes, outsourcing. Près de 40 acquisitions (petites ou grandes entreprises) ont été réalisées en 40 ans. Avec ses clients, Capgemini conçoit et met en œuvre les solutions business et technologiques qui correspondent à leurs besoins et leur apporte les résultats auxquels ils aspirent. Capgemini revendique un style de travail qui lui est propre, la « Collaborative Business Expérience », et s'appuie sur un mode de production mondialisé, le « Right shore ».



## a. Historique :

Capgemini a été créée par Serge KAMPF le 1er Octobre 1967 à Grenoble, en France, sous le nom de « Sogeti » (Société pour la gestion de l'entreprise et traitement de l'information).

Le groupe, comme beaucoup de ses concurrents, s'est ensuite constitué à travers de multiples acquisitions dans tous les secteurs d'activités liés aux services informatiques : consulting, intégration de systèmes, outsourcing.



1967	• Fondation sous le nom de Sogeti à Grenoble par Serge KAMPF
1975	• Acquisition de CAP et Gemini Computer Systems
1989	• Fusion de CapGemini Sogeti France avec Sesa et Création de CAP Sesa
1990	• Création de Gemini Consulting et entrée en bourse de Paris
2000	• Acquisition et fusion avec ERNST & YOUNG Consulting
2002	• Evolution vers le consulting, Technology, Outsourcing et services
2004	• Changement de nom pour la dénomination actuelle : Capgemini
2007	• Création de Capgemini TS Maroc

## b. Domaines d'expertise :

Capgemini intervient au niveau des domaines suivants :

La réglementation, la production d'électricité (nucléaire, fossile, renouvelable), Les infrastructures électriques, gazière et activités réglementées. « Smart Energy », La fourniture d'énergie, Les services énergétiques, La distribution d'eau, la collecte et le traitement des eaux usées, La gestion de déchets, Les technologies propres, y inclus les énergies renouvelables.

## c. Secteurs d'activité :

Le groupe dispose d'une stratégie pertinente et d'un savoir-faire très solide pour développer ses relations clients dans plusieurs secteurs notamment :

- **Le Secteur public** : les équipes assurent la modernisation des services, tout en introduisant des réformes budgétaires significatives à l'attention des entreprises du service public, des administrations et des autorités locales ;
- **Services universels** : en tant que leader mondial en matière de systèmes informatiques appliqués aux réseaux et aux compteurs électriques intelligents, Capgemini aide les entreprises à maîtriser l'évolution permanente des réglementations ;
- **Services financiers** : Le groupe vise à simplifier les applications et les infrastructures informatiques des établissements financiers à ses clients pour leur permettre de garder une longueur d'avance en s'appuyant sur des modèles économiques innovants ;
- **Télécommunications** : Proposition des solutions innovantes reposant sur une connaissance approfondie du marché, ainsi que sur un savoir-faire technologique en termes de réseaux.

#### d. Valeurs :

Pour une culture de l'entreprise qui favorise un soutien individuel et mutuel des collaborateurs et une véritable collaboration, Capgemini s'appuie sur sept valeurs :

### Nos 7 valeurs



Solidarité



Honnêteté



Audace



Simplicité



Confiance



Plaisir



Liberté

au cœur de toutes nos actions

- **La solidarité** : la capacité à partager solidairement les bons et les mauvais moments.
- **La liberté** : ou encore la créativité, l'innovation, l'indépendance d'esprit et le respect des autres dans leurs cultures, leurs habitudes et leurs différences, ce qui est indispensable dans un Groupe présent dans plusieurs pays et plus d'une centaine de nationalités différentes.
- **Le plaisir** : sans lequel tout projet d'entreprise est difficile, voire impossible à réaliser.
- **La simplicité** : c'est la discrétion, la modestie réelle, le bon sens, l'attention portée aux autres et le soin mis à se faire comprendre d'eux, c'est la franchise des relations dans le travail, la décontraction, le sens de l'humour.
- **La confiance** : elle implique la volonté de responsabiliser les hommes et les équipes, et de confronter les responsables aux effets de leurs actions et décisions. La confiance implique également l'ouverture d'esprit et une grande transparence dans la circulation de l'information.
- **L'honnêteté** : le refus de toutes pratiques déloyales dans la conduite des affaires en vue d'obtenir un contrat ou un avantage particulier.
- **L'audace** : le goût d'entreprendre, de l'envie de prendre des risques et de s'engager, dans le respect d'un principe général de prudence et de lucidité sans lequel le manager audacieux peut se révéler dangereux.

## e. Métiers :

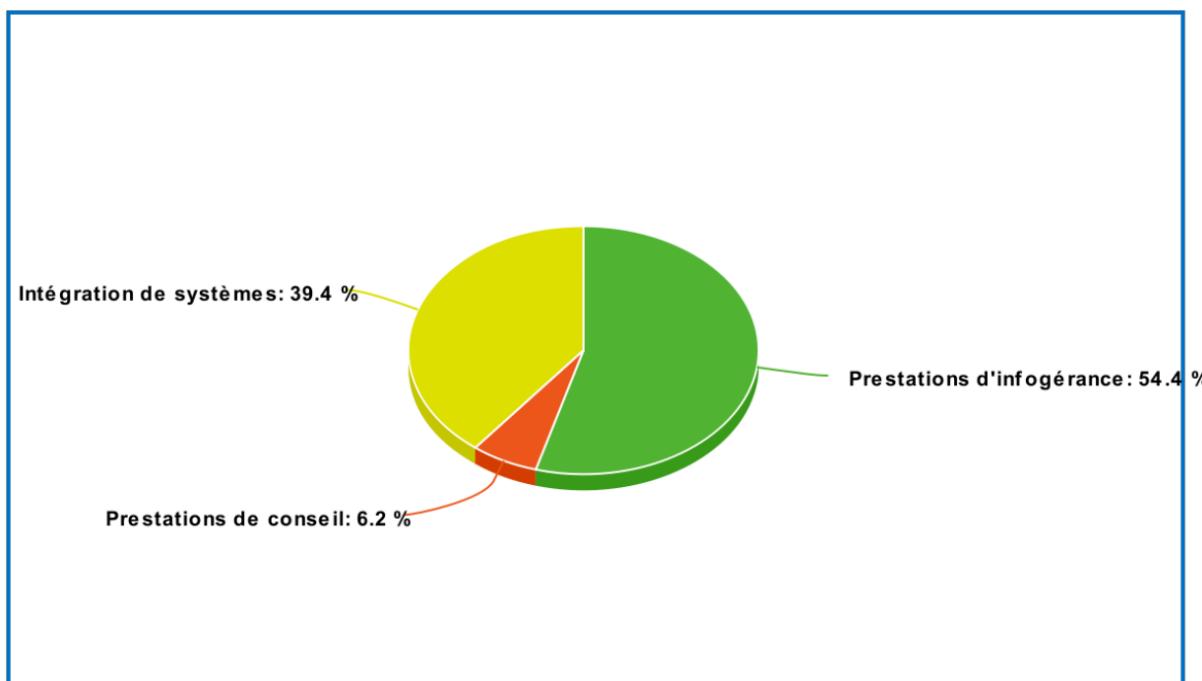
Capgemini est l'un des leaders mondiaux dans les domaines du conseil, des services informatiques, de l'infogérance et de l'outsourcing. Il définit ses métiers en quatre grandes catégories en voici une brève description de chacun :

- **Le Conseil (CS)** : apanage d'une entité spécifique (Capgemini Consulting), cette activité a pour objet d'aider les clients à identifier, structurer et exécuter les chantiers de transformation.
- **L'intégration de système (TS)** : il s'agit de concevoir, développer et mettre en œuvre tous projets techniques d'intégration de systèmes et de développement d'applications informatiques, du plus petit au plus grand.
- **L'Infogérance (OS)** : Capgemini guide et assiste ses clients dans l'externalisation totale ou partielle de leurs systèmes d'information et des activités s'y rattachant.
- **Les Services informatiques de proximité (LPS)** : assuré par une entité spécifique (Sogeti), ce métier a pour objet de fournir des services informatiques adaptés à des besoins locaux en matière d'infrastructures, d'applications, d'ingénierie et d'exploitation.

## f. Chiffres :

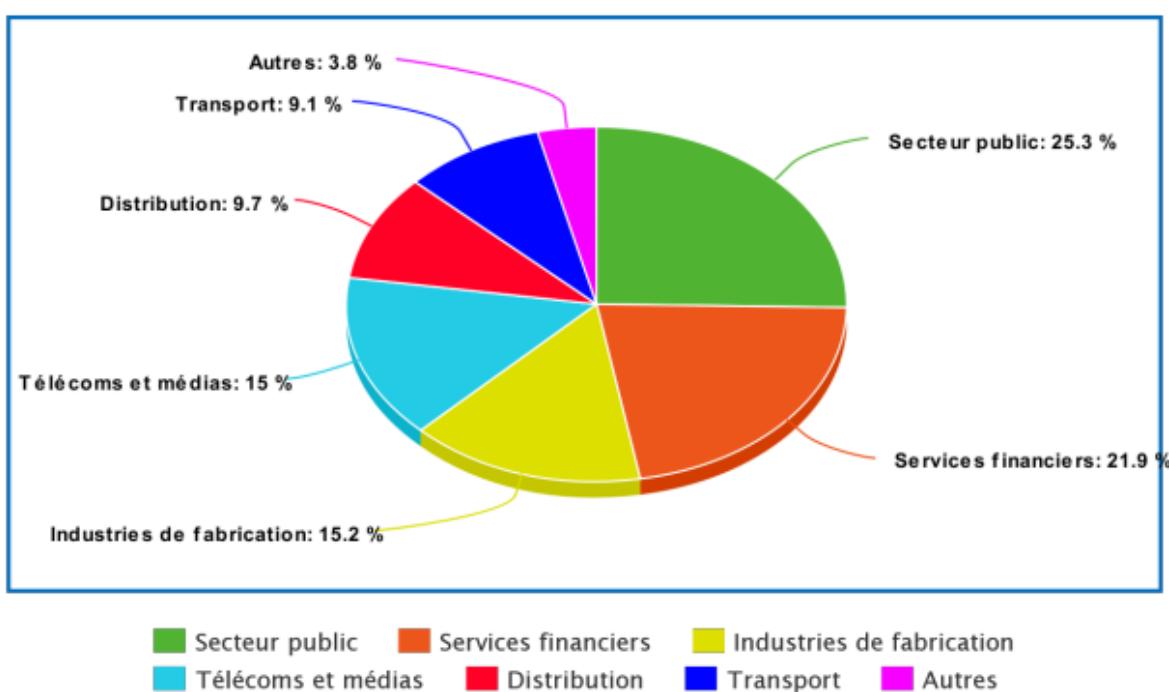
CAPGEMINI figure parmi les premiers groupes mondiaux de services informatiques. Le CA par activité se répartit comme suit :

- Prestations d'infogérance (54,4 %) : gestion d'infrastructures Informatiques, traitement de paiements par carte ;
- Prestations de conseil (6,2 %) ;
- Intégration de systèmes (39,4 %).



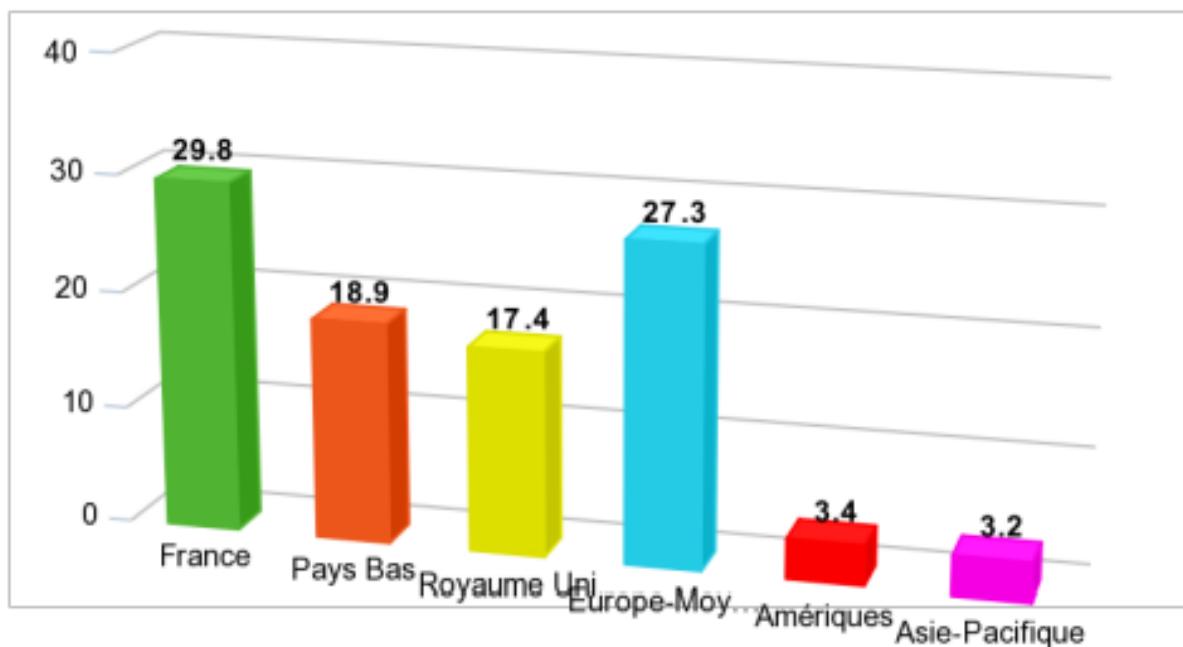
Le CA par marché se ventile entre :

- Secteur public (25,3 %) ;
- Services financiers (21,9 %) ;
- Industries de fabrication (15,2 %) ;
- Télécoms et médias (15 %) ;
- Distribution (9,7 %) ;
- Transport (9,1 %) ;
- Autres (3,8 %).



La répartition géographique du CA est la suivante :

- France (29,8 %) ;
- Pays Bas (18,9 %) ;
- Royaume Uni (17,4 %) ;
- Europe-Moyen Orient-Afrique (27,3 %) ;
- Amériques (3,4 %) ;
- Asie-Pacifique (3,2 %).



### g. Capgemini Maroc :

Capgemini s'installe en 2007 au Maroc, sous le nom Capgemini TS. Capgemini Maroc participe aujourd'hui à des projets importants de développement et de maintenance de systèmes d'information pour le compte d'une dizaine d'acteurs francophones majeurs notamment dans le secteur public, l'énergie, les services financiers et les télécommunications.

En s'implantant à Casa Nearshore en 2010, Capgemini répond à la stratégie d'industrialisation des grands groupes français qui recherchent des partenaires capables de les accompagner sur l'ensemble de leurs projets informatiques. Parallèlement à la croissance exponentielle de son activité, CTSM est devenue un des premiers employeurs au Maroc dans le domaine des technologies de l'information.

<b>Nom</b>	Capgemini Technology services
<b>Forme Juridique</b>	Société anonyme
<b>Année de création</b>	2007

<b>Adresse</b>	Casablanca : Casanearshore - shore 8 et 16 Rabat : Technopolis bâtiment B7
<b>Effectif</b>	Plus de 2000
<b>PDG</b>	Moncef BENABDESLAM

## **2. Environnement de travail :**

### **a. CloudBox :**

Il s'agit d'une entité sous forme d'une offre au sein de Capgemini TS Maroc, ses principaux services c'est la gestion des infrastructures informatiques et la migration vers le Cloud. CloudBox accompagne ses clients dans leur transformation digitale ayant pour ligne de mettre en place une politique cloud-first voire cloud-only, en s'appuyant sur une offre entreprise-grade et des méthodologies éprouvées par les plus grands experts du secteur.

CloudBox propose :

- Un accompagnement end-to-end (sécurité, conseil, conception, build, managed services) pour une mise en œuvre de cloud public, privé ou hybride.
- Une organisation multidisciplinaire, flexible (multi-lingues, 24/7) travaillant avec des méthodologies innovantes tel qu'Agile, DevSecOps et autre.

Le processus métier au sein de CloudBox se présente comme suit :



- ⇒ **Étape 1:** Conception et architecture, cette étape consiste à organiser un Workshop d'accompagnement et d'explication aux clients dans le but de comprendre les besoins métier, une identification des données et les applications permettant une projection vers le cloud, et finalement la définition et la planification de la stratégie de migration.
- ⇒ **Étape 2 :** Build et Test, il s'agit d'une approche de bout en bout qui assure la migration des applications vers le cloud, la validation de l'architecture cible par les partenaires cloud respectant les normes de sécurité et l'implémentation ainsi que le déploiement de l'infrastructure cible.

⇒ Étape 3 : Run, l'offre présente des services multi-cloud, une disponibilité 24/7/365, français/anglais et un processus d'amélioration continue.

L'écosystème de CloudBox est le suivant :



L'organisation et la hiérarchie au sein de CloudBox :



**b. Certifications :**

## Microsoft Certified Azure Fundamentals

ACHRAF AZOUAGH

Has successfully completed the requirements to be recognized as a Microsoft Certified: Azure Fundamentals.

Date of achievement: February 03, 2022



Satya Nadella  
Chief Executive Officer



## Microsoft Certified Microsoft Azure Administrator Associate

ACHRAF AZOUAGH

Has successfully completed the requirements to be recognized as a Microsoft Certified: Azure Administrator Associate.

Date of achievement: May 08, 2022

Valid until: May 08, 2023



Satya Nadella  
Chief Executive Officer



- **Azure fundamentals (AZ-900)** : elle porte sur les bases du Cloud Computing sous Microsoft Azure, en parcourant presque tous les services proposés par ce fournisseur Cloud, et en apprenant à manipuler l'environnement (Le portail Azure, stockage, machines virtuelles, AZ Cli, etc.).
- **Azure Administrator Associate (AZ 104)** : elle permet d'avoir de l'expertise en gestion et surveillance de l'environnement Microsoft Azure d'une organisation. Le candidat à cet examen doit avoir environ six mois d'expérience pratique dans l'administration d'Azure, ainsi qu'une solide compréhension des services Azure de base, des charges de travail Azure, de la sécurité et de la gouvernance. De plus, ce potentiel examiné doit avoir une expérience de l'utilisation de PowerShell, Azure CLI, du portail Azure et des modèles Azure Resource Manager.

---

*Chapitre II : Étude théorique autour du  
Cloud & DevOps*

---

Dans cet axe nous allons aborder l'état de l'art des différentes technologies et approches constitutives de mon PFE à savoir le Cloud Computing, DevOps, Testing, la conteneurisation, CI/CD et le monitoring.

## 1. Cloud Computing :

### a. Définition :

Le Cloud Computing est la fourniture de services informatiques (notamment des serveurs, du stockage, des bases de données, la gestion réseau, des logiciels, des outils d'analyse, l'intelligence artificielle) via Internet (le cloud) dans le but d'offrir une innovation plus rapide, des ressources flexibles et des économies d'échelle. En règle générale, vous payez uniquement les services cloud que vous utilisez (réduisant ainsi vos coûts d'exploitation), vous gérez votre infrastructure plus efficacement et vous adaptez l'échelle des services en fonction des besoins de votre entreprise.

La majorité des organisations choisissent les services de Cloud Computing pour réduire les investissements dans les coûts d'infrastructure, les coûts de maintenance et pour assurer la disponibilité des ressources 24 heures sur 24 ; le Cloud Computing est une solution plus efficace et plus rentable qu'un centre de données traditionnel.

### b. Histoire :

L'idée du Cloud Computing a pris naissance en 1990 et surtout en 1991 avec la naissance d'Internet et la mise sur le marché du logiciel CERN qui a été le premier logiciel accessible par le Web. Ensuite, l'idée a progressé avec l'apparition de nouvelles solutions IT, le lancement du navigateur Mosaic en 1993 et celui du navigateur Netscape en 1994. En 1995, la découverte d'EBay et d'Amazon a encore accéléré le processus. Enfin, le lancement en 1996 de l'assistant Palm PDA a donné une nouvelle impulsion.

La première utilisation de l'expression « Cloud Computing » remonte à 1997, lorsque Ramnath Chellappa, professeur en systèmes d'information et en management, l'a utilisée pour décrire un nouveau modèle de gestion de l'informatique, dans lequel les limites ne seraient plus définies par des problématiques techniques mais par des choix économiques. L'intérêt de ce modèle réside notamment dans la transition d'une partie des coûts informatiques.

En 2002, Amazon lance le premier service Cloud à la suite d'une surcharge de flux important des clients sur leur site, le reste de l'année ces serveurs ne servaient à rien, ils ont donc décidé de les louer.

Ce service permet aux entreprises d'adapter leurs infrastructures en fonction de leurs besoins et elles n'ont pas à investir de grosse somme car c'est une sorte de location. Le terme Cloud Computing sera rendu populaire en 2006 par le directeur exécutif de Google, Éric Schmidt, qui qualifiait ainsi de « nuage Internet » l'externalisation de données et d'applications qui auparavant étaient situées sur les serveurs et ordinateurs des sociétés, des organisations ou des particuliers.

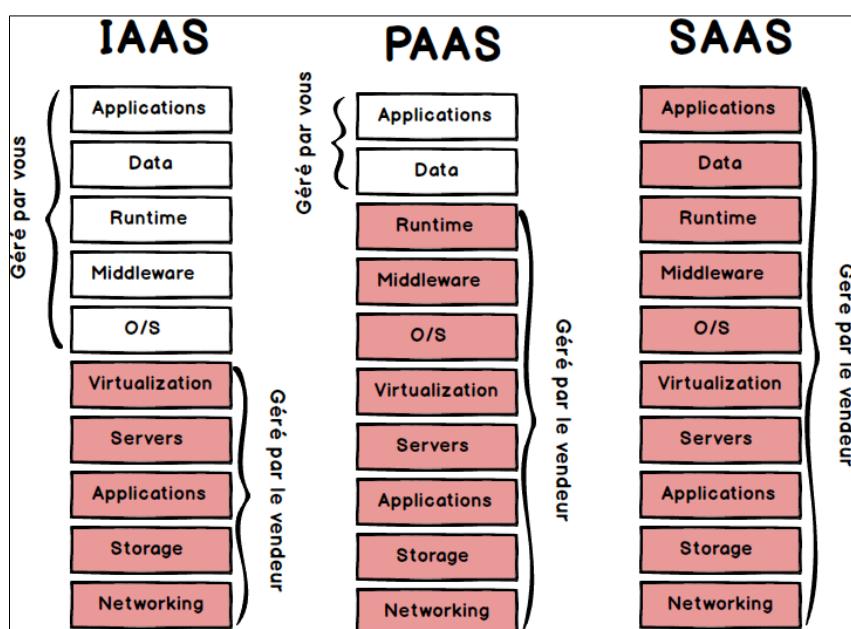
L'époque des grands investissements en capital dans les infrastructures logicielles et informatiques appartient désormais au passé pour toute entreprise qui choisit d'adopter le modèle de Cloud Computing pour l'achat de services informatiques.

### c. Types des services Cloud :

En fonction des services fournis, les services de cloud computing sont divisés en trois grandes catégories : **Infrastructure as a Service (IaaS)**, **Platform as a Service (PaaS)** et **Software as a Service (SaaS)**.

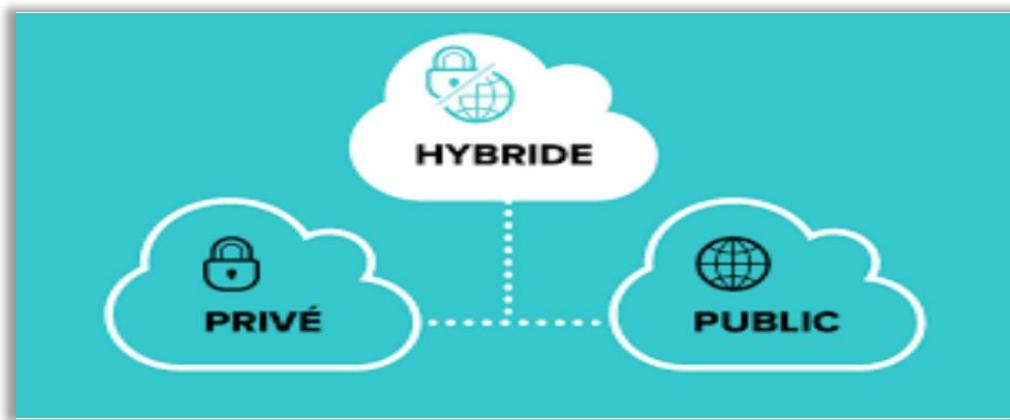
- **Infrastructure as a Service** : L'IaaS offre au client une infrastructure externe. Le fournisseur prend en charge l'installation des serveurs de fichiers, les réseaux et le stockage des données. De cette façon, le client n'a pas besoin d'acheter les équipements liés à ces ressources : il les loue au prestataire. En revanche, le client est responsable de ses applications, de ses données et du système d'exploitation. Les principales entreprises qui fournissent IaaS sont AWS, Rackspace Open Cloud, IBM Smart Cloud, Microsoft Azure, etc ;

- **Platform as a Service** : L'expression plateforme en tant que service (PaaS, Platform-as-a-Service) qualifie les services de cloud computing qui offrent un environnement à la demande pour développer, tester, fournir et gérer des applications logicielles. PaaS est conçu pour permettre aux développeurs de créer rapidement des applications web ou mobiles sans avoir à se préoccuper de la configuration ou de la gestion de l'infrastructure de serveurs, de stockage, de réseau et de bases de données nécessaires au développement. Les fournisseurs courants de PaaS incluent AWS, Salesforces.com, Microsoft Azure, Oracle Cloud, SAP et OpenShift ;
- **Software as a Service** : Le logiciel en tant que service (SaaS, Software-as-a-Service) est une méthode de diffusion d'applications logicielles via Internet, à la demande et en général sur abonnement. Avec le SaaS, les fournisseurs de services cloud hébergent et gèrent les applications logicielles et l'infrastructure sous-jacente, et gèrent la maintenance, par exemple la mise à niveau des logiciels et l'application des correctifs de sécurité. Les utilisateurs se connectent à l'application via Internet, en général par l'intermédiaire d'un navigateur web sur leurs téléphones, leurs tablettes ou leurs PCs. Les principaux fournisseurs de SaaS sont Microsoft 365, Zoho, Salesforce, SAP, Google G Suite, etc ;



#### d. Types de déploiements Cloud :

Le choix d'un type de cloud ou de service cloud est décisif. Tous les Clouds sont différents et chaque service cloud permet de résoudre un problème différent. Le type de déploiement présente un aperçu sur l'hébergement, l'accès, le niveau de sécurité et plusieurs autres options.



- **Cloud Public** : Un cloud public est détenu et exploité par un fournisseur de services cloud tiers, qui propose des ressources de calcul, telles que des serveurs et du stockage, via Internet. Microsoft Azure, AWS et Google Cloud sont tous des exemples de cloud public. Dans un cloud public, tout le matériel, tous les logiciels et toute l'infrastructure sont la propriété du fournisseur du cloud.



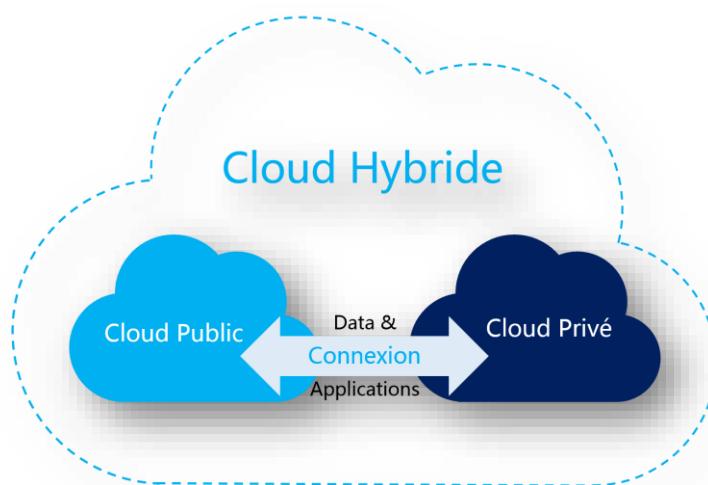
- **Cloud Privé** : On parle de ce type de Cloud lorsque l'infrastructure informatique sous-jacente est spécifique à un client unique, avec un accès entièrement isolé.

De nos jours, les entreprises créent des Clouds privés dans des datacenters hors site et loués à des fournisseurs. Cette tendance a classifié des sous-types de Clouds privés, notamment :

Cloud Privé Géré	Cloud Privé Dédié
Créé et utilisé par les clients, déployé, configuré et géré par un fournisseur tiers.	C'est un accès à un Cloud au sein du fournisseur Cloud.

Le cloud privé permet aux entreprises de profiter de certains des avantages d'évolutivité et d'agilité du cloud computing sans oublier qu'il leurs permettent de surmonter les problèmes de sécurité et de conformité dus aux Clouds publics. Cependant, les cloud privés sont généralement plus chers et plus difficiles à maintenir que leurs homologues publics.

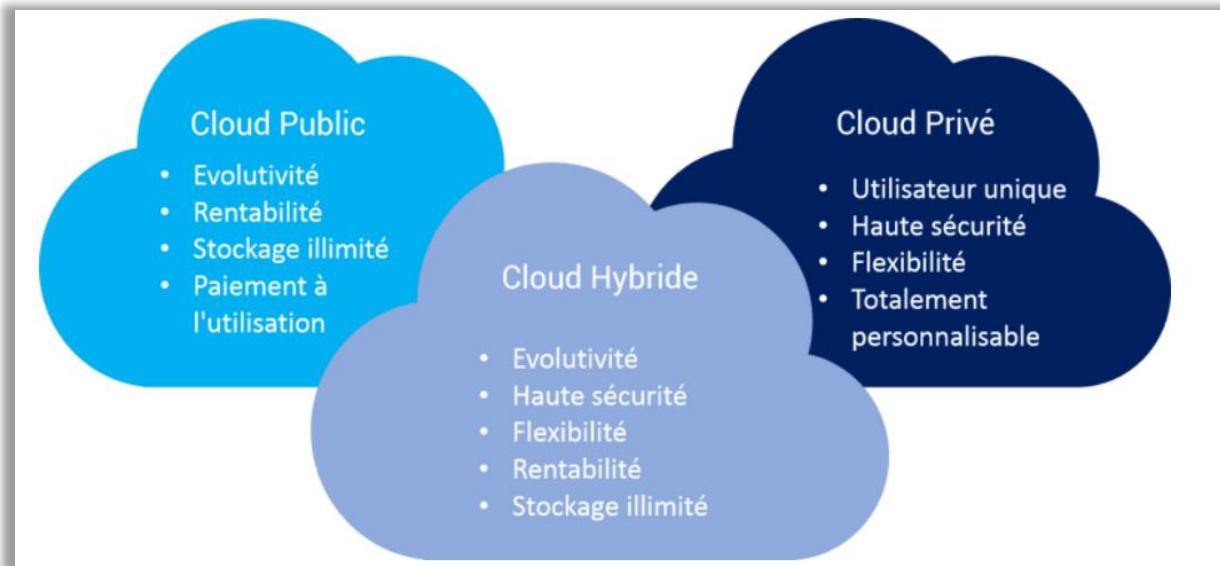
- **Cloud Hybride :** Il s'agit de combiner le cloud public et le cloud privé, qui sont gérés comme un environnement unique. Les applications peuvent donc être déplacées entre différents environnements séparés qui restent connectés. Les organisations ont parfois tendance à disposer de plusieurs Clouds gérés indépendamment (cloud public, cloud privé et/ou cloud hybride), communément connu sous le nom d'un environnement multiclouds. Il peut devenir un cloud hybride lorsqu'une forme d'intégration ou d'orchestration permet de connecter ses plusieurs Clouds.



## e. Avantages du Cloud Computing :

- **Coût (CapEx)** : le Cloud Computing élimine la nécessité d'investir dans du matériel et des logiciels, de configurer et de gérer des centres de données sur site : racks de serveurs, alimentation électrique permanente pour l'alimentation et le refroidissement, experts informatiques pour la gestion de l'infrastructure. La facture est vite salée ;
- **Vitesse** : la plupart des services de Cloud Computing sont fournis en libre-service et à la demande. D'énormes ressources de calcul peuvent donc être mises en œuvre en quelques minutes et en quelques clics, offrant ainsi aux entreprises un haut niveau de flexibilité et les dégageant de la pression liée à la planification de la capacité ;
- **Mise à l'échelle mondiale** : il est possible de mettre en œuvre la quantité nécessaire de ressources informatiques, par exemple plus ou moins de puissance de calcul, de stockage ou de bande passante, au moment où elles sont nécessaires, là où elles sont nécessaires ;
- **Productivité** : le Cloud Computing élimine la nécessité d'investir dans du matériel et des logiciels, de configurer et de gérer des centres de données sur site : racks de serveurs, alimentation électrique permanente pour l'alimentation et le refroidissement, experts informatiques pour la gestion de l'infrastructure. La facture est vite salée ;
- **Performance** : les plus grands services de Cloud Computing s'exécutent sur un réseau de centres de données sécurisés, dont le matériel est régulièrement mis à niveau pour assurer des performances rapides et efficaces. Ceci offre plusieurs avantages par rapport à un centre de données classique, y compris un temps de latence réseau réduit pour les applications et de plus grandes économies d'échelle ;
- **Fiabilité** : le Cloud Computing simplifie la sauvegarde des données, la récupération d'urgence et la continuité des activités. Il rend ces activités moins coûteuses, car les données peuvent être mises en miroir sur plusieurs sites redondants au sein du réseau du fournisseur ;
- **Sécurité** : de nombreux fournisseurs de cloud offrent un vaste éventail de stratégies, technologies et contrôles qui renforcent globalement votre situation de sécurité, contribuant ainsi à protéger vos données, vos applications et votre infrastructure contre des menaces potentielles ;

De plus les avantages cités ci-dessus, il existe d'autres qui sont relatifs à chaque type de Cloud comme montré dans la figure suivante :



## f. Inconvénients du Cloud Computing :

- **Coût (OpEx)** : le Cloud Computing élimine la nécessité d'investir dans du matériel et des logiciels, de configurer et de gérer des centres de données sur site : racks de serveurs, alimentation électrique permanente pour l'alimentation et le refroidissement, experts informatiques pour la gestion de l'infrastructure. La facture est vite salée ;
- **Le cadre légal et territorial** : les données transférées dans le cloud ne sont pas forcément présentes sur le territoire national. Par conséquent, sauf mention contraire du prestataire de service, on ne sait pas précisément à quel endroit sont stockées les données. De plus, on ne possède aucun droit d'accès physique à ces données ;
- **La sécurité** : les réseaux informatiques sont potentiellement attaquables et les services virtuels peuvent être mis hors fonction ;
- **La disponibilité** : les services Cloud offerts dépendent de la connexion internet. Une panne peut entraîner une perturbation ;

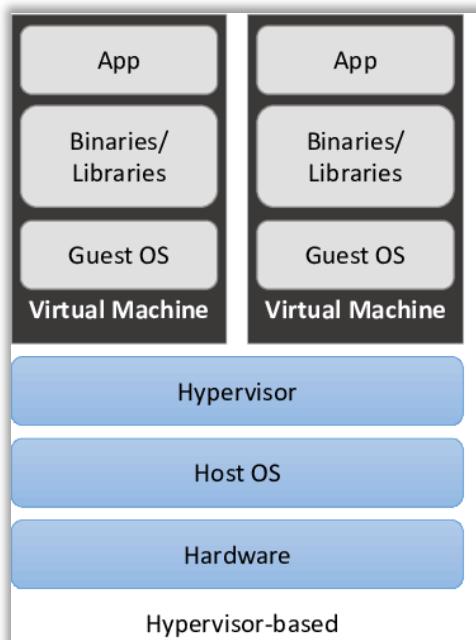
## g. Les fournisseurs Cloud :



- **Microsoft Azure** : Azure est la plateforme Cloud de Microsoft lancée en février 2010, Elle regroupe divers services de Cloud Computing. Microsoft Azure permet de profiter de ressources de Cloud Computing à la demande. Il permet aux entreprises de faire d'importantes économies en évitant les dépenses de systèmes et administration à savoir la mise en place des centres de données, la maintenance, la mise à jour, l'électricité et bien d'autres charges. Comme chez les concurrents, on retrouve notamment un service de stockage, des machines virtuelles, et des réseaux de diffusion de contenu. Azure propose également des services exploitant les technologies propriétaires de Microsoft.  
Microsoft Azure se considère parmi les géants du Cloud Computing et se positionne par son large réseau établi par la société mère Microsoft depuis des nombreuses années, sa technologie évolutive ainsi que sa simplicité dans la gestion des services;
- **Google Cloud Platform** : le Cloud Computing élimine la nécessité d'investir dans du matériel et des logiciels, de configurer et de gérer des centres de données sur site : racks de serveurs, alimentation électrique permanente pour l'alimentation et le refroidissement, experts informatiques pour la gestion de l'infrastructure. La facture est vite salée ;
- **Amazon Web Services** : division du groupe américain de commerce électronique Amazon créé en 2006, AWS est un fournisseur Cloud offrant une centaine de services. Nombreux sont les organisations de petite et grande échelle qui ont choisi AWS par son large réseau et sa technologie évolutive leur permettant de bénéficier des produits internationaux basés sur le cloud : calcul, stockage, bases de données, analyse, mise en réseau, services mobiles, outils de développement, outils de gestion, Internet des Objets, sécurité et applications d'entreprise. Ces services aident les organisations à travailler plus vite, à réduire leurs coûts informatiques, assurer la haute disponibilité ainsi que la scalabilité ;

## h. Cloud et Virtualisation :

La virtualisation consiste à dématérialiser le comportement et les données d'un serveur ou d'une machine, de façon à faire tourner plusieurs de ces instances dématérialisées sur un même serveur physique.



La virtualisation et le Cloud Computing sont deux concepts différents, mais ils sont également complémentaires.

- **Différence :** la virtualisation est une technologie permettant de créer plusieurs environnements simulés ou ressources spécialisées à partir d'un seul système physique. Le Cloud Computing permet d'orchestrer facilement la gestion de ces instances virtuelles et de transformer la délivrance de ces ressources en un service, facturable à la consommation. Il s'agit d'un environnement qui dissocie, regroupe et partage des ressources évolutives sur un réseau. Pour conclure, la virtualisation est une technologie alors que le cloud est un environnement.
- **Complémentarité :** La virtualisation est une partie intégrante du fonctionnement des Clouds. En effet, le cloud regroupe et automatise des ressources virtuelles pour une utilisation à la demande, en allouant les ressources virtuelles dans des pools centralisés et en ajoutant une couche logicielle de gestion, on passe de la virtualisation vers le cloud. Les administrateurs peuvent donc contrôler l'infrastructure, les plateformes, les applications et les données qui seront utilisées dans l'environnement cloud.

## 2. DevOps :

### a. Contexte et Histoire :

Des années avant l'apparition de cette approche, le contexte informatique souffrait tous les jours des problèmes de communication entre les administrateurs système (Ops) et les développeurs (Dev) qui les empêchaient de mener les missions et accomplir les projets. En effet, le travail auparavant reposait sur une méthodologie plus rigide : L'équipe de développement logicielle ou applicative se charge de collecter les besoins métiers et de les développer. Elle teste ensuite le logiciel ou l'application lorsqu'elle est finalisée, si le logiciel ou l'application répond aux besoins métiers, le code source est mis à disposition à l'équipe opérationnelle pour la partie exploitation. La discontinuité de la communication entraîne une ignorance des obstacles que chaque équipe pourrait rencontrer. Cela crée un mur de confusion entre les développeurs qui tendent vers les changements et les tests répétitifs et les administrateurs qui ont besoin de garantir la stabilité.



Depuis l'année 2007 et jusqu'à son lancement vers juin 2009, le DevOps était un projet suite aux détections des besoins et problèmes faites par plusieurs ingénieurs, notamment **Patrick Debois**, un administrateur système consultant ayant pour mission la migration des données pour le gouvernement Belge.

## b. Présentation :

Le DevOps est une approche en informatique qui vise à réunir les développeurs et les opérationnels à travers plusieurs étapes, pratiques et outils de manière de plus en plus fluide et agile.

Le DevOps propose des solutions plus rapidement dans le but de satisfaire les clients d'autant plus tout en favorisant la communication et la collaboration entre les équipes chargées du développement et des opérations IT.

Cette approche de développement logiciel implique un développement continu, des tests continus, une intégration continue, un déploiement continu et une surveillance continue tout au long du cycle de vie du logiciel afin de développer des logiciels de haute qualité, de raccourcir le cycle de développement, et d'augmenter par conséquent la satisfaction et l'expérience client.

## c. Fonctionnement :

L'approche DevOps se révèle comme un déploiement continu avec : Un développement et des tests constants. Une intégration et une mise en œuvre constante : déploiement avec des processus fiables. Une surveillance constante et une validation de la qualité opérationnelle. Les étapes de l'approche DevOps sont étalées sur les phases principales suivantes :

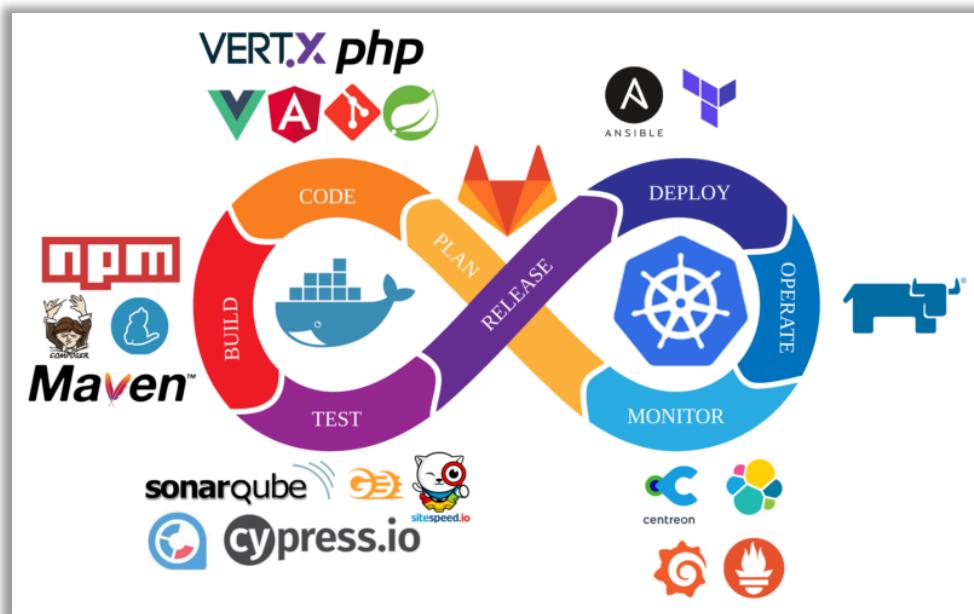
Au cours de la phase de planification, les équipes DevOps imaginent, définissent et décrivent les fonctionnalités des applications et des systèmes qu'elles créent.

Ensuite, La phase de développement comprend tous les aspects du codage (écriture, test, révision et intégration du code par les membres de l'équipe), de la génération de ces codes dans des artefacts de build qui pouvant être déployés dans divers environnements.

Lors de la phase de livraison, les équipes définissent un processus de gestion de mise en production ponctué d'étapes d'approbation manuelle claires. Elles déterminent

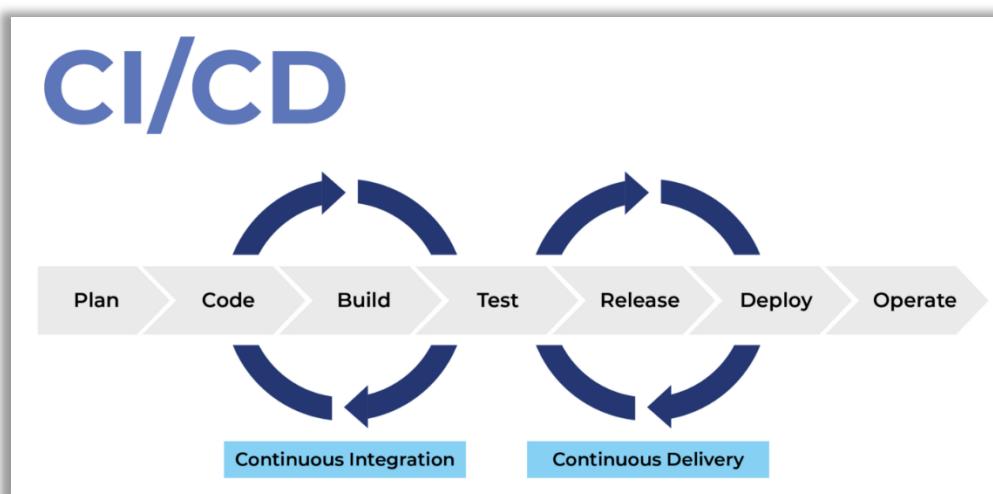
également des portes automatisées que franchissent les applications entre les étapes jusqu'à leur mise à la disposition des clients.

La phase d'exploitation implique la maintenance, la supervision et le dépannage des applications dans les environnements de production.



#### d. CI/CD pipeline :

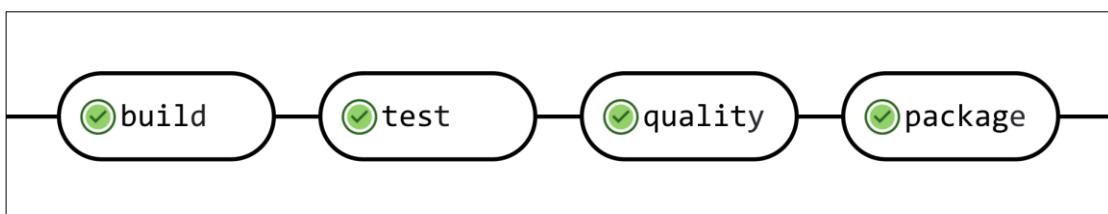
Un pipeline CI/CD est une série d'étapes à réaliser en vue de distribuer une nouvelle version d'un logiciel.



Un pipeline CI/CD utilise la surveillance et l'automatisation pour améliorer le processus de développement des applications, en particulier lors des phases d'intégration et de tests ainsi que pendant la distribution et le déploiement.

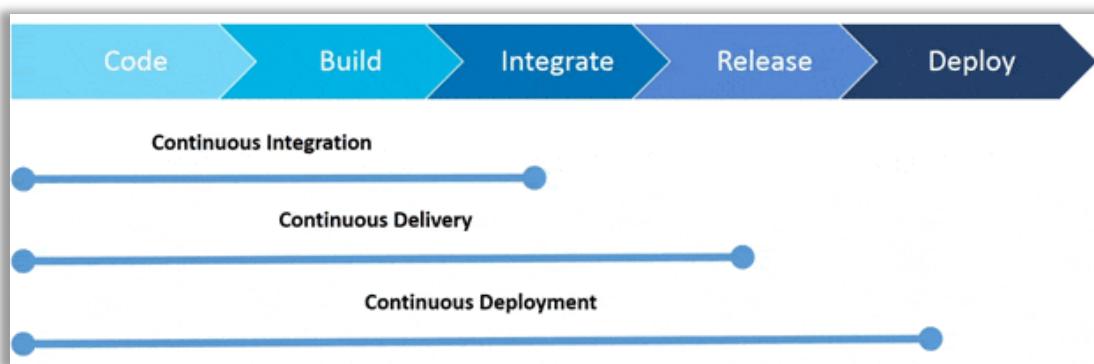
Les phases de ce pipeline avec leurs outils sont présentées comme suit :

- **Intégration continue** : l'équipe DEV applique régulièrement ses modifications sur un référentiel partagé. Ensuite, et avant d'envoyer leur code, les développeurs peuvent choisir d'exécuter des tests sur des unités locales pour le vérifier davantage avant son intégration. Un service d'intégration continue crée et exécute automatiquement des tests unitaires sur les nouveaux changements de codes pour détecter immédiatement n'importe quelle erreur, la corriger le plus rapidement possible et réduire le temps nécessaire pour la vérification et la validation de la mise à jour du logiciel ;

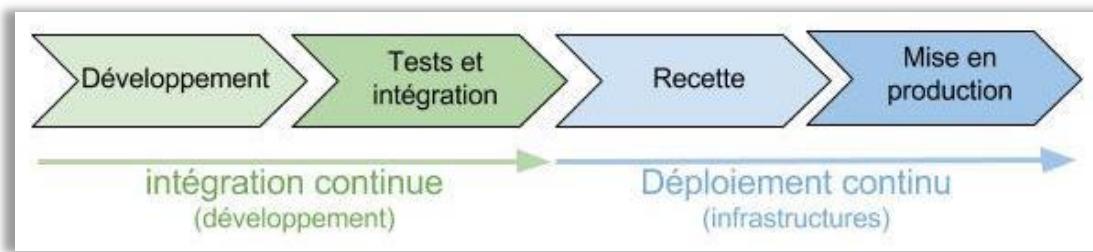


⇒ Outils : Git, Github, Jenkins, Selenium, Maven...

- **Livraison continue** : la livraison continue s'arrête avant la production, c'est une étape importante pour s'assurer que le code soit prêt pour la production. À chaque fois qu'un nouvel artefact de build est disponible, il est automatiquement placé dans l'environnement souhaité et déployé ;



- **Déploiement continu :** Le déploiement continu (CD) est la suite du processus de l'intégration continue (CI). Après avoir valider les tests d'intégration par l'équipe DEV, c'est le moment pour automatiser les actions de déploiements composées des tests faits sur l'environnement de qualification pour s'assurer du bon fonctionnement de la nouvelle fonctionnalité du logiciel en production.



⇒ Outils : Ansible, Puppet, Chef, Terraform, Docker ...

### e. Avantages :

- **Détruire le mur de confusion :** c'est le principal avantage de cette approche, la collaboration entre les équipes leurs permettent de faire le compromis entre la stabilité et le changement et entraînent plusieurs autres objectifs ;
- **La rapidité :** la relation dynamique entre les développeurs et les opérateurs ainsi que l'automatisation des processus permettent d'accélérer les déploiements et la livraison des logiciels ;
- **La fiabilité :** la résilience et la tolérance aux pannes sont toutes des conséquences directes de la manière continue avec laquelle toutes les étapes de l'approche DevOps se font ;
- **La gestion des risques :** le DevOps permet d'identifier les facteurs de risque au début du cycle de vie de l'application, détecter et réparer d'une manière rapide et continue les problèmes ou les erreurs le plus tôt possible ;
- **Apprentissage continu :** les équipes DevOps hautement performantes adoptent un état d'esprit de croissance. Elles améliorent leurs processus continuellement et accélèrent leur capacité à innover et à s'adapter au marché ;
- **La concurrence :** la création rapide et rentable des logiciels permet de créer la concurrence ce qui mène à des produits de qualité avec un prix convenable et une satisfaction marquée de la part du client.

## f. DevOps & Cloud :

Le rassemblement de développeurs et des opérationnels conduit à une véritable appétence, cette cohésion mène à un rendement de plus en plus croissant.

Le Cloud est un service qui évolue facilement et rapidement par opposition à la solution On-Premise beaucoup plus stagnée. Le fournisseur s'en occupe pour se concentrer d'autant plus sur le développement du logiciel que son maintien.

Les administrateurs (Ops) ont accès, grâce au Cloud, à un service qui leur permet une grande flexibilité de volume, de stockage et qui propose tous les outils DevOps dont ils ont besoin pour être autonomes et productifs. Ceci permet à ces équipes d'avoir la main sur la mise en production et l'analyse du besoin des développeurs, sans avoir à penser aux problèmes de coupure de serveurs physiques, à la maintenance ou aux capacités.

Les services que proposent les fournisseurs Cloud (IaaS, PaaS ou SaaS) tel le codage d'infrastructure, IoT et Big Data sont accessibles et simplifiés dans des interfaces web, mais aussi en ligne de commande et par API, et sont en outre parfaitement adaptés pour le DevOps.

Les fournisseurs Cloud proposent également des outils DevOps intégrés à leurs services pour faciliter l'adoption par les équipes DevOps.

## 3. Testing :

### a. Type de tests :

Parmi les types de test les plus connus nous présentons les niveaux suivants :

- **Les tests unitaires** : Ils permettent de vérifier le bon fonctionnement d'une partie déterminée d'un logiciel ou d'une portion d'un programme. Il s'agit pour le développeur de tester un module sans prendre en compte le reste du programme afin de s'assurer qu'il répond aux spécifications fonctionnelles et qu'il fonctionne correctement en toutes circonstances.

- **Les tests d'intégrations :** Ils viennent après la validation des tests unitaires. Ils ont pour but de vérifier la cohésion des parties développées séparément et de valider la communication entre les modules. Dans cette étape, il faut prévoir tous les cas de communication inattendus pouvant se produire entre ces modules.
- **Les tests de validation :** Ils ont pour objectif de s'assurer que le logiciel respecte effectivement toutes les exigences formalisées par le client qui décrivent ce que le client est en droit d'en attendre.
- **Les tests système :** Les tests système de logiciel ou de matériel réfèrent à un processus de test d'un système intégré afin d'évaluer sa conformité aux exigences spécifiées. Ils permettent de vérifier si tous les composants fonctionnent ensemble dans leur environnement cible.
- **Les tests de non-régression :** Les tests de non-régression permettent de vérifier que des modifications n'ont pas altérées le fonctionnement de l'application. L'utilisation d'outils de tests, dans ce domaine, permet de faciliter la mise en place de ce type de tests. Ils doivent être lancés à chaque livraison. Pour améliorer le ROI des tests et tenir les cadences de livraison ces tests sont généralement automatisés pour accélérer les phases de tests et concentrer les tests manuels sur le test des évolutions.

## b. Tests manuels vs Tests automatisés :

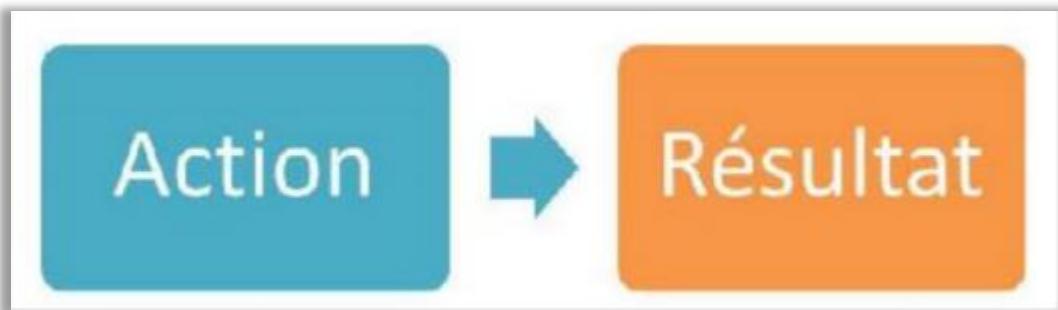
Les problèmes rencontrés le plus souvent dans une entreprise quand il s'agit de tests sont lors de l'automatisation et le moment de tester manuellement. Tous les tests ne peuvent pas être automatisés et la plupart du temps est difficile de décider quoi automatiser et de tester manuellement. Voici quelques avantages et inconvénients des deux méthodes :

### **Tests manuels :**

Les tests manuels sont exécutés manuellement par un testeur d'après un plan de test donné : le testeur entre les données de test via une interface ; lance les tests ; observe les résultats et les compare avec les résultats attendus. En cas d'erreurs, le testeur réalise une analyse qui fera l'objet d'une fiche d'anomalie si nécessaire

Avantages	Inconvénients
-Pouvoir commencer avant que l'application soit parfaitement stable. -Tester en parallèle des opérations de corrections.	-L'impossibilité de respecter les délais -Une activité laborieuse, longue, coûteuse et peu reproductible.
-Créer des scripts de test simples rapidement.	-La répétition des mêmes tests devient très pénible pour le testeur
-Implication souvent plus facile d'utilisateurs totalement néophyte dans la démarche de test.	-Le temps consacré aux tests de nonrégression est significatif
	-Lorsque des applications doivent fonctionner sur plusieurs platesformes, la charge des tests manuels croît proportionnellement en multipliant les risques d'erreurs humaines et d'incohérences. « Livre blanc, Borland »

Un test manuel bien écrit est une suite d'actions entraînant chacune un résultat à vérifier



Prenons un exemple : lire un mail :

Etape	Action	Résultat
Etape 1	Ouvrir l'application	L'application s'ouvre et affiche la boite de réception
Etape 2	Appuyer sur le premier mail de la boite de réception	Le contenu du mail s'affiche avec le nom de la personne l'ayant envoyé
Etape 3	Appuyer sur retour	La boite de réception est de nouveau affiché

Ici si un résultat n'est pas validé ou une action impossible alors le cas est en échec.  
Sinon, il est en succès.

### Test automatisé :

Contrairement aux tests manuels, le support d'outil qui décharge le testeur. Dans ce cas de test, le lancement des tests ; l'enregistrement des résultats ; se fait de façon automatique à partir d'un cas de test manuel bien écrit on peut développer un cas de test automatisé. Si on reprend l'exemple précédent alors il suffit de coder chaque partie du test. On commence par l'ouverture de l'application en appuyant sur l'icône, puis on vérifie que la boîte de réception est bien affichée (par exemple en cherchant la présence dans la partie haute de l'écran du texte « Boîte de réception »), ainsi de suite.

Avantages	Inconvénients
-L'automatisation des tests permet : - d'accélérer les cycles de validation et optimisent la qualité logicielle	-L'automatisation n'est pas adaptée pour tous les tests.
-Se libérer des tâches répétitives en cas des tests de régression	-L'entreprise doit donc prévoir un projet « pilote » afin de se rendre compte si les outils d'automatisation sont en adéquation avec ses attentes et ses logiciels.
-La fiabilité puisque ces tests peuvent vérifier toujours aussi consciencieusement les résultats, même après une énième exécution, ce qui est loin d'être le cas avec un humain.	-L'automatisation est coûteuse par rapport aux tests manuels et le retour sur investissement n'est pas immédiat
	-Les cas de tests automatisés devront alors être maintenus. Ceci engendre forcément une charge, un coût de test supplémentaire

## 4. Description de l'automatisation :

Un test automatisé est un test dont l'exécution ne nécessite pas l'intervention d'un humain. L'exécution de tests automatisés requiert donc l'utilisation de solutions informatiques dont le but est d'exécuter des actions, soit spécifiquement dans un navigateur web, soit plus généralement au niveau du système d'exploitation

### a. Objectif de l'automatisation :

Chaque groupe de développement logiciel teste ses produits, cependant le logiciel livré a toujours des défauts. Des ingénieurs de test s'efforcent de les trouver avant que le produit ne soit livré, mais avec tout modification la probabilité qu'ils réapparaissent est élevé, même avec les meilleurs processus de d'développement. L'utilisation de logiciel automatisée est la meilleure façon d'augmenter l'efficacité des tests, et la conformité du logiciel. L'automatisation des tests permet ainsi de garder la trace des anciens bogues et de les ré-exécuter automatiquement pour voir s'ils n'ont pas été cassés pour assurer les objectifs suivants :

- Economiser du temps et de l'argent
- Augmenter la qualité
- Augmenter la couverture de test
- Automatisation des tests fait ce que les tests manuels ne peuvent pas faire (garantir exécution de tous les tests avant tout livraison...)

### b. L'apport de l'automatisation :

Aujourd'hui les organisations de développement logiciel sont face à un grand challenge:

- Améliorer la qualité des applications complexes.
- Accélérer les délais et avec des budgets serrés.
- Fournir des solutions informatiques « gagnantes ».

Pour faire face à ce grand challenge, ces organisations de développement, ont choisi l'automatisation du processus de test pour optimiser la qualité des logiciels Y a-t-il un apport considérable/une forte valeur ajoutée par rapport au test manuels ? Quelle sont les meilleures pratiques ? Est-ce que l'automatisation répond-elle généralement ? L'automatisation des tests fonctionnels permet de multiples avantages :

**Constituer une bibliothèque de ressources :** l'expérience démontre que l'automatisation permet de réemployer des tests déjà existants sans en recréer à chaque nouveau projet. Et les tests réemployables sont davantage utilisés et permettent aux équipes de corriger un plus grand nombre d'erreurs et de constituer une bibliothèque de ressources.

**La cohérence :** Lors de la réutilisation des tests, les équipes d'assurance qualité accèdent à de nouveaux seuils de cohérence. L'automatisation permet la cohérence au processus de test, grâce à une procédure reproductive de documentation des résultats permettant de recréer et vérifier les erreurs pour accélérer leur résolution

**Des gains de productivité :** L'automatisation permet au département d'assurance qualité, de lancer des tests sans surveillance, et valider simultanément le bon fonctionnement d'une application sur plusieurs plates-formes, navigateurs et environnements. Cette opération libère des ressources pour se focaliser sur d'autres aspects de la qualité. Ces gains de productivité permettent de raccourcir les cycles de test et améliorer la qualité logicielle.

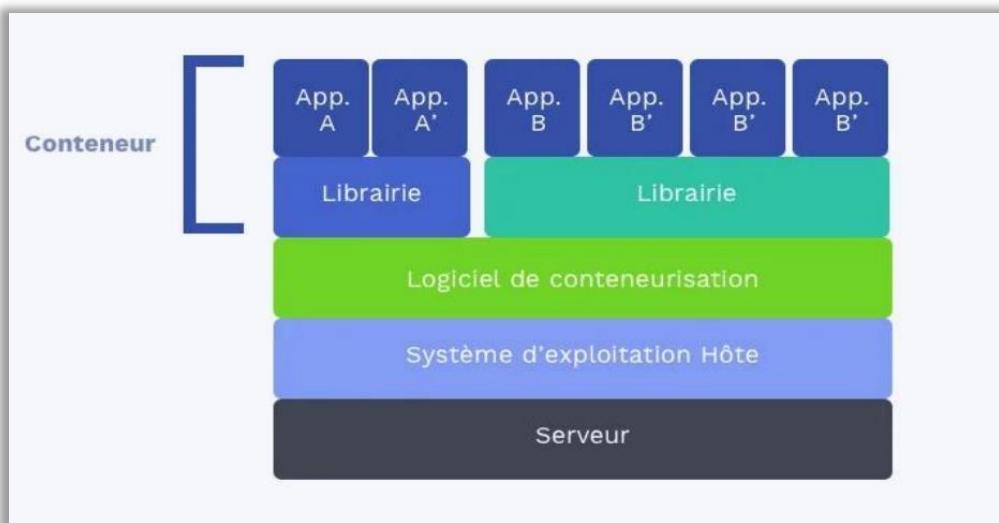
## 5. Conteneurisation :

### a. Contexte et définition :

Dans une approche traditionnelle, le code est développé dans un environnement informatique spécifique. Lorsqu'il est transféré vers un nouvel emplacement, cet environnement entraîne souvent des bogues et des erreurs.

La conteneurisation consiste à encapsuler le code logiciel et de le regrouper avec les fichiers de configuration, les bibliothèques et les dépendances associés nécessaires à son exécution afin qu'il puisse s'exécuter de manière uniforme et cohérente sur n'importe quelle infrastructure.

### b. Fonctionnement :



Chaque conteneur peut exécuter une application web ou un service dans son intégralité, les conteneurs isolent les applications les unes des autres sur un système d'exploitation partagé. Les applications en conteneur s'exécutent sur un hôte de conteneurs qui à son tour s'exécute sur le système d'exploitation (Linux ou Windows).

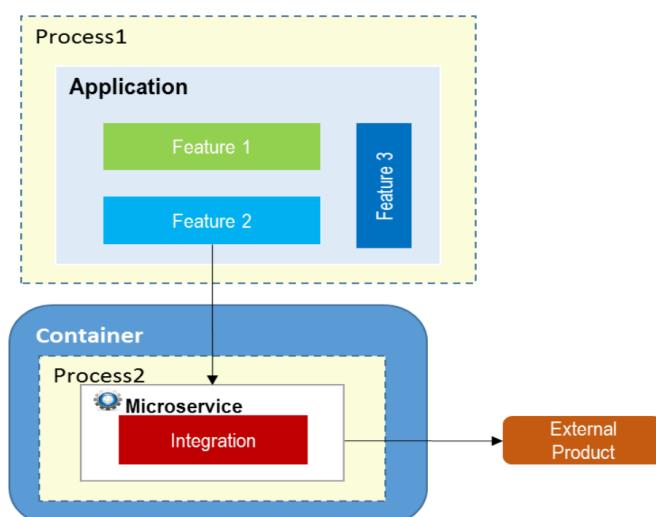
Les conteneurs logiciels agissent comme une unité de déploiement logiciel standard qui peut contenir du code et des dépendances différentes. Cette façon de mettre les logiciels en conteneur permet aux développeurs et aux informaticiens de les déployer dans les environnements en minimisant les modifications.

### c. Avantages :

La conteneurisation offre aux équipes de développeurs plusieurs avantages notamment :

- **La portabilité** : le résultat de la conteneurisation est un package exécutable de logiciels capable d'être portable et de fonctionner de manière uniforme et cohérente sur n'importe quelle plateforme ou cloud ;
- **L'agilité** : les conteneurs sont une étape presque primordiale dans une approche DevOps réglant les problèmes de communication et mène à la cohérence entre les équipes Dev et Ops ;
- **La rapidité** : plus de systèmes d'exploitation à démarrer ou des configurations indépendantes à gérer, les temps de démarrage et de manipulation sont donc accélérés ;
- **La tolérance aux pannes** : possibilité d'identification et de correction des erreurs sans affecter le fonctionnement continu des autres conteneurs ;
- **La sécurité** : l'architecture d'isolement qui caractérise la conteneurisation et les autorisations de sécurité pouvant être définies, empêchent par défaut l'invasion de code malveillant et des composants indésirables d'affecter d'autres conteneurs ou le système hébergeur.

### d. Conteneurs et microservices :

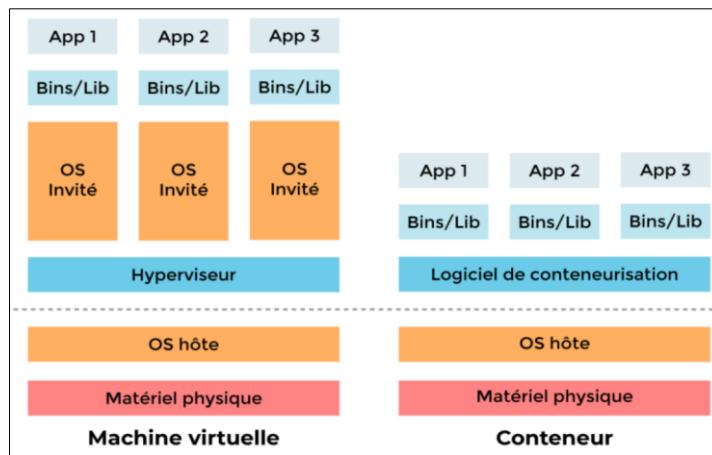


Les microservices sont un modèle logiciel qui combine une application logicielle avec l'interface utilisateur associée et la base de données sous-jacente en une seule unité

sur une plateforme de serveur unique ce qui révèle plus de granularité des composants d'une application contrairement au modèle monolithique. La communication entre les microservices se fait à travers des API et des interfaces REST (HTTP).

La conteneurisation et les microservices transforment les applications en ensembles de services ou de composants plus petits, portables, évolutifs, efficaces et plus faciles à gérer. Un microservice bénéficie en étant développé dans un conteneur des avantages de la conteneurisation : La portabilité, la compatibilité, l'agilité et l'automatisation.

### e. Conteneurisation vs Virtualisation :



Le point commun entre la virtualisation et la conteneurisation c'est que toutes les deux permettent à plusieurs types de logiciels (Linux ou Windows) d'être exécutés dans un seul environnement.

La virtualisation permet à chaque application ainsi que ses fichiers, bibliothèques et dépendances associés, y compris une copie du système d'exploitation (OS), sont regroupés sous forme de machine virtuelle.

Les conteneurs sont de plus en plus légers. Ils partagent le noyau du système d'exploitation de la machine, d'autres bacs et bibliothèques communs peuvent être partagés entre plusieurs conteneurs.

Contrairement aux machines virtuelles, les conteneurs ne sont pas regroupés dans une copie du système d'exploitation. Au lieu de cela, le moteur d'exécution du conteneur est installé sur le système d'exploitation du système hébergeur, devenant

ainsi le canal par lequel tous les conteneurs du système informatique partagent le même système d'exploitation.

Plusieurs conteneurs peuvent alors fonctionner sur la même capacité de calcul qu'une seule VM, ce qui améliore l'efficacité des serveurs, et permet de réduire les coûts des serveurs.

## **6. Monitoring et alerting :**

### **a. Définition et contexte :**

La supervision ou monitoring est le processus de collecte de métriques sur le fonctionnement du matériel et des logiciels d'un environnement IT, qui permet de s'assurer que les applications et les services sont pris en charge comme prévu.

En tenant compte de ce qui précède dans cet axe, le monitoring et le DevOps sont indissociable pour assurer la continuité car en effet, l'objectif de toute entreprise est de pouvoir anticiper toute perte de productivité, de gagner en agilité et de mettre en place rapidement des mesures correctives en cas d'anomalies ou de dysfonctionnement.

Les données collectées lors de la supervision peuvent servir à toutes les équipes informatiques responsables d'un projet.

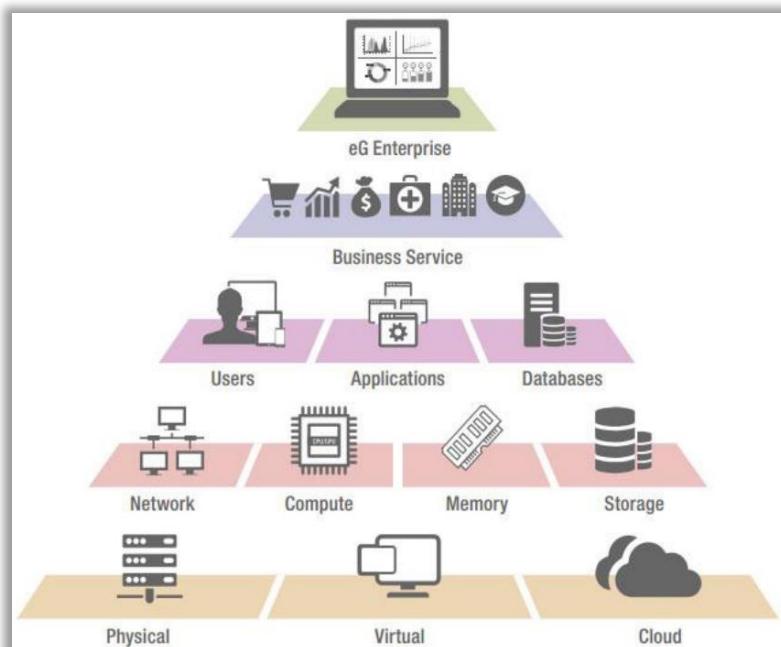
### **b. Processus et fonctionnement :**

Comme il s'agit d'un processus, la supervision se fait principalement en quatre étapes :

- **Planification** : dans cette étape, les superviseurs choisissent et mettent en place les outils convenables à leur collecte.
- **Visualisation** : les métriques sont disponibles sous formes de données, de graphes ou de notifications.
- **Analyse** : après la collecte et la visualisation, l'analyse et les remarques se font dans le but d'agir en cas de problèmes ou d'ambiguités.

- **Compte rendu** : après l'analyse et la résolution, les superviseurs créent des rapports contenant tout le processus.

Le monitoring permet de superviser l'ensemble du Système d'Information de l'entreprise sur des volets comme le réseau et matériel, les systèmes ou les applications.



Supervision réseau et matérielle :

- ⇒ **Commutateurs et routeurs** : disponibilité, interrogation des sondes, alertes ;
- ⇒ **Serveurs** : Disponibilité, charge, état ;
- ⇒ **Imprimantes** : Disponibilité, état de l'imprimante et des consommables.

Supervision des systèmes :

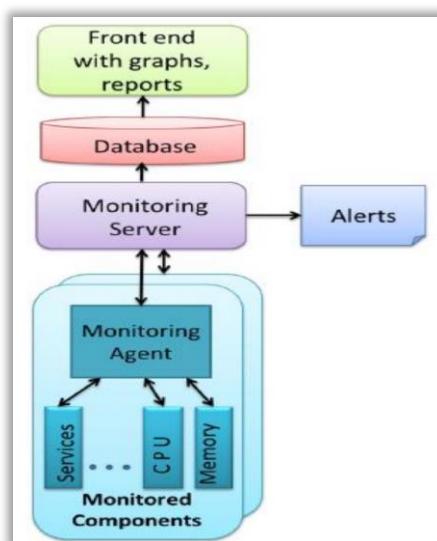
- ⇒ **Commutateurs** : utilisation des ressources, métrologie ;
- ⇒ **Serveurs** : utilisation des ressources.

Supervision des applications et services :

- ⇒ **Disponibilité.**
- ⇒ **Cohérence des réponses aux interrogations.**
- ⇒ **Performances.**

La supervision pour les administrateurs systèmes fournit les informations sur le fonctionnement des systèmes comme l'utilisation CPU, l'occupation de la mémoire physique, l'espace libre des disques durs, l'état de la table de partitionnement du disque, et plusieurs d'autres.

La surveillance IT peut être conçue avec ou sans agent. Les agents sont des programmes indépendants installés sur les périphériques surveillés pour collecter des données sur les données de performances du matériel ou des logiciels et les transférer sur un serveur de gestion. La surveillance sans agent utilise les protocoles de communication existants pour émuler un agent et exercer à peu près les mêmes fonctions.



### c. Avantages de la supervision :

- **Production optimale** : la visibilité que présente la supervision permet au système de ne plus être sujet à des pannes et des temps d'arrêt réguliers. La supervision impacte la production en réduisant les efforts manuels.
- **Sécurité des systèmes** : les erreurs systèmes ou réseau ne sont pas les seuls problèmes que peut subir un système informatique, mais une attaque malveillante qui entraîne des temps d'arrêt. La supervision nous permet de détecter les anomalies liées à la sécurité informatique.
- **Réduction des coûts** : le monitoring permet d'anticiper et de résoudre les pannes plus rapidement un problème qui n'est pas détecté au bon moment et qui peut engendrer des dégâts de plus en plus amplifiés, ce qui mène à des coûts supplémentaires.

## 7. Conclusion :

Cet axe présente les grandes lignes théoriques qui encadrent mon projet et m'aide à le bien placer. J'ai mis l'accent sur le Cloud, le DevOps, le Testing, la conteneurisation et le monitoring avec leurs contextes, définitions et fonctionnements détaillés ainsi que leurs avantages.

Dans le prochain axe, je vais me concentrer sur les besoins derrière mon projet, l'explication de ma solution et la justification de choix des méthodes, approches et outils.

---

*Chapitre III : Analyse des besoins et  
justification des choix*

---

**Je vais aborder dans ce chapitre par ordre chronologique les étapes depuis les analyses jusqu'à la mise en œuvre de ma solution, en passant par la justification des choix**

## **1. Analyse des besoins :**

Les besoins ressentis dans mon contexte s'inscrivent dans la thématique du Cloud et DevOps. L'objectif principal du projet est de plus en plus tourner vers l'automatisation, de passer de l'exécution manuelle des tests qui est bien sûr importante, mais demande plus de temps et de charges à l'exécution automatique qui aide beaucoup dans les cas de tests répétitifs.

Choisir la plateforme Cloud adéquate, les méthodes ainsi que les outils nécessaires pour présenter une solution de plus en plus performante est bien notre mission dans ce projet.

L'analyse des besoins porte sur deux volets : les besoins fonctionnels et les besoins non fonctionnels.

### **a. Besoins fonctionnels :**

- **L'automatisation des tests** : a pour objectif de simplifier autant que possible les efforts de test grâce aux scripts. Le test est alors exécuté selon celui-ci, les résultats sont signalés et comparés aux résultats des essais antérieurs. Son principal intérêt réside dans le fait qu'il permet de gagner du temps et de l'argent.
- **Provisionnement des ressources Cloud** : le choix du fournisseur Cloud doit être judicieux, il est dû à plusieurs facteurs : coût, services, simplicité, évolutivité, tolérance aux pannes et plusieurs d'autres ;
- **Monitoring & alerting** : Il est indispensable d'implémenter un système de monitoring qui va superviser le résultat des tests appliqués sur les portails web du client pour avoir une bonne réactivité de la part de l'équipe Pilotage 24/7.
- **Conteneurisation** : mettre en évidence l'utilité de la conteneurisation et sa valeur ajoutée dans notre contexte ;
- **Intégration continue** : elle représente une complémentarité à la phase de Testing en réduisant le temps nécessaire à chaque phase d'intégration.

### **b. Besoins non fonctionnels :**

- **Extensibilité** : Le système doit pouvoir supporter l'évolution et l'extensibilité de ses composants

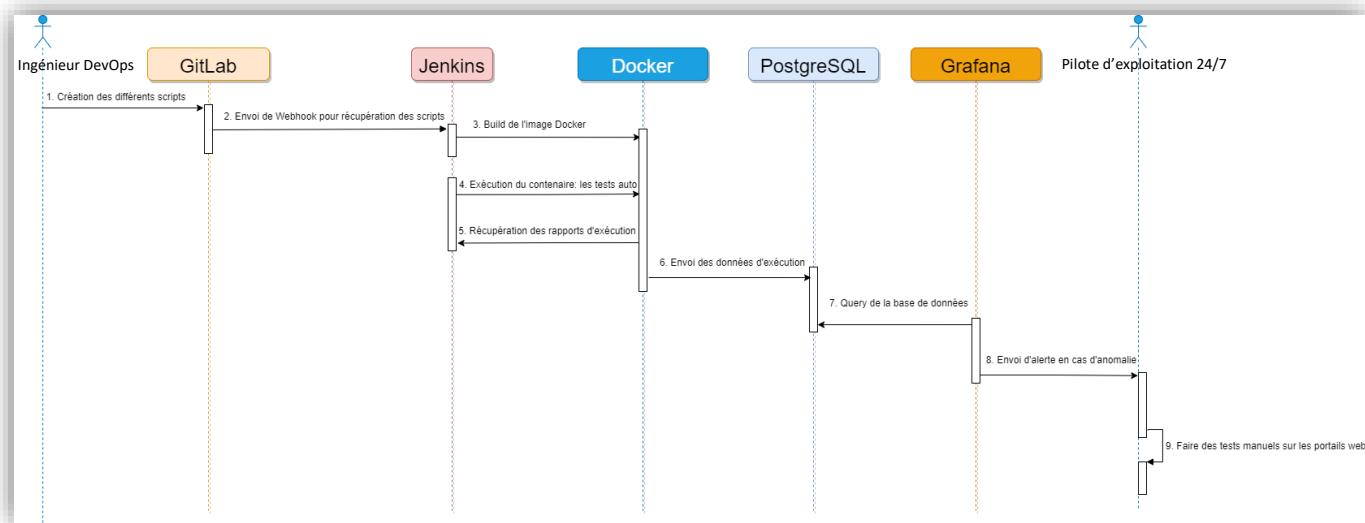
- **Simplicité** : c'est un critère qui facilite le déploiement que ça soit au niveau de l'implémentation des codes ou bien au niveau des différents outils de notre solution ;
- **Performance** : le temps de réponse de chaque outil doit être minimal, et l'efficacité garantie ;
- **Adaptabilité** : tous les outils doivent être adaptables entre eux menant à une complémentarité au niveau de l'architecture globale.

## 2. Conception des tests :

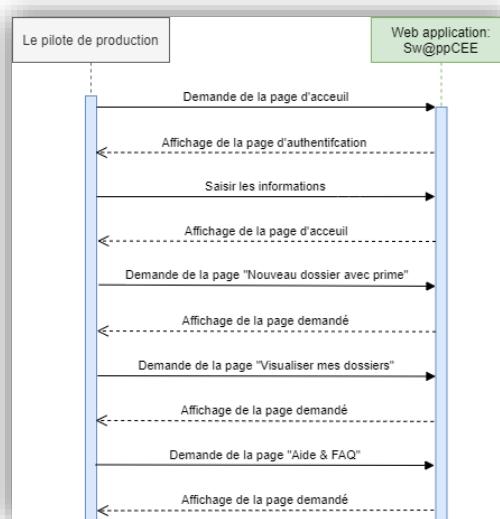
### a. Diagrammes de séquence :

- **Diagramme de séquence de système :**

La figure suivante représente le diagramme de séquence du système :

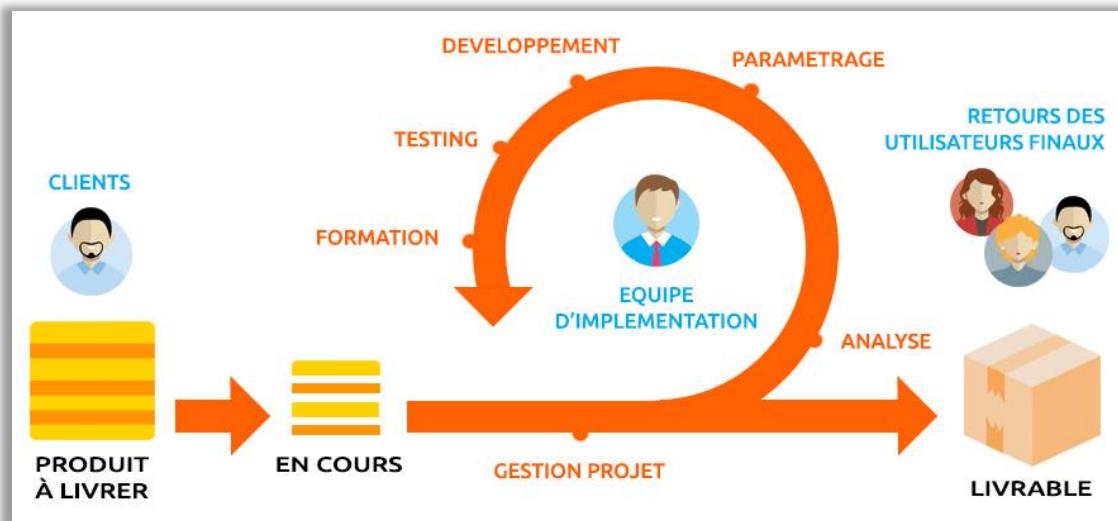


- **Diagramme de séquence de l'application Web SW@PPCEE :**



### 3. Conduite du projet :

#### a. Méthode Agile :



Il s'agit d'une méthodologie de gestion de projet qui découle du Manifeste Agile édité en 2001 développé par plusieurs développeurs logiciels afin d'améliorer les process et réduire les taux d'échec.

Contrairement à la méthode traditionnelle qui consiste à planifier totalement le projet avant la phase développement, la méthode agile place le client au cœur du projet et le fragmente en plusieurs sous-parties que l'équipe qui en a la charge se doit d'atteindre progressivement en ajustant si nécessaire les objectifs pour répondre le plus rapidement possible aux attentes du client.

La méthodologie Agile permet une grande flexibilité. En effet le client est plus libre et il peut changer d'avis en cours de projet. Les imprévus sont aussi pris en compte et l'équipe projet peut réagir dans un temps très court.

Elle favorise également la collaboration et la communication fréquentes avec le client. Le client est ainsi fortement impliqué dans le projet ce qui permet d'instaurer une relation de confiance entre l'équipe et le client.

Les étapes de la méthode agile se présentent comme suit :

- **Etape 1 :** définir le cadre et les exigences du projet.

- **Etape 2 :** établir et prioriser l'ensemble des actions à réaliser pour atteindre l'objectif final tout en se référant au cadre du projet ainsi que les exigences des clients. Cet ensemble est connu sous le nom du Backlog, et on peut définir deux colonnes principales : Le Backlog complet, dans lequel on ajoute toutes les demandes et besoins des clients, et le Backlog du sprint, au sein duquel nous retrouvons les tâches sur lesquelles nous allons travailler pendant le sprint.
- **Etape 3 :** travailler sur chacune des tâches définies dans le Backlog et organiser des réunions au quotidien pendant toute la durée du sprint afin de synchroniser l'avancement de l'équipe.
- **Etape 4 :** réaliser une revue de sprint dans laquelle on récolte le maximum de feedbacks auprès des clients et collaborateurs.
- **Etape 5 :** recommencer toutes les étapes, la méthode agile est enchaînée.

## b. Diagramme de Gantt :

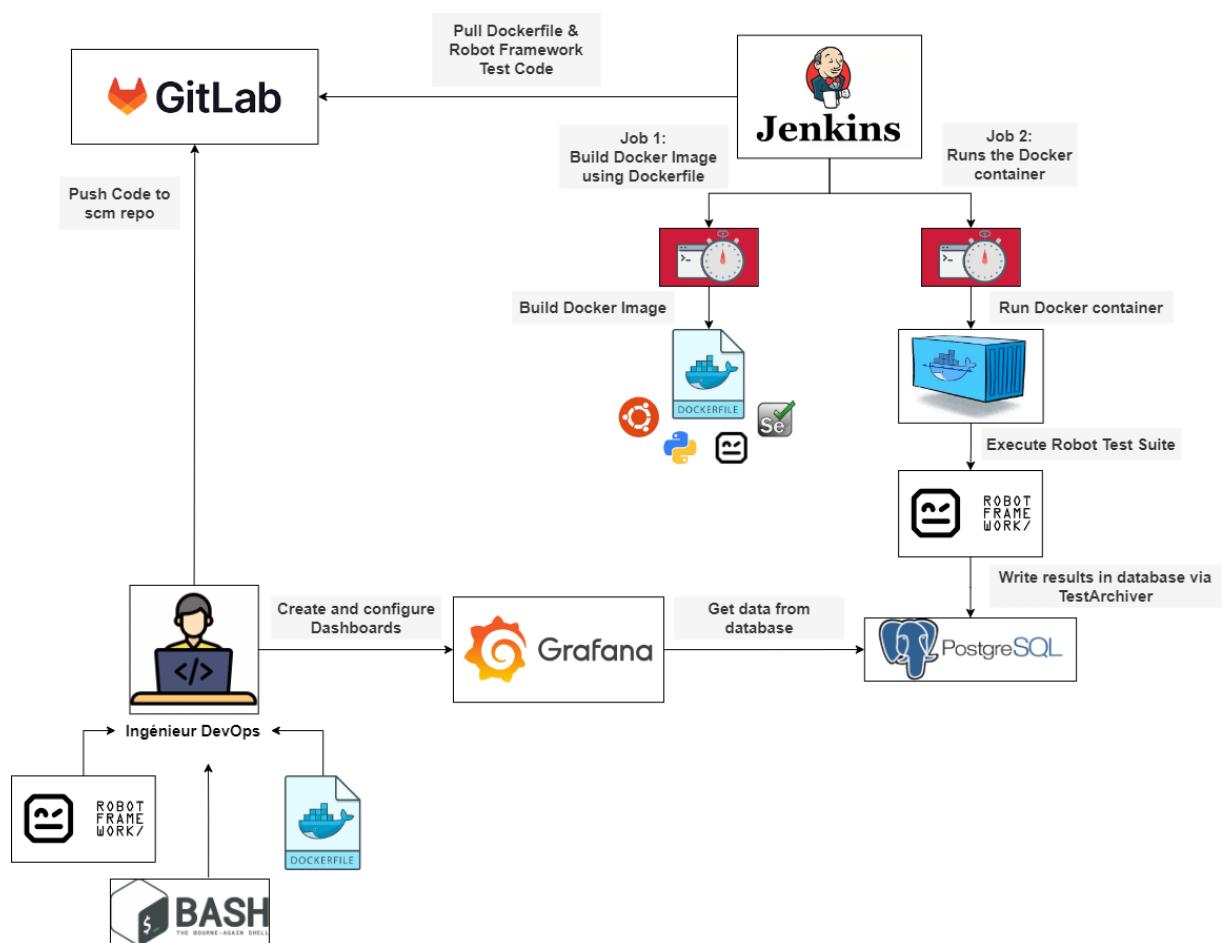
Nous avons opté pour une planification dès le début du projet, ce qui nous a permis de dresser un inventaire des tâches à accomplir. Nous avons donc estimé la durée et la dépendance des tâches en fonction de l'expérience acquise lors de nos études et projets académiques.



## 4. Présentation de la solution :

L'objectif est de mettre en place une solution d'automatisation (Processus et Scripts) pour l'équipe de supervision 24/7 et les administrateurs afin de leur préparer des Dashboards Webtesting adéquat à la supervision sur les plans suivants : Testing, DevOps et monitoring en se basant sur les approches correspondantes et sur les outils efficaces et rentables.

### a. Architecture globale :



## b. Etapes de la réalisation :

- ⇒ **Etape 1** : cette étape marque le début du projet et consiste à développer des tests fonctionnels automatisés (Webtesting) qui assurent le bon fonctionnement des portails web.
- ⇒ **Etape 2** : la préparation de l'environnement d'exécution des tests Robot Framework sur une instance AWS.
- ⇒ **Etape 3** : la conteneurisation qui consiste à conteneuriser les tests moyennant l'outil Docker.
- ⇒ **Etape 4** : la création du processus d'intégration continue au niveau de Gitlab et Jenkins et sa liaison avec AWS pour la création des ressources.
- ⇒ **Etape 5** : la mise en place directe des stacks de monitoring Grafana.

## 5. Outils, approches, concepts utilisés et justifications :

### a. Gestion de projet :

Pour effectuer et suivre tous les tests, l'équipe ENGIE IT a choisi d'utiliser l'outil Jira. Jira est un logiciel de suivi de tickets, de projets et gestion des incidents.



C'est dans cet outil que les besoins sont listés. Les managers d'équipes s'occupent alors de bien les définir en tickets, et de les répartir entre les développeurs et testeurs. Cet outil est tout à fait adapté par les équipes Agile.

### b. Les fournisseurs Cloud :

Comme cité dans le deuxième chapitre, les fournisseurs Cloud deviennent de plus en plus nombreux. Trois géants se distinguent et entrent en compétitivité : **Amazon Web Services (AWS)**, **Microsoft Azure** et **Google Cloud**.

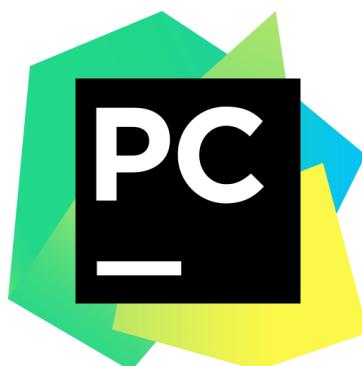
Le choix du fournisseur AWS est dû au partenariat entre Amazon et Capgemini. Cette solution aide les entreprises à aller plus loin en termes de technologies Cloud.



Ce partenariat avec AWS renforce le programme de services Cloud de Capgemini, « *Cloud Choice* », qui est le fruit de la collaboration avec un large écosystème de partenaires Cloud et inclut la gamme de services suivante : évaluation et stratégie Cloud d'entreprise, preuve de concept, migration d'applications dans le Cloud, plateforme as-a-service, infrastructure as-a-service, processus métier as-a-service et cybersécurité.

### c. Editeur de code :

PyCharm est un environnement de développement intégré utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django.



#### d. Automatisation des tests :

Le choix de l'outil d'automatisation était en faveur de Robot Framework :

<b>Environnement et langage</b>	 Robot Framework	 Python
<b>Bibliothèque</b>	 SeleniumLibrary	

#### Robot Framework :

Robot Framework est un outil d'automatisation de test, développé au départ par Nokia Networks puis devenu open source, qui dispose d'une communauté active et réactive.

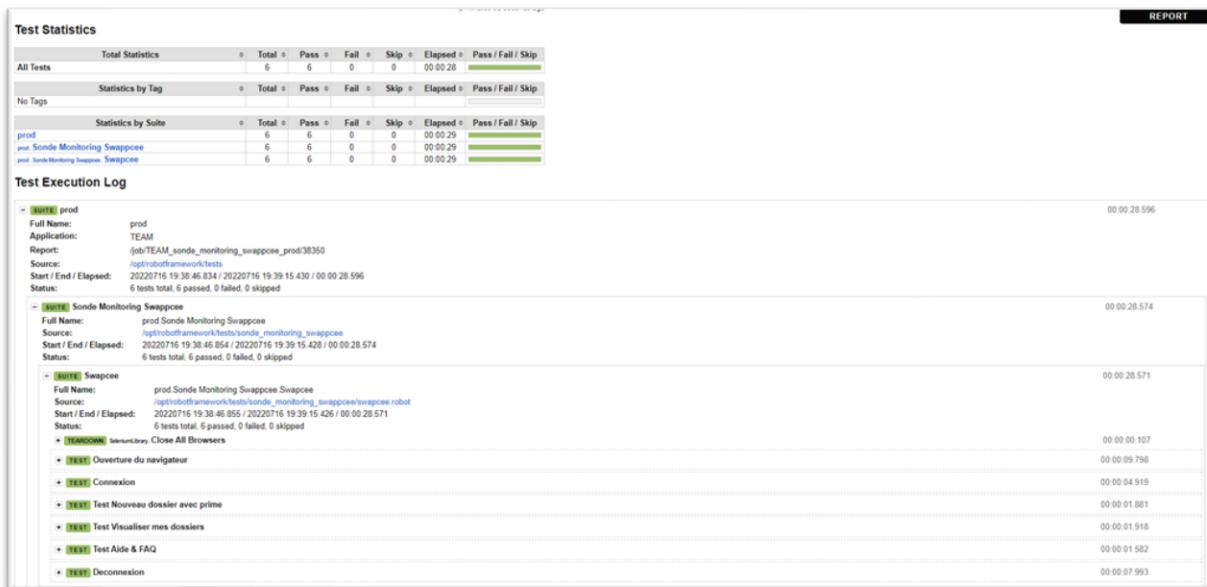
Robot Framework permet d'automatiser des tests IHM, des APIs, des BDD, serveurs et même des environnements mobiles (IOS, Android) ... Bref, il permet d'exécuter un très large panel de tests d'intégration et de tests système. Il nécessite un environnement Python pour lancer l'exécution des tests.

Cet outil se base sur des librairies (développées en Python ou Java) qui recensent les mots-clés nécessaires à l'écriture des tests (un mot-clé équivaut à une méthode). Par exemple, pour les tests IHM, Robot Framework utilise la librairie Selenium (fonctionnalités, webdrivers...). Pour les tests IHM, il existe des extensions (chrome ou firefox) qui permettent d'enregistrer les actions utilisateurs et qui génère un script pouvant être rejoué.

Les points forts remontés de Robot Framework sont les suivants :

- ⇒ **Un fort gain de productivité (temps d'exécution et d'analyse)**
- ⇒ **La flexibilité des tests (IHM, API, bout en bout...) et la possibilité à être multi-navigateur**

⇒ Les tests sont simples à écrire par des développeurs et des testeurs ainsi que les rapports générés sont complets et facile à lire.



## Python :

Python est le langage de programmation open source le plus employé par les informaticiens. Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels. En effet, parmi ses qualités, Python permet notamment aux développeurs de se concentrer sur ce qu'ils font plutôt que sur la manière dont ils le font. Il a libéré les développeurs des contraintes de formes qui occupaient leur temps avec les langages plus anciens. Ainsi, développer du code avec Python est plus rapide qu'avec d'autres langages.

## SeleniumLibrary :

SeleniumLibrary est une bibliothèque de test Web pour Robot Framework qui utilise l'outil Selenium en interne. Le projet est hébergé sur Github et les téléchargements peuvent être trouvées à partir de PyPI.

## e. Gestion de code source :

Les logiciels de gestion de code source sont des outils permettant de travailler avec efficacité sur des projets de développement logiciel, grâce à une méthode de modification de code collaborative, au suivi des modifications et à la gestion des modifications.

Ci-dessous un aperçu sur les outils les plus répandus :



GitLab est un gestionnaire web libre de référentiel git écrit en Ruby et lancé en 2011. GitLab fournit une gestion centralisée des référentiels Git, permettant aux utilisateurs d'avoir le contrôle complet de leurs référentiels ou projets. Il permet également d'automatiser les tests ainsi que la livraison de codes.



GitHub est un service de référentiel web de gestion de code source, créé en 2008. Il garantit un espace aux développeurs pour stocker leurs projets et concevoir des logiciels en parallèle. GitHub fournit des fonctions de collaboration et de contrôle d'accès.



Bitbucket est un service web d'hébergement et de gestion de développement logiciel utilisant le logiciel de gestion de versions Git développé en Python et lancé en 2008. Il permet aux développeurs de disposer d'un espace unique pour planifier des projets, collaborer autour du code, tester et déployer.

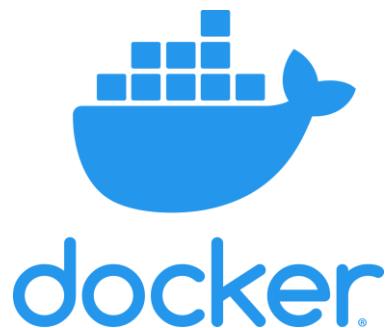
Ci-dessous un tableau récapitulant la comparaison entre ces outils :

	 GitLab	 GitHub	 Bitbucket
<b>Open source</b>	Oui	Non	Non
<b>Licence</b>	Gratuit (version communauté)	Gratuit (version communauté)	Payant
<b>Gestion de version</b>	Git	Git	Git, Mercurial
<b>Intégration Continue</b>	Intégré et à travers d'autres outils	A travers d'autres outils	A travers d'autres outils
<b>Suivi de problèmes</b>	Oui	Oui	Avec Jira

D'après l'étude menée sur ces outils et une comparaison basée sur les critères prioritaires de notre solution nous avons décidé de travailler à l'aide de GitLab comme outil de gestion de code.

## f. Conteneurisation :

Le choix d'un outil de conteneurisation était en faveur de Docker : c'est l'outil le plus populaire.



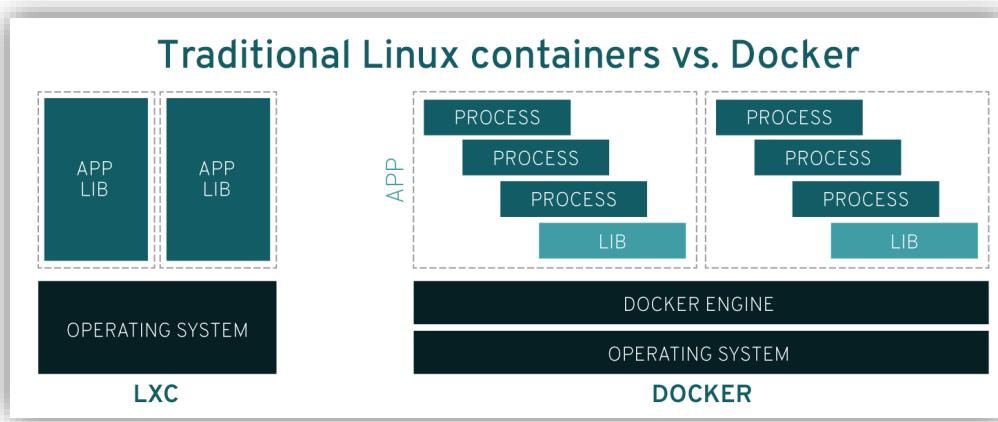
Docker est un outil gratuit lancé en 2013 qui peut empaqueter une application et ses dépendances dans un conteneur isolé, pouvant être exécuté sur n'importe quel serveur.

Docker repose sur la décomposition des applications : c'est-à-dire la capacité de réparer ou de mettre à jour une partie d'une application sans devoir désactiver l'ensemble de cette dernière.

Il réutilise les couches qui composent un fichier image Docker pour la construction de nouveaux conteneurs, accélérant ainsi le processus de construction. Docker permet de restaurer les versions précédentes, ce qui est considéré comme avantage pour cet outil.

De plus et contrairement à la virtualisation traditionnelle, Docker permet un déploiement rapide en facilitant la gestion des conteneurs sans avoir à redémarrer le système d'exploitation.

A la lumière de ces considérations qui justifient notre choix pour cet outil, nous avons utilisé Docker au niveau du Testing pour mettre en évidence son utilité et sa force.



### **g. Intégration continue :**

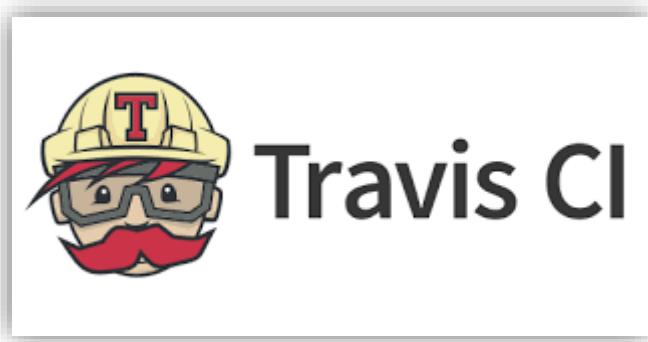
Nous avons besoin, afin de mener notre projet vers l'approche DevOps et spécialement de faire un choix judicieux d'un outil d'intégration continue.

Voici un aperçu sur les trois outils les plus populaires :



Jenkins est un outil d'intégration continu Open Source créé en 2011 et développé à l'aide du langage de programmation Java. Il permet de tester et de rapporter les changements effectués sur une large base de code en temps réel. Il supporte les outils SCM tels que Subversion, git, etc. Il permet d'exécuter des scripts Shell et des projets Ant ou Maven et dispose de nombreux plug-ins qui le rendent compatible avec tous

les langages de programmation et une grande majorité de systèmes de contrôle de version et de référentiels. Jenkins permet aux utilisateurs de concevoir et livrer des applications à grande échelle rapidement et supporte la conception, le déploiement et l'automatisation dans la plupart des projets.



Travis CI est un outil logiciel d'intégration continue Open Source écrit en Ruby et publié en 2011, permettant de compiler, tester et déployer des projets hébergés sur GitHub. Les exécutions de builds et de tests sont déclenchées, automatiquement toutes les fois qu'un commit est réalisé, et poussées vers un référentiel GitHub. La configuration consiste à placer un fichier `travis.yml` dans le répertoire racine du référentiel. Travis CI est gratuit pour les projets Open Source, payant pour les projets commerciaux ou privés.



CircleCI est une plateforme d'intégration continue et de déploiement continu basée sur le cloud (SaaS) créée en 2011, permettant d'automatiser le processus de construction, de test et de déploiement. CircleCI possède une configuration YAML facile à manipuler, à installer et ne nécessite pas de serveur dédié pour fonctionner. Il est convenable aux petits projets qui nécessitent un démarrage rapide.

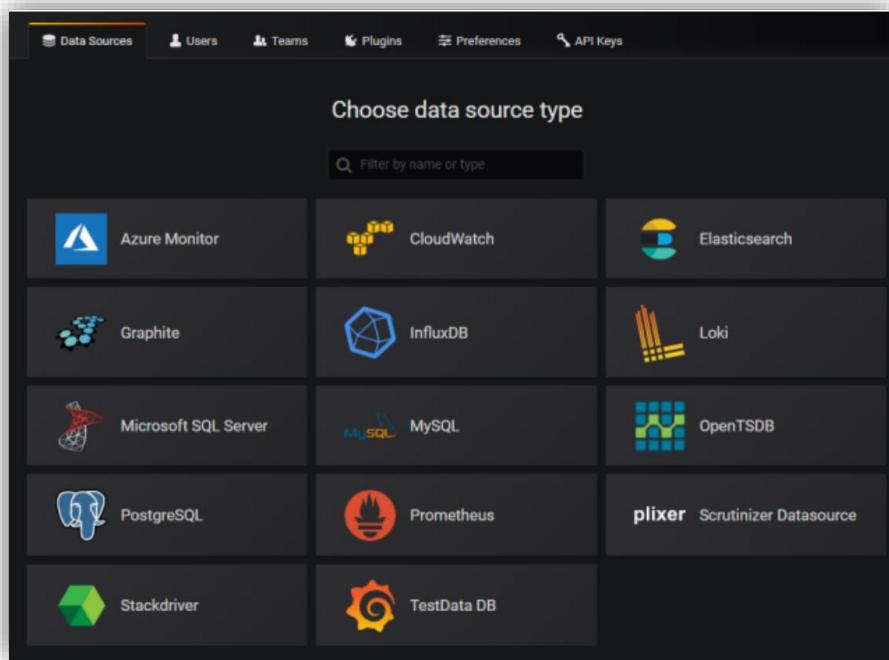
- Comparaison et conclusion :

			
<b>Opensource</b>	Oui	Oui	Non
<b>Licence</b>	Gratuit	Gratuit, avec version payante	Payant
<b>Installation</b>	Facile	Difficile	Aucune (SaaS)
<b>Gestionnaire de code source</b>	GitLab, GitHub, Bitbucket, TFS, CVS, Perforce	GitHub, Bitbucket	GitHub, Bitbucket
<b>Systèmes d'exploitation</b>	Linux, MacOs, Windows, Unix	Linux, MacOs	Linux, MacOs

#### h. Outil de visualisation et alerting :



Grafana est un logiciel libre permettant la visualisation des métriques et données. Il permet de réaliser des tableaux de bord et des graphiques depuis plusieurs sources qui sont généralement des bases de données contenant les métriques collectées tels que Prometheus, Influxdb, Postgresql, Zabbix et autres.



Grafana fournit tous les outils pour agréger, organiser et analyser des données de bases de données hétérogènes ou d'applications différentes en toutes flexibilité.

Grafana fournit des plugins de type panels pour mettre en forme les données collectées.

Ces données peuvent être visualisées sous des formes différentes pour les séries chronologiques et non chronologiques : tableaux, histogrammes, diagrammes circulaires ou du texte.



Les points suivants nous ont poussé vers le choix de Grafana comme outil de visualisation de nos métriques :

- **Open source** : Grafana est gratuit et nous pouvons l'enrichir avec notre propre code, une grande communauté de développeurs assure son amélioration, la correction des erreurs et la sécurité de l'outil.
- **Multiplateformes et multi sources de données** : Grafana est configurable sur Linux, Mac, Windows. L'outil peut afficher des données stockées dans plusieurs types de bases de données (Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, PostgreSQL et autres) ;
- **Popularité** : la puissance de cette solution suscite la confiance d'un grand nombre de développeurs, administrateurs système et entreprises ;
- **Personnalisable** : Grafana est conçu pour répondre exactement à tous les besoins qui concernent la visualisation. Il offre la possibilité de créer et modifier les Dashboards selon les préférences. Il procure également la possibilité d'y intégrer d'autres plugins pour plus de flexibilité ;
- **Simplicité** : Grafana ne prend pas beaucoup de temps pour arriver à faire le nécessaire avec.

---

## *Chapitre IV : Réalisation du projet*

---

**Nous allons aborder ce quatrième et dernier chapitre la partie de la réalisation et la mise en œuvre des tests automatisés fonctionnels des interfaces de l’application web « SwappCEE », et l’intégration de ces tests avec Jenkins, ainsi que l’environnement de développement, langage de programmation et la préparation de l’environnement de travail sur GitLab et la configuration des Dashboards Grafana**

## **1. Introduction :**

Après avoir terminé la partie de la documentation, de l'analyse des besoins et le benchmark des outils, j'aborde, dans ce chapitre, la partie de la réalisation qui constitue le dernier axe de notre rapport.

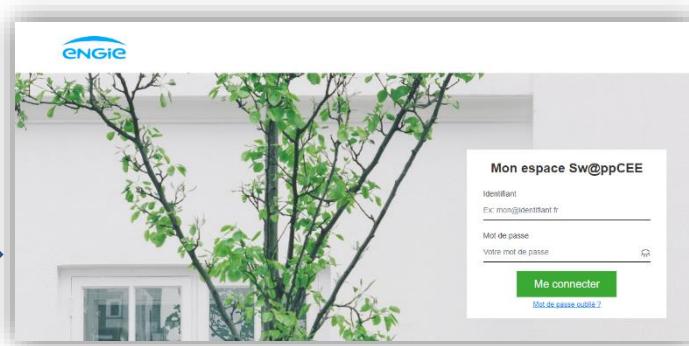
Je vais commencer tout d'abord par les scripts Robot Framework que j'ai réalisé. Je vais ensuite entamer la partie monitoring et puis la conteneurisation pour arriver finalement à l'intégration continue.

Dans cet axe, je vais exposer l'architecture détaillée de chaque partie de l'architecture globale et je vais démontrer également toutes les étapes de la réalisation de ce projet.

## **2. Processus des cas de tests automatisés :**

Avant de décrire le format des tests, présentons un cas d'usage qui servira de support à toutes les explications ultérieures. L'exemple que nous utiliserons sera sur [SW@PPCEE](#), un site qui permet aux clients de constituer un dossier travaux.

Page d'authentification de Swappcee →



Imaginons qu'en tant que personnel de SW@PPCEE, nous souhaitons valider le fonctionnement de ce système de manière automatique. Concrètement, la séquence d'actions sera la suivante :

⇒ **Etape 1 :** Accéder à la page <https://swappcee.engie.fr/>

⇒ **Etape 2 :** Entrer le login et mot de passe.

⇒ **Etape 3 :** Envoyer le formulaire.

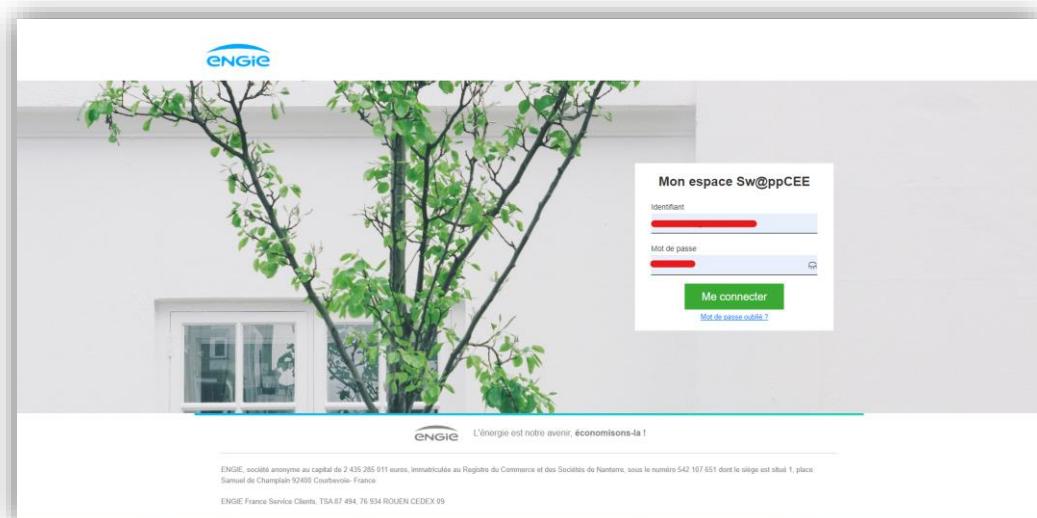
⇒ **Etape 4 :** Attendre le chargement de la page d'accueil.

⇒ **Etape 5 :** Vérifier qu'un mot-clé signifiant la bonne santé du site est bien présent.

Nous allons nous baser sur ce scénario pour assurer la disponibilité des différentes applications web.

### a. Test sur l'interface Login :

Comme le montre la figure suivante la page d'authentification, en exécutant le test pour vérifier le fonctionnement de cette page, le scénario de test va saisir les informations d'authentification. Le test réussi si l'application procède à nous rediriger vers la prochaine page.



REPORT	
- <b>SEND Monitoring Swappce</b>	00:00:28.668
Full Name: prod.Sonde Monitoring Swappce	
Source: /nginsoftframework/testscenarios_monitoring_swappce_swappce	
Start / End / Elapsed: 2022/07/18 20:38:48.155 / 2022/07/18 20:39:16.829 / 00:00:28.674	
Status: 6 tests total, 6 passed, 0 failed, 0 skipped	
- <b>SEND Swappce</b>	00:00:28.668
Full Name: prod.Sonde Monitoring Swappce_Swappce	
Source: /nginsoftframework/testscenarios_monitoring_swappce/swappce_nginsoft	
Start / End / Elapsed: 2022/07/18 20:38:49.157 / 2022/07/18 20:39:16.825 / 00:00:28.674	
Status: 6 tests total, 6 passed, 0 failed, 0 stopped	
• <b>TESTS</b> SondeMonitoring Close All Browsers	00:00:00.162
• <b>TESTS</b> Overture du navigateur	00:00:00.069
- <b>TEST Connexion</b>	00:00:04.737
Full Name: prod.Sonde Monitoring Swappce_Swappce Connexion	
Start / End / Elapsed: 2022/07/18 20:38:58.124 / 2022/07/18 20:39:02.951 / 00:00:04.737	
Status: 10 tests total, 10 passed, 0 failed, 0 skipped	
• <b>RETRY</b> SondeMonitoring Wait Until Page Contains Element \$[User].30	00:00:00.019
• <b>RETRY</b> SondeMonitoring Input Text \$[User].30	00:00:00.027
• <b>RETRY</b> SondeMonitoring Input Text \$[User].30	00:00:00.038
• <b>RETRY</b> SondeMonitoring Click Element \$[User].30	00:00:00.209
• <b>RETRY</b> SondeMonitoring Wait Until Page Contains Bienvenue sur votre portail.30	00:00:03.104
• <b>RETRY</b> SondeMonitoring Capture Page Screenshot 2 SWAPPCE_Accueil.png	00:00:00.725
Documentation: Takes a screenshot of the current page and embeds it into a log file.	
Start / End / Elapsed: 2022/07/18 20:39:02.136 / 2022/07/18 20:39:02.951 / 00:00:00.725	
• <b>INFO</b>	

Après l'authentification, la page d'accueil de l'application web s'affiche (Voir rapport d'exécution Robot Framework ci-dessus)

## b. Test sur Formulaire du Dossier avec prime :

Ici, le test bascule vers la page « Nouveau Dossier avec prime » et vérifie le bon fonctionnement.

## c. Test sur Visualiser mes dossiers :

The screenshot shows the ENGIE website interface for managing work orders. At the top, there's a blue header bar with the ENGIE logo and navigation links: 'Accueil CEE', 'Nouveau dossier avec prime', 'Visualiser mes dossiers', and 'Aide & FAQ'. Below the header, a message indicates new offers available from June 15, 2022. The main content area is titled 'Dossier travaux FIT' and displays a table of work orders. The columns include: Numéro, Nom et prénom du bénéficiaire, Ville travaux, Cumac et prime, Situation, Statut, Mise à jour, and Actions. There are 5 items listed in the table.

This screenshot shows a test log for the 'Visualiser mes dossiers' feature. It includes a timestamped log entry and a screenshot of the application interface. The log details the steps taken to perform the test, including selecting the 'Visualiser mes dossiers' tab and capturing a screenshot of the resulting page. The screenshot shows the same 'Dossier travaux FIT' view as the previous image, with a redacted URL at the top.

## d. Test sur Aide & FAQ :

The screenshot shows the 'Aide & FAQ' section of the ENGIE website. The top navigation bar is identical to the 'Visualiser mes dossiers' page. The main content area is titled 'FAQ aux questions' and lists three frequently asked questions with '+' icons to expand them: 'Comment constituer un dossier CEE en ligne?', 'Comment suivre un dossier?', and 'Lexique'. At the bottom of the page, there's a small note about energy saving and a copyright notice.

- **TEST: Test Aide & FAQ**

Full Name: prod.Sonde Monitoring Swappee.Swappee.TestAide & FAQ  
Start / End / Elapsed: 20220718 20:39:06.934 / 20220718 20:39:08.520 / 00:00:01.586  
Status: **PASS**

- **KEYWORD: Report Test Aide & FAQ**  
Start / End / Elapsed: 20220718 20:39:06.935 / 20220718 20:39:08.520 / 00:00:01.585
  - + **KEYWORD: SeleniumLibrary** Wait Until Page Contains Element \${TabAide&FAQ} 30
    - + **KEYWORD: SeleniumLibrary** Click Element \${TabAide&FAQ}
    - + **KEYWORD: SeleniumLibrary** Wait Until Page Contains Faire aux questions, 30
    - + **KEYWORD: SeleniumLibrary** Capture Page Screenshot 4 SWAPPCEE\_Aide&FAQ.png
Documentation: Takes a screenshot of the current page and embeds it into a log file.  
Start / End / Elapsed: 20220718 20:39:07.738 / 20220718 20:39:08.520 / 00:00:00.782

20:39:08.520 [INFO]

## e. La déconnexion :

- **TEST: Deconnexion**

Full Name: prod.Sonde Monitoring Swappee.Swappee.Deconnexion  
Start / End / Elapsed: 20220718 20:39:08.521 / 20220718 20:39:16.662 / 00:00:08.141  
Status: **PASS**

- **KEYWORD: Report Deconnexion**  
Start / End / Elapsed: 20220718 20:39:08.521 / 20220718 20:39:16.662 / 00:00:08.141
  - + **KEYWORD: SeleniumLibrary** Wait Until Page Contains Element \${lienDeconnexion}, 30
    - + **KEYWORD: SeleniumLibrary** Click Element \${lienDeconnexion}
    - + **KEYWORD: SeleniumLibrary** Wait Until Page Contains Etes-vous sûr de vouloir vous déconnecter ?, 30
      - + **KEYWORD: SeleniumLibrary** Capture Page Screenshot 5 SWAPPCEE\_DeconnexionValidation.png
      - + **KEYWORD: SeleniumLibrary** Click Element \${boutonDeconnexion}
      - + **KEYWORD: Built-in** Sleep 3s
      - + **KEYWORD: SeleniumLibrary** Capture Page Screenshot 6 SWAPPCEE\_Deconnexion.png
Documentation: Takes a screenshot of the current page and embeds it into a log file.  
Start / End / Elapsed: 20220718 20:39:14.025 / 20220718 20:39:16.662 / 00:00:02.637

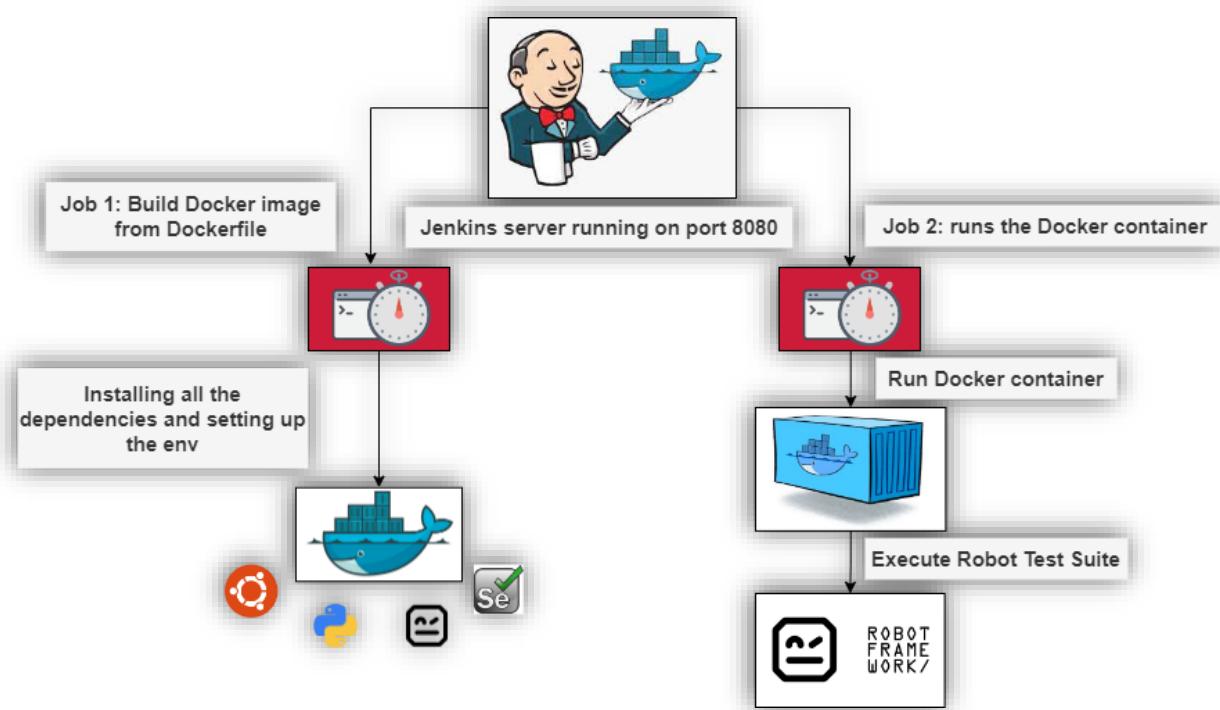
20:39:16.662 [INFO]

## 3. Conteneurisation :

### a. Architecture :

Avec la technologie Docker, nous pouvons traiter les conteneurs comme des machines virtuelles très légères et modulaires. En outre, ces conteneurs nous offrent une grande flexibilité : nous pouvons les créer, déployer, copier et déplacer d'un environnement à un autre, ce qui vous permet d'optimiser nos applications pour le cloud.

Nous proposons l'architecture suivante qui illustre l'utilisation de la conteneurisation pour le déploiement des tests Robot Framework.



Voilà donc une infrastructure que j'ai mise en place avec jenkins un très bon outil d'intégration continue et docker qui est un système de container linux ultra léger basé sur les cgroups, lxc et aufs.

L'idée est la suivante pour un build:

- Jenkins créé une image Docker depuis un dockerfile.
- Jenkins exécute le container Docker.
- Les tests Robot Framework se lance sur le container.
- Jenkins récupère le résultat du Build (logs, métadonnées, artifacts) à travers le Plugin Robot Framework installé sur Jenkins.

De cette manière le Build est toujours lancé dans un environnement sain et contrôlé, vous n'avez pas besoin de maintenir des slaves Jenkins et ça scale particulièrement

bien puisque vous pouvez rajouter des serveurs docker capables de lancer pleins de containers.

## b. Dockerfile :

Docker peut créer des images automatiquement en lisant les instructions d'un fichier Docker. Un Dockerfile est un document texte qui contient toutes les commandes qu'un utilisateur peut appeler sur la ligne de commande pour assembler une image. À l'aide de docker Build, les utilisateurs peuvent créer une build automatisée qui exécute successivement plusieurs instructions de lignes de commande.

```
1 #Base Image
2 FROM ubuntu
3 LABEL version="latest" maintainer="Lonewolf <azouaghachraf@gmail.com>"
4
5 #update the image
6 RUN apt-get update
7 RUN apt install software-properties-common -y
8 RUN add-apt-repository ppa:deadsnakes/ppa
9 RUN apt-get update
10 RUN apt-get upgrade -y
11 #install python
12 RUN apt install -y python3.8
13 RUN apt install -y python3-pip
14 RUN pip3 install psycopg2-binary
15 RUN pip3 install testarchiver
16 RUN python3 --version
17
18 #install robotframework and seleniumlibrary
19 RUN pip3 install robotframework
20 RUN pip3 install robotframework-seleniumlibrary
21 RUN pip3 install robotframework-seleniumlibrary
22
23 #The following is needed for Chrome and Chromedriver installation
24 RUN apt-get install -y xvfb
25 RUN apt-get install -y zip
26 RUN apt-get install -y wget
27 RUN apt-get install ca-certificates
28 RUN apt-get install -y libnss3-dev libasound2 libxss1 libappindicator3-1 libindicator7 gconf-service libgconf-2-4 libpango1.0-0 xdg-utils fonts-liberation
29 RUN apt-get install -y libcurl3-gnutls libcurl3-nss libcurl14 libgbm1
30 RUN wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
31 RUN dpkg -i google-chrome*.deb
32 RUN rm google-chrome*.deb
33 RUN wget -N http://chromedriver.storage.googleapis.com/99.0.4844.51/chromedriver_linux64.zip
34 RUN unzip chromedriver_linux64.zip
35 RUN chmod +x chromedriver
36 RUN cp /chromedriver /usr/local/bin
37 RUN rm chromedriver_linux64.zip
38
39 # Set the reports directory environment variable
40 ENV ROBOT_REPORTS_DIR /opt/robotframework/reports
41
42 # Set the tests directory environment variable
43 ENV ROBOT_TESTS_DIR /opt/robotframework/tests
44
45 RUN mkdir -p ${ROBOT_REPORTS_DIR}
46 RUN mkdir -p ${ROBOT_TESTS_DIR}
47
48 COPY bin/runtest.sh /opt/robotframework/bin/
49 COPY bin/db_config.json /opt/robotframework/bin/
50
51 RUN chmod +x /opt/robotframework/bin/*
52
53 ENV PATH=/opt/robotframework/bin:$PATH
54
55 COPY Tests/ /opt/robotframework/tests/
56
57 CMD ["runtest.sh"]
```

La première chose que doit-être faite est de créer un fichier nommé "Dockerfile", puis on définit dans celui-ci l'**image que vous allez utiliser comme base**, grâce à l'instruction `FROM`. Dans mon cas, j'ai utilisé une image de base Ubuntu LTS.

L'instruction `FROM` n'est utilisable qu'une seule fois dans un Dockerfile.

Ensuite, j'ai utilisé l'instruction `RUN` pour exécuter une commande dans mon conteneur. Avec cette commande, j'ai préparé l'environnement d'exécution tout en installant tout d'abord Python, le driver chrome, Robot Framework avec la librairie Selenium, TestArchiver et quelques autres dépendances...

Après, j'ai utilisé l'instruction `ENV` qui permet de déclarer une variable d'environnement. Dans ce cas, j'ai déclaré deux variables :

`${ROBOT_REPORTS_DIR}` = Stock le chemin des rapports de tests Robot Framework.

`${ROBOT_TESTS_DIR}` = Stock le chemin des tests ayant l'extension « `.robot` »

Puis, la commande `COPY` qui permet d'ajouter un fichier dans l'image.

Et finalement, la commande `CMD` qui permet d'exécuter une commande au démarrage du conteneur (dans notre cas, on exécute un script bash)

Ci-dessous les logs montrant les étapes effectuées lors du Build du Dockerfile sur Jenkins :

```

Sortie de la console
Started by user LoneWolf
Running as $SYSTEM
Building workspace /var/lib/jenkins/workspace/BuildingJob
The recommended git tool is: NONE
using credential bff447cf-6444-4139-a2e2-61c9e1de258
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/BuildingJob/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://gitlab.com/azouaghachraf/CheckFlushAHM # timeout=10
Fetching upstream changes from https://gitlab.com/azouaghachraf/CheckFlushAHM
> git --version # timeout=10
> git -v # timeout=10
> git version 'git version 2.20.1'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://gitlab.com/azouaghachraf/CheckFlushAHM+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision cd9ee015381623aaa0f19f338dc2a0aeef96 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout cd9ee015381623aaa0f19f338dc2a0aeef96 # timeout=10
Commit message: "Update libvirt-test"
> git rev-list --no-walk 7a1c708295fd19a5cf5d58ea74c0e87c181a # timeout=10
[BuildingJob] $ /bin/sh -xe /tmp/jenkins1679577297683809397.sh
* sudo docker build -t lonewolf/webtestingImage /var/lib/jenkins/workspace/BuildingJob
Sending build context to Docker daemon 267.3kB

Step 1/39 : FROM ubuntu
--> 41449fb348
Step 2/39 : RUN apt-get update
--> Using cache
--> daf1d449725
Step 3/39 : RUN apt-get update
--> Using cache
--> b4d44652c2eb
Step 4/39 : RUN apt install software-properties-common -
--> Using cache
--> 877e548da279
Step 5/39 : RUN add-apt-repository ppa/deadsnakes/ppa
--> Using cache

```

### c. Script Bash :

```

runttest.sh 422 bytes
1 #!/bin/sh
2
3 # Clean Reports folder
4 rm -f /opt/robotframework/reports/*
5
6 # Run Robot Tests
7 robot -M Application:application -d /opt/robotframework/reports /opt/robotframework/tests/Team_Swappcee.robot
8
9 #Run TestArchiver
10 testarchiver --config /opt/robotframework/bin/db_config.json --dont-require-ssl --team 'TEAM' --series jenkins_build_no --format robotframework /opt/robotframework
11

```

Ce script est exécuté au démarrage du conteneur avec la commande CMD, il commence tout d'abord par nettoyer les rapports obsolètes. Ensuite, il procède à lancer les scripts de tests Robot Framework avec la commande « robot » tout en précisant quelques informations comme le nom de l'application, le répertoire des rapports générés, et l'emplacement du script Robot Framework à exécuter. Une fois les tests sont achevés, un programme nommé « TestArchiver » s'occupe de la création des métadonnées et l'insertion des informations et résultats d'exécution des tests dans une base de données PostgreSQL hébergée dans le serveur Host, ci-dessous le fichier JSON utilisé afin de se connecter à la base de données :

```

db_config.json 180 bytes
1 {
2     "db_engine": "postgresql",
3     "database": "robot_results",
4     "host": "176.31.150.100",
5     "port": "5432",
6     "user": "robot",
7     "password": "XXXXXXXXXX",
8     "require_ssl": "False"
9 }
10
11

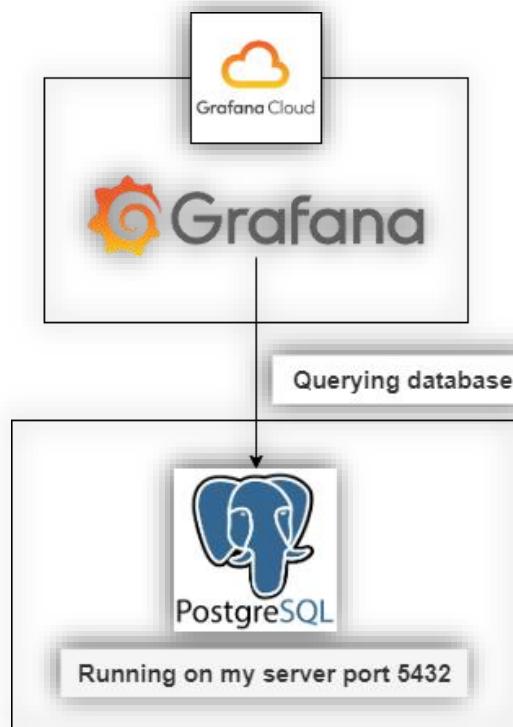
```

## 4. Monitoring :

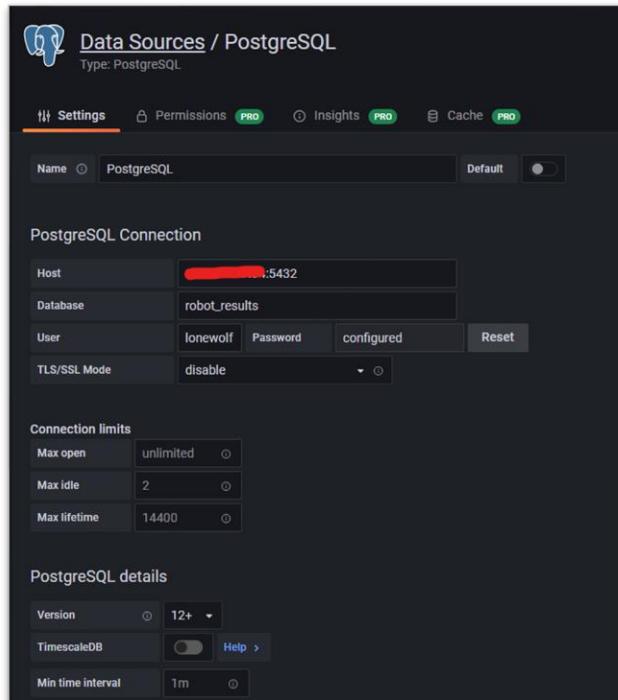
Grafana est une plateforme open source de monitoring, analyse et visualisation des données systèmes en temps réel. L'objectif de cette solution est de présenter facilement et de façon intuitive une grande quantité de données issues de sources différentes.

Après réception des données, Grafana les analyse et les présente sur des Dashboards intuitifs, faciles à lire et largement personnalisables.

Dans ce projet, j'ai utilisé la version Grafana Cloud. Donc, j'ai économisé les ressources en minimisant le maximum les coûts d'hébergement.



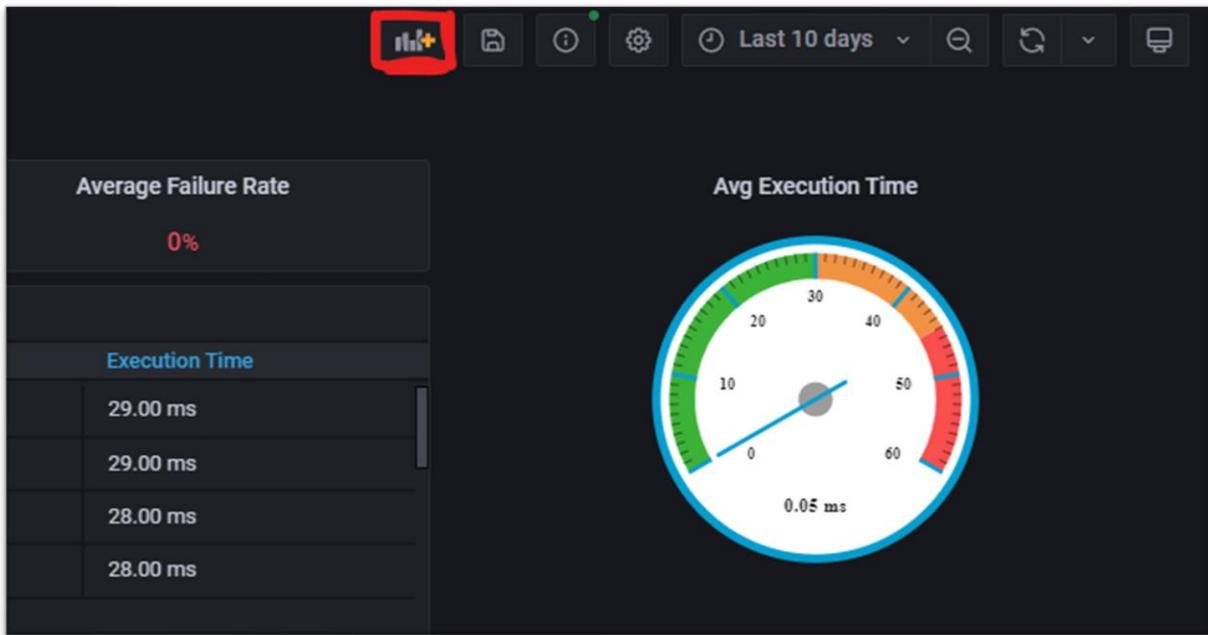
La connexion entre PostgreSQL et Grafana se fait en configurant la partie des data sources :



The screenshot shows the "Data Sources / PostgreSQL" configuration page in Grafana. The "Type" is set to "PostgreSQL". The "Settings" tab is selected. The "Name" field is set to "PostgreSQL". The "Default" checkbox is checked. The "PostgreSQL Connection" section includes fields for "Host" (redacted), "Database" (robot\_results), "User" (lonewolf), "Password" (configured), and "TLS/SSL Mode" (disabled). The "Connection limits" section includes "Max open" (unlimited), "Max idle" (2), and "Max lifetime" (14400). The "PostgreSQL details" section includes "Version" (12+), "TimescaleDB" (disabled), and "Min time interval" (1m).

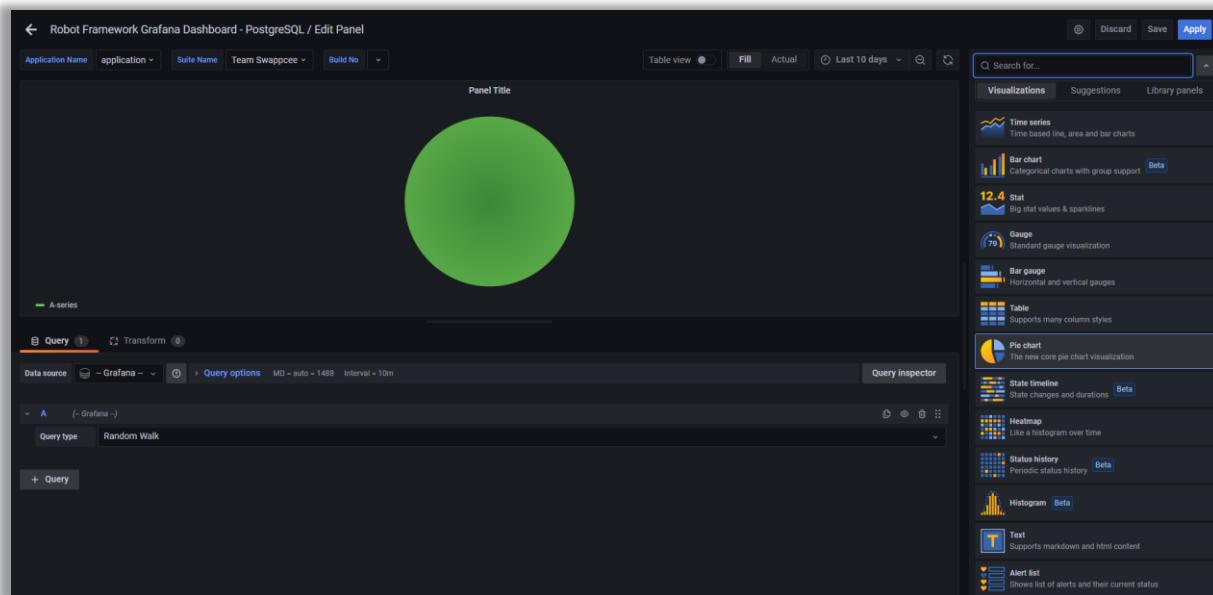
Maintenant, il est temps de créer les Dashboards :

Tout d'abord, il faut cliquer sur ce bouton « Add new panel »



Puis on est présenté par ce menu à droite où on choisit le type de Dashboard qu'on veut.

Ici, j'ai choisi un Dashboard de type Camembert :

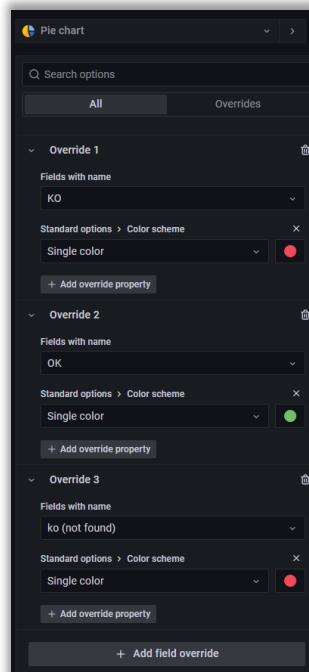


Ensuite, on doit configurer quelques paramètres du camembert comme :

La couleur d'affichage : La couleur verte pour les tests **OK** et la couleur rouge pour les tests **KO**.

Les légendes : affichage des nombre et pourcentages des **OK** et **KO**.

Ci-dessous la configuration complète :



Après, on passe à la configuration des données à afficher :

The screenshot shows the Data Studio Query editor with a PostgreSQL data source selected. A red box highlights the SQL query area:

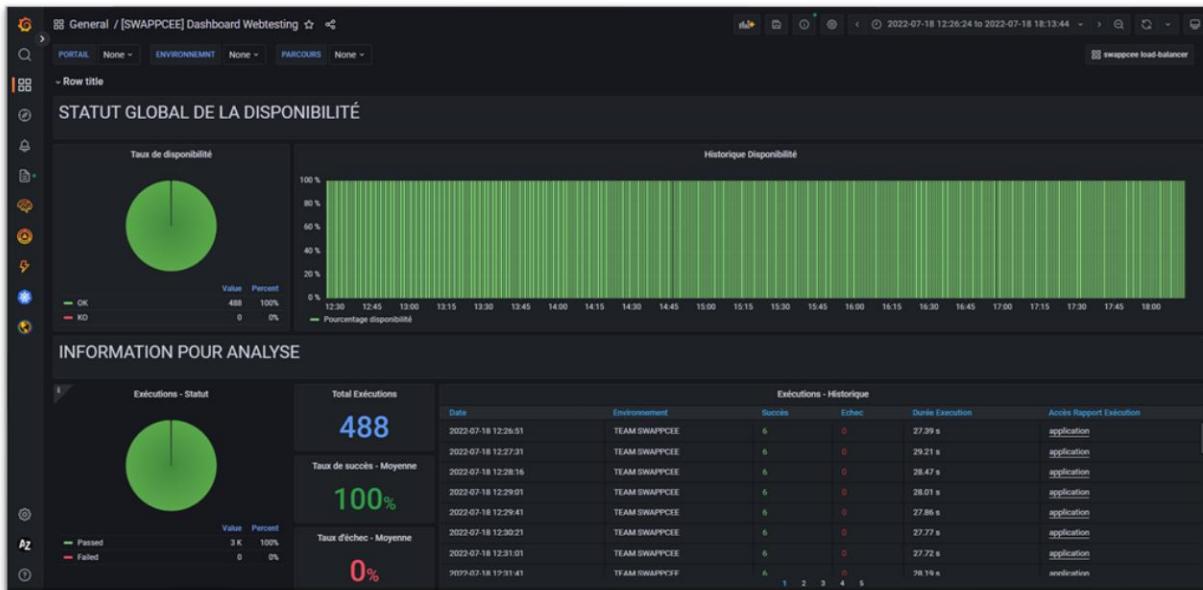
```

SELECT
    floor(extract(epoch from start_time)/43200)*43200 AS "time",
    count( CASE
        WHEN (failed > 0) THEN (1)::bigint
        END ) as "KO",
    (count(failed) - count( CASE
        WHEN (failed > 0) THEN (0)::bigint
        END )) as "OK"
FROM
    automation_execution_results_synthetic
  
```

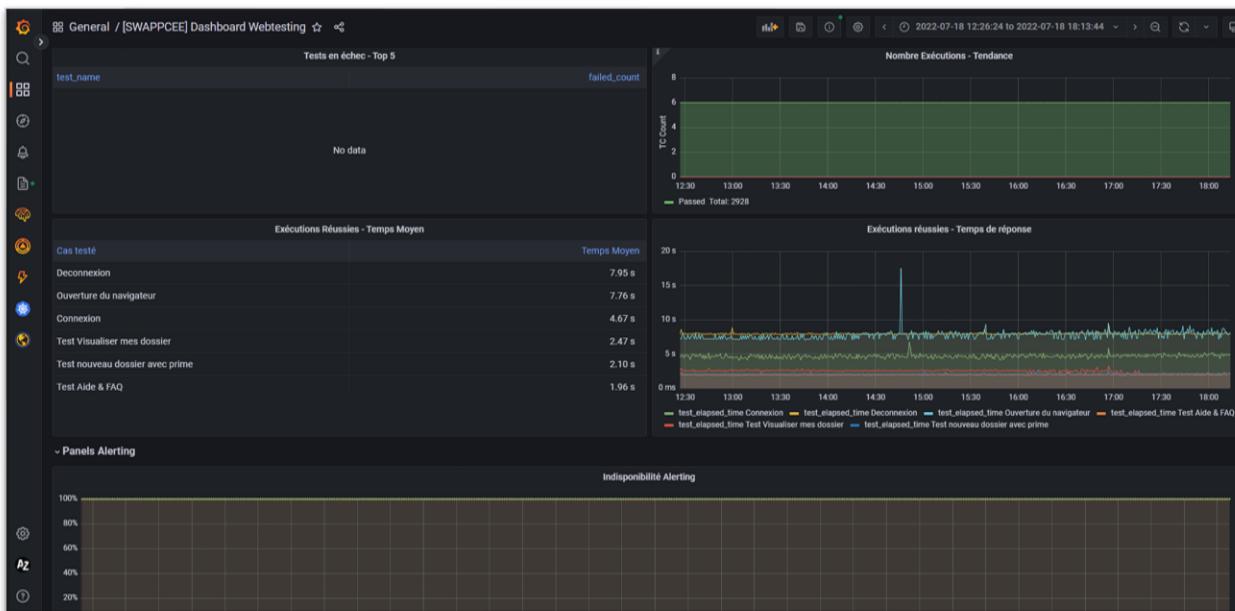
On commence par préciser la source de données configurée auparavant. Puis, on sélectionne le type de Query... Dans ce cas, j'ai sélectionné « requête SQL ». On pourra ajouter plus de Query. Mais ici, je me suis contenter que d'une seule Query A.

En se basant sur cette démarche, j'ai créé et configuré d'autres tableaux de bord :

## Première page



## Deuxième Page



## 5. Intégration de Jenkins :

### a. Configuration de l'environnement de Jenkins :

Jenkins est un serveur d'automatisation gratuit et open source. Il aide à automatiser les parties du développement logiciel liées à la création, aux tests et au déploiement, facilitant l'intégration continue et la livraison continue et la livraison continue. Il s'agit d'un système basé sur un serveur qui s'exécute dans des conteneurs de servlets tels qu'un Apache Tomcat. Il prend en charge les outils de contrôle de version, notamment AccuRev, CVS, Subversion, Git, Mercurial, Perforce, ClearCase et RTC, et peut exécuter des projets Apache Ant, Apache Maven et sbt ainsi que des scripts shells arbitraires et des commandes batch Windows.

Jenkins est disponible dans de nombreuses versions : tout d'abord, vous pouvez opter pour une version hebdomadaire, mise à jour par Jenkins toutes les semaines.

Download Jenkins 2.346.2 LTS for:		Download Jenkins 2.360 for:	
Generic Java package (.war) <small>SHA-256: ef4c4a5dbea6a5c8755ed94fb692ce676086288aadf10cf9aae771fc27b77a</small>			
Docker		Docker	
Ubuntu/Debian		Ubuntu/Debian	
CentOS/Fedora/Red Hat		CentOS/Fedora/Red Hat	
Windows		Windows	
openSUSE		openSUSE	
FreeBSD		FreeBSD	
Gentoo		Gentoo	
macOS		macOS	
OpenBSD		OpenBSD	
OpenIndiana Hipster			

Après connexion, nous avons l'interface suivante :

The screenshot shows the Jenkins dashboard for the 'Monitoring and Alerting Webtesting Project'. The main area displays two active builds:

Nom du projet	Dernier succès	Dernier échec	Dernière durée	Robot Results + Duration Trend
BuildingJob	1 mo. 1 j #13	3 mo. 15 j #2	19 s	
RunningJob	6 j 18 h #12546	s. o.	36 s	

On the left sidebar, under 'Historique des constructions', there are links for 'Nouveau Item', 'Utilisateurs', 'Historique des constructions', 'Éditer cette vue', 'Supprimer cette vue', 'Relations entre les builds', 'Vérifier les empreintes numériques', 'Administre Jenkins', 'Mes vues', 'Ressources Verrouillables', and 'Créer une Vue'.

At the bottom right of the dashboard, it says 'REST API Jenkins 2.332.2'.

Pour démarrer un nouveau projet d'intégration continue, on va créer un nouveau job.

Ensuite, il est nécessaire de donner un nom au projet et de choisir ce qu'on veut atteindre :

- **Freestyle project** : Jenkins associe la gestion des versions à un système de build.
- **Pipeline** : créer un pipeline entre plusieurs agents de build.
- **Projet multi-configuration** : Choisir cette option si on a un projet qui nécessite une variété de paramètres, par exemple parce qu'on utilise plusieurs environnements de test.
- **Organization Folder** : un dossier est un conteneur dans lequel on peut stocker des objets imbriqués.
- **Multibranch Pipeline** : On peut créer directement plusieurs pipelines.

The screenshot shows the Jenkins 'Create New Item' dialog. The title bar says 'Jenkins'. The main area has a heading 'Saisissez un nom' with a text input field containing 'Robot Framework Job'. Below it, there's a note 'Champ obligatoire'. To the right, there are four options with icons and descriptions:

- Construire un projet free-style**: A yellow icon of a person at a desk. Description: Ceci est la fonction principale de Jenkins qui sert à builder (construire) votre projet. Vous pouvez intégrer tous les outils de gestion de version avec tous les systèmes de build. Il est même possible d'utiliser Jenkins pour tout autre chose qu'un build logiciel.
- Pipeline**: A blue icon of a pipeline. Description: Organise des activités de longue durée qui peuvent s'étendre sur plusieurs agents de construction. Adapté pour la création des pipelines (anciennement connues comme workflows) et/ou pour organiser des activités complexes qui ne s'adaptent pas facilement à des tâches de type libre.
- Construire un projet multi-configuration**: A green icon of a gear. Description: Adapté aux projets qui nécessitent un grand nombre de configurations différentes, comme des environnements de test multiples, des binaires spécifiques à une plateforme, etc.
- Dossier**: A grey icon of a folder. Description: Crée un conteneur qui stocke des objets imbriqués. Utile pour grouper ensemble des éléments. Contrairement à une vue qui n'est qu'un filtre, un dossier crée un espace de nommage distinct, de sorte que vous pouvez avoir plusieurs éléments du même nom tant qu'ils se trouvent dans des dossiers différents.
- Organization Folder**: A grey icon of a folder. Description: Creates a set of multibranch project subfolders by scanning for repositories.
- Pipeline Multibranches**: A grey icon of a pipeline. Description: Crée un exemple de projets Pipeline en se basant sur les branches détectées dans le dépôt du gestionnaire de code source.

At the bottom right of the dialog is a blue 'OK' button.

## b. Préparation à l'intégration des tests avec Jenkins :

Dans ce projet, nous allons nous contenter sur les projets freestyle.

The screenshot shows the Jenkins 'BuildingJob' configuration page. The top navigation bar includes 'Tableau de bord', 'BuildingJob', and other Jenkins navigation links. The main configuration area has a 'General' tab selected, followed by tabs for 'Gestion de code source', 'Ce qui déclenche le build', 'Environnements de Build', 'Build', and 'Actions à la suite du build'. The 'General' tab contains a 'Description' section with the text: 'Ce job va importer la repository qui contient tous les fichiers nécessaires pour construire ce projet. Puis, il va créer une image Docker depuis un script Dockerfile.' Below this is a 'Plain text' link and a 'Prévisualisation' link. At the bottom of the configuration area, there are three checkboxes:

- Ce build a des paramètres ?
- GitHub project
- Utiliser un répertoire de travail spécifique ?

A la page suivante, on peut configurer notre projet selon six catégories. Commençons par la gestion du code source. Le programme donné se trouve dans un référentiel GitLab. Il faut indiquer le bon répertoire et branche sous « Git ».

The screenshot shows the 'Gestion de code source' (Code Source Management) section of a Jenkins job configuration. It includes fields for 'Repository URL' (set to https://gitlab.com/azouaghachref/CheqFlushAEM), 'Credentials' (set to 'git /\*/\*/\*'), and 'Branches to build' (set to '\*/\*master'). There are also sections for 'Additional Behaviours' and an 'Avancé...' (Advanced) button.

A l'étape suivante, on va sélectionner ce qui déclenche le Build. Cela déterminera dans quelles situations Jenkins devra effectuer un Build.

Ici, on crée un Webhook qui pointe sur l'URL du serveur Jenkins et qui se déclenche toujours après l'action

The screenshot shows the 'Ce qui déclenche le build' (What triggers the build) section. It includes a 'Enabled GitLab triggers' section with 'Push Events' checked, and other options like 'Push Events in case of branch delete', 'Opened Merge Request Events' (checked), 'Build only if new commits were pushed to Merge Request', 'Accepted Merge Request Events', and 'Closed Merge Request Events'. There is also a 'Rebuild open Merge Requests' dropdown set to 'Never', and a 'Comment (regex) for triggering a build' field containing 'Jenkins please retry a build'. At the bottom, there are checkboxes for 'GitHub hook trigger for GITScm polling' and 'Scrutinisation de l'outil de gestion de version'.

Un Webhook est un outil configurable léger et écrit en Go, qui permet de créer facilement des points de terminaison HTTP (hooks) sur un serveur donné, qu'on peut utiliser pour exécuter des commandes configurées. On peut également transmettre les

données de la requête HTTP (telles que les entêtes, la charge utile ou les variables de requête) à des commandes. webhook permet également de spécifier des règles qui doivent être satisfaites pour que le hook soit déclenché.

Achraf Azouagh > CheckFlushAEM > Integration Settings > Jenkins

Search page

## Jenkins

Run CI/CD pipelines with Jenkins when you push to a repository, or when a merge request is created, updated, or merged. [Learn more](#).

Enable integration  
 Active

**Trigger**

Push  
 Trigger event for pushes to the repository.

Merge request  
 Trigger event when a merge request is created, updated, or merged.

Tag push  
 Trigger event for new tags pushed to the repository.

**Jenkins server URL**  
  
 The URL of the Jenkins server.

**SSL verification**  
 Enable SSL verification  
 Clear if using a self-signed certificate.

**Project name**  
  
 The name of the Jenkins project. Copy the name from the end of the URL to the project.

**Username**  
  
 The username for the Jenkins server.

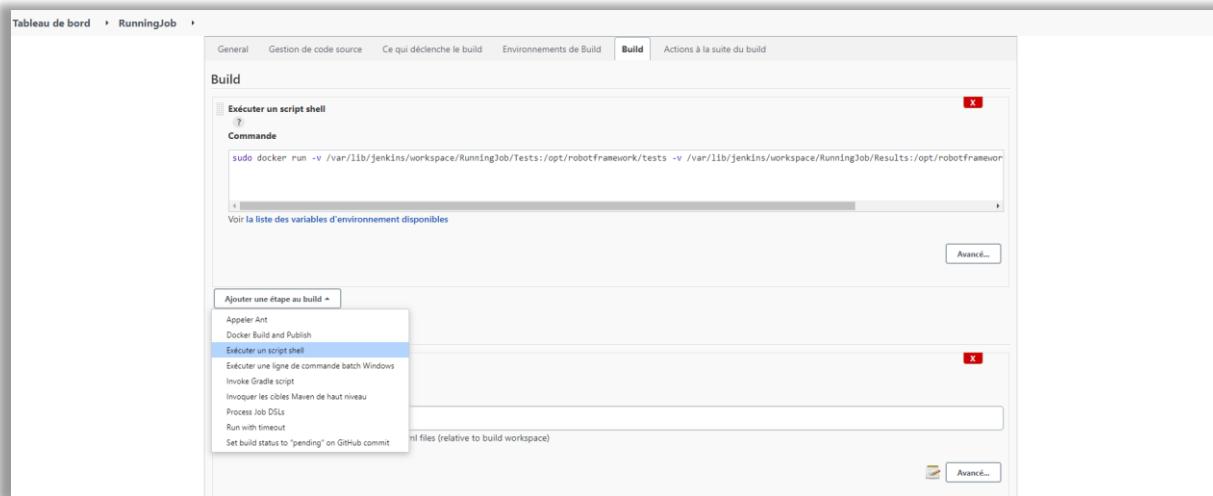
**Enter new password.**  
  
 Leave blank to use your current password.

**Recent events**

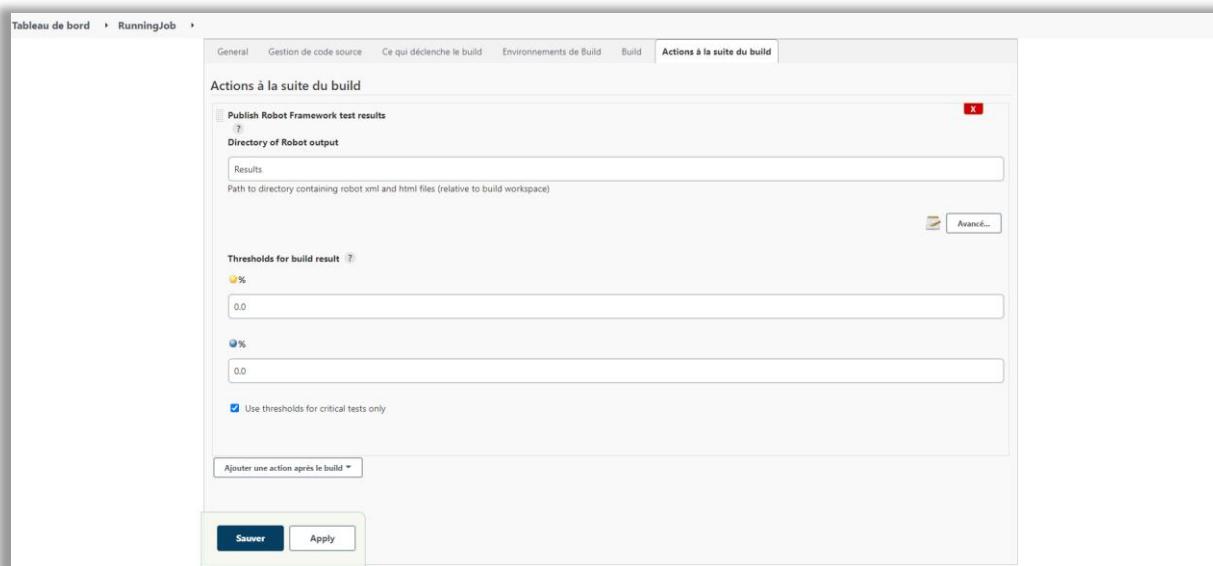
GitLab events trigger webhooks. Use the request details of a webhook to help troubleshoot problems. [How do I troubleshoot?](#)

Status	Trigger	Elapsed time	Request time	
200	Push Hook	0.56 sec	26 minutes ago	<a href="#">View details</a>
200	Push Hook	0.44 sec	28 minutes ago	<a href="#">View details</a>
200	Push Hook	0.83 sec	29 minutes ago	<a href="#">View details</a>

Après la gestion du code source, on passe à la partie Build qui a pour but de définir l'intérêt du job. Dans notre cas, nous allons sélectionner « exécuter un script shell » puis nous allons entrer la commande que nous voulons. Ici, nous basculons vers le deuxième Job « RunningJob » qui va exécuter l'image Docker créée par « BuildingJob ». Donc, la commande sera « Docker run ... » :



Après cela, Jenkins nous donne encore la possibilité d'exécuter des actions postérieures au Build. Les tests sont particulièrement importants ici : connectez Jenkins à Robot Framework avec l'option « Publish Robot Framework test results »



Ci-dessous le résultat attendu de l'intégration du plugin Robot Framework :

The screenshot shows the Jenkins interface for a 'Monitoring and Alerting Webtesting Project' under 'RunningJob'. The 'Robot Results' section is selected. It displays 'Robot Framework Test Results' with the following details:

- Executed:** 20220725 18:19:10.716
- Duration:** 0:00:27.145 (+0:00:00.086)
- Status:** 0 critical test, 0 passed, 0 failed, 0 skipped  
6 test total ( $\pm 0$ ), 6 passed, 0 failed, 0 skipped
- Results:** report.html, log.html, Original result files

Two charts are present:

- Test Result Trend:** A bar chart showing the number of test cases across various build numbers (#12558, #12559, #12563, #12568, #12573, #12578, #12583, #12588, #12593). All bars are green, indicating passed tests.
- Duration Trend:** A line chart showing duration in seconds across the same build numbers. The values remain relatively constant around 25-30 seconds.

The 'Test Suites' section shows:

Name	Failed tests (0)	Total tests (6)	Duration (0:00:27.145)
Team Swappee	0 (0)	6 (6)	0:00:27.145 (+0:00:00.086)

## c. Rapport des tests au niveau de Jenkins :

The screenshot shows the Jenkins terminal output for a 'RunningJob' under 'Monitoring and Alerting Webtesting Project'. The output is as follows:

```

Tableau de bord > Monitoring and Alerting Webtesting Project > RunningJob > #12596
> git --version # `git version 2.20.1`  

using GIT_ASKPASS to set credentials  

> git fetch --tags --prune --progress -- https://gitlab.com/azouaghachraf/CheckFlushAEM +refs/heads/*:refs/remotes/origin/* # timeout=10  

+ git config remote.origin.fetch '+refs/heads/*:refs/remotes/origin/*'  

+ git config core.sparsecheckout # timeout=10  

+ git config core.sparsecheckout # timeout=10  

Commit message: "Deleted Devdata/env.json"  

> git rev-list --no-walk 9f39d7c9346e7d41fe389e9ebcbff710b99f1c6 # timeout=10  

[RunningJob] $ /bin/sh -c /tmp/jenkins1200959459598097613.sh  

+ sudo docker run -v /var/lib/jenkins/workspace/RunningJob/Tests:/opt/robotframework/tests -v /var/lib/jenkins/workspace/RunningJob/Results:/opt/robotframework/reports lonewolf/webtestingimage  

=====
Team Swappee :: TEAM_SWAPPEE
=====
Ouverture du navigateur | PASS |
Conexion | PASS |
Test nouveau dossier avec prime | PASS |
Test Visualiser mes dossier | PASS |
Test Aide & FAQ | PASS |
Deconnexion | PASS |
=====
Team Swappee :: TEAM_SWAPPEE
=====
4 tests, 4 passed, 0 failed
=====
Output: /opt/robotframework/reports/output.xml
Log: /opt/robotframework/reports/log.html
Report: /opt/robotframework/reports/report.html
Parsing: '/opt/robotframework/reports/output.xml'.
Robot results publisher started...
-Parsing output xml:
Done!
-Copying log files to build dir:
Done!
-Assigning results to build:
Done!
-Checking thresholds:
Done!
Done publishing Robot results.
Triggering a new build of RunningJob
Finished: SUCCESS

```

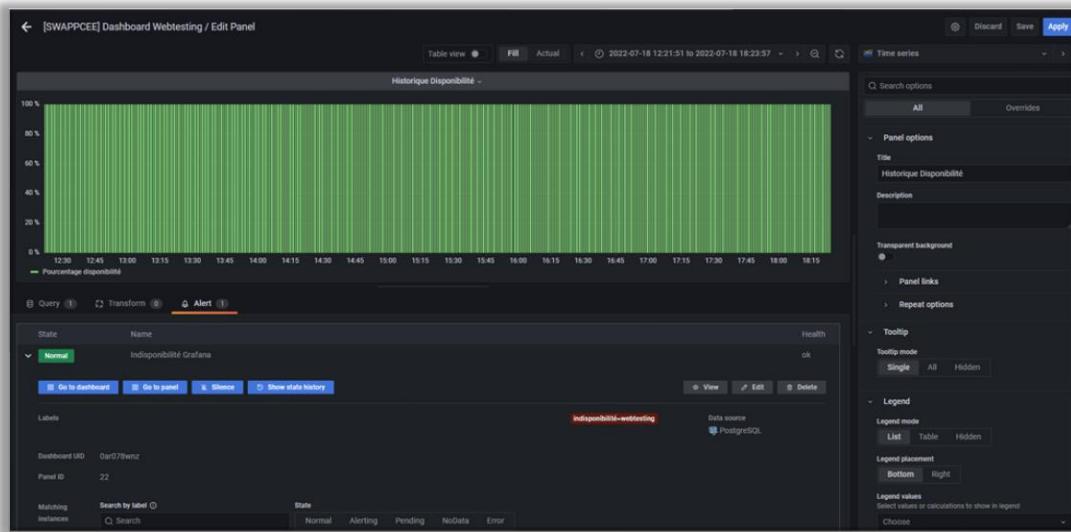
## 6. Alerting :

Une autre fonctionnalité très prisée de Grafana est la possibilité de configurer des alertes, qui sont envoyées quand un événement anticipé est déclenché. Ces alertes

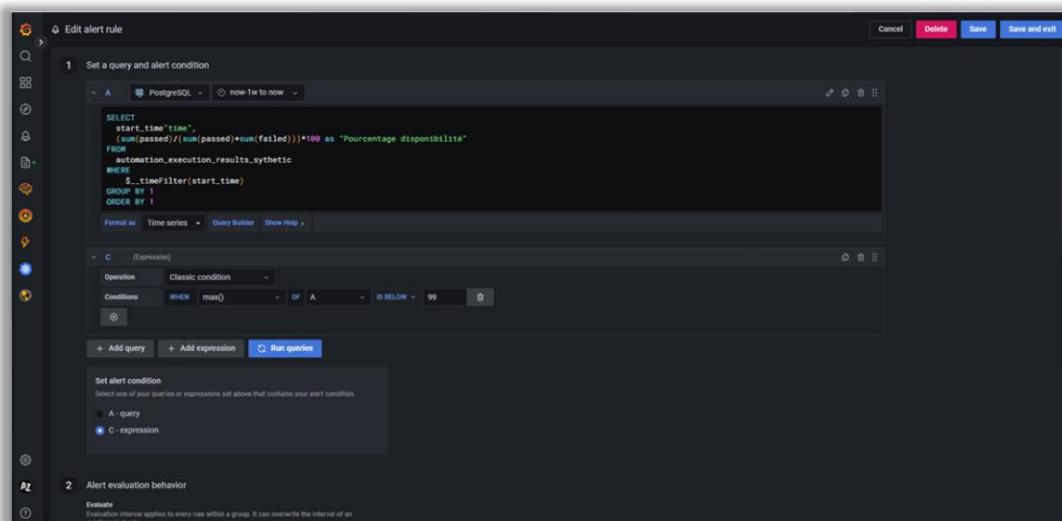
peuvent être envoyées via Slack ou toute autre application de communication utilisée par l'équipe.

## a. Créer une alerte Grafana :

Pour ce faire, j'ai configuré l'alerte sur le tableau de bord principal (Time series).



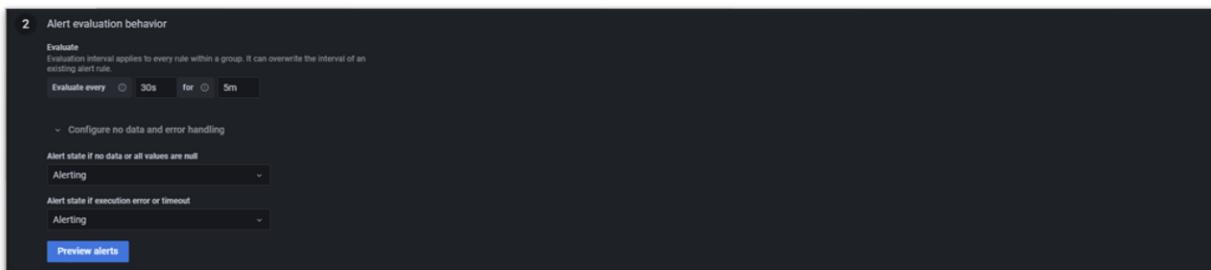
On va vers l'édition du Dashboard, puis on bascule à l'onglet « Alert » et on ajoute une alerte. La configuration se fait comme suit :



On commence tout d'abord en se basant sur la requête SQL faite sur le tableau de bord principal, ensuite on ajoute une autre Query que l'on appelle « Alert Condition ». Ici, on précise quand l'alerte doit-être déclenchée.

Depuis la capture ci-dessus, on voit bien les paramètres utilisés :

- **When** : c'est la fonction qui va vous retourner une valeur. Pour moi, il est plus représentatif de jouer sur max() puisqu'on veut obtenir la valeur maximale de la requête A.
- **Is Below** : permet de désigner une valeur en deçà de laquelle il faut faire attention. Dans le cas présent, nous voulons savoir si la valeur est inférieure à 99.



- **If no data or all values are null** : permet de dire à Grafana : « si tu reçois aucune valeur, ne déclenche pas d'erreur » (c'est le cas quand on lance un site web ou un service pour la première fois).
- **If execution error or timeout** : par contre si la base de données renvoie une erreur, il faut déclencher une alerte.

Maintenant que la condition d'alerte est paramétrée, nous allons y associer un service qui a déjà été configuré.

## b. Associer l'alerte à un service de messagerie :

Nous devons tout d'abord créer un Template :

Puis on doit configurer un point de contact :

The screenshot shows the Grafana Alerting interface. The top navigation bar has tabs for 'Alert rules', 'Contact points' (which is selected), 'Notification policies', 'Silences', 'Alert groups', and 'Admin'. Below the tabs, there's a section for 'Alertmanager' set to 'Grafana'. The main area is titled 'Update contact point' for 'indisponibilite Grafana'. It includes fields for 'Name' (set to 'indisponibilite Grafana'), 'Contact point type' (set to 'Email'), and an 'Address' field containing 'pilotagecasa@gmail.com'. There are sections for 'Optional Email settings' (with 'Single email' checked) and 'Message' (containing '[#PROD][SWAPPCEE] Indisponibilité Alerting alert'). At the bottom, there are buttons for 'Test', 'Duplicate', and 'Delete'.

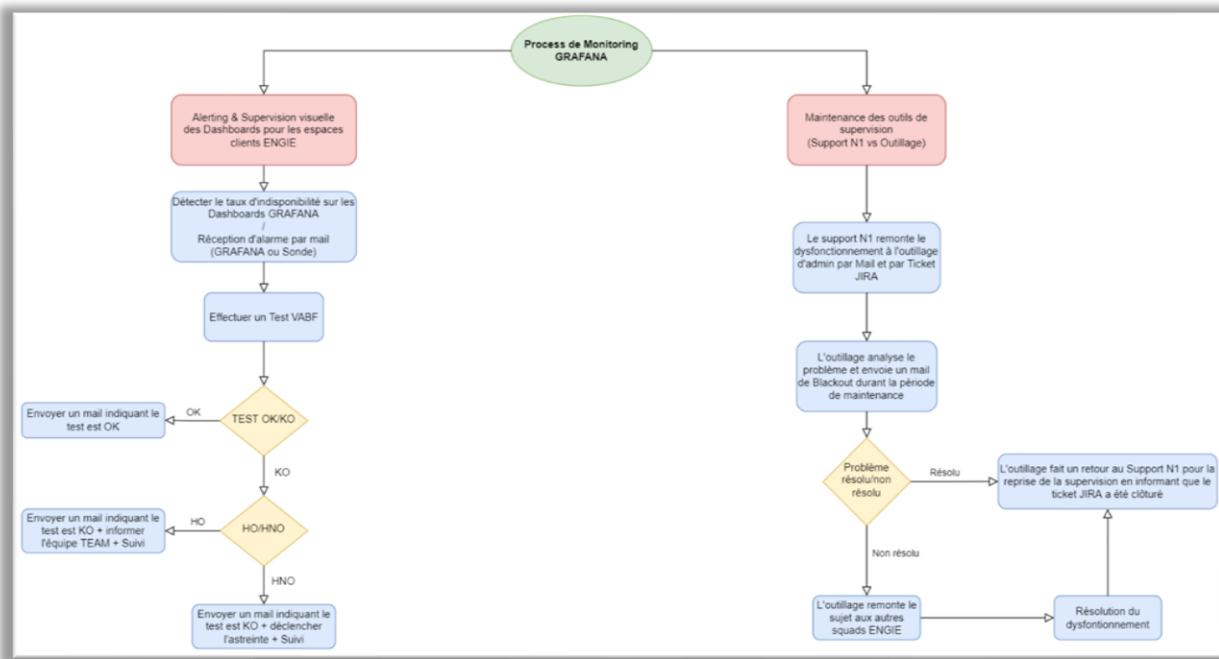
Ci-dessus, on voit bien que le type d'alerte est un mail destiné à l'équipe pilotage 24/7 pour procéder à la gestion de l'incident.

Ci-dessous un exemple de l'alerte envoyé par mail à l'équipe pilotage :

The screenshot shows an email from 'GRAFANA <grafana@[REDACTED]>' to 'DL MA PILOTAGE CASA'. The subject is '[Alerting] [#PROD][SWAPPCEE] Indisponibilité Alerting alert'. The email body starts with '[Alerting] [#PROD][SWAPPCEE] Indisponibilité Alerting alert' and informs the recipient that the availability of the Portal 'SWAPPCEE' (https://swappcee.engine.fr/) is below 100%. It provides a link to the Grafana dashboard for investigation. A table shows the metric 'SWAPPCEE - Pourcentage disponibilité' with a value of 33.333. Below the table, a message states 'No image renderer available/installed' and advises installing the Grafana Image Renderer plugin. At the bottom, there are buttons for 'View your Alert rule' and 'Go to the Alerts page'.

## 7. Gestion de l'incident :

Après la réception de l'alerte d'indisponibilité par l'équipe pilotage 24/7, les pilotes en shift procèdent à effectuer un test VABF manuel (Validation au bon fonctionnement) qui va permettre de confirmer ou bien négliger l'alerte selon le résultat, ci-dessous un workflow qui montre le traitement de l'incident :



## 8. Conclusion :

Dans ce dernier chapitre, on a illustré le déroulement de la réalisation de nos cas de test de l'application, le langage de programmation, présenté les outils DevOps et les outils Monitoring & Alerting.

---

## Conclusion & perspectives

---

Au terme de ces trois mois de stage, nous avons réussi les objectifs que nous nous sommes fixés, à savoir la mise en place d'un système de supervision et alerte à base des solutions d'intégration et de déploiement continues en mode DevOps incluant des outils de webtesting, versioning & monitoring efficaces et bien sélectionnés.

Ce stage m'a été bénéfique sur plusieurs plans. Au niveau fonctionnel et technique, il m'a permis d'une part de raffiner mes capacités d'abstraction et de conception des scripts de webtesting qui a pour but de valider des portails web au bon fonctionnement et le monitoring via l'outil Grafana qui a permis de bien surveiller l'application et de renforcer l'aspect de sécurité qui est un aspect primordial pour chaque application web. D'autre part, il m'a incité à découvrir et manipuler plusieurs outils incontournables pour instaurer l'approche DevOps. Nous avons également franchi le pas sur un nouveau monde : celui du Testing & DevSecOps.

Au niveau relationnel et professionnel, ce stage m'offert d'une part, la chance de travailler avec une équipe d'expertise qui m'a transmis une bonne maîtrise fonctionnelle des enjeux auxquels un ingénieur Cloud & DevOps doit faire face. D'autre part, il m'a offert l'opportunité de m'appliquer dans un projet à grande échelle stratégique mettant le point sur l'importance du monitoring. En plus de cela, mon travail au sein de Capgemini m'a permis de gagner en termes de métrication et formations.

En effet, j'ai réussi à passer plusieurs certifications dans le domaine du DevOps et Cloud à savoir : Microsoft Azure Fundamentals (AZ-900), Microsoft Azure Administrator (AZ-104) & Microsoft Azure Developer (AZ-204). Cela m'a aidé à monter en compétences dans un domaine très courtisé actuellement tel que le DevOps. Quant aux perspectives de ce projet, on envisage de configurer le webtesting sur plusieurs environnements et non seulement la production.

Je remercie aussi Capgemini et les personnes qui ont œuvré en son sein pour l'accomplissement de cette possibilité ; également l'ensemble des enseignants, personnel et acteurs de l'Ecole Supérieur de Technologie de Casablanca pour m'avoir permis, à travers l'ensemble de ma formation, d'aboutir à ce projet.