

Résumé

Afin de réduire le délai de commercialisation et produire des logiciels de qualité, Mobelite choisit d'adopter les pratiques DevOps dans les projets de ses clients.

Dans ce contexte, ce projet vise à améliorer et optimiser une solution hybride qui combiner les ressources locales de l'entreprise avec les ressources du cloud pour créer des pipelines de livraison continue qui assurent la gestion des changements, depuis le code source jusqu'à la mise en production. Il intègre l'outil Ansible pour la gestion de la configuration et l'automatisation du déploiement.

Aussi, il insère un système de recovery et de failover à base de Kubernetes et il utilise des services AWS comme Elastic Container Registry pour le repository. Toutes les images Docker sont déployées dans des conteneurs.

Après avoir effectué une étude comparative des solutions existantes, identifié les besoins et conçu l'architecture du projet, il est nécessaire de mettre en place un pipeline de livraison continue optimisé.

Ce projet a permis de minimiser le temps de déploiement dans les différents environnements et d'avoir une résilience contre les pannes d'infrastructures.

Mots clés : DevOps, Kubernetes, AWS, Ansible, Pipeline, Monitoring, cloud hybride.

Abstract

To reduce time-to-market and produce quality software, Mobelite has chosen to adopt DevOps practices in its clients projects.

In this context, this project aims to improve and optimize a Hybrid cloud solution that combine the local ressources and the cloud for continuous delivery pipeline that handles changes from source code to production. It integrates the Ansible tool for configuration management and deployment automation. Additionally, it incorporates a recovery and failover system based on Kubernetes and utilizes AWS services like Elastic Container Registry for storing all Docker images in containers.

After conducting a comparative study of existing solutions, identifying the needs, and designing the project's architecture, it became necessary to implement an optimized continuous delivery pipeline.

This project has minimized deployment time in different environments and provided resilience against infrastructure failures.

Keywords : DevOps, Kubernetes, AWS, Ansible, Pipeline, Monitoring, Hybrid cloud.

Liste des abréviations

AWS Amazon Web Services

VPC Virtual Private Cloud

EC2 Elastic Compute Cloud

IAM Identity and Access Management

EKS Elastic Kubernetes Service

ECR Amazon Elastic Container Registry

ELB Elastic Load Balancing

ALB Application Load Balancer

CI/CD Continuous integration and continuous deployment

K8s Kubernetes

DevOps Développement logiciel (dev) et de l'administration des infrastructures informatiques (ops)

Table des matières

Introduction générale	0
1 Cadre général du projet et étude de l'existant	2
1.1 Société d'accueil	2
1.2 Problématique	3
1.3 Notions de base	3
1.3.1 Microservice	4
1.3.2 DevOps	4
1.3.3 Cloud	5
1.3.3.1 Modèles de services	5
1.3.3.2 Les modèles de déploiement	6
1.4 Méthodologie de développement adoptée	7
1.4.1 Méthode agile	7
1.4.2 Présentation de la méthodologie Scrum	8
1.4.3 Acteurs de la méthode agile Scrum	8
1.4.4 Évènements de la méthode agile Scrum	8
1.4.5 Les artefacts	9
1.5 Planification des sprints	9
1.6 Les solutions existantes	10
1.6.1 Azure Devops	10
1.6.2 Amazon Web Service DevOps	11
1.6.3 Google Cloud Platform DevOps	12
1.7 Critique des solutions existantes	13
1.8 Solution proposée	14
2 Spécification et analyse des besoins	16
2.1 Acteurs	16
2.2 Besoins fonctionnels	17
2.3 Besoins non-fonctionnels	17
2.4 Product Backlog	18
2.5 Diagramme de cas d'utilisation Global	19
2.5.1 Cas d'utilisation "Gérer code source"	21
2.5.2 Cas d'utilisation "Suivi de la build d'application"	22
2.5.3 Cas d'utilisation "Gérer la configuration de déploiement"	23
2.5.3.1 Cas utilisation "Gérer liste des images Docker"	24
2.5.4 Cas utilisation "Gérer les artefacts d'application"	25
3 Conception	27
3.1 Conception du système	27
3.1.1 Diagramme de séquence global	27
3.1.2 Diagramme de séquence Détailée	29

3.1.2.1	Diagramme de séquence « Configuration de l'architecture cloud »	29
3.1.2.2	Diagramme de séquence « Configuration de cluster local »	29
3.1.2.3	Diagramme de séquence « Configuration de pipeline »	30
3.1.3	Diagramme de classe	31
3.1.4	Diagramme de déploiement	32
3.2	Méthodologie de conception	33
3.3	Architecture technologique de la solution	35
3.4	Conclusion	40
4	Sprint2 : Configuration de la Pipeline.	41
4.1	Objectif du Sprint	42
4.2	Architecture Détailée	42
4.2.1	Diagramme Cas d'utilisation Détailée	42
4.2.2	Diagramme de sequance Détailée	44
5	Sprint3 : Déploiement de l'application sur le cluster.	45
5.1	Objectif du Sprint	46
5.2	Architecture Détailée	46
5.2.1	Diagramme Cas d'utilisation Détailée	46

Table des figures

1.1	différents domaines de mobelite	3
1.2	Architecture monolithique vs Micro services	4
1.3	Cycle de vie DevOps	5
1.4	Différents architecture entre SaaS ,PaaS et IaaS	6
1.5	Différents type du cloud	7
1.6	Processus de méthode Scrum	9
1.7	Diagramme de Gantt	10
1.8	Azure DevOps	11
1.9	Les Services AWS utilisé pour DevOps.	12
1.10	Interface de GCP Pipeline.	13
2.1	Diagramme de cas d'utilisation Globale	20
2.2	Cas d'utilisation :Gérer code source	21
2.3	Cas d'utilisation :Suivi la build d'application	22
2.4	Cas d'utilisation :Gérer la configuration de déploiement	23
2.5	Cas d'utilisation :Gérer liste des images Docker	24
2.6	Cas d'utilisation :Gérer les artefacts d'application	25
3.1	Diagramme de séquence Global	28
3.2	Diagramme de séquence « Configuration de l'architecture cloud »	29
3.3	Diagramme de séquence « Configuration de cluster local »	30
3.4	Diagramme de séquence « Configuration de pipeline »	31
3.5	Diagramme de classe global	32
3.6	Diagramme de déploiement global	33
3.7	Diagramme de déploiement global	34
3.8	Architecture global	36
3.9	Partie local	37
3.10	Partie AWS	39
4.1	cas d'utilisation "Configuration de la Pipline"	43
4.2	diagramme de séquence "Configuration de la Pipline"	44
5.1	cas d'utilisation "Déploiement de l'application sur le cluster."	47

Liste des tableaux

1.1 Tableau comparatif	14
2.1 Tableau du Product Back-log	19
2.2 Description de cas d'utilisation :Gérer code source	21
2.3 Description de cas d'utilisation :Suivi du build de l'application	22
2.4 Description de cas d'utilisation :Gérer la configuration de déploiement	23
2.5 Description de cas d'utilisation :Gérer liste des images Docker	24
2.6 Description de cas d'utilisation :Gérer les artefacts d'application	25
4.1 Description de cas d'utilisation	43
5.1 Description de cas d'utilisation	47

Introduction générale

La montée en puissance de l'entreprise numérique nécessite une profonde révision des méthodes de développement d'applications. Il n'est plus viable d'attendre six mois pour obtenir les livrables de développement. Avec les exigences de rapidité sur le marché et l'évolution des projets dans des configurations logicielles et d'infrastructure de plus en plus complexes, qui entraînent des risques opérationnels et de planification, il est devenu essentiel d'industrialiser les tests et de simplifier les déploiements en production. En rapprochant les équipes de développement, de test et d'exploitation, le DevOps répond précisément à ce défi numérique. Les entreprises en sont maintenant pleinement conscientes.

Parmi les pratiques couramment utilisées en DevOps, on retrouve la livraison continue et la gestion de la configuration. La livraison continue est une approche logicielle qui permet aux organisations de fournir rapidement et efficacement de nouvelles fonctionnalités aux utilisateurs. L'idée fondamentale de la livraison continue est de créer un processus reproductible et fiable d'amélioration progressive pour faire passer le logiciel du concept au client. L'objectif de la livraison continue est de permettre un flux constant de changements vers la production en utilisant un pipeline automatisé de logiciels. C'est ce pipeline de livraison continue qui rend tout cela possible.

Ce projet s'inscrit dans cette perspective en cherchant à créer un espace partagé entre une infrastructure Cloud et un Datacenter local de livraison continue en intégrant Ansible ,Kubernetes et des services AWS pour créer des conteneurs pour le monitoring. Mobelite a entrepris ce projet afin de résoudre plusieurs de ses problématiques.

Ce rapport présente les principales réalisations effectuées au cours de ce projet et est structuré en quatre chapitres :

Dans le premier chapitre,nous présentons l'entreprise d'accueil Mobelite , après nous explorons la problématique de notre sujet.Puis nous présentons les notions de base comme microservices ,DevOps et cloud. Ensuite, nous allons définir la méthodologie de développement utilisée .Enfin, nous allons présenter la planifications des sprints.

Dans le deuxième chapitre, nous présentons une étude comparative des solutions existantes les plus répandues sur le marché et les plus adéquate, ainsi que l'outil choisi dans chaque solution.

Dans le troisième chapitre, analyse et conception, nous exposerons les besoins

fonctionnels et non fonctionnels, les acteurs, les cas d'utilisations et l'architecture physique globale.

Le dernier chapitre est consacré pour présenter les tâches réalisées pour implémenter notre projet.

Chapitre 1

Cadre général du projet et étude de l'existant

Introduction

Ce chapitre est consacré à la présentation du cadre général de notre projet. En premier lieu, nous présentons la société d'accueil. Par la suite, nous présentons le cadre du projet qui explique la problématique . Après, on définit par une présentation de la méthodologie que nous allons adopter pour le développement de notre projet. On finit par la présentation des solutions existantes sur le marché avec une étude comparative de ces dernières et présentation de la solution proposée.

1.1 Société d'accueil

Mobelite est une agence spécialisée dans le conseil en stratégie mobile, la conception et le développement d'applications mobiles , de sites web et le marketing mobile. Mobelite est forte d'une équipe d'experts dans la réalisation d'applications mobiles sur les plateformes les plus répandues et les applications Web. Mobelite dispose d'équipes commerciales et marketing à Paris (France), et d'équipes de développement à Tunis et Monastir (Tunisie).

Ainsi, Mobelite est une société experte dans la création des sites web intuitifs et ergonomiques pour tous les supports soit desktop, tablette ou mobile et avec les plus récentes technologies et Framework de développement comme React JS, Node JS et Angular. Les équipes de mobelite maîtrisent parfaitement la conception et le développement d'applications natives iOS et Android pour tous les différents supports que ce soit smartphone, tablette, Watch et TV. Mobelite excelle dans le conseil de ses clients dans différentes parties de projets comme l'analyse des besoins, UX/UI, la conception, le design, le développement, SEO, DevOps et l'hébergement. Tout ça est réalisé selon la méthodologie Agile, afin de maximiser les possibilités d'itérations sur le concept, et d'introduire plus de flexibilité sur les arbitrages fonctionnels à faire(voir figure 1.2).

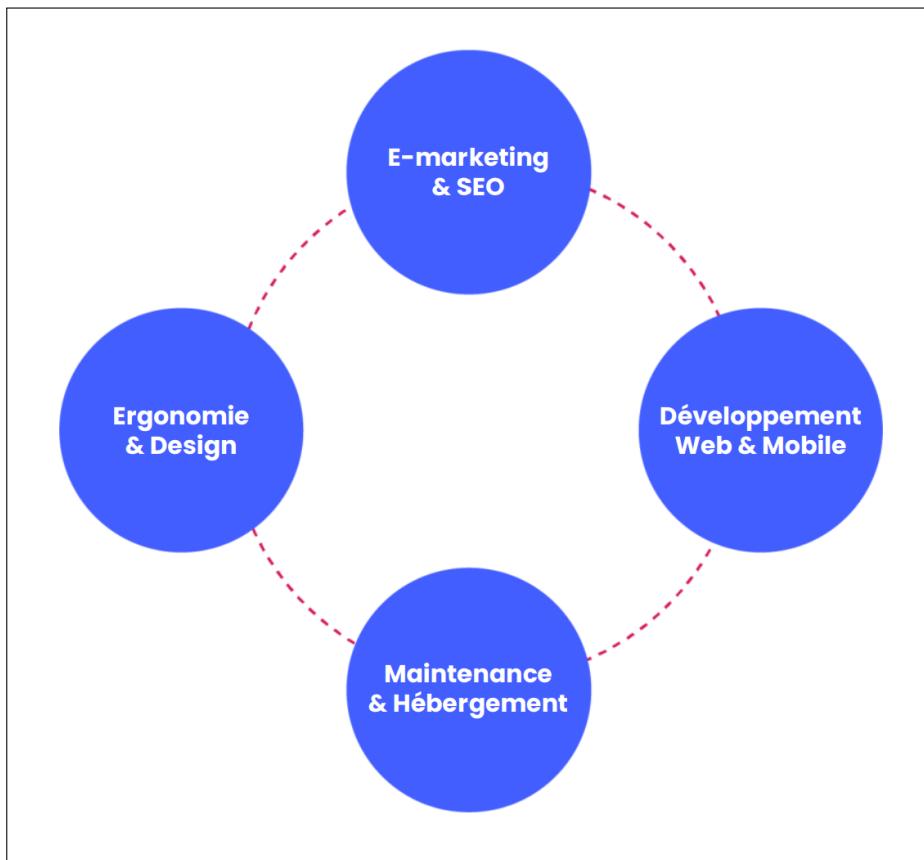


FIGURE 1.1 – différents domaines de mobelite

1.2 Problématique

Le déploiement des systèmes de gestion de code source tells que Nexus, SonarQube ou le Monitoring sur un seul serveur local présente plusieurs problèmes qui affectent la performance de serveur et qui rendent l'exécution des différentes applications ou l'ajout des nouvelles fonctionnalités plus difficile . Aussi, la maintenance de plusieurs applications en même temps sera compliquée et nécessite une planification minutieuse pour éviter l'interruption des processus d'autres applications. Les besoins d'entreprise changent au cours des temps et la capacité d'un seul serveur sera insuffisante pour rependre à la charge des données. Les mise à jour des applications peuvent impacter d'autres applications à cause de limites des ressources. En terme de sécurité, une attaque sur le serveur cause une perte des données très grande que sera difficile de récupérer en conséquence de l'utilisation d'un seul serveur. C'est dans ce cadre que la société souhaite créer son propre solution .

1.3 Notions de base

Dans ce qui suit,nous présentons les notions de base que nous utiliserons pour réaliser le projet.

1.3.1 Microservice

Avant l'apparition de l'architecture micro-service, les applications sont construites de manière monolithique, qui est considérée aujourd'hui comme méthode impertinente. L'utilisation de l'architecture monolithique, rend l'application très volumineuse avec des difficultés à enrichir les fonctions et le traitement des problèmes. Les microservices créent une application unique à partir de plusieurs petits services reliés de manière flexible. Ces services ont leur propre logique et leur propre base de données pour un usage spécifique. Chaque service peut être développé, mis à jour, déployé et mis à l'échelle sans affecter les autres services. Les mises à jour logicielles peuvent être plus fréquentes, ce qui améliore la fiabilité, la disponibilité et les performances(Voir figure 1.3).

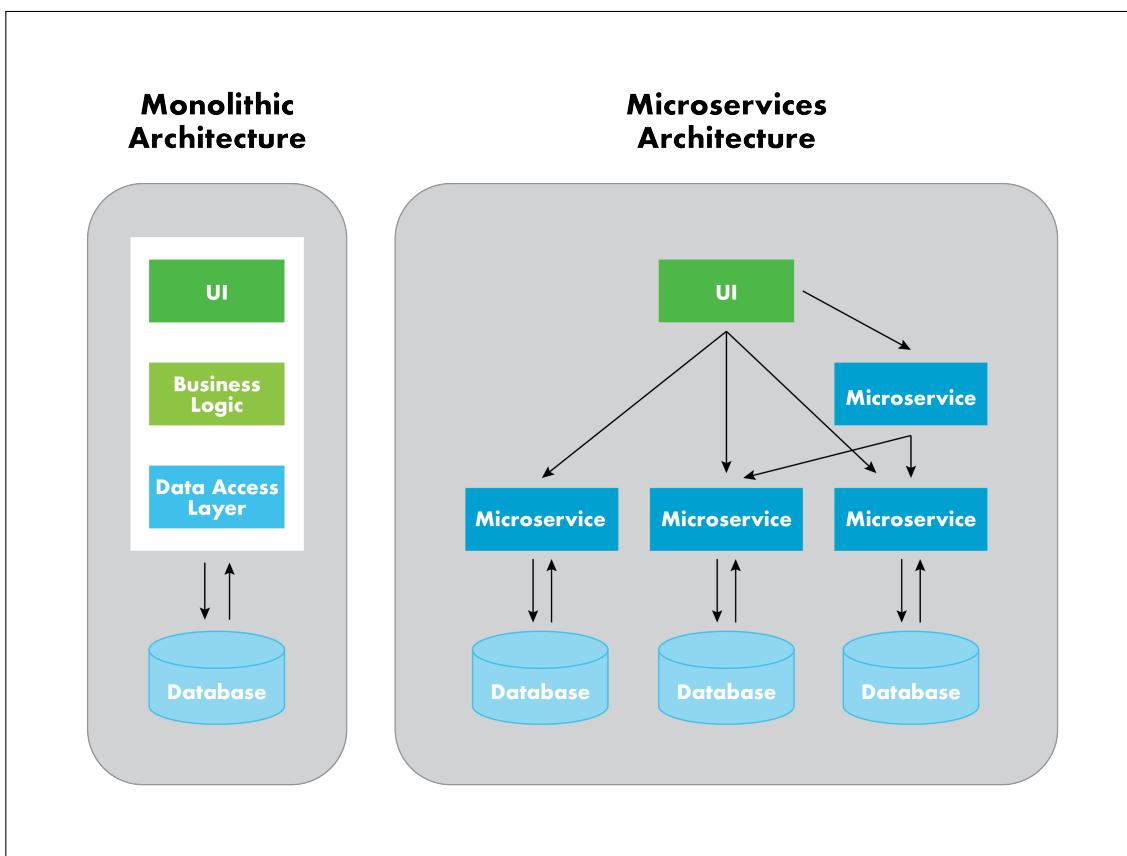


FIGURE 1.2 – Architecture monolithique vs Micro services

1.3.2 DevOps

Combinant développement (Dev) et opérations (Ops), DevOps est l'union des personnes, des processus et des technologies destinés à fournir continuellement de la valeur aux clients. DevOps permet la coordination et la collaboration des rôles autrefois cloisonnés (développement, opérations informatiques, ingénierie qualité et sécurité) pour créer des produits plus performantes et plus fiables. En adoptant une culture DevOps ,ainsi que des pratiques et outils DevOps, les équipes peuvent mieux répondre aux besoins des clients, accroître la confiance suscitée par les

applications qu'elles développent, et atteindre plus rapidement les objectifs de leur entreprise [1] (voir figure 1.4).

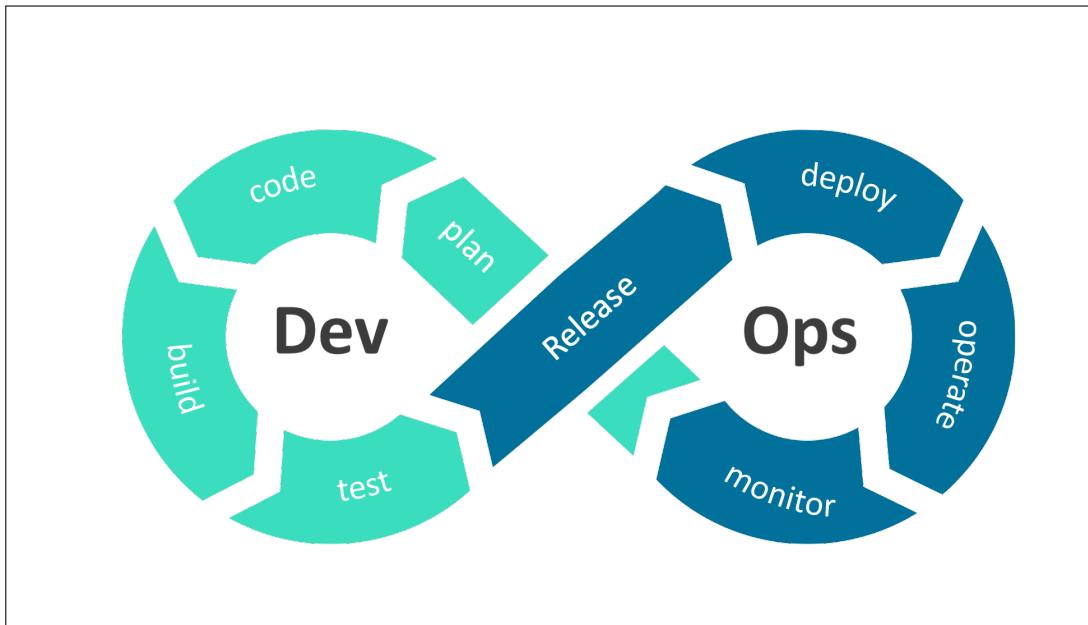


FIGURE 1.3 – Cycle de vie DevOps

1.3.3 Cloud

Le cloud n'est pas une entité physique, mais un vaste réseau de serveurs distants éparpillés tout autour de la planète, reliés entre eux, et destinés à fonctionner comme un écosystème unique. Ces serveurs sont conçus pour stocker et gérer des données, exécuter des applications, ou fournir du contenu ou des services (vidéos diffusées en continu, courrier web, logiciels bureautiques de productivité et autres réseaux sociaux). Au lieu d'accéder à des fichiers et données stockées sur un ordinateur local ou personnel, vous accédez à ces ressources en ligne à partir de n'importe quel appareil compatible avec Internet : les informations sont disponibles en tout lieu et tout le temps.[2]

1.3.3.1 Modèles de services

Il existe 3 modèles de services sur le cloud :

-Software as a Service (SaaS) : Le Software as a Service, également connu sous le nom de SaaS, est un service basé sur le cloud où, au lieu de télécharger un logiciel que votre PC de bureau ou votre réseau professionnel peut exécuter et mettre à jour, vous accédez à une application via un navigateur internet. L'application logicielle peut être un logiciel de bureautique ou de communication unifiée parmi un large éventail d'autres applications professionnelles disponibles[3] (voir figure 1.4).

-Platform as a Service (PaaS) : La Platform-as-a-service (PaaS) est un type d'offre de cloud computing dans lequel un fournisseur de services fournit une plateforme à ses clients, leur permettant de développer, d'exécuter et de gérer des applications commerciales sans avoir à construire et à maintenir l'infrastructure que ces processus de développement de logiciels requièrent généralement(Voir figure 1.4)[4].

-Infrastructure as a Service (IaaS) : Infrastructure as a service (IaaS) est un type de service de cloud computing qui offre des ressources de calcul, de stockage et de mise en réseau essentielles à la demande, et sur une base de paiement à l'utilisation(Voir figure 1.4)[5].

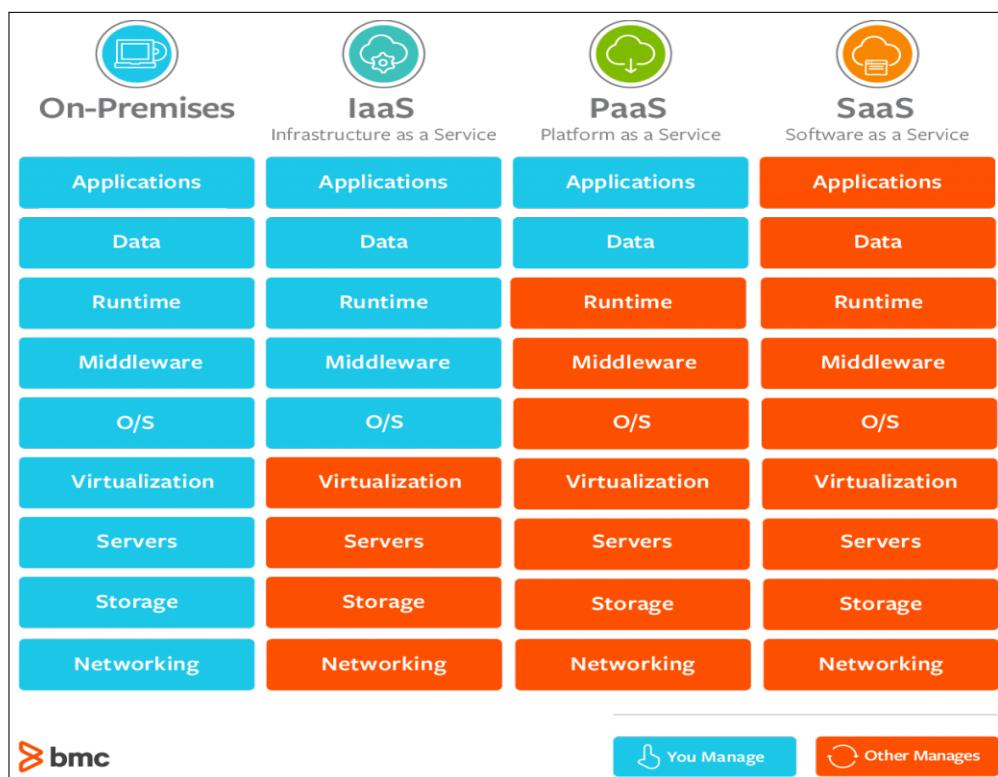


FIGURE 1.4 – Différents architecture entre SaaS ,PaaS et IaaS

1.3.3.2 Les modèles de déploiement

Il existe 3 modèles de déploiement sur le cloud :

-Cloud Privé : Le cloud privé est un modèle informatique qui offre un environnement propriétaire dédié à une seule entité commerciale. Comme les autres types d'environnements du cloud computing, le cloud privé fournit des ressources informatiques étendues et virtualisées via des composants physiques stockés sur place ou dans le centre de données d'un fournisseur[6].

-Cloud Public : Le cloud public est un type de calcul dans lequel les ressources sont proposées par un fournisseur tiers via Internet, et sont partagées par les or-

ganisations et les individus qui souhaitent les utiliser ou les acheter [7].

-Cloud Hybride : Un cloud hybride est un environnement informatique mixte dans lequel des applications s'exécutent à l'aide d'une combinaison de ressources de calcul, de stockage et de services dans différents environnements (clouds publics et clouds privés, y compris des centres de données sur site ou en périphérie)[8](voir figure 1.5).

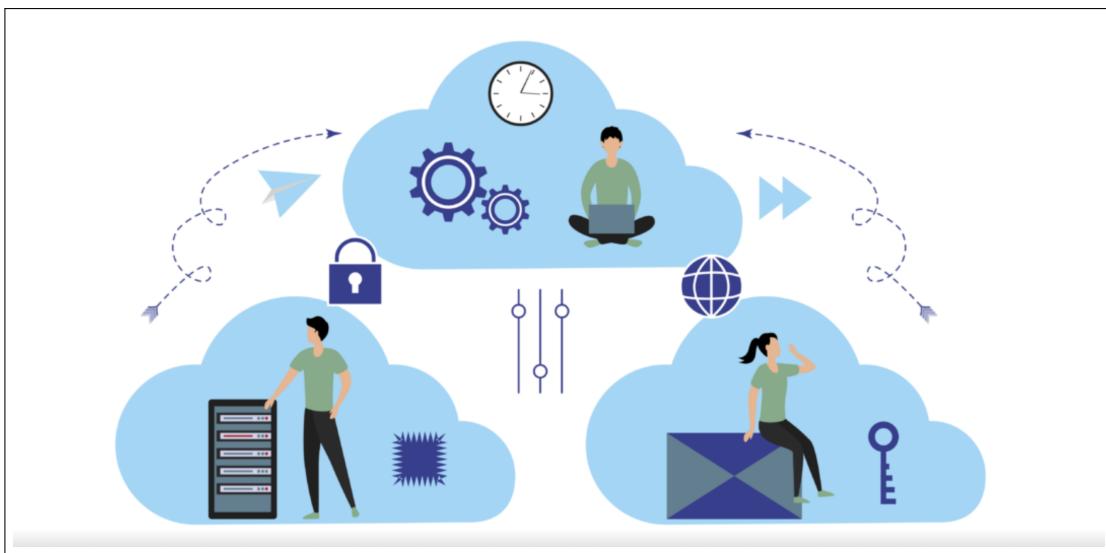


FIGURE 1.5 – Différents type du cloud

1.4 Méthodologie de développement adoptée

Avant de réaliser un projet informatique, il est essentiel de sélectionner une démarche de travail et une procédure de suivi pour obtenir un logiciel stable. Il s'agit d'un cadre permettant de structurer, de planifier et de contrôler le développement d'une application.

Pour la réalisation de notre projet nous avons opté pour la méthodologie agile SCRUM car elle améliorer la flexibilité de projet et réduit le temps de livraison de produits complet au client. En effet , la plupart des projets réalisés dans l'entreprise appliquent cette méthodologie.

1.4.1 Méthode agile

Agile est une méthode de gestion de projet conçue pour distribuer en continu les logiciels opérationnels en fonction d'itérations rapides. Il permet aux équipes de développer progressivement un produit de qualité et d'adapter leur processus en fonction des besoins du projet.

1.4.2 Présentation de la méthodologie Scrum

Scrum est une méthodologie agile pour l'élaboration, la réalisation et le soutien de projets complexes. Elle est basée sur la division du produit sur différentes itération sprints. Scrum est plus utile lorsque les exigences sont variables et peuvent changer beaucoup de fois au cours du cycle de vie, ce qui donne donc la capacité d'avoir un projet flexible capable de traiter les changements fréquents.

1.4.3 Acteurs de la méthode agile Scrum

Méthode agile Scrum se compose par 3 acteurs ce suit :

-Scrum Master : Il s'agit de la personne chargée d'orienter l'équipe de travail vers la bonne voie, d'assurer la bonne pratique des règles de la méthode Scrum et d'organiser son équipe.

-Product Owner : il représente les clients et assure que leurs besoins et visions soient réalisés dans le projet. Il travaille généralement en collaboration directe avec l'équipe de développement.

-Equipe de développement : sont les personnes responsables de la transformation des besoins du client. Ces besoins sont définis par le Product Owner à travers des fonctions utilisables. L'équipe est composée d'au moins 3 personnes jouant les rôles de développeur ou de concepteur.

1.4.4 Évènements de la méthode agile Scrum

Scrum définit cinq types d'évènements :

-Le sprint : Chaque sprint est de durée maximale de 4 semaines pendant laquelle une version de produit est réalisée. Une fois un sprint terminé un nouveau a déjà commencé avec une liste de fonctionnalités et un objectif à réaliser.

-Planification d'un sprint : À chaque début de sprint, cette réunion est conçue pour déterminer les tâches à réaliser pendant le sprint. L'organisation de ces tâches est effectuée par l'équipe de développement et le Product Owner.

-Mêlée quotidienne : C'est une réunion d'une durée de 15 minutes faite par l'équipe chaque jour pour définir l'objectif de la journée et identifier les obstacles s'il y en a quelques-uns.

-Revue de sprint : Représente le travail réalisé par l'équipe au cours du sprint et le comparer avec le produit attendu.

-Rétrospective de sprint : Le but de cet événement est de déterminer les problèmes intervenus dans la période du sprint, l'efficacité des outils et déterminer ce qui peut être amélioré.

1.4.5 Les artefacts

-Product Backlog : Il s'agit d'une liste des besoins et exigences à recueillir pour créer le produit désiré, ce document est la responsabilité de Product Owner(voir figure 1.6).

-Sprint Backlog : C'est l'ensemble des données permettant la réalisation des objectifs du sprint. Ce document est mis à jour par l'équipe de développement régulièrement.

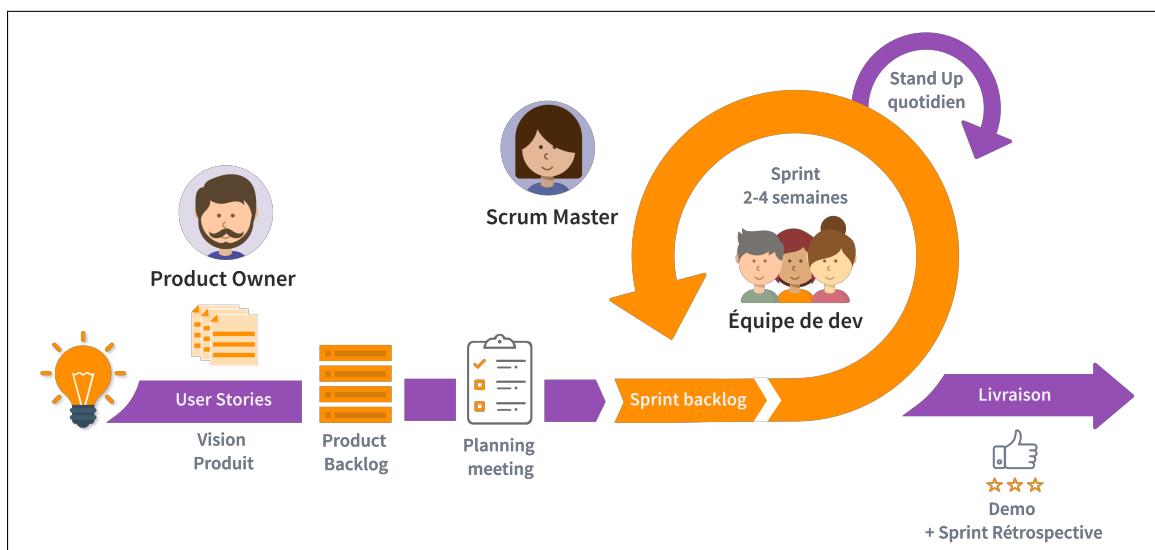


FIGURE 1.6 – Processus de méthode Scrum

1.5 Planification des sprints

Pour une meilleure optimisation du développement du projet nous avons divisé le travail en des Sprints présentés dans un diagramme de Gantt qui décrit l'état d'avancement dans le temps des différentes activités (voir figure 1.7) :

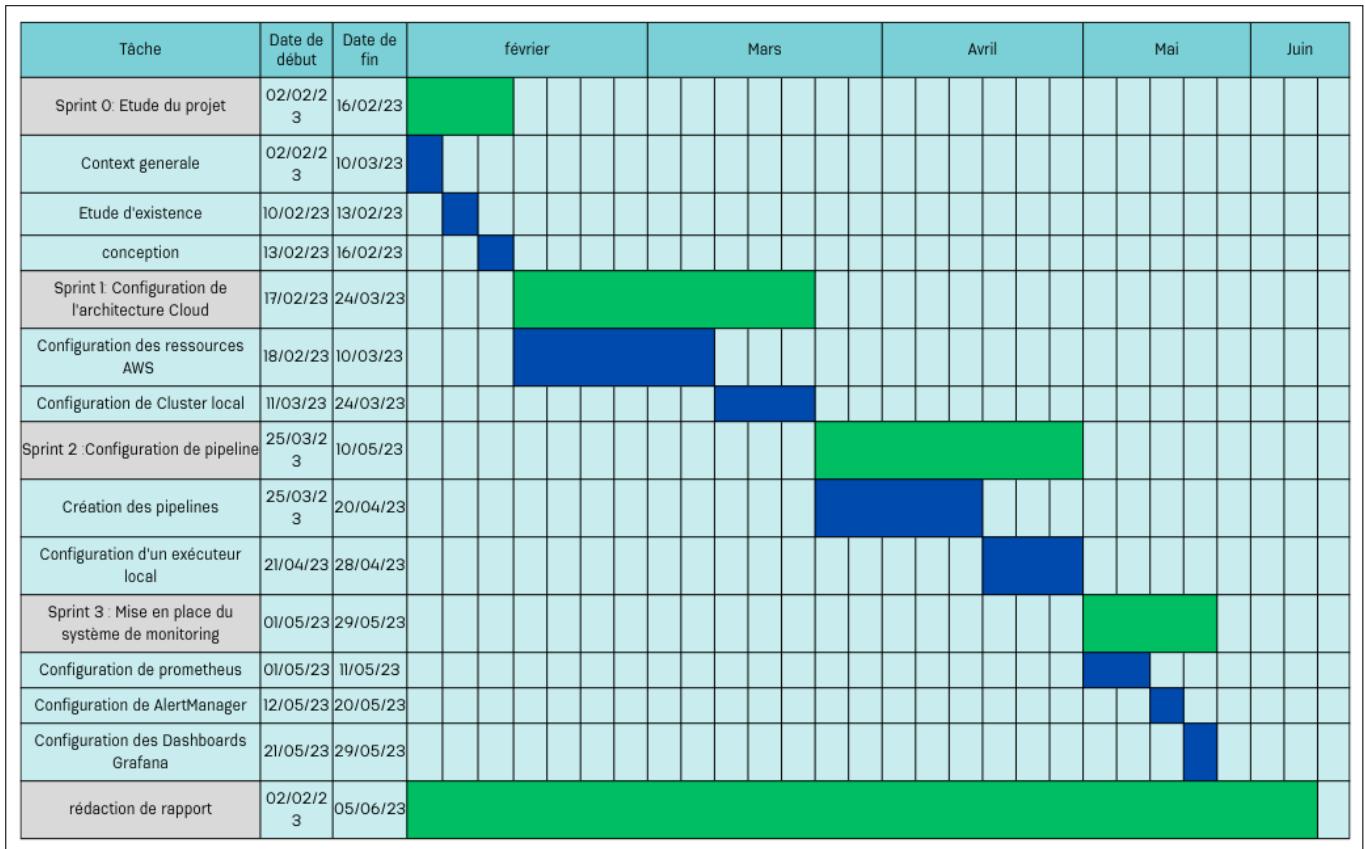


FIGURE 1.7 – Diagramme de Gantt

1.6 Les solutions existantes

On va passer en revue les solutions similaires à notre solution. Nous allons étudier les points forts ainsi que les points faibles de ces solutions afin de montrer pourquoi nous devrions développer notre propre solution. Voici une présentation de certaines solutions existantes :

1.6.1 Azure Devops

Azure DevOps prend en charge une culture collaborative et un ensemble de processus qui rassemblent les développeurs, les responsables de projets et les contributeurs pour développer des logiciels. Elle permet aux organisations de créer et d'améliorer les produits à un rythme plus rapide que possible avec les approches traditionnelles de développement de logiciels. Azure DevOps Services vous donne également accès aux serveurs de génération et de déploiement cloud et aux insights sur les applications. Démarrer gratuitement et créez une organisation. Ensuite, chargez votre code pour partager ou contrôler le code source. Commencez à suivre votre travail à l'aide de Scrum, Kanban ou d'une combinaison de méthodes [9] (voir figure 1.8). Elle a des points forts mais aussi des points faibles :

- Points forts :

– Azure DevOps offre une haute disponibilité, sécurité solide et offre de bonnes options d'évolutivité.

– Azure DevOps est un outil informatique qui adapte en fonction des exigences du projet sans problème, offre à l'utilisateur une flexibilité en matière de gestion de l'infrastructure.

- **Points faibles :**

– Azure DevOps propose de nombreuses fonctions prêtes à l'emploi, mais la personnalisation peut être limitée. Les organismes ayant des besoins spéciaux peuvent avoir de la difficulté à adapter l'outil à leurs besoins précis.

– Azure DevOps fonctionne particulièrement sur cloud qui nécessite une connexion internet pour fonctionner, l'absence d'accès hors ligne peut constituer un inconvénient dans les situations où les personnes doivent travailler à distance avec une connectivité Internet limitée.

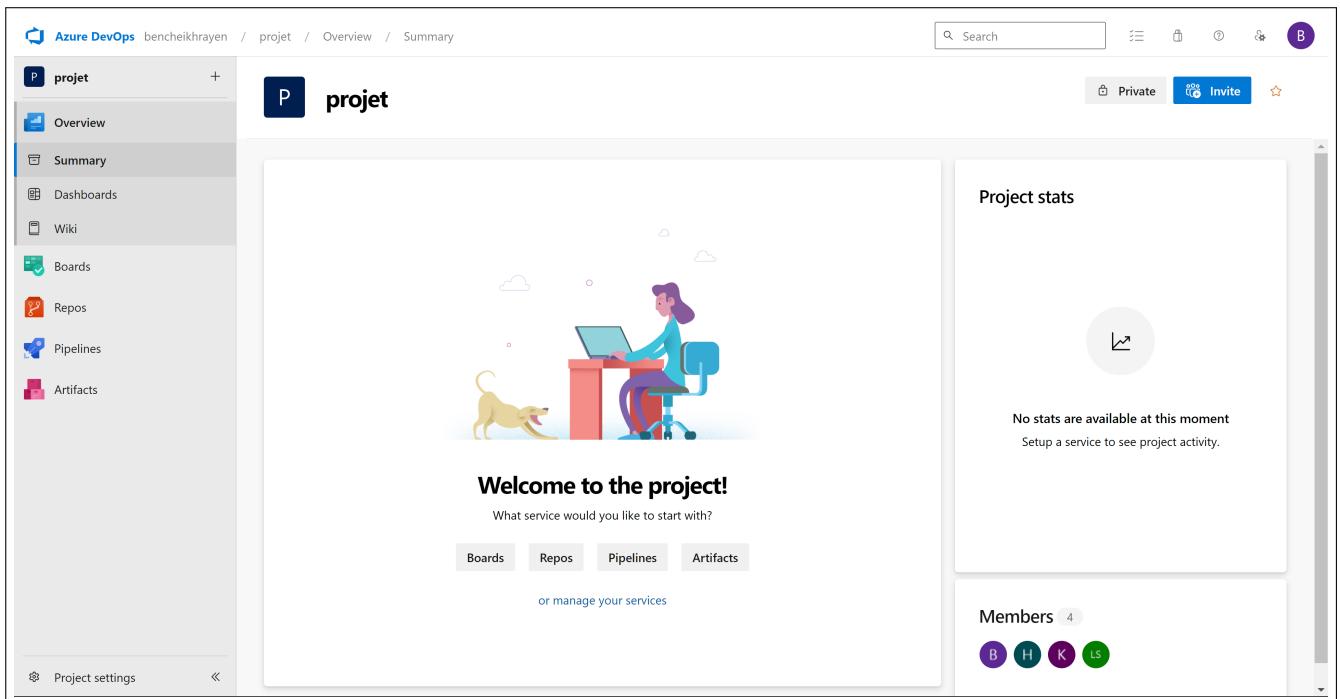


FIGURE 1.8 – Azure DevOps

1.6.2 Amazon Web Service DevOps

AWS fournit un ensemble de services flexibles, conçus pour permettre aux entreprises de créer et livrer des produits avec plus de rapidité et de fiabilité à l'aide d'AWS et des pratiques de DevOps. Ces services simplifient la mise en service et la gestion de l'infrastructure, le déploiement de code d'application, l'automatisation des processus de publication de logiciel et le suivi des performances de l'application et de l'infrastructure [10] (voir figure 1.9).

Cependant, comme toute autre technologie, AWS présente sur des points forts et des points faibles :

- Points forts :

– Aws offre une grande évolutivité pour les entreprises peuvent facilement augmenter ou réduire leurs ressources en fonction de leurs besoins .Aussi il présente un niveau de fiabilité élevé et il fournit aux entreprises une gamme de services, notamment le stockage, la gestion de bases de données, la puissance de calcul et l'analyse

– AWS fournit des outils de redondance et de tolérance aux défaillances, tels que l'auto-scaling et la disponibilité de plusieurs zones. Ces caractéristiques renforcent la disponibilité des applications et assurent des performances fiables.

- Points faibles :

– AWS est complexe à mettre en place et à gérer ses ressources. Aussi, les entreprises doivent prendre des mesures pour garantir la protection de leurs données.

– L'adoption d'AWS DevOps peut créer une dépendance envers les différents services, ce qui peut mener à un verrouillage des fournisseur.

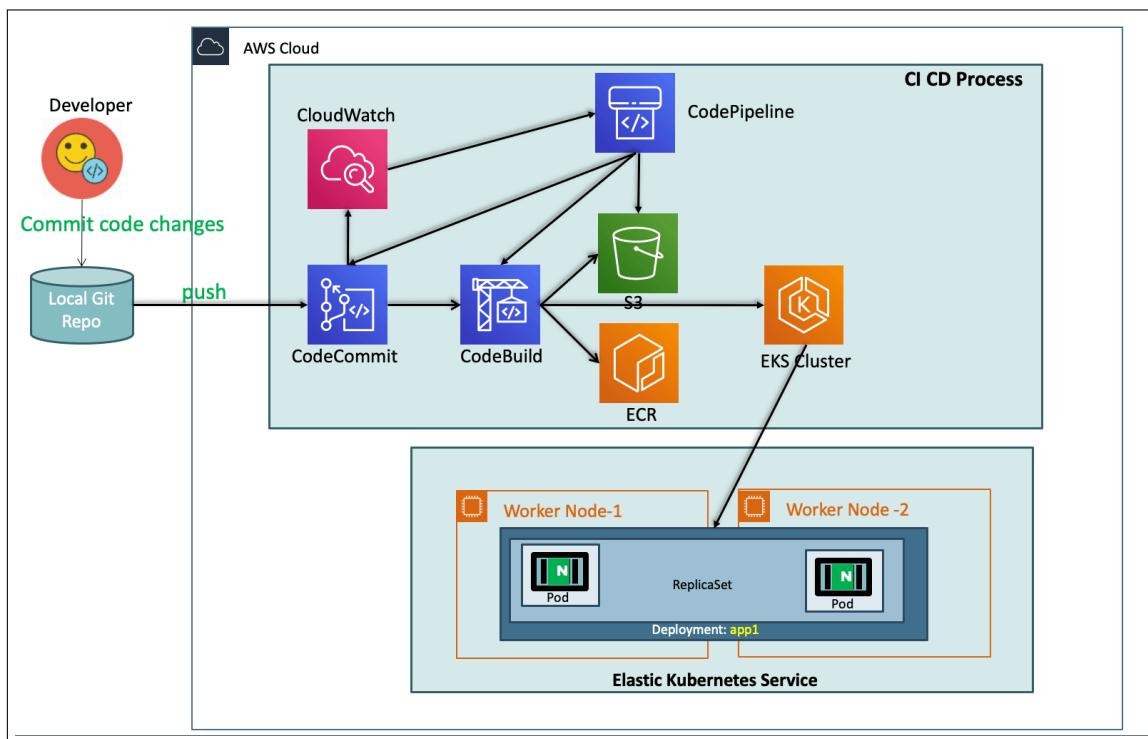


FIGURE 1.9 – Les Services AWS utilisé pour DevOps.

1.6.3 Google Cloud Platform DevOps

Google propose de nombreux services et fonctionnalités visant à aider les ingénieurs DevOps à disposer de tout ce dont ils ont besoin pour respecter les normes de qualité et de sécurité les plus élevées tout en automatisant la majorité du processus. De plus, vous travaillez avec des outils GCP DevOps efficaces qui non seulement augmentent la qualité des applications, mais améliorent également la vitesse du cycle de développement. Ces outils DevOps s'adressent directement aux ingénieurs logiciels car ils permettent une configuration rapide et simple, avec une

interface intuitive qui facilite l'utilisation efficace des outils et méthodologies de livraison continue et de déploiement continu (intégration continue/ déploiement continu(CI/CD))[11](Voir figure 1.10).

Ainsi GCP a des points forts et des points faibles :

- **Points forts :**

- GCP fournit des services d'intégration et de développement d'applications et assure une bonne sécurité.
- GCP DevOps rende facile la collaboration entre les équipes de développement et les équipes opérationnelles grâce à des services partagés.

- **Points faibles :**

- Quelque intégration peut être plus limitées avec les services et les outils de Google par rapport à des solutions locales. De plus, la configuration initiale du GCP est complexe en raison de la grande diversité de services et d'options disponibles.
- L'utilisation efficace de GCP DevOps exige un personne qualifié spécialisé dans les technologies cloud, l'automatisation et la gestion de l'infrastructure.

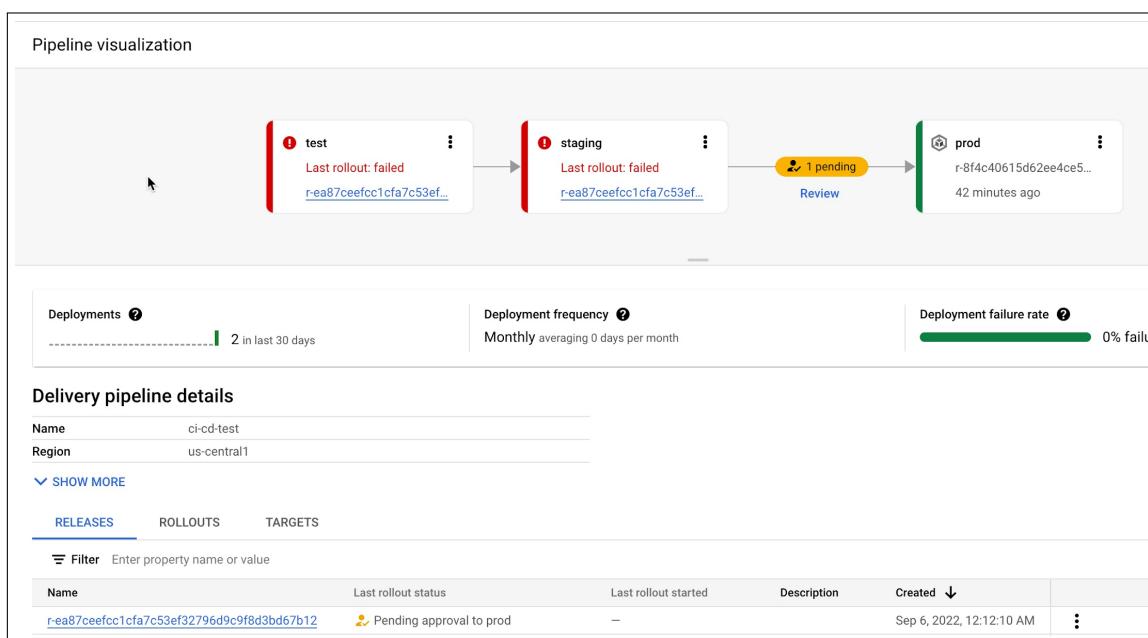


FIGURE 1.10 – Interface de GCP Pipeline.

1.7 Critique des solutions existantes

Nous allons comparer les solutions présentées ci-dessus selon plusieurs critères qu'on va définir par rapport aux besoins de l'entreprise(voir tableau 1.1).

C1 -La confidentialité des données d'application : La solution doit privilégier la confidentialité des données puisque certains clients préfèrent leur

	Azure DevOps	Amazon Web Service DevOps	Google Cloud Platform DevOps
C1	X	X	X
C2	✓	✓	✓
C3	✓	✓	X
C4	X	X	X

Légende :
 X : La solution ne conforme pas à la critère.
 ✓ : La solution conforme à la critère.

TABLE 1.1 – Tableau comparatif

application exécutée sur le serveur local de la société.

C2 -Intégration de solutions source libre : Amazon cloud, azure cloud ou toute autre solution cloud doit intégrer certaines software open source comme services à utiliser dans votre architecture cloud.Bien que les logiciels libres puissent être hautement personnalisables, le niveau de personnalisation fourni par un fournisseur de services peut être restreint.

C3 -Coût de projet : L'utilisation d'Amazon Cloud ainsi tout autre fournisseur pour déployer une application dans le Cloud semble peu coûteux au début, mais après que la charge de travail augmente, les coûts aussi augmentent. Il s'agit de trouver une solution qui profite du cloud et des avantages du logiciel libre.

C4 -Problèmes de compatibilité : La personnalisation peut présenter des problèmes de compatibilité, surtout si la personnalisation interagit avec d'autres services ou composants au cours du déploiement. Cela peut entraîner des temps d'arrêt ou d'autres problèmes de performance.

En se basant sur le tableau comparatif ci-dessus, on constate que les solutions basées uniquement sur le cloud ne sont pas utiles, et qu'une combinaison de ressources cloud et locales est conforme à notre projet.

Notons que cette évaluation ne concerne que notre situation, dans d'autres cas les solutions existantes peuvent être utile.

1.8 Solution proposée

Pour rectifier les problèmes auxquels l'équipe DevOps fait face chaque jour et afin de préparer un environnement de développement flexible, Mobelite nous a proposé de mettre en place une solution Hybride cloud qui bénéficie à la fois de certains services de cloud avec un accès privé aux données. En effet, le projet consiste à implémenter la conteneurisation qui nous offrira une virtualisation des ressources de manière légère, flexible et puissante. Le déploiement d'un kubernetes

cluster qui est un système de gestion de conteneurs qu'on va utiliser pour optimiser le déploiement des applications avec l'utilisation d'autres produits comme Ansible pour l'optimisation de cette infrastructure. La solution a des objectifs importants pour l'entreprise comme la réduction des coûts de développement et d'exploitation. Le Cycle de développement sera plus court grâce à l'automatisation et le déploiement rapide des nouveaux environnements. Aussi, la solution garantira la supervision totale et continue de la plateforme, et la haute disponibilité du système.

Conclusion

Dans ce chapitre nous avons présenté la société d'accueil, la problématique et la méthodologie de développement. Aussi, nous avons réalisé une étude de quelques solutions existantes, puis nous avons réalisé une comparaison de l'existant pour connaître les solutions compatibles avec notre sujet. Après cela, nous avons présenté la solution proposée par mobelite.

Chapitre 2

Spécification et analyse des besoins

Introduction

L'objectif de l'étape de spécification et d'analyse des besoins est de déterminer les fonctionnalités différentes attendues du système. Au cours de ce chapitre, nous présentons d'abord les acteurs concernés dans notre système. Ensuite, nous allons illustrer les besoins fonctionnels et non fonctionnels. Ensuite nous allons détailler le product backlog de notre projet. Ces exigences seront exprimées enfin sous forme de diagramme de cas d'utilisation qui sera détaillé par des scénarios possibles.

2.1 Acteurs

En génie logiciel et plus particulièrement en UML, un acteur est une entité qui définit le rôle joué par un utilisateur ou par un système qui interagit avec le système modélisé[12].

–Développeur :C'est l'utilisateur qui peut modifier le code source d'un objectif donné (ajouter une fonctionnalité, corriger l'erreur, etc.) puis les déposer dans Github et ensuite suivre la compilation de l'application.

– Ingénieur DevOps : Est considéré comme l'acteur principal, son rôle est la mise en place du cluster, pipeline, surveiller l'état du système et configurer son infrastructure.

– Client : C'est l'utilisateur qui peut accéder à l'application déployée sur le cluster au moyen d'un site Web.

2.2 Besoins fonctionnels

Pour le bon fonctionnement de notre projet, il est nécessaire de définir concrètement les fonctionnalités qui seront implémentées dans le but de les rendre plus appropriées aux besoins de l'entreprise. Pour cela nous avons allons présenter les besoins fonctionnels de notre projet :

- Mettre en place une solution hybride qui permet à l'entreprise d'exécuter des applications localement tout en profitant des avantages des services cloud.
- Orchestrer les conteneurs pour garantir une haute disponibilité pour le déploiement et l'exécution des applications.
- Automatiser le processus de déploiement avec Ansible pour aider avec les charges de travail.
- Surveiller l'état du déploiement.
- Créer un répertoire pour toutes les images Docker dans ECR (Elastic Container Registry).
- Enregistrer les artefacts dans un répertoire Nexus.
- Utiliser Prometheus et Grafana pour surveiller le cluster kubernetes.
- Assurer la qualité de code source avec SonarQube.
- Gérer automatique des étapes de déploiement de clusters.

2.3 Besoins non-fonctionnels

Les besoins non fonctionnels établissent toutes les conditions nécessaires au bon fonctionnement du système et à l'amélioration de la qualité des services. Et pour répondre aux exigences fonctionnelles, notre projet doit respecter une série de propriétés contribuant à une meilleure qualité de la solution obtenue. Nous déterminerons l'ensemble des contraintes à respecter pour garantir le bon déroulement du projet. Parmi les critères nous citons :

- Confidentialité : Notre solution permettra de garantir la sécurité des données,

des applications et des utilisateurs.

–Performance : Rapidité du déploiement d’application sur des pods kubernetes et assurer la bonne exécution de ses applications.

texttt

–Disponibilité : Les clusters Kubernetes sont conçus pour permettre une évolutivité horizontale, ce qui signifie que les applications peuvent être déployées sur plusieurs nœuds et gérer de manière dynamique les charges de travail en fonction des besoins.

–Maintenance : La modification d’un déploiement dans un kubernetes cluster doit être facilement maintenable et adaptable à de nouvelles exigences de sorte qu’il peut y avoir un changement ou l’ajout de nouvelle fonctionnalité.

–Portabilité : Le système doit être flexible et capable de prendre en charge de nouvelles fonctions et extensions et doit pouvoir fonctionner avec un nouveau composant (RAM,stockage) et avec une modification minimale.

–Extensibilité : Le système doit maintenir ses hautes performances sous pression et ajuster ses paramètres pour répondre à la demande, possibilité d’ajouter des nœuds en cas de montée en charge.

2.4 Product Backlog

Product Backlog est une liste de toutes les tâches connues pour être traitées par le produit. Il s’agit de la seule source des besoins pour toute modification du produit. Les besoins sont précisés par les user stories. Un "user story" se présente sous la forme suivante (voir tableau 2.1) :

id	Theme	Acteur	Description	Priorité
id 1	gérer code source	Développeur	Permet de déposer le code ou le récupérer	élevée
id2	Suivre build d'application	Développeur	Peut voir la compilation de l'application	élevée
id3	analyser qualité de code	Ingénieur DEVOPS	Permet d'analyser le code source d'un projet et de détecter les erreurs de qualité avec SonarQube .	élevée
id 4	gérer liste des images docker	Ingénieur DEVOPS	Consiste à répertorier et organiser les images Docker qui ont été créées .	élevée
id 5	gérer les artefacts d'application	Ingénieur DEVOPS	Consiste à stocker et à gérer les différents composants d'une application, les bibliothèques, les configurations, les scripts et autres ressources .	élevée
id 6	Configurer l'architecture cloud	Ingénieur DEVOPS	Consiste à concevoir et à déployer une infrastructure informatique dans le cloud, en utilisant des services cloud pour répondre aux besoins.	élevée
id 7	Configurer le pipeline	Ingénieur DEVOPS	Est un processus important pour automatiser le processus de développement, de test et de déploiement d'une application.	élevée
id 8	Surveiller l'état du cluster	Ingénieur DEVOPS	Surveille l'état de tous les nœuds dans le cluster pour s'assurer qu'ils sont en bon état de fonctionnement et qu'ils répondent aux demandes des applications.	élevée
id 9	gérer la configuration du déploiement	Ingénieur DEVOPS	Assure que les applications sont déployées de manière cohérente et fiable	élevée
id 10	accès à l'application déployée	Client	s'assure qu'elle est accessible uniquement par les utilisateurs autorisés.	élevée

TABLE 2.1 – Tableau du Product Back-log

2.5 Diagramme de cas d'utilisation Global

Le diagramme de cas d'utilisation global est utilisé pour une représentation du comportement fonctionnel d'un système logiciel.

Le diagramme ci-dessous présente les cas d'utilisation généraux et les acteurs qui interagissent avec le système(Voir figure 2.1).

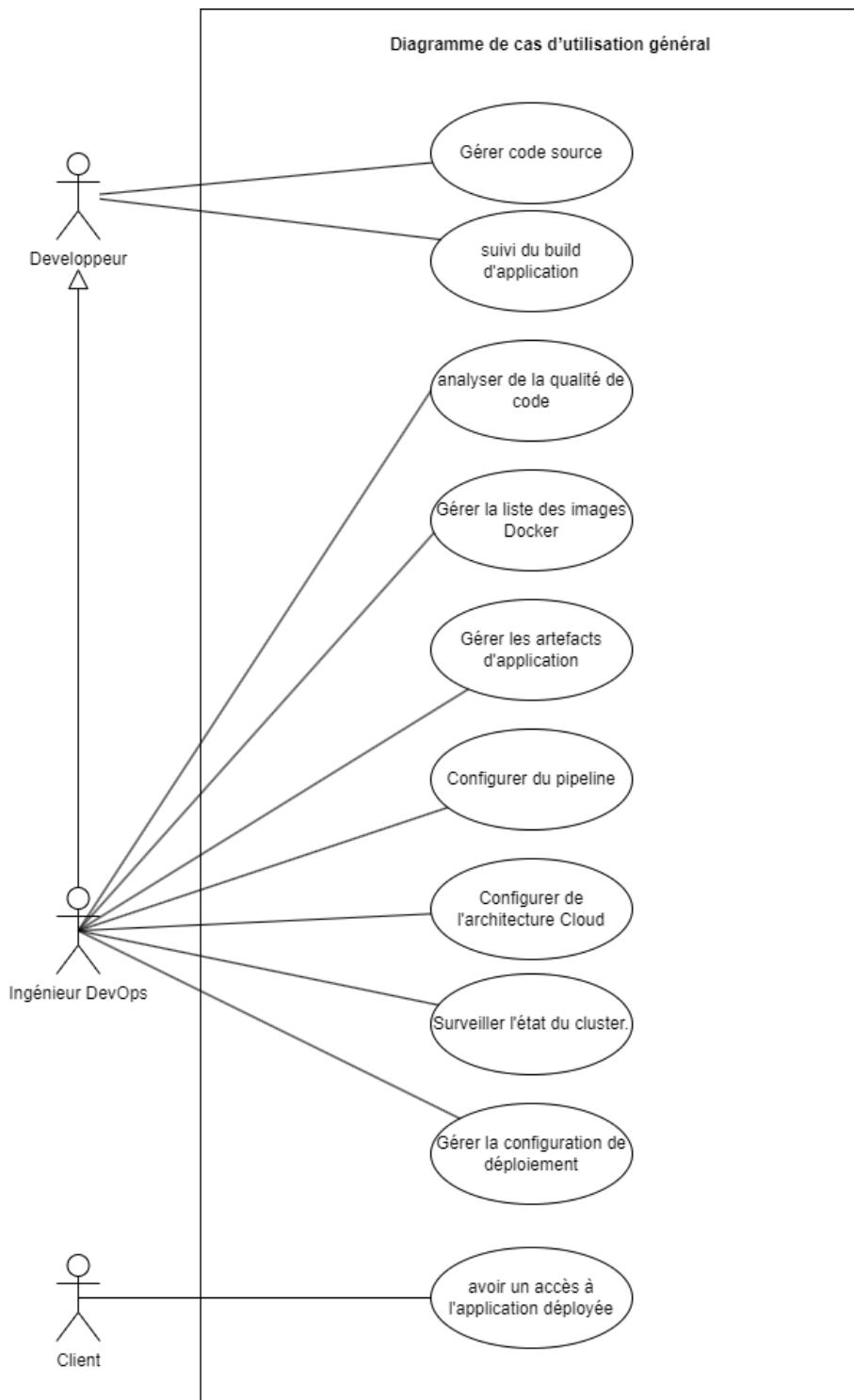


FIGURE 2.1 – Diagramme de cas d'utilisation Globale

Dans ce qui suit , le raffinement des différents cas d'utilisation.

2.5.1 Cas d'utilisation "Gérer code source"

Dans cette section, nous décrirons de façon détaillée ce cas d'utilisation.

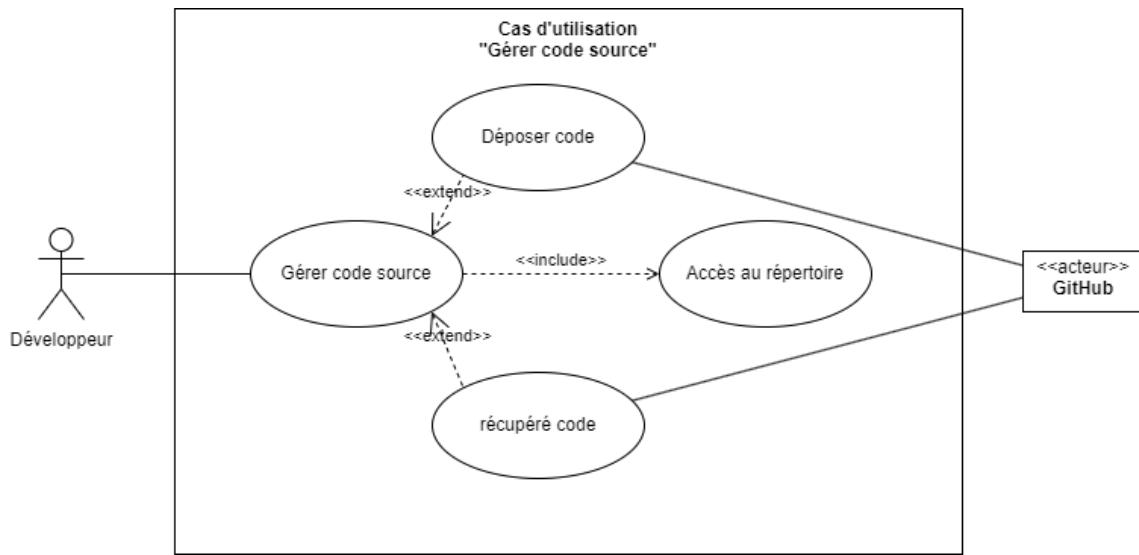


FIGURE 2.2 – Cas d'utilisation :Gérer code source

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle du principales fonctionnalités mentionnées ci-dessus (Voir tableau 2.2) :

Titre	Gérer code source
Acteur	Développeur
Description	le développeur peut déposer ou récupérer le code à travers le command "pull" ou "push".
Pré conditions	Une connexion établie entre le PC du développeur et le répertoire Git de l'application.
Post conditions	Code envoyé au répertoire git.
Scénario nominal	1- Avoir un accès au répertoire. 2 - Gérer code source. 3 - Déposer ou récupérer le code dans le répertoire github.
Scénario alternatif	1- La connexion entre la machine du développeur et Git ne peut pas être établie. 2 - Le code ne peut pas être envoyé. 3 - Retour à l'étape 1 du Scénario nominal.

TABLE 2.2 – Description de cas d'utilisation :Gérer code source

2.5.2 Cas d'utilisation "Suivi de la build d'application"

Dans cette section, nous décrirons de façon détaillée ce cas d'utilisation.

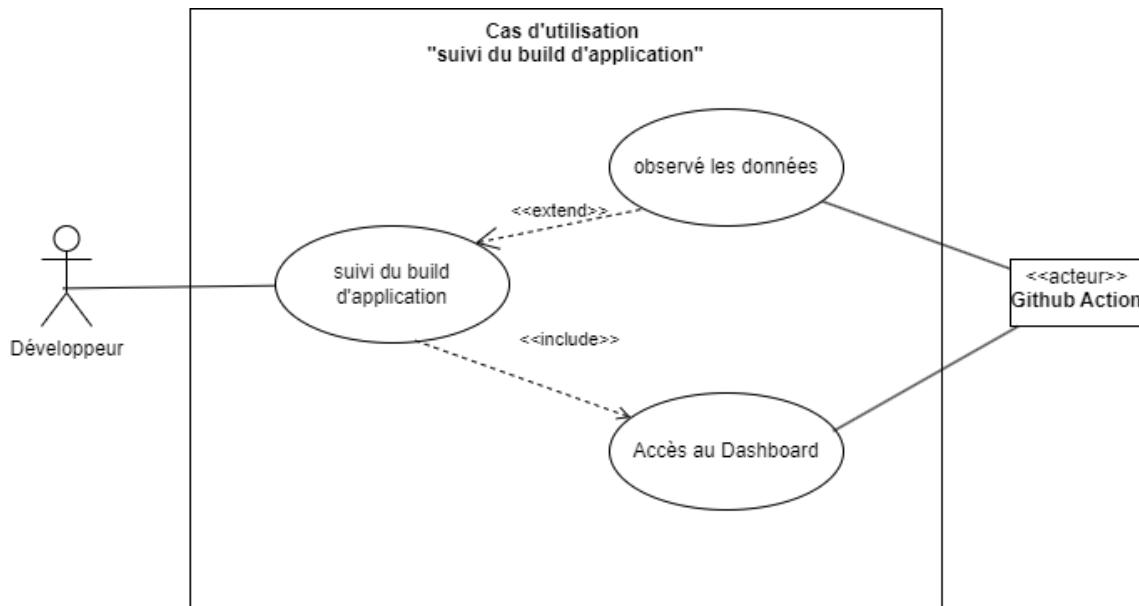


FIGURE 2.3 – Cas d'utilisation :Suivi la build d'application

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle des principales fonctionnalités mentionnées ci-dessus(voir tableau 2.3) :

Titre	Suivi du build de l'application
Acteur	Développeur
Description	Le développeur peut suivre la construction d'application pour connaître si la modification ajoutée au code source est valide ou non.
Pré conditions	Une connexion établie entre le développeur et Github actions.
Post conditions	Affichage de Dashboard Github Actions.
Scénario nominal	1- Avoir un accès au Dashboard 2- Suivi du build de l'application 3- L'observation de la construction de l'application rapporte un succès.
Scénario alternatif	1 - La connexion entre développeur et Github Actions ne peut pas être établie. 2 - Retour à l'étape 1 du Scénario nominal.

TABLE 2.3 – Description de cas d'utilisation :Suivi du build de l'application

2.5.3 Cas d'utilisation "Gérer la configuration de déploiement"

Dans cette section, nous décrirons de façon détaillée ce cas d'utilisation (voir figure 2.4).

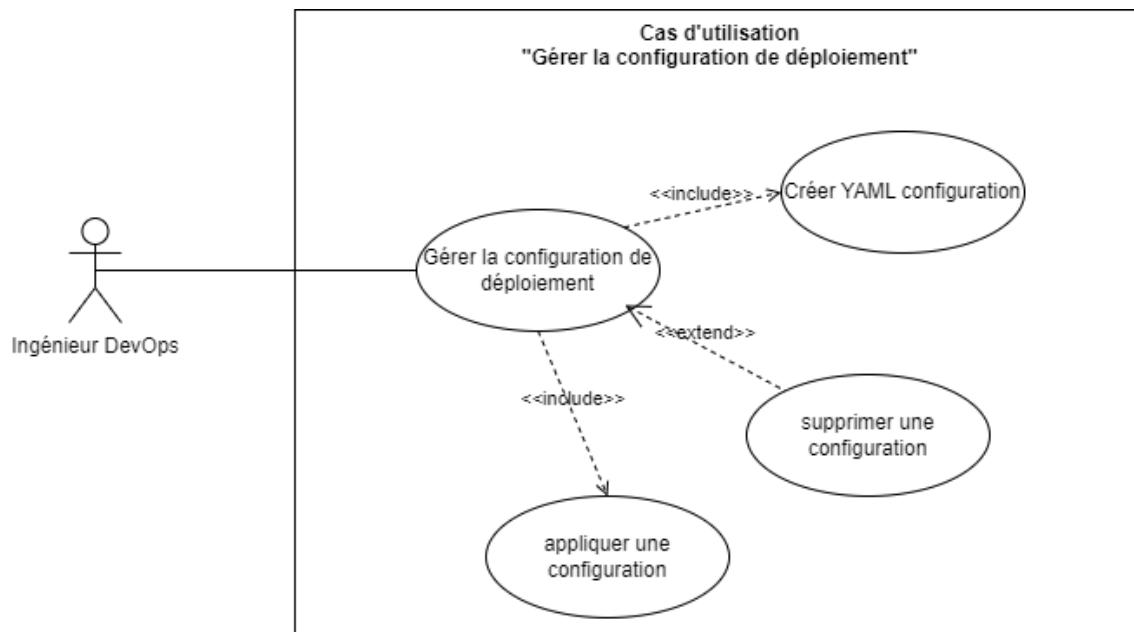


FIGURE 2.4 – Cas d'utilisation :Gérer la configuration de déploiement

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle des principales fonctionnalités mentionnées ci-dessus (voir table 2.4) :

Titre	Gérer la configuration de déploiement
Acteur	Ingénieur DevOps
Description	L'ingénieur DevOps gère la configuration de déploiement pour assurer la meilleure performance du cluster.
Pré conditions	Fonctionnement correct du cluster EKS.
Post conditions	Une configuration ou modification d'une configuration déjà existante.
Scénario nominal	<ol style="list-style-type: none"> 1 - Lancer la configuration du déploiement. 2 - La configuration est lancée dans le contrôle plane d'EKS.
Scénario alternatif	<ol style="list-style-type: none"> 1 - La nouvelle configuration n'est pas flexible. 2 - Un problème apparaît causé par la nouvelle configuration. 3 - Retour à l'étape 1 du Scénario nominal.

TABLE 2.4 – Description de cas d'utilisation :Gérer la configuration de déploiement

2.5.3.1 Cas utilisation "Gérer liste des images Docker"

Dans cette section, nous décrirons de façon détaillée le cas d'utilisation (voir figure 2.5).

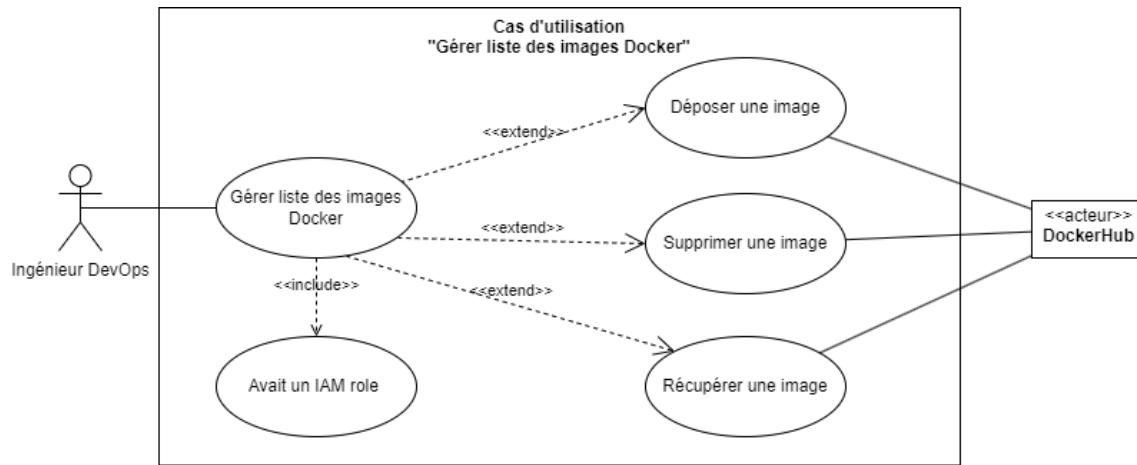


FIGURE 2.5 – Cas d'utilisation :Gérer liste des images Docker

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle des principales fonctionnalités mentionnées ci-dessus (voir table 2.5) :

Titre	Gérer liste des images Docker
Acteur	Ingénieur DevOps
Description	L'ingénieur DevOps gère la liste des images Docker
Pré conditions	Connexion au répertoire de l'application dans ECR.
Post conditions	Une image modifiée ,supprimée ou ajoutée.
Scénario nominal	1 - Se connecter ou l'enregistrement de l'image n'est pas valide. 2 - Configurer liste des images Docker . 2 - Déposer ou récupérer le code dans le répertoire ECR. 3 - L'image est enregistrée dans la répertoire ECR.
Scénario alternatif	1 - La connexion ou l'enregistrement de image n'est pas valide. 2 - Echec du dépôt ou récupération l'image Docker . 3 - Retour à l'étape 1 du Scénario nominal.

TABLE 2.5 – Description de cas d'utilisation :Gérer liste des images Docker

2.5.4 Cas utilisation "Gérer les artefacts d'application"

Dans cette section, nous décrirons de façon détaillée ce cas d'utilisation (voir figure 2.6).

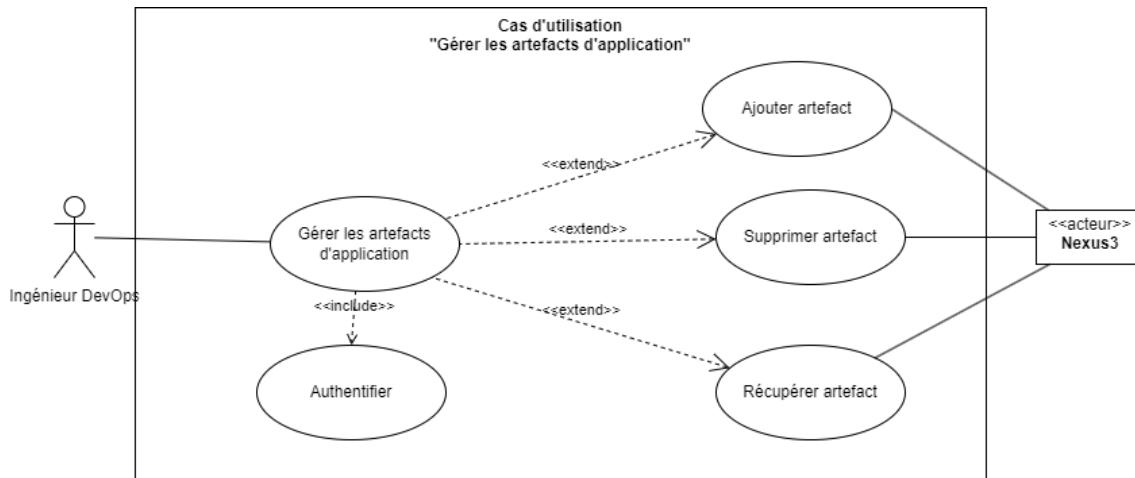


FIGURE 2.6 – Cas d'utilisation :Gérer les artefacts d'application

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle des principales fonctionnalités mentionnées ci-dessus (voir table 2.6) :

Titre	Gérer les artefacts de l'application
Acteur	Ingénieur DevOps
Description	L'ingénieur DevOps gère la liste des artefacts pour assurer l'enregistrement de différentes versions d'application.
Pré conditions	Connexion HTTP au serveur Nexus.
Post conditions	Une répertoire avec les différentes artefacts de l'application.
Scénario nominal	1 - Saisir les données d'authefication pour accéder à l'interface . 2 - Configurer les artifacts d'application . 3 - la connexion est réussite et la modification dans le répertoire est enregistrée dans Nexus .
Scénario alternatif	1 - La connexion échoue ou la modification n'est pas enregistrée. 2 - Retour à l'étape 1 du Scénario nominal.

TABLE 2.6 – Description de cas d'utilisation :Gérer les artefacts d'application

Conclusion

Dans ce chapitre nous avons décrit les différents besoins fonctionnels et non fonctionnels. Aussi, nous avons décrit les différents acteurs de système. Puis nous avons présentée les "User stories" en forme d'un product backlog. Enfin, nous avons illustrée les cas d'utilisation avec des diagrammes. Dans le chapitre suivant nous passerons à la phase de conception du projet.

Chapitre 3

Conception

Introduction

À travers ce chapitre, nous commencer la conception de notre application qui présente une étape fondamentale qui précède la réalisation. Nous commençons par une description des différents diagrammes de séquence, classe et déploiement. Nous procérons ensuite à la définition de la méthodologie de conception. Et nous terminons par représenter l'architecture technique de notre projet.

3.1 Conception du système

Dans ce qui suit, nous allons présenter les différents diagrammes de conception.

3.1.1 Diagramme de séquence global

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique[13] (voir figure 3.1).

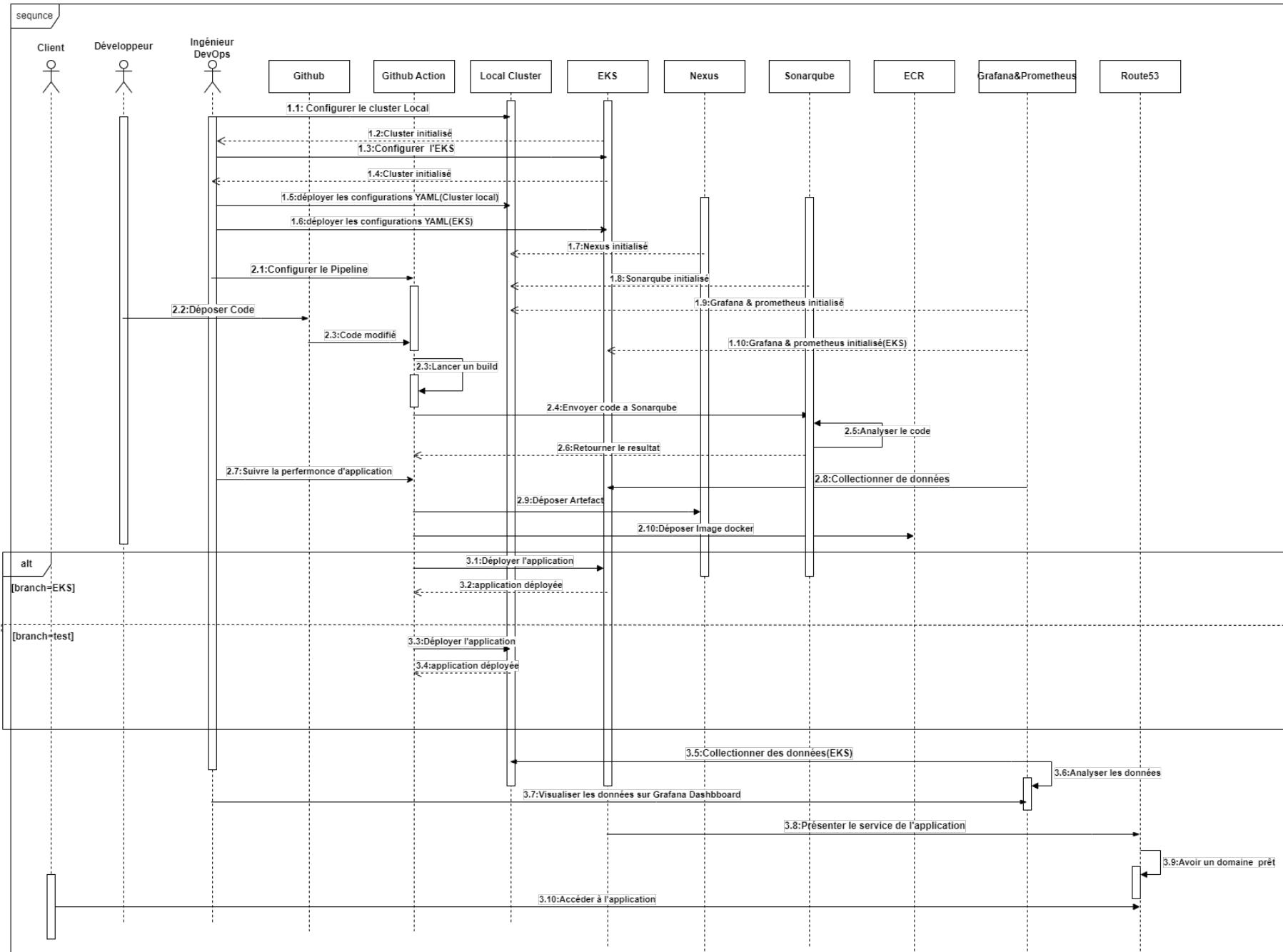


FIGURE 3.1 – Diagramme de séquence Global

3.1.2 Diagramme de séquence Détailée

Pour une meilleure présentation et illustration de projet nous allons décrire les différents cas d'utilisations principales avec son diagramme de séquence détaillée.

3.1.2.1 Diagramme de séquence « Configuration de l'architecture cloud »

Nous détaillons ci-dessous les interactions diagramme de séquence « Configuration de l'architecture cloud »

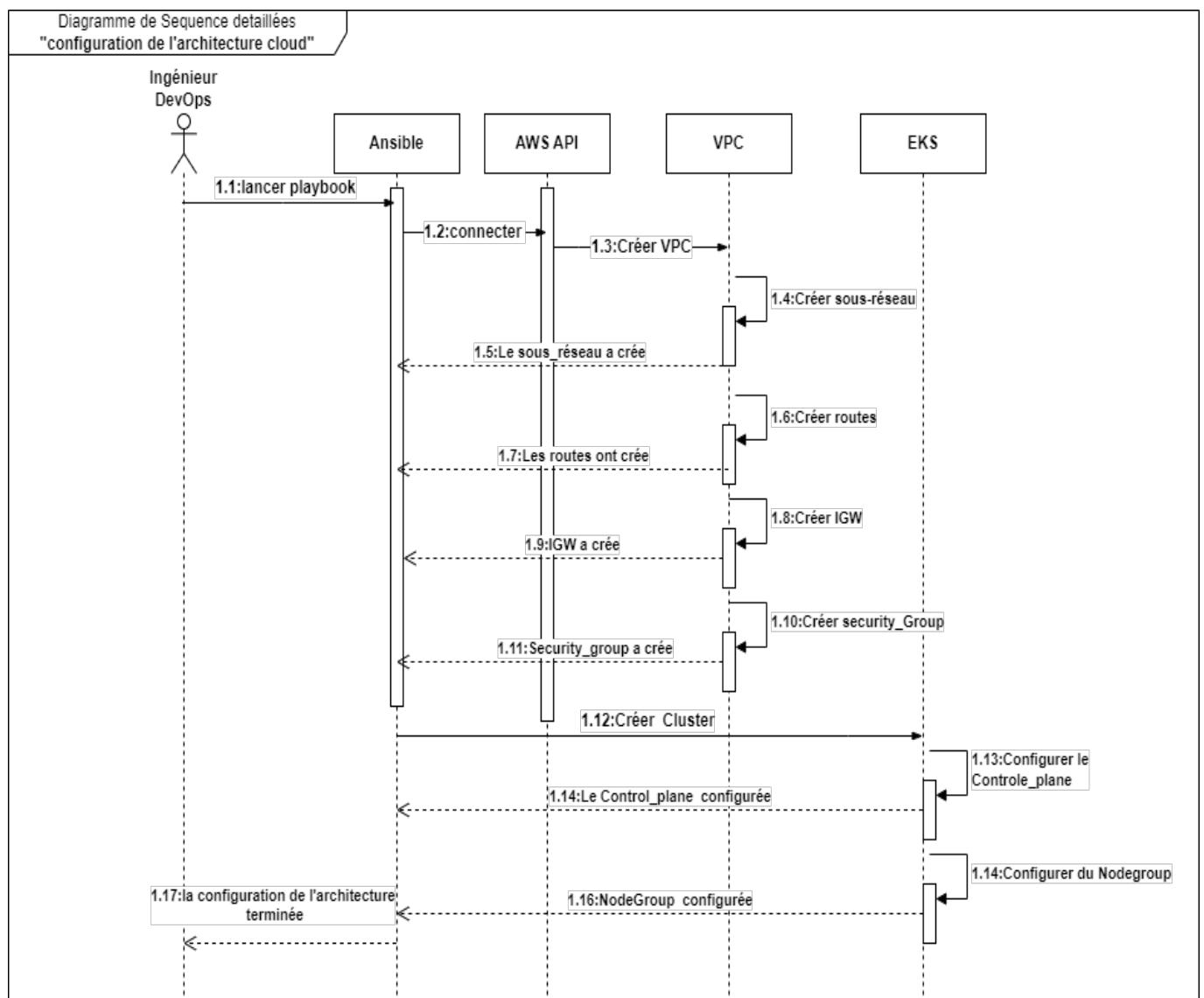


FIGURE 3.2 – Diagramme de séquence « Configuration de l'architecture cloud »

3.1.2.2 Diagramme de séquence « Configuration de cluster local »

Nous détaillons ci-dessous les interactions de diagramme de séquence « Configuration de cluster local » (voir figure 4.3)

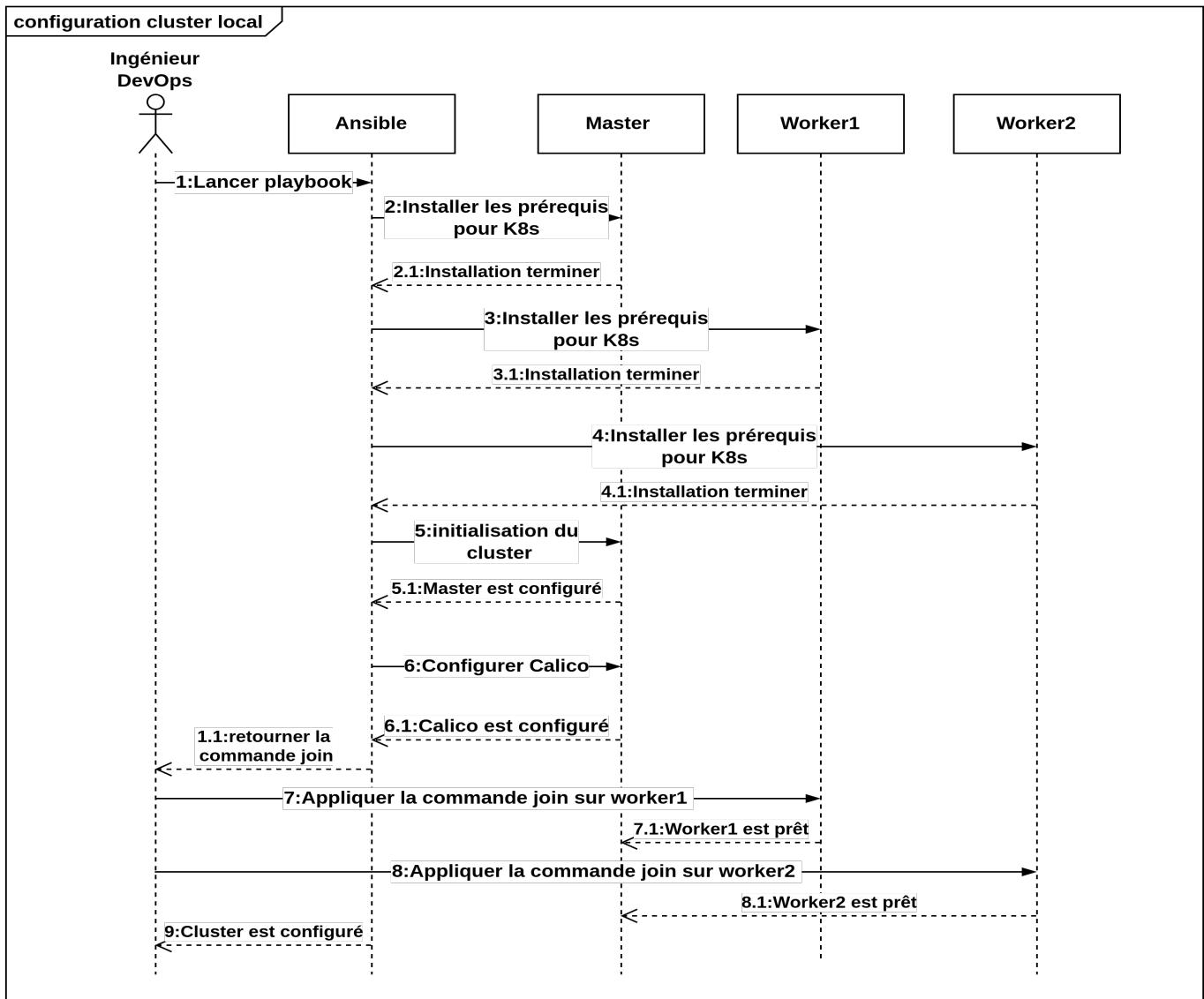


FIGURE 3.3 – Diagramme de séquence « Configuration de cluster local »

3.1.2.3 Diagramme de séquence « Configuration de pipeline »

Nous détaillons ci-dessous les interactions de diagramme de séquence « Configuration de pipeline » (voir figure 4.4)

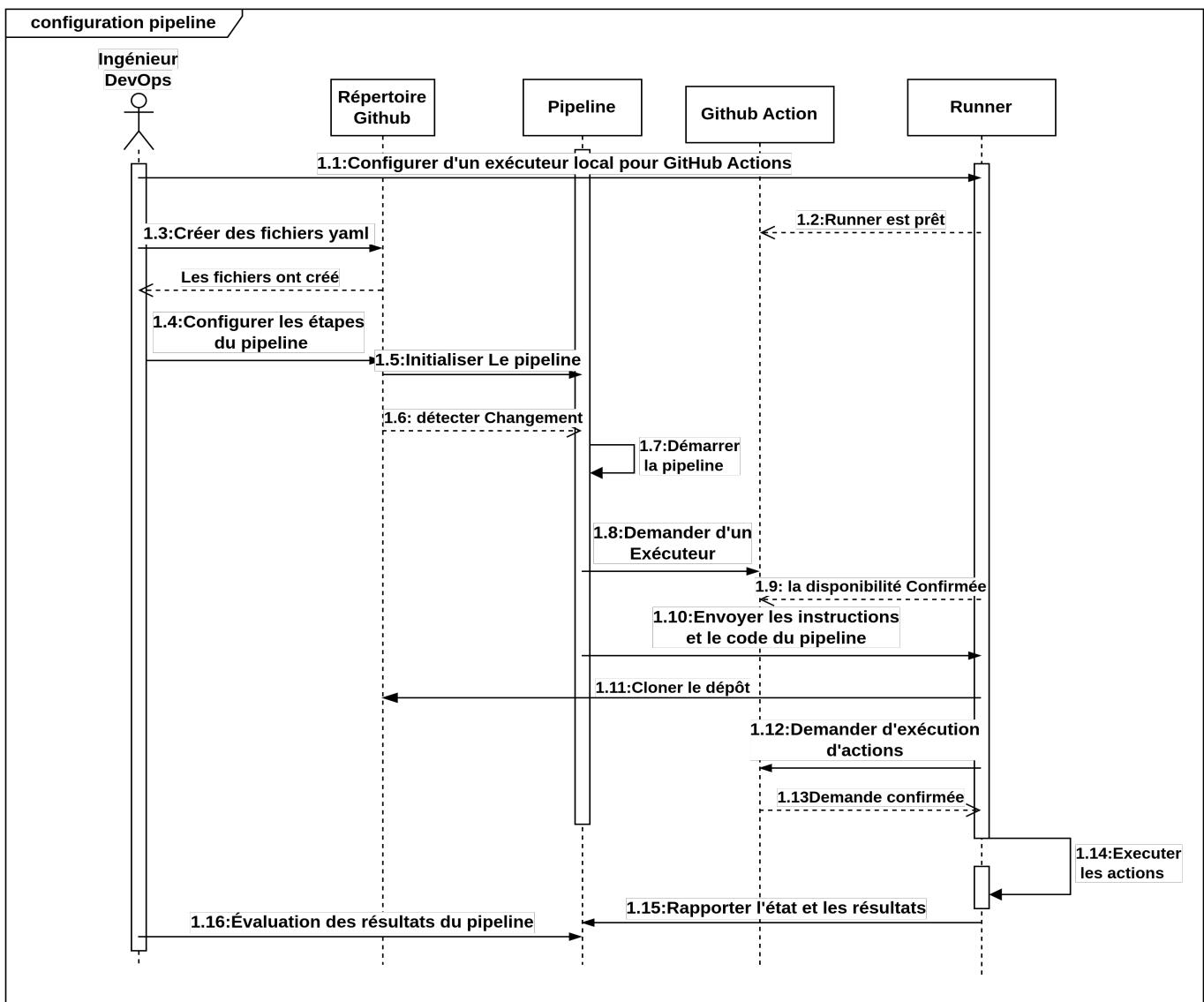


FIGURE 3.4 – Diagramme de séquence « Configuration de pipeline »

3.1.3 Diagramme de classe

Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces du système ainsi que leurs relations. La figure ci-dessous représente le diagramme de classe de notre travail (voir figure 4.5)[14]

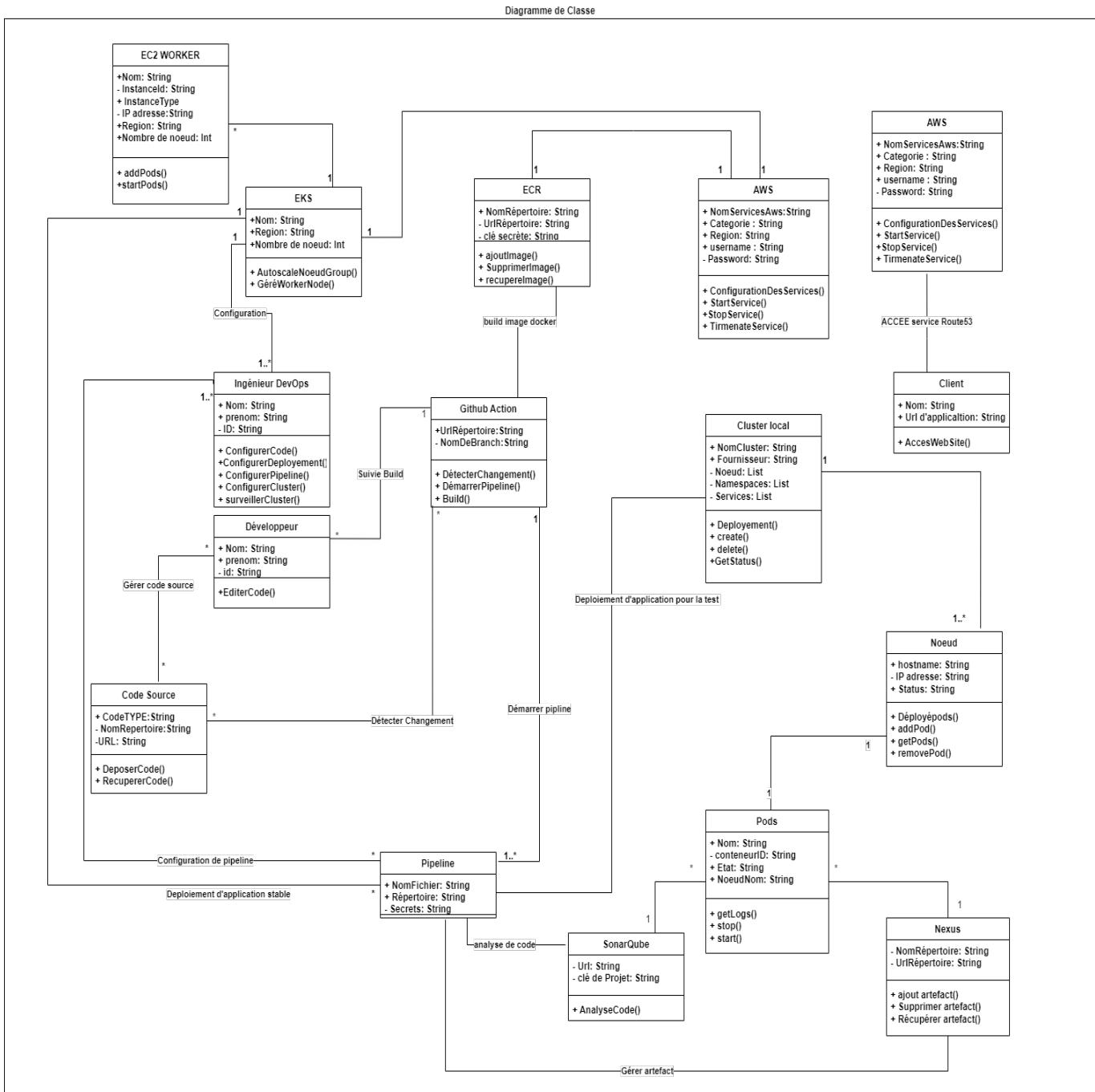


FIGURE 3.5 – Diagramme de classe global

3.1.4 Diagramme de déploiement

Un diagramme de déploiement est une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont répartis ainsi que leurs relations entre eux (voir figure 4.6).[15]

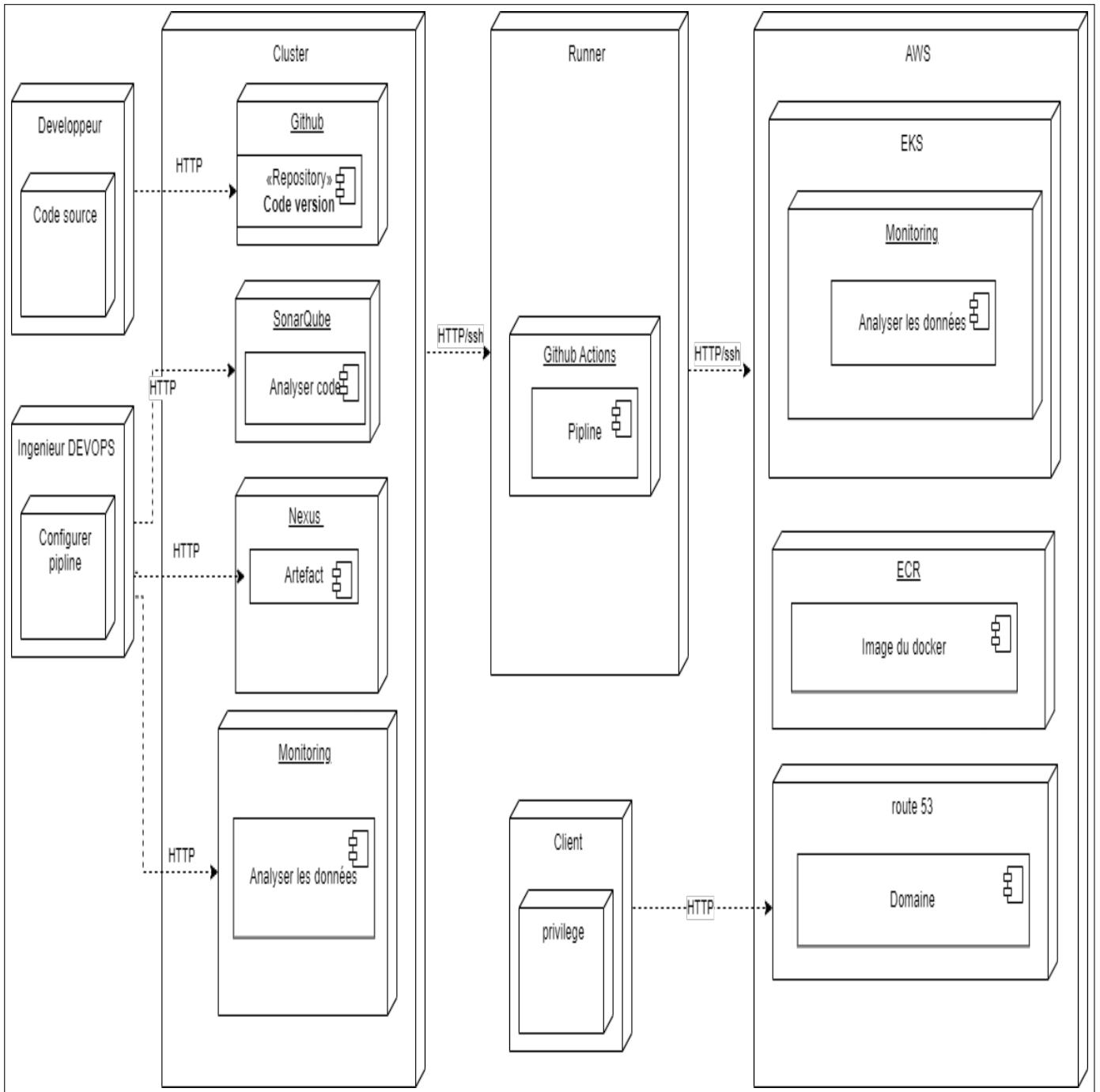


FIGURE 3.6 – Diagramme de déploiement global

3.2 Méthodologie de conception

Il est indispensable de choisir un méthodologie de développement et nous avons choisi la méthodologie (CI/CD).

L’Intégration Continue (CI) et la Livraison Continue (CD) regroupent un ensemble de principes et de pratiques permettant aux équipes de développement d’apporter des changements au code informatique de façon plus fiable et plus fréquente. L’implémentation du CI/CD est au cœur des méthodologies de

développement agile et DevOps. Elle permet aux équipes de développement logiciel de se focaliser sur les besoins de l'entreprise, la qualité du code et la cybersécurité. Les étapes de déploiement sont automatisées. Ainsi, les pipelines CI/CD permettent aux entreprises d'améliorer fréquemment leurs applications tout en s'appuyant sur un processus de livraison fiable. La standardisation des builds, les tests, l'automatisation du déploiement laissent les équipes se focaliser sur l'amélioration des applications plutôt que sur des détails techniques.

Cette pratique est idéale pour la méthode DevOps, car elle évite un mauvais alignement entre des développeurs désirant pousser le code trop fréquemment et les équipes ops en quête de stabilité des applications. L'automatisation permet de pousser des changements de code plus fréquemment, tandis que les configurations standardisées et le testing continu améliorent la stabilité.[?](Voir figure 4.7)

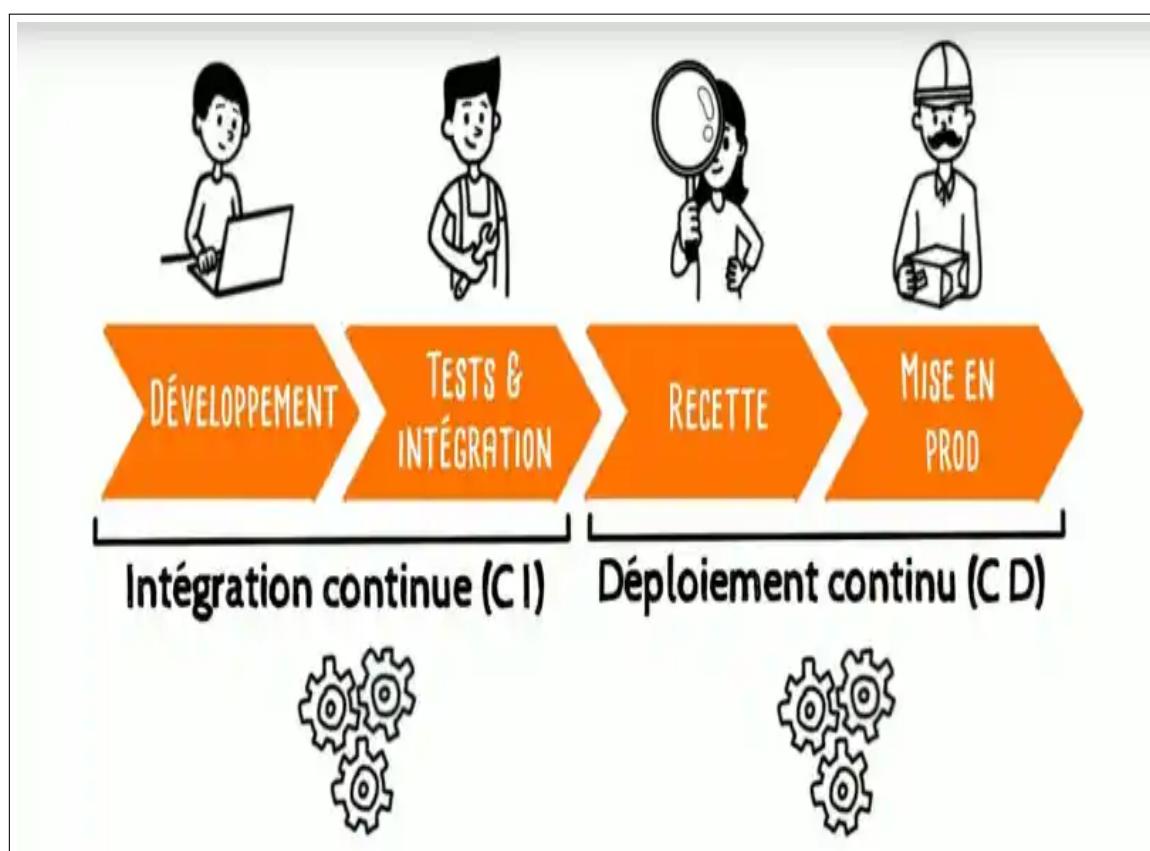


FIGURE 3.7 – Diagramme de déploiement global

3.3 Architecture technologique de la solution

Dans cette partie nous allons présenter l'architecture globale de notre projet.(voir figure 4.8)

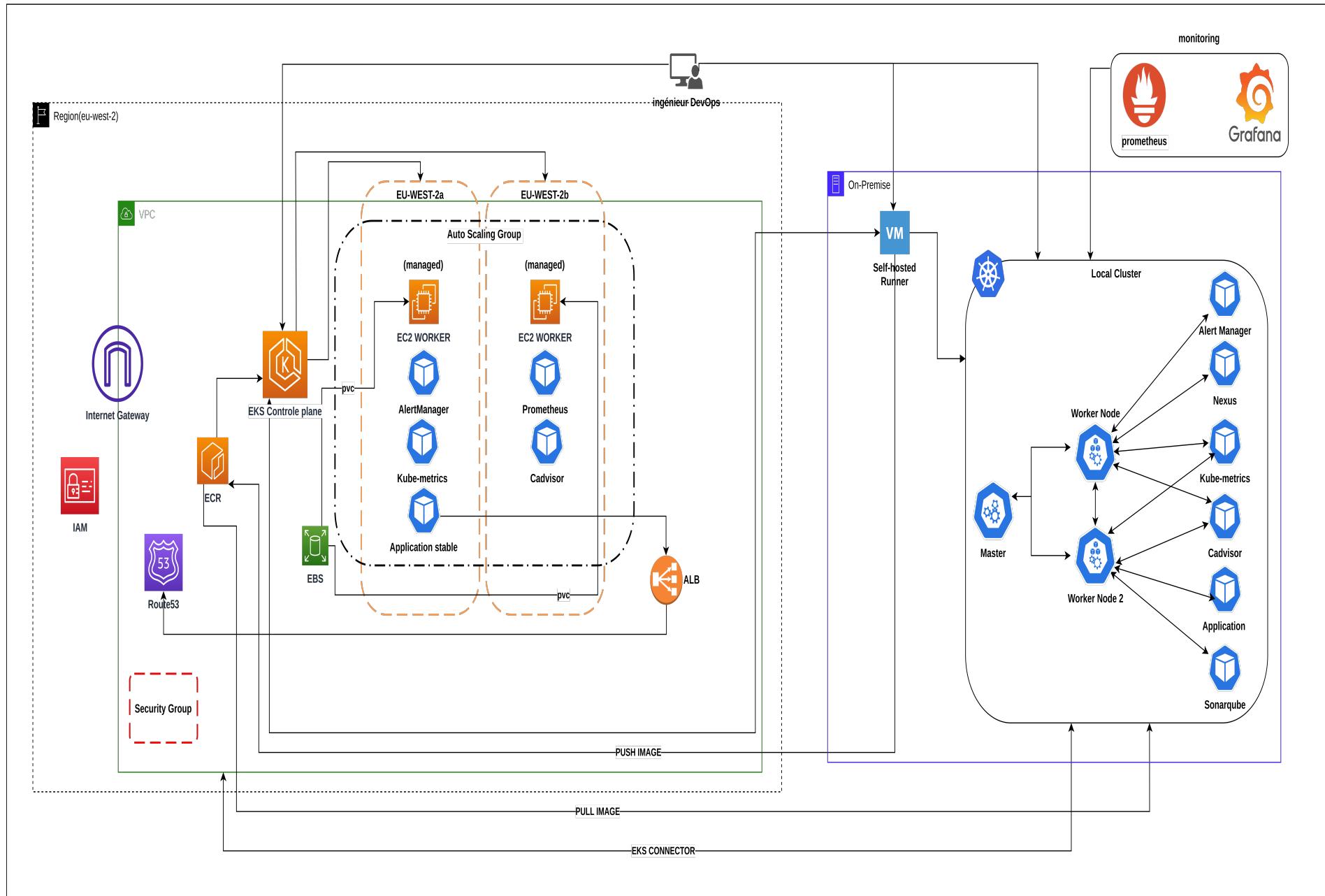


FIGURE 3.8 – Architecture global

Pour plus de clarté, nous allons diviser notre architecture globale en deux parties.

–Partie local :

La figure (figure : 4.9) représente les ressources utiliser localement dans le projet.

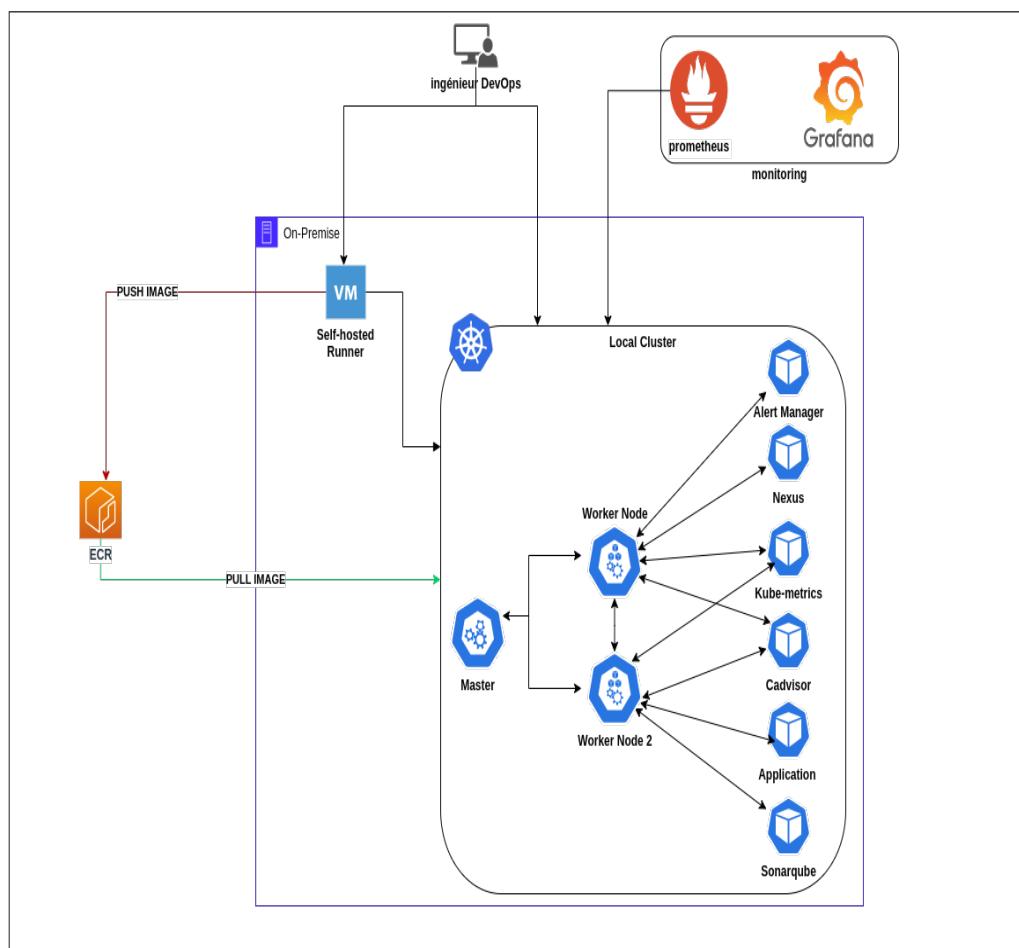


FIGURE 3.9 – Partie local

- Le cluster local dans l'architecture globale fonctionne comme un environnement dédié pour les tests et le développement. Il est configuré et géré par l'ingénieur devops, qui dispose des droits et des permissions nécessaires. Le cluster local est composé de plusieurs composants clés qui facilitent le développement efficace et tester les flux de travail.

- Dans le cluster local, plusieurs nœuds ou machines virtuelles sont fournis pour créer un environnement similaire à la configuration de production. Ces nœuds exécutent des conteneurs ou d'autres outils de test, permettant possible de simuler le comportement et les performances de l'environnement de production localement. L'ingénieur de devons supervise la configuration et la gestion du cluster local, en assurant sa disponibilité et sa stabilité

pour les activités de test et de développement.

- Pour surveiller le cluster local, l'ingénieur devops déploie Grafana et Prometheus, qui fournissent des informations précieuses sur les performances et la santé du cluster. Ces outils de surveillance permettent de suivre l'utilisation des ressources, de détecter et de résoudre les problèmes rapidement, ce qui permet l'amélioration continue et l'optimisation de l'environnement de test local.

- L'ingénieur devops configure également un exécuteur local pour GitHub Actions. Cet exécuteur permet de déclencher des pipelines. Dans le cas de la branche principale, un pipeline démarre et les actions de déploiement nécessaires sont exécutées sur le cluster local. Cela permet l'ingénieur devops de valider les changements dans un environnement qui ressemble beaucoup à la configuration de production, assurant robustesse et fiabilité.

- L'ingénieur de devops configure l'environnement pour extraire les images de conteneurs d'un dépôt ECR privé, qui stocker et gérer les images de conteneurs. Les images du dépôt sont séparées par des versions telles que 1.0.0 , 0.0.1 pour les tests.

–Partie AWS :

La figure ci-dessous représente les ressources cloud utiliser dans notre projet.

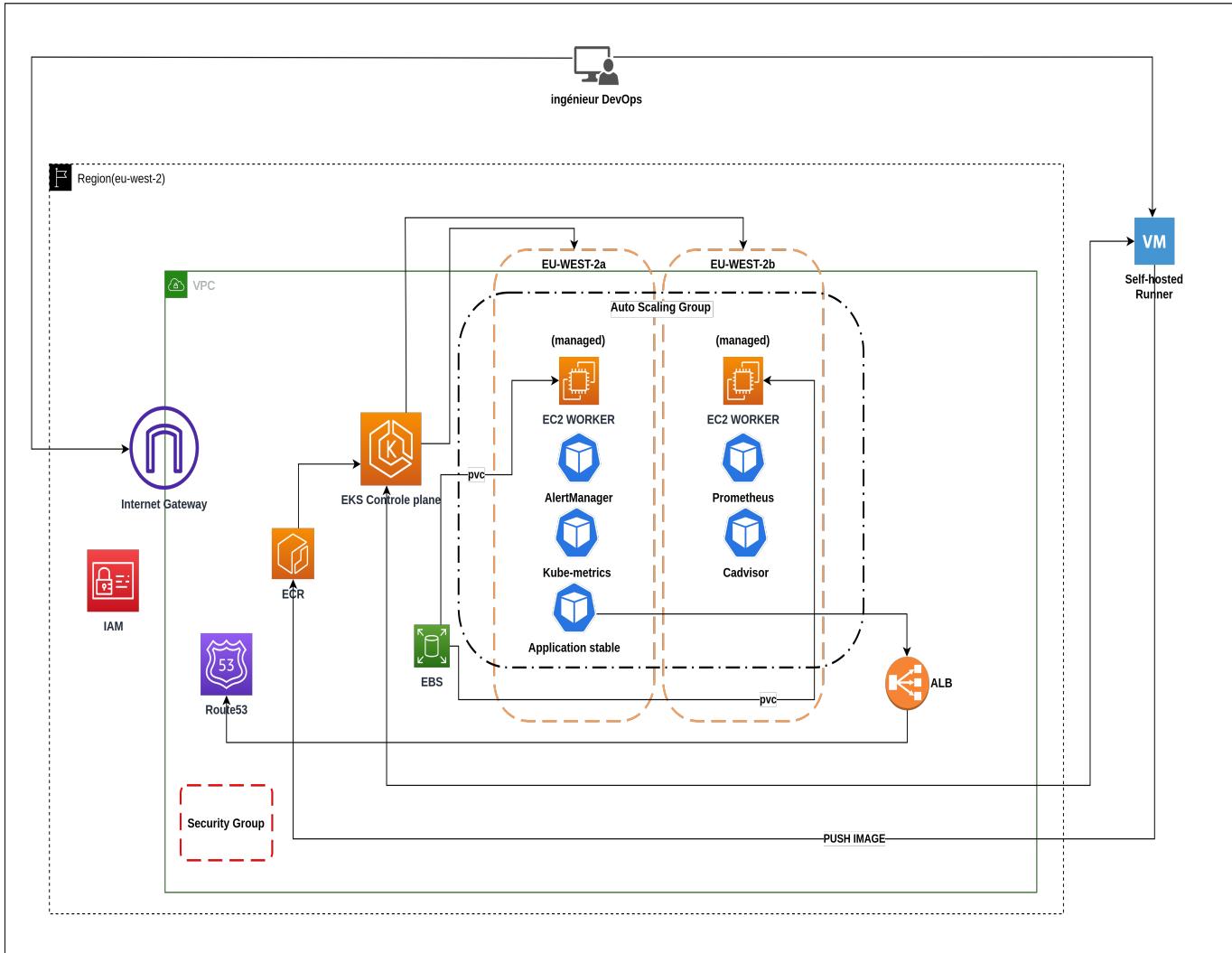


FIGURE 3.10 – Partie AWS

- Le cluster EKS se compose de nœuds de travail, qui sont des instances EC2 qui exécutent les conteneurs contenant la charge de travail de l'application. L'ingénieur de devops assure l'approvisionnement et la mise à l'échelle appropriés des nœuds de travailleurs en fonction des exigences de l'application et de la charge de travail. Cela implique de configurer des politiques de mise à l'échelle automatique pour ajuster automatiquement le nombre de nœuds de travail afin de gérer efficacement les différents charges de travail.

- Pour gérer le trafic entrant et le distribuer à travers les conteneurs du cluster EKS, un Load balancer est lancé pour aider à atteindre une haute disponibilité et une tolérance aux pannes en transférer les demandes vers les conteneurs, assurant ainsi une performance et une fiabilité optimales de l'application.

- Lorsque la branche EKS est déclenchée par des développeurs leur code source, un pipeline dédié est lancé. Pour exécuter les actions de déploiement nécessaires sur le cluster EKS, un exé-

cuteur configuré par l'ingénieur devops utilise son contexte et ses autorisations pour interagir efficacement avec le cluster. Cela permet un déploiement continu de l'application dans l'environnement de production hébergé sur le cluster EKS.

• La surveillance du cluster EKS est également une responsabilité de l'ingénieur de devops. Prometheus installer sur EKS, fournissent des informations complètes sur les performances, l'utilisation des ressources et les paramètres de santé du cluster. Ce outil permettant à l'ingénieur de détecter et de résoudre rapidement les problèmes potentiels, assurant le bon fonctionnement de l'environnement de production.

3.4 Conclusion

Dans ce chapitre nous avons décrit les différents diagrammes de conception. Aussi, nous avons décrit la méthodologie de conception utiliser dans notre projet. Puis nous avons présenter l'architecture global qui expliquer la fonctionnement du projet. Dans le chapitre suivante nous passerons à la phase de réalisation du projet.

Chapitre 4

Sprint2 :Configuration de la Pipline.

Introduction

Après avoir vue la configuration de l'architecture cloud aws nous présentent la deuxième sprint, nous commençons par l'objectif de sprint puis l'architecture globale ensuite nous détaillons le diagramme d'utilisation et séquence détaille, Finalement, nous allons exposer la partie réalisation.

4.1 Objectif du Sprint

Ce sprint est à propos de configurer pipeline dont le but est d'améliorer l'efficacité et la qualité du développement logiciel, tout en minimisant les erreurs et les temps d'arrêt.

4.2 Architecture Détaillée

Dans cette partie nous allons présenter l'architecture détaillée correspond au sprint.

4.2.1 Diagramme Cas d'utilisation Détaillée

Pour éliminer toute ambiguïté et éclaircir le cas d'utilisation, nous présentons le cas en détail.

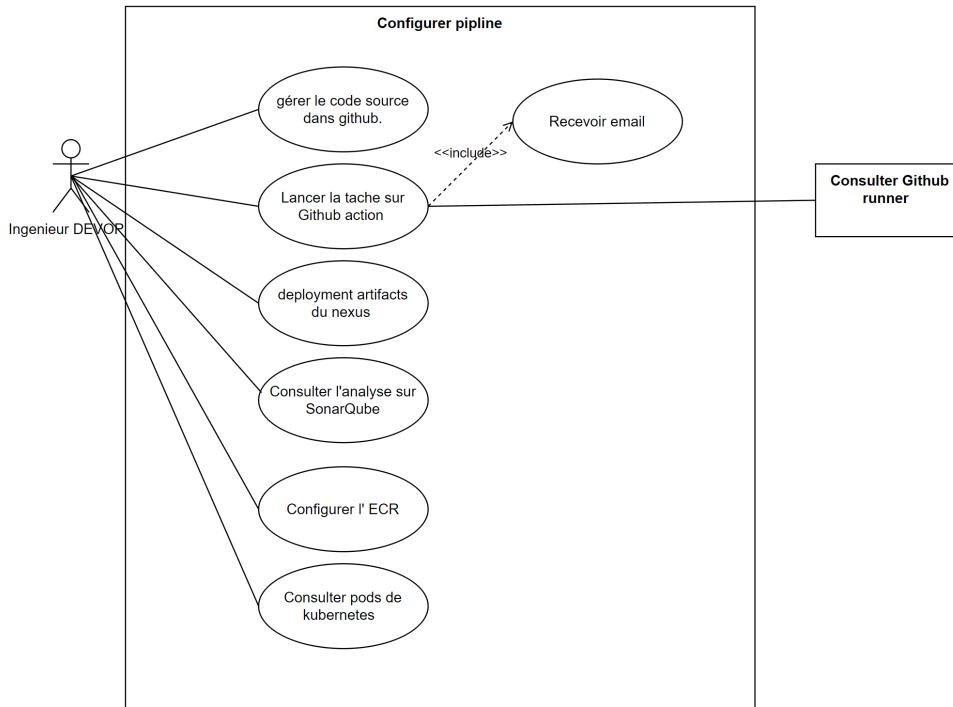


FIGURE 4.1 – cas d'utilisation "Configuration de la Pipeline"

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle du principales fonctionnalités mentionnées ci-dessus :

Titre	Configuration de la Pipeline
Acteur	Ingénieur DevOps
Description	L'ingénieur DevOps gérait la configuration de la Pipeline et la création du cluster .
Pré conditions	Comprendre le processus de déploiement de l' application, aussi disposer d'une infrastructure appropriée pour prendre en charge le déploiement , ainsi identifier les étapes du processus de déploiement, les dépendances et les outils nécessaires. et pour automatiser les étapes du pipeline en utilisant des scripts et des configurations.
Post conditions	une architecture configurait correctement.
Scénario nominal	Déploiement continu est réussite et la configuration réussite.
Scénario alternatif	problème de compatibilité avec l'infrastructure existante ou la configuration n'est pas correcte.

TABLE 4.1 – Description de cas d'utilisation

4.2.2 Diagramme de séquence Détailée

Nous détaillons ci-dessous les interactions générales du système

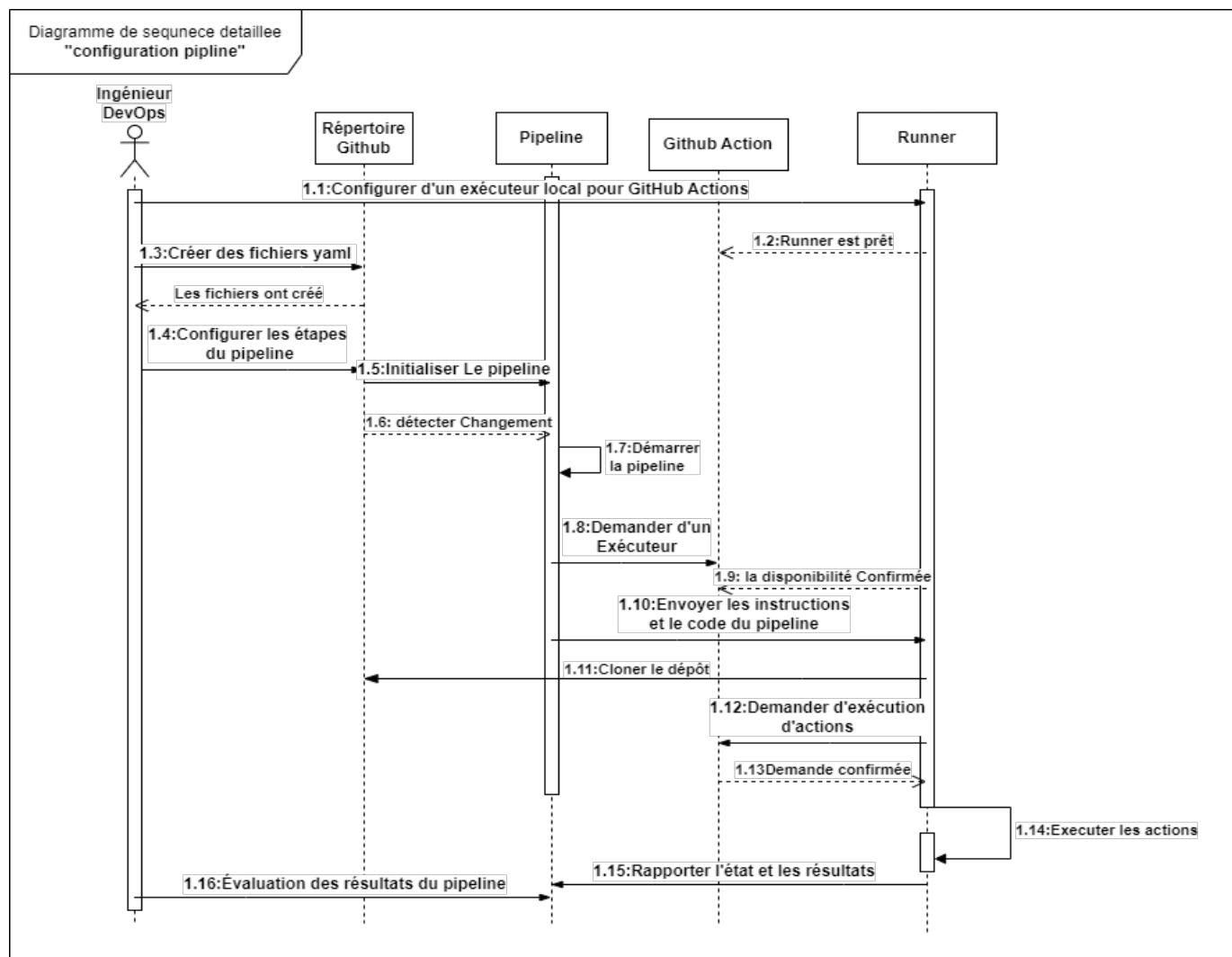


FIGURE 4.2 – diagramme de séquence "Configuration de la Pipeline"

Chapitre 5

Sprint3 : Déploiement de l'application sur le cluster.

Introduction

Après avoir déduit la configuration du pipeline nous présentent la troisième sprint, nous commençons par l'objectif de sprint puis l'architecture globale ensuite nous détaillons le diagramme d'utilisation et séquence détaillé, Finalement, nous allons exposer la partie réalisation

5.1 Objectif du Sprint

l'objectif du déploiement de l'application sur un cluster est d'assurer une disponibilité élevée, une évolutivité, une tolérance aux pannes, une gestion efficace des ressources, une sécurité, une facilité de déploiement et une surveillance pour fournir une expérience utilisateur optimale dans un environnement de cluster.

5.2 Architecture Détaillée

Dans cette partie nous allons présenter l'architecture détaillée correspond au sprint.

5.2.1 Diagramme Cas d'utilisation Détaillée

Afin d'éliminer toute ambiguïté et de clarifier le cas d'utilisation, nous présentons le cas de façon détaillée.

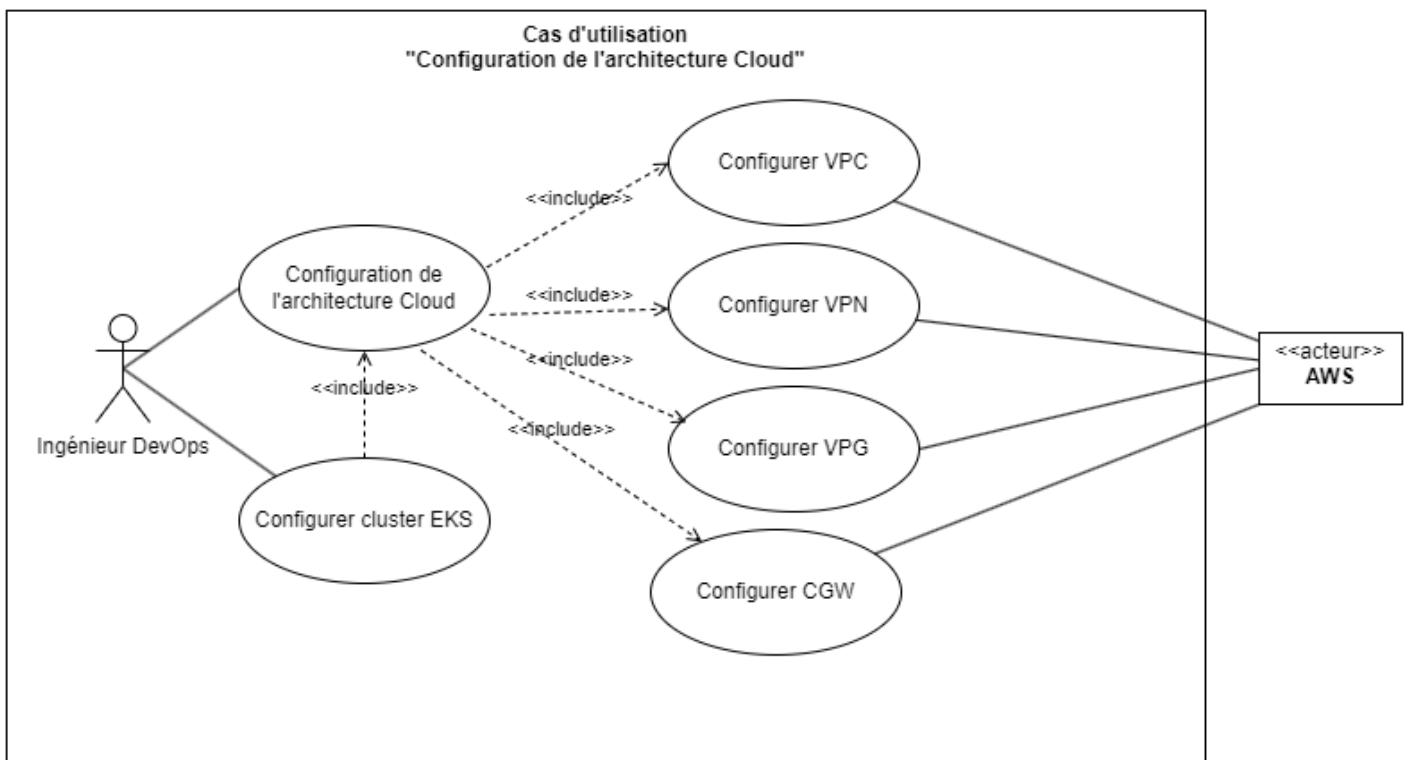


FIGURE 5.1 – cas d'utilisation "Déploiement de l'application sur le cluster."

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle du principales fonctionnalités mentionnées ci-dessus :

Titre	Déploiement de l'application sur le cluster.
Acteur	Ingénieur DevOps
Description	L'ingénieur DevOps gérait transférer l'application et ses composants dans un environnement de cluster, dans lequel elle peut être exécutée et gérée de façon efficace et évolutive .
Pré conditions	La configuration de cluster ainsi la configuration de l'application et la gestion des dépendances .
Post conditions	l'application deploye sur cluster et bien fonctionne .
Scénario nominal	l'application est déployée avec succès sur le cluster, fonctionne de manière stable et répond aux exigences spécifiées.
Scénario alternatif	problèmes lors de la configuration du cluster ou problèmes de configuration de l'application , de réseau ou de ressources insuffisantes.

TABLE 5.1 – Description de cas d'utilisation

Bibliographie

- [1] Microsoft Azure. Définition de devops. [Online ; accessed 01-mars-2023].
- [2] Microsoft Azure. Définition de cloud. [Online ; accessed 01-mars-2023].
- [3] Oracle. Définition saas. [Online ; accessed 01-mars-2023].
- [4] Oracle. Définition de paas. [Online ; accessed 01-mars-2023].
- [5] Microsoft Azure. Définition de iaas. [Online ; accessed 01-mars-2023].
- [6] Oracle. Définition de cloud prive. [Online ; accessed 01-mars-2023].
- [7] Google. Définition de public cloud. [Online ; accessed 01-mars-2023].
- [8] Goggle. Définition de cloud hybride. [Online ; accessed 01-mars-2023].
- [9] Microsoft Azure. Azure devops. [Online ; accessed 01-mars-2023].
- [10] Amazon web service. Amazon devops. [Online ; accessed 01-mars-2023].
- [11] Bluelight. Gcp devops. [Online ; accessed 01-mars-2023].
- [12] Wikipedia. Définition d'acteur dans un système. [Online ; accessed 01-mars-2023].
- [13] Wikipedia. Diagramme de séquence. [Online ; accessed 01-mars-2023].
- [14] Wikipedia. Diagramme de classe. [Online ; accessed 01-mars-2023].
- [15] Wikepedia. Diagramme de déploiement. [Online ; accessed 01-mars-2023].