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, Pipline, 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

Table des figures

Liste des tableaux

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. On finit par une présentation de la méthodologie que nous allons adopter pour le développement de notre projet.

1.1 Cadre général du projet

1.1.1 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 platefores 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 le 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).

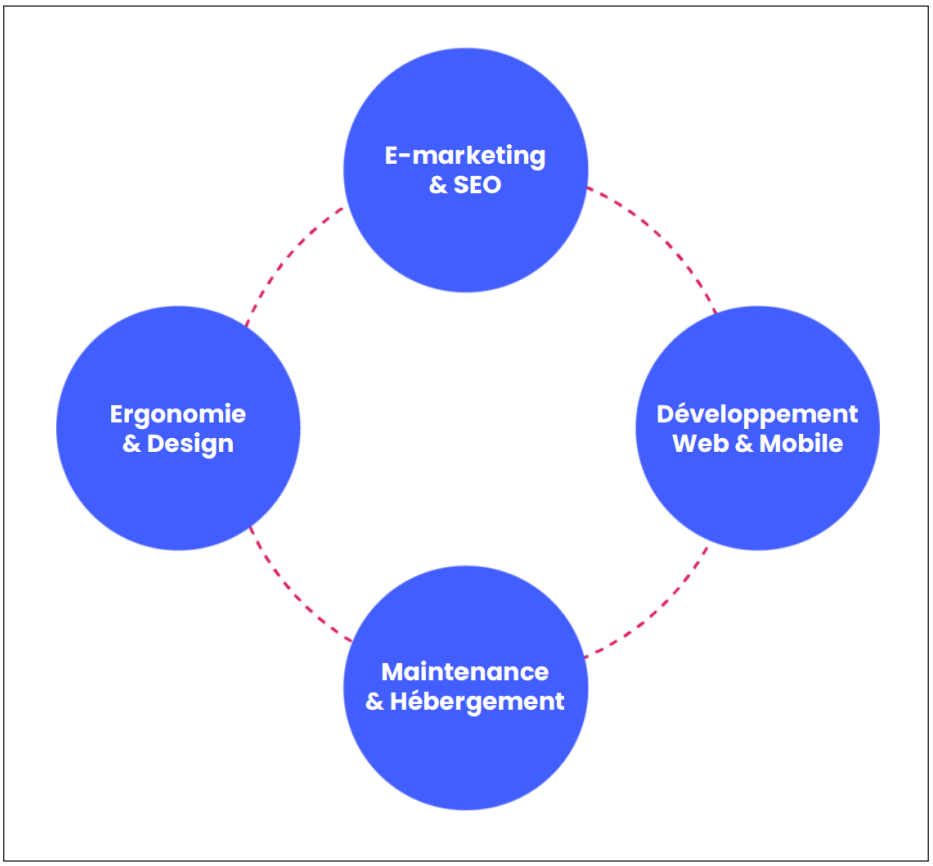

FIGURE 1.1 – différents domaines de mobelite

1.1.2 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 diffèrentes 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 .

1.1.3 Notion de base

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

1.1.3.1 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).

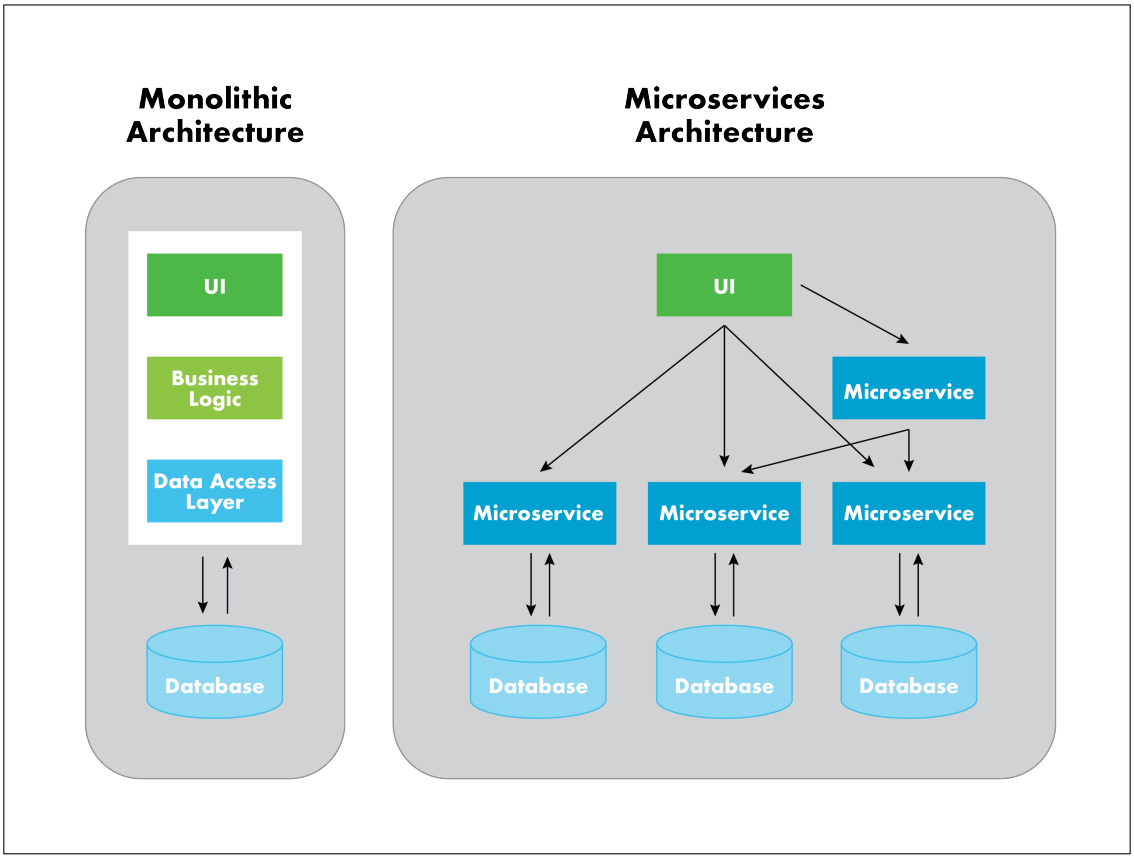

FIGURE 1.2 – Architecture monolithique vs Micro services

1.1.3.2 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).

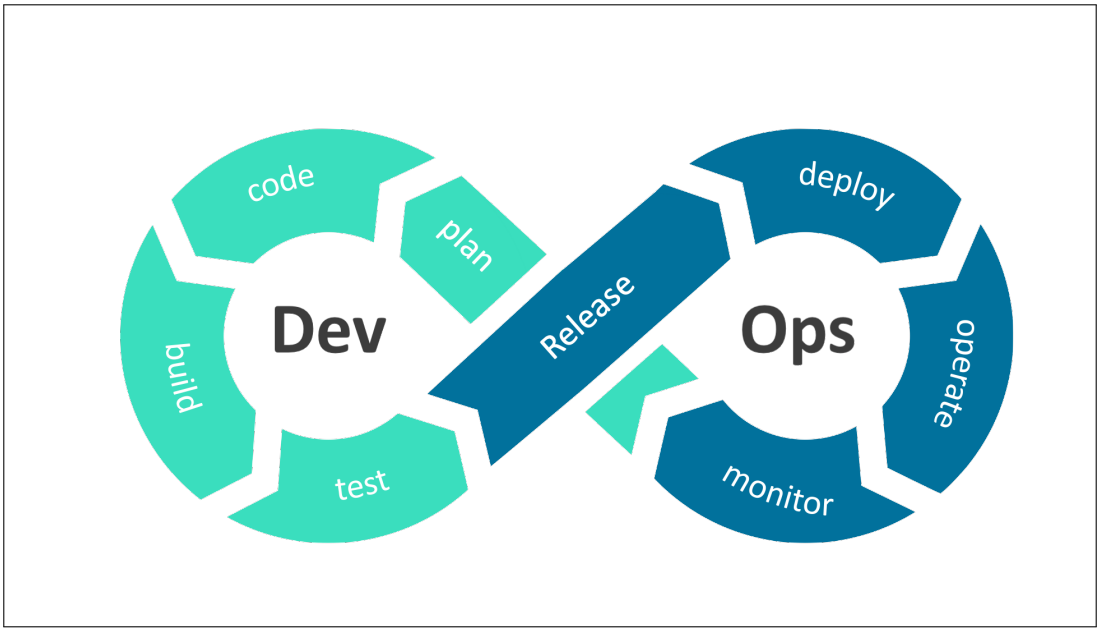

FIGURE 1.3 – Cycle de vie DevOps

1.1.3.3 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]

1.1.3.3.1 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].

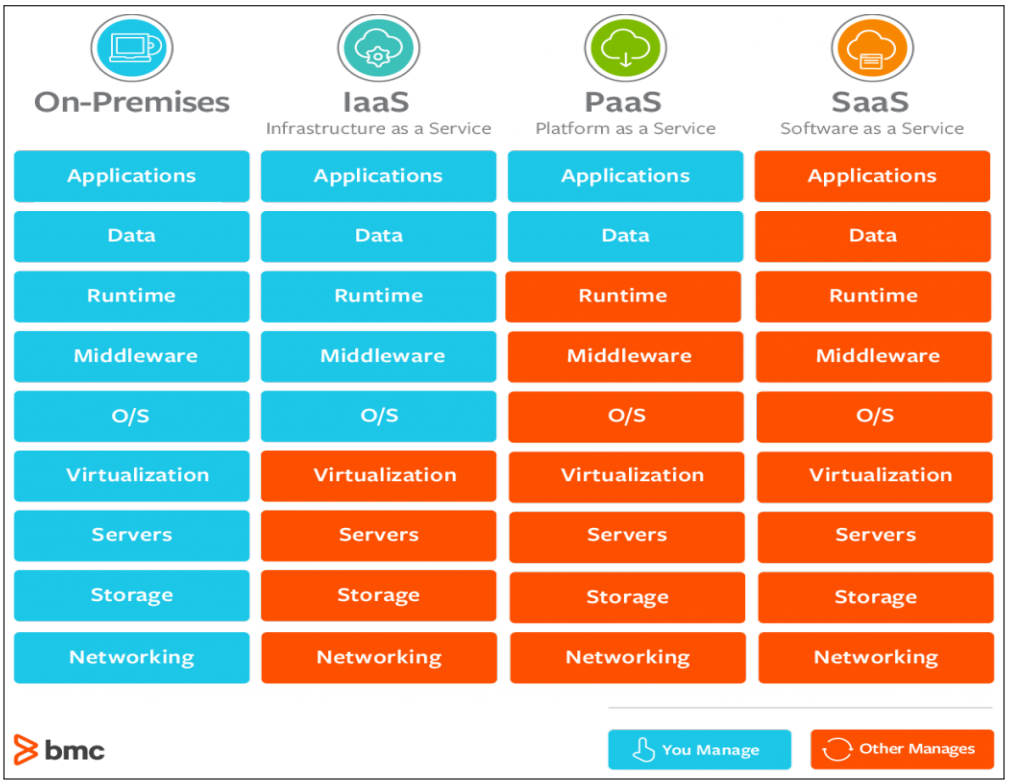

FIGURE 1.4 – Différents architecture entre SaaS ,PaaS et IaaS

1.1.3.3.2 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 organisations 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).

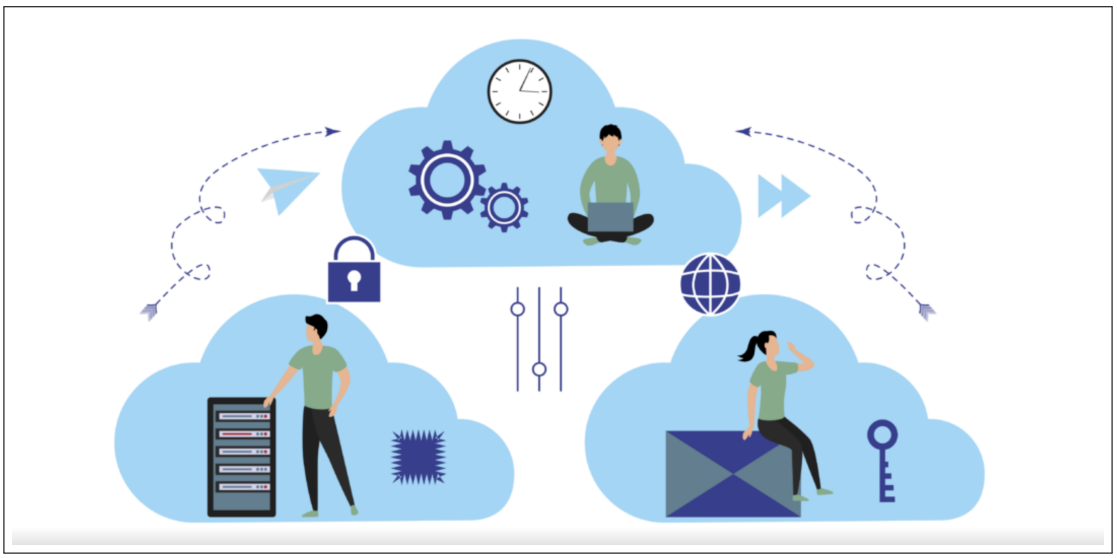

FIGURE 1.5 – Différents type du cloud

1.1.4 Méthodologie de développement

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.

1.1.4.1 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.

1.1.4.1.1 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 exigenes 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.

1.1.4.1.2 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.

1.1.4.1.3 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 objéctif 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.

1.1.4.1.4 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.

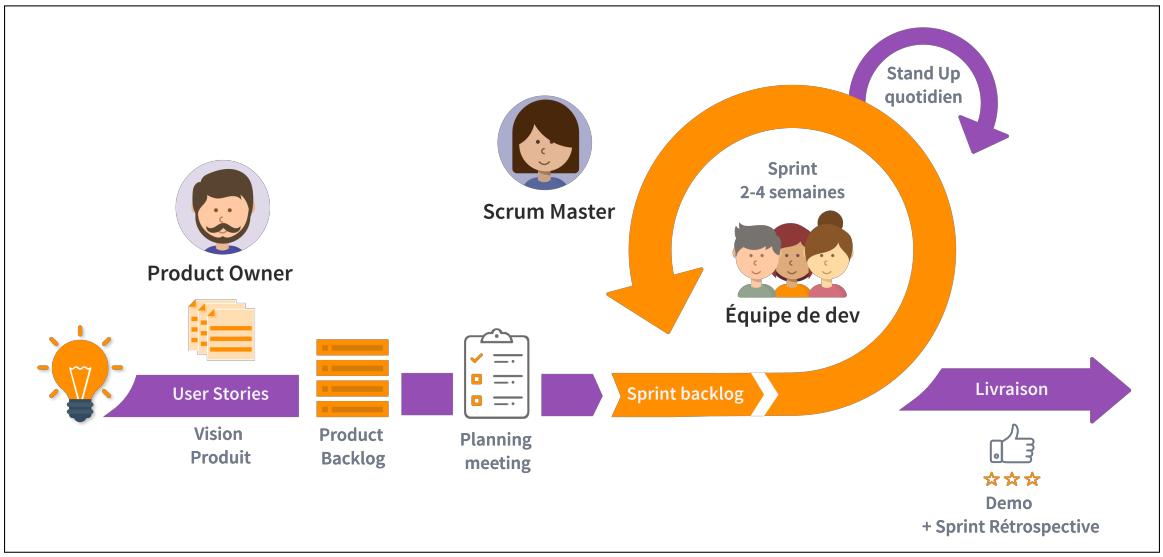

FIGURE 1.6 – Processus de méthode Scrum

1.1.5 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) :

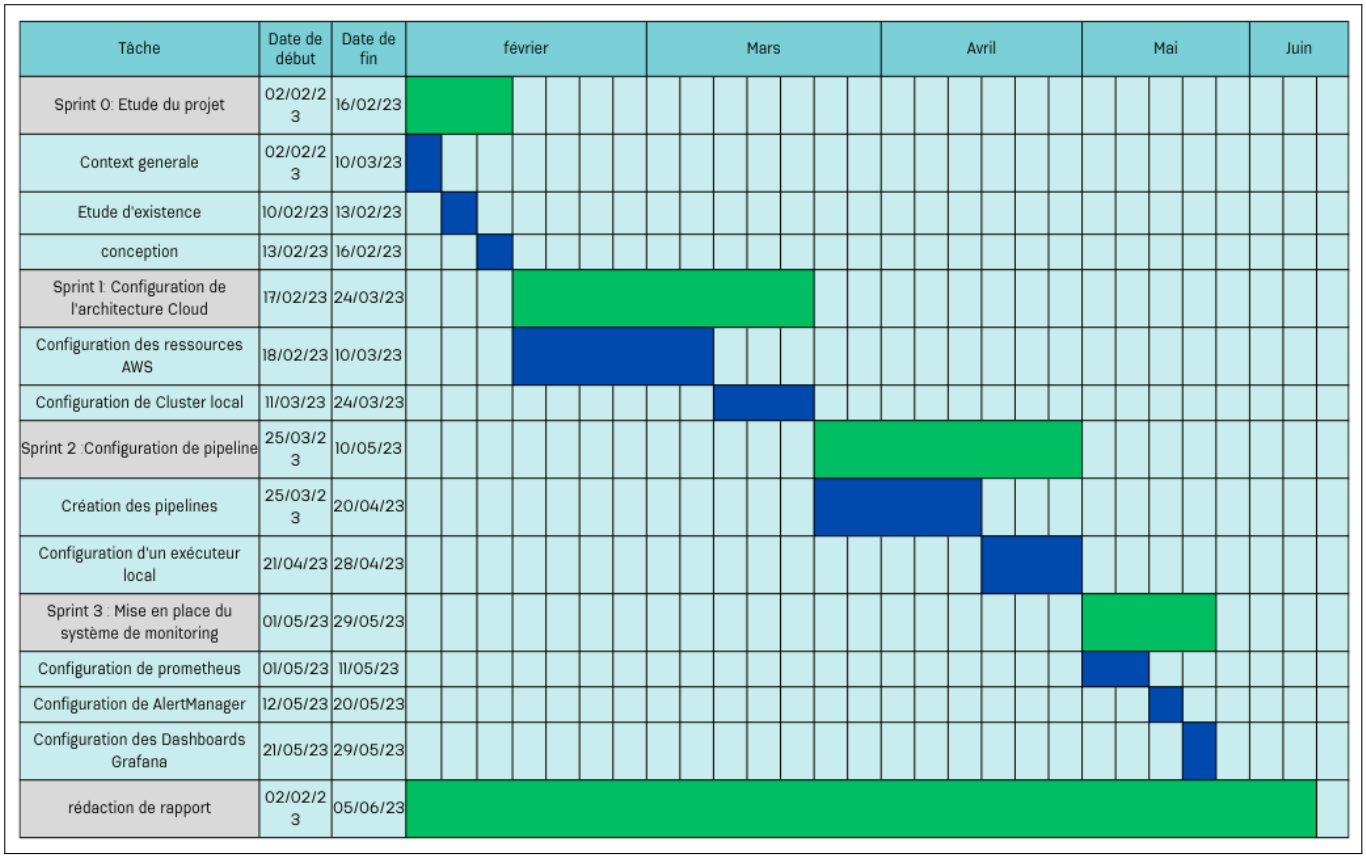

FIGURE 1.7 – Diagramme de Gantt

1.1.6 Conclusion

Dans ce chapitre nous avons présenté la société d'accueil, la problématique et la méthodologie de développement. Dans le prochain chapitre, nous allons faire une étude de l'existant à travers une comparaison des solutions sur le marché afin de proposer notre solution.

Chapitre 2

Etude de l'existant

Introduction

L'objectif de ce deuxième chapitre est la présentation des solutions existantes sur le marché avec une étude comparative et présentation de la solution proposée.

2.1 Les solutions existantes

L'entreprises a des besoins sur mesure. C'est pourqouien 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 developper notre propre solution. Voici une présentation de certaines solutions existantes :

2.1.1 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. Démarrez 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.(Voir figure 2.1)[9] Elle a des avantages mais aussi des inconvenients :

-Avantage : Azure offre une haute disponibilité,sécurité solide et offre de bonnes options d'évolutivité.

-Inconvenient : oblige à mettre tous vos oeufs dans le même panier et la facilité d'accès peut être problématique pour certaines entreprises.

-Collaboration tools :

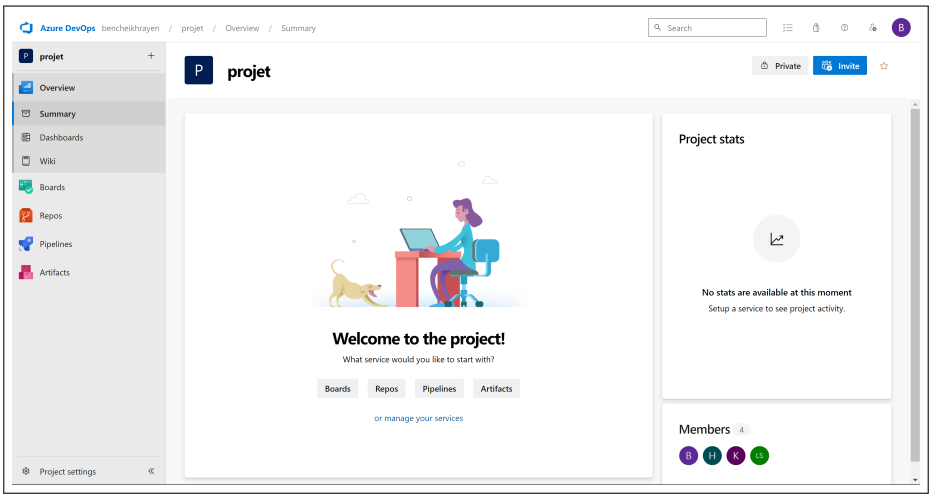

FIGURE 2.1 – Azure DevOps

2.1.2 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(Voir figure 2.2).Cependant, comme toute autre technologie, AWS présente des avantages et des inconvénients : [10]

-Avantage : 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

-Inconvenient : 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.

FIGURE 2.2 – Amazon Web Services

2.1.3 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))(Voir figure 2.3).Ainsi GCP a des points forts et des points faibes :[11]
-Points forts : GCP offre également des services de développement et d'intégration d'applications aussi une bonne Sécurité .

-Points faibles : certaines intégrations peuvent être plus étroites avec les services et les outils de Google par rapport aux solutions tierces.Aussi ,la configuration initiale de GCP est complexe en raison de la vaste gamme de services et d'options disponibles.

FIGURE 2.3 – Google Cloud Platform

2.2 Critique des solutions existantes

Nous allons comparer les solutions présentées ci-dessus selon plusieur qu'on definire par rapport aux besoins de l'entreprise.

	Azure Devops	Amazon Web Service DevOps	Google Cloud Platform DevOps
C 1	X		
C 2		X	X
C 3		X	
C 4			

En se basant sur le tabeau comparatif ci-dessous,on constate que AWS est plus fiable pour notre sujet mais il n'est pas compatible avec notre projet.

C1 - La confidentialité des données d'application : La solution doit privilégier la confidentialité des données puisque certains clients préfèrent leur application exécutée sur le serveur local de la société.

C2 - Intégration de solutions open source : 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.

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

2.3 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 certain 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.

2.4 Conclusion

Dans ce chapitre, nous avons réalisé une étude de quelques solutions existantes, puis nous avons réalisé une comparaaison de l'existant pour connaître les solutions compatibles avec notre sujet. Après cela, nous avons présente la solution proposée par mobelite.

Chapitre 3

Planification de projet

Introduction

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

3.1 Acteur

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 la configurer son infrastructure.

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

3.2 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 d'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.

3.3 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 propriété 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 :

–Sécurité : 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.

3.4 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 3.1) :

id	Theme	Acteur	Description	Priorité
id 1	gérer code source	Développeur	il permet de déposer le code ou le récupérer	élevée
id2	Suivre build d'application	Développeur	il peut voir la compilation de l'application	élevée
id3	analyser qualité de code	Ingénieur DEVOPS	il 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 pipline	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'etat du cluster	Ingénieur DEVOPS	surveiller 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 deploiement	Ingénieur DEVOPS	assurer que les applications sont déployées de manière cohérente et fiable	élevée
id 10	acces a l'application deployée	Client	configurer pour s'assurer qu'elle est accessible uniquement par les utilisateurs autorisés.	élevée

TABLE 3.1 – Tableau du Product Back-log

3.5 Conception de système

Dnas ce qui suit,nous allons présenter les différent diagrammes de conception. //[0.2cm]

3.5.1 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 3.2)

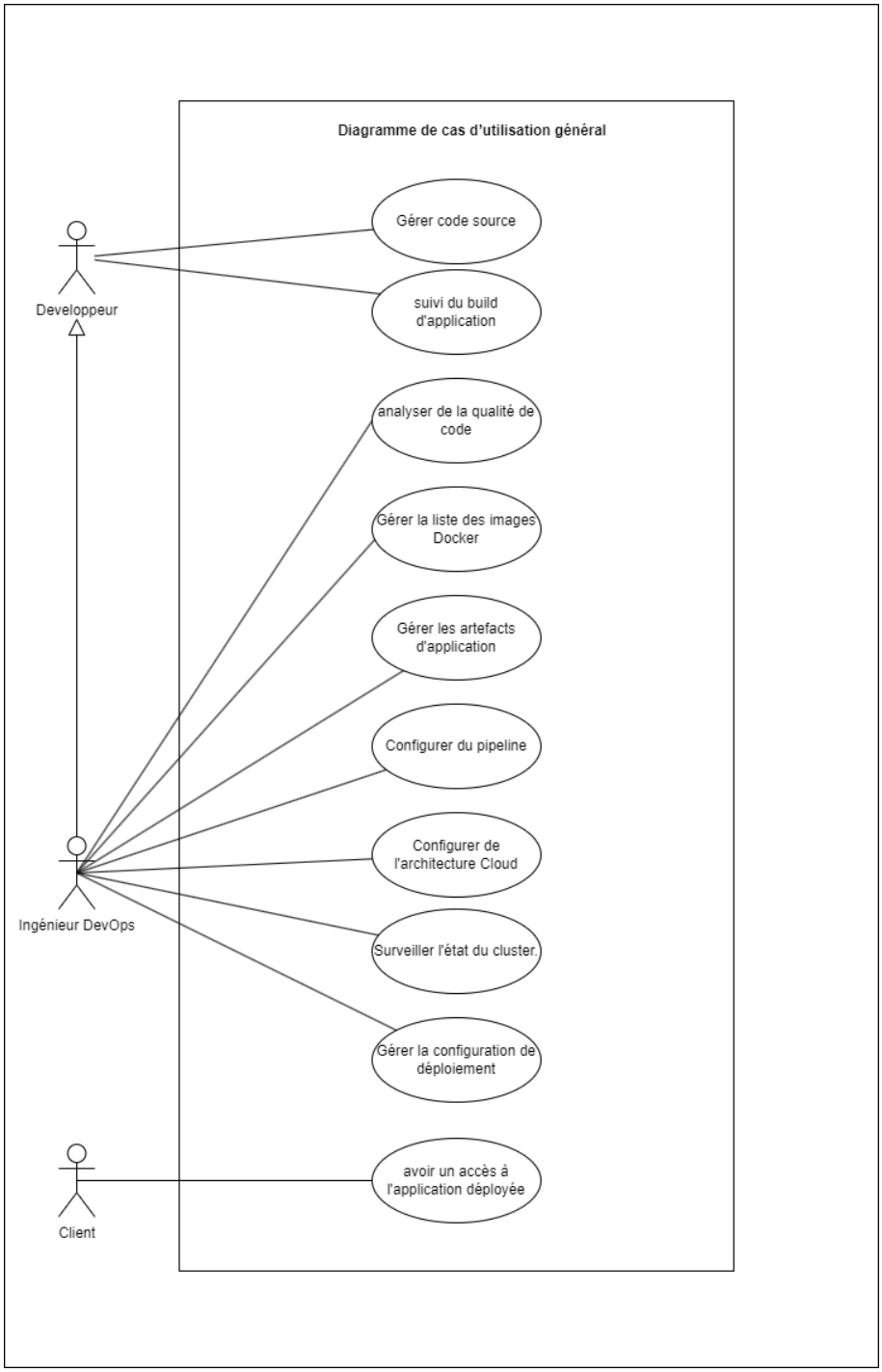

FIGURE 3.1 – Diagramme de cas d'utilisation Globale

3.5.2 Raffinement des cas d'utilisation

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

3.5.2.1 Cas d'utilisation "Gérer code source"

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

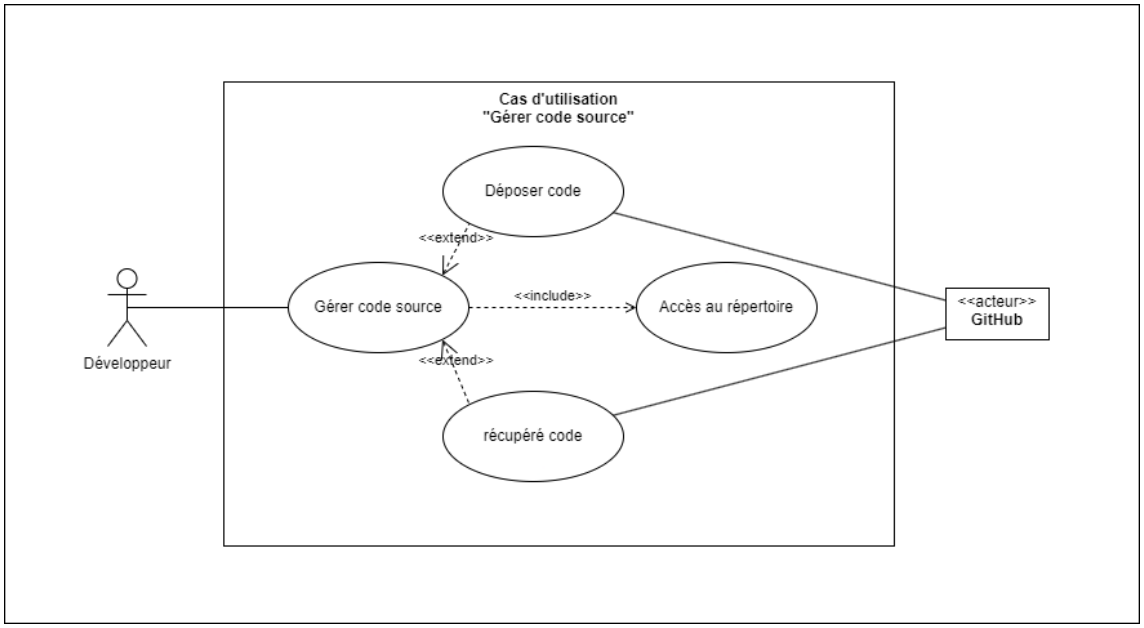

FIGURE 3.2 – Cas d'utilisation :Gérer code source

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle du principales fonctionnalités mentionnées cidessus (Voir tableau 3.2) :

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 récuperer 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 3.2 – Description de cas d'utilisation

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

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

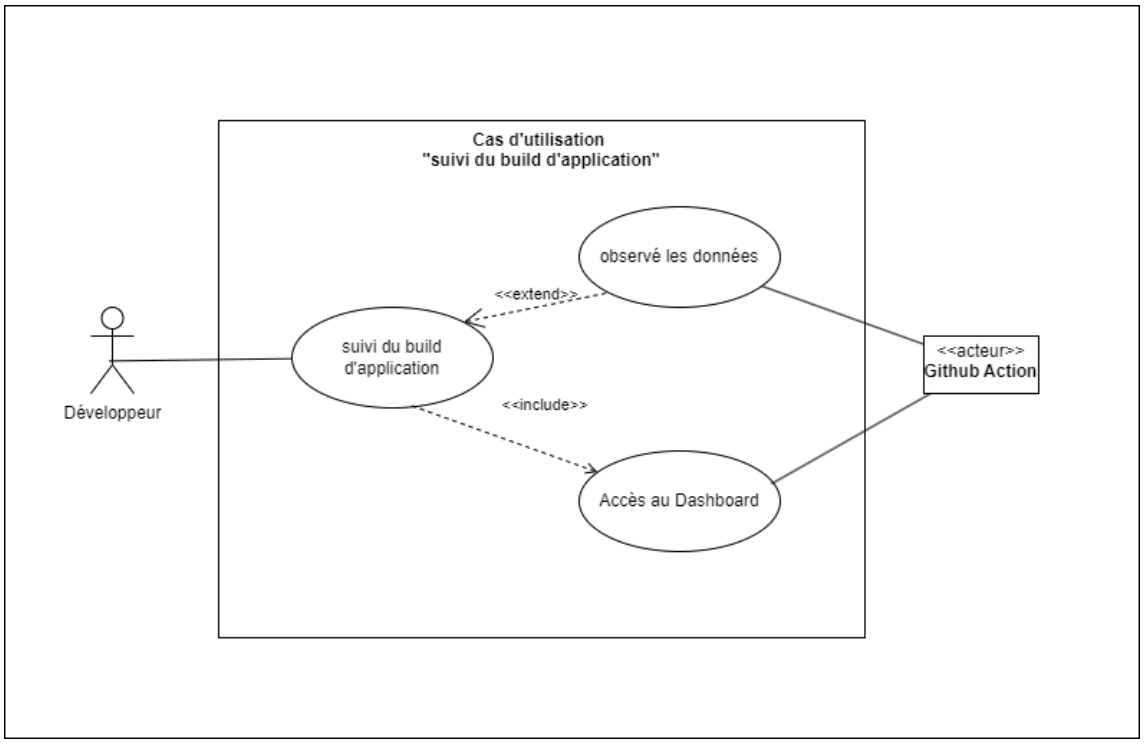

FIGURE 3.3 – Cas d'utilisation :Suivie la build d'application

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle des principales fonctionnalités mentionnées ci-dessus(voir tableau 3.3) :

Titre	Suivi du build de l'application
Acteur	Développeur
Description	le développeur peut suive 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- Avait un accès au Dashbord 2- Suivre du build d'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 3.3 – Description de cas d'utilisation

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

Dans cette section, nous décrirons de façon détaillée ce cas d'utilisation(voir figure 3.4).

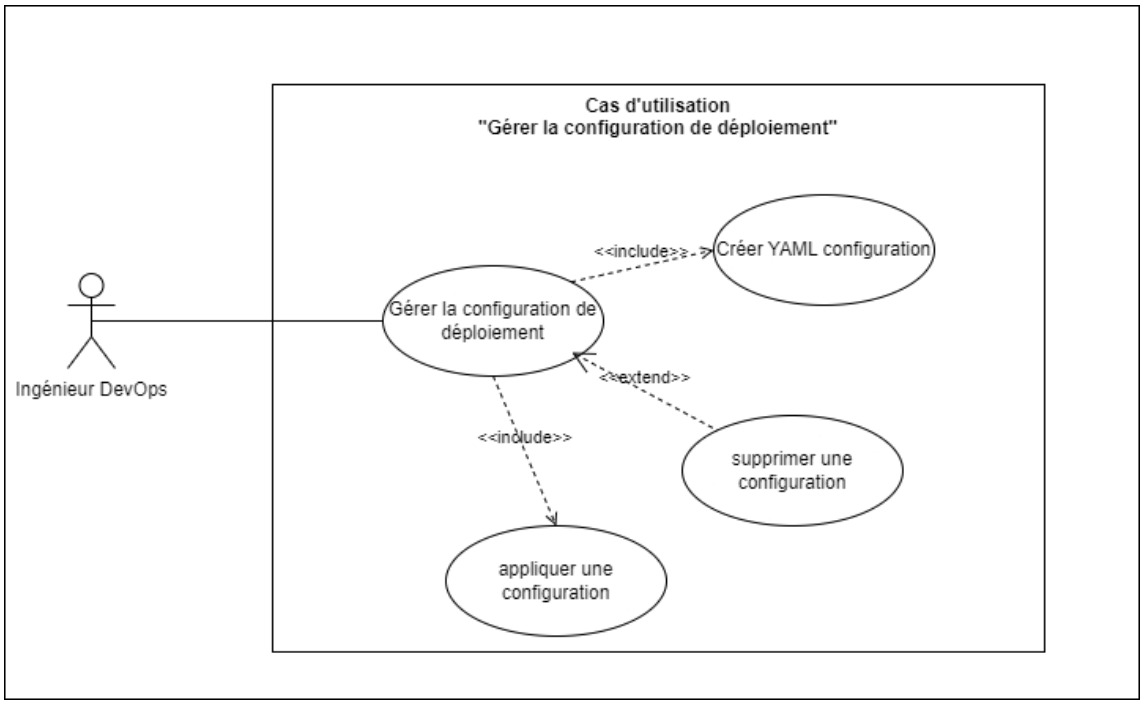

FIGURE 3.4 – Cas d'utilisation :Gérer la configuration de déploiement

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle des principales fonctionnalités mentionnées ci-dessus (voir table 3.4) :

Titre	Gérer la configuration de déploiement
Acteur	Ingénieur DevOps
Description	L'ingénieur DevOps gére 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 existe.
Scénario nominal	1 - Réagir la configuration du déploiement. 2 - La configuration est lancée dans le contrôle plane d'EKS.
Scénario alternatif	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 3.4 – Description de cas d'utilisation

3.5.2.4 Cas utilisation"Gérer liste des images Docker"

Dans cette section, nous décrirons de façon détaillée le cas d'utilisation (voir figure 3.5).

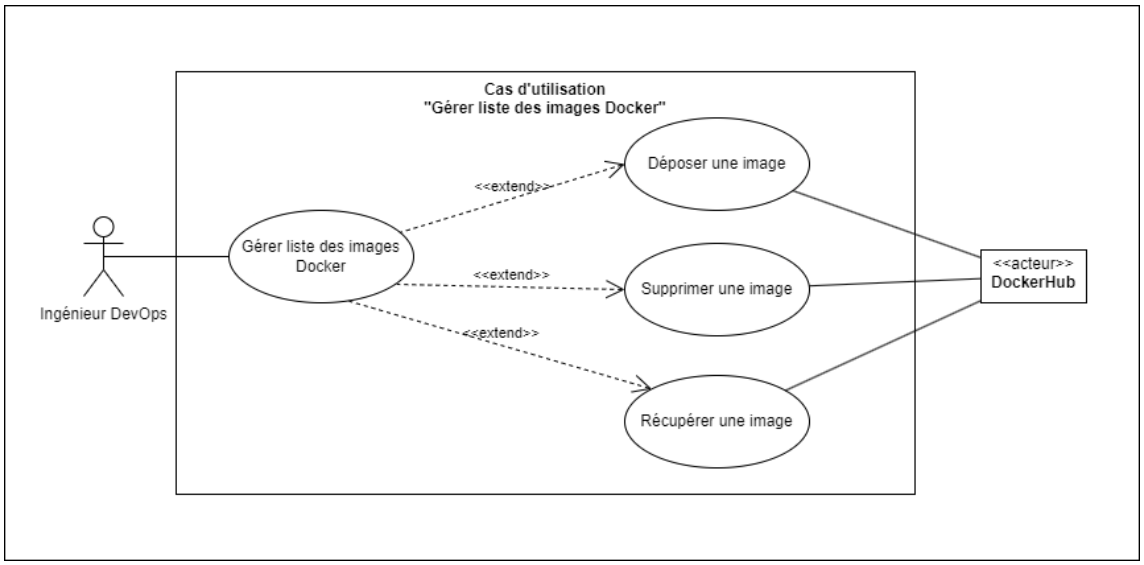

FIGURE 3.5 – Cas d'utilisation :Gérer liste des images Docker

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle des principales fonctionnalités mentionnées ci-dessus (voir table 3.5) :

Titre	Gérer liste des images Docker
Acteur	Ingénieur DevOps
Description	L'ingénieur DevOps gére la liste des images Docker
Pré conditions	Connexion au répertoire de l'application dans ECR.
Post conditions	un image modifiée ,supprimée ou ajoutée.
Scénario nominal	1 - Configurer la liste des images Docker . 2 - Déposer ou récuperer 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 de déposer ou récuperer l'image Docker . 3 - Retour à l'étape 1 du Scénario nominal.

TABLE 3.5 – Description de cas d'utilisation

3.5.2.5 Cas utilisation"Gérer les artefacts d'application"

Dans cette section, nous décrirons de façon détaillée ce cas d'utilisation (voir figure 3.6).

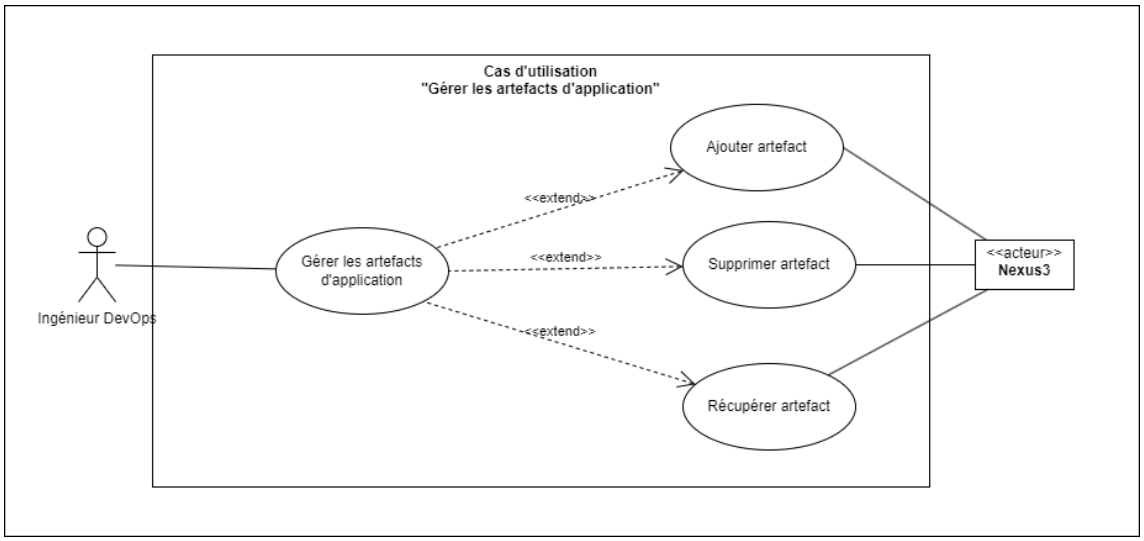

FIGURE 3.6 – Cas d'utilisation :Gérer les artefacts d'application

Pour expliquer le diagramme de cas d'utilisation, nous représentons la description textuelle des principales fonctionnalités mentionnées ci-dessus (voir table 3.6) :

Titre	Gérer les artefacts d'application
Acteur	Ingénieur DevOps
Description	L'ingénieur DevOps gére 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 différente artefacts d'application.
Scénario nominal	1 - Configurer l'artifacts d'application . 2 - la connexion est réussite et la modification dans le répertoire est enregistré 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 3.6 – Description de cas d'utilisation

3.5.3 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 dessous ci-dessous représente le diagramme de classe de notre travail(voir figure 3.7)[13]

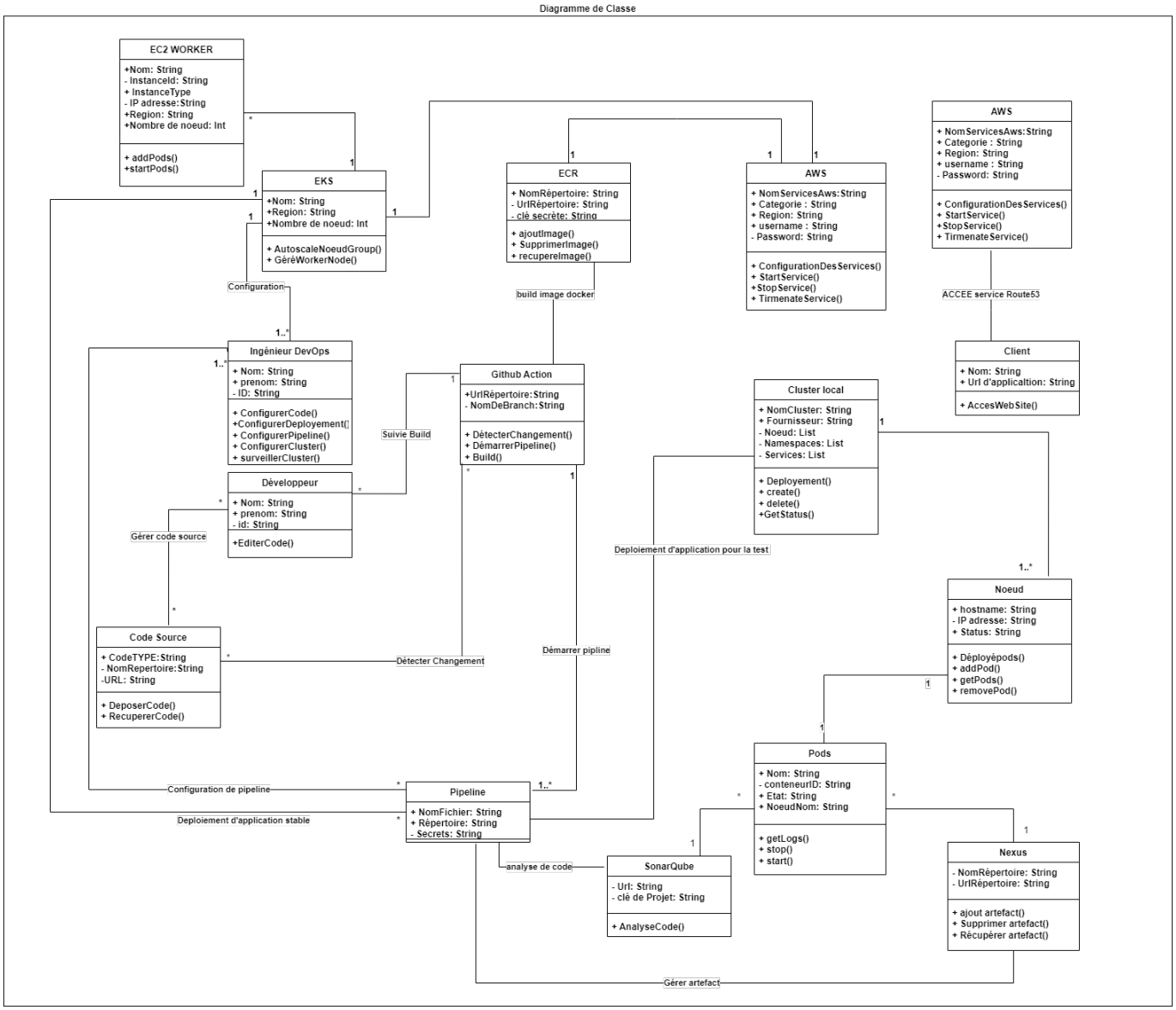

FIGURE 3.7 – Diagramme de classe global

3.5.4 Diagramme de séquence

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique (voir figure 3.8).[14]

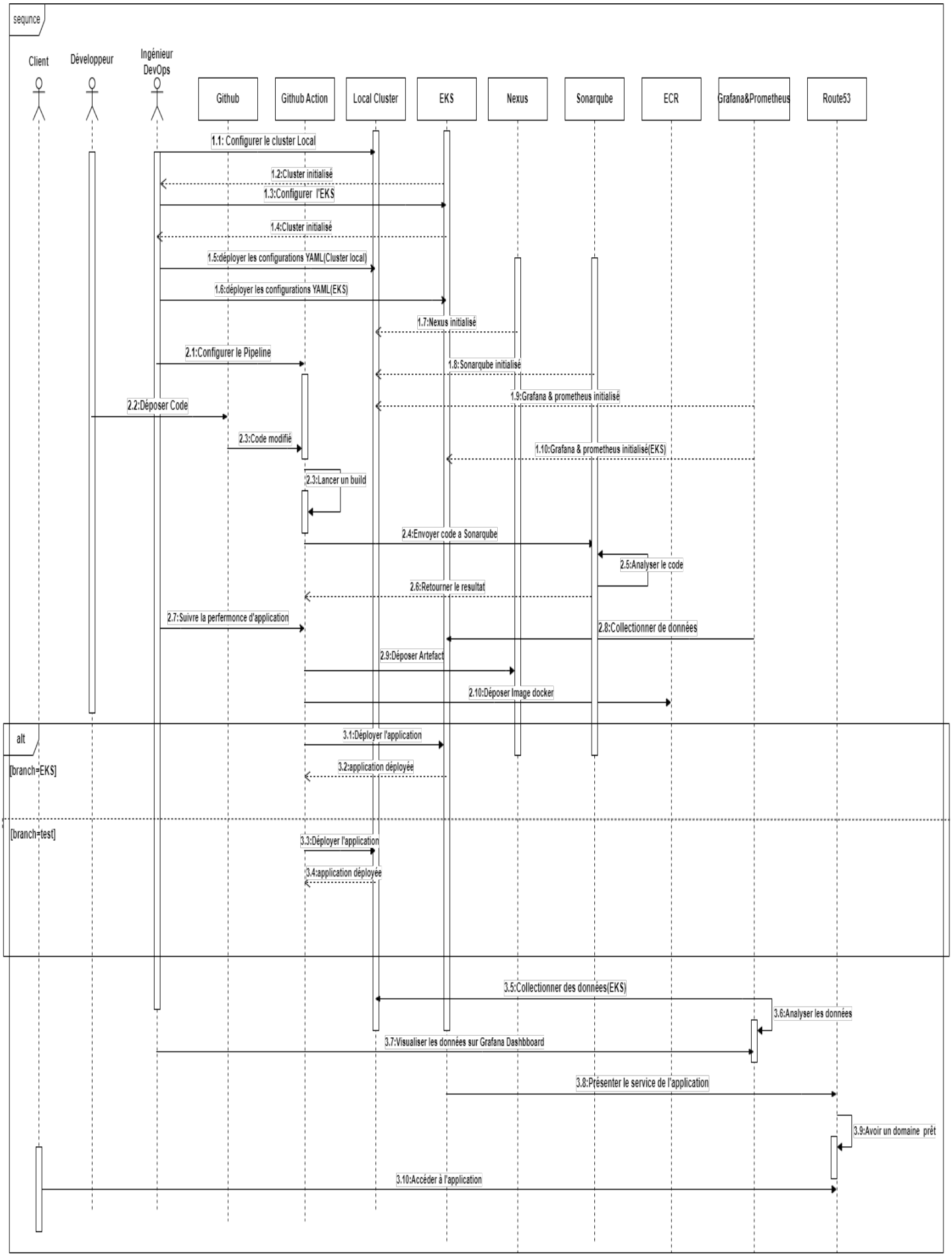

FIGURE 3.8 – Diagramme de séquence global

3.5.5 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 3.9).[15]

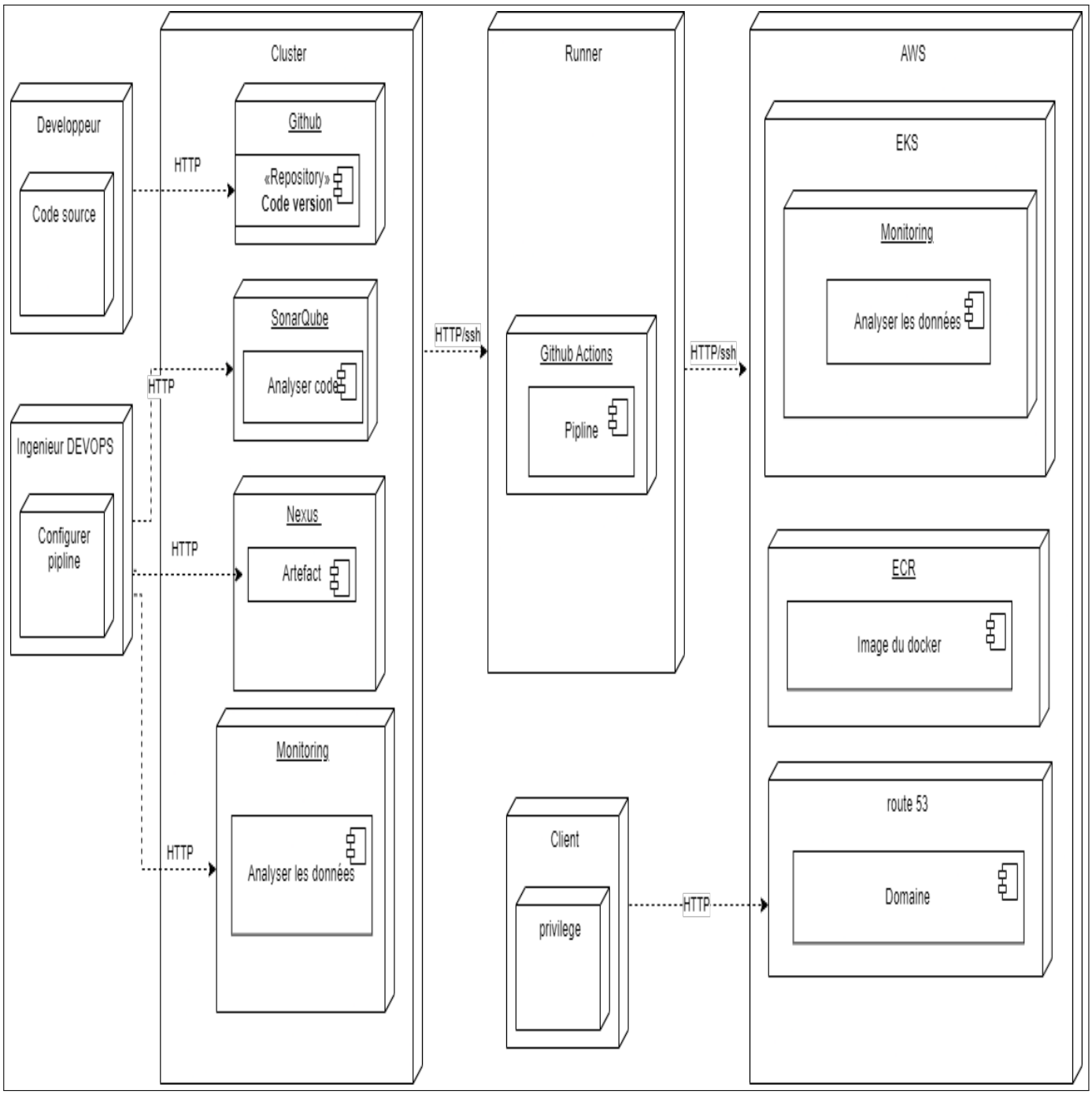

FIGURE 3.9 – Diagramme de déploiement global

3.6 Application de la méthodologie de développement

Dans ce qui suit , nous allons presenter l'application de SCRUM à notre problématique.

3.7 Planification des sprints

Pour une meilleure optimisation du développement du projet nous avons divisé le travail sur 5 Sprints présentées comme suit(voir table 3.7) :

Sprint 0	Objectifs du sprint	Durée
Sprint1	Configuration de l'architecture cloud AWS.	
	Configuration de le cluster EKS.	
Sprint2	Configuration de la Pipeline.	
Sprint3	Déploiement de l'application sur le cluster.	
Sprint4	Configuration du système de la surveillance.	
	Configuration de route53 pour l'application déployée.	

TABLE 3.7 – Planification des sprints

3.8 Conclusion

Dans ce chapitre nous avons présentée la spécification de besoin fonctionnel et non fonctionnel. Aussi, nous avons décrit les différents diagrammes de conception puis nous avons présenté la méthodologie de conception. Enfin nous avons réalisé la planification des sprint. Dans le dernier, chapitre nous passerons à la phase de réalisation du sprint 1.

Chapitre 4

Sprint1 : Configuration de l'architecture cloud AWS.

Introduction

Après avoir déduit le besoin de du projet et identifie les acteurs nous présentent le premier 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 a pour but de configurer l'architecture cloud qui en va l'utiliser dans notre projet, la solution cloud choisit et l'hybride cloud nous allons configurer une VPC puis connectée à notre data center a travers un VPN finalement nous allons configurer un cluster EKS et établir une connexion entre une machine locale et le cluster.

4.2 Architecture Global

Dans cette partie en va proposer l'architecture globale utilisée pour la configuration de l'infrastructure cloud.

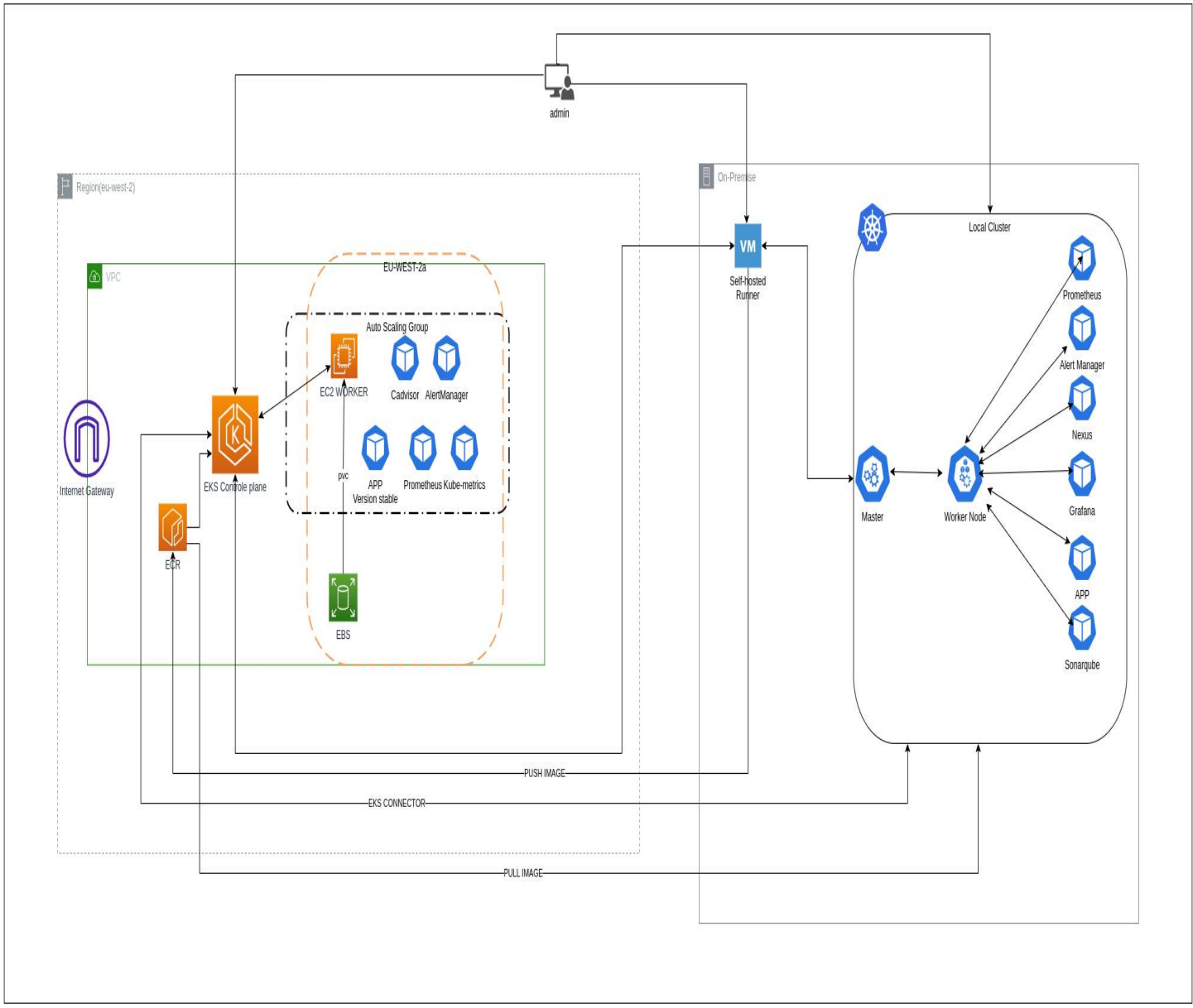

FIGURE 4.1 – Architecture Cloud

4.3 Architecture Détaillée

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

4.3.1 Diagramme Cas d'utilisation Détaillée

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

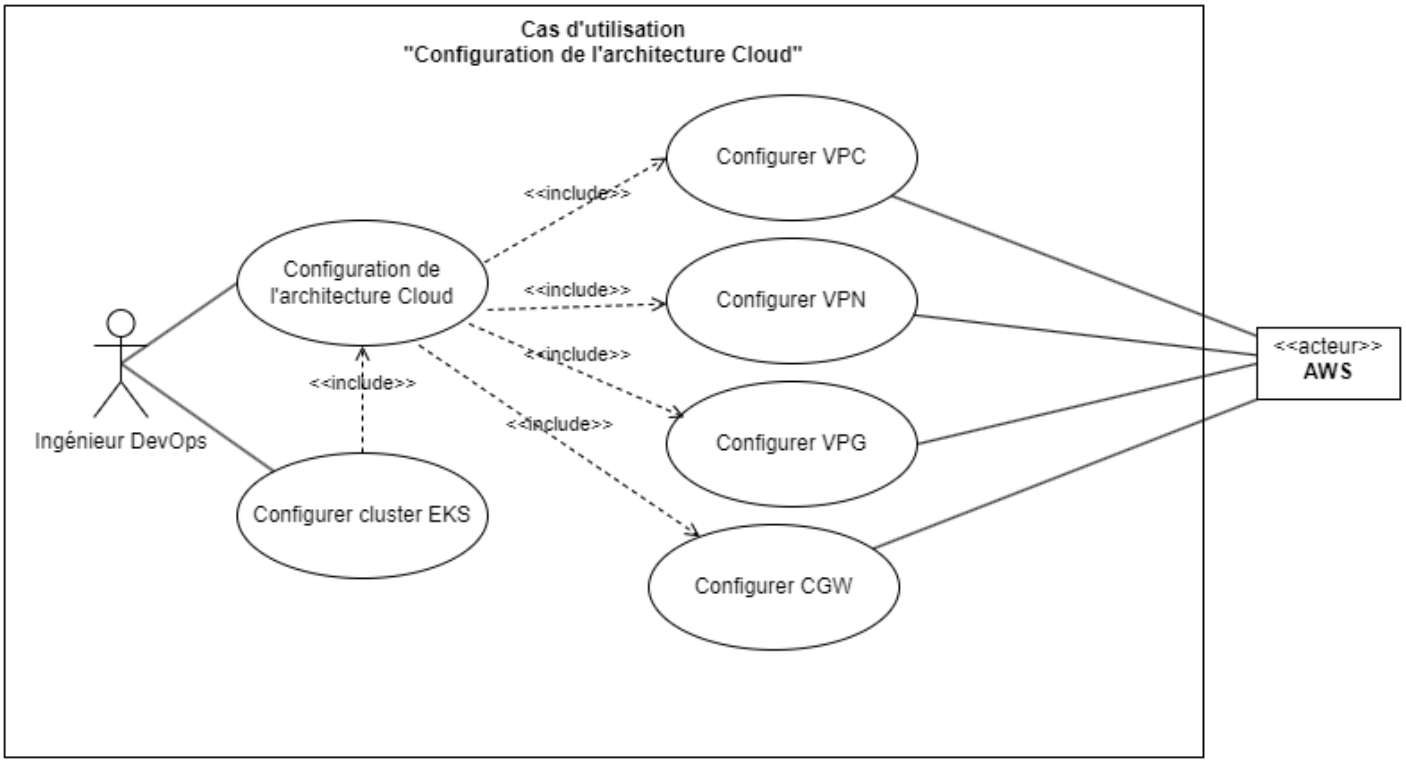

FIGURE 4.2 – cas d'utilisation"Configuration de l'architecture cloud AWS"

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

Titre	Configuration de l'architecture cloud AWS
Acteur	Ingénieur DevOps
Description	L'ingénieur DevOps gérait la configuration de l'architecture et la création du cluster EKS.
Pré conditions	Connexion a amazon web services.
Post conditions	une architecture configurait correctement.
Scénario nominal	la connexion est réussite et la configuration réussite.
Scénario alternatif	La connexion échoue entre le cluster et la machine locale ou la configuration n'est pas correcte.

TABLE 4.1 – Description de cas d'utilisation

4.3.2 Diagramme de séquence Détaillée

Nous détaillons ci-dessous les interactions générales du système

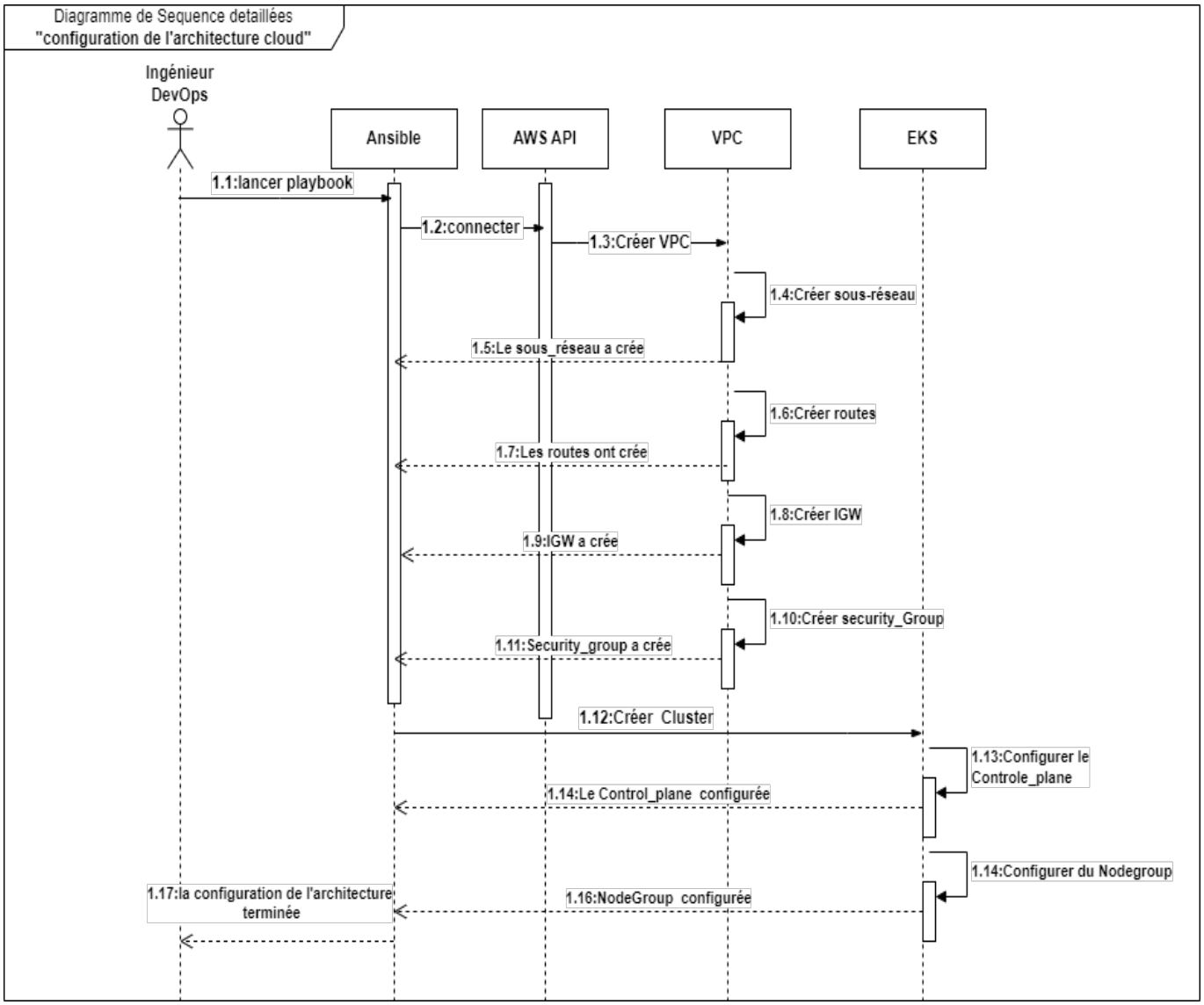

FIGURE 4.3 – Architecture Cloud

4.4 Realisation

4.5 Conclusion

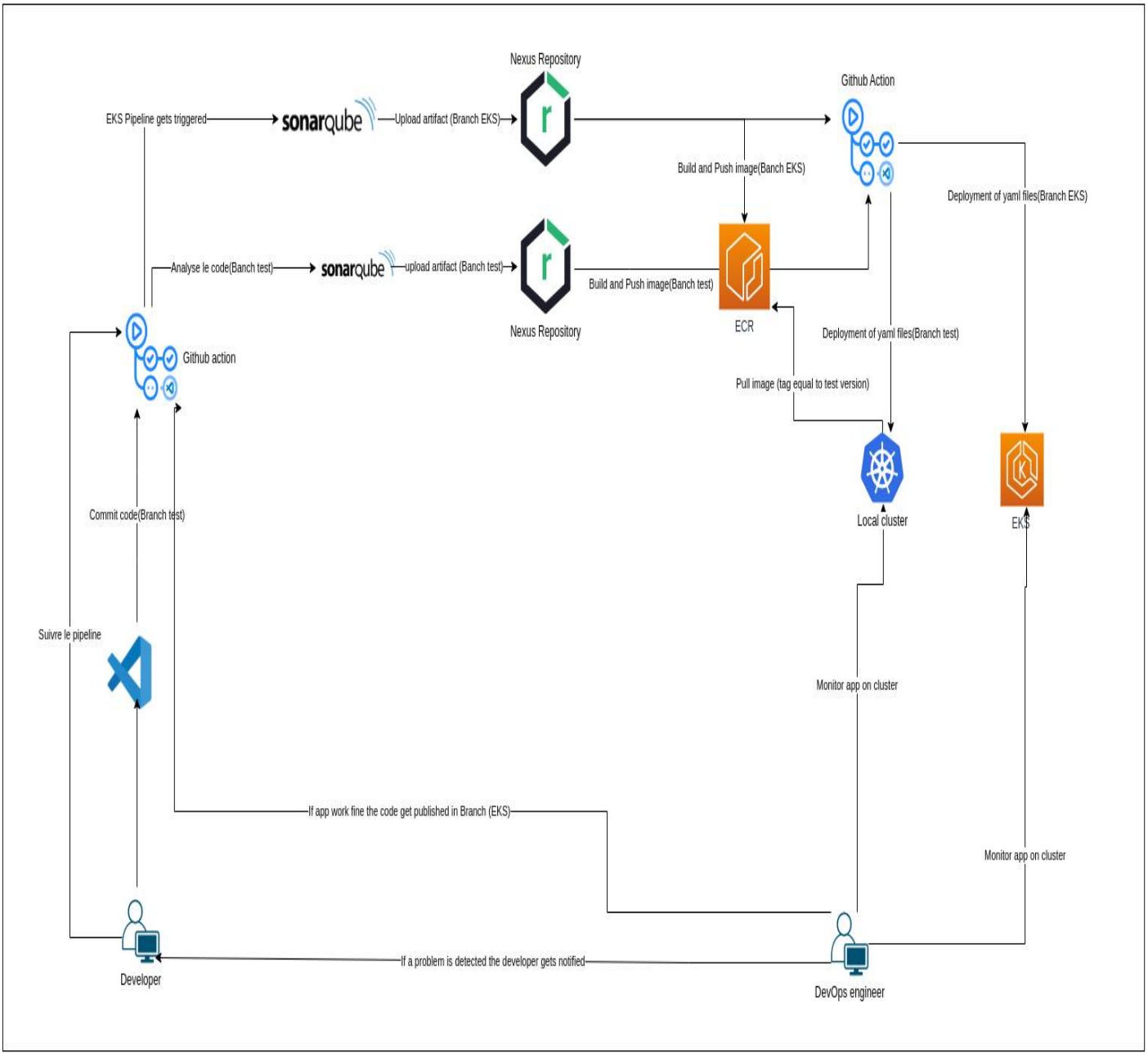

FIGURE 4.4 – la configuration de pipeline

Chapitre 5

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.

5.1 Objectif du Sprint

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

5.2 Architecture Détaillée

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

5.2.1 Diagramme Cas d'utilisation Détaillée

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

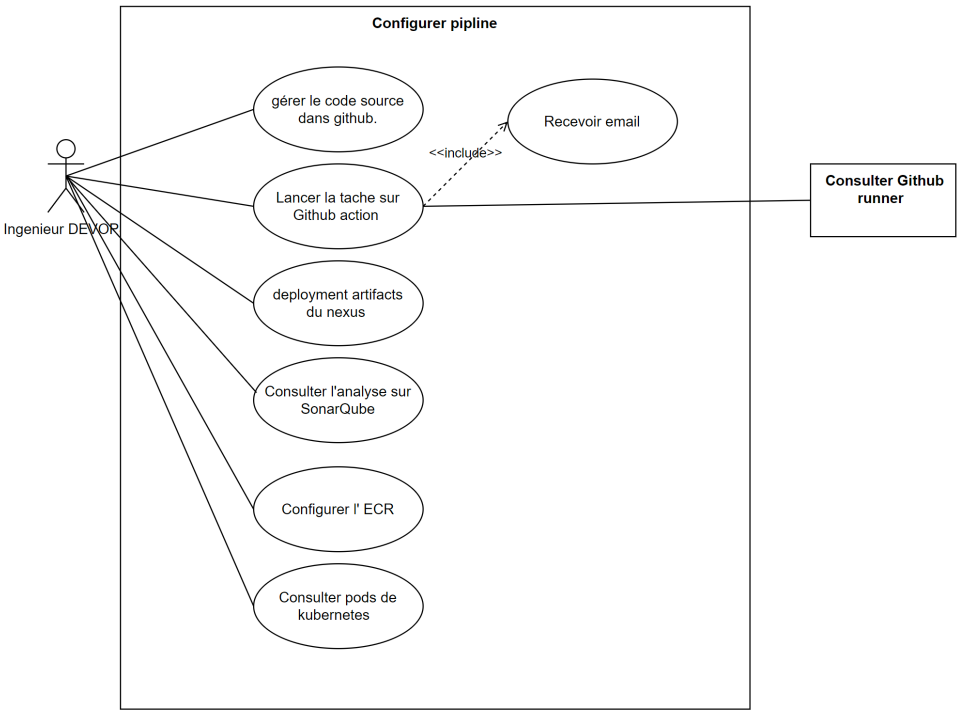

FIGURE 5.1 – cas d'utilisation"Configuration de la Pipline"

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

Titre	Configuration de la Pipline
Acteur	Ingénieur DevOps
Description	L'ingénieur DevOps gérait la configuration de la Pipline 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 5.1 – Description de cas d'utilisation

5.2.2 Diagramme de sequance Détaillée

Nous détaillons ci-dessous les interactions générales du système

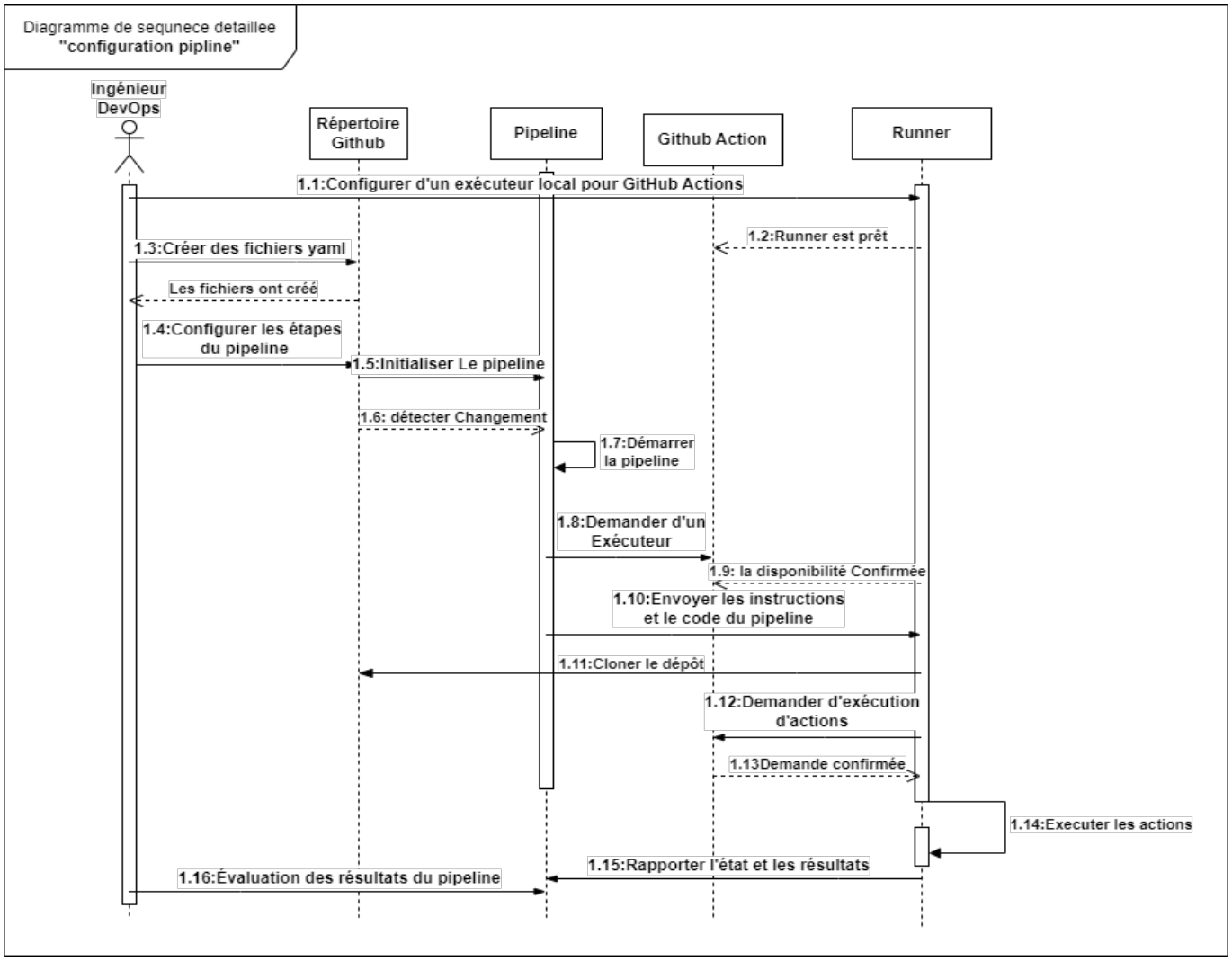

FIGURE 5.2 – diagramme de séqunce"Configuration de la Pipline"

Chapitre 6

Sprint3 : Déploiement de l'application sur le cluster.

Introduction

Après avoir déduit la configuration du pipline nous présentent la troisieme 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

6.1 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.

6.2 Architecture Détaillée

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

6.2.1 Diagramme Cas d'utilisation Détaillée

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

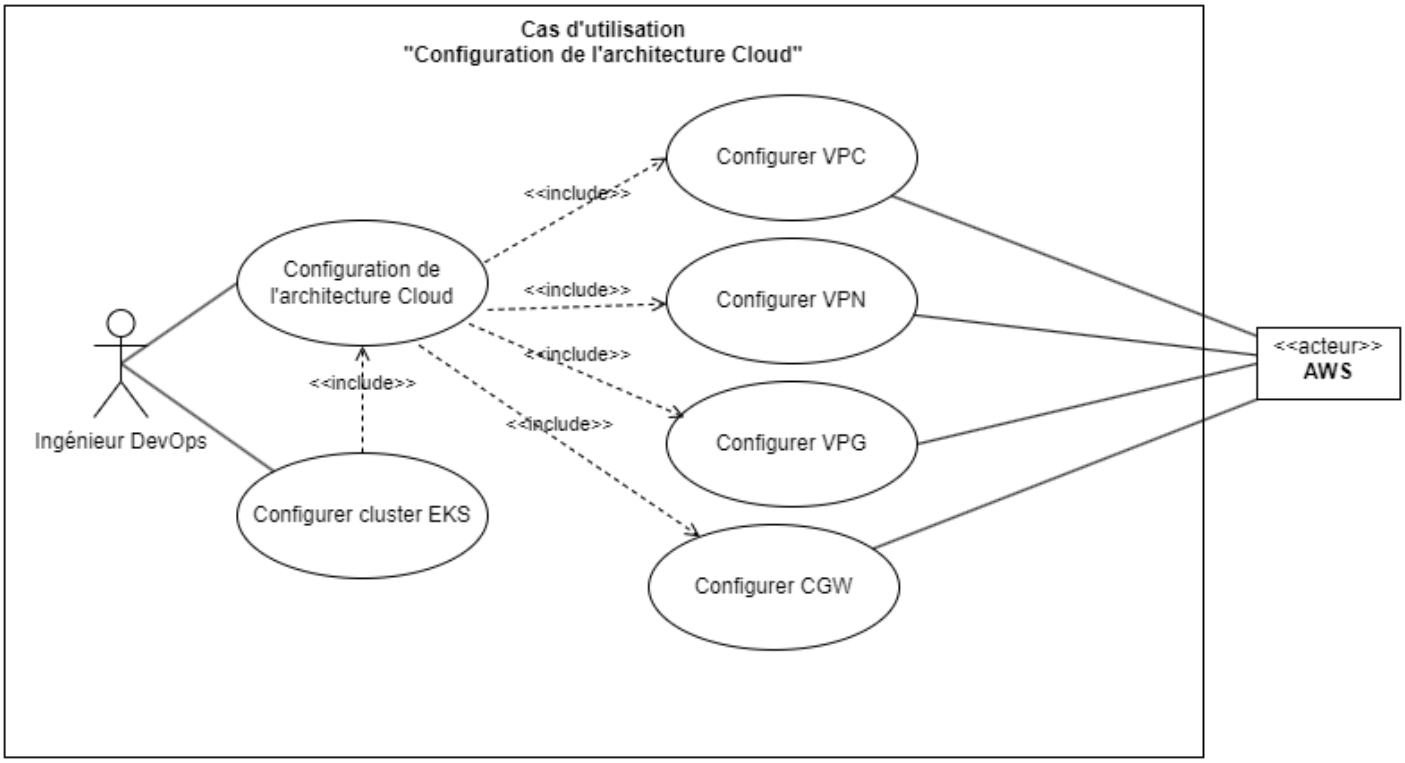

FIGURE 6.1 – cas d'utilisation"Déploiement de l'application sur le cluster."

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 depandances .
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 6.1 – 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 classe. [Online ; accessed 01-mars-2023].

[14] Wikipedia. Diagramme de séquence. [Online ; accessed 01-mars-2023].

[15] Wikepedia. Diagramme de déploiement. [Online ; accessed 01-mars-2023].