Ingetis
École d'ingénierie informatique

ECOLE PMN
PASSERELLE DES METIERS DU NUMERIQUE
GROUPE FITEC

GE HealthCare

ESI - Titre RNCP de Niveau 7

IBTool: Développement, Migration et Déploiement d'une Application Cloud-Native pour GE Healthcare

Memoire

Nom Prénom	ALINIA Sana
Nom Prénom de la tutrice	CHATELLARD Magalie
Numéro de matricule	Nº 029

Aout 2024
Paris, France

REMERCIEMENT

Je tiens à exprimer ma profonde gratitude envers L'École La Passerelle des Métiers du Numérique - La PMN, pour leur confiance et l'accompagnement précieux qui m'ont guidé tout au long de la réalisation de mon projet professionnel.

Je remercie sincèrement mes professeurs de la PMN pour la qualité de leurs enseignements techniques, qui ont considérablement enrichi mes compétences. Un remerciement particulier est adressé à Madame Magalie Chatellard, ma tutrice, dont le soutien constant, les conseils avisés et l'encouragement inlassable ont été des piliers essentiels tout au long de ce parcours.

Je souhaite également exprimer ma gratitude à mon manager, Madame Magali Wissocq, ainsi qu'à l'équipe RH de GE Healthcare, qui ont su reconnaître et valoriser mon potentiel, m'ouvrant ainsi les portes de l'entreprise. Ma reconnaissance va également à mes collègues chez GE Healthcare, dont les conseils éclairés et le soutien inestimable ont été d'une grande aide tout au long de mon alternance.

Cette expérience d'alternance a joué un rôle déterminant dans la construction de mon avenir professionnel, et je suis profondément reconnaissant pour les opportunités qui m'ont été offertes. Enfin, je tiens à adresser un remerciement chaleureux au jury pour le temps et l'attention consacrés à l'évaluation de ce mémoire.

RESUME

Ce mémoire traite du développement et de la transformation d'IBTool, une application web interne utilisée par GE Healthcare pour gérer les plaintes des clients. J'ai d'abord développé IBTool en tant qu'application on-premises, mais l'outil présentait des limitations telles qu'une scalabilité réduite et des coûts de maintenance élevés. Ces défis ont conduit à la décision de migrer l'application vers une architecture cloud-native sur Amazon Web Services (AWS).

La migration vers le cloud a impliqué plusieurs étapes, dont la conception d'une nouvelle architecture, l'implémentation de services AWS comme Amazon ECS, EC2, et l'Application Load Balancer (ALB), ainsi que l'intégration d'un pipeline CI/CD pour automatiser les déploiements. Grâce à Terraform, l'infrastructure as code a été utilisée pour automatiser la gestion des configurations, assurant ainsi la cohérence entre les différents environnements.

Les résultats sont clairs : IBTool est devenu plus réactif aux variations de la demande, a amélioré sa disponibilité en réduisant les temps d'arrêt, et a renforcé la sécurité des données. Le déploiement automatisé a également permis de réduire les erreurs humaines et d'accélérer les mises à jour.

Cette migration a non seulement rendu IBTool plus robuste et performant, mais elle offre aussi un exemple précieux pour d'autres organisations envisageant des transitions similaires vers le cloud. Ce projet ouvre la voie à de futures améliorations, telles que l'intégration d'outils d'analyse avancés et l'exploration de l'architecture serverless.

TABLE OF CONTENTS

LIST DES FIGURES

ABREVIATIONS

Abréviation	Signification
IB	Install Base
CSO	Customer Satisfaction Opportunity
COM	Complaint (Bug ou problème signalé par le client)
GE	General Electric
R&D	Recherche et Développement
IRM	Imagerie par Résonance Magnétique
AW	Advantage Workstation
SSO	Single Sign On
CRUD	Create Read Update Delete
PQM	Post-market Quality Management
CT	Computed Tomography (ou Tomodensitométrie)
FDA	Food and Drug Administration
AWS	Amazon Web Services
EC2	Elastic Compute Cloud
ECS	Elastic Container Service
ALB	Application Load Balancer
SSL	Secure Sockets Layer
vCPU	Virtual Central Processing Unit
S3	Simple Storage Service
IAM	Identity and Access Management
VPC	Virtual Private Cloud
CI/CD	Continuous Integration / Continuous Deployment
IaC	Infrastructure as Code
KMS	Key Management Service
ACM	AWS Certificate Manager
ECR	Elastic Container Registry
JSON	JavaScript Object Notation
API	Application Programming Interface
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
SaaS	Software as a Service
WAF	Web Application Firewall
REST	Representational State Transfer
MFA	Multi-Factor Authentication
SDK	Software Development Kit

1. INTRODUCTION ET CONTEXTE DU PROJET

1.1 L'ENTREPRISE ET LES EQUIPES

GE Healthcare, filiale de General Electric, est un leader mondial dans le domaine des technologies médicales, avec une présence dans plus de 100 pays. Le site de Buc, France, qui abrite le siège européen de GE Healthcare, se concentre principalement sur la recherche et le développement (R&D) de dispositifs d'imagerie médicale avancée, tels que les systèmes d'IRM, de tomodensitométrie, et de rayons X. Ces technologies, associées à des logiciels de surveillance clinique, permettent aux professionnels de santé de prendre des décisions plus rapides et mieux informées, améliorant ainsi la qualité des soins. Dans ce contexte d'innovation continue, la gestion efficace des retours clients, notamment les plaintes liées aux produits logiciels, revêt une importance stratégique. Le projet IBTool s'inscrit précisément dans cette perspective, avec pour objectif de moderniser et d'optimiser le processus de gestion des plaintes clients.

L'équipe Install Base (IB), au sein de laquelle j'ai effectué mon projet d'alternance, est responsable du suivi et de la résolution des problèmes signalés par les clients. L'ancien système de gestion des plaintes, utilisé par l'équipe IB, présentait plusieurs limites, tant sur le plan de l'efficacité que de la convivialité. C'est dans ce cadre que mon projet s'est articulé autour du développement et de la migration d'un nouvel outil, l'IBTool, conçu pour répondre aux besoins évolutifs de l'entreprise en matière de gestion des plaintes.

La présente mémoire retrace le développement d'IBTool, depuis sa conception en environnement on-premises jusqu'à sa migration vers une architecture cloud-native sur Amazon Web Services (AWS). Le projet s'est concentré sur l'amélioration de la scalabilité, de la performance, et de la sécurité de l'application, tout en optimisant les coûts opérationnels et en facilitant la maintenance. Cette transformation a nécessité l'intégration de technologies modernes et l'adoption de meilleures pratiques en matière de développement et de déploiement logiciel, que ce document explore en détail.

1.2 PROBLEMATIQUE ET OBJECTIFS

L'outil actuel utilisé par l'équipe IB pour le suivi des plaintes présente plusieurs limitations qui affectent l'efficacité et la qualité des analyses. Parmi ces limitations, on note l'utilisation de plusieurs outils non intégrés, des données disparates, une visibilité insuffisante sur les enregistrements critiques, et une interface utilisateur peu intuitive.

1.2.1 PROBLEMATIQUE DU DEVELOPPEMENT D'IBTOOL

L'ancien outil utilisé par l'équipe IB pour le suivi et la gestion des plaintes présentait plusieurs limitations majeures qui compromettaient l'efficacité du processus de traitement des réclamations. Ces limitations engendraient non seulement des pertes de temps, mais aussi des risques d'erreurs

humaines et de mauvaise gestion des dossiers critiques. Les principaux problèmes identifiés étaient les suivants :

1. **Fragmentation des Informations** : Les données relatives aux plaintes étaient dispersées sur plusieurs systèmes non intégrés (TWD, PQM, SFDC, ClearQuest, etc.). Cette fragmentation rendait difficile la synchronisation des informations, entraînant des incohérences entre les différents outils. De plus, les utilisateurs devaient naviguer entre plusieurs interfaces, ce qui augmentait la complexité du processus et les risques de doublons ou de pertes d'information.
2. **Manque de Visibilité sur les Dossiers Sensibles** : L'ancien outil ne fournissait pas une visibilité adéquate sur les enregistrements sensibles et critiques, tels que les dossiers de sécurité ou les plaintes réglementaires. Les parties prenantes avaient des difficultés à accéder rapidement aux informations nécessaires, ce qui pouvait entraîner des retards dans la prise de décision et une gestion inefficace des dossiers prioritaires.
3. **Actions Manuelles et Temps de Traitement** : Le processus de suivi des plaintes reposait sur de nombreuses actions manuelles, chronophages et sujettes à des erreurs humaines. Cela incluait la mise à jour manuelle des informations dans plusieurs systèmes et le suivi manuel des statuts des réclamations. Cette approche ralentissait considérablement le processus d'analyse et augmentait le risque de négliger des dossiers importants.
4. **Interface Utilisateur Peu Intuitive** : L'interface de l'ancien outil était jugée peu conviviale par les utilisateurs. La complexité de l'interface augmentait la courbe d'apprentissage pour les nouveaux utilisateurs et rendait l'utilisation quotidienne de l'outil fastidieuse, ce qui pouvait entraîner une baisse de productivité.

1.2.1.1 RESOLUTION DES PROBLEMES

Le développement du nouvel IBTool a permis de résoudre ces problèmes de manière significative. Le nouvel outil centralise toutes les informations relatives aux plaintes dans une interface unique, améliorant ainsi la synchronisation et la traçabilité des données. La visibilité sur les enregistrements critiques a été renforcée, permettant aux utilisateurs d'accéder facilement aux dossiers sensibles. De plus, l'automatisation de nombreuses tâches a réduit le besoin d'interventions manuelles, accélérant ainsi le processus d'analyse. Enfin, une interface utilisateur améliorée offre une expérience plus intuitive, ce qui facilite l'adoption de l'outil par l'équipe.

1.2.2 PROBLEMATIQUE DE LA MIGRATION VERS LE CLOUD

Après le développement du nouvel IBTool, il est devenu évident que pour maximiser son efficacité, une migration vers une architecture cloud-native était nécessaire. L'ancienne infrastructure sur site posait plusieurs défis qui limitaient la scalabilité, la disponibilité et la maintenabilité de l'outil. Les principaux défis à surmonter dans cette deuxième phase étaient les suivants :

5. **Scalabilité Limitée** : L'infrastructure actuelle ne permettait pas de répondre efficacement aux variations de la demande. En période de forte activité, les ressources disponibles étaient insuffisantes, ce qui entraînait des ralentissements et une dégradation des performances. À

l'inverse, pendant les périodes de faible activité, les ressources sous-utilisées représentaient un gaspillage de coûts.

6. **Disponibilité et Résilience** : L'outil était déployé sur une infrastructure locale qui ne pouvait pas garantir une haute disponibilité en cas de défaillance. Un problème technique dans le centre de données local pouvait entraîner une indisponibilité de l'outil pour les utilisateurs, ce qui pouvait avoir des conséquences graves, notamment pour la gestion des dossiers critiques.
7. **Maintenance et Mises à Jour** : La gestion des serveurs sur site nécessitait une maintenance régulière et des interventions manuelles pour les mises à jour, ce qui était non seulement coûteux, mais aussi source de risques d'erreurs humaines. De plus, le déploiement de nouvelles fonctionnalités était lent et complexe, retardant la livraison de nouvelles améliorations aux utilisateurs.
8. **Optimisation des Coûts** : L'infrastructure sur site ne permettait pas une utilisation optimale des ressources, avec des coûts fixes élevés pour maintenir des serveurs en fonctionnement, indépendamment de la charge réelle. Ce modèle était inefficace, surtout dans un contexte où la demande pouvait fluctuer de manière significative.

1.2.2.1 RESOLUTION DES PROBLEMES

La migration vers le cloud a permis de surmonter ces défis en tirant parti des capacités de l'infrastructure AWS. Le nouvel IBTool cloud-native peut désormais évoluer automatiquement en fonction de la demande, garantissant des performances optimales à tout moment. De plus, la redondance des données sur plusieurs zones de disponibilité AWS assure une haute résilience, minimisant les risques d'indisponibilité. Les processus de déploiement et de maintenance sont désormais automatisés, réduisant ainsi les coûts et les risques d'erreurs, tout en accélérant la mise en œuvre des nouvelles fonctionnalités. Enfin, l'approche de tarification à la demande d'AWS permet une optimisation des coûts, en n'utilisant que les ressources nécessaires au bon fonctionnement de l'outil.

1.3 OBJECTIFS COMBINES

Les objectifs du projet couvrent ainsi à la fois le développement d'un nouvel IBTool plus performant et sa transformation en une application cloud-native :

- **Amélioration des Performances** : Assurer une meilleure réactivité de l'outil, une synchronisation des données en temps réel, et une interface utilisateur améliorée.
- **Scalabilité et Disponibilité** : Déployer IBTool sur une architecture capable de s'adapter automatiquement aux pics de charge, tout en garantissant une haute disponibilité.
- **Sécurité Renforcée** : Protéger les données sensibles des utilisateurs grâce aux fonctionnalités de sécurité avancées offertes par AWS.
- **Optimisation des Coûts** : Réduire les coûts opérationnels en ne consommant que les ressources nécessaires à un moment donné.

Dans les sections suivantes, nous explorerons le développement d'IBTool ainsi que les services AWS spécifiques utilisés dans sa migration vers une architecture cloud-native. Nous aborderons la conception et l'implémentation de cette nouvelle architecture, la stratégie de déploiement,

ainsi que les bénéfices et contributions de cette transformation. Cette analyse détaillée fournira une compréhension claire de la manière dont IBTool, à travers son développement initial et sa transformation cloud, tire parti des services AWS pour atteindre ses objectifs et offrir une expérience utilisateur supérieure.

2. CONCEPTION ET ARCHITECTURE

2.1 ARCHITECTURE DE L'IBTOOL ON-PREMISES

Avant la migration vers une architecture cloud-native, IBTool était déployé dans un environnement on-premises. Cette application web interne, essentielle pour GE Healthcare, permettait de gérer les plaintes des clients concernant les produits de l'entreprise. L'architecture on-premises d'IBTool reposait sur une structure en trois couches, comprenant un frontend, un backend, et une base de données, le tout orchestré via Docker pour simplifier le déploiement.

2.1.1 INFRASTRUCTURE PHYSIQUE

L'IBTool on-premises était déployé dans des conteneurs Docker, chacun dédié à un composant spécifique de l'application. Les principaux éléments de cette architecture étaient les suivants :

- **Frontend** : Le frontend de l'application était développé en JavaScript, utilisant Node.js, HTML, CSS, et la bibliothèque DevExtreme jQuery pour créer des composants d'interface utilisateur. Ce frontend interagissait avec le backend via des appels API, envoyant et recevant des données nécessaires pour gérer les plaintes des clients. Un mécanisme de mise en cache était également mis en place dans le frontend pour optimiser les performances et réduire les temps de réponse.
- **Backend** : Le backend était également développé en JavaScript, utilisant Express.js pour lancer une API REST qui traitait les requêtes provenant du frontend. Le backend communiquait directement avec la base de données MongoDB pour stocker et récupérer les informations sur les plaintes. Un système de cache basé sur Redis était intégré au backend pour améliorer les performances, en minimisant les accès directs à la base de données pour les requêtes fréquentes.
- **Base de Données** : La base de données MongoDB servait de dépôt central pour toutes les informations relatives aux plaintes des clients. Les données incluaient les détails des produits, les statuts des réclamations, et les actions correctives entreprises. MongoDB, en tant que base de données NoSQL, offrait une flexibilité importante pour stocker des données structurées et non structurées.

2.1.3 AMELIORATION DES PERFORMANCES

2.1.3.1 MISE EN CACHE

Afin d'optimiser les performances de l'application IBTool, des mécanismes de mise en cache ont été mis en place à la fois côté client et côté serveur. Cette stratégie vise à réduire les temps de réponse et à améliorer l'efficacité globale de l'application en limitant les accès répétés à la base de données.

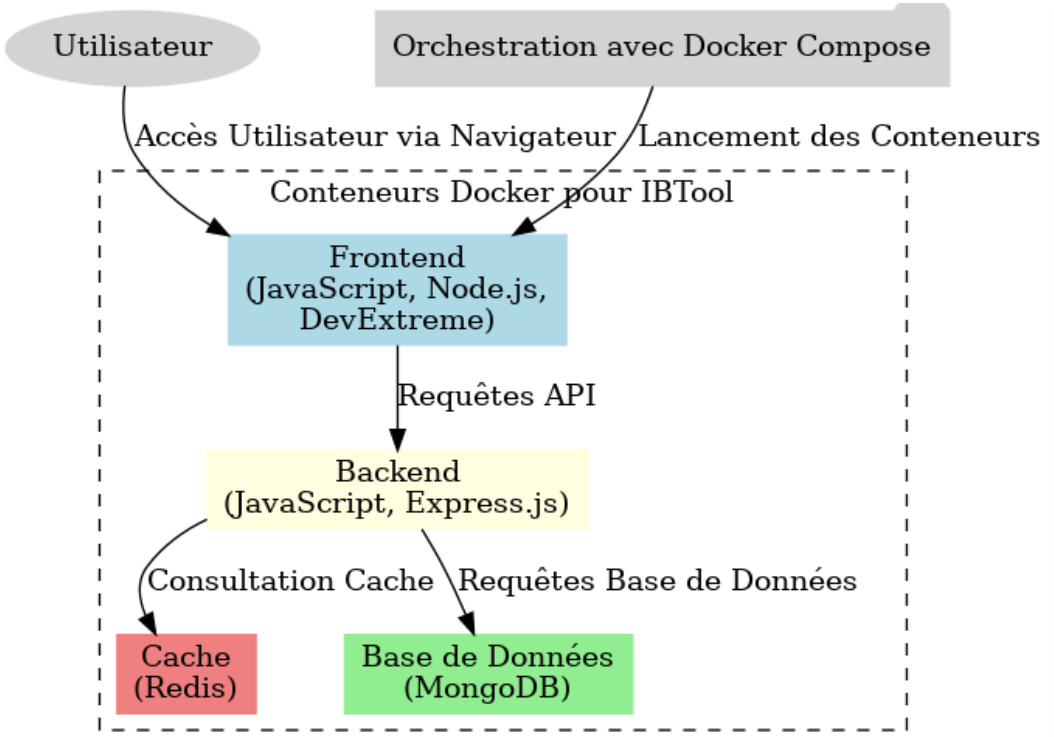

Figure 1 Architecture On-Premises d'IBTool avec Conteneurs Docker

Côté client, la mise en cache permet de stocker temporairement les données fréquemment utilisées directement dans la mémoire du navigateur. Ainsi, lorsque l'utilisateur interagit avec l'application et nécessite des données déjà consultées, celles-ci sont rapidement récupérées à partir du cache local, réduisant la dépendance aux requêtes serveurs répétitives et améliorant ainsi la réactivité de l'interface utilisateur.

Du côté serveur, Redis a été utilisé comme solution de mise en cache en mémoire pour gérer les données issues de la base de données MongoDB. Grâce à Redis, les informations fréquemment demandées sont stockées temporairement en mémoire, permettant une récupération plus rapide lors de requêtes ultérieures sans solliciter la base de données. Ce mécanisme est particulièrement efficace pour les données qui changent rarement mais sont souvent consultées.

Par exemple, lors d'une requête pour récupérer des enregistrements, le serveur vérifie d'abord si les données sont déjà disponibles dans le cache Redis. Si elles le sont, elles sont immédiatement renvoyées à l'utilisateur, évitant ainsi une requête supplémentaire à MongoDB. Si elles ne sont pas présentes, le serveur récupère les données depuis la base de données, les retourne à l'utilisateur, puis les stocke dans Redis pour des consultations futures.

Cette approche de mise en cache permet de réduire de manière significative les temps de réponse de l'application, d'éviter une surcharge inutile du serveur, et d'optimiser l'utilisation des ressources

disponibles. En conséquence, IBTool bénéficie d'une performance accrue, offrant une expérience utilisateur plus fluide et plus réactive.

2.1.3.2 MIGRATION VERS REACT

La migration vers React dans le côté front-end présente de nombreux avantages potentiels. Actuellement, cette migration est en cours et apporte déjà des améliorations significatives en termes de performances et de réutilisabilité du code. En termes de performances, React offre une architecture optimisée qui permet de réduire le temps de chargement des pages, d'améliorer la fluidité de l'interface utilisateur et d'optimiser l'utilisation des ressources du navigateur. Grâce à sa méthode de rendu virtuel, React minimise les opérations coûteuses de manipulation du DOM, ce qui se traduit par une expérience utilisateur plus réactive et des temps de réponse plus rapides.

De plus, la migration vers React facilite la réutilisabilité du code. Grâce à la création de composants modulaires, il est possible de découper l'interface utilisateur en petites parties réutilisables, ce qui permet de gagner du temps et de réduire les erreurs de développement. Les composants React peuvent être facilement réutilisés dans différentes parties de l'application, ce qui favorise également la cohérence et la maintenabilité du code.

En outre, la communauté et l'écosystème React sont très actifs, ce qui facilite le développement et l'intégration de nouvelles fonctionnalités. De nombreux outils, bibliothèques et documentations sont disponibles pour accompagner le développement avec React, ce qui permet de gagner du temps et d'éviter de réinventer la roue.

Ces avantages combinés font de la migration vers React une décision stratégique pour l'amélioration globale de notre application. Cependant, il convient de souligner que la migration est un processus complexe qui nécessite du temps et des ressources. Par conséquent, la migration vers React sera achevée dans le cadre de travaux futurs, afin de permettre une transition en douceur sans compromettre la stabilité de l'application actuelle.

2.1.4 STRUCTURE ET PAGES DE L'INTERFACE UTILISATEUR D'IBTOOL

La structure des pages dans IBTool est illustrée à la Fig. 8. Une brève explication de chaque page sera donnée plus loin dans cette section.

Le backend se compose de deux parties principales : a) traiter les requêtes http reçues dans ses endpoints et b) se connecter à la base de données et effectuer des opérations CRUD. Les principales technologies utilisées dans la partie backend sont les suivantes :

- NodeJS
- Express
- MongoDB
- Webpack
- Git/Github

- Docker

J'ai également utilisé différentes libraires de npm pour encoder/déchiffrer les informations d'identification des utilisateurs, se connecter à la base de données, builder le projet, convertir json en csv, etc.

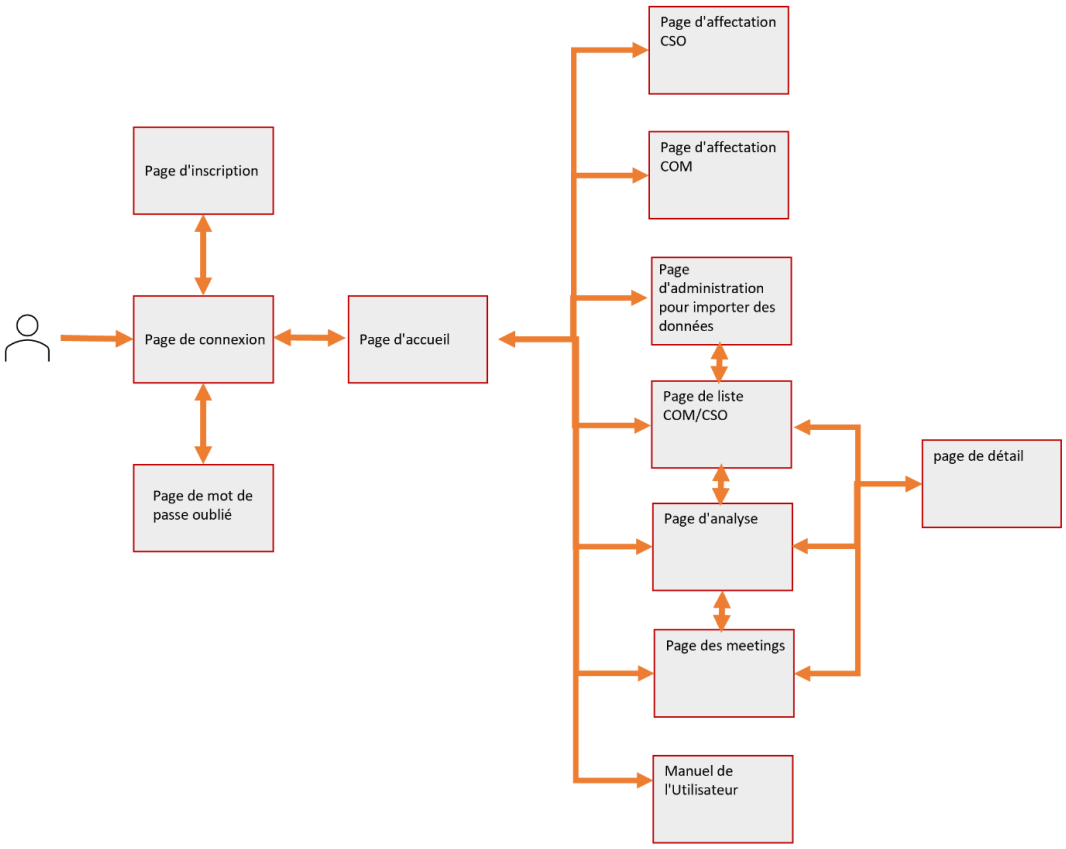

Figure 2 Pages de IBTool. Certains liens ne sont pas affichés pour plus de simplicité

2.1.5 SCHEMA DE BASE DE DONNEES ET DU SYSTEME

Pour la base de données, j'ai choisi MongoDB (une base de données NoSQL) pour les raisons suivantes : Tout d'abord, les données que nous devons stocker (par exemple, la liste des COM/CSO) contiennent de nombreux champs, mais certains ou plusieurs d'entre eux peuvent être vides.

L'utilisation d'une base de données de relations comme MySql peut gaspiller de l'espace tandis qu'une base de données basée sur des documents comme MongoDB ne stocke que les informations remplies et économise de l'espace. Deuxièmement, MongoDB est évolutif et plus rapide. Troisièmement, il existe des outils et des libraires javascript qui soutiennent MongoDB et facilitent le development. Le schéma de la base de données est illustré à la Fig. 3.

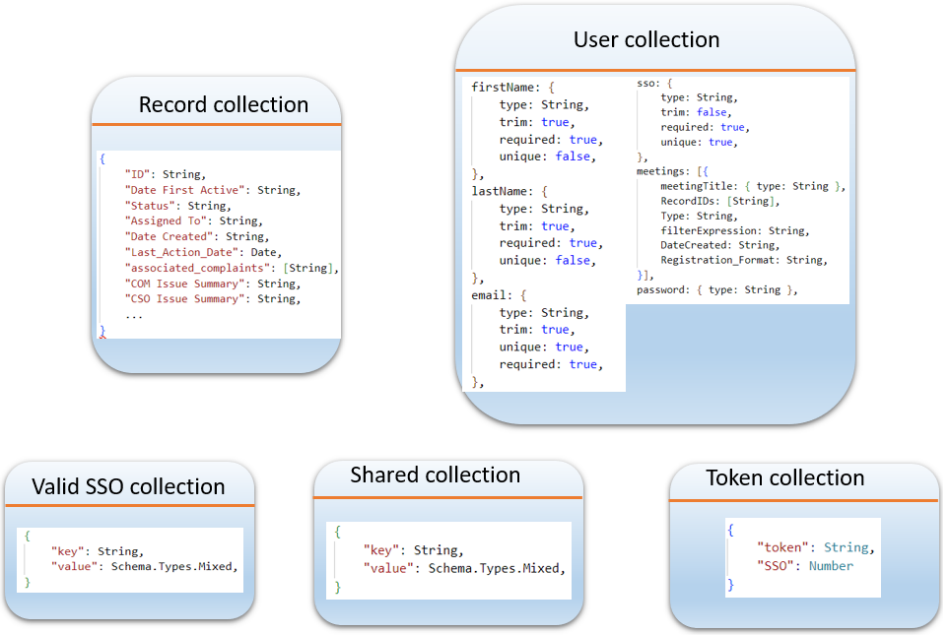

Figure 3 Schéma de base de données pour IBTool.

De nombreux champs de la collection Record ne sont pas affichés en raison d'un manque d'espace. Les collections "User" et "Record" sont connectées à l'aide de COM-ID / CSO-ID. La collection "Shared" stocke toutes les informations qui ne sont pas destinées à un utilisateur spécifique (par exemple, la dernière mise à jour COM). La collection "Valid SSO" stocke la liste des utilisateurs SSO autorisés à s'enregistrer et à utiliser IBTool.

2.1.2 LIMITATIONS DE L'ARCHITECTURE ON-PREMISES

Bien que l'architecture on-premises d'IBTool ait été efficace pour les besoins initiaux de l'équipe, elle présentait plusieurs limitations :

- **Scalabilité** : L'ajout de nouvelles ressources pour faire face à une augmentation de la charge nécessitait une intervention manuelle, limitant la capacité de l'application à évoluer rapidement en réponse aux besoins.
- **Maintenance** : La gestion des conteneurs et des serveurs physiques nécessitait une maintenance régulière, augmentant les coûts et le temps d'arrêt potentiel de l'application.
- **Résilience** : Une panne matérielle dans le centre de données local pouvait rendre l'application indisponible, affectant la capacité de l'équipe à gérer les plaintes critiques des clients.
- **Complexité de la Gestion des Dépendances** : Bien que Docker ait simplifié le déploiement, la gestion des dépendances entre les différents services (frontend, backend, base de données) dans un environnement on-premises restait complexe et nécessitait une surveillance constante.

L'architecture on-premises d'IBTool, malgré ses avantages initiaux, a montré des limites significatives en termes de scalabilité, de maintenance, et de résilience. Ces défis ont conduit à la décision de migrer IBTool vers une architecture cloud-native, permettant ainsi de tirer parti des avantages offerts par le cloud pour améliorer la performance, la disponibilité, et la flexibilité de l'application.

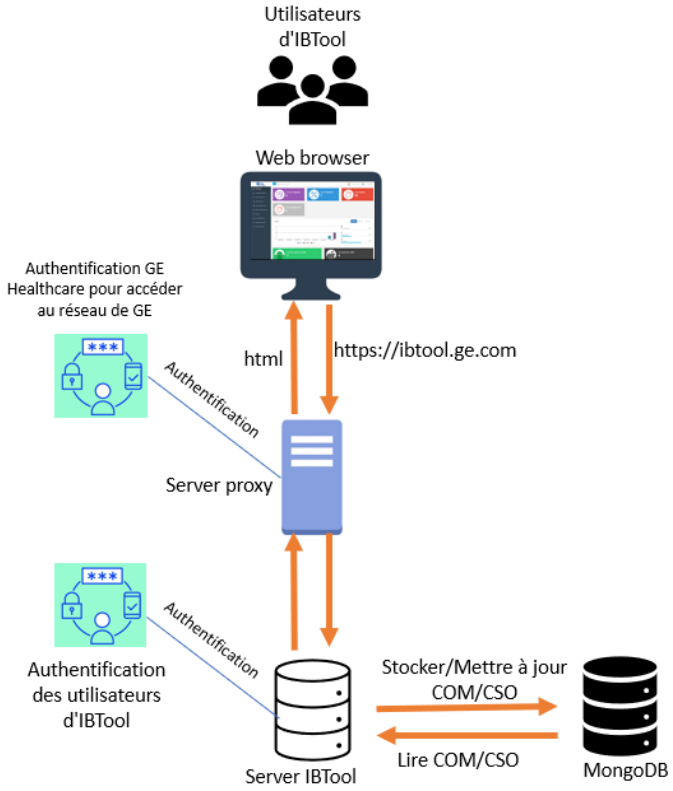

Figure 4 Accès à IBTool par les utilisateurs

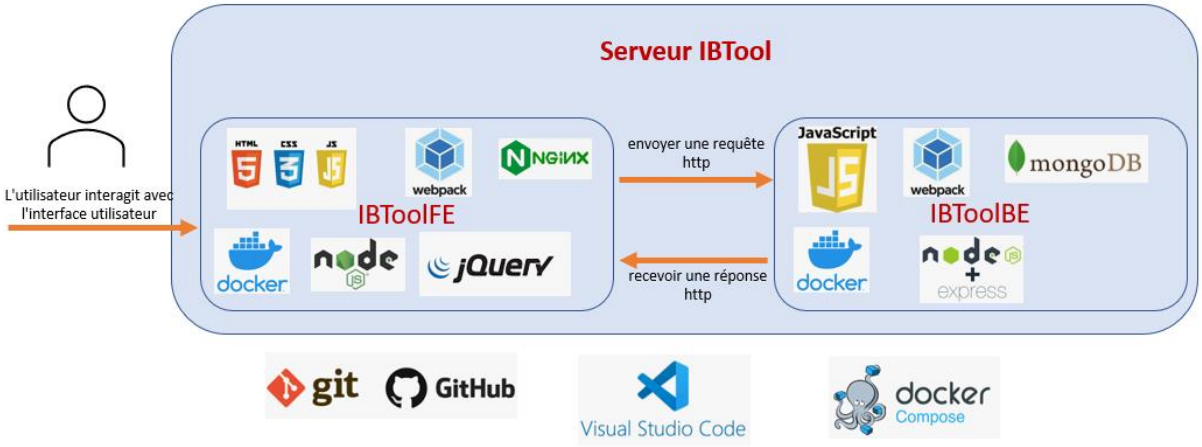

Figure 5 Composants de IBTool

2.2 ARCHITECTURE DE L'IBTOOL CLOUD-NATIVE

2.2.1 AVANTAGES DE CLOUD COMPUTING

L'informatique en nuage a profondément modifié la manière dont les applications sont conçues, déployées et administrées. En adoptant les services cloud, les organisations, comme GE Healthcare avec IBTool, peuvent bénéficier d'une flexibilité accrue, d'une meilleure évolutivité, et d'une réduction

des coûts, par rapport à une infrastructure traditionnelle sur site. Pour IBTool, la migration vers une architecture cloud-native a permis d'améliorer considérablement les fonctionnalités et les performances de l'application.

L'un des principaux avantages du cloud pour IBTool réside dans sa capacité à évoluer en fonction de la demande. Contrairement à une infrastructure on-premises, où les ressources sont souvent limitées et difficiles à ajuster, les ressources cloud peuvent être augmentées ou réduites en temps réel, garantissant ainsi que l'application puisse répondre efficacement aux fluctuations de la charge de travail. Cela évite le surdimensionnement des ressources, qui est à la fois coûteux et inefficace.

En outre, l'informatique en nuage offre un modèle de tarification à la demande, permettant à IBTool de payer uniquement pour les ressources effectivement utilisées. Cela se traduit par une réduction significative des coûts liés à la maintenance du matériel physique et à la gestion des serveurs. De plus, les fournisseurs de cloud comme AWS proposent des options de haute disponibilité en utilisant plusieurs centres de données et des solutions de redondance. Ainsi, même en cas de panne matérielle ou d'incidents, IBTool reste opérationnel, assurant une continuité de service essentielle pour la gestion des plaintes des clients.

La sécurité est un autre aspect crucial amélioré par l'adoption du cloud. Les services cloud offrent des fonctionnalités avancées telles que le chiffrement des données et la gestion des identités, renforçant ainsi la sécurité d'IBTool et garantissant la conformité avec les normes de protection des données. Enfin, la flexibilité et l'agilité offertes par les plateformes cloud permettent un développement, un test et un déploiement rapides de nouvelles fonctionnalités, accélérant ainsi l'innovation au sein de l'application.

2.2.2 APPLICATIONS CLOUD-NATIVE

Les applications cloud-native sont spécifiquement conçues pour tirer parti des capacités de l'informatique en nuage, en offrant une résilience, une évolutivité et une gestion améliorées. Pour IBTool, cela signifie repenser son architecture pour exploiter pleinement les services cloud, ce qui se traduit par une amélioration notable de ses performances et de sa fiabilité.

En adoptant une architecture en microservices, IBTool est divisé en plusieurs services indépendants, chacun pouvant être développé, déployé et mis à l'échelle séparément. Cette approche permet non seulement une plus grande flexibilité dans le développement, mais aussi une meilleure gestion des ressources. De plus, la conteneurisation avec Docker assure que chaque composant d'IBTool, qu'il s'agisse du frontend, du backend ou de la base de données, soit encapsulé dans un conteneur, garantissant un déploiement cohérent et portable à travers différents environnements.

L'intégration et le déploiement continus (CI/CD) sont mis en œuvre pour automatiser le processus de construction, de test et de déploiement des modifications de code. Cela permet d'assurer que chaque mise à jour d'IBTool est déployée de manière fluide et rapide, réduisant ainsi les interruptions de service. Enfin, l'orchestration dynamique, via des outils comme Amazon ECS, permet de gérer

efficacement le déploiement et la montée en charge des conteneurs d'IBTool, garantissant ainsi une utilisation optimale des ressources cloud.

En somme, l'adoption d'une architecture cloud-native pour IBTool non seulement améliore sa performance, mais assure également une meilleure gestion, une plus grande résilience, et une capacité à évoluer avec les besoins de l'entreprise.

2.3 CONCEPTION DE L'ARCHITECTURE CLOUD-NATIVE POUR IBTOOL

L'architecture cloud-native d'IBTool a été conçue pour exploiter pleinement les services AWS, assurant ainsi scalabilité, disponibilité, et sécurité. Cette architecture est composée de plusieurs composants intégrés, chacun jouant un rôle spécifique dans le déploiement et le fonctionnement de l'application. Le diagramme suivant offre une vue d'ensemble complète de l'architecture conçue pour IBTool, illustrant comment différents services AWS sont intégrés pour former un système robuste et efficace.

Le diagramme d'architecture illustre les composants clés suivants :

- **Service Frontend** : Déployé sous forme de conteneur Docker, ce composant est responsable de l'interface utilisateur d'IBTool. Il gère les interactions des utilisateurs et affiche les données récupérées depuis les services backend.
- **Service Backend** : Également déployé sous forme de conteneur Docker, le backend gère la logique métier et les points de terminaison API d'IBTool. Il traite les requêtes venant du frontend et interagit avec la base de données pour stocker et récupérer les informations.
- **Service de Base de Données** : MongoDB, déployé en tant que conteneur Docker, est utilisé pour le stockage des données. Il contient toutes les informations relatives aux plaintes des clients.

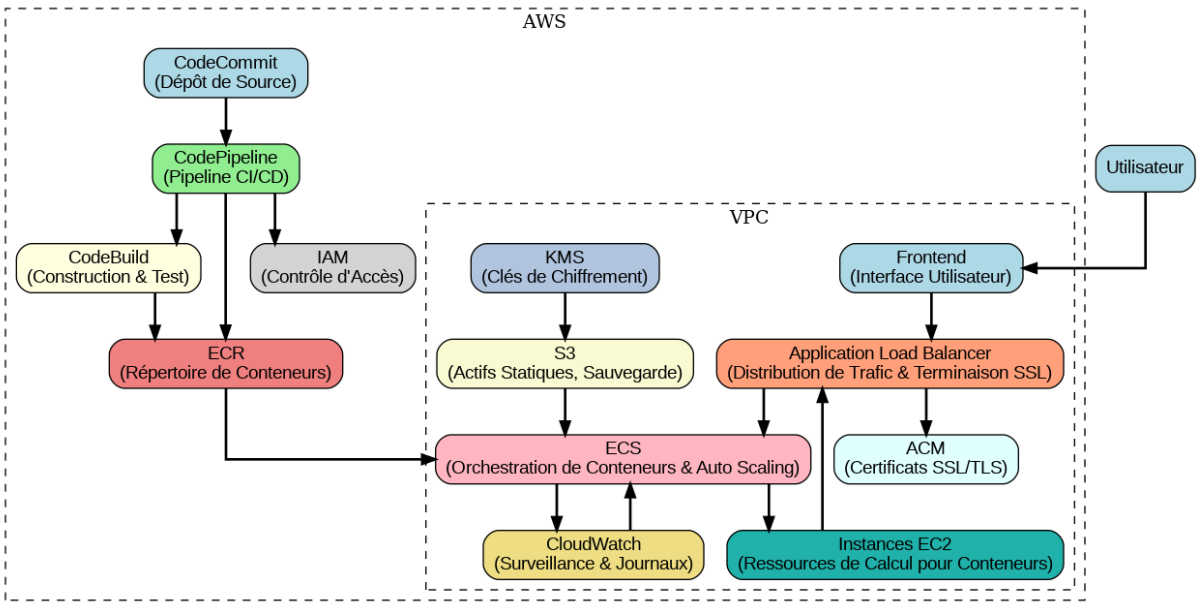

Figure 6 Diagramme d'architecture

- **Équilibrage de Charge** : Un Application Load Balancer (ALB) est mis en place pour distribuer le trafic entrant. Il assure une haute disponibilité en dirigeant le trafic uniquement vers les instances saines et gère également la terminaison SSL en utilisant des certificats gérés par ACM.
- **Interaction Utilisateur** : L'utilisateur final accède à IBTool via un navigateur web, interagissant principalement avec l'interface utilisateur pour soumettre et examiner les plaintes.
- **Orchestration des Conteneurs et Calcul** : Le service ECS (Elastic Container Service) gère le déploiement et la montée en charge des conteneurs Docker exécutant les services frontend, backend et base de données. ECS garantit que l'application peut évoluer en fonction de la demande et maintient l'état souhaité des services.
- **Instances EC2** : Ces instances fournissent les ressources de calcul nécessaires pour exécuter les tâches ECS. Elles sont gérées par des groupes d'Auto Scaling, qui ajustent automatiquement le nombre d'instances en fonction de la charge de travail.
- **Intégration et Déploiement Continus (CI/CD)** : Le processus CI/CD est automatisé via CodePipeline, qui intègre CodeCommit, CodeBuild et ECR. Les développeurs poussent leurs modifications de code vers CodeCommit, déclenchant ainsi le pipeline de déploiement.
- **Sécurité et Gestion des Accès** : IAM (Identity and Access Management) gère les contrôles d'accès et les permissions pour les ressources AWS, garantissant que seules les entités autorisées peuvent effectuer certaines actions. KMS (Key Management Service) chiffre les données sensibles stockées dans MongoDB et S3, assurant ainsi la sécurité et la conformité réglementaire. ACM (AWS Certificate Manager) gère les certificats SSL/TLS pour sécuriser les communications entre l'application et ses utilisateurs.
- **Stockage et Sauvegarde** : S3 (Simple Storage Service) est utilisé pour le stockage évolutif des actifs statiques, des sauvegardes, et d'autres données, garantissant une haute durabilité et disponibilité.
- **Surveillance et Journalisation** : CloudWatch surveille la performance et la santé de l'application. Il collecte des métriques et des logs provenant de divers services AWS, permettant une gestion proactive et un dépannage efficace.

2.3.1 APERÇU DU WORKFLOW

Le workflow global de l'architecture cloud-native d'IBTool inclut les étapes suivantes :

- **Développement et Pipeline CI/CD** : Les développeurs committent du code dans CodeCommit. CodePipeline déclenche ensuite CodeBuild pour compiler et tester le code, puis pousse les images Docker vers ECR. CodePipeline déploie ensuite ces nouvelles images sur les services ECS.
- **Déploiement et Scalabilité** : ECS exécute les conteneurs pour le frontend, le backend, et MongoDB. L'ALB route le trafic vers les tâches ECS appropriées en fonction des vérifications de santé et des règles de routage. Auto Scaling ajuste le nombre d'instances EC2 et de tâches ECS en fonction des métriques de CloudWatch.
- **Sécurité et Gestion des Données** : IAM contrôle l'accès aux ressources AWS, garantissant que seules les entités autorisées peuvent accéder aux données sensibles. KMS chiffre les données stockées dans MongoDB et S3, les protégeant ainsi contre tout accès non autorisé. ACM gère les certificats SSL/TLS pour sécuriser les communications.

- **Surveillance et Maintenance** : CloudWatch collecte et surveille les logs et les métriques provenant d'ECS, d'EC2 et d'autres services AWS. Des alarmes et des tableaux de bord sont configurés dans CloudWatch pour surveiller la performance et la santé de l'application.

2.3.2 DECOMPOSITION DES COMPOSANTS

Chaque composant de l'architecture joue un rôle crucial pour assurer que l'application fonctionne de manière fluide et efficace dans un environnement cloud. Les sous-sections suivantes détaillent les composants individuels et leurs interactions au sein de l'architecture cloud-native d'IBTool.

2.4 INTEGRATION DES SERVICES

2.4.1 INTEGRATION D'ECR AVEC ECS POUR IBTOOL

Dans le cadre du projet IBTool, l'intégration entre Amazon Elastic Container Registry (ECR) et Amazon Elastic Container Service (ECS) joue un rôle crucial pour assurer un déploiement fluide et une gestion efficace des services conteneurisés de l'application. Cette section détaille le flux de travail spécifique impliqué dans cette intégration, en se concentrant sur les étapes clés du stockage, de la récupération et du déploiement des images, ainsi que sur la gestion des mises à jour et des retours en arrière.

Le flux de travail entre ECR et ECS pour IBTool se décompose en trois phases principales : le stockage des images, la récupération des images et la gestion du déploiement. Ces processus sont illustrés par le diagramme de la figure 7.

2.4.2 STOCKAGE DES IMAGES

La première étape du flux de travail est la construction des images Docker pour les services frontend, backend et MongoDB d'IBTool. Ces images peuvent être construites localement par les développeurs ou automatiquement via le pipeline CI/CD. Ce pipeline CI/CD garantit que chaque image est construite, testée et validée avant d'être stockée. Une fois les images construites, elles sont poussées vers Amazon ECR, qui agit comme le dépôt central pour toutes les images Docker associées à IBTool. Cette étape est cruciale car elle centralise le stockage de toutes les images conteneurisées, les rendant facilement accessibles pour un déploiement dans différents environnements.

2.4.3 RECUPERATION DES IMAGES

La phase suivante consiste à récupérer les images depuis ECR. Amazon ECS extrait automatiquement les images Docker les plus récentes d'ECR chaque fois qu'il doit déployer ou mettre à jour un service au sein de l'application IBTool. Ce processus de récupération garantit que les versions les plus récentes des services frontend, backend et MongoDB sont déployées, exploitant les avantages de la conteneurisation pour des environnements cohérents et reproductibles.

2.4.4 GESTION DU DEPLOIEMENT

Une fois les images récupérées par ECS, elles sont déployées sur le cluster pour exécuter les divers services d'IBTool. ECS gère le déploiement des services pour le frontend, le backend et MongoDB, assurant que chaque service fonctionne de manière optimale. Un aspect essentiel de ce flux de travail est la capacité de mises à jour et de retours en arrière automatiques. Lorsqu'une nouvelle version d'un service est disponible, l'image Docker mise à jour est poussée vers ECR, et ECS déploie automatiquement la nouvelle version. Si des problèmes sont détectés après le déploiement, ECS permet de revenir facilement à une version stable précédente, minimisant ainsi les perturbations pour l'application IBTool.

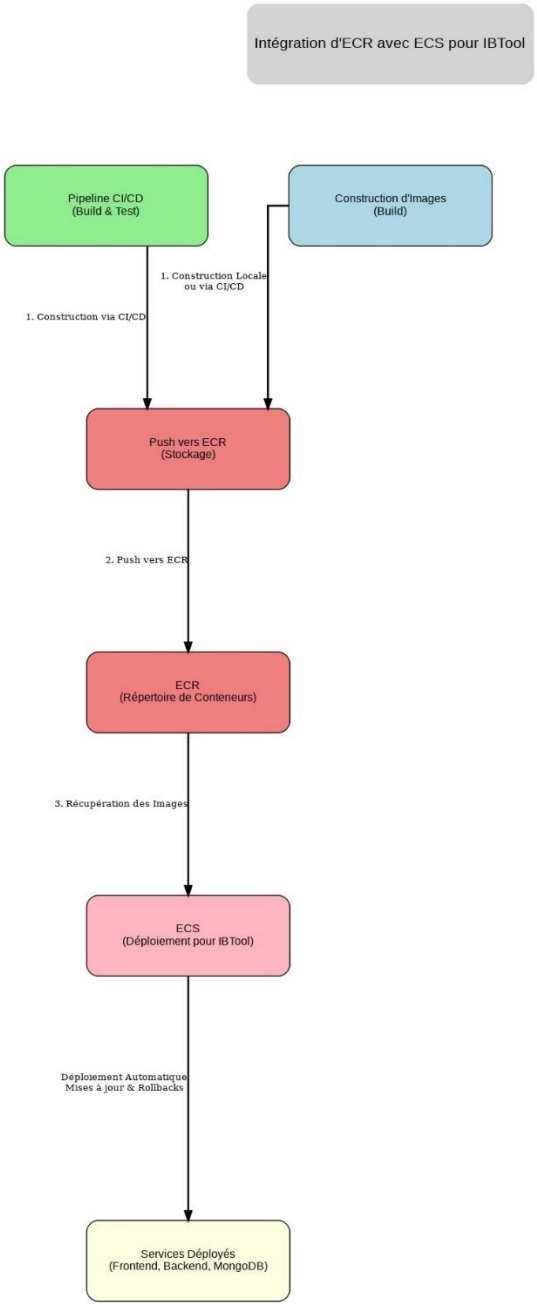

Figure 7 Intégration entre ECR et ECS

La figure 7 illustre le flux de travail d'intégration entre ECR et ECS pour l'application IBTool, montrant le processus de construction, de stockage, de récupération et de déploiement des images Docker, ainsi que la gestion des mises à jour et des retours en arrière.

2.4.5 COLLABORATION ENTRE ECS ET EC2 POUR IBTOOL

La collaboration entre Amazon Elastic Container Service (ECS) et Amazon Elastic Compute Cloud (EC2) est essentielle au déploiement et à la gestion réussis de l'application IBTool. Cette section décrit le flux de travail qui permet à ECS d'utiliser efficacement les instances EC2 pour fournir les ressources de calcul nécessaires à l'exécution des conteneurs Docker hébergeant les services IBTool.

Le flux de travail entre ECS et EC2 comprend trois phases clés : la gestion du cluster, la définition des tâches et le déploiement des services. Le flux de travail spécifique pour IBTool est détaillé dans la figure 8.

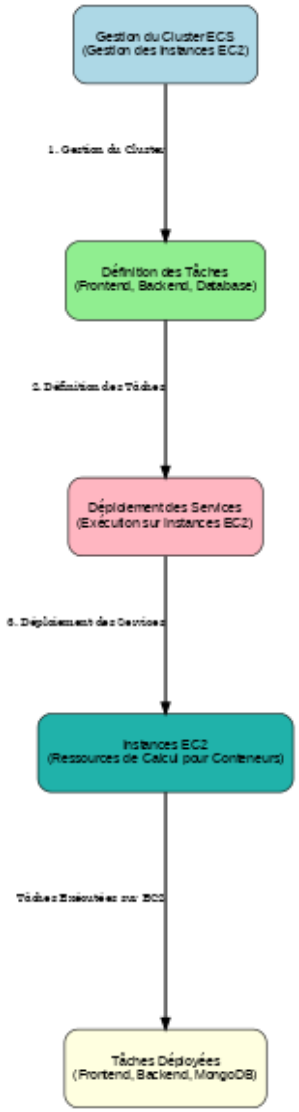

Figure 8 Collaboration entre ECS et EC2

2.4.6 GESTION DU CLUSTER

La première étape consiste en la gestion du cluster ECS, où ECS orchestre un cluster d'instances EC2. Ces instances EC2 agissent en tant que ressources de calcul qui hébergeront les conteneurs Docker d'IBTool. ECS gère le cycle de vie des instances EC2, s'assurant qu'elles sont correctement dimensionnées et disponibles pour gérer les services conteneurisés requis par IBTool. ECS ajuste automatiquement la mise à l'échelle des instances EC2 en fonction de la charge et de la demande de l'application, garantissant ainsi qu'IBTool dispose toujours de la capacité de calcul nécessaire pour gérer efficacement les demandes des utilisateurs.

2.4.7 DEFINITION DES TACHES

La phase de définition des tâches implique la spécification des tâches pour les services frontend, backend et base de données d'IBTool. Chaque tâche définit l'image Docker à utiliser, la quantité de CPU et de mémoire requise, ainsi que les configurations réseau nécessaires pour que les conteneurs fonctionnent efficacement. Pour IBTool, les définitions de tâches incluent :

- La tâche Frontend, qui gère l'interface utilisateur basée sur React.
- La tâche Backend, qui gère le serveur API Express.js.
- La tâche Base de Données, qui exécute le service MongoDB, où sont stockées toutes les données relatives aux plaintes des clients.

Ces définitions de tâches sont cruciales car elles fournissent à ECS les informations nécessaires pour déployer et gérer les conteneurs au sein des instances EC2.

2.4.8 DEPLOIEMENT DES SERVICES

Enfin, dans la phase de déploiement des services, ECS déploie les tâches définies sur les instances EC2 au sein du cluster. ECS s'assure que chaque tâche fonctionne comme prévu et surveille en continu leur état pour maintenir leur bonne santé. Pour IBTool, cela signifie que les services frontend, backend et MongoDB sont activement exécutés sur les instances EC2, ECS ajustant automatiquement la scalabilité des services en fonction de la demande des utilisateurs. Cette configuration permet à IBTool de maintenir une haute disponibilité et des performances optimales, quelles que soient les fluctuations de la charge de travail.

2.4.9 APPLICATION LOAD BALANCER (ALB) DANS IBTOOL

L'Application Load Balancer (ALB) joue un rôle crucial pour garantir que le trafic entrant vers IBTool est efficacement géré et traité de manière sécurisée. En répartissant les requêtes de manière équilibrée entre les services ECS, l'ALB améliore la disponibilité et la fiabilité de l'application, garantissant aux utilisateurs une performance constante.

Le flux de travail comprend plusieurs étapes clés :

- Distribution du Trafic : L'ALB reçoit les requêtes HTTP/HTTPS entrantes des utilisateurs et les dirige intelligemment vers les tâches ECS appropriées. Cela garantit que la charge est répartie de manière équilibrée, empêchant qu'une seule tâche ne soit submergée.
- Vérifications de Santé : Pour maintenir une haute disponibilité, l'ALB effectue en continu des vérifications de santé sur les tâches ECS. Si une tâche est jugée non saine, l'ALB redirige automatiquement le trafic vers les tâches saines, maintenant ainsi la fiabilité de l'application.
- Terminaison SSL : L'ALB gère la terminaison SSL, en utilisant des certificats gérés par AWS Certificate Manager (ACM). Cela décharge le traitement SSL des tâches ECS, réduisant leur charge et assurant une communication sécurisée entre l'utilisateur et IBTool.

Le diagramme ci-dessus illustre comment l'ALB fonctionne au sein de l'architecture d'IBTool, montrant le flux de trafic de l'utilisateur vers les tâches ECS, les vérifications de santé effectuées par l'ALB, et le processus de terminaison SSL.

2.4.10 GESTION DE LA SECURITE AVEC IAM, KMS ET ACM

La sécurité est une priorité pour IBTool, et AWS offre une suite complète d'outils pour gérer le contrôle d'accès, le chiffrement et les communications sécurisées. Le diagramme suivant illustre comment IAM, KMS et ACM sont intégrés dans l'architecture IBTool pour garantir une sécurité robuste.

IAM est utilisé pour définir des rôles et des politiques qui contrôlent l'accès aux ressources AWS. Des permissions spécifiques sont attribuées aux tâches ECS, leur permettant d'interagir de manière sécurisée avec d'autres services tout en respectant le principe du moindre privilège.

KMS est responsable du chiffrement des données sensibles stockées dans la base de données MongoDB et dans le stockage S3 d'IBTool. En gérant les clés de chiffrement de manière sécurisée, KMS garantit que seules les entités autorisées peuvent accéder aux données chiffrées, les protégeant ainsi contre tout accès non autorisé. ACM fournit et gère les certificats SSL/TLS pour IBTool, garantissant que toutes les communications entre les utilisateurs et l'application sont sécurisées. La fonction de renouvellement automatique des certificats d'ACM assure que les connexions sécurisées sont maintenues sans interruption.

Ce diagramme montre visuellement comment IAM, KMS et ACM travaillent ensemble pour sécuriser l'application IBTool, garantissant que l'accès est contrôlé, les données sont chiffrées et la communication reste sécurisée. Cette intégration est essentielle pour maintenir la confidentialité, l'intégrité et la disponibilité des données et services d'IBTool.

2.4.11 SURVEILLANCE ET JOURNALISATION AVEC CLOUDWATCH

Amazon CloudWatch offre des capacités complètes de surveillance et de journalisation, permettant de suivre la performance et l'état de santé d'IBTool. Le diagramme suivant illustre comment CloudWatch est intégré dans l'architecture d'IBTool pour assurer une surveillance continue et une gestion proactive.

CloudWatch collecte des métriques provenant des tâches ECS, des instances EC2, et d'autres services AWS. Ces métriques fournissent des informations sur l'utilisation des ressources telles que l'utilisation du CPU, la mémoire, et la latence des requêtes.

Des tableaux de bord sont créés dans CloudWatch pour visualiser ces métriques clés. Ces visualisations permettent de surveiller la santé globale de l'application en temps réel.

CloudWatch collecte et stocke également les journaux applicatifs, les journaux des tâches ECS, et les journaux systèmes. Ces journaux sont essentiels pour le dépannage et la résolution des problèmes.

Des alertes sont configurées dans CloudWatch pour notifier l'équipe en cas de problèmes de performance ou d'anomalies. Cela permet de réagir rapidement et d'assurer la continuité du service.

Ce diagramme montre comment CloudWatch est utilisé pour surveiller les performances d'IBTool et collecter des journaux détaillés, garantissant ainsi que l'application fonctionne de manière optimale et que tout problème potentiel est détecté et traité en temps opportun.

2.5 OUTILS ET TECHNOLOGIES UTILISES

Dans cette section, les aspects techniques du développement d'IBTool sont expliqués en détail. Le développement d'IBTool comprend à la fois le frontend et le backend, chacun utilisant des technologies spécifiques pour répondre aux besoins du projet. De plus, l'intégration des services AWS et des outils de gestion d'infrastructure comme Terraform joue un rôle clé dans la migration vers une architecture cloud-native.

Les utilisateurs d'IBTool accèdent à l'application via un navigateur web. Ils doivent d'abord être authentifiés pour accéder au réseau de GE Healthcare, puis se connecter à IBTool avec leurs identifiants. Le serveur IBTool héberge les services frontend et backend, qui sont déployés sous forme de conteneurs Docker. La migration vers le cloud s'appuie sur AWS pour gérer l'infrastructure, offrant ainsi scalabilité, sécurité et résilience.

2.5.1 FRONTEND

La partie frontend d'IBTool se concentre sur l'interface utilisateur et la gestion des requêtes envoyées au backend pour récupérer les données à afficher. Voici les technologies principales utilisées :

- **HTML/CSS** : Pour la structure et le style des pages web.
- **JavaScript** : Comme langage de programmation principal pour le frontend.
- **NodeJS** : Pour exécuter JavaScript côté serveur, permettant une interaction fluide entre le frontend et le backend.
- **DevExtreme** : Une toolkit JavaScript basée sur jQuery, utilisée pour construire les composants de l'interface utilisateur.
- **Webpack** : Utilisé pour regrouper les modules JavaScript et optimiser le code pour le déploiement.

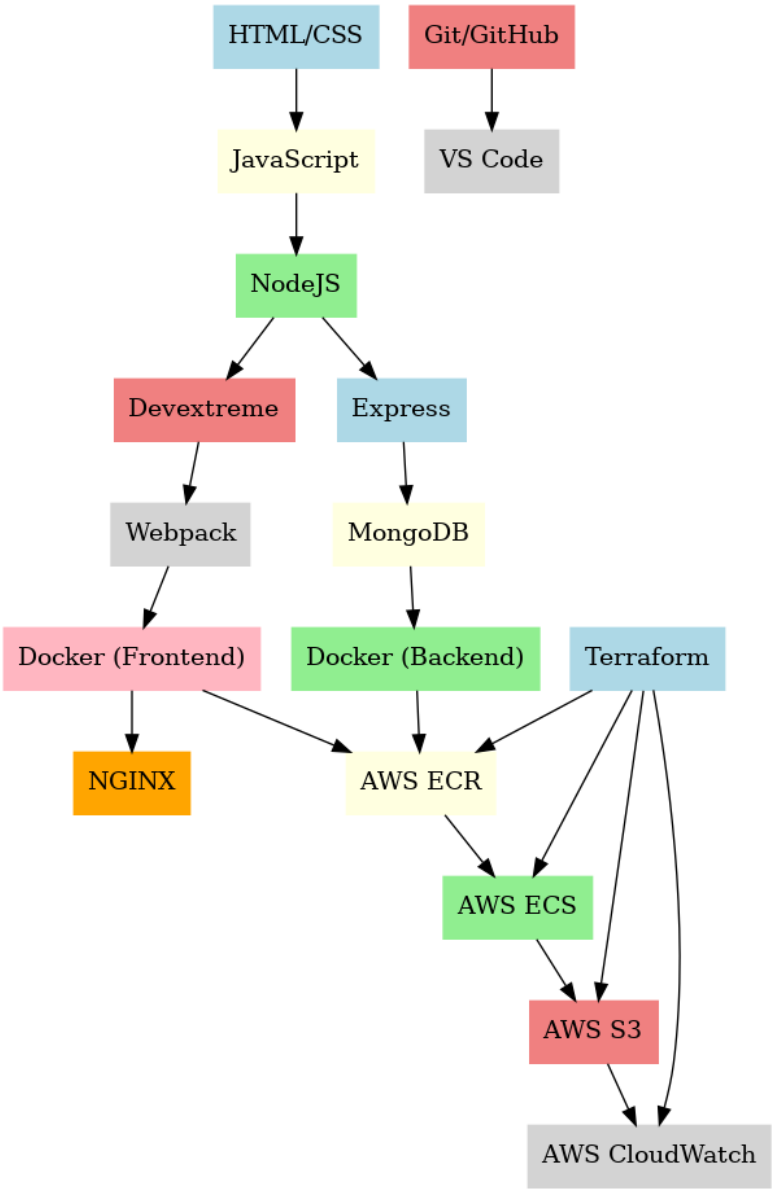

Figure 9 Outils et Technologies Utilisés dans IBTool Cloud.

- **Docker** : Le frontend est conteneurisé à l'aide de Docker, assurant un déploiement cohérent sur différents environnements.
- **NGINX** : Serveur web utilisé pour lancer l'application frontend dans un environnement local ou en production

2.5.2 BACKEND

Le backend d'IBTool est responsable du traitement des requêtes HTTP et de la gestion des opérations sur la base de données MongoDB. Les technologies utilisées incluent :

- **NodeJS** et **Express** : Pour créer une API REST qui traite les requêtes des utilisateurs et interagit avec la base de données.
- **MongoDB** : Une base de données NoSQL choisie pour sa flexibilité et sa capacité à gérer des données non structurées, adaptée aux besoins d'IBTool.
- **Docker** : Comme pour le frontend, le backend est conteneurisé pour faciliter le déploiement et la gestion des versions.

2.5.3 INTEGRATION AWS ET CLOUD

Avec la migration d'IBTool vers le cloud, plusieurs services AWS sont intégrés pour améliorer les performances et la gestion de l'application :

- **Terraform** : Outil d'infrastructure en tant que code (IaC) utilisé pour orchestrer et déployer l'infrastructure cloud sur AWS.
- **AWS ECR (Elastic Container Registry)** : Stocke les images Docker des services frontend, backend et MongoDB.
- **AWS ECS (Elastic Container Service)** : Gère le déploiement et la montée en charge des conteneurs Docker sur les instances EC2.
- **AWS S3** : Utilisé pour le stockage des actifs statiques, des sauvegardes et d'autres données critiques.
- **AWS CloudWatch** : Permet la surveillance des performances et la collecte des logs, fournissant des informations essentielles pour la gestion proactive de l'application.

2.5.4 GESTION DES VERSIONS ET COLLABORATION

Pour la gestion des versions et la collaboration entre les développeurs :

- **Git/GitHub** : Utilisé pour le contrôle de version et les revues de code.
- **Visual Studio Code** : L'éditeur de code privilégié pour le développement de l'application.

Le diagramme ci-dessous illustre la connexion entre les différents outils et technologies utilisés dans le développement et la gestion d'IBTool, à la fois pour l'infrastructure on-premises et la nouvelle architecture cloud-native.Pour la base de données, j'ai choisi MongoDB (une base de données NoSQL) pour les raisons suivantes : Tout d'abord, les données que nous devons stocker (par exemple, la liste des COM/CSO) contiennent de nombreux champs, mais certains ou plusieurs d'entre eux peuvent être vides. L'utilisation d'une base de données de relations comme MySql peut gaspiller de l'espace tandis qu'une base de données basée sur des documents comme MongoDB ne stocke que les informations remplies et économise de l'espace. Deuxièmement, MongoDB est évolutif et plus rapide. Troisièmement, il existe des outils et des libraires javascript qui soutiennent MongoDB et facilitent le development. Le schéma de la base de données est illustré à la Fig. 9.

De nombreux champs de la collection Record ne sont pas affichés en raison d'un manque d'espace. Les collections "User" et "Record" sont connectées à l'aide de COM-ID / CSO-ID. La collection "Shared" stocke toutes les informations qui ne sont pas destinées à un utilisateur spécifique (par exemple, la

dernière mise à jour COM). La collection "Valid SSO" stocke la liste des utilisateurs SSO autorisés à s'enregistrer et à utiliser IBTool.

3. MISE EN ŒUVRE ET DEPLOIEMENT

3.1 L'IBTOOL ON-PREMISES

3.1.1 DEVELOPPEMENT

Le développement d'IBTool a été une entreprise cruciale pour GE Healthcare, visant à améliorer la gestion des plaintes des clients concernant les produits de l'entreprise. Ce projet a été structuré en deux parties principales : le développement du front-end, hébergé dans le répertoire ibtool-fe, et celui du back-end, situé dans le répertoire ibtool-be. Chacune de ces composantes a été conçue avec soin pour répondre à des besoins spécifiques en matière d'interface utilisateur, de gestion des données, et de performance globale de l'application.

3.1.1.1 LE FRONT-END D'IBTOOL

Le front-end d'IBTool, contenu dans le répertoire ibtool-fe, constitue l'interface utilisateur de l'application. Il est essentiel car c'est par ce biais que les utilisateurs interagissent avec le système, soumettant et consultant les plaintes des clients. Le choix des technologies pour le développement du front-end a été guidé par plusieurs considérations.

Tout d'abord, l'objectif principal était de créer une interface intuitive et réactive, capable de fonctionner efficacement sur différents navigateurs et appareils. Pour cela, des technologies web modernes telles que **HTML**, **CSS**, et **JavaScript** ont été utilisées. **Node.js** et **npm** ont été employés pour gérer les dépendances et exécuter JavaScript côté serveur, tandis que **Devextreme**, une boîte à outils JavaScript basée sur **jQuery**, a été intégrée pour simplifier le développement de composants d'interface utilisateur complexes et interactifs.

Le choix de ces technologies permet de créer une expérience utilisateur fluide, où les données sont rapidement récupérées et affichées à l'écran. En outre, **Webpack** a été utilisé pour empaqueter les fichiers front-end, garantissant ainsi une meilleure performance grâce à la réduction de la taille des fichiers et à l'optimisation du chargement des ressources.

3.1.1.2 LE BACK-END D'IBTOOL

Le back-end, situé dans le répertoire ibtool-be, est le moteur de l'application, responsable du traitement des requêtes des utilisateurs, de la gestion des données, et de l'application des règles métier. Cette partie du développement a été réalisée en **Node.js** avec le framework **Express.js**, ce qui a permis de créer une API RESTful robuste et extensible.

L'utilisation d'**Express.js** a facilité la gestion des requêtes HTTP, tout en offrant un cadre léger mais puissant pour construire des routes et des middlewares nécessaires au bon fonctionnement de l'application. Le back-end est également responsable de la communication avec la base de données **MongoDB**, où toutes les informations relatives aux plaintes des clients sont stockées. MongoDB, une base de données NoSQL, a été choisie pour sa capacité à gérer des données non structurées et pour sa flexibilité, répondant ainsi aux besoins de stockage des différentes catégories de plaintes, qui peuvent varier en structure et en contenu.

3.1.2 PROCESSUS DE DEVELOPPEMENT ET INTEGRATION

Le développement d'IBTool a suivi une méthodologie agile, permettant une adaptation rapide aux besoins changeants et une intégration continue des nouvelles fonctionnalités. Chaque modification apportée au code, qu'il s'agisse du front-end ou du back-end, était rigoureusement testée avant d'être intégrée dans la branche principale du dépôt Git. Cette approche a permis de maintenir un haut niveau de qualité du code tout en minimisant les régressions. Un aspect clé du développement a été l'utilisation de **Docker** pour containeriser les deux composantes de l'application. Les conteneurs Docker assurent que l'application fonctionne de manière cohérente, quel que soit l'environnement dans lequel elle est déployée, que ce soit sur un serveur de développement local ou sur un serveur de production on-premises.

3.1.3 OBJECTIFS ET RESULTATS

L'objectif principal du développement d'IBTool était de créer une application capable de centraliser toutes les informations relatives aux plaintes des clients, facilitant ainsi leur gestion et leur résolution.

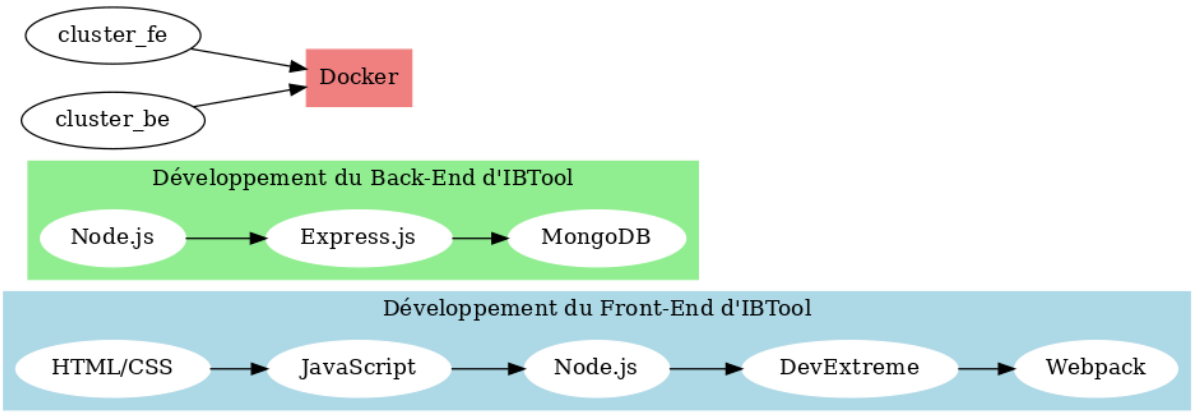

Figure 10 Processus de Développement d'IBTool

Le choix des technologies et des méthodologies utilisées a permis de développer une application évolutive, performante, et sécurisée, capable de répondre aux exigences élevées du secteur de la santé. Le développement d'IBTool représente une étape significative pour GE Healthcare dans son

engagement à améliorer la satisfaction des clients et à maintenir la qualité de ses produits. Ce projet illustre comment l'adoption de technologies modernes et de pratiques de développement rigoureuses peut conduire à la création d'applications robustes et fiables, prêtes à répondre aux défis actuels et futurs de l'entreprise.

3.1.4 DEPLOIEMENT

Le déploiement d'IBTool dans un environnement on-premises chez GE Healthcare a été conçu pour répondre à des besoins spécifiques de gestion des plaintes clients tout en s'intégrant parfaitement aux infrastructures existantes. L'approche on-premises offre un contrôle total sur l'infrastructure, la sécurité, et les processus de maintenance, ce qui est crucial pour une entreprise opérant dans le domaine sensible de la santé. Toutefois, cette méthode présente également des défis en termes de maintenance, de mise à jour, et de scalabilité, que cette section s'efforce de détailler.

3.1.4.1 POURQUOI UN DEPLOIEMENT ON-PREMISES ?

L'une des principales raisons pour lesquelles GE Healthcare a choisi un déploiement on-premises pour IBTool est le besoin de maintenir un contrôle strict sur les données sensibles des patients. Le secteur de la santé est régi par des réglementations rigoureuses, telles que le RGPD en Europe et HIPAA aux États-Unis, qui imposent des exigences strictes en matière de protection des données. En conservant le déploiement d'IBTool sur des serveurs internes, GE Healthcare peut s'assurer que toutes les données critiques restent sous son contrôle direct, réduisant ainsi les risques liés à la confidentialité et à la sécurité des informations.

3.1.4.2 LES TECHNOLOGIES UTILISEES

Le déploiement d'IBTool on-premises repose sur une combinaison de technologies modernes pour garantir la robustesse et la fiabilité du système. L'application est composée de plusieurs composants, chacun ayant un rôle spécifique dans l'ensemble du processus.

- **Docker** : Docker est utilisé pour containeriser les différentes parties de l'application, y compris le frontend, le backend, et la base de données MongoDB. Cela permet d'assurer une portabilité et une cohérence des environnements, facilitant le déploiement et la gestion des versions.
- **Node.js et Express.js** : Le backend d'IBTool est développé en Node.js avec Express.js, offrant un environnement rapide et léger pour gérer les requêtes HTTP et interagir avec la base de données.
- **MongoDB** : En tant que base de données NoSQL, MongoDB est choisie pour sa capacité à gérer des ensembles de données non structurés ou semi-structurés, ce qui est essentiel pour les informations de plaintes clients qui peuvent varier considérablement d'un cas à l'autre.
- **Docker Compose** : Utilisé pour orchestrer les différents conteneurs Docker, Docker Compose simplifie le processus de mise en place de l'application, en veillant à ce que tous les services démarrent dans le bon ordre et puissent interagir correctement.

3.1.4.3 PROCESSUS DE DEPLOIEMENT

Le processus de déploiement d'IBTool sur les serveurs de GE Healthcare implique plusieurs étapes clés. Tout d'abord, les images Docker du frontend, du backend, et de la base de données sont créées et testées en local. Ces images sont ensuite compressées et transférées sur les serveurs internes de GE Healthcare. Une fois sur le serveur, les images sont décompressées et chargées dans Docker pour être prêtes à l'exécution.

Le fichier docker-compose.yml est essentiel pour ce déploiement. Ce fichier contient les instructions nécessaires pour orchestrer les conteneurs Docker, y compris les liens entre les services, les volumes de données, et les paramètres réseau. Le déploiement se fait simplement en exécutant la commande docker-compose up, qui lance tous les services dans les conteneurs Docker de manière cohérente.

3.1.4.4 SAUVEGARDE ET RESTAURATION DE LA BASE DE DONNEES

La gestion des données étant critique pour IBTool, des procédures spécifiques ont été mises en place pour la sauvegarde et la restauration de la base de données MongoDB. Les outils mongodump et mongorestore sont utilisés pour créer des instantanés de la base de données et les restaurer si nécessaire. Ces processus garantissent que les données sont protégées contre les pertes éventuelles et peuvent être récupérées rapidement en cas d'incident.

Le diagramme ci-dessous illustre le processus global de déploiement d'IBTool dans l'environnement on-premises de GE Healthcare

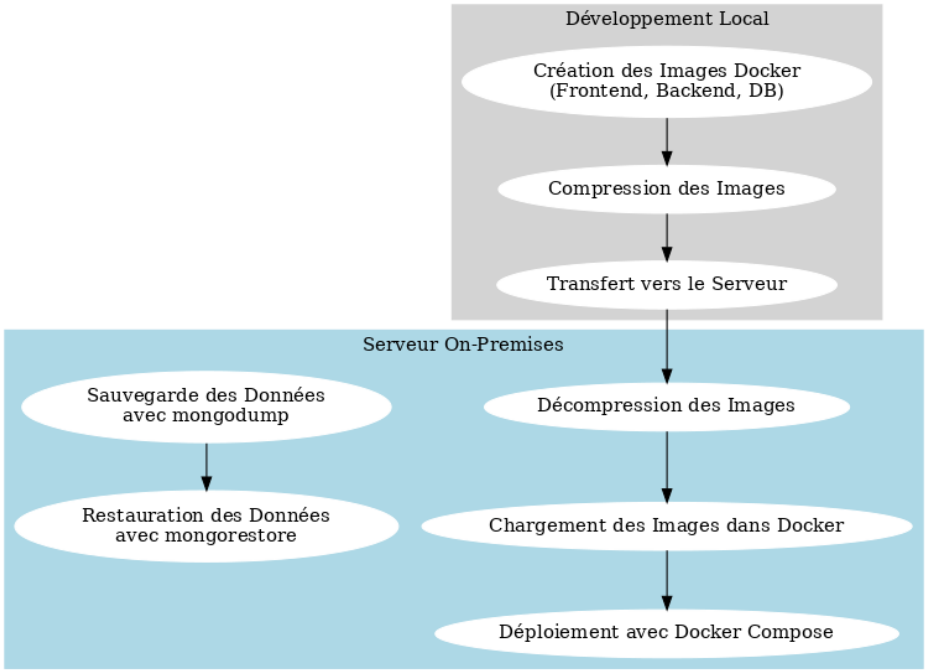

Figure 11 Processus de Déploiement d'IBTool On-Premises

3.1.4.5 AVANTAGES ET LIMITES

Le déploiement on-premises d'IBTool offre plusieurs avantages, notamment un contrôle total sur l'infrastructure, une sécurité accrue des données, et une conformité simplifiée avec les réglementations locales. Cependant, cette approche présente également des limitations, telles qu'une scalabilité réduite par rapport aux solutions cloud, des coûts de maintenance plus élevés, et une complexité accrue dans la gestion de l'infrastructure..

Ce choix de déploiement reflète un équilibre entre la nécessité de sécurité et de contrôle, et les défis techniques liés à la gestion d'une infrastructure on-premises. À mesure que les exigences de l'application évoluent, des considérations pour une éventuelle migration vers une solution cloud peuvent être réévaluées pour tirer parti de la flexibilité et de la scalabilité offertes par le cloud computing.

3.2 L'IBTOOL CLOUD-NATIVE

3.2.1 INFRASTRUCTURE EN TANT QUE SERVICE

Dans le cadre de la migration d'IBTool vers une architecture cloud-native, l'utilisation de l'infrastructure en tant que service (IaaS) a été cruciale pour automatiser la gestion et le déploiement de l'infrastructure sur Amazon Web Services (AWS). Le recours à l'infrastructure en tant que code (IaC), en particulier via Terraform, a permis de définir, déployer, et gérer l'infrastructure cloud de manière programmable et reproductible.

3.2.1.1 POURQUOI L'INFRASTRUCTURE EN TANT QUE CODE ?

L'infrastructure en tant que code (IaC) permet de décrire l'infrastructure nécessaire sous forme de fichiers de configuration, rendant ainsi le processus de gestion de l'infrastructure non seulement plus rapide, mais aussi moins sujet aux erreurs. Pour IBTool, cela signifie une gestion plus efficace des ressources cloud, une réduction des risques d'incohérences entre les environnements de développement, de test et de production, et une capacité accrue à automatiser les déploiements.

L'utilisation de Terraform, un outil populaire pour l'IaC, a permis de définir toutes les ressources AWS nécessaires pour IBTool dans des fichiers de configuration. Ces ressources incluent les instances EC2, les clusters ECS, les équilibrages de charge (ALB), les groupes de sécurité, et les dépôts ECR.

3.2.1.2 COMMENT TERRAFORM A ETE UTILISE POUR IBTOOL ?

Terraform a joué un rôle central dans la mise en place de l'infrastructure cloud d'IBTool. Les configurations Terraform décrivent chaque composant de l'infrastructure sous forme de code, permettant ainsi une gestion efficace et un déploiement automatisé.

1. **Définition des ressources :** Les fichiers de configuration Terraform définissent les ressources nécessaires, telles que les instances EC2, les clusters ECS, et les dépôts ECR. Chaque ressource est codée avec ses paramètres spécifiques, garantissant une configuration cohérente.
2. **Automatisation des déploiements :** Avec Terraform, le processus de déploiement de l'infrastructure est entièrement automatisé. Un simple fichier de configuration peut déployer ou mettre à jour l'ensemble de l'infrastructure en une seule commande.
3. **Gestion des versions et collaboration :** Terraform permet de suivre les modifications apportées à l'infrastructure au fil du temps, facilitant ainsi la gestion des versions. De plus, les configurations peuvent être stockées dans un système de contrôle de version comme Git, ce qui simplifie la collaboration entre les équipes.

3.2.1.3 DIAGRAMME DE L'INFRASTRUCTURE

Le diagramme ci-dessous illustre l'architecture d'infrastructure d'IBTool déployée via Terraform sur AWS, mettant en évidence les interactions entre les différents services AWS.

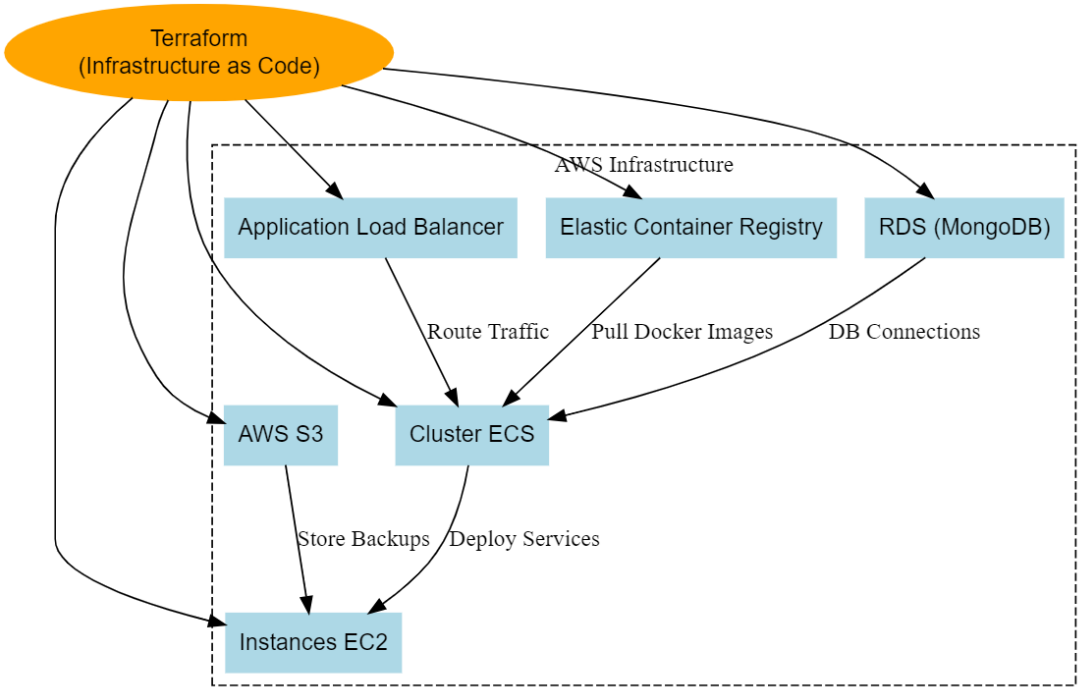

Figure 12 Diagramme de l'Infrastructure AWS d'IBTool Déployée via Terraform

3.2.1.4 BENEFICES DE L'IAC POUR IBTOOL

L'adoption de Terraform et de l'IaC a apporté plusieurs avantages clés pour IBTool :

1. **Réduction du Temps de Déploiement :** Les processus de déploiement, qui autrefois prenaient des jours, peuvent désormais être réalisés en quelques minutes grâce à l'automatisation.

2. **Cohérence des Environnements :** Grâce à Terraform, chaque environnement (développement, test, production) peut être déployé de manière identique, éliminant ainsi les erreurs dues à des configurations manuelles.
3. **Scalabilité et Flexibilité :** L'IaC permet une adaptation rapide de l'infrastructure aux besoins changeants, en facilitant la montée en charge ou la réduction des ressources en fonction des demandes.
4. **Optimisation des Coûts :** L'infrastructure peut être ajustée en temps réel pour éviter le surprovisionnement, ce qui permet de mieux gérer les coûts.

En conclusion, l'infrastructure en tant que service via Terraform a été un élément clé du succès de la migration d'IBTool vers le cloud. Elle a permis d'automatiser et d'optimiser les déploiements tout en garantissant une cohérence et une flexibilité maximales.

3.2.2 CONFIGURATION DU PIPELINE CI/CD

3.2.2.1 MISE EN PLACE DU REFERENTIEL CODECOMMIT

Amazon CodeCommit est utilisé pour héberger le code source d'IBTool, offrant un service de contrôle de version sécurisé et évolutif. Le référentiel créé au sein de CodeCommit sert de centre névralgique pour tout le code source d'IBTool, y compris les composants frontend et backend ainsi que les fichiers de configuration essentiels. Le processus commence par la création d'un nouveau référentiel spécifiquement conçu pour stocker la base de code d'IBTool. Une fois le référentiel établi, le code source existant y est transféré, garantissant ainsi que l'ensemble du code est maintenu dans un référentiel unique et cohérent.

Pour sécuriser ce référentiel, des politiques IAM sont mises en place pour contrôler l'accès. Ces politiques garantissent que seuls les développeurs autorisés peuvent apporter des modifications au code, protégeant ainsi ce dernier contre toute modification non autorisée. Cette configuration offre à IBTool un contrôle centralisé du code source, assurant une source unique de vérité pour sa base de code tout en sécurisant l'accès grâce à des protocoles de gestion des accès stricts.

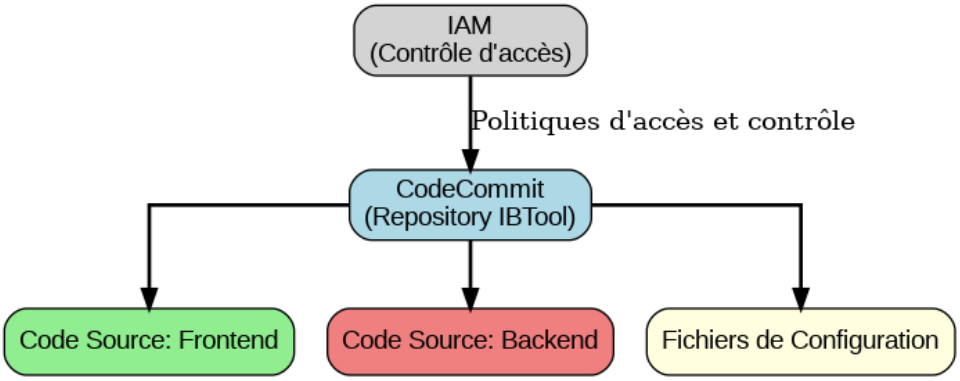

Figure 13 Contrôle d'accès

Cette architecture permet non seulement de simplifier la gestion du code mais également de renforcer la sécurité, garantissant que la base de code d'IBTool est bien protégée et facilement gérable.

3.2.2.2 AUTOMATISATION DES CONSTRUCTIONS AVEC CODEBUILD

AWS CodeBuild joue un rôle essentiel dans l'automatisation du processus de construction pour IBTool. L'automatisation commence par la création d'un fichier `buildspec.yml` au sein du référentiel CodeCommit. Ce fichier définit les commandes de construction et spécifie l'environnement dans lequel le processus de construction se déroulera. Une fois les spécifications de construction définies, un projet CodeBuild est configuré, le reliant au code source d'IBTool dans CodeCommit. La configuration du projet comprend la mise en place de l'environnement, qui, dans ce cas, utilise Docker pour créer et gérer l'environnement de construction.

À chaque modification du code dans le référentiel, CodeBuild est automatiquement déclenché pour compiler le code le plus récent et exécuter des tests afin de s'assurer qu'il fonctionne comme prévu. Ce processus aboutit à la création d'images Docker, essentielles pour les étapes de déploiement suivantes. L'intégration de CodeBuild dans le pipeline CI/CD garantit que le code est systématiquement construit et testé, réduisant ainsi le risque d'introduire des erreurs dans le processus de déploiement.

Figure 14 Automatisation de build

Ce processus de construction automatisé garantit non seulement l'intégrité du code d'IBTool, mais soutient également l'intégration continue, facilitant la gestion des mises à jour fréquentes et des améliorations.

3.2.2.3 AUTOMATISATION DU DEPLOIEMENT AVEC CODEPIPELINE

AWS CodePipeline est utilisé pour automatiser le processus de déploiement de bout en bout pour IBTool, depuis le moment où le code est validé jusqu'à son déploiement en production. Le pipeline est structuré en plusieurs étapes, chacune représentant une phase critique du processus de déploiement.

Le pipeline commence par l'étape **Source**, qui est connectée au référentiel CodeCommit d'où le code le plus récent est extrait. Ensuite, l'étape **Build** s'intègre à CodeBuild pour compiler le code, exécuter les tests nécessaires, et pousser les images Docker résultantes vers l'Elastic Container Registry (ECR). La dernière étape, **Deploy**, est responsable du déploiement de ces images Docker sur ECS, où elles sont orchestrées selon des définitions de tâches et des configurations de service prédéfinies.

L'automatisation fournie par CodePipeline réduit le besoin d'intervention manuelle, permettant des déploiements rapides et fiables de nouvelles fonctionnalités et de corrections de bugs. Ce processus rationalisé est essentiel pour maintenir l'agilité et la réactivité d'IBTool en environnement de production.

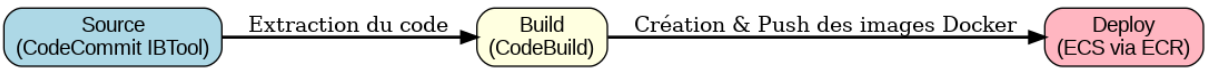

Figure 15 Automatisation du Déploiement avec CodePipeline pour IBTool

En automatisant l'ensemble du processus de déploiement, IBTool bénéficie de cycles de publication plus rapides et d'un pipeline de déploiement d'applications plus fiable, garantissant que les mises à jour sont livrées rapidement et efficacement.

3.2.3 GESTION DES CONTENEURS

3.2.3.1 STOCKAGE DES IMAGES DE CONTENEURS DANS ECR

Pour gérer les conteneurs de manière efficace, IBTool utilise Amazon Elastic Container Registry (ECR) pour stocker les images Docker des services frontend, backend et MongoDB. La configuration a été réalisée en créant des référentiels distincts pour chaque service au sein d'ECR. Ces référentiels permettent de centraliser le stockage des images Docker, garantissant ainsi leur accessibilité et leur sécurité. Les images construites via CodeBuild sont ensuite étiquetées et poussées vers ces référentiels à l'aide de commandes Docker spécifiques.

La gestion des versions des images Docker dans ECR est également une étape cruciale du processus. En utilisant les fonctionnalités intégrées d'ECR, il est possible de gérer différentes versions des images, facilitant ainsi les retours en arrière si nécessaire. Cela assure non seulement une meilleure gestion des versions, mais aussi une flexibilité accrue en cas de besoin de modifications rapides ou de corrections de bugs.

L'utilisation d'Amazon ECR permet à IBTool de bénéficier d'un stockage centralisé pour toutes les images Docker, assurant leur sécurité et leur accessibilité. En outre, la gestion des versions facilite le déploiement continu tout en offrant la possibilité de revenir facilement à des versions antérieures en cas de problème.

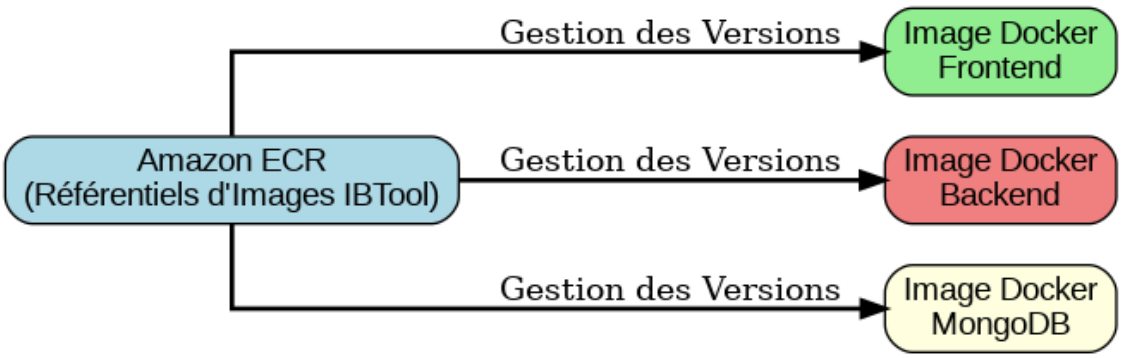

Figure 16 Gestion des Référentiels ECR pour IBTool

3.2.3.2 DEPLOIEMENT DES CONTENEURS AVEC ECS

Amazon Elastic Container Service (ECS) est l'outil principal utilisé pour orchestrer le déploiement et la gestion des conteneurs Docker pour IBTool. Une fois les images Docker stockées dans ECR, elles sont déployées sur un cluster ECS, spécialement configuré pour héberger ces services. Le processus

commence par la création d'un cluster ECS, où des définitions de tâches sont ensuite établies pour spécifier les images Docker à utiliser, les besoins en ressources et les variables d'environnement.

Les services ECS sont ensuite configurés pour chaque tâche, incluant des politiques de mise à l'échelle et d'équilibrage de charge. Cette orchestration permet une gestion automatisée des conteneurs, garantissant que les services d'IBTool sont toujours disponibles et peuvent s'adapter dynamiquement à la demande.

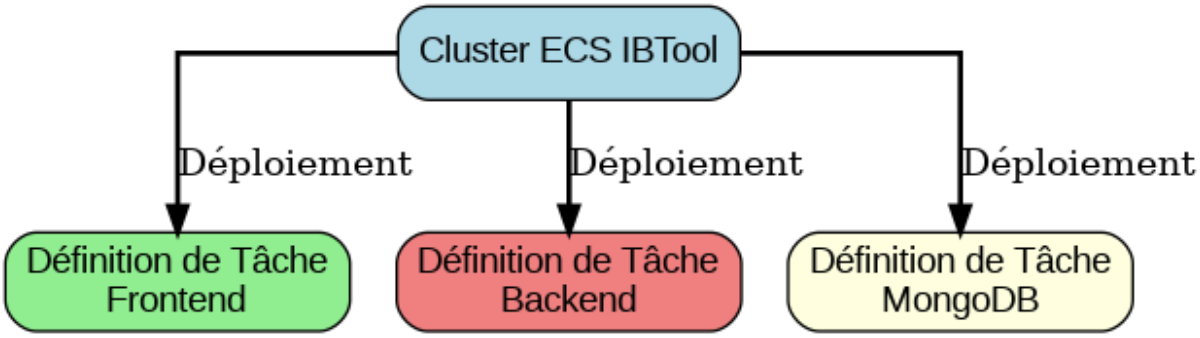

Figure 17 Cluster ECS et Définitions de Tâches pour IBTool

Cette configuration permet à IBTool d'automatiser le déploiement et la mise à l'échelle des conteneurs Docker, garantissant ainsi une performance optimale en ajustant dynamiquement le nombre de tâches en fonction de la demande.

3.2.3.3 MISE A L'ECHELLE ET GESTION DES INSTANCES EC2

Les instances EC2 fournissent la puissance de calcul nécessaire pour exécuter les tâches ECS d'IBTool. Le déploiement des instances EC2 est configuré au sein du cluster ECS, avec des types d'instances et des groupes de sécurité adaptés aux besoins spécifiques des services d'IBTool. Une fois les instances lancées, des groupes d'Auto Scaling sont mis en place pour ajuster automatiquement le nombre d'instances en fonction de la charge de travail.

Pour garantir une surveillance continue, Amazon CloudWatch est utilisé pour monitorer les performances des instances EC2. Des alarmes sont configurées pour notifier l'équipe en cas de dépassement des seuils de CPU, de mémoire ou d'autres métriques critiques, permettant ainsi une gestion proactive des ressources.

Grâce à la mise en place de groupes d'Auto Scaling et à la surveillance continue via CloudWatch, IBTool bénéficie de ressources de calcul évolutives qui s'ajustent automatiquement en fonction de la demande, tout en optimisant les coûts en réduisant le nombre d'instances durant les périodes de faible activité.

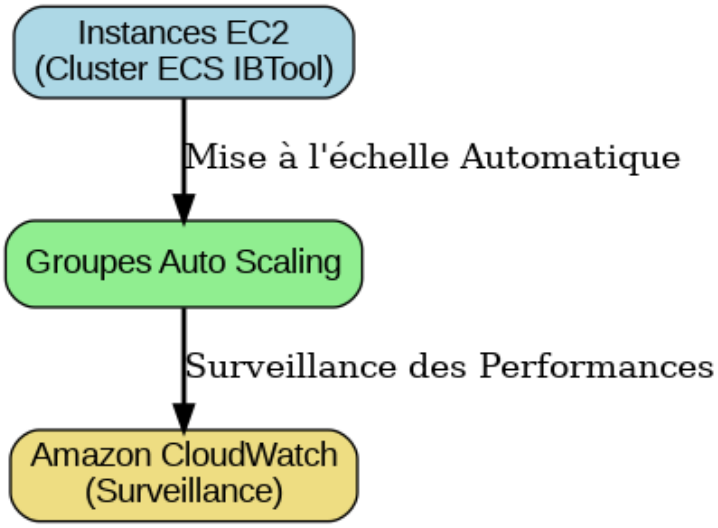

Figure 18 Mise à l'échelle automatique et Surveillance des instances EC2 pour IBTool

3.2.4 GESTION DU TRAFIC ET ÉQUILIBRAGE DE CHARGE

3.2.4.1 CONFIGURATION DE L'APPLICATION LOAD BALANCER (ALB)

Pour assurer la haute disponibilité et la fiabilité d'IBTool, un Application Load Balancer (ALB) a été configuré pour distribuer le trafic entrant vers les tâches ECS appropriées. La première étape a consisté à mettre en place l'ALB, en configurant des écouteurs pour le trafic HTTP et HTTPS. Les écouteurs ont été configurés pour intercepter les requêtes entrantes et les diriger vers les services adéquats.

Ensuite, des groupes cibles ont été créés pour les services frontend et backend, chacun étant associé aux tâches ECS correspondantes. Les règles de routage ont été définies pour diriger le trafic en fonction du chemin d'URL. Par exemple, les requêtes vers l'URL racine sont dirigées vers le service frontend, tandis que les requêtes avec le préfixe /api sont dirigées vers le service backend.

Enfin, des contrôles de santé ont été configurés pour surveiller l'état des tâches ECS. Cela garantit que le trafic est uniquement dirigé vers des instances en bonne santé, maintenant ainsi la fiabilité de l'application et optimisant la performance.

Le diagramme illustre la configuration du Load Balancer pour IBTool et le processus de routage du trafic. L'ALB est configuré pour distribuer le trafic entre les services frontend et backend en fonction des règles de routage définies par les chemins URL. Les contrôles de santé permettent de s'assurer que seul le trafic vers des tâches ECS en bon état est autorisé, ce qui améliore la fiabilité et les performances de l'application.

3.2.4.2 ROUTAGE DU TRAFIC VERS LES SERVICES ECS

L'ALB utilise des groupes cibles pour acheminer le trafic vers les services ECS appropriés, en s'appuyant sur les contrôles de santé et les règles de routage configurées. Des règles d'écoute ont été créées dans

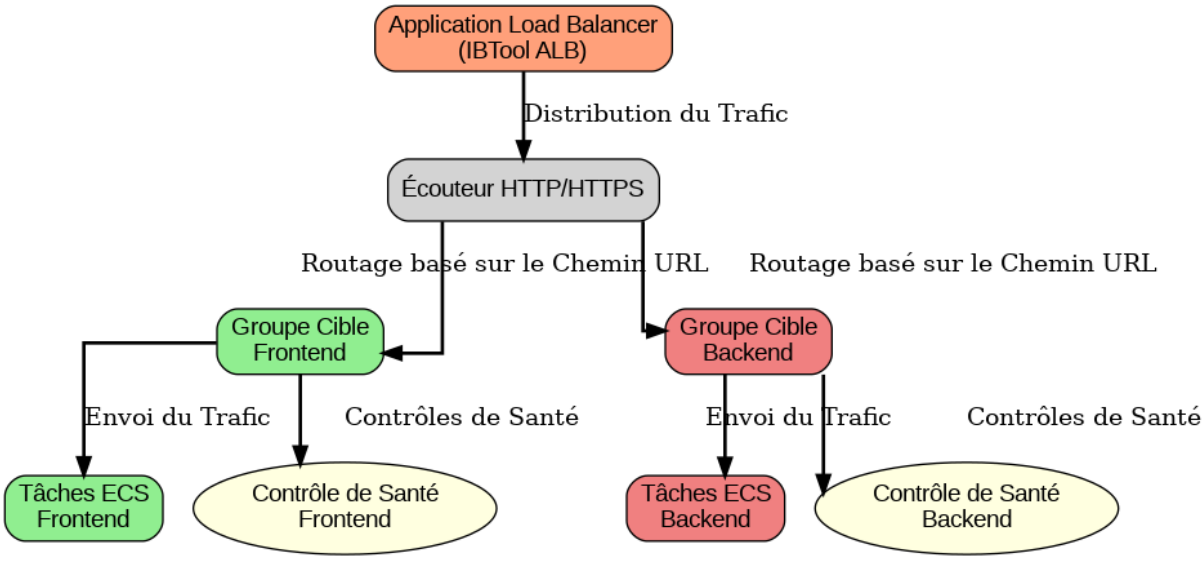

Figure 19 Configuration et Routage du Trafic avec ALB pour IBTool

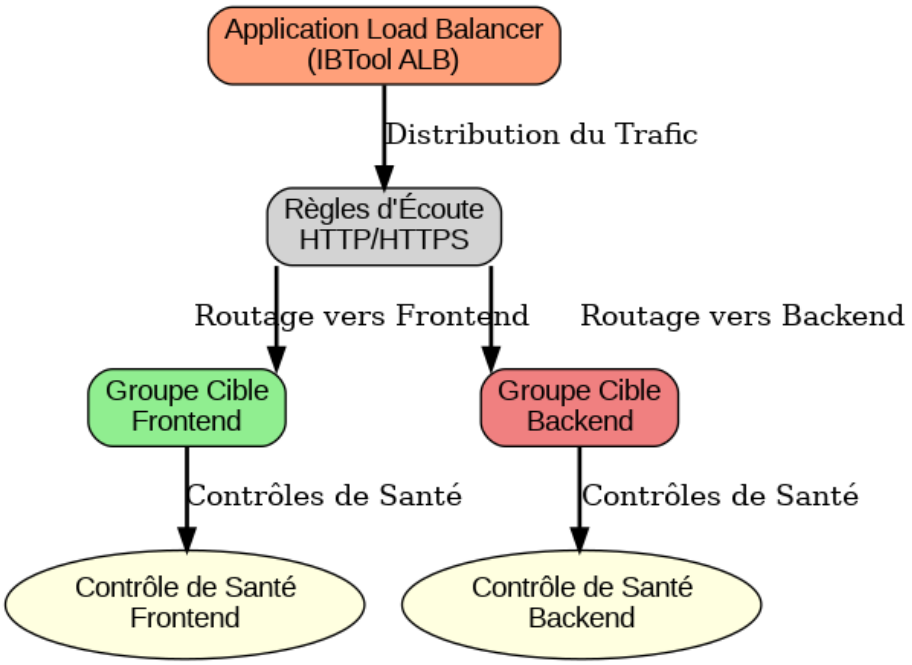

Figure 20 Routage du Trafic et Contrôles de Santé pour IBTool

l'ALB pour diriger le trafic en fonction des chemins URL spécifiés. Par exemple, les requêtes vers /api sont redirigées vers le backend, tandis que celles vers la racine sont dirigées vers le frontend.

Les contrôles de santé sont essentiels pour vérifier que les instances ECS fonctionnent correctement. Si une instance ne passe pas le contrôle de santé, l'ALB la retire du groupe cible et redirige le trafic vers des instances saines. Cette surveillance continue garantit une gestion fiable du trafic et améliore l'expérience utilisateur en maintenant l'application disponible et performante.

Ce diagramme montre comment l'ALB dirige le trafic vers les services ECS appropriés en fonction des règles d'écoute configurées et des contrôles de santé en temps réel. En ajustant les règles de routage et en surveillant en continu l'état des tâches, IBTool assure une gestion fiable du trafic, garantissant une expérience utilisateur améliorée avec des performances et une disponibilité optimales.

L'intégration d'un Application Load Balancer (ALB) dans l'infrastructure d'IBTool permet de gérer efficacement le trafic entrant et d'assurer une répartition optimale de la charge entre les services frontend et backend. Les configurations détaillées, y compris les règles de routage basées sur les chemins URL et les contrôles de santé, contribuent à maintenir une haute disponibilité et des performances optimales pour l'application.

3.2.5 MISE EN ŒUVRE DE LA SECURITE

3.2.5.1 GESTION DES ROLES ET POLITIQUES IAM

Pour assurer un contrôle d'accès rigoureux et une gestion des permissions sur les ressources d'IBTool, AWS Identity and Access Management (IAM) a été mis en place. J'ai défini des rôles IAM spécifiques pour les tâches ECS, les instances EC2, ainsi que pour d'autres services AWS utilisés par IBTool. Chaque rôle a été configuré avec les permissions nécessaires, en fonction des besoins spécifiques de chaque service.

Ces rôles IAM ont ensuite été associés à des politiques qui spécifient les autorisations requises pour accéder aux services essentiels tels que S3, ECR, et CloudWatch. De plus, IAM a été utilisé pour gérer l'accès des utilisateurs, garantissant que seules les personnes autorisées peuvent effectuer des actions spécifiques sur les ressources d'IBTool.

Le diagramme illustre comment les rôles IAM et les politiques sont configurés pour gérer l'accès aux ressources d'IBTool. Chaque rôle est défini avec des permissions spécifiques et est associé aux politiques IAM pour garantir un contrôle d'accès granulaire et une sécurité renforcée.

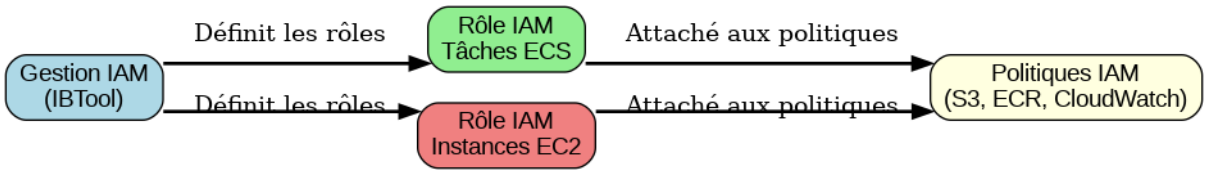

Figure 21 Rôles et Politiques IAM pour IBTool

3.2.5.2 CHIFFREMENT DES DONNEES AVEC KMS

Pour protéger les données sensibles d'IBTool, AWS Key Management Service (KMS) a été utilisé pour chiffrer les données stockées dans les bases de données et sur S3. J'ai généré des clés de chiffrement à l'aide de KMS, qui ont ensuite été utilisées pour chiffrer les données critiques, y compris celles stockées dans MongoDB et sur S3. L'accès aux clés de chiffrement est strictement contrôlé à l'aide de politiques IAM, garantissant que seules les entités autorisées peuvent décrypter les données. Cela

renforce la sécurité des données en protégeant les informations sensibles contre tout accès non autorisé.

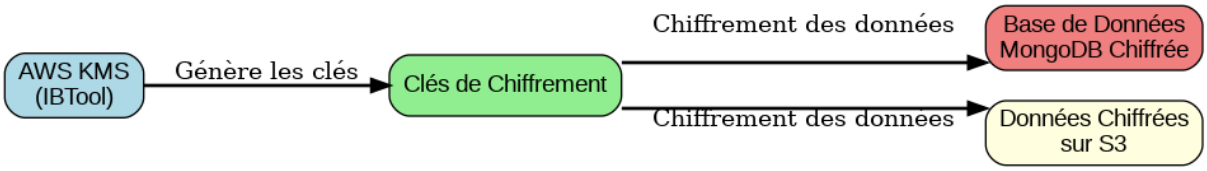

Figure 22 Flux de Chiffrement KMS pour IBTool

Ce diagramme montre comment AWS KMS est utilisé pour chiffrer les données sensibles d'IBTool. Les clés générées par KMS assurent que les données dans MongoDB et S3 sont protégées contre tout accès non autorisé, renforçant ainsi la sécurité globale de l'application.

3.2.5.3 GESTION DES CERTIFICATS SSL/TLS AVEC ACM

Pour garantir des communications sécurisées entre IBTool et ses utilisateurs, AWS Certificate Manager (ACM) a été utilisé pour provisionner et gérer les certificats SSL/TLS. Les certificats nécessaires pour les noms de domaine utilisés par IBTool ont été demandés et attachés à l'Application Load Balancer (ALB) afin de permettre le trafic HTTPS sécurisé.

De plus, ACM automatise le renouvellement des certificats, garantissant ainsi que les communications restent sécurisées sans interruption. Cette approche simplifie la gestion des certificats tout en assurant une protection continue des données en transit.

Ce diagramme illustre comment AWS ACM gère les certificats SSL/TLS pour IBTool. En automatisant le provisionnement et le renouvellement des certificats, ACM assure des communications sécurisées et simplifie la gestion des certificats SSL/TLS nécessaires à la protection des données en transit.

La mise en œuvre de la sécurité pour IBTool à l'aide des services AWS tels qu'IAM, KMS, et ACM garantit une protection robuste des ressources et des données. Chaque composant de cette infrastructure de sécurité joue un rôle essentiel pour assurer que seules les entités autorisées peuvent accéder aux ressources critiques, que les données sont protégées par un chiffrement fort, et que les communications sont sécurisées à l'aide de certificats SSL/TLS.

3.2.6 SURVEILLANCE ET JOURNALISATION

3.2.6.1 CONFIGURATION DES ALARMES ET TABLEAUX DE BORD CLOUDWATCH

Pour assurer une surveillance proactive de la performance et de la santé d'IBTool, Amazon CloudWatch a été mis en place. J'ai configuré des métriques spécifiques collectées à partir des tâches ECS, des instances EC2, et d'autres services AWS essentiels. Ces métriques fournissent des informations précieuses sur l'utilisation du CPU, la mémoire, et la latence des requêtes.

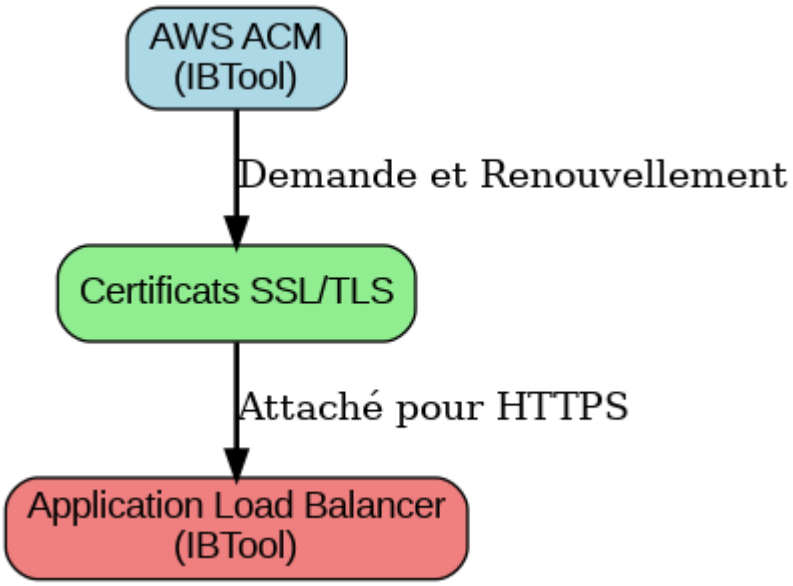

Figure 23 Gestion des Certificats ACM pour IBTool

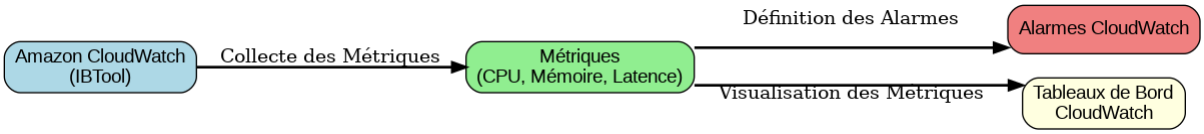

Figure 24 Tableaux de Bord et Alarmes CloudWatch pour IBTool

Des alarmes CloudWatch ont ensuite été définies pour surveiller ces métriques critiques. Ces alarmes sont configurées pour alerter l'équipe de développement dès que des seuils prédéfinis sont atteints, permettant une intervention rapide avant que les utilisateurs ne soient impactés. Pour visualiser ces informations en temps réel, des tableaux de bord CloudWatch ont été créés. Ces tableaux de bord offrent une vue d'ensemble des performances d'IBTool, facilitant ainsi le suivi et l'analyse des données.

Ce diagramme montre comment CloudWatch est utilisé pour surveiller IBTool. Les métriques collectées sont utilisées pour définir des alarmes et créer des tableaux de bord, assurant ainsi une surveillance proactive et en temps réel de la performance de l'application.

3.2.6.2 UTILISATION DE CLOUDWATCH LOGS POUR LES INSIGHTS APPLICATIFS

CloudWatch Logs a été configuré pour collecter et stocker les journaux des tâches ECS d'IBTool, fournissant des insights précieux sur le comportement et la performance de l'application. Les tâches ECS ont été configurées pour envoyer leurs journaux applicatifs à CloudWatch Logs, où ils sont organisés en groupes et flux de journaux.

La gestion efficace de ces journaux permet de les rechercher et de les analyser facilement. En utilisant CloudWatch Logs Insights, il est possible de réaliser des requêtes sur les données de journaux pour identifier des tendances, des anomalies, ou des problèmes spécifiques. Cette capacité d'analyse approfondie est essentielle pour le dépannage et l'optimisation des performances d'IBTool.

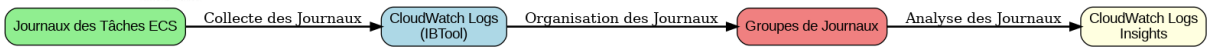

Figure 25 CloudWatch Logs et Insights pour IBTool

Ce diagramme illustre comment CloudWatch Logs est utilisé pour collecter et analyser les journaux d'IBTool. Les journaux sont collectés, organisés, et analysés, offrant des insights détaillés sur les performances et le comportement de l'application, essentiels pour le dépannage et l'amélioration continue. La mise en place de CloudWatch pour la surveillance et la journalisation dans IBTool a permis d'assurer une gestion proactive et efficace de l'application. Grâce aux tableaux de bord et aux alarmes, l'équipe peut surveiller en temps réel les performances et recevoir des alertes en cas de problème. De plus, l'analyse des journaux avec CloudWatch Logs Insights fournit des informations précieuses pour l'amélioration continue d'IBTool, garantissant ainsi une performance optimale et une expérience utilisateur améliorée.

4. RESULTATS ET ANALYSE

4.1 ÉTUDE DE CAS: DEPLOIEMENT ET EXPLOITATION D'IBTOOL

4.1.1 VUE D'ENSEMBLE D'IBTOOL

IBTool est une application web interne développée pour examiner, suivre et gérer les plaintes des clients de GE Healthcare concernant ses produits. Cet outil joue un rôle crucial dans le maintien de la satisfaction des clients et l'assurance qualité des produits. L'application se compose d'une interface frontend pour permettre aux utilisateurs de soumettre et de consulter les plaintes, d'un service backend pour traiter et gérer les données, ainsi que d'une base de données MongoDB où toutes les informations associées sont stockées.

Avant sa migration vers le cloud, IBTool était hébergé sur une infrastructure on-premises, ce qui posait plusieurs défis, notamment une évolutivité limitée, des coûts de maintenance élevés, et des difficultés à garantir une haute disponibilité. Pour surmonter ces limitations, GE Healthcare a décidé de migrer IBTool vers AWS, exploitant ainsi les capacités cloud-native pour améliorer les performances, la scalabilité, et l'efficacité des coûts.

4.1.2 EXIGENCES ET OBJECTIFS DU PROJET

Le projet de migration d'IBTool vers le cloud avait pour principaux objectifs :

- **Scalabilité** : Assurer qu'IBTool puisse gérer une augmentation des charges de travail en ajustant automatiquement les ressources disponibles.
- **Haute Disponibilité** : Réduire au minimum les temps d'arrêt et garantir une disponibilité continue de l'application.
- **Efficacité Opérationnelle** : Diminuer les efforts de maintenance et rationaliser le processus de déploiement.

- **Optimisation des Coûts** : Exploiter le modèle de tarification à la demande d'AWS pour réduire les coûts d'exploitation.
- **Sécurité Renforcée** : Mettre en œuvre des mesures de sécurité robustes pour protéger les données sensibles.

4.2 PROCESSUS DE DEPLOIEMENT

Le déploiement d'IBTool sur AWS s'est déroulé en plusieurs étapes clés, chacune visant à assurer une architecture solide, évolutive et sécurisée.

Étape 1 : Création du Dépôt CodeCommit

Tout d'abord, un dépôt CodeCommit nommé IBTool-Repo a été créé pour centraliser le code source de l'application. Le code existant pour le frontend, le backend, et la configuration de la base de données a été intégré dans ce dépôt. Les rôles et politiques IAM ont ensuite été configurés pour gérer les accès de manière sécurisée.

Étape 2 : Construction et Test avec CodeBuild

Un projet CodeBuild a été configuré pour le dépôt IBTool-Repo. Un fichier buildspec.yml a été utilisé pour définir les commandes de construction, incluant la création des images Docker et les tests unitaires. Les builds ont été automatiquement déclenchés lors des commits, assurant que les dernières modifications étaient toujours testées et prêtes pour le déploiement.

Étape 3 : Configuration d'ECR pour les Images Docker

Des dépôts ECR distincts ont été créés pour les images Docker des services frontend, backend, et MongoDB. Les images construites par CodeBuild ont été poussées vers les dépôts ECR respectifs, centralisant ainsi le stockage des images pour un accès facile et un déploiement cohérent.

Étape 4 : Configuration d'ECS et des Instances EC2

Un cluster ECS nommé IBTool-Cluster a été mis en place pour héberger les conteneurs Docker. Les définitions de tâches ont été créées pour chaque service (frontend, backend, MongoDB), spécifiant les images Docker et les ressources requises. Les services ECS ont été configurés pour gérer et mettre à l'échelle chaque tâche de manière indépendante.

Étape 5 : Mise en Place de l'Équilibrage de Charge avec ALB

Un Application Load Balancer (ALB) a été configuré pour distribuer le trafic entrant vers les tâches ECS appropriées. Les groupes cibles ont été définis pour les services frontend et backend, et des règles de routage ont été mises en place pour assurer une direction correcte du trafic en fonction des chemins URL.

Étape 6 : Mise en Place des Mesures de Sécurité

Les rôles et politiques IAM ont été établis pour contrôler l'accès aux ressources AWS. AWS KMS a été utilisé pour chiffrer les données sensibles, et AWS Certificate Manager (ACM) a été employé pour gérer les certificats SSL/TLS, garantissant une communication sécurisée.

Étape 7 : Surveillance et Journalisation avec CloudWatch

Des métriques CloudWatch ont été configurées pour surveiller les tâches ECS, les instances EC2, et d'autres ressources AWS. Des alarmes et des tableaux de bord ont été créés pour surveiller les performances et alerter en cas d'anomalies.

4.3 INSIGHTS OPERATIONNELS

4.3.1 SURVEILLANCE DE LA PERFORMANCE APPLICATIVE

Après le déploiement, il est crucial de surveiller les performances d'IBTool pour s'assurer qu'elle réponde aux attentes en matière de service. Amazon CloudWatch offre des capacités complètes pour surveiller les métriques clés de performance et la santé du système.

Les métriques surveillées incluent l'utilisation du CPU et de la mémoire pour les tâches ECS et les instances EC2, la latence des requêtes pour identifier les goulets d'étranglement, et les taux d'erreurs pour détecter et résoudre rapidement les problèmes. Par exemple, lors d'une augmentation de la charge, CloudWatch a signalé une hausse de l'utilisation du CPU, ce qui a déclenché l'ajout automatique d'instances EC2 supplémentaires et le lancement de tâches backend supplémentaires pour maintenir la réactivité de l'application.

4.3.2 EXPERIENCES DE MISE A L'ÉCHELLE ET MAINTENANCE

Les capacités d'auto-scaling d'ECS et d'EC2 jouent un rôle clé dans l'efficacité opérationnelle d'IBTool, permettant à l'application de s'adapter aux charges de travail variables sans nécessiter d'intervention manuelle. Les événements de scalabilité programmée ont été utilisés pour anticiper les pics de trafic pendant les heures de pointe, tandis que la scalabilité dynamique, basée sur les métriques CloudWatch, a permis à IBTool de répondre rapidement aux surcharges imprévues en ajustant les ressources en temps réel. Pour la maintenance, des mises à jour automatiques via CodePipeline ont permis de réduire les temps d'arrêt et d'éviter les erreurs manuelles. De plus, des sauvegardes régulières de la base de données MongoDB ont été stockées sur S3 pour garantir la disponibilité et l'intégrité des données. Par exemple, lors d'une fenêtre de maintenance programmée, des mises à jour de code ont été déployées via CodePipeline sans interruption notable pour les utilisateurs, démontrant l'efficacité du déploiement automatisé et de la surveillance continue.

5. DISCUSSION ET RETOURS D'EXPERIENCE

5.1 LEÇONS TIREES

La migration d'IBTool vers une architecture cloud-native a présenté plusieurs défis, chacun nécessitant des solutions spécifiques pour être surmonté. Ce processus a permis de dégager des enseignements

clés, tant sur les défis rencontrés que sur les meilleures pratiques à adopter pour des déploiements cloud réussis.

5.1.1 DEFIS ET SOLUTIONS

5.1.1.1 GESTION COMPLEXE DE LA CONFIGURATION

La gestion des configurations de multiples services AWS s'est révélée complexe au début du projet. Chaque service nécessite une configuration spécifique, et la coordination de ces configurations pour assurer un fonctionnement harmonieux de l'ensemble du système représentait un véritable défi.

Solution : Pour surmonter ce problème, j'ai mis en place une infrastructure as code (IaC) en utilisant Terraform. Cette approche a permis d'automatiser la gestion des configurations, assurant ainsi une cohérence et une répétabilité des déploiements à travers différents environnements. Terraform a simplifié la gestion des configurations en centralisant et en standardisant les processus, réduisant ainsi le risque d'erreurs humaines.

5.1.1.2 PREOCCUPATIONS EN MATIERE DE SECURITE

La sécurité des données lors de la migration était une priorité absolue. Assurer que les données sensibles restent protégées pendant et après la migration vers le cloud était crucial, notamment en ce qui concerne le chiffrement des données et la gestion des accès.

Solution : J'ai a exploité les services de sécurité d'AWS tels que IAM (Identity and Access Management), KMS (Key Management Service), et ACM (AWS Certificate Manager). Ces services ont été utilisés pour appliquer des pratiques de sécurité strictes, y compris la gestion des identités et des accès, le chiffrement des données sensibles, et la gestion des certificats SSL/TLS pour garantir des communications sécurisées.

Voici un tableau récapitulatif des défis rencontrés et des solutions mises en place :

Tableau 1 Les defis

Défi	Solution
Gestion Complexe de la Configuration	Mise en place d'une infrastructure as code avec Terraform
Préoccupations en Matière de Sécurité	Utilisation des services de sécurité AWS (IAM, KMS, ACM)

5.1.2 MEILLEURES PRATIQUES POUR LES DEPLOIEMENTS CLOUD-NATIVE

À partir de l'expérience acquise lors de la migration d'IBTool, plusieurs meilleures pratiques ont été identifiées pour garantir le succès des déploiements cloud-native.

Automatisation avec Infrastructure as Code (IaC) : L'utilisation d'IaC, notamment avec Terraform, permet d'automatiser le provisionnement et la gestion des ressources cloud, garantissant ainsi la cohérence et la reproductibilité des environnements de développement, de test, et de production.

Cette approche réduit les risques liés à la configuration manuelle et facilite le déploiement rapide de nouvelles fonctionnalités.

Surveillance Robuste : La mise en place d'une surveillance continue des performances et de la santé du système via Amazon CloudWatch est essentielle. La surveillance proactive permet de détecter et de résoudre les problèmes avant qu'ils n'affectent les utilisateurs finaux, assurant ainsi une disponibilité et une performance optimales.

Automatisation des Pipelines de Déploiement : L'intégration continue (CI) et le déploiement continu (CD) avec des outils comme AWS CodePipeline permettent de rationaliser le processus de déploiement, réduisant ainsi les erreurs humaines et accélérant la mise en production des mises à jour. Cette automatisation assure que les modifications du code sont systématiquement testées et déployées de manière sécurisée.

Priorisation de la Sécurité : La sécurité doit être une priorité à chaque étape du processus de déploiement. L'utilisation des services de sécurité AWS, comme IAM pour la gestion des accès, KMS pour le chiffrement des données, et ACM pour la gestion des certificats, permet de renforcer la sécurité globale de l'application.

Le diagramme ci-dessous illustre les meilleures pratiques pour les déploiements cloud-native, en soulignant leur interconnexion et leur importance dans la réussite globale du projet IBTool.

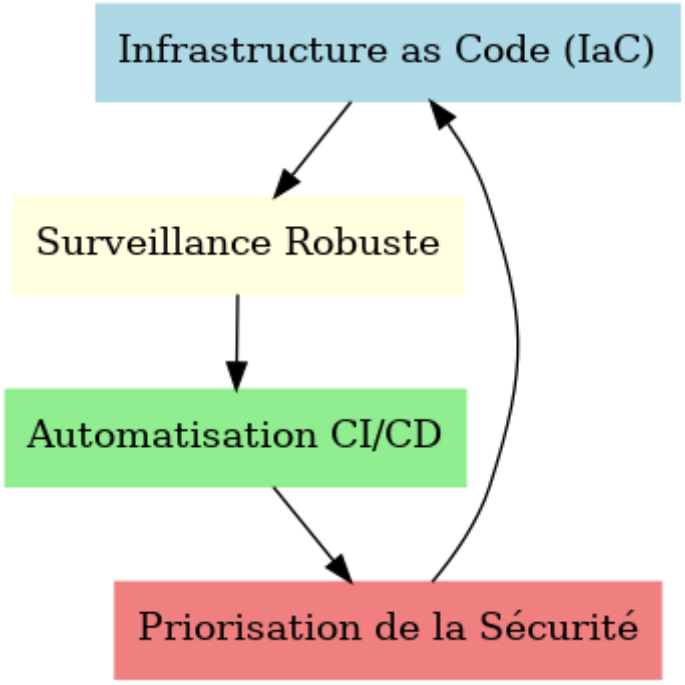

Figure 26 Meilleures Pratiques pour les Déploiements Cloud-Native

La migration d'IBTool vers une architecture cloud-native sur AWS a conduit à des améliorations significatives en termes de scalabilité, de performance, de sécurité, et d'efficacité opérationnelle. L'intégration des différents services AWS a permis de créer une infrastructure robuste, résiliente et

rentable, capable de mieux répondre aux besoins des utilisateurs et de s'adapter aux futures exigences.

Les travaux futurs se concentreront sur l'exploration de services AWS supplémentaires, tels que l'intégration de l'apprentissage automatique pour des analyses prédictives et l'utilisation d'architectures serverless pour réduire la charge opérationnelle. Ces améliorations continueront à renforcer la capacité d'IBTool à offrir un service de haute qualité tout en minimisant les coûts et les efforts de maintenance.

L'étude de cas a démontré l'application pratique et les avantages d'une architecture cloud-native pour des applications d'entreprise, fournissant ainsi des insights précieux et des meilleures pratiques pour des projets de migration similaires.

5.2 SCALABILITE ET PERFORMANCE

5.2.1 ÉVOLUTIVITE ET PERFORMANCE

L'une des principales raisons du passage d'IBTool à une architecture cloud-native réside dans la capacité à évoluer automatiquement en fonction de la demande. L'intégration d'AWS Auto Scaling avec Amazon ECS et EC2 garantit qu'IBTool peut gérer des charges de travail variables sans intervention manuelle. Grâce à cette configuration, IBTool peut ajuster dynamiquement le nombre de tâches ECS et d'instances EC2 en fonction de la demande en temps réel, optimisant ainsi l'utilisation des ressources et maintenant des performances optimales, même en période de forte affluence.

Tableau 2 Métriques d'Auto Scaling et Déclencheurs pour IBTool

Métrique	Seuil de Déclenchement	Action d'Auto Scaling
Utilisation du CPU (ECS)	> 75%	Augmentation du nombre de tâches ECS
Utilisation de la mémoire	> 70%	Ajout d'instances EC2 supplémentaires
Latence des requêtes (Backend)	> 200ms	Augmentation des tâches ECS pour le service backend
Période d'inactivité	< 20%	Réduction du nombre d'instances EC2

Ce tableau illustre les métriques surveillées par Auto Scaling pour IBTool et les actions déclenchées en réponse à des changements de charge, assurant ainsi une utilisation optimale des ressources tout en maintenant des coûts sous contrôle.

5.2.2 OPTIMISATION DES PERFORMANCES AVEC ALB ET ECS

L'association de l'Application Load Balancer (ALB) et d'Amazon ECS améliore les performances d'IBTool en distribuant efficacement le trafic et en gérant les services conteneurisés. ALB distribue les requêtes entrantes vers les tâches ECS saines, évitant ainsi qu'une seule tâche ne devienne un goulet d'étranglement. De plus, ALB effectue des

contrôles de santé en continu et ne dirige le trafic que vers les instances en bon état, réduisant ainsi la latence et améliorant l'expérience utilisateur.

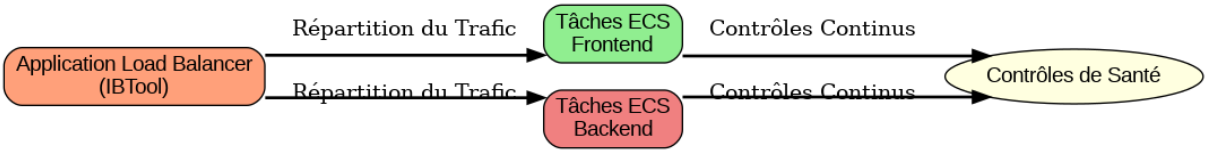

Figure 27 Répartition du Trafic et Contrôles de Santé avec ALB pour IBTool

Le diagramme ci-dessus montre comment l'ALB distribue les requêtes aux tâches ECS tout en surveillant leur état grâce aux contrôles de santé. Cette approche garantit une performance constante de l'application en minimisant la latence et en maintenant un service fluide.

5.3 AMELIORATIONS DE LA SECURITE

AWS offre des services de sécurité robustes qui renforcent considérablement la posture de sécurité d'IBTool. En utilisant IAM, KMS et ACM, IBTool assure un accès sécurisé, un chiffrement des données et des communications cryptées. Les rôles et politiques IAM contrôlent précisément l'accès aux ressources, tandis que KMS garantit que les données sensibles, telles que celles stockées dans MongoDB et S3, sont chiffrées. De plus, ACM gère les certificats SSL/TLS, permettant des communications HTTPS sécurisées entre les utilisateurs et l'application.

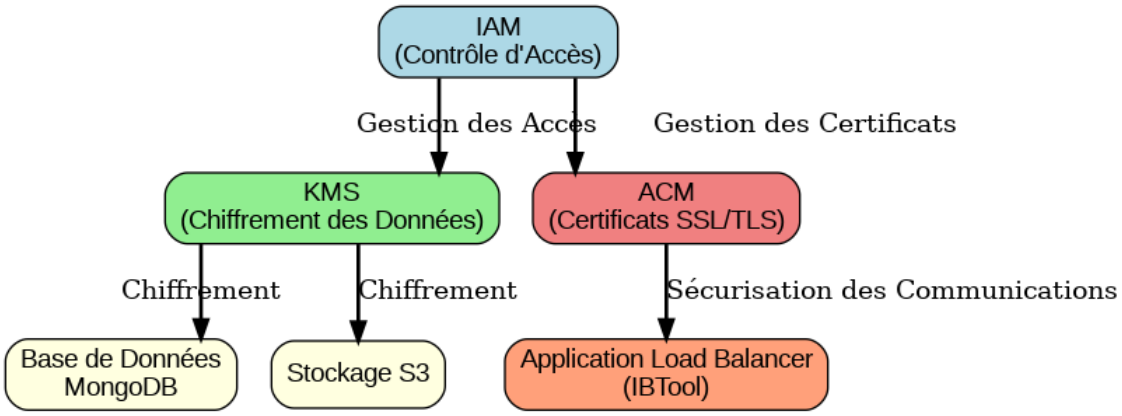

Figure 28 Intégration des Composants de Sécurité pour IBTool

Ce diagramme illustre l'intégration des services de sécurité d'AWS avec IBTool. L'utilisation combinée d'IAM, KMS, et ACM garantit un contrôle d'accès granulaire, un chiffrement fort des données et des communications sécurisées, renforçant ainsi la protection des informations sensibles.

5.4 EFFICACITE OPERATIONNELLE

5.4.1 GESTION SIMPLIFIEE AVEC ECS ET EC2

Amazon ECS et EC2 jouent un rôle clé dans la simplification de la gestion des applications conteneurisées pour IBTool, réduisant ainsi la charge opérationnelle. ECS automatise le déploiement et la mise à l'échelle des conteneurs Docker, garantissant que l'application fonctionne de manière fluide sans nécessiter d'intervention manuelle. Parallèlement, les instances EC2 sont gérées via des groupes d'Auto Scaling, qui ajustent automatiquement le nombre d'instances en fonction de la charge de travail, simplifiant ainsi la gestion des ressources.

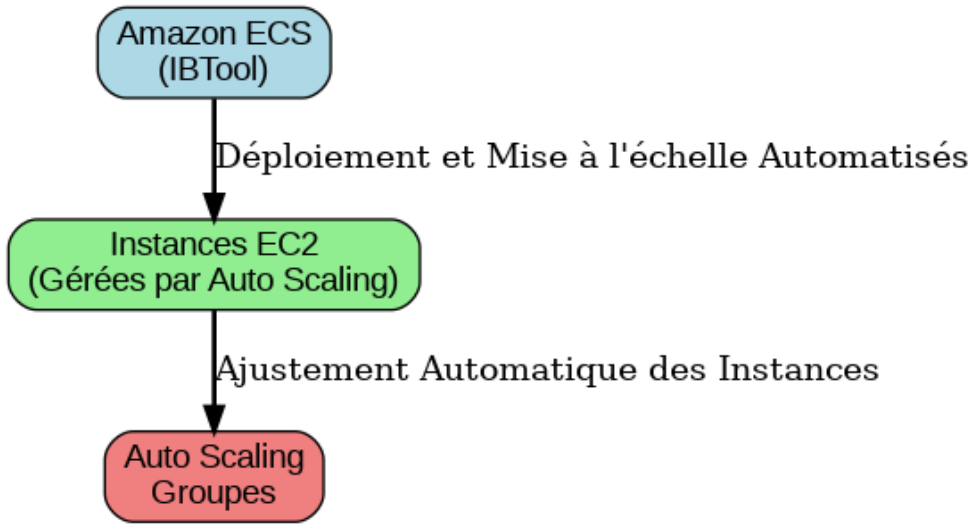

Figure 29 Gestion des Ressources ECS et EC2 pour IBTool

Ce diagramme montre comment Amazon ECS et EC2, avec l'aide des groupes d'Auto Scaling, automatisent la gestion des ressources pour IBTool. Cette automatisation garantit que l'application peut s'adapter aux variations de charge sans intervention manuelle, tout en optimisant les ressources disponibles.

5.5 EFFICACITE DES COUTS

5.5.1 OPTIMISATION DES COUTS AVEC AUTO SCALING ET STOCKAGE S3

Les services AWS offrent des mécanismes d'économie qui aident IBTool à optimiser ses dépenses opérationnelles. Grâce à l'Auto Scaling, le nombre d'instances en cours d'exécution est ajusté automatiquement en fonction de la demande, garantissant qu'IBTool ne paie que pour les ressources réellement utilisées. De plus, Amazon S3 fournit un stockage évolutif et rentable pour les actifs statiques et les sauvegardes, avec une tarification basée sur l'utilisation réelle.

Tableau 3 Comparaison des Coûts Avant et Après la Migration

Catégorie	Avant la Migration	Après la Migration	Économie
Coûts d'infrastructure	1000 €/mois	700 €/mois	30%
Stockage	300 €/mois	200 €/mois	33%

Maintenance manuelle	200 €/mois	50 €/mois	75%
Total	1500 €/mois	950 €/mois	37%

Ce tableau montre l'économie de coûts réalisée par IBTool grâce à la migration vers une architecture cloud-native, principalement en raison de l'optimisation des ressources et du stockage.

5.5.2 GESTION ET SUIVI DES COUTS

AWS fournit des outils permettant à IBTool de surveiller et de gérer ses coûts de manière efficace. AWS Cost Explorer offre des rapports détaillés sur l'utilisation des ressources et les coûts associés, aidant IBTool à identifier les domaines où il est possible de réduire les dépenses. De plus, des alertes budgétaires peuvent être configurées pour notifier l'équipe lorsque les coûts dépassent les seuils prédéfinis, assurant ainsi un meilleur contrôle des dépenses.

Ce diagramme montre comment AWS Cost Explorer et les alertes budgétaires sont utilisés pour surveiller et contrôler les coûts liés à l'infrastructure d'IBTool, garantissant une gestion financière efficace.

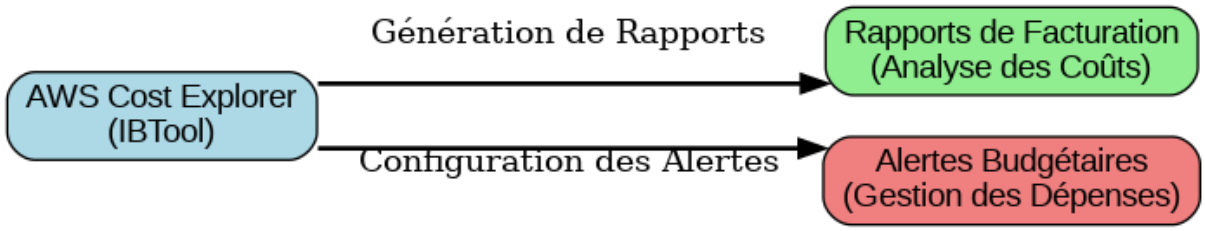

Figure 30 Gestion des Coûts AWS pour IBTool

L'intégration des services AWS dans l'infrastructure d'IBTool a permis non seulement d'améliorer l'efficacité opérationnelle, mais aussi de réaliser des économies de coûts significatives. Grâce à une gestion simplifiée des ressources et à un contrôle précis des dépenses, IBTool est en mesure de maintenir une performance optimale tout en réduisant ses coûts opérationnels.

6. CONCLUSION ET PERSPECTIVES

6.1 RESUME DES RESULTATS

6.1.1 REALISATIONS ET CONTRIBUTIONS

Le projet IBTool a représenté une avancée majeure en deux étapes cruciales : son développement initial en tant qu'outil on-premises et sa migration ultérieure vers une architecture cloud-native sur AWS. Chacune de ces étapes a résolu des problèmes spécifiques et apporté des bénéfices significatifs.

Dans sa phase on-premises, IBTool a permis de centraliser la gestion des plaintes au sein de l'équipe Install Base de GE Healthcare, en remédiant à la fragmentation des informations, en améliorant la

visibilité sur les dossiers sensibles, et en automatisant des tâches qui étaient auparavant manuelles et chronophages. L'interface utilisateur a été rendue plus intuitive, facilitant ainsi l'adoption et l'efficacité de l'outil par l'équipe.

La migration vers le cloud a ensuite permis de surmonter les limitations de l'infrastructure on-premises, en apportant des améliorations en termes de scalabilité, de disponibilité, de sécurité, et de réduction des coûts. En intégrant des services AWS tels qu'Amazon ECS, EC2, et l'Application Load Balancer, IBTool a gagné en robustesse et en résilience. Cette transformation a permis à l'outil de s'adapter dynamiquement aux besoins changeants, tout en garantissant une performance optimale et une sécurité renforcée.

6.1.2 POINTS CLES

Les leçons tirées de ce projet sont nombreuses et offrent des pistes pour de futurs projets similaires. L'automatisation des processus, notamment pour le déploiement et la gestion des ressources, s'est révélée essentielle pour maintenir l'efficacité et la fiabilité du système. La sécurité, dès le début du projet, a été priorisée pour garantir la protection des données sensibles. Enfin, la surveillance continue des performances et l'optimisation des ressources ont permis de maintenir des coûts sous contrôle tout en améliorant la réactivité de l'application.

6.2 CONTRIBUTIONS

Le développement et la migration d'IBTool ont permis d'apporter des contributions significatives, non seulement pour GE Healthcare, mais aussi pour la compréhension des avantages des architectures cloud-native dans les environnements d'entreprise. Ce projet démontre comment une approche méthodique, combinant développement sur site et migration vers le cloud, peut transformer un outil critique en une solution encore plus performante et flexible.

En partageant les défis rencontrés et les solutions mises en œuvre, le projet IBTool offre un cadre de référence pour d'autres organisations qui envisagent des projets similaires. Il illustre comment les services cloud peuvent être utilisés pour améliorer la gestion des plaintes, tout en maintenant un haut niveau de sécurité et d'efficacité opérationnelle.

6.3 AMELIORATIONS FUTURES

6.3.1 AMELIORATIONS POTENTIELLES

Bien que le projet ait été un succès, il reste encore des axes d'amélioration pour IBTool. L'intégration d'outils d'analyse avancés comme Amazon QuickSight pourrait permettre de mieux exploiter les données collectées pour des rapports plus détaillés. De plus, l'adoption de l'apprentissage automatique avec AWS SageMaker pourrait offrir des capacités prédictives pour anticiper les tendances et les plaintes futures. Enfin, explorer des architectures serverless avec AWS Lambda pourrait réduire encore davantage la charge opérationnelle.

6.3.2 TENDANCES FUTURES EN INFORMATIQUE EN NUAGE

À mesure que le cloud computing évolue, IBTool pourrait bénéficier de nouvelles tendances comme l'edge computing, qui réduirait la latence en traitant les données plus près de leur source. L'intégration d'intelligences artificielles plus avancées et de stratégies hybrides ou multi-cloud pourrait aussi renforcer l'efficacité et la résilience de l'outil.

6.4 CONCLUSION

Le projet IBTool, du développement initial en on-premises à la migration réussie vers le cloud, témoigne de la capacité à transformer un outil stratégique en une solution encore plus performante et adaptable. En tirant parti des services AWS, IBTool a non seulement atteint ses objectifs initiaux, mais a aussi jeté les bases pour une amélioration continue.

Ce projet offre une feuille de route claire pour d'autres organisations, illustrant comment une approche combinée de développement et de migration vers le cloud peut offrir des résultats tangibles et durables. En continuant à exploiter les dernières tendances technologiques, IBTool est bien positionné pour maintenir sa pertinence et son efficacité, garantissant ainsi un service de qualité pour GE Healthcare et ses clients.

BIBLIOGRAPHIE

[1] https://en.wikipedia.org/wiki/Software_maintenance - Décrit les aspects essentiels de la maintenance logicielle.
[2] https://www.ibm.com/products/rational-clearquest - Outil de gestion des changements et des configurations logicielles par IBM.
[3] https://www.broadcom.com/products/software/value-stream-management/rally?bc_lang=en-us - Outil de gestion de projets et d'agilité, utilisé pour le suivi des user stories et des tâches.
[4] https://www.gehealthcare.fr/ - Page officielle de GE Healthcare France.
[5] https://js.devexpress.com/ - Bibliothèque JavaScript pour le développement d'applications web.
[6] https://webpack.js.org/ - Module bundler pour JavaScript, utilisé pour empaqueter les fichiers et ressources associés.
[7] https://www.docker.com/ - Plateforme pour développer, expédier et exécuter des applications dans des conteneurs.
[8] https://www.nginx.com/ - Serveur web open-source, également utilisé comme reverse proxy et load balancer.
[9] https://expressjs.com/ - Framework web minimaliste pour Node.js.
[10] https://www.mongodb.com/ - Base de données NoSQL orientée documents.
[11] https://mongoosejs.com/ - Bibliothèque de modélisation d'objets pour Node.js, facilitant la gestion des données dans MongoDB.
[12] https://docs.docker.com/compose/ - Outil pour définir et gérer des applications multi-conteneurs avec Docker.
[13] https://fr.wikipedia.org/wiki/GE_Healthcare - Page Wikipedia décrivant GE Healthcare.
[14] https://aws.amazon.com/ - Page officielle d'Amazon Web Services (AWS).
[15] https://www.terraform.io/ - Outil d'infrastructure as code (IaC) permettant de définir et de fournir des ressources cloud.
[16] https://aws.amazon.com/ecs/ - Service de gestion de conteneurs permettant d'exécuter et de gérer des conteneurs Docker sur AWS.
[17] https://aws.amazon.com/ec2/ - Service de serveur virtuel scalable sur le cloud.
[18] https://aws.amazon.com/s3/ - Service de stockage d'objets avec une haute durabilité et scalabilité.
[19] https://aws.amazon.com/cloudformation/ - Service permettant de modéliser et de configurer les ressources AWS.
[20] https://aws.amazon.com/iam/ - Service de gestion de l'accès sécurisé aux services et ressources AWS.
[21] https://azure.microsoft.com/en-us/ - Page officielle de Microsoft Azure, une plateforme cloud qui offre un large éventail de services pour la création, le déploiement et la gestion des applications.
[22] https://cloud.google.com/ - Page officielle de Google Cloud, une plateforme cloud qui propose des services pour le calcul, le stockage, l'IA et l'apprentissage automatique.
[23] https://cloud.ibm.com/ - Page officielle d'IBM Cloud, qui offre des solutions cloud pour l'infrastructure, les logiciels et les services de gestion des données.

GLOSSAIRE

Terme	Description
IBTool	Outil développé pour la gestion des plaintes au sein de GE Healthcare, utilisé pour améliorer l'analyse et le suivi des plaintes clients.
Cloud-Native	Architecture logicielle qui utilise pleinement les avantages des services cloud comme AWS pour la scalabilité, la résilience, et l'efficacité opérationnelle.
On-Premises	Réfère à l'infrastructure informatique hébergée localement sur les serveurs de l'entreprise, plutôt que sur le cloud.
AWS (Amazon Web Services)	Ensemble de services de cloud computing offert par Amazon, utilisés pour héberger et gérer des applications comme IBTool.
ECR (Elastic Container Registry)	Service AWS pour stocker, gérer et déployer des images de conteneurs Docker.
ECS (Elastic Container Service)	Service AWS permettant la gestion des conteneurs Docker sur un cluster, facilitant le déploiement d'applications comme IBTool.
EC2 (Elastic Compute Cloud)	Service AWS permettant de louer des serveurs virtuels pour exécuter des applications sur le cloud.
IAM (Identity and Access Management)	Service AWS pour gérer de manière sécurisée les identités et les accès aux ressources AWS.
KMS (Key Management Service)	Service AWS permettant de créer et gérer des clés de chiffrement pour sécuriser les données sensibles.
ACM (AWS Certificate Manager)	Service AWS pour gérer les certificats SSL/TLS, assurant des communications sécurisées entre le client et le serveur.
ALB (Application Load Balancer)	Service AWS permettant de distribuer le trafic entrant sur plusieurs instances EC2 pour assurer la haute disponibilité et la tolérance aux pannes.
CI/CD (Continuous Integration/Continuous Deployment)	Processus automatisé pour intégrer et déployer du code de manière continue, réduisant les erreurs et accélérant les cycles de publication.
Terraform	Outil d'infrastructure as code (IaC) permettant de définir et de déployer des ressources cloud de manière automatisée et cohérente.
Docker	Outil permettant de créer, déployer et exécuter des applications dans des conteneurs, facilitant leur portabilité et leur gestion.
Surveillance et Journalisation	Processus de suivi des performances et des événements d'une application pour assurer son bon fonctionnement et sa sécurité.
Auto Scaling	Fonctionnalité AWS qui ajuste automatiquement la capacité des services pour maintenir une performance stable et un coût optimisé.
S3 (Simple Storage Service)	Service de stockage d'objets sur AWS, utilisé pour stocker et récupérer de grandes quantités de données de manière scalable.
Interface Utilisateur (IU)	Partie visible de l'application avec laquelle les utilisateurs interagissent, telle que celle développée pour IBTool.
Scalabilité	Capacité d'une application ou d'un service à gérer une augmentation de la charge de travail sans affecter les performances.