ESI - Titre RNCP de Niveau 7

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

Livre Blanc

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

Aout 2024
Paris, France

REMERCIEMENT

Je tiens avant tout à exprimer ma profonde gratitude envers L'École La Passerelle des Métiers du Numérique - La PMN. Leur confiance, ainsi que l'accompagnement précieux que j'ai reçus, m'ont guidé tout au long de la réalisation de mon projet professionnel.

Je remercie également mes professeurs de la PMN pour la qualité de leurs enseignements techniques, qui ont largement contribué à enrichir mes compétences. Je tiens à adresser des remerciements particuliers à Madame Magalie Chatellard, ma tutrice, pour son soutien constant, ses conseils avisés, et son encouragement tout au long de ce parcours.

Je souhaite adresser une mention spéciale à mon manager, 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 gratitude s'étend également à mes collègues chez GE Healthcare, dont les conseils avisés et le soutien inestimable ont été d'une grande aide tout au long de mon alternance.

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

RESUME

Ce livre blanc décrit le développement et la transformation d'IBTool, une application web interne de GE Healthcare conçue pour examiner, suivre et gérer les plaintes des clients. Initialement déployée en tant que solution on-premises, IBTool rencontrait plusieurs problèmes majeurs : une scalabilité limitée qui freinait son expansion, des coûts de maintenance élevés qui pesaient sur les ressources, et des difficultés à garantir une haute disponibilité, ce qui pouvait affecter la satisfaction client.

Pour résoudre ces défis, l'application a été migrée vers une architecture cloud-native sur Amazon Web Services (AWS). Le projet a débuté avec la conception et le déploiement initiaux d'IBTool sur une infrastructure locale, puis a évolué vers une architecture cloud intégrant des services AWS tels qu'Amazon ECS, EC2, et l'Application Load Balancer (ALB). Un pipeline CI/CD automatisé a été mis en place avec CodeCommit, CodeBuild, et CodePipeline pour améliorer le processus de déploiement.

Cette transformation a permis de résoudre les problèmes clés identifiés : la scalabilité a été augmentée grâce à l'auto-scaling, permettant à IBTool de répondre efficacement aux fluctuations de la demande ; la haute disponibilité a été renforcée par la répartition du trafic sur plusieurs zones de disponibilité AWS ; et les coûts de maintenance ont été réduits grâce à l'automatisation des déploiements, limitant les erreurs humaines.

Ce projet ouvre également la voie à des améliorations futures, telles que l'intégration d'outils d'analyse avancés, l'adoption de l'apprentissage automatique, et l'exploration de solutions serverless, visant à optimiser encore davantage les performances et la flexibilité d'IBTool.

TABLE OF CONTENTS

1. INTRODUCTION ET CONTEXTE DU PROJET

Dans le domaine en constante évolution des technologies de la santé, la gestion efficace des retours clients, en particulier des plaintes liées aux produits logiciels, est essentielle pour maintenir des standards élevés de qualité et de satisfaction client. Chez GE Healthcare, où sont développées des technologies d'imagerie médicale innovantes telles que les systèmes IRM et CT, la gestion rapide et efficace des plaintes des clients a un impact direct sur la fiabilité des produits et la confiance des utilisateurs.

1.1 CONTEXTE ET MOTIVATION DU PROJET

Ce mémoire documente le développement et la migration ultérieure d'IBTool, une application web interne conçue pour rationaliser le processus de gestion des plaintes clients chez GE Healthcare. Le projet a été entrepris pour résoudre les limitations de l'ancien système on-premises utilisé par l'équipe Install Base (IB), qui était confronté à une fragmentation des données, à des processus manuels laborieux, et à une interface utilisateur peu intuitive. Ces problèmes ralentissaient non seulement la résolution des plaintes, mais posaient également des risques significatifs pour l'exactitude des données et l'efficacité opérationnelle.

1.2 OBJECTIFS ET CONTRIBUTIONS

L'objectif principal de ce projet était double : d'abord, développer un outil robuste et convivial capable de centraliser toutes les données liées aux plaintes, améliorant ainsi la synchronisation des données, la visibilité et l'efficacité globale de la gestion; ensuite, migrer cet outil vers une architecture cloud-native sur Amazon Web Services (AWS), afin d'améliorer la scalabilité, la disponibilité, et l'efficacité des coûts. La migration visait également à tirer parti des technologies cloud pour automatiser les déploiements, renforcer la sécurité et réduire les coûts d'exploitation.

Ce mémoire retrace l'évolution d'IBTool, depuis son développement initial dans un environnement on-premises traditionnel jusqu'à sa transformation en une application cloud-native. Il explore les défis rencontrés lors de ces deux phases, les solutions mises en œuvre, et l'impact de ces changements sur les opérations de GE Healthcare. En intégrant des pratiques de développement modernes et des technologies cloud, IBTool a non seulement

amélioré la manière dont les plaintes des clients sont gérées, mais a également ouvert la voie à de futures innovations au sein de l'entreprise.

1.3 PROBLEMATIQUE ET PERTINENCE DU PROJET

Le système de gestion des plaintes original souffrait de plusieurs défauts critiques : les données étaient dispersées sur plusieurs plateformes non intégrées, ce qui rendait difficile la cohérence des informations; la mise à jour manuelle des dossiers était longue et sujette aux erreurs; et le manque d'une interface intuitive freinait l'adoption par les utilisateurs et leur productivité. Ces problèmes ont conduit au développement d'un nouvel outil—IBTool—capable de consolider toutes les données pertinentes et de simplifier le flux de travail.

Cependant, même après le déploiement réussi d'IBTool en environnement on-premises, les limitations de l'infrastructure sont devenues apparentes. Le système avait du mal à évoluer pendant les périodes de forte utilisation, était exposé à des pannes potentielles en raison de défaillances matérielles, et engendrait des coûts de maintenance élevés. Ces défis ont souligné la nécessité d'une solution basée sur le cloud, capable de fournir la flexibilité, la fiabilité et l'efficacité économique requises.

1.4 STRUCTURE DU MEMOIRE

Ce mémoire est organisé pour d'abord aborder le développement et le déploiement d'IBTool dans sa forme on-premises, suivi d'un compte rendu détaillé de sa migration vers AWS. Il couvre les principes de conception, les choix technologiques et les méthodologies employées, ainsi qu'une analyse approfondie des résultats et des améliorations futures. Les sections finales offrent des perspectives sur les implications plus larges de ce projet pour GE Healthcare et d'autres organisations envisageant une transition vers des architectures cloud-native.

2. CONCEPTION ET ARCHITECTURE

2.1 ARCHITECTURE DE L'IBTOOL ON-PREMISES

Avant de migrer vers une architecture cloud-native, IBTool fonctionnait dans un environnement on-premises. Cette configuration était basée sur une architecture à trois niveaux : un frontend, un backend, et une base de données, tous orchestrés via Docker pour

faciliter le déploiement. Cette structure permettait à GE Healthcare de gérer efficacement les plaintes de ses clients.

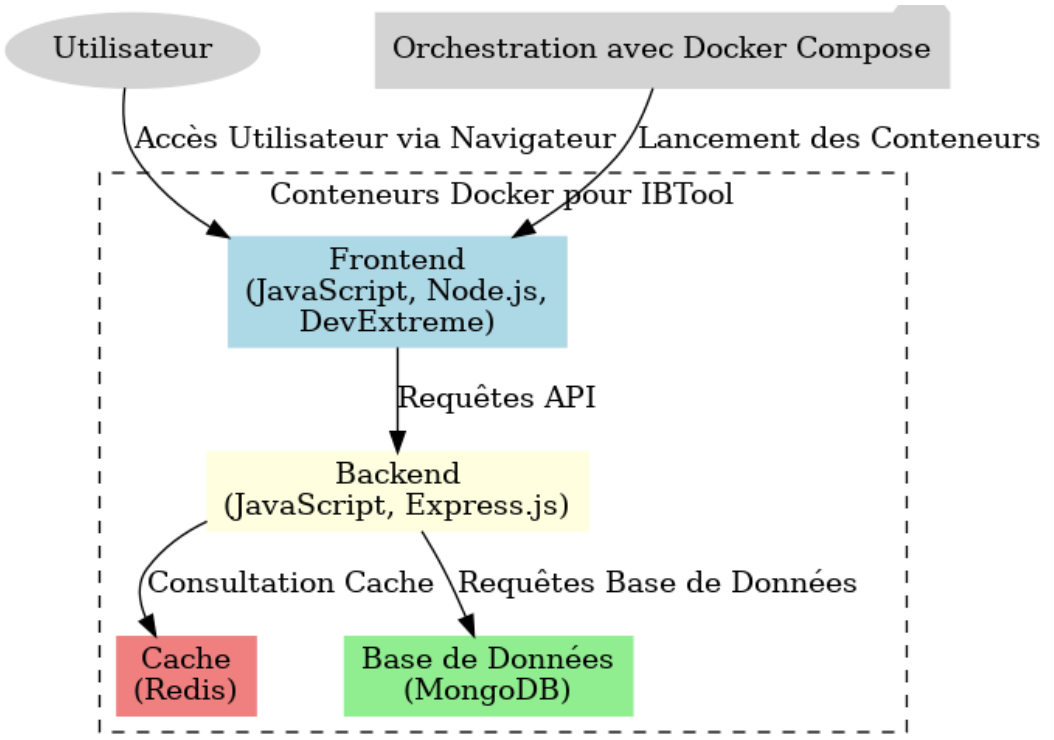

Figure 1 Architecture On-Premises d'IBTool avec Conteneurs Docker

L'infrastructure on-premises d'IBTool reposait sur des conteneurs Docker, chaque conteneur étant dédié à un composant spécifique de l'application. Le frontend, développé en JavaScript avec Node.js, HTML et CSS, s'appuyait sur la bibliothèque DevExtreme pour créer des composants d'interface utilisateur. Ce frontend interagissait avec le backend via des appels API pour traiter et afficher les données des plaintes. Le backend, également développé en JavaScript avec Express.js, gérait une API REST qui se connectait directement à MongoDB pour le stockage et la récupération des informations liées aux plaintes. MongoDB servait de dépôt central, offrant la flexibilité nécessaire pour gérer à la fois des données structurées et non structurées.

Pour optimiser les performances de l'application, un système de mise en cache a été mis en place. Côté client, la mise en cache dans le navigateur permettait de stocker temporairement les données fréquemment utilisées, réduisant ainsi les temps de réponse. Côté serveur, Redis

était utilisé pour minimiser les accès à la base de données, améliorant ainsi l'efficacité globale de l'application.

Toutefois, cette architecture on-premises présentait des limites significatives, notamment en termes de scalabilité, de maintenance et de résilience. L'ajout de nouvelles ressources nécessitait des interventions manuelles, et la gestion des serveurs physiques augmentait les coûts tout en exposant l'application à des risques de temps d'arrêt. Ces défis ont conduit à la décision de migrer IBTool vers une architecture cloud-native, plus adaptée aux besoins évolutifs de l'entreprise.

2.2 ARCHITECTURE DE L'IBTOOL CLOUD-NATIVE

La migration d'IBTool vers une architecture cloud-native a permis de tirer parti des avantages du cloud computing, offrant une plus grande scalabilité, une meilleure disponibilité et une sécurité renforcée. Contrairement à l'infrastructure on-premises, l'architecture cloud-native permet à IBTool de s'adapter dynamiquement aux fluctuations de la demande, garantissant des performances optimales sans les contraintes d'une infrastructure fixe.

L'architecture cloud-native repose sur une approche en microservices, où chaque composant de l'application – frontend, backend, et base de données – est déployé dans des conteneurs Docker, gérés par AWS Elastic Container Service (ECS). Cette approche permet de déployer et de mettre à l'échelle chaque composant indépendamment, selon les besoins. Un Application Load Balancer (ALB) répartit le trafic entrant entre les différents services ECS, assurant ainsi une haute disponibilité et une meilleure résilience de l'application. Les performances et la santé de l'application sont surveillées en continu grâce à AWS CloudWatch, ce qui permet une gestion proactive et une réponse rapide aux incidents.

La sécurité des données a été considérablement améliorée grâce à l'intégration de services AWS tels que IAM pour la gestion des accès, KMS pour le chiffrement des données, et ACM pour la gestion des certificats SSL/TLS. Cette infrastructure cloud-native offre non seulement une meilleure protection des données, mais elle réduit également les coûts opérationnels tout en augmentant l'efficacité globale d'IBTool.

La conception de l'architecture cloud-native d'IBTool a été pensée pour maximiser les avantages des services AWS, en garantissant une scalabilité, une sécurité et une disponibilité optimales. Les conteneurs Docker, utilisés pour chaque composant de l'application, sont gérés par ECS, tandis que les images Docker sont stockées dans AWS Elastic Container Registry (ECR). Les instances EC2, organisées en groupes d'Auto Scaling, fournissent les ressources de calcul nécessaires pour exécuter ces conteneurs, en ajustant leur nombre en fonction de la charge de travail.

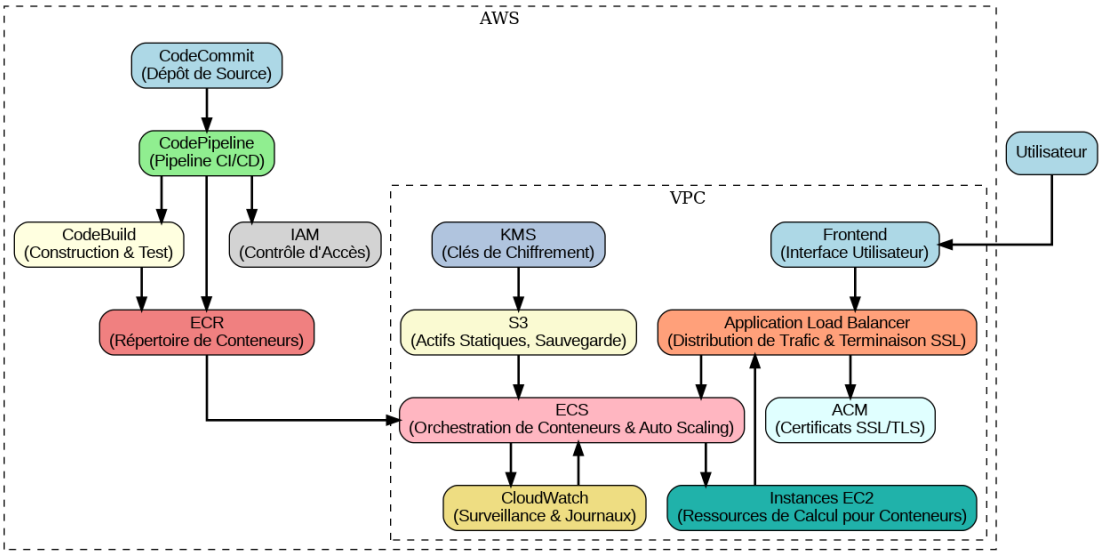

Figure 2 Diagramme d'architecture IBTool cloud native

La sécurité des données est une priorité, avec IAM qui gère les accès aux ressources AWS, KMS qui assure le chiffrement des données, et ACM qui gère les certificats SSL/TLS pour sécuriser les communications. AWS S3 est utilisé pour le stockage durable et évolutif des données critiques, tandis que le pipeline CI/CD automatisé via CodePipeline assure des déploiements rapides et sans interruption.

En adoptant cette architecture cloud-native, IBTool bénéficie d'une performance améliorée, d'une résilience accrue et d'une gestion simplifiée, ce qui permet à GE Healthcare de répondre plus efficacement aux besoins évolutifs de ses clients.

3. MISE EN ŒUVRE ET DEPLOIEMENT

3.1 DEVELOPPEMENT D'IBTOOL

Le développement d'IBTool a été une étape cruciale pour GE Healthcare, visant à améliorer la gestion des plaintes clients. Le projet s'est concentré sur deux principaux composants : le front-end et le back-end.

3.1.1 LE FRONT-END

Le front-end, développé avec des technologies web modernes comme HTML, CSS, et JavaScript, constitue l'interface par laquelle les utilisateurs interagissent avec l'application. Des outils comme Node.js et DevExtreme ont été utilisés pour créer une interface réactive et fluide, optimisant ainsi l'expérience utilisateur.

3.1.2 LE BACK-END

Le back-end, construit avec Node.js et Express.js, gère les requêtes des utilisateurs, applique les règles métier et interagit avec MongoDB pour stocker les informations relatives aux plaintes. MongoDB, choisi pour sa flexibilité dans la gestion des données non structurées, permet de répondre efficacement aux différents besoins de stockage.

Tout au long du développement, une approche agile a été adoptée, permettant une adaptation rapide aux besoins changeants. Chaque modification était rigoureusement testée avant d'être intégrée dans le code principal, assurant ainsi une qualité constante. Pour garantir la cohérence et la portabilité entre les environnements, Docker a été utilisé pour containeriser le front-end et le back-end, simplifiant ainsi le déploiement.

3.2 DEPLOIEMENT ON-PREMISES

Le déploiement d'IBTool dans un environnement on-premises chez GE Healthcare a permis de maintenir un contrôle strict sur l'infrastructure et la sécurité, tout en s'intégrant parfaitement aux infrastructures existantes. Cette méthode de déploiement, bien qu'efficace pour

répondre aux besoins actuels, présente des défis en termes de scalabilité et de coûts de maintenance.

3.3 MIGRATION VERS LE CLOUD

Pour relever les défis de scalabilité et de coûts de maintenance, une migration vers une architecture cloud-native a été envisagée. L'utilisation de l'infrastructure en tant que service (IaaS) sur Amazon Web Services (AWS) a permis d'automatiser la gestion et le déploiement de l'infrastructure.

3.3.1 INFRASTRUCTURE AS CODE (IAC) AVEC TERRAFORM

Grâce à Terraform, un outil d'infrastructure en tant que code (IaC), l'infrastructure cloud d'IBTool a été configurée, déployée, et gérée de manière programmée et reproductible. Cette approche a permis de réduire le temps de déploiement, d'assurer une cohérence entre les environnements, de mieux gérer la scalabilité, et d'optimiser les coûts.

3.3.2 GESTION DU CODE SOURCE ET DE DEPLOIMENT

La gestion du code source a été centralisée dans Amazon CodeCommit, avec des politiques d'accès strictes pour garantir la sécurité.

AWS CodeBuild a automatisé le processus de construction, assurant que chaque modification du code est systématiquement testée et prête pour le déploiement. AWS CodePipeline a ensuite automatisé l'ensemble du processus de déploiement, depuis la validation du code jusqu'à sa mise en production.

Pour garantir une gestion efficace des conteneurs, IBTool utilise Amazon Elastic Container Registry (ECR) pour stocker les images Docker. Ces images sont ensuite déployées sur Amazon Elastic Container Service (ECS), qui orchestre leur gestion et leur scalabilité en fonction des besoins.

3.3.3 SCALABILITE ET GESTION DU TRAFIC

Des groupes d'Auto Scaling sur EC2 assurent que les ressources de calcul s'ajustent automatiquement à la charge de travail. Un Application Load Balancer (ALB) a été configuré pour gérer le trafic entrant, en distribuant les requêtes vers les services ECS appropriés et en surveillant en continu leur état pour maintenir la performance de l'application.

3.3.4 RENFORCEMENT DE LA SECURITE

La sécurité de l'infrastructure a été renforcée grâce à AWS Identity and Access Management (IAM) pour la gestion des rôles et des accès, AWS Key Management Service (KMS) pour le chiffrement des données sensibles, et AWS Certificate Manager (ACM) pour la gestion des certificats SSL/TLS, garantissant des communications sécurisées entre IBTool et ses utilisateurs.

3.3.5 SURVEILLANCE ET OPTIMISATION DES PERFORMANCES

Enfin, Amazon CloudWatch a été mis en place pour surveiller en temps réel les performances d'IBTool. Des métriques spécifiques ont été configurées pour alerter en cas de dépassement de seuils critiques, et CloudWatch Logs a été utilisé pour collecter et analyser les journaux, offrant des insights précieux pour le dépannage et l'optimisation continue des performances.

La transition d'IBTool vers une architecture cloud-native a permis d'améliorer sa scalabilité, sa flexibilité et sa sécurité, tout en optimisant les coûts et en automatisant les processus de déploiement et de gestion. Cette transformation assure que l'application est prête à répondre aux défis futurs tout en offrant une expérience utilisateur optimale.

4. RESULTS AND ANALYSIS

4.1 VUE D'ENSEMBLE ET OBJECTIFS DU PROJET

IBTool est une application web essentielle pour GE Healthcare, utilisée pour gérer les plaintes clients et assurer la qualité des produits. Initialement déployée sur une infrastructure on-premises, l'application rencontrait plusieurs défis, tels que des limitations en matière de scalabilité, des coûts de maintenance élevés, et une disponibilité limitée. Pour résoudre ces problèmes, une migration vers Amazon Web Services (AWS) a été mise en œuvre, avec pour

objectifs principaux de rendre l'application plus scalable, de réduire les coûts, d'améliorer la disponibilité, et de renforcer la sécurité.

4.2 PROCESSUS DE DEPLOIEMENT SUR AWS

La migration d'IBTool vers le cloud a été réalisée en plusieurs étapes méthodiques. Le code source de l'application a d'abord été centralisé dans un dépôt Amazon CodeCommit, garantissant un contrôle d'accès sécurisé grâce à des politiques IAM bien définies. Ensuite, le processus de construction du code a été automatisé avec AWS CodeBuild, permettant de compiler et tester les dernières modifications avant leur déploiement.

Les images Docker des différents services (frontend, backend, MongoDB) ont ensuite été centralisées dans Amazon Elastic Container Registry (ECR), facilitant leur gestion et leur déploiement. Ces images ont été déployées sur un cluster ECS configuré pour héberger les conteneurs Docker, chaque service étant géré et mis à l'échelle de manière indépendante.

Pour garantir une gestion efficace du trafic et maintenir une haute disponibilité, un Application Load Balancer (ALB) a été mis en place. Celui-ci répartit le trafic en fonction des besoins, tout en surveillant la santé des services pour rediriger automatiquement les requêtes en cas de défaillance.

La sécurité des données a été renforcée grâce à AWS Key Management Service (KMS) pour le chiffrement, et AWS Certificate Manager (ACM) pour la gestion des certificats SSL/TLS, assurant des communications sécurisées entre l'application et ses utilisateurs. Enfin, Amazon CloudWatch a été déployé pour surveiller les performances de l'application en temps réel, avec des alarmes et des tableaux de bord configurés pour réagir rapidement en cas d'anomalies.

4.3 INSIGHTS OPERATIONNELS

Après la migration, la surveillance continue des performances d'IBTool a révélé une amélioration significative de la réactivité et de la stabilité de l'application. Les métriques de CloudWatch ont permis de suivre l'utilisation des ressources en temps réel, ce qui a été

essentiel pour maintenir l'efficacité de l'application, notamment lors de pics de charge. Par exemple, lorsque l'utilisation du CPU a augmenté, des instances EC2 supplémentaires ont été automatiquement ajoutées pour maintenir la performance.

Les capacités d'auto-scaling d'ECS et d'EC2 ont permis à IBTool de s'adapter rapidement aux variations de la charge de travail, sans intervention manuelle. Cette flexibilité a été particulièrement utile lors des périodes de forte demande, où l'application a pu ajuster ses ressources en temps réel pour éviter les surcharges.

Les processus de maintenance ont également été optimisés grâce à des mises à jour automatiques via AWS CodePipeline, qui ont réduit les temps d'arrêt et minimisé les risques d'erreurs. De plus, des sauvegardes régulières de la base de données MongoDB sur Amazon S3 ont assuré la disponibilité des données, même en cas de panne.

La migration d'IBTool vers une architecture cloud-native sur AWS a non seulement résolu les limitations de l'infrastructure on-premises, mais a également permis d'améliorer considérablement la performance, la scalabilité, et la sécurité de l'application, tout en optimisant les coûts et en facilitant la maintenance.

5. CONCLUSION ET TRAVAUX FUTURS

La migration d'IBTool vers une architecture cloud-native sur AWS a été un tournant majeur, mais il est important de ne pas oublier les étapes cruciales qui ont précédé cette transition. Le développement initial d'IBTool a joué un rôle fondamental en optimisant l'analyse et le suivi des plaintes pour l'équipe IB de GE Healthcare. Ce développement a permis de centraliser et de structurer la gestion des plaintes, répondant ainsi aux besoins spécifiques de l'entreprise en matière de satisfaction client et d'assurance qualité des produits.

Au départ, IBTool a été conçu pour être déployé dans un environnement on-premises. Ce choix était motivé par la nécessité de garder un contrôle strict sur les données sensibles et de garantir la conformité avec les réglementations en vigueur dans le secteur de la santé. Le développement de l'application a impliqué la création d'un front-end intuitif et réactif, ainsi

qu'un back-end robuste capable de gérer des volumes importants de données de plaintes clients. MongoDB a été choisi comme base de données pour sa capacité à gérer des données non structurées, et l'utilisation de Docker a permis d'assurer une cohérence et une portabilité entre les environnements de développement et de production. Cependant, les limitations de l'infrastructure on-premises, telles que l'absence de scalabilité et les coûts de maintenance élevés, ont motivé la décision de migrer IBTool vers une architecture cloud-native. Cette migration a non seulement résolu ces problèmes, mais a aussi apporté des améliorations significatives en termes de performance, de sécurité, et d'efficacité opérationnelle.

Le recours à l'infrastructure as code (IaC) avec Terraform a simplifié la gestion des configurations complexes et a permis une automatisation des déploiements, réduisant ainsi les erreurs humaines et améliorant la cohérence entre les différents environnements. La sécurité des données a été renforcée grâce à l'intégration des services de sécurité AWS, assurant que les informations sensibles sont protégées de bout en bout.

La surveillance continue des performances via Amazon CloudWatch, combinée aux capacités d'auto-scaling d'ECS et d'EC2, a permis à IBTool de s'adapter dynamiquement aux variations de charge, garantissant une performance optimale tout en optimisant les coûts. Cette flexibilité et cette robustesse nouvellement acquises ont non seulement amélioré l'expérience utilisateur, mais ont également permis de rationaliser les opérations quotidiennes.

En perspective, il existe encore des opportunités pour renforcer IBTool. L'intégration d'outils d'analyse avancés et l'exploration de l'apprentissage automatique pourraient ajouter des dimensions prédictives à l'application, améliorant ainsi encore sa capacité à gérer les plaintes des clients. De plus, l'adoption d'architectures serverless pourrait réduire la charge opérationnelle tout en augmentant la flexibilité.

Le développement et la migration d'IBTool ont permis à GE Healthcare de disposer d'un outil puissant pour l'analyse et le suivi des plaintes, répondant ainsi aux besoins critiques de l'entreprise.