ACHRAF OUJJIR & LINA BELLAHMIDI

Génie Informatique

Ecole Mohammadia des Ingénieurs

RAPPORT DE STAGE

Du 10 Juillet au 25 Août 2023

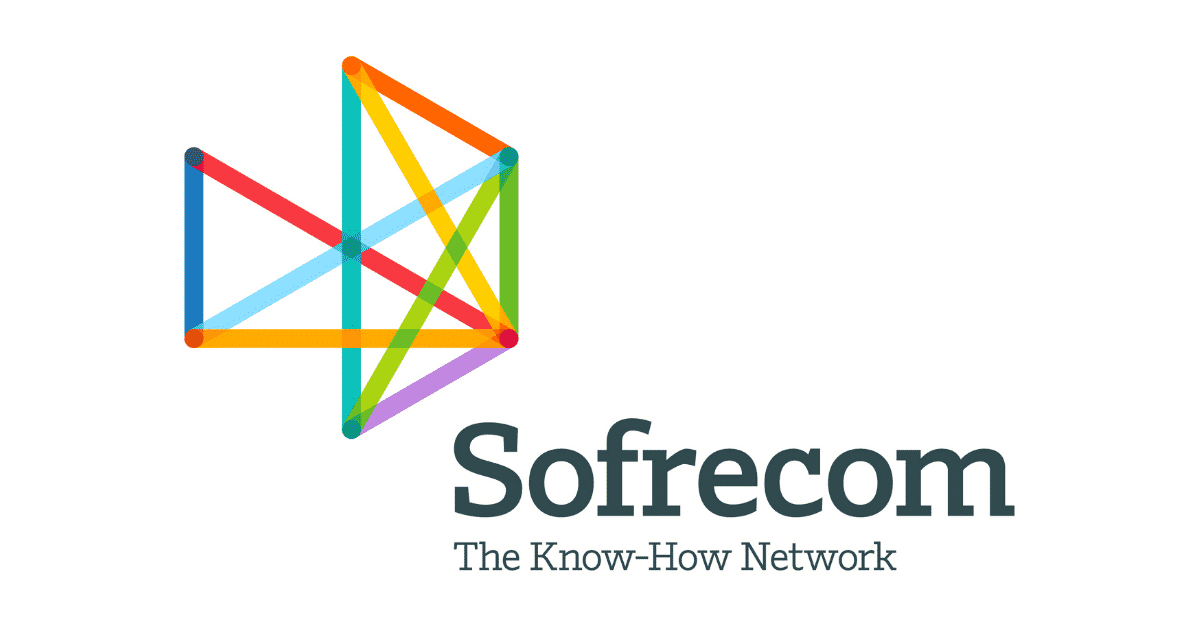
Au sein de l’entreprise Sofrecom

Sous la tutelle de Mr Zakaria DRISSI EL MILIANI Tech/Lead Ingénieur Data,

Mr Mouad ALLAT Ingénieur Data pour l’entreprise,

et de,

Mme Fatima-Zahra BELOUADHA chef du département informatique à l’EMI



# **REMMERCIEMENTS**

Je tiens tout d’abord à remercier l’Ecole Mohammadia des Ingénieurs, et particulièrement Mme Fatima-Zahra BELOUADHA pour m’avoir fait confiance. L’écoute et l’accompagnement dont j’ai bénéficié m’ont permis de trouver rapidement un stage, dans le but d’affiner mon projet professionnel et d’obtenir mon diplôme d’ingénieur d’état.

À ce titre, je souhaiterais remercier tout particulièrement Mr Zakaria DRISSI EL MILIANI, qui m’a épaulé(e) et conseillé(e) et qui m’a surtout transmis son expertise dans le domaine de la Data. Ainsi que son équipe, notamment Mr Mouad ALLAT Ingénieur data et Mr Abdelghani LAQDOUR.

Ce stage m’a permis d’affiner certaines pistes pour bâtir une carrière en data engineering et signe l’aboutissement de ma 1ère année du cycle ingénieur.

Je n’oublie pas non plus ma mère qui m’a sans cesse soutenue dans l’élaboration de mon projet professionnel et m’a aidée à chaque étape de ce rapport de stage.

Remerciements spéciaux à mon relecteur et correcteur qui a contribué, grâce à ses conseils et recommandations, à l’élaboration et au bon déroulé de mon rapport de stage.

SOMMAIRE

Table des matières

[**REMMERCIEMENTS** 2](#_Toc143595916)

[**LISTE DES TABLEAUX ET FIGURES** 3](#_Toc143595917)

[**LISTE DES ABREVATIONS** 5](#_Toc143595918)

[**GLOSSAIRE** 6](#_Toc143595919)

[**INTRODUCTION** 7](#_Toc143595920)

[**CHAPITRE 1 : DECRYPTAGE DE L’ENTREPRISE ET LE DEFI D’AUTOMATISER LE DEPLOIEMENT DES TEMPLATES NIFI** 8](#_Toc143595921)

[**I)** **SOFRECOM EN LUMIERE : UN REGARD APPROFONDIE SUR L’ENTREPRISE** 8](#_Toc143595922)

[**II)** **PROBLEMATIQUE** 10](#_Toc143595923)

[**III)** **FEUILLE DE ROUTE : MISSIONS ET OBJECTIFS DU PROJET** 10](#_Toc143595924)

[1) Simuler l’environnement de production : 10](#_Toc143595925)

[2) Pipeline de déploiement : 11](#_Toc143595926)

[**Chapitre 2 : Automatisation du déploiement des templates NiFi sur les instances NiFi** 12](#_Toc143595927)

[**I)** **Simulation de l'environnement de production localement :** 13](#_Toc143595928)

[1) Technologies utilisées : 13](#_Toc143595929)

[2) Installation de Minikube, Kubectl et Helm : 13](#_Toc143595930)

[3) Configuration du cluster avec Nifikop et Zookeeper : 14](#_Toc143595931)

[4) Création du cluster et des pods : 14](#_Toc143595932)

[5) Aperçu global : 15](#_Toc143595933)

[6) Accès http://localhost:8080: 16](#_Toc143595934)

[**II)** **Conception et Architecture de la pipeline :** 17](#_Toc143595935)

[1) Préparation du script python nifi\_templates\_deploy.py 18](#_Toc143595936)

[2) Préparation du script main.tf : 19](#_Toc143595937)

[3) Création du pipeline sur GitLab avec le fichier .gitlab-ci.yml: 19](#_Toc143595938)

[4) Conclusion : 20](#_Toc143595939)

[**CONCLUSION** 22](#_Toc143595940)

[**BIBLIOGRAPHIE** 23](#_Toc143595941)

[**ANNEXES** 24](#_Toc143595942)

# **LISTE DES TABLEAUX ET FIGURES**

Figure 1 : Aperçu global du projet 8

Figure 2 : Logo de l’entreprise Sofrecom 8

Figure 3: NiFi sur GKE 10

Figure 4: Commandes de création de l’environnement Minikube 13

Figure 5 : Création du conteneur Minikube sur Docker 13

Figure 6: Commandes d’installation de NiFiKop et Zookeeper 14

Figure 7 : Commandes de création du NiFi Cluster 14

Figure 8 : Aperçu du fihier persistentnificluster.yaml 15

Figure 9 : Tableau de bord Kubernetes (1) 15

Figure 10 : Tableau de bord Kubernetes (2) 15

Figure 11 : NiFi accessible par le navigateur 16

Figure 12 : Architecture du projet 18

Figure 13 : Script du déclanchement de la pipeline 20

# **LISTE DES ABREVATIONS**

# **GLOSSAIRE**

# **INTRODUCTION**

Du 15 juillet au 25 aout 2023, j’ai eu la chance de rejoindre le département Data de Sofrecom en tant que stagiaire sous la gouvernance de Mr. Zakaria DRISSI EL MILIANI. Naturellement attiré par le milieu de la data, j’ai très rapidement décidé d’orienter mes recherches de stage en ce sens, ayant pour projet de mêler passion et vie professionnelle au sortir des études.

Sofrecom, un cabinet de conseil en télécommunications de premier plan, a réalisé d'incroyables avancées dans le domaine des données. Réputée pour son expertise, l'entreprise exploite l'analyse avancée et des solutions sur mesure pour autonomiser les opérateurs télécoms. De la maintenance prédictive à l'optimisation des réseaux et à l'amélioration des expériences utilisateur, l'approche novatrice axée sur les données de Sofrecom a redéfini les normes de l'industrie. Avec un accent sur l'IoT, la 5G et les technologies émergentes, l'entreprise continue de façonner le paysage des télécommunications grâce à des informations et des solutions basées sur les données.

Intégré au département d'ingénierie des données, j'ai eu l'occasion de participer à la définition de stratégies visant à atteindre les objectifs de l'équipe. Cette implication s'est concrétisée notamment par des missions axées sur l'exploration et l'intégration de nouvelles technologies. Mon rôle principal consistait à m'initier à l'outil NiFi, où j'ai rapidement développé une compréhension approfondie de ses fonctionnalités. En parallèle, j'ai activement contribué à un projet majeur : l'automatisation du déploiement des templates NiFi, une composante cruciale pour l'infrastructure data. Cette expérience m'a permis d'appliquer mes nouvelles connaissances et de jouer un rôle concret dans l'amélioration de l'efficacité opérationnelle de l'équipe.

Ce stage a également été l’occasion pour moi de découvrir le monde professionnel. En rejoignant les rangs de Sofrecom, je me suis tout d’abord familiarisée avec le fonctionnement interne d’une entreprise. Rapidement mis à l’aise par les membres de mon équipe, j’estime avoir beaucoup appris au cours de ces deux mois et ai désormais la certitude d’avoir fait le bon choix d’orientation.

Afin de refléter au mieux ces quelques semaines et le contexte dans lequel s’est déroulé mon arrivée, il semble approprié de se pencher, en premier lieu, sur la place de Sofrecom dans le secteur de la data en France, mais également aux nouveaux objectifs de l’entreprise et à la problématique de l’évolution des attentes de sa clientèle. Enfin, il sera question de la nature des missions qui m’ont été confiées et des différents savoirs que j’ai pu acquérir.

# **CHAPITRE 1 : DECRYPTAGE DE L’ENTREPRISE ET LE DEFI D’AUTOMATISER LE DEPLOIEMENT DES TEMPLATES NIFI**



Au cœur du contexte professionnel se déploie un univers de données en perpétuelle mutation. Cette section se propose d'entreprendre une exploration approfondie de cet environnement tout en abordant les enjeux stratégiques qui encadrent l'automatisation des templates NiFi. Une analyse minutieuse des défis techniques et organisationnels qui ont émaillé mon stage sera présentée dans ce qui suit.

Figure 1 : Aperçu global du projet

## **SOFRECOM EN LUMIERE : UN REGARD APPROFONDIE SUR L’ENTREPRISE**

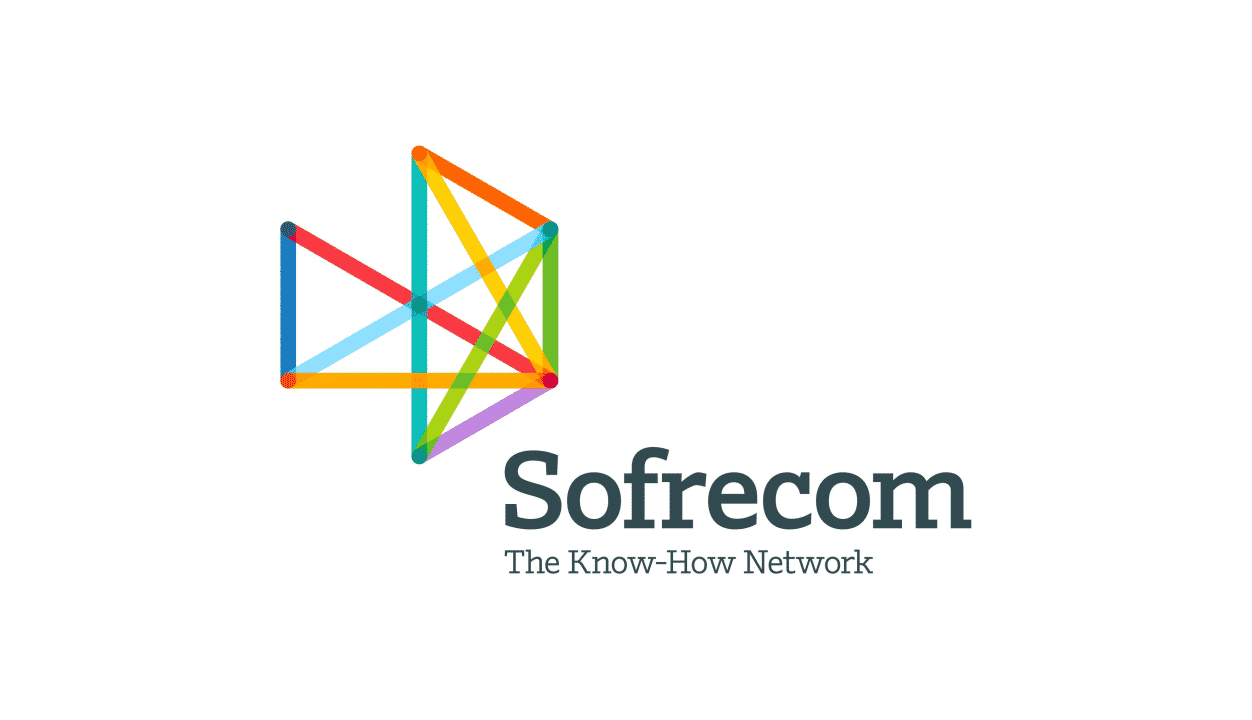
Sofrecom est une société de conseil et d'ingénierie spécialisée dans les domaines des télécommunications et de la transformation digitale. Fondée en 1968 en France, Sofrecom est devenue une référence internationale dans le secteur, apportant son expertise à de nombreux opérateurs de télécommunications, fournisseurs de services et entreprises du monde entier. L'une des facettes impressionnantes de l'expérience de Sofrecom réside dans son engagement et ses exploits remarquables dans le domaine de la data, qui ont grandement contribué à façonner l'avenir de l'industrie.

Figure 2 : Logo de l’entreprise Sofrecom

Au fil des années, Sofrecom a développé une solide compétence dans la collecte, le traitement et l'analyse de données massives, souvent désignées sous le terme "Big Data". Grâce à sa maîtrise des technologies émergentes et à son approche innovante, la société a été à l'avant-garde de l'exploitation des données pour aider les entreprises à prendre des décisions plus éclairées, à améliorer leur efficacité opérationnelle et à mieux comprendre les besoins de leurs clients.

L'exploit le plus notable de Sofrecom dans le domaine de la data réside dans sa capacité à concevoir et à mettre en œuvre des solutions sur mesure pour les opérateurs télécoms, leur permettant d'optimiser leurs réseaux, d'anticiper les pannes, de gérer le trafic de manière efficace et d'améliorer la qualité des services offerts. Grâce à des techniques avancées d'analyse prédictive et de machine learning, Sofrecom a aidé ses clients à transformer des volumes massifs de données en informations exploitables, créant ainsi un avantage concurrentiel significatif.

En outre, Sofrecom a contribué de manière significative au développement de projets innovants liés à l'Internet des objets (IoT), à la 5G et à d'autres technologies émergentes. Ses experts en data ont collaboré avec des opérateurs pour déployer des solutions de gestion intelligente des réseaux et des services, offrant une expérience utilisateur optimale tout en maximisant l'efficacité des ressources.

En résumé, Sofrecom a laissé une empreinte remarquable dans le domaine de la data en offrant des solutions novatrices qui ont permis aux entreprises de mieux exploiter leur potentiel, de tirer parti des données à leur disposition et de façonner l'avenir de l'industrie des télécommunications et de la transformation digitale.

## **PROBLEMATIQUE**

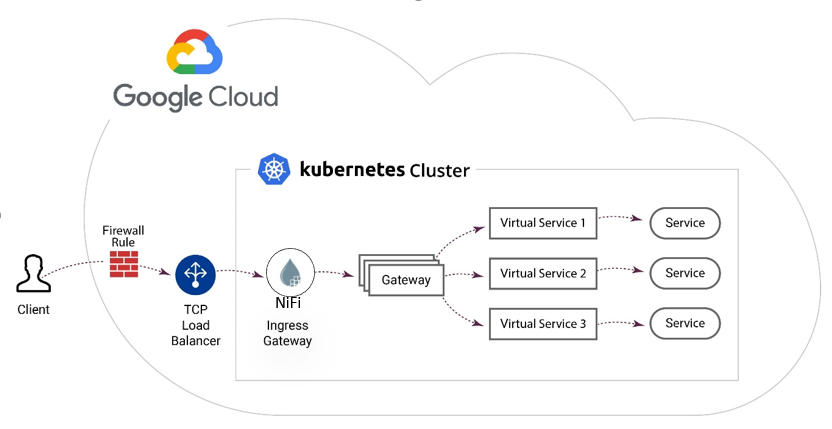
La problématique réside dans le processus manuel et répétitif de déploiement des modèles NiFi, qui engendre des retards significatifs et ouvre la porte à d'éventuelles erreurs humaines. Cette situation devient particulièrement préoccupante lorsque l'on réalise l'importance cruciale des modèles NiFi dans les tâches quotidiennes des ingénieurs en traitement de données. Ces professionnels comptent sur les modèles NiFi pour éviter de devoir créer à maintes reprises des flux de données identiques, parfois complexes et chronophages à configurer. Ces modèles, généralement au format XML, sont ensuite déployés manuellement par les ingénieurs sur les instances de NiFi. Cela se fait au sein des clusters NiFi hébergés sur GKE (Google Kubernetes Engine), une infrastructure mise à disposition spécifiquement pour les besoins de ces ingénieurs.

Figure 3: NiFi sur GKE

## **FEUILLE DE ROUTE : MISSIONS ET OBJECTIFS DU PROJET**

Au cœur de ce projet, se trouvent des missions cruciales et des objectifs clés visant à surmonter les défis associés au déploiement manuel des modèles NiFi. Dans cette perspective, deux étapes majeures se distinguent : la simulation de l'environnement GKE par le biais de Minikube, suivie de l'établissement d'une pipeline de déploiement. Cette section met en lumière ces deux étapes fondamentales, démontrant comment elles s'intègrent pour résoudre la problématique liée à l'efficacité et à la précision du processus de déploiement.

### Simuler l’environnement de production :

Dans le cadre de notre collaboration, mon binôme et moi avons entrepris une étape essentielle : recréer l'environnement GKE (Google Kubernetes Engine) en utilisant l'outil Minikube. Cette démarche s'est avérée cruciale, car Minikube nous a offert la possibilité de simuler localement un cluster Kubernetes, constituant ainsi une représentation virtuelle fidèle de notre environnement cible, le GKE. Grâce à cette approche, nous avons pu travailler au sein d'un espace contrôlé et isolé, où nous avons pu émuler toutes les caractéristiques et les fonctionnalités spécifiques au GKE. Ce processus nous a permis d'expérimenter diverses configurations, d'effectuer des tests approfondis et d'apporter des ajustements en toute sécurité. Ces efforts préliminaires s'avéreront sans aucun doute essentiels pour le déploiement ultérieur dans un environnement GKE réel, en nous donnant une base solide pour la réussite de notre projet.

### Pipeline de déploiement :

Après avoir méticuleusement simulé l'environnement GKE à l'aide de Minikube, mon binôme et moi avons amorcé la conception et l'exécution d'une pipeline ingénieuse. Cette pipeline a été spécifiquement conçue pour automatiser de bout en bout le déploiement et l'instanciation des modèles NiFi. Grâce à cette infrastructure bien pensée, nous avons eu la possibilité de réaliser des tests exhaustifs pour nous assurer que chaque étape du processus fonctionne. Cette approche méthodique nous a permis de garantir que les modèles NiFi sont déployés de manière cohérente et précise sur les instances adéquates, tout en éliminant les erreurs potentielles liées à des interventions manuelles. Notre travail sur cette pipeline représente une avancée significative vers un processus de déploiement plus efficace et fiable, contribuant ainsi à l'optimisation globale de notre projet.

# **Chapitre 2 : Automatisation du déploiement des templates NiFi sur les instances NiFi**

Comme mentionné dans la problématique ci-dessus, le problème du déploiement manuel répétitif des templates NiFi prend beaucoup de temps et laisse une très grande marge pour l’erreur humaine. Ceci est très palpable quand on découvre l’importance des templates NiFi dans le travail journalier des Data Engineers. Ces derniers utilisent les templates NiFi afin de ne pas avoir à faire et refaire les mêmes DataFlows qui s’avèrent parfois très longs et qui prennent beaucoup de temps à composer et à paramétrer. Il y a plusieurs exemples de DataFlows qui se composent de plusieurs ProcessGroups et de plusieurs Processors. En total, ceci peut arriver à des centaines de composantes qui sont prédéfinis. Ceci facilite largement le travail des Data Engineers qui n’ont donc plus qu’à changer les paramètres de contexte et personnaliser le Data Flow si cela s’avère nécessaire.

On peut donc dire que les templates NiFi aident les Data Engineers à être plus productifs et à avoir plus de temps à consacrer pour des problèmes plus difficiles. En plus d’avoir des templates prêts à être utilisés qui sont partagés entre les Data Engineers de la société dans un Repository dédié aux templates. Ces templates (en format xml) sont alors déployés par les Data Engineers sur des instances de NiFi sur les NiFiCluster de GKE (Google Kubernetes Engine) qui sont mis en disposition des Data Engineers.

Pour faciliter encore plus la chose et accélérer le déploiement des templates, nous avons implémenté un pipeline afin d’automatiser le téléchargement et l’instanciation des templates NiFi sur les instances NiFi au sein de Minikube. Pour ce faire, nous avons utilisé une panoplie d’outils et de technologies proposés par notre encadrant. La première partie de ce chapitre va couvrir alors les technologies utilisées lors de notre projet et qui ont été d’une grande utilité

Afin de concevoir et d’implémenter notre pipeline, nous avons tout d’abord eu à simuler l’environnement GKE sur nos machines locales afin de tester et améliorer notre solution pour qu’elle réponde au mieux au besoin exprimé par les Data Engineers. Pour cette simulation de l’environnement de Production, nous avons utilisé Minikube et l’opérateur Nifikop pour le déploiement d’un NifiCluster qui nous a servi comme environnement de test. La deuxième partie de ce chapitre va couvrir comment on a pu simuler l’environnement de Production le plus proche à la réalité afin de pouvoir concevoir et implémenter la solution la plus réaliste possible à notre problématique.

La troisième partie de ce chapitre va couvrir l’architecture qu’on a choisie pour notre pipeline et qui comporte les outils et technologies mentionnés précédemment. Le diagramme d’architecture dans cette partie montre comment les outils mentionnés sont utilisés en synergie afin de garantir la bonne implémentation du pipeline. Nous avons également consacré une partie pour l’implémentation de notre solution qui montre des captures d’écran exhibant comment on a pu déployer automatiquement et juste par un le déclenchement de notre pipeline un ensemble de templates (en format xml) vers une instance NiFi sur notre cluster Minikube tout en gardant une traçabilité pour la date du déploiement et les templates déployés.

## **Simulation de l'environnement de production localement :**

### Coders Classroom: Kubernetes for Developers #3: kubectl CLIHELM Logo Helm, Software, Logos, LogoTechnologies utilisées :

****



**NiFiKop**

### Installation de Minikube, Kubectl et Helm :

Minikube nous a permis de créer une version miniature d'un cluster Kubernetes au sein de notre propre machine, fournissant une plateforme de test fiable. Cela nous a permis d'expérimenter, de développer et de déployer des applications dans un environnement de type Kubernetes, tout en évitant la complexité d'une infrastructure complète. Kubectl, d'autre part, s'est révélé indispensable pour interagir avec le cluster Kubernetes, nous permettant de déployer et de gérer des ressources, de surveiller l'état des pods et de contrôler divers aspects du cluster. Enfin, l'utilisation de Helm a grandement facilité la gestion des déploiements d'applications en automatisant la configuration, la gestion des dépendances et la mise à l'échelle des applications.

**a) Commandes**



Figure 4: Commandes de création de l’environnement Minikube

**b) Aperçu**

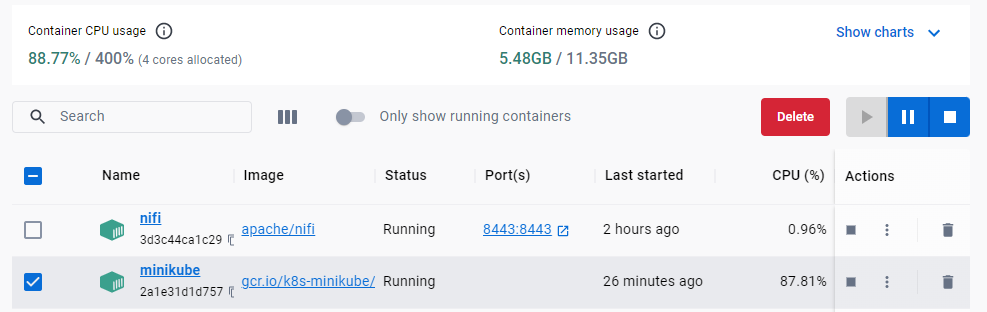


Figure 5 : Création du conteneur Minikube sur Docker

### Configuration du cluster avec Nifikop et Zookeeper :

Nifikop, un opérateur Kubernetes spécialement conçu pour Apache NiFi, a joué un rôle essentiel dans la simplification du déploiement et de la gestion des instances NiFi au sein du cluster. En utilisant le graphique Helm de Nifikop, nous avons pu configurer et allouer de manière transparente des ressources, garantissant des performances optimales tout en respectant les contraintes de ressources. Cela a accéléré le processus de configuration de NiFi au sein du cluster, facilitant ainsi les phases de test et de développement. Zookeeper, quant à lui, s'est révélé être un composant crucial pour soutenir les opérations de NiFi. Déployé à l'aide de Helm, Zookeeper a établi un service de coordination résilient et distribué, indispensable pour maintenir la fiabilité des systèmes distribués de NiFi.



Figure 6: Commandes d’installation de NiFiKop et Zookeeper

### Création du cluster et des pods :

#### **Commandes**

****

Figure 7 : Commandes de création du NiFi Cluster

#### **Aperçu du fichier de configuration**



Figure 8 : Aperçu du fihier persistentnificluster.yaml

### Aperçu global :

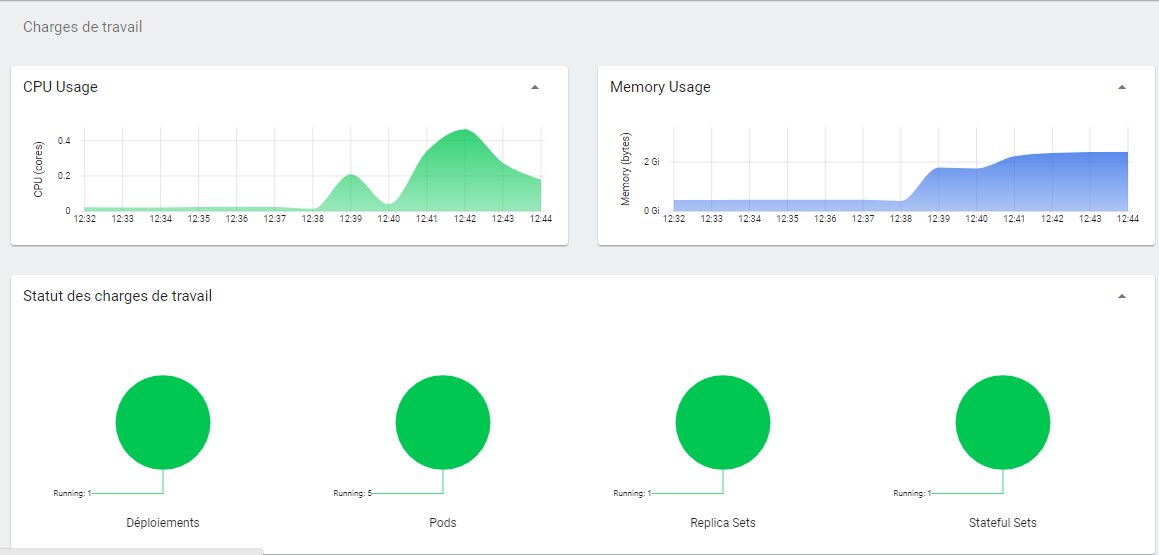


Figure 9 : Tableau de bord Kubernetes (1)

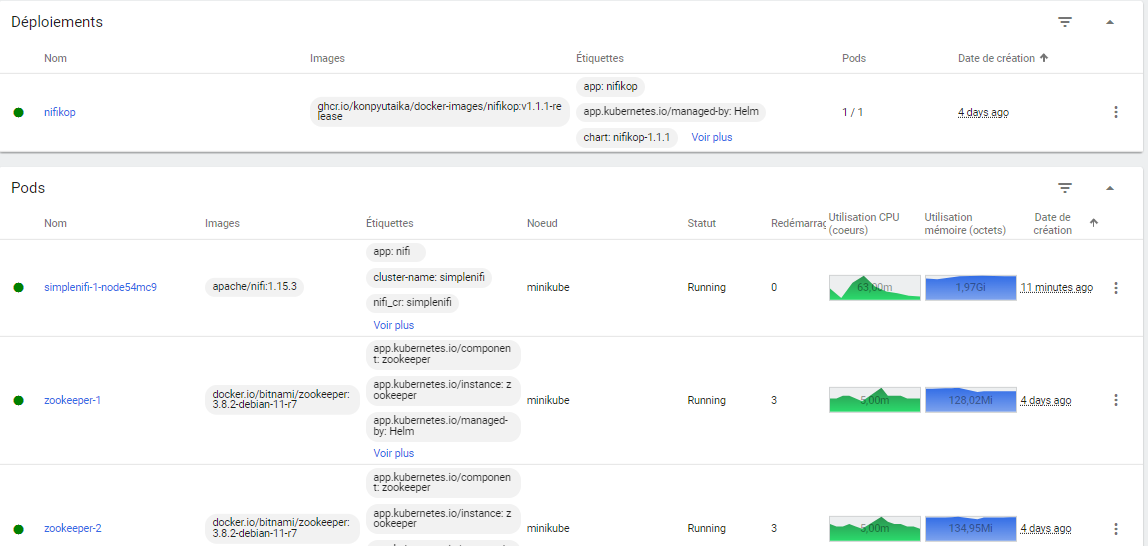


Figure 10 : Tableau de bord Kubernetes (2)

### Accès <http://localhost:8080>:

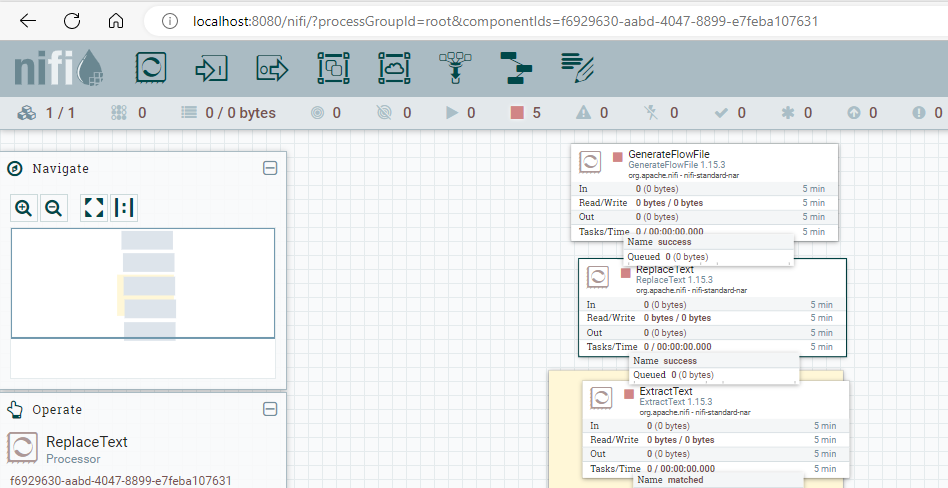


Figure 11 : NiFi accessible par le navigateur

## **Conception et Architecture de la pipeline :**

Afin d’implémenter notre pipeline, nous avons utilisé plusieurs outils et services qui nous ont permis de garantir le bon déroulement des étapes et l’automatisation de tout le processus de déploiement. Les outils que nous avons utilisés sont : GitLab, Terraform, Docker, Minikube, Nifikop et Nifi.



Figure 12 : Architecture du projet

Comme illustré en détail dans le schéma architectural présenté ci-dessus, notre pipeline trouve son point de départ dans l'environnement de GitLab. Ce mécanisme s'enclenche sur notre machine locale avec GitLab Runner, qui orchestre habilement les actions requises. La raison d’avoir utilisé GitLab Runner est pour avoir accès à notre cluster Minikube local afin de tester le pipepline avant de travailler sur GKE.

Poursuivant le cheminement du processus, nous rencontrons le script principal Terraform, baptisé "main.tf", jouant un rôle crucial. Ce script ne se contente pas uniquement de garantir le bon fonctionnement de Minikube, mais il agit également en tant que catalyseur pour l'exécution du script python "nifi\_templates\_deploy.py".

Le script python en question joue alors son rôle, interagissant avec l’API Rest offerte de NiFi afin d’uploader le template originalement déposé comme fichier xml et l’instancier sur le pod Nifi en question. Ce pipleine permet le déploiement des modèles (templates) spécifiés.

**Implémentation**

Pour l’implémentation de notre architecture, nous avons suivi les étapes suivantes. Nous avons commencé tout d’abord par la simulation de l’environnement de production dans lequel nous allons tester de façon itérative nos scripts et finalement tout notre pipeline. Lors de l’implémentation de notre architecture, nous avons commencé par les éléments les plus proches à l’instance de NiFi.

### Préparation du script python nifi\_templates\_deploy.py

C’est le script principal de notre pipeline et qui va interagir directement avec l’API Rest de Nifi. Ce script se compose de 4 parties :

*a) Récupérer du Token d’accès*

*b) Uploader les fichier xml du templates spécifiés*

*c) Instancer des ProcessGroup pour les templates uploadés*

*d) Déplacer les fichiers xml des templates instanciés vers un nouveau répertoire qui a une nomination correspondant à la date du jour du déploiement.*

Ces quatre étapes étaient faites manuellement et pour chaque template ce qui posait un problème pour les data engineers puisque ça prend beaucoup de temps surtout pour les templates qu’on veut instancier plusieurs fois. Faire cette tâche manuellement peut donner une grande marge d’erreur humaine qu’on peut éviter. C’est donc une problématique qui diminue la productivité des collaborateurs et met en risque le bon travail des dataflows NiFi.

Pour la première partie, elle n’est possible que quand on travaille avec le protocole https. Pour la deuxième, on a utilisé l’endpoint correspondant pour faire le téléchargement des fichiers xml tout en retenant les ids de ces templates pour pouvoir les instancier dans l’étape suivante en tant que ProcessGroups. Enfin, pour avoir une certaine traçabilité des templates déployés, nous avons travaillé sur un process qui va faire en sorte que les templates téléchargés et instanciés soient déplacés vers un autre dossier qui a comme nom la date du jour. Cette dernière étape a été faite pour garder une certaine traçabilité des templates.

### Préparation du script main.tf :

Pour ce script, il a pour rôle de s’assurer que le cluster Minikube est bien en marche et qu’on pourra procéder pour l’exécution du script nifi\_templates\_deploy.py. Il ajoute également une autre couche d’abstraction sur le script python.

### Création du pipeline sur GitLab avec le fichier .gitlab-ci.yml:

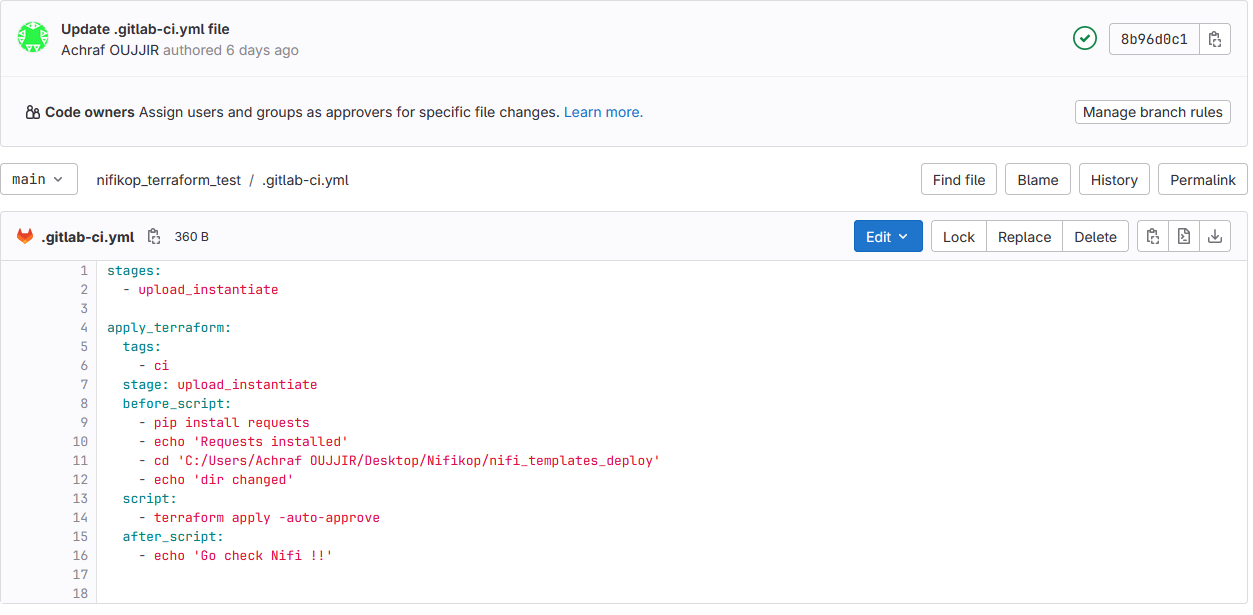
Pour ce pipeline, nous avons listé les étapes à accomplir par GitLab Runner afin d’exécuter le script main.tf sur notre machine locale qui va à son tour exécuter le script python qui va se charger du téléchargement et de l’instanciation des templates en question sur l’instance de NiFi à utiliser. On peut voir ci-dessous une capture d’écran du script de notre pipeline :

Figure 13 : Script du déclanchement de la pipeline

Comme le montre la capture d’écran ci-dessus, nous avons énoncé le stage nommé upload\_instanciate qui est le seul stage de notre pipeline et dans lequel nous avons inclus un seul job apply\_terraform. Ce job sera responsable du déclenchement du fichier main.tf sur notre machine locale.

Avant de déclencher le script terraform sur notre machine locale, nous avons inclus plusieurs commandes dans la partie before\_script afin de garantir le bon déroulement du pipeline. Parmi ces commandes, on a tout d’abord l’installation du package requests qui sera indispensable pour l’exécution de notre script python et ensuite le changement du working directory vers celui comportant le script python ainsi que les templates à télécharger et instancier. Il est à mentionner que nous avons ajouté des commandes echo qui nous ont aidé lors du débogage de notre pipeline et qui nous confirme le bon déroulement des commandes.

Pour le script principal de notre pipeline, nous avons comme commande principale le déclenchement du script main.tf avec la commande : terraform apply -auto-approve. Enfin, après l’exécution du script, nous avons exécuté une commande echo pour voir le bon déroulement de notre pipeline.

Le script de le pipeline a pour rôle principal de déclencher le script de Terraform qui va, à son tour, déclencher le script python.

### Conclusion :

Comme on a vu dans ce chapitre, nous avons examiné en détail les différentes facettes de notre approche visant à résoudre le problème du déploiement manuel et répétitif des templates NiFi. À travers une exploration approfondie des outils, des méthodes de simulation d'environnement de production et de l'architecture mise en place pour notre pipeline, nous avons éclairé les étapes clés de notre solution.

Nous avons identifié les contraintes inhérentes au déploiement manuel des templates NiFi, mettant en lumière la perte de temps et le risque d'erreur humaine qui en découlent. La valorisation des templates NiFi en tant qu'élément essentiel du flux de travail quotidien des Data Engineers a été soulignée, montrant comment ils permettent d'améliorer la productivité en automatisant les flux de données récurrents. De plus, en mettant en évidence le partage de templates dans un référentiel dédié, nous avons montré comment cette approche favorise la collaboration et la réutilisation au sein de l'équipe.

Nous avons ensuite abordé la simulation de l'environnement de production, une étape cruciale pour concevoir et tester notre solution. L'utilisation de Minikube et de l'opérateur Nifikop a permis de recréer fidèlement l'environnement GKE, offrant ainsi une plate-forme de développement réaliste pour itérer et perfectionner notre pipeline.

L'architecture détaillée de notre pipeline a été présentée, mettant en évidence la synergie entre les divers outils et technologies que nous avons utilisés. Le diagramme d'architecture a clairement illustré comment ces éléments s'associent pour garantir la mise en œuvre réussie du pipeline, automatisant le téléchargement et l'instanciation des templates NiFi sur les instances NiFi de notre cluster Minikube.

L'implémentation de notre solution a été abordée, avec des captures d'écran fournissant des preuves tangibles de la réalisation de notre objectif. Nous avons démontré comment notre pipeline permet non seulement le déploiement automatisé des templates NiFi, mais assure également la traçabilité des déploiements, offrant ainsi une visibilité essentielle pour le suivi et la gestion.

En somme, ce chapitre a mis en lumière notre démarche méthodique pour résoudre efficacement la problématique du déploiement des templates NiFi. Les outils sélectionnés, la simulation de l'environnement de production et l'architecture bien pensée de notre pipeline convergent pour offrir une solution pratique, efficiente et traçable, répondant aux besoins cruciaux des Data Engineers et contribuant ainsi à l'amélioration globale de la gestion des flux de données au sein de notre environnement opérationnel. Le prochain chapitre mettra en évidence les résultats obtenus grâce à cette solution novatrice et leur impact sur l'efficacité opérationnelle de notre équipe.

# **CONCLUSION**

# **BIBLIOGRAPHIE**

# **ANNEXES**