Conception et Architecture du 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.

GitLab

.gitlab-ci.yml

Machine Locale

Docker

Minikube

Nifikop

Zookeeper Cluster

Pod

NifiCluster

NifiPod

nifi\_templates\_deploy.py

Terraform

main.tf

Pod

Pod

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.

1. Préparation du script python *nifi\_templates\_deploy.py* qui 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 :
2. Récupérer du Token d’accès
3. Uploader les fichier xml du templates spécifiés
4. Instancer des ProcessGroup pour les templates uploadés
5. 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.

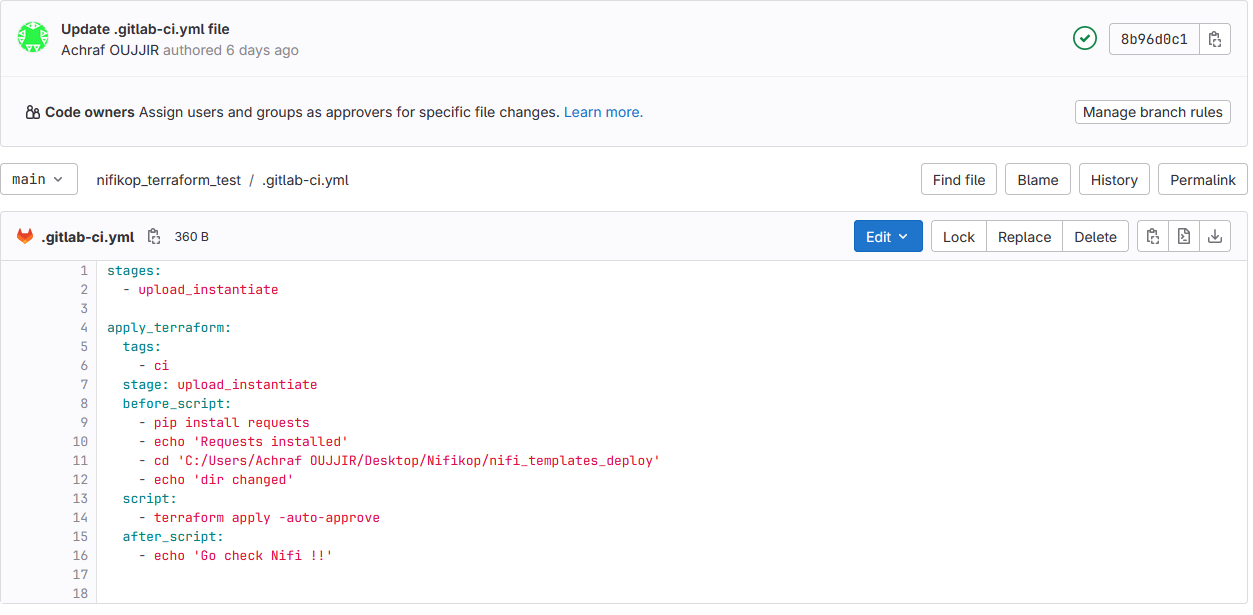
Pour la première partie, elle n’est possible que quand on travaille avec le protocole *https*. Pour la deuxième, on a utilisé l’endpoint correspondant pour faire le téléchargement des fichiers xml tout en retenant les ids de ces templates pour pouvoir les instancier dans l’étape suivante en tant que 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 répondre à un besoin potentiel qui est celui du suivi

1. 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. Dans ce fichier main.tf, on pourra également ajouter plus de ressources pour créer un autre NiFiCluster avec plusieurs pods et y déployer les templates désirés. Ce fichier pourra donc être modifié au fur et à mesure des besoins que l’équipe exprime.

1. 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 :



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.