Etudiante : Mariame SANGARE

Classe : M2 GL

MISE EN PLACE DE PIPELINE CICD AVEC HELM

Dans le cadre de l'automatisation du déploiement des applications modernes, la mise en place d'une **pipeline CI/CD** est essentielle pour assurer une livraison rapide et fiable. Ce projet vise à **déployer une application Spring Boot** en utilisant **Docker, Docker Hub, Kubernetes et Helm**, tout en intégrant un pipeline GitHub Actions pour automatiser les différentes étapes du cycle de vie applicatif.

I. Prérequis pour la Mise en Place de la Pipeline CI/CD

Avant de commencer, assurez-vous d'avoir les outils suivants installés et configurés :

- ✓ Java 17+ → Nécessaire pour exécuter l'application Spring Boot
- ✓ Maven → Pour la gestion des dépendances et la compilation du projet.
- ✓ Git → Pour gérer le code source et les versions.
- ✓ Docker → Pour construire et exécuter l'image de l'application.
- ✓ Docker Hub → Un compte pour stocker les images Docker.
- ✓ Kubernetes (Minikube ou un cluster réel) → Pour l'orchestration des conteneurs.
- ✓ Helm → Pour gérer le déploiement Kubernetes via des charts Helm.

II. Mise en place proprement dite

L'application doit être transformée en une image Docker pour être déployée dans Kubernetes. Pour cela, on commence par créer un fichier Dockerfile à la racine du projet.

- ➢ On crée un conteneur Docker léger qui exécute une application Spring Boot en utilisant **OpenJDK 17**, expose le **port 8081**, et exécute automatiquement le fichier stock-ms.jar. Cela permet un déploiement rapide et portable de l'application.

```
FROM openjdk:17-jdk-slim

LABEL maintainer="mariame mariame@groupeisi.com"

EXPOSE 8081

COPY target/stock-ms.jar stock-ms.jar

ENTRYPOINT ["java", "-jar", "stock-ms.jar"]
```

- Ensuite, on construit l'image Docker avec la commande suivante :

```
docker build -t stock-ms:1.0 .
```

- On va générer un chart Helm pour gérer le déploiement sur Kubernetes : **helm create spring-boot-app**

  Cela crée un dossier **spring-boot-app** contenant des fichiers de configuration. Il faut ensuite modifier le fichier values.yaml pour définir l'image Docker et le service :

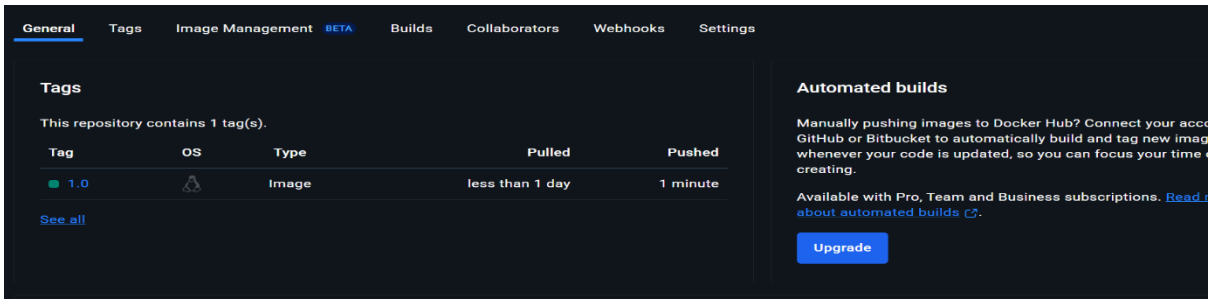
```yaml
namespace: devops

replicaCount: 3

image:
  repository: mariames/stock-ms
  tag: 1.0
  pullPolicy: IfNotPresent

service:
  type: NodePort
  port: 81
  targetPort: 8081
  nodePort: 30081

spring:
  profiles:
    active: prod
serviceAccount:
  create: false
  name: default
```

- Avant de déployer, on peut tester le chart en exécutant :

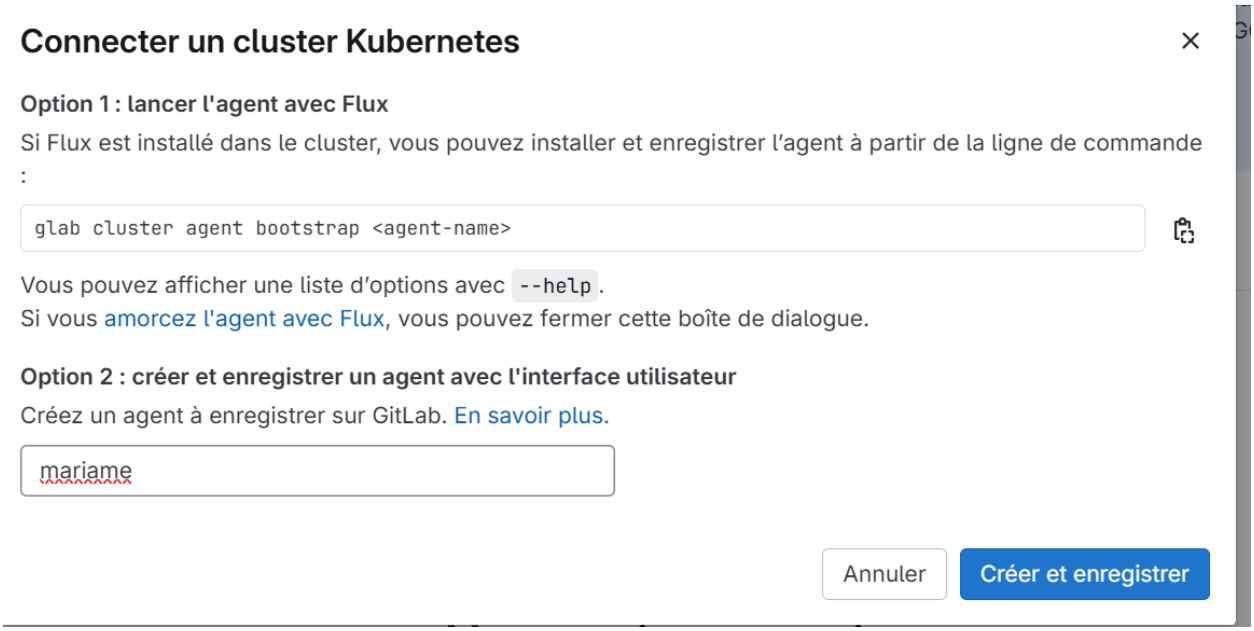
```
PS C:\Users\Mariame\Desktop\2024\M2GL\SEMESTRE 1\JEE\microservice-ms> helm install spring-boot-app ./spring-boot-app
NAME: spring-boot-app
LAST DEPLOYED: Wed Feb 26 11:01:13 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
PS C:\Users\Mariame\Desktop\2024\M2GL\SEMESTRE 1\JEE\microservice-ms>
```

- Maintenant il faut qu'on taggue le projet et pusher vers docker hub bien évidemment on s'est d'abord connecté à notre docker hub.

```
PS C:\Users\Mariame\Desktop\2024\M2GL\SEMESTRE 1\JEE\microservice-ms> docker tag stock-ms:1.0 mariames/stock-ms:1.0
PS C:\Users\Mariame\Desktop\2024\M2GL\SEMESTRE 1\JEE\microservice-ms> docker push mariames/stock-ms:1.0
The push refers to repository [docker.io/mariames/stock-ms]
```



➢ On se connecte à notre dépôt gitlab pour ajouter un agent de connexion à notre cluster



➢ Ensuite il nous génère des credentials pour la connexion.On copie les commandes

mariame créé avec succès.

Jeton d'accès de l'agent

glagent-B6FndVYyVXjDoTTA1Ahszs8WTyBQuX9-1GMo9a7C2NTy59o4Jw

L'agent utilise le jeton pour se connecter à GitLab.

Vous ne pourrez plus revoir ce jeton après avoir fermé cette fenêtre.

Effectuer l'installation avec Helm (recommandé)

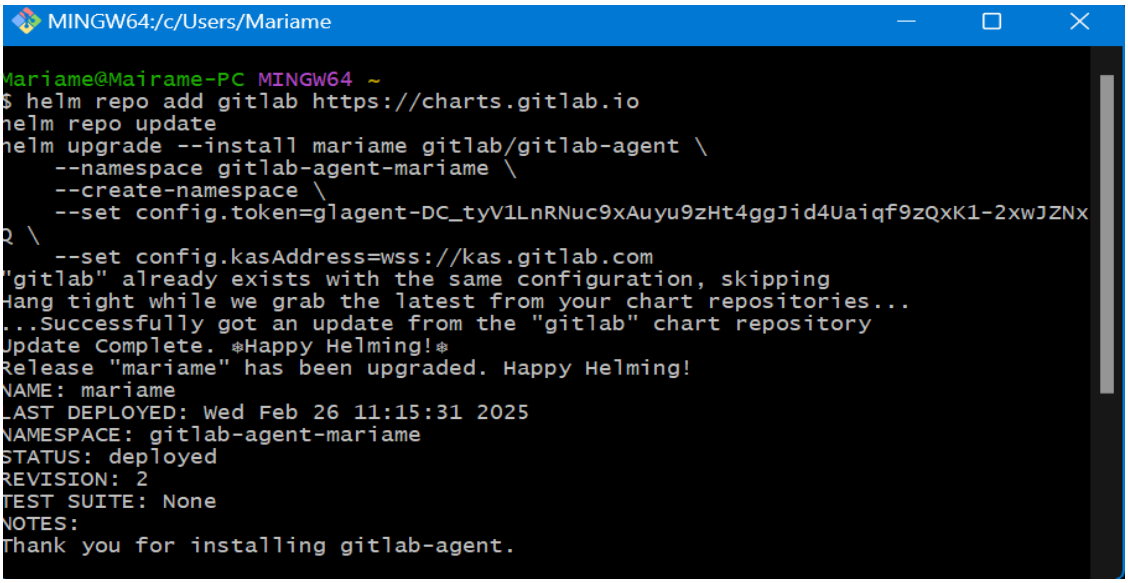
Depuis un terminal, connectez-vous à votre cluster, puis exécutez cette commande. Cette dernière contient le jeton. Utilisez une version de Helm compatible avec votre version de Kubernetes (consultez la stratégie de prise en charge des versions de Helm).

```
helm repo add gitlab https://charts.gitlab.io
helm repo update
helm upgrade --install mariame gitlab/gitlab-agent \
    --namespace gitlab-agent-mariame \
    --create-namespace \
    --set config.token=glagent-B6FndVYyVXjDoTTA1Ahszs8WTyBQuX9-1GMo9a7C2NTy59o4Jw \
    --set config.kasAddress=wss://kas.gitlab.com
```

Méthodes d'installation avancées

Voir la documentation de l'installation avancée. Assurez-vous d'être en possession de votre jeton d'accès.

- Ouvrir le terminal et y coller les commandes



- On check pour vérifier le statut de notre connexion avec le cluster kubernetes

Agents du projet

Nom	État de la connexion	Dernier contact	Version	Agent ID	Configuration
mariame	Connectés	à l'instant	17.9.0	2127494	Configuration par défaut

Lien gitlab : https://gitlab.com/mariames/microservice-ms.git

Repo: mariame