

Support de cours sur la mise en place d'un pipeline CI/CD d'une application informatique

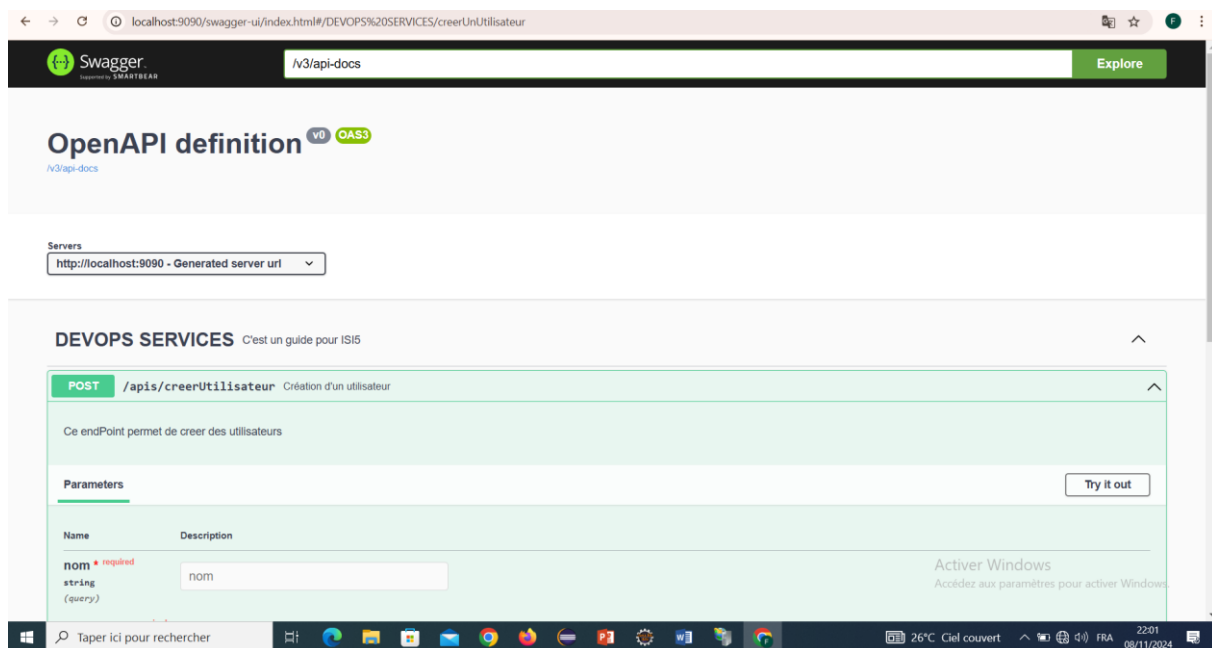
Outils de CI/CD : Gitlab-CI

Outils de versionning : Gitlab

Contexte : L'application dont il faut automatiser le CI/CD est un microservice développé sur java-springboot.

Nom de l'application : Devops

Au démarrage le microservice expose un service web pour créer un utilisateur tel que l'illustre l'image ci-dessous. C'est le tp que nous avons fait en classe. Avant l'enregistrement d'un utilisateur, le système s'assure que les données saisies sont valides avant tout ajout dans la bd. En cas de données invalides, l'opération d'ajout est annulée. **(NB :Le projet est joint au tutoriel).**

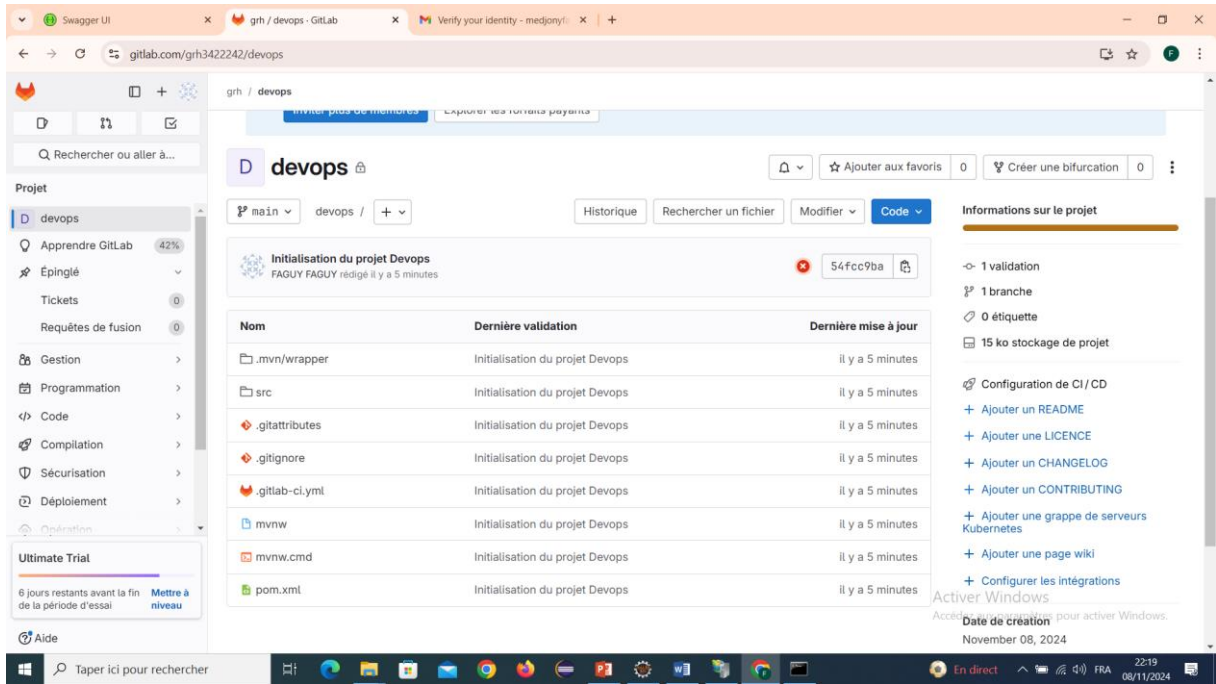


Spécifications des jobs à automatiser dans le pipeline l'application :

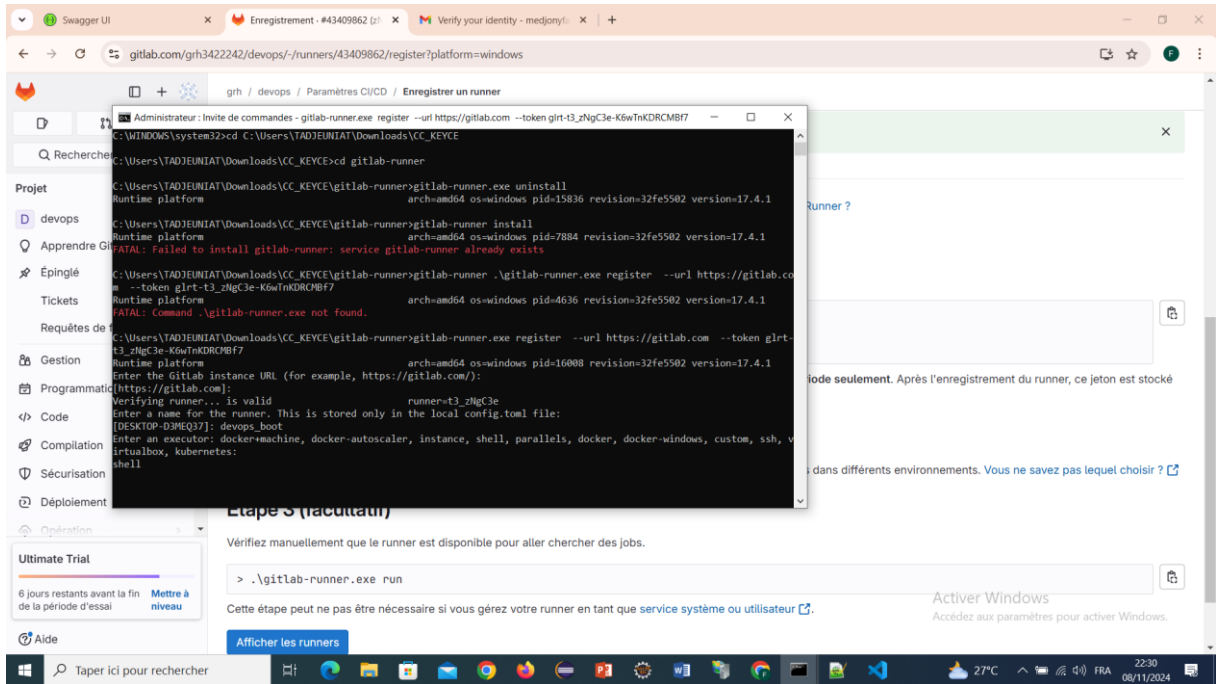
- Intégration continue des codes dans la branche principale
- Tests unitaire des composants applicatifs
- Compilation du code pour la génération de l'exécutable maven install.
- Déploiement de l'application sur docker (création d'une image docker pour l'application et creation/exécution du conteneur de l'application.

Démarche à suivre.

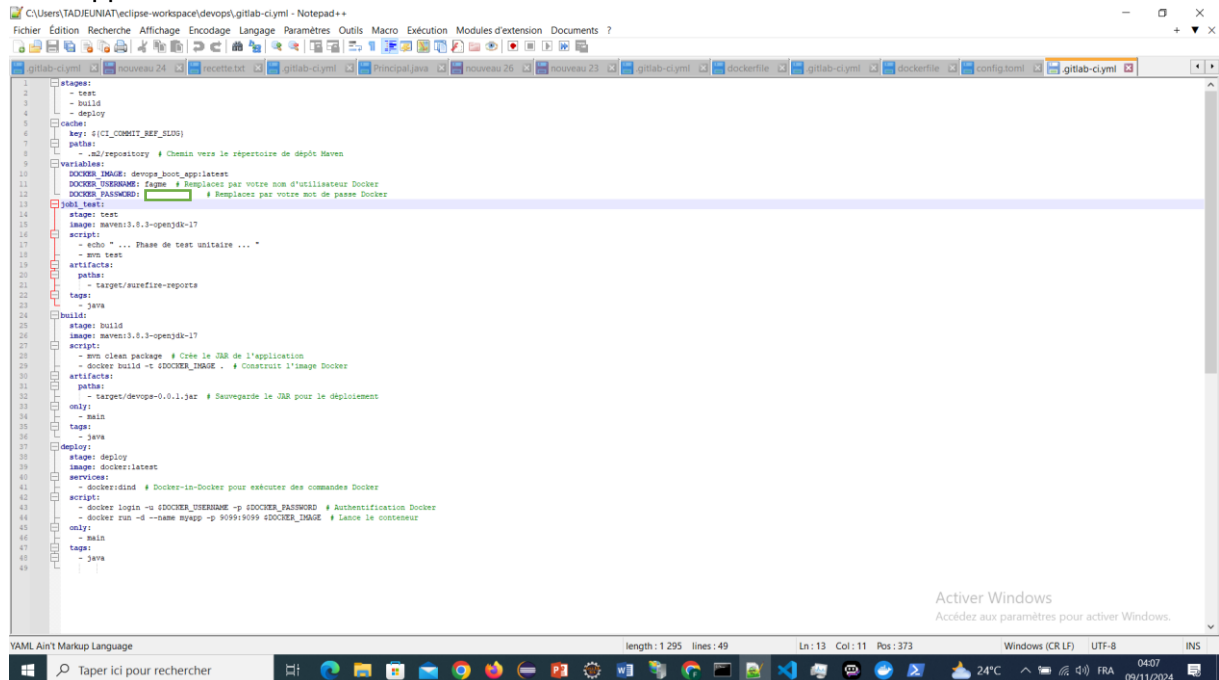
Créer un dépôt distant de l'application dans une instance gitlab et faire un push des codes sources.



Etapes de configuration de mes runners telles que démontrées au cours. La figure ci-dessous illustre l'état en exécution de mon runner). Le runner attend la présence d'un job pour l'exécuter.



Etape de configuration de mon pipeline CI/CD pour la compilation, tests et déploiement de mon application dans docker.



```
1 stages:
2   - test
3   - build
4   - deploy
5
6 cache:
7   key: ${CI_COMMIT_REF_SLUG}
8   paths:
9     - .m2/repository # Chemin vers le répertoire de dépôt Maven
10
11 variables:
12   DOCKER_IMAGE: devops_bootstrap
13   DOCKER_USERNAME: faguy # Remplacez par votre nom d'utilisateur Docker
14   DOCKER_PASSWORD: # Remplacez par votre mot de passe Docker
15
16 job: test
17   stage: test
18   image: maven:3.8.3-openjdk-17
19   script:
20     - echo " ... Phase de test unitaire ... "
21     - mvn test
22   artifacts:
23     paths:
24       - target/surefire-reports
25   tags:
26     - java
27
28 job: build
29   stage: build
30   image: maven:3.8.3-openjdk-17
31   script:
32     - mvn clean package # Crée le JAR de l'application
33     - docker build -t $DOCKER_IMAGE . # Construit l'image Docker
34   artifacts:
35     paths:
36       - target/devops-0.0.1.jar # Sauvegarde le JAR pour le déploiement
37   only:
38     - main
39   tags:
40     - java
41
42 job: deploy
43   stage: deploy
44   image: docker:latest
45   services:
46     - docker:dind # Docker-in-Docker pour exécuter des commandes Docker
47   script:
48     - docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD # Authentification Docker
49     - docker run -d --name myapp -p 8080:8080 $DOCKER_IMAGE # Lance le conteneur
50   only:
51     - main
52   tags:
53     - java
```

1- La section **stages** définit l'ordre des étapes dans le pipeline. Les étapes sont exécutées dans l'ordre où elles sont listées.

test : Exécute les tests unitaires.

build : Compile le code et crée un paquet (JAR ou autre).

deploy : Déploie l'application.

2- Configuration du cache

Cela permet de conserver les dépendances Maven entre les exécutions du pipeline.

- **key** : Utilise la variable `${CI_COMMIT_REF_SLUG}` pour créer un cache unique par branche, évitant les conflits entre les dépendances.
- **paths** : Spécifie que le répertoire `.m2/repository`, qui contient les dépendances Maven, doit être mis en cache.

3- **Déclaration de variables** : Ces variables peuvent être utilisées dans les scripts des jobs.

- **DOCKER_IMAGE** : Définit le nom de l'image Docker à construire.
- **DOCKER_USERNAME et DOCKER_PASSWORD** : Fournissent les informations d'identification pour se connecter à Docker Hub (ou un autre registre Docker).

4- Job `job1_test`

- **stage** : Indique que ce job appartient à l'étape de test.
- **image** : Utilise l'image Docker `maven:3.8.3-openjdk-17`, qui contient Maven et OpenJDK 17.
- **Script** : Les commandes à exécuter :
 - Affiche un message indiquant le début des tests.
 - Exécute `mvn test` pour lancer les tests unitaires.
- **artifacts** : Spécifie que les rapports de test générés dans `target/surefire-reports` doivent être conservés pour être accessibles après l'exécution du job.
- **tags** : Permet de spécifier que ce job doit être exécuté sur un runner ayant la balise `java`.

5- Job `build`

Job de construction :

stage : Appartient à l'étape de construction.

image : Utilise la même image que pour le job de test.

script :

mvn clean package : Compile le code et crée le fichier JAR.

docker build -t \$DOCKER_IMAGE : Construit l'image Docker à partir du Dockerfile présent dans le répertoire courant.

artifacts : Sauvegarde le fichier JAR généré pour le déploiement.

only : Indique que ce job ne s'exécute que sur la branche main.

tags : Doit être exécuté sur un runner avec la balise java.

6- Job deploy

Il s'agit du Job de déploiement :

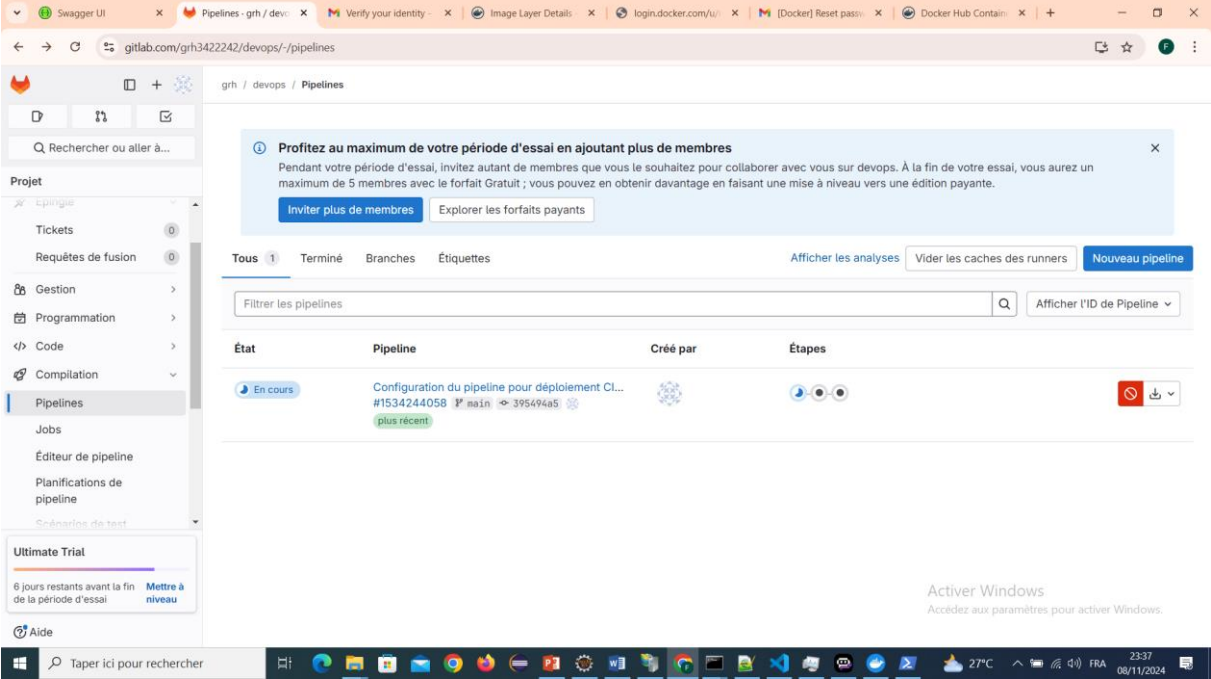
- stage : Appartient à l'étape de déploiement.
- image : Utilise docker:latest, qui contient les outils nécessaires pour exécuter des commandes Docker.
- services : Utilise docker:dind (Docker-in-Docker) pour permettre l'exécution de commandes Docker dans le job.
- script :
 - docker login : Authentifie l'utilisateur Docker avec le nom d'utilisateur et le mot de passe fournis.
 - docker run : Lance le conteneur à partir de l'image Docker construite précédemment, mappant le port 9099 du conteneur au port 9099 de l'hôte.
- only : Ce job ne s'exécute que sur la branche main.
- tags : Doit être exécuté sur un runner avec la balise java.

Exécution du pipeline.

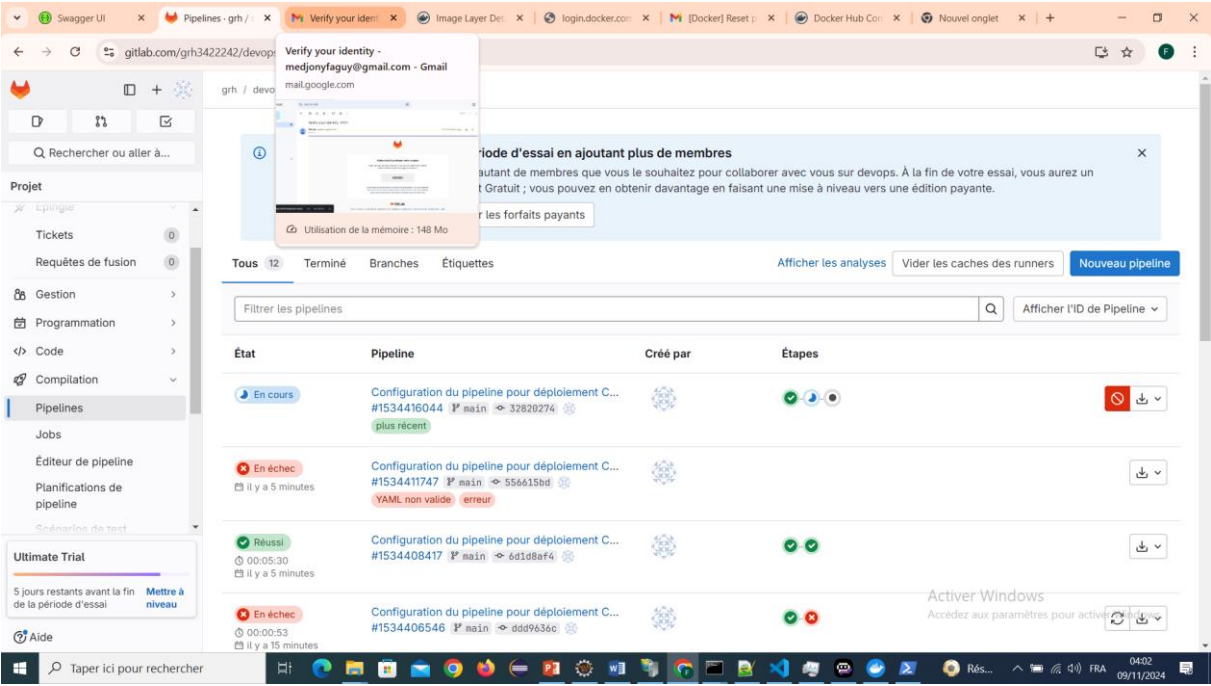
Après configuration de mon pipeline, je push le code dans le repository distant. Gitlab détectera une mise à jour qui déclenchera l'exécution de mon pipeline tel que l'illustre l'image ci-dessous

The screenshot shows the GitLab web interface for a pipeline configuration. The browser tabs include Swagger UI, Pipeline - grh / devops, Verify your identity, Image Layer Details, login.docker.com/u/, [Docker] Reset pass, Docker Hub Contain, and others. The address bar shows the URL: gitlab.com/grh342242/devops/-/pipelines/1534244058. The left sidebar contains navigation links: Projets, Tickets, Requêtes de fusion, Gestion, Programmation, Code, Compilation, Pipelines, Jobs, Éditeur de pipeline, Planifications de pipeline, and Scénarios de test. The main content area displays the pipeline configuration for 'grh / devops / Pipelines / #1534244058'. A blue banner at the top encourages adding more members to the trial. Below this, the configuration is titled 'Configuration du pipeline pour déploiement CICD oui' with buttons 'Annuler le pipeline' and 'Supprimer'. The status shows 'En cours' (In progress) for the pipeline, created by 'FAGUY FAGUY' for the 'main' branch. It indicates '3 jobs' and 'En cours, mis en file d'attente pendant 1 secondes'. The pipeline diagram shows three jobs: 'test' (with sub-job 'job1_test'), 'build', and 'deploy'. The 'test' job is currently running. At the bottom, there is an 'Ultimate Trial' banner and an 'Active Windows' notification.

L'image ci-dessous nous montre mon pipeline est en cours exécution de 3 jobs



L'image ci-dessous est l'illustration d'un job (de test) qui a réussi. La 2nd étape est en cours d'exécution.



L'image ci-dessous est la détail la console de visualisation de l'exécution des scripts du premier job (de test) : Les tests unitaires ont marché.

The screenshot displays the GitLab CI/CD interface for a job named 'test' (ID: #8314455381). The console output shows the execution of a Spring Boot application with unit tests. The tests passed successfully, with a total duration of 10.178 seconds. The output includes the following details:

- Tests run: 12, Failures: 0, Errors: 0, Skipped: 0
- Time elapsed: 0.183 s
- Build Success
- Total time: 10.178 s
- Finished at: 2024-11-09T04:02:22+01:00
- Git revision: 32fe5582
- Git branch: 17-4-stable
- GO version: go1.22.5
- Built: 2024-10-11T15:55:13+0000

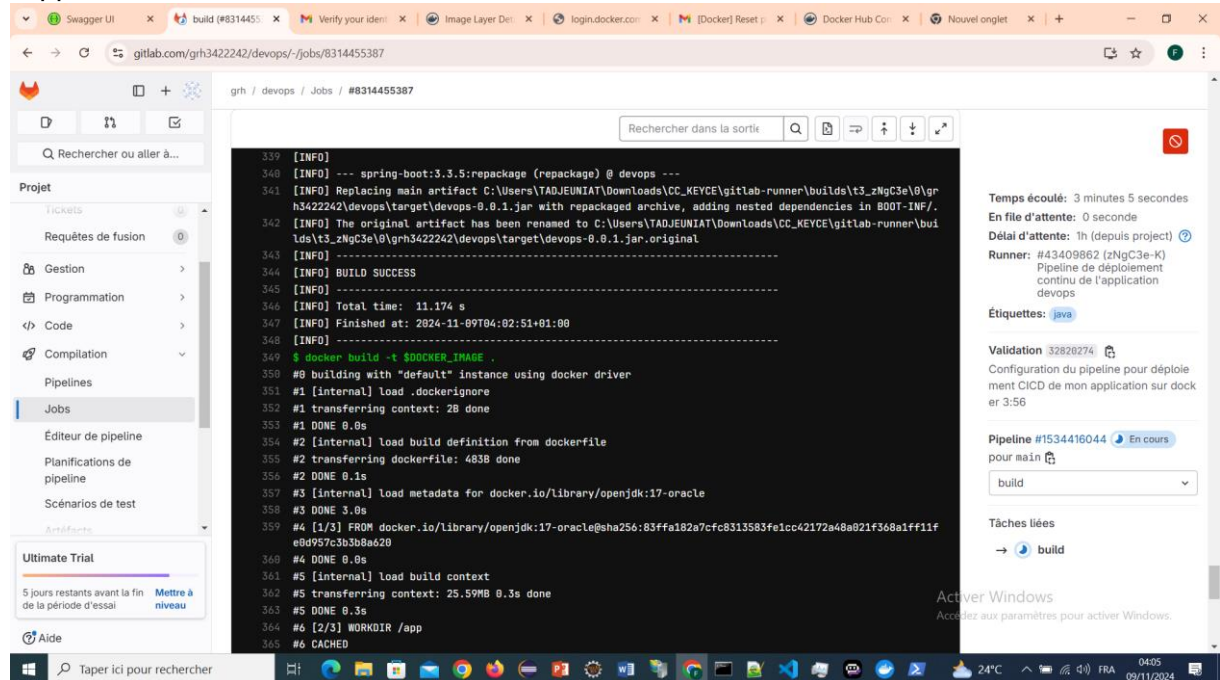
On the right side, the job status is 'Terminé' (Completed) with a duration of 26 seconds. The pipeline configuration for deployment is also visible.

The screenshot displays the GitLab CI/CD interface for a job named 'test' (ID: #8314455381). The console output shows the execution of a Spring Boot application with unit tests. The tests failed due to a runtime exception. The output includes the following details:

- Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
- Time elapsed: 0.178 s
- Build Success
- Total time: 10.178 s
- Finished at: 2024-11-09T04:02:22+01:00
- Git revision: 32fe5582
- Git branch: 17-4-stable
- GO version: go1.22.5
- Built: 2024-10-11T15:55:13+0000

On the right side, the job status is 'Terminé' (Completed) with a duration of 26 seconds. The pipeline configuration for deployment is also visible.

L'image ci-dessous est la console d'exécution de job de build. Elle nous montre l'état d'avancement détaillé des activités du 2ème job ; Nous pouvons voir que le fichier exécutable devop.0.0.1.jar a été généré. Gitlab-ci procède à la construction de l'image docker de l'application. C'est une étape peut chronophage relativement à la taille de l'application.



Rechercher dans la sortie

```

339 [INFO] --- spring-boot:3.3.5:repackage (repackage) @ devops ---
340 [INFO] Replacing main artifact C:\Users\TADJEUNIAI\Downloads\CC_KEYCE\gitlab-runner\builds\t3_zNgC3e\0\gr
h3422242\devops\target\devops-0.0.1.jar with repackaged archive, adding nested dependencies in BOOT-INF/.
342 [INFO] The original artifact has been renamed to C:\Users\TADJEUNIAI\Downloads\CC_KEYCE\gitlab-runner\buil
ds\t3_zNgC3e\0\grh3422242\devops\target\devops-0.0.1.jar.original
343 [INFO] -----
344 [INFO] BUILD SUCCESS
345 [INFO] -----
346 [INFO] Total time: 11.174 s
347 [INFO] Finished at: 2024-11-09T04:02:51+01:00
348 [INFO] -----
349 $ docker build -t $DOCKER_IMAGE .
350 #0 building with "default" instance using docker driver
351 #1 [internal] load .dockerignore
352 #1 transferring context: 2B done
353 #1 DONE 0.0s
354 #2 [internal] load build definition from dockerfile
355 #2 transferring dockerfile: 483B done
356 #2 DONE 0.1s
357 #3 [internal] load metadata for docker.io/library/openjdk:17-oracle
358 #3 DONE 3.0s
359 #4 [1/3] FROM docker.io/library/openjdk:17-oracle@sha256:83ffa182a7cfc8313583f1cc42172a48a021f368a1ff11f
e0d9957c3b3b9a620
360 #4 DONE 0.0s
361 #5 [internal] load build context
362 #5 transferring context: 25.59MB 0.3s done
363 #5 DONE 0.3s
364 #6 [2/3] WORKDIR /app
365 #6 CACHED

```

Temps écoulé: 3 minutes 5 secondes
En file d'attente: 0 seconde
Délai d'attente: 1h (depuis project)
Runner: #43409862 (zNgC3e-K)
Pipeline de déploiement continu de l'application devops

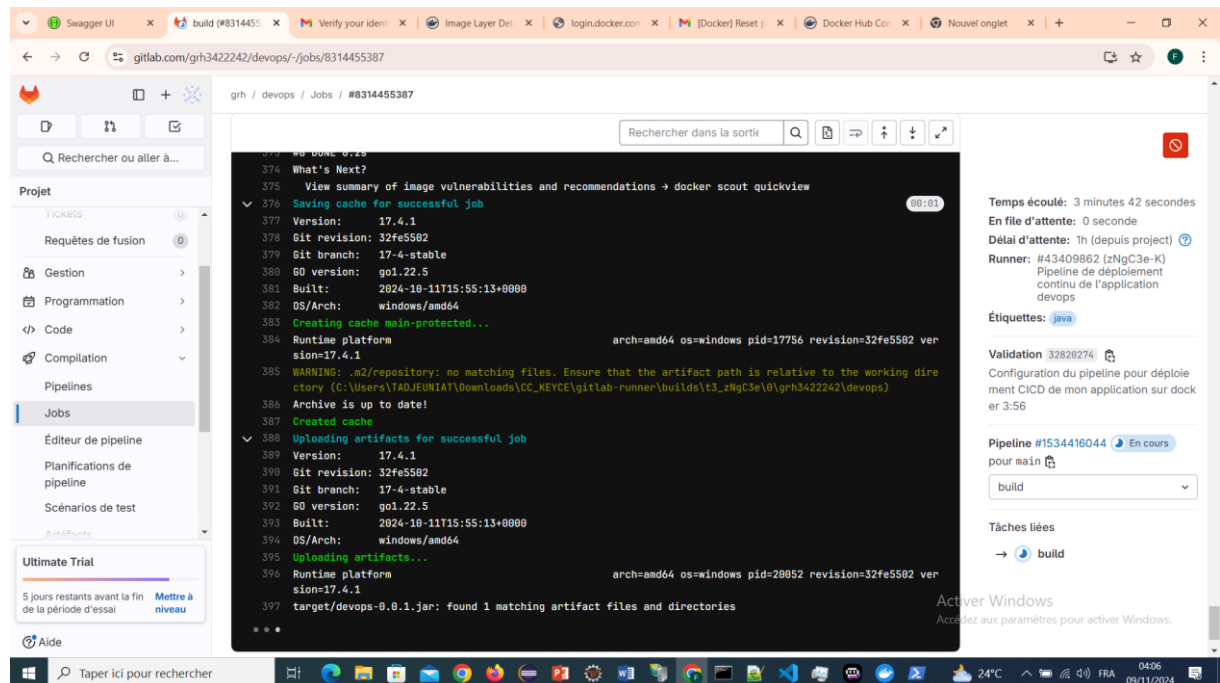
Étiquettes: java

Validation 32820274
Configuration du pipeline pour déploie ment CI/CD de mon application sur dock er 3:56

Pipeline #1534416044 En cours
pour main

Tâches liées
→ build

Activer Windows
Accédez aux paramètres pour activer Windows.



Rechercher dans la sortie

```

374 We done 0.2s
375 What's Next?
376 View summary of image vulnerabilities and recommendations → docker scout quickview
377 Saving cache for successful job
378 Version: 17.4.1
379 Git revision: 32fe5502
380 Git branch: 17-4-stable
381 GO version: go1.22.5
382 Built: 2024-10-11T15:55:13+0000
383 OS/Arch: windows/amd64
384 Creating cache main-protected...
385 Runtime platform arch=amd64 os=windows pid=17756 revision=32fe5502 ver
sion=17.4.1
386 WARNING: .m2/repository: no matching files. Ensure that the artifact path is relative to the working dire
ctory (C:\Users\TADJEUNIAI\Downloads\CC_KEYCE\gitlab-runner\builds\t3_zNgC3e\0\grh3422242\devops)
387 Archive is up to date!
388 Created cache
389 Uploading artifacts for successful job
390 Version: 17.4.1
391 Git revision: 32fe5502
392 Git branch: 17-4-stable
393 GO version: go1.22.5
394 Built: 2024-10-11T15:55:13+0000
395 OS/Arch: windows/amd64
396 Uploading artifacts...
397 Runtime platform arch=amd64 os=windows pid=20052 revision=32fe5502 ver
sion=17.4.1
398 target/devops-0.0.1.jar: found 1 matching artifact files and directories

```

Temps écoulé: 3 minutes 42 secondes
En file d'attente: 0 seconde
Délai d'attente: 1h (depuis project)
Runner: #43409862 (zNgC3e-K)
Pipeline de déploiement continu de l'application devops

Étiquettes: java

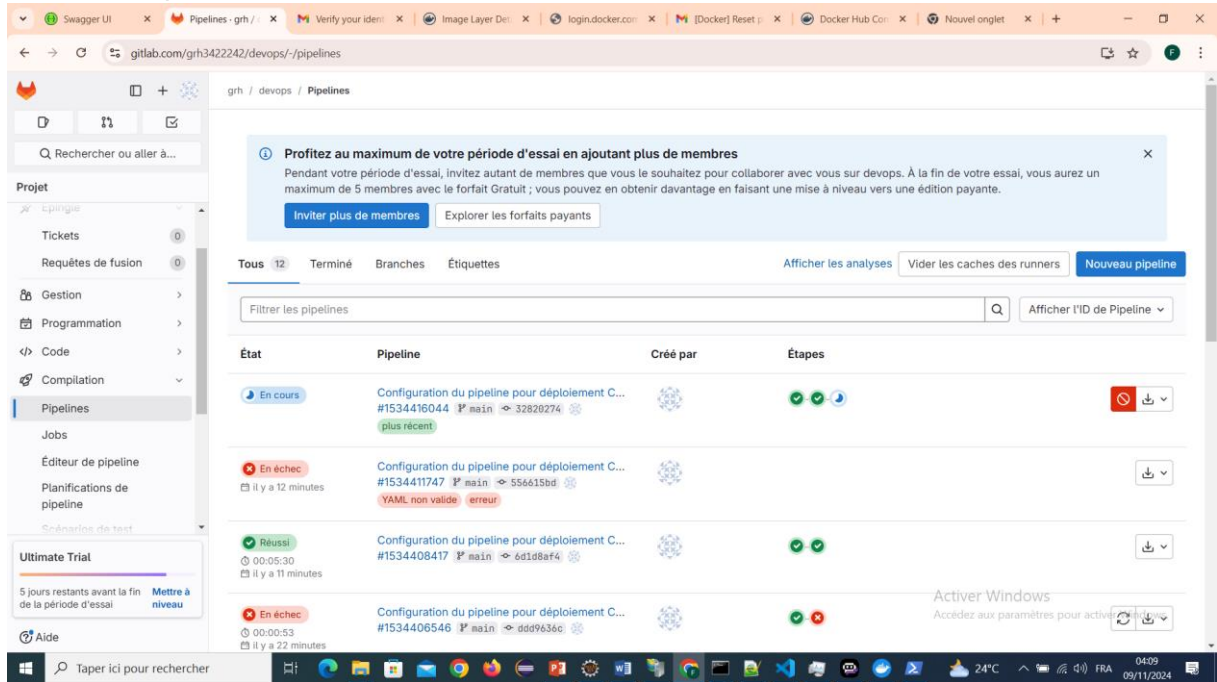
Validation 32820274
Configuration du pipeline pour déploie ment CI/CD de mon application sur dock er 3:56

Pipeline #1534416044 En cours
pour main

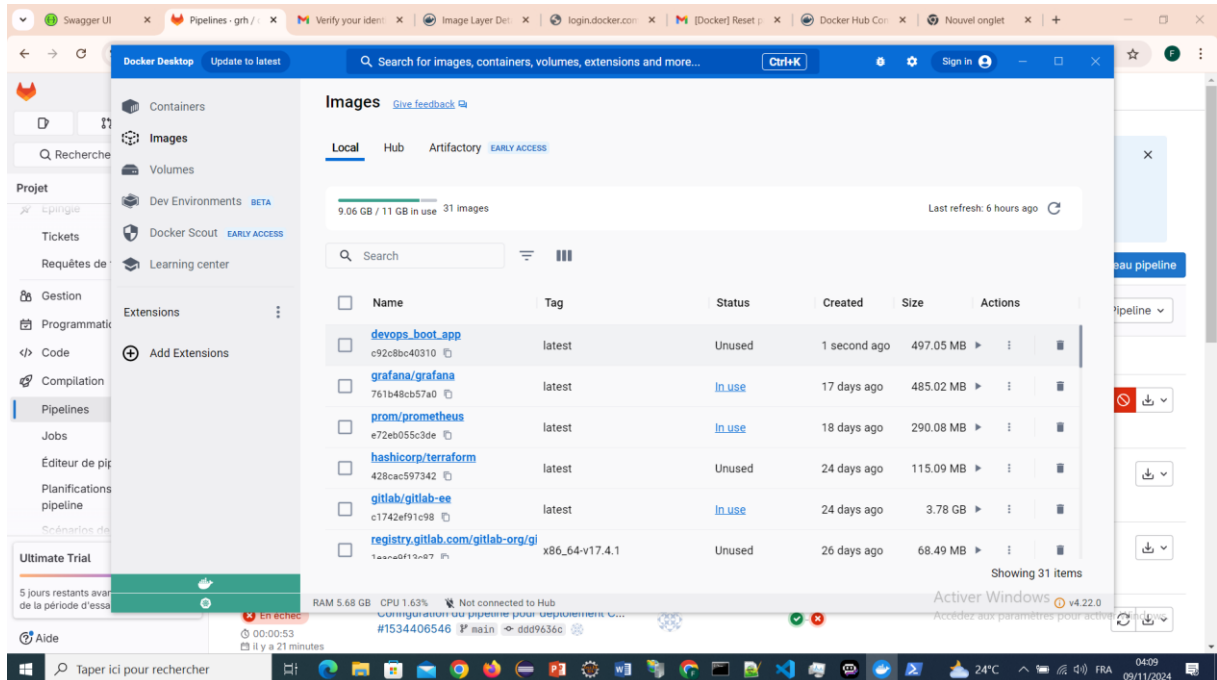
Tâches liées
→ build

Activer Windows
Accédez aux paramètres pour activer Windows.

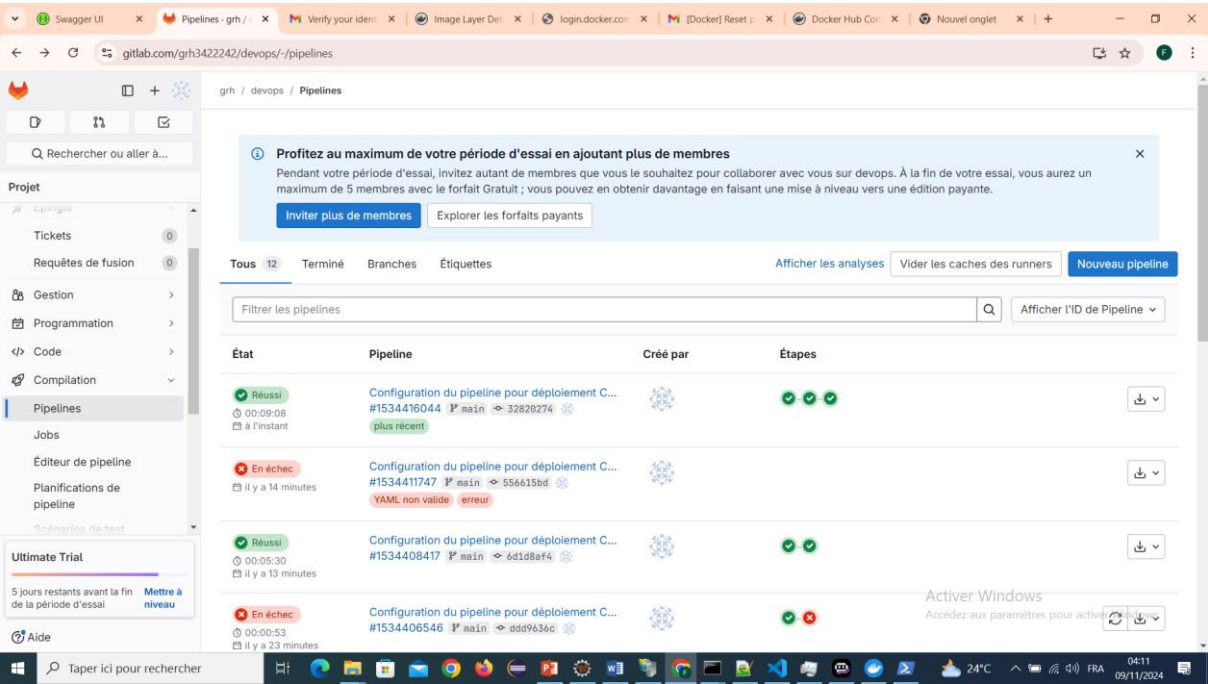
L'image ci-dessous nous montre que les deux étapes sont abouties. La troisième et dernière étape est en cours..



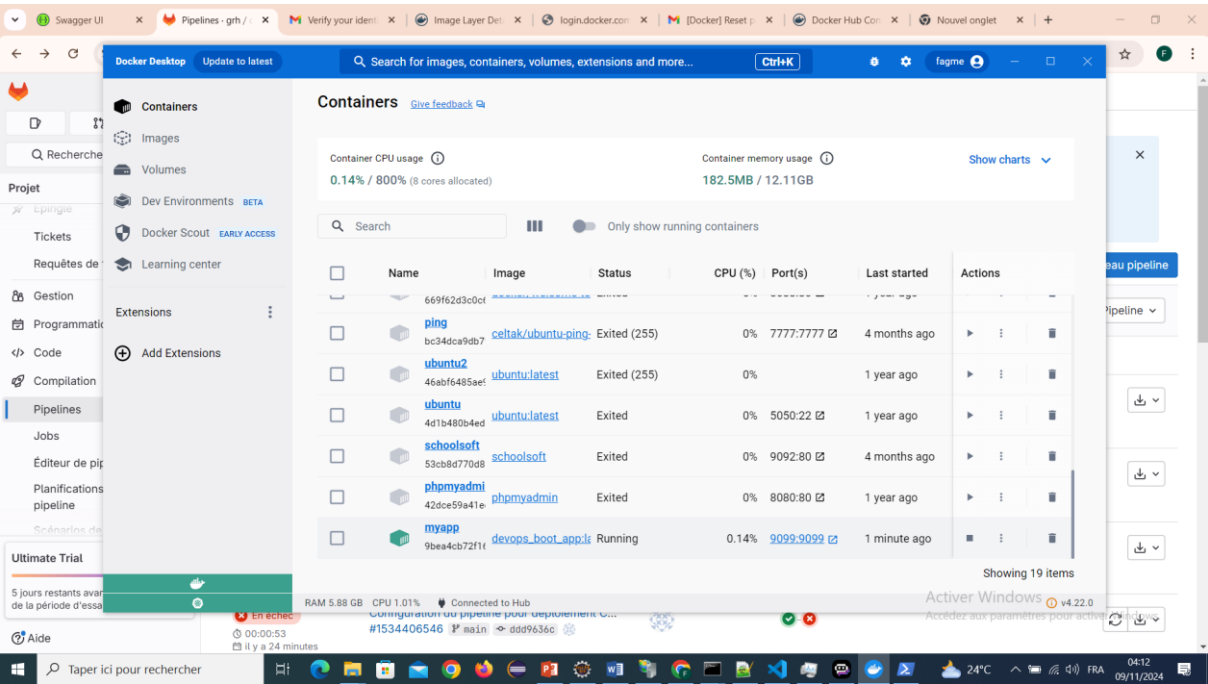
L'image docker est bien construite. Nous pouvons voir l'image dans la rubrique images de mon logiciel docker (image ci-dessous)



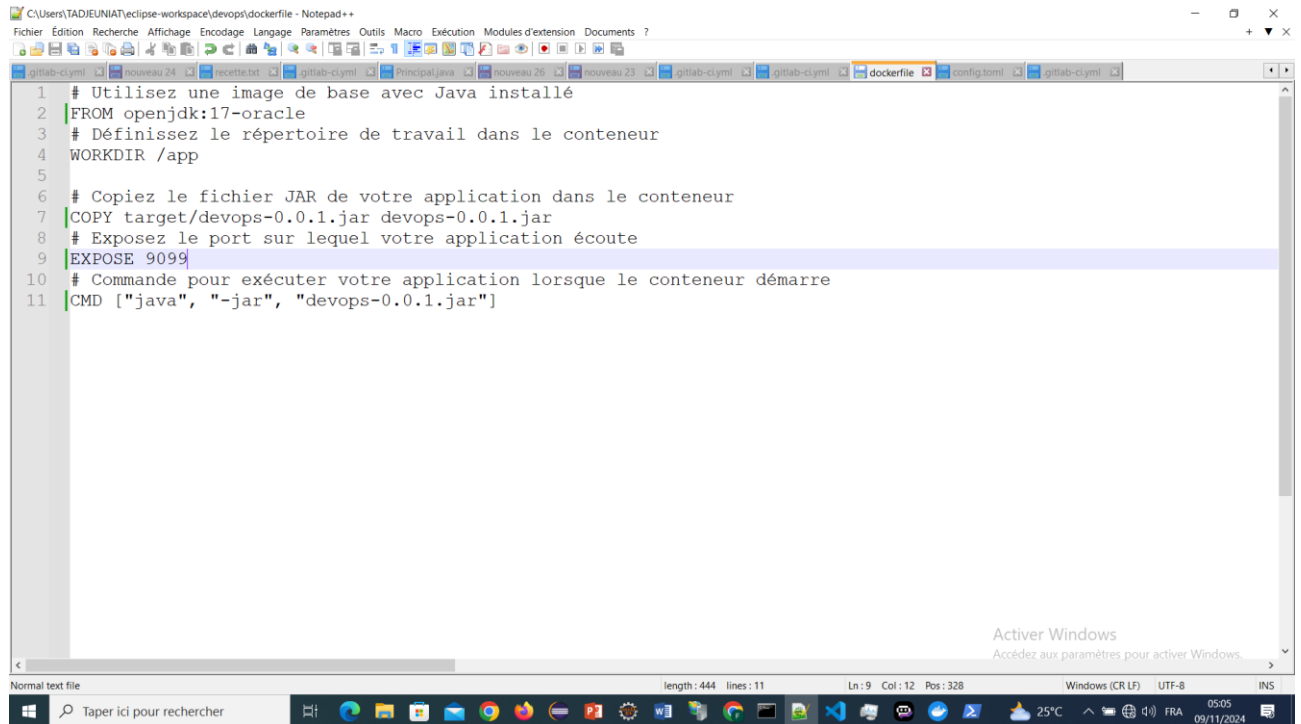
Etat réussi de notre pipeline (figure ci-dessous)



Notre conteneur applicative représentant notre instance de l'application en cours d'exécution.



Mon dockerfile posé à la racine du projet



```
1 # Utilisez une image de base avec Java installé
2 FROM openjdk:17-oracle
3 # Définissez le répertoire de travail dans le conteneur
4 WORKDIR /app
5
6 # Copiez le fichier JAR de votre application dans le conteneur
7 COPY target/devops-0.0.1.jar devops-0.0.1.jar
8 # Exposez le port sur lequel votre application écoute
9 EXPOSE 9099
10 # Commande pour exécuter votre application lorsque le conteneur démarre
11 CMD ["java", "-jar", "devops-0.0.1.jar"]
```

Merci