**Enoncé Chapitre 3**

1. Configurer un workflow CI GitHub Actions pour compiler le code à chaque modification majeure
2. Faites une modification sur votre code et commiter pour vérifier le lancement automatique du workflow
3. Dockériser les deux microservices et configurer la pipeline Jenkins de façon à builder les images et les déployer sur le docker registry
4. Faites monter l’application avec Docker-compose en local

**Solution**

1. Aller sur Github Actions et créer un workflow de type « Java with Maven »

Commiter le code sous .github/workflows, le workflow se lance automatiquement

Si vous rencontrez l’erreur suivante :

Error in phase Update dependency graph:

HTTP Status 403 for request POST <https://api.github.com/repos/dependency-graph/snapshots> github actions

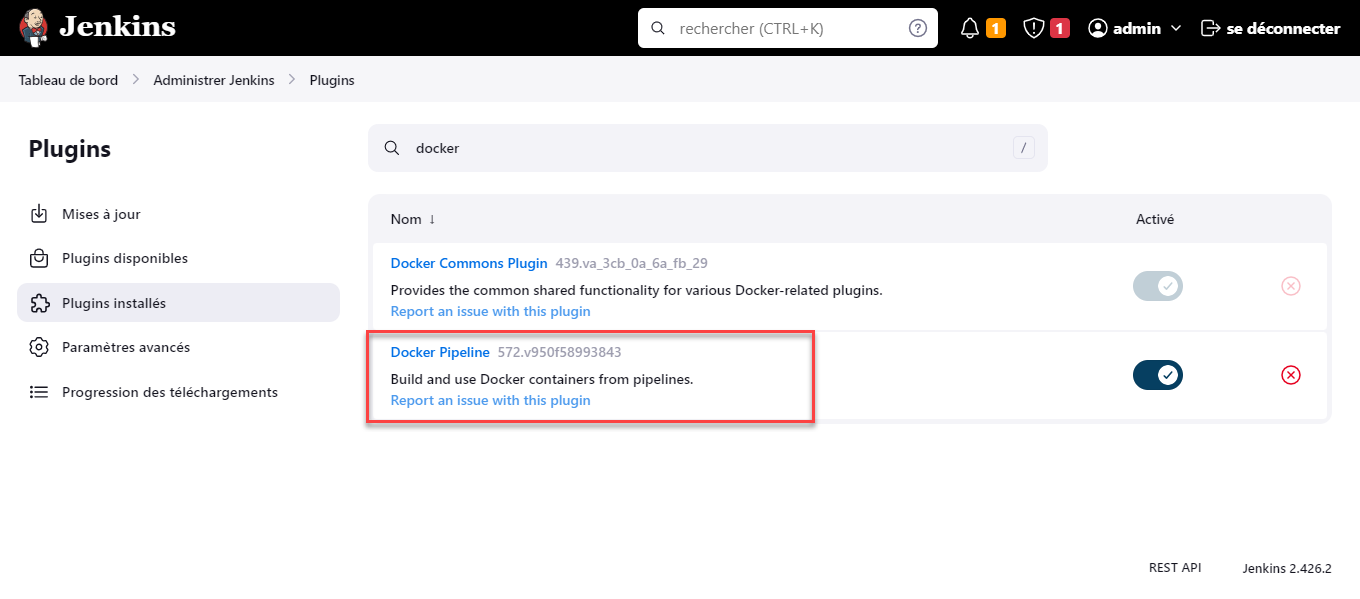
Solution:

Go to settings --> Actions --> General --> Workflow permissions

Machine generated alternative text:
aymendr / 
springbootsample 
O Code @ Issues 
Pull requests 
I General 
Access 
AN Collaborators 
Moderation options 
Code and automation 
Branches 
Tags 
Rules 
@ Actions 
General 
Runners 
(B Webhooks 
Actions Projects 
General 
Repository name 
springbootsample 
Template repository 
Wiki 
Q 
C) Security 
Rename 
Type to search 
Insights Settings 
Template repositories let users generate new repositories With the same directory structure and files. Learn more about template repositories. 
Require contributors to sign off on web-based commits 
Enabling this setting will require contributors to sign off on commits made through GitHub's web interface. Signing off is a way for 
contributors to affirm that their commit complies With the repositoryls terms, commonly the Developer Certificate of Origin (DCO). Learn 
more about signing off on commits. 
Default branch 
The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are 
automatically made, unless you specify a different branch. 

Machine generated alternative text:
Secrets and variables 
Integrations 
GitHub Apps 
Email notifications 
O 
O 
Require approval for first-time contributors Who are new to GitHub 
Only first-time contributors Who recently created a GitHub account will require approval to run workflows. 
Require approval for first-time contributors 
Only first-time contributors will require approval to run workflows. 
Require approval for all outside collaborators 
Save 
Workflow permissions 
Choose the default permissions granted to the GITHUB TOKEN when running workflows in this repository. You can specify 
more granular permissions in the workflow using YAML. Learn more about managing permissions. 
Read and Write permissions 
Workflows have read and Write permissions in the repository for all scopes. 
O 
Read repository contents and packages permissions 
Workflows have read permissions in the repository for the contents and packages scopes only. 
Choose whether GitHub Actions can create pull requests or submit approving pull request reviews. 
Allow GitHub Actions to create and approve pull requests 
Save 

1. Le workflow se lance normalement automatiquement suite à un push vers la branch main
2. Installer le plugin Jenkins Docker Pipeline depuis le menu d’administration Jenkins :



Configurer la pipeline dans le projet jenkins :

pipeline {

agent any

stages {

stage("Checkout Code") {

steps {

git branch: 'main', url: 'https://github.com/aymendr/formation\_devOps\_prise\_en\_mains'

}

}

stage("Build") {

steps {

bat 'mvn clean install'

}

}

stage("SonarQube Analyse") {

steps {

withSonarQubeEnv('sonarqube') {

bat 'mvn sonar:sonar'

}

}

}

stage("Build Docker Images") {

steps {

bat 'docker build -t aymendr/notification notification/.'

bat 'docker build -t aymendr/product-service product-service/.'

}

}

stage("Push Docker Images to Hub") {

steps {

withDockerRegistry(credentialsId: 'loginpwddocker') {

bat 'docker push aymendr/notification'

bat 'docker push aymendr/product-service'

}

}

} }}

1. Voici le fichier **compose.yaml** à placer sous la racine du projet

services:  
 notification:  
 build: notification  
 restart: always  
 ports:  
 - 8082:8082  
 networks:  
 - spring-mysql  
 depends\_on:  
 - product-service  
 product-service:  
 build: product-service  
 restart: always  
 ports:  
 - 8081:8081  
 networks:  
 - spring-mysql  
 environment:  
 - spring.datasource.url=jdbc:mysql://db:3306/product\_db?allowPublicKeyRetrieval=true  
 depends\_on:  
 - db  
 db:  
 *# We use a mariadb image which supports both amd64 & arm64 architecture* image: mysql:latest  
 *# If you really want to use MySQL, uncomment the following line  
 #image: mysql:8.0.27* restart: always  
 environment:  
 - MYSQL\_DATABASE=product\_db  
 - MYSQL\_USER=springuser  
 - MYSQL\_PASSWORD=springuser  
 - MYSQL\_ROOT\_PASSWORD=springuser  
 ports:  
 - 3306:3306  
 networks:  
 - spring-mysql  
networks:  
 spring-mysql:

Exécuter la commande docker compose up