

LAB 6 : Empaquetage d'une application dans un conteneur docker avec un Pipeline jenkins

Prérequis :

Attribuer l'autorisation nécessaire au jenkins pour lancer des commandes docker.

- Ajoutez l'utilisateur « jenkins » au groupe « docker »

```
sudo usermod -aG docker jenkins
```

- Redémarrez le service jenkins pour démarrer avec les nouvelles permissions

```
sudo systemctl restart jenkins.service
```

Etape 1 : Chargement du projet depuis le SCM

```
stage('Checkout') {  
    steps {  
        git branch: 'main',  
            url: 'https://github.com/medsalahmeddeb/AppWebJava'  
    }  
}
```

Etape 2 : Construction de l'application à travers de l'outil maven

```
stage('Build'){  
    tools{  
        maven 'MAVEN3'  
    }  
    steps{  
        sh 'mvn clean install'  
    }  
}
```

Etape 3 : création de l'image à partir du Dockerfile

Préparation de Dockerfile pour empaqueter l'application générée durant l'étape 2 suite à la construction de l'application par maven.

```
# Dockerfile  
FROM tomcat:8.0.20-jre8  
COPY target/AppWebJava*.war /usr/local/tomcat/webapps/AppWebJava.war
```

Ajout du stage pour la création de l'image

```

stage('Build Docker Image') {
    steps {
        sh "docker build -t meddeb/appwebjava:1.0.0
          /var/lib/jenkins/workspace/${JOB_NAME}"
    }
}

```

Remarque :

JOB_NAME : c'est une variable d'environnement créée avec jenkins contenant le nom du job encours d'exécution

Etape 4 : publication de l'image sur le docker hub

Afin de publier l'image créée dans le référentiel « docker hub », il faut créer un credential jenkins contenant le mot de passe du repository du projet

Administrer Jenkins > Manage Credentials > Jenkins > Identifiants globaux > Ajouter des identifiants

```

stage('Upload To DockerHub') {
    environment {
        PASS = credentials('pass-dockerhub')
    }
    steps {
        echo "${PASS}"
        sh "docker login -u 'meddeb' -p \"${PASS}\""
        sh "docker push meddeb/appwebjava:1.0.0"
    }
}

```

Etape 5 : création du conteneur

```

stage('Deploy app container') {
    steps {
        sh "docker rm -f appwebjava"
        sh "docker run -d -p 80:8080 --name appwebjava
meddeb/appwebjava:1.0.0"
    }
}

```

Script final :

```
pipeline {
  agent any

  stages {
    stage('Checkout') {
      steps {
        git branch: 'main',
            url: 'https://github.com/medsalahmeddeb/AppWebJava'
      }
    }
    stage('Build'){
      tools{
        maven 'MAVEN3'
      }
      steps{
        sh 'mvn clean install'
      }
    }

    stage('Build Docker Image') {
      steps {
        sh "docker build -t meddeb/appwebjava:1.0.0
/var/lib/jenkins/workspace/${JOB_NAME}"
      }
    }

    stage('Upload To DockerHub') {
      environment {
        PASS = credentials('pass-dockerhub')
      }
      steps {
        echo "${PASS}"
        sh "docker login -u 'meddeb' -p \"${PASS}\""
        sh "docker push meddeb/appwebjava:1.0.0"
      }
    }

    stage('Deploy app container') {
      steps {
        sh "docker rm -f appwebjava"
        sh "docker run -d -p 80:8080 --name appwebjava
meddeb/appwebjava:1.0.0"
      }
    }
  }
}
```

Modification du projet pour utiliser maven en tant que conteneur

NB : il faut installer le plugin « **Docker Pipeline** » pour utiliser docker en tant agent dans Jenkins Pipeline

Exemple de Pipeline avec image docker "**maven:3.5.3-jdk-10-slim**"

```
pipeline {
  agent {
    docker {
      image 'maven:3.5.3-jdk-10-slim'
    }
  }
  stages {
    stage('Build') {
      steps {
        sh 'mvn -v'
      }
    }
  }
}
```

```

pipeline {
    agent none

    stages {
        stage('Checkout') {
            agent any
            steps {
                git branch: 'main', url:
'https://github.com/medsalahmeddeb/AppWebJava'
            }
        }
        stage('Build'){
            agent {
                docker {
                    image 'maven:3.5.3-jdk-10-slim'
                }
            }
            steps{
                sh 'mvn clean install'
            }
        }

        stage('Build Docker Image') {
            agent any
            steps {
                sh "docker build -t meddeb/appwebjava:1.0.0
/var/lib/jenkins/workspace/${JOB_NAME}"
            }
        }

        stage('Upload To DockerHub') {
            agent any
            environment {
                PASS = credentials('pass-dockerhub')
            }
            steps {
                echo "${PASS}"
                sh "docker login -u 'meddeb' -p \"${PASS}\""
                sh "docker push meddeb/appwebjava:1.0.0"
            }
        }

        stage('Deploy app container') {
            agent any
            steps {
                sh "docker rm -f appwebjava"
                sh "docker run -d -p 80:8080 --name appwebjava
meddeb/appwebjava:1.0.0"
            }
        }
    }
}

```