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

Etape 1 : Chargement du projet depuis le SCM

```
stage('Chargemet depuis Github'){

        git branch: 'main', changelog: false, poll: false,

        url: 'https://github.com/medsalahmeddeb/AppWebJava'

    }
```

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

```
    stage('Construction  du projet par MAVEN'){

        def mvnHome = tool name: 'MAVEN3', type: 'maven'

        sh "${mvnHome}/bin/mvn clean package"

    }
```

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
```

Durant l'étape 3 et afin de crée l'image empaquetant l'application AppWebJava.war, il faut exécuter des commandes docker et celui-ci exige d'avoir permission les permissions admin.

Création d'un tunnel SSH entre le compte jenkins te le compte root :

- Par défaut le compte jenkins il n'a pas la permission d'exécuter des commande sur le serveur (shell : /bibn/false), vous pouvez vérifier le type de shell dans le fichier /etc/passwd

```
[root@localhost ~]# grep jenkins /etc/passwd
jenkins:x:987:981:Jenkins Automation Server:/var/lib/jenkins:/bin/false
```

⇨ Il faut modifier le shell pour pouvoir exécuter des commandes :

```
[root@localhost ~]# chsh -s /bin/bash jenkins
```

- Création des paramètres d'accès SSH :
    - Lancez une session SSH pour l'utilisateur jenkins

```
[root@localhost ~]# su - jenkins
Dernière connexion : jeudi 28 janvier 2021 à 09:46:03 CET sur pts/0
-bash-4.2$
```

    - Générez les clés

```
-bash-4.2$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa):
Created directory '/var/lib/jenkins/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_rsa.
Your public key has been saved in /var/lib/jenkins/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:T0nCtvekX8TfZIIhAQxoRV8gj1zXFSjArDbXgt92J2s
jenkins@localhost.localdomain
The key's randomart image is:
+---[RSA 2048]----+
|     +===++. oo. |
|    o. B+o..o    |
|           .E.   |
|           ..    |
+----[SHA256]-----+
```

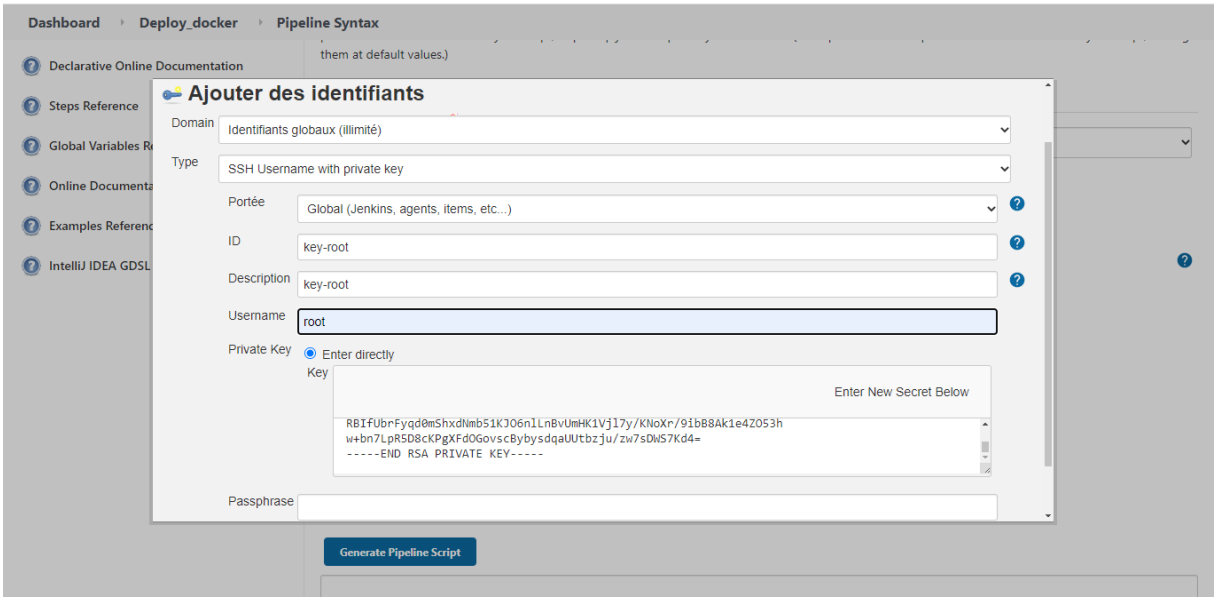
    - Copiez la clé publique de l'utilisateur jenkins dans le compte root où docker est installé. Si docker est installé dans une machine distante, changez « localhost » par « L'IP de la machine distante ».

```
-bash-4.2$ ssh-copy-id root@localhost
/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/var/lib/jenkins/.ssh/id_rsa.pub"
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is
SHA256:8HB6ZSbBDRixAUN1hcDN75VPQizlGZTyY/xSaWVwQYo.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
root@localhost's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@localhost'"
and check to make sure that only the key(s) you wanted were added.
```

- Dans l'outil SSH Agent renseigner les paramètres d'authentification préparés. Vous pouvez utiliser l'outil Pipeline Syntaxe pour préparer le code englobant les commandes ssh. La clé privée est disponible dans le fichier /var/lib/jenkins/.ssh/id_rsa




```
    stage('Build Docker Image'){

        sshagent(['docker']) {

            sh "ssh root@192.168.208.131 docker build -t
meddeb/appwebjava:1.0.0 /var/lib/jenkins/workspace/Deploy_docker"

        }

    }
```

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

```
    stage('Upload To DockerHub'){

        sshagent(['docker']) {

            sh "ssh root@192.168.208.131 docker login -u 'meddeb' -p
'password'"

            sh "ssh root@192.168.208.131 docker push
meddeb/appwebjava:1.0.0"

        }

    }
```

Etape 5 : création du conteneur

```
    stage("Deploiement d'un conteur sur la machine"){

        sshagent(['docker']) {

             sh "ssh root@192.168.208.131 docker rm -f appwebjava "

             sh "ssh root@192.168.208.131 docker run -d -p 80:8080 --name appwebjava meddeb/appwebjava:1.0.0"

        }

    }
```

Script final :

```
node{
    stage('Chargemet depuis Github'){
        git branch: 'main', changelog: false, poll: false,
        url: 'https://github.com/medsalahmeddeb/AppWebJava'
    }

    stage('Construction  du projet par MAVEN'){
        def mvnHome = tool name: 'MAVEN3', type: 'maven'
        sh "${mvnHome}/bin/mvn clean package"
    }

    stage('Build Docker Image'){
        sshagent(['docker']) {
            sh "ssh root@192.168.208.131 docker build -t
meddeb/appwebjava:1.0.0 /var/lib/jenkins/workspace/Deploy_docker"
        }
    }

    stage('Upload To DockerHub'){
        sshagent(['docker']) {
            sh "ssh root@192.168.208.131 docker login -u 'meddeb' -p
'password'"
            sh "ssh root@192.168.208.131 docker push
meddeb/appwebjava:1.0.0"
        }
    }

    stage("Deploiement d'un conteur sur la machine"){
        sshagent(['docker']) {
             sh "ssh root@192.168.208.131 docker rm -f appwebjava "
             sh "ssh root@192.168.208.131 docker run -d -p 80:8080
                         --name appwebjava meddeb/appwebjava:1.0.0"

        }
    }
}
```