ATELIER : DEVOPS

Mohamed HAMMOUDA

Jenkins

A Successful Path To Continuous Integration And Continuous Delivery

PLAN DE L'ATELIER

1	INTRODUCTION AU DEVOPS
2	LE CONTRÔLE DES VERSIONS : GIT & GITLAB
3	LE CONTRÔLE DE QUALITÉ DES LOGICIELS
4	LES CONTENEURS APPLICATIVES : DOCKER
5	INTÉGRATION CONTINUE ET DÉPLOIEMENT CONTINU

PLAN DE L'ATELIER

1	INTRODUCTION AU DEVOPS
2	LE CONTRÔLE DES VERSIONS : GIT & GITLAB
3	LE CONTRÔLE DE QUALITÉ DES LOGICIELS
4	LES CONTENEURS APPLICATIVES : DOCKER
5	INTÉGRATION CONTINUE ET DÉPLOIEMENT CONTINU

JENKINS

JENKINS INSTALLATION

Windows :
https://www.blazemeter.com/blog/how-to-install-jenkins-on-windows

Ubuntu 20 :
https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-20-04-fr

OpenJDK 11 :
https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-on-ubuntu-20-04-fr

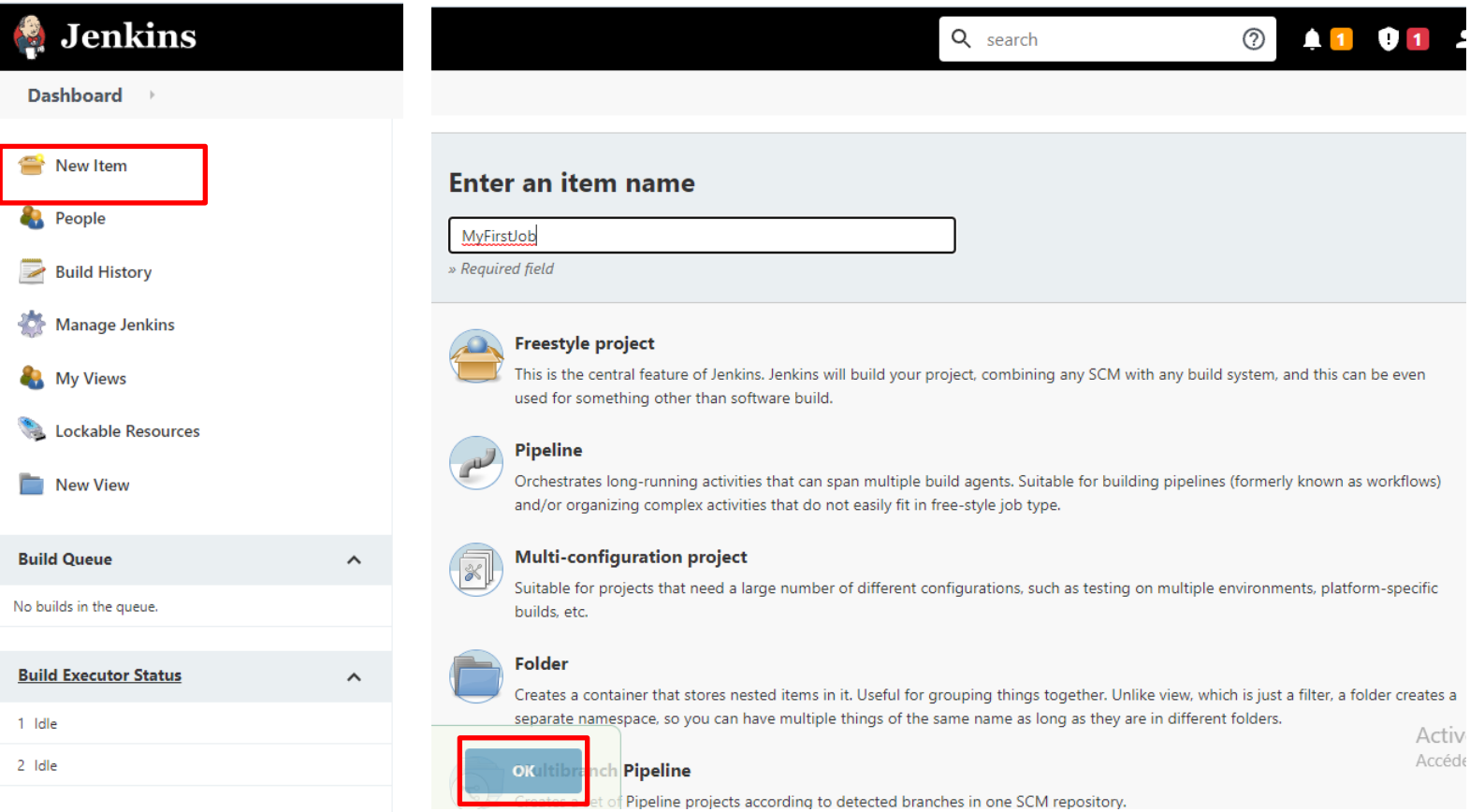
JENKINS

JENKINS INSTALLATION

- Jenkins est un outil open source de serveur d'automatisation. Il aide à automatiser les parties du développement logiciel, et facilite l'intégration continue et la livraison continue.
  - L'**intégration continue** est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée.
  - La **livraison continue** est une approche d'ingénierie logicielle dans laquelle les équipes produisent des logiciels dans des cycles courts, ce qui permet de le mettre à disposition à n'importe quel moment. Le but est de construire, tester et diffuser un logiciel plus rapidement.
- Écrit en Java, Jenkins fonctionne dans un **conteneur** de servlets tel qu'Apache Tomcat, ou en mode autonome avec son propre serveur Web embarqué.
- Il s'interface avec des systèmes de gestion de versions, des système de build automatisé, des outils de test, des depôts d'image, aussi bien que des scripts en shell Unix ou batch Windows.

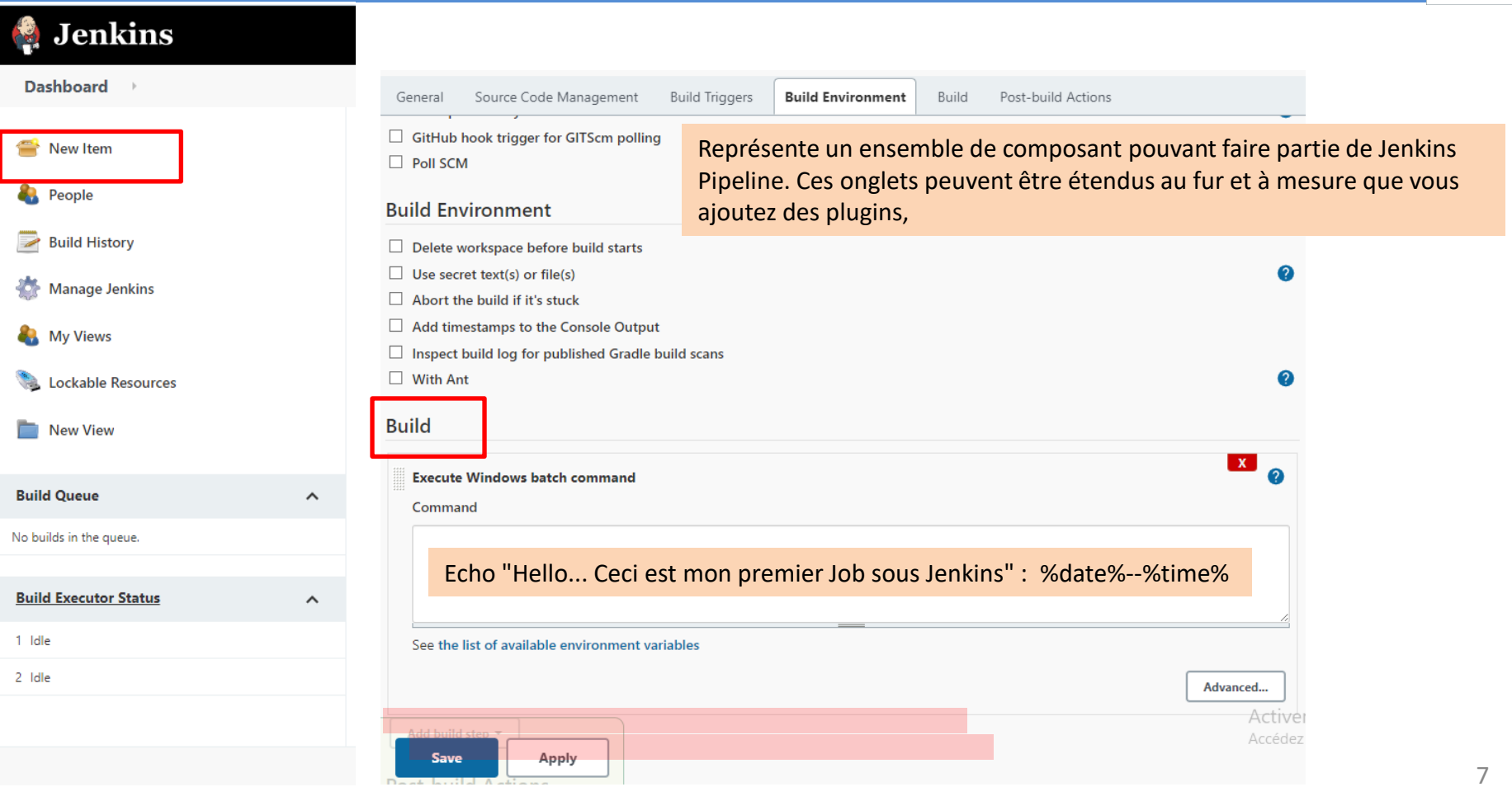
JENKINS

FIRST JOB : New Item: Freestyle project



JENKINS

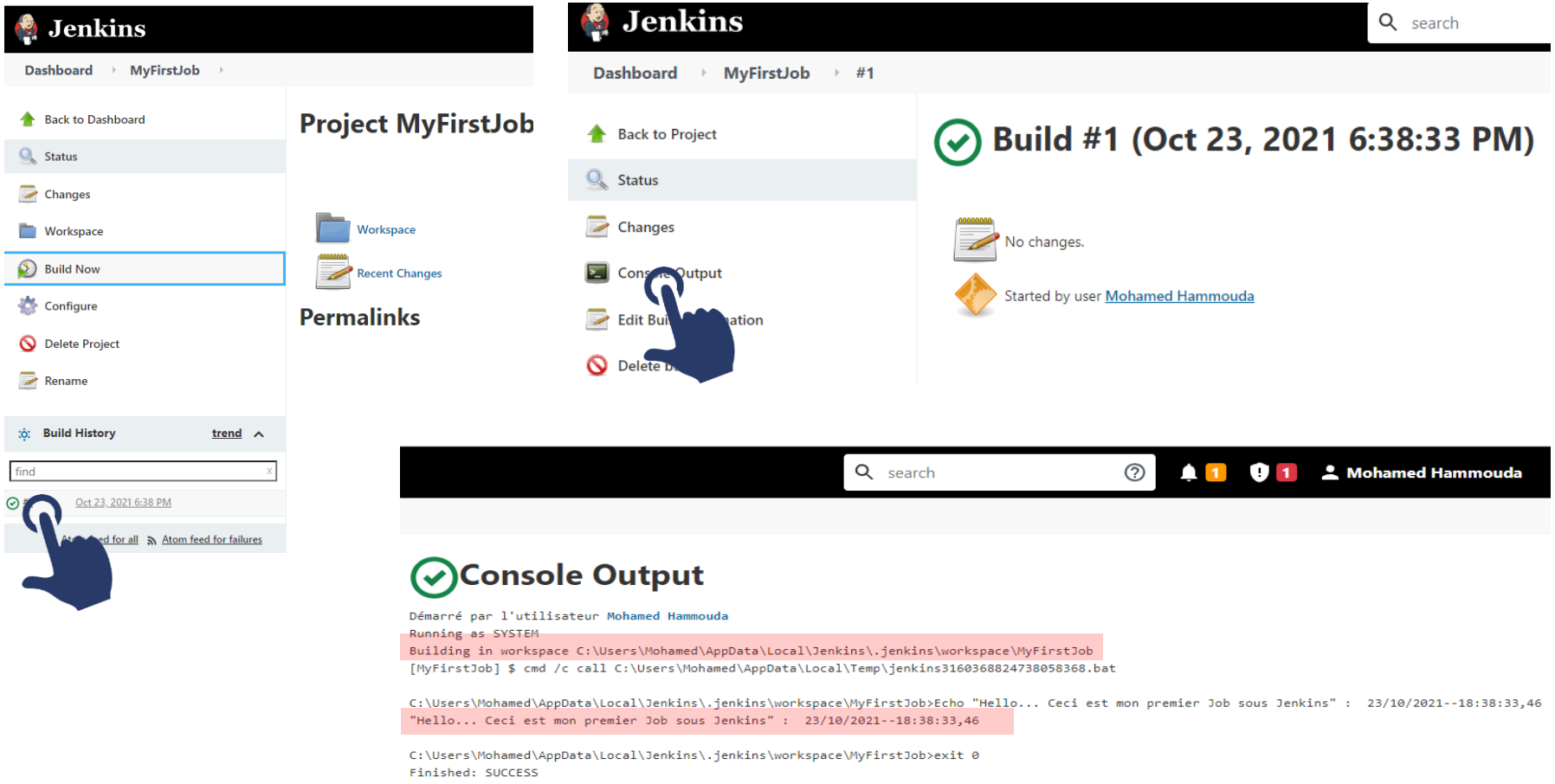
FIRST JOB : New Item: Freestyle project



Représente un ensemble de composant pouvant faire partie de Jenkins Pipeline. Ces onglets peuvent être étendus au fur et à mesure que vous ajoutez des plugins,

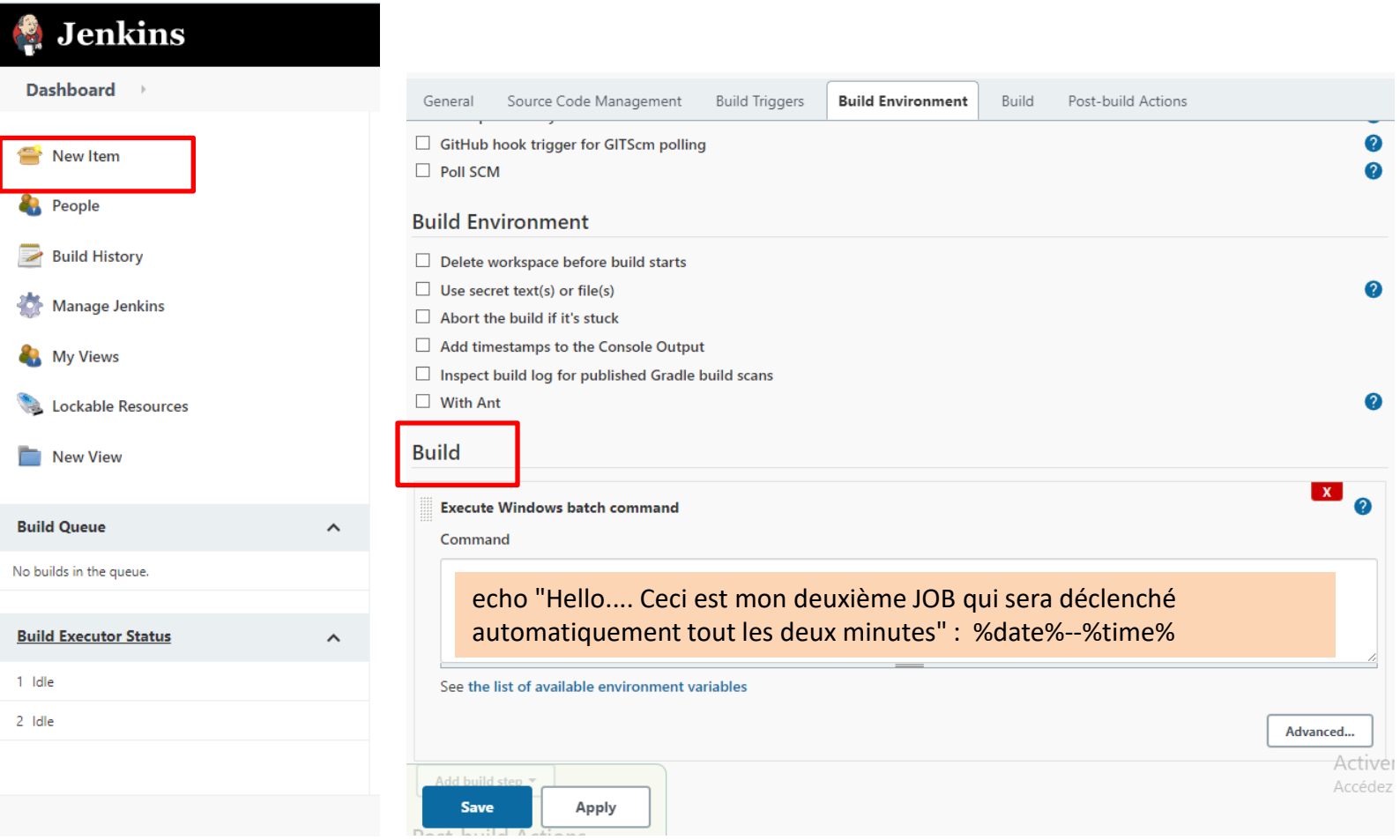
JENKINS

FIRST JOB : Build Now :



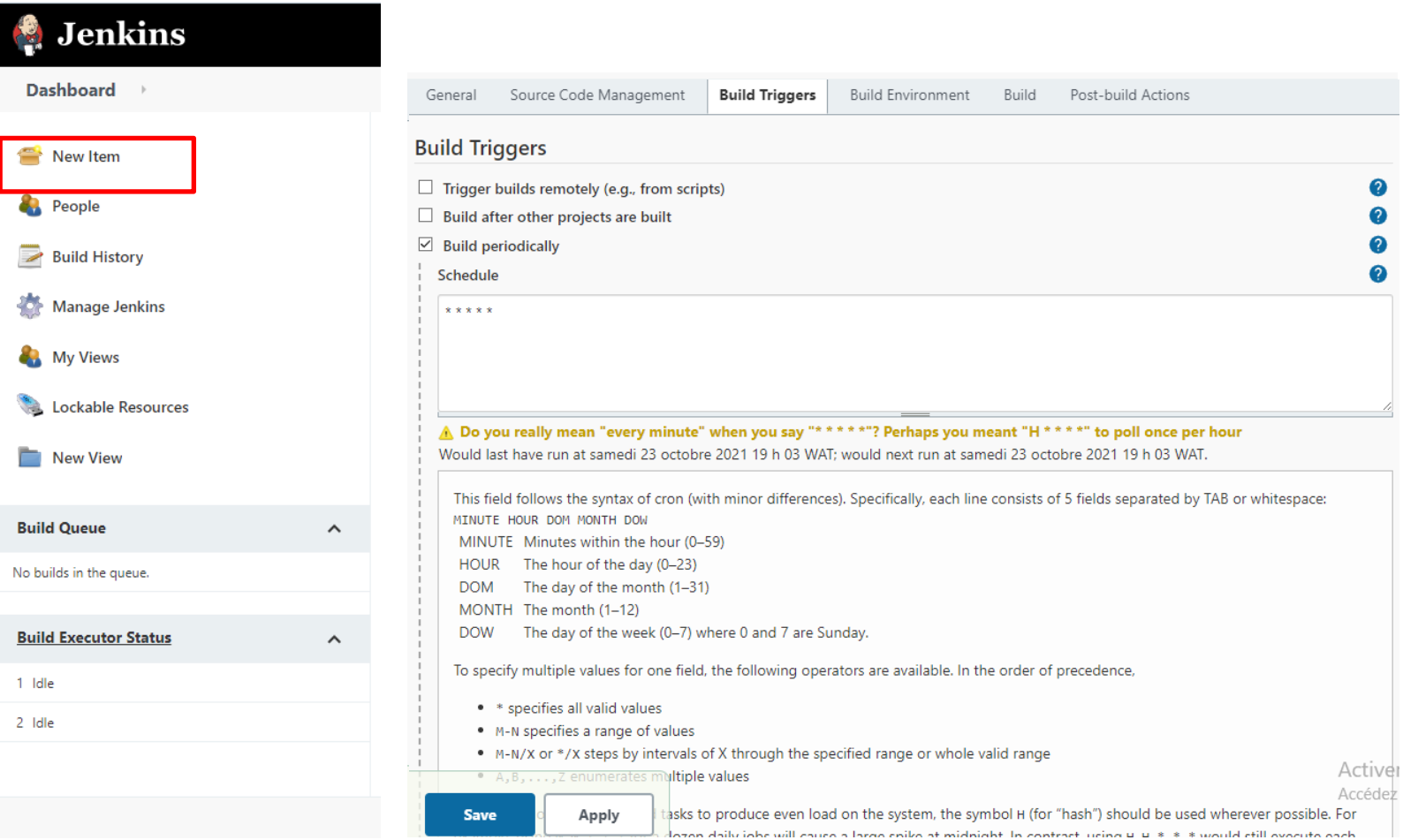
JENKINS

SECOND JOB : Déclenchement automatique des jobs



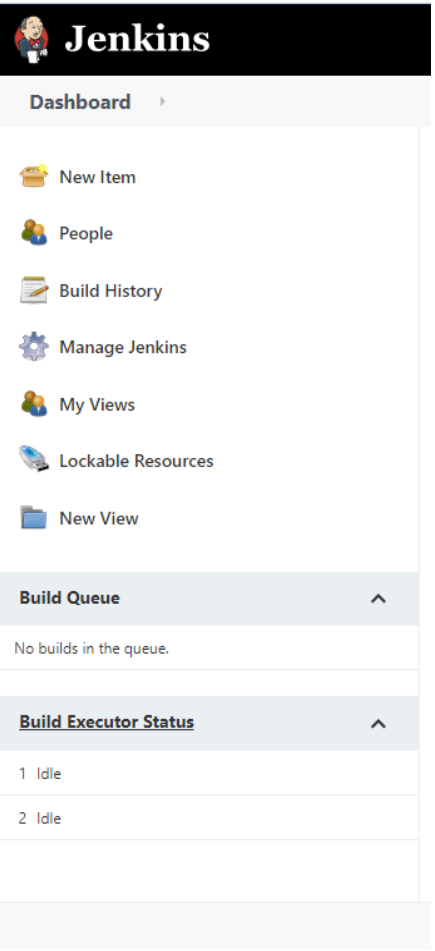
JENKINS

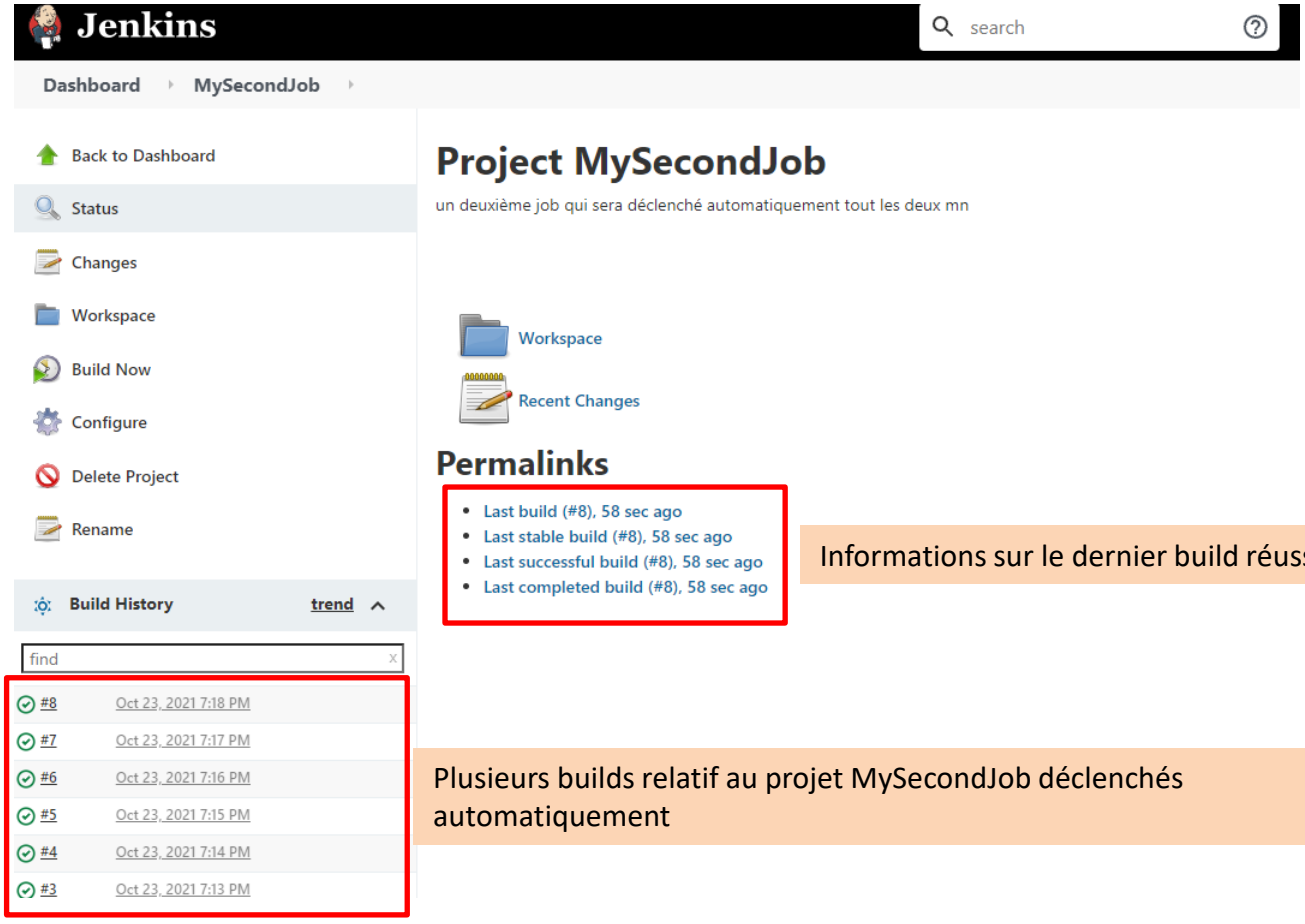
SECOND JOB : Déclenchement automatique des jobs



JENKINS

SECOND JOB : Déclenchement automatique des jobs





Informations sur le dernier build réussi

Plusieurs builds relatif au projet MySecondJob déclenchés automatiquement

JENKINS

JENKINS PIPELINE

Pipeline est un module de plus en plus populaire permettant de configurer les tâches au travers du code (**script**). Ce dernier peut être sauvegardé avec le reste du code source de votre projet, sur votre serveur de contrôle de version.

Les scripts peuvent s'écrire de deux manières différentes, soit en utilisant une syntaxe de script, soit une syntaxe déclarative. Voici l'architecture d'un Pipeline scripté :

```
node {
            stage('Build') {
                        //
            }
            stage('Test') {
                        //
            }
            stage('Deploy') {
//          }
}
```

Syntaxe Script

```
pipeline {
            agent any
            stages {
                        stage('Build') {
                                    steps { // }
                        }
                        stage('Test') {
                                    steps { // }
                        } stage('Deploy') {
                                    steps { // }
                        }
            }
}
```

Syntaxe Déclarative

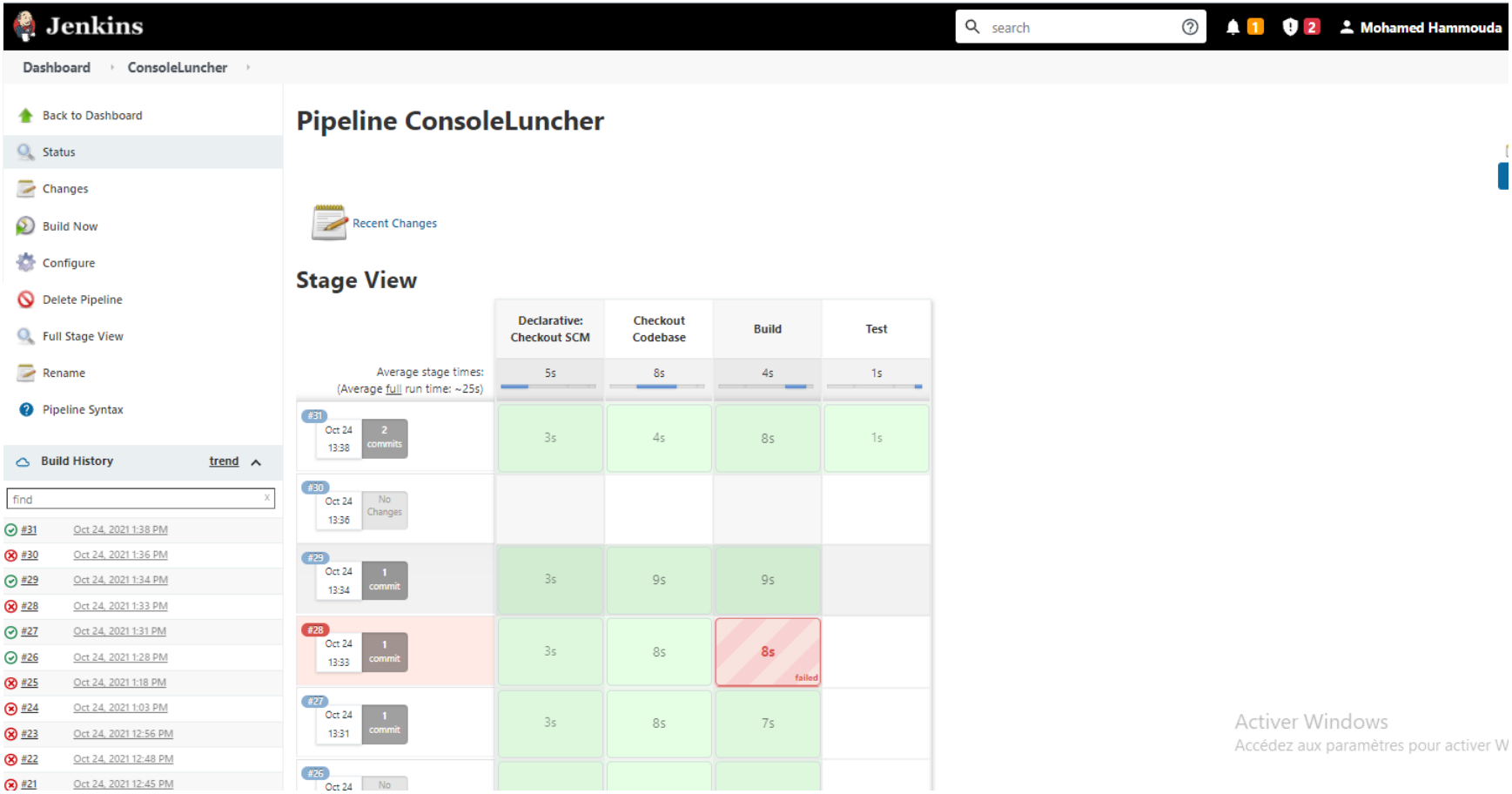
JENKINS

JENKINS PIPELINE

```
pipeline {
 agent any
  stages {
   stage('Checkout Codebase') {
    steps {
     cleanWs()
     checkout scm: [$class: 'GitSCM',userRemoteConfigs: [[credentialsId: 'GitHubSShkey',url:
'git@github.com:MohamedHammouda/ConsoleLuncher.git']]]   }
    }
   stage('Build'){
    steps{
     bat 'mkdir lib'
     bat 'cd lib'
     bat 'curl.exe -o "lib/junit-platform-console-standalone-1.8.1.jar"
"https://repo1.maven.org/maven2/org/junit/platform/junit-platform-console-standalone/1.8.1/junit-platform-console-
standalone-1.8.1.jar"'
     bat 'javac -d target -cp target;"lib/junit-platform-console-standalone-1.8.1.jar" src/test/java/FirstUnitTest.java'
     }
   }
   stage('Test'){
    steps{
     bat 'java -jar "lib/junit-platform-console-standalone-1.8.1.jar" -cp target -c FirstUnitTest'
     }      }
  }     }
```

JENKINS

JENKINS PIPELINE

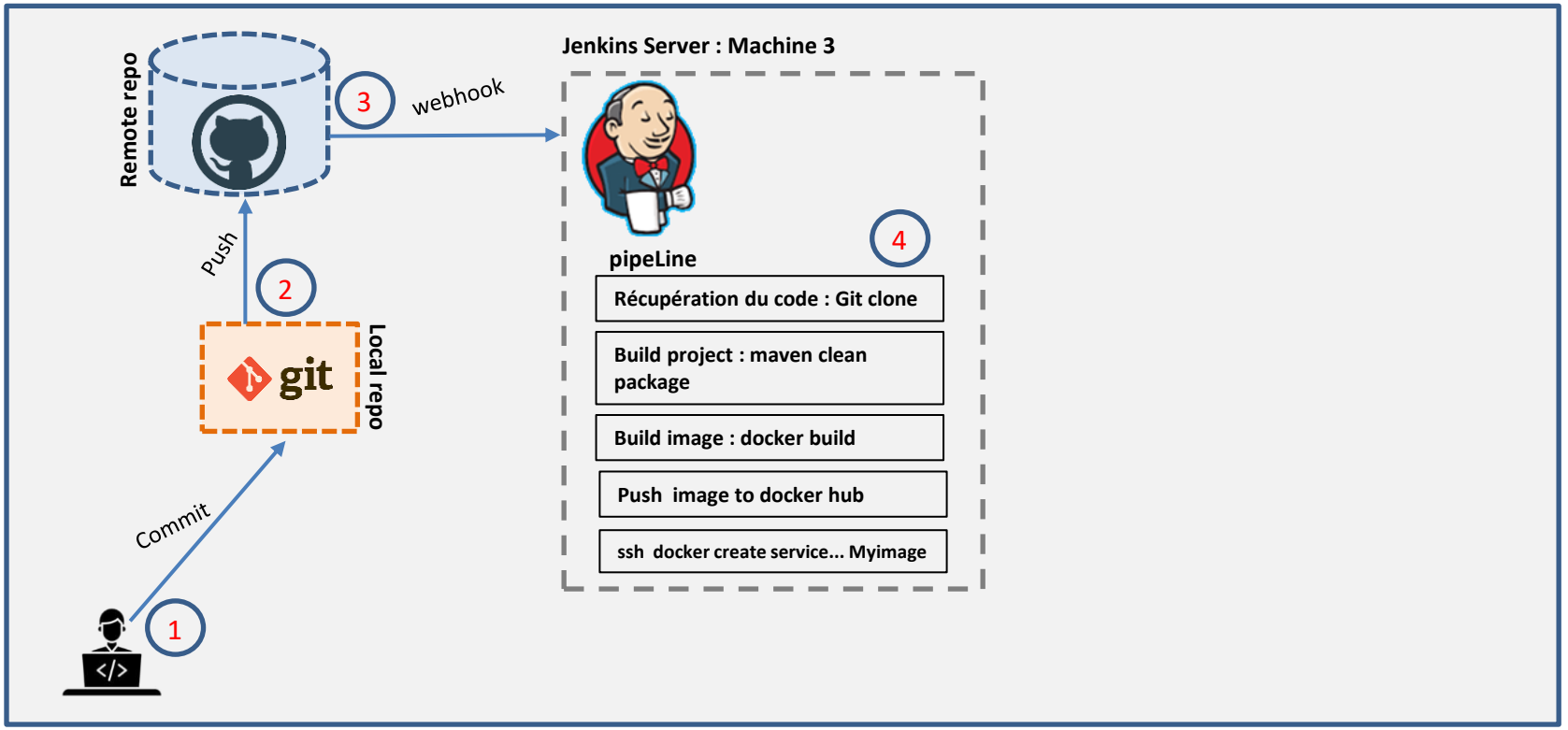


JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

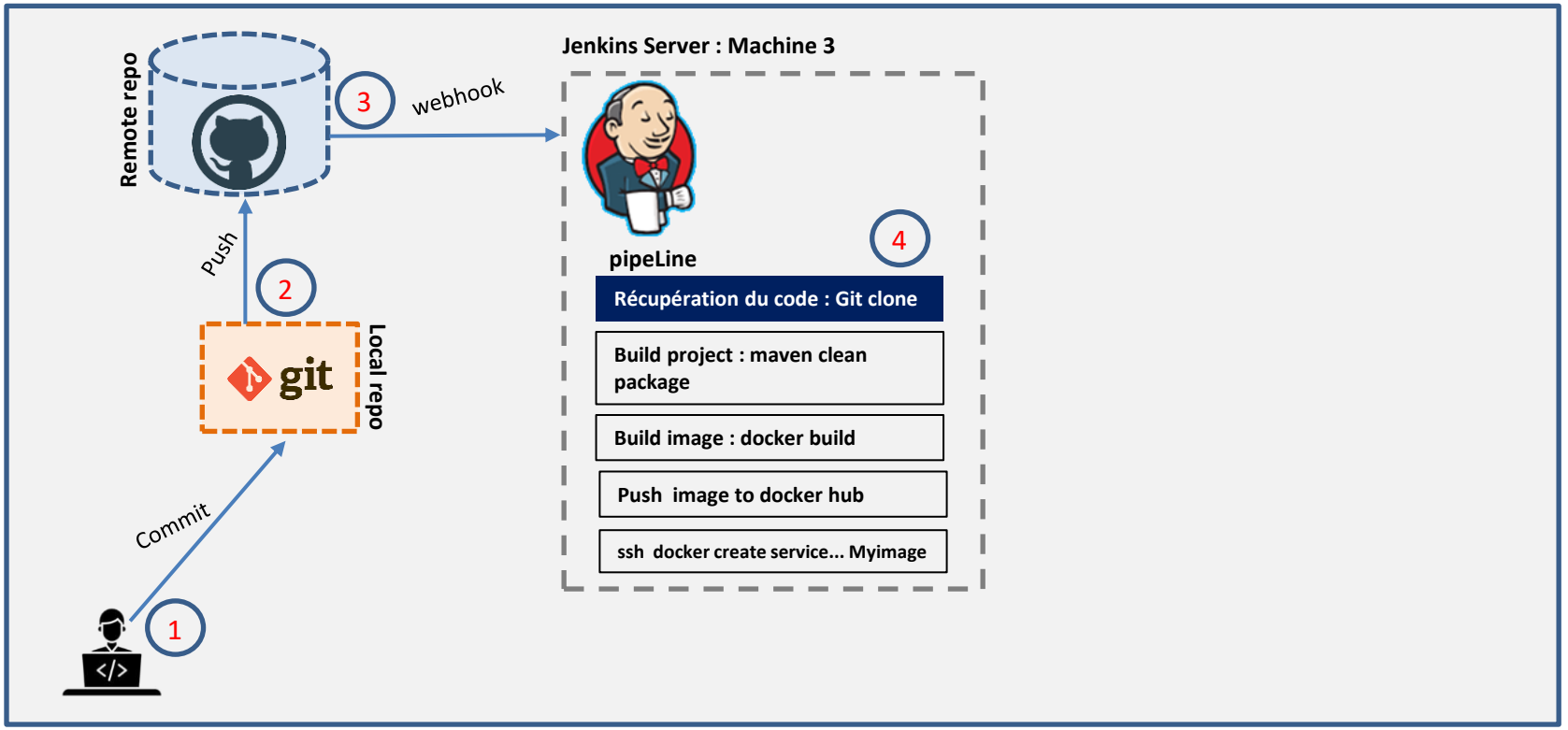


JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

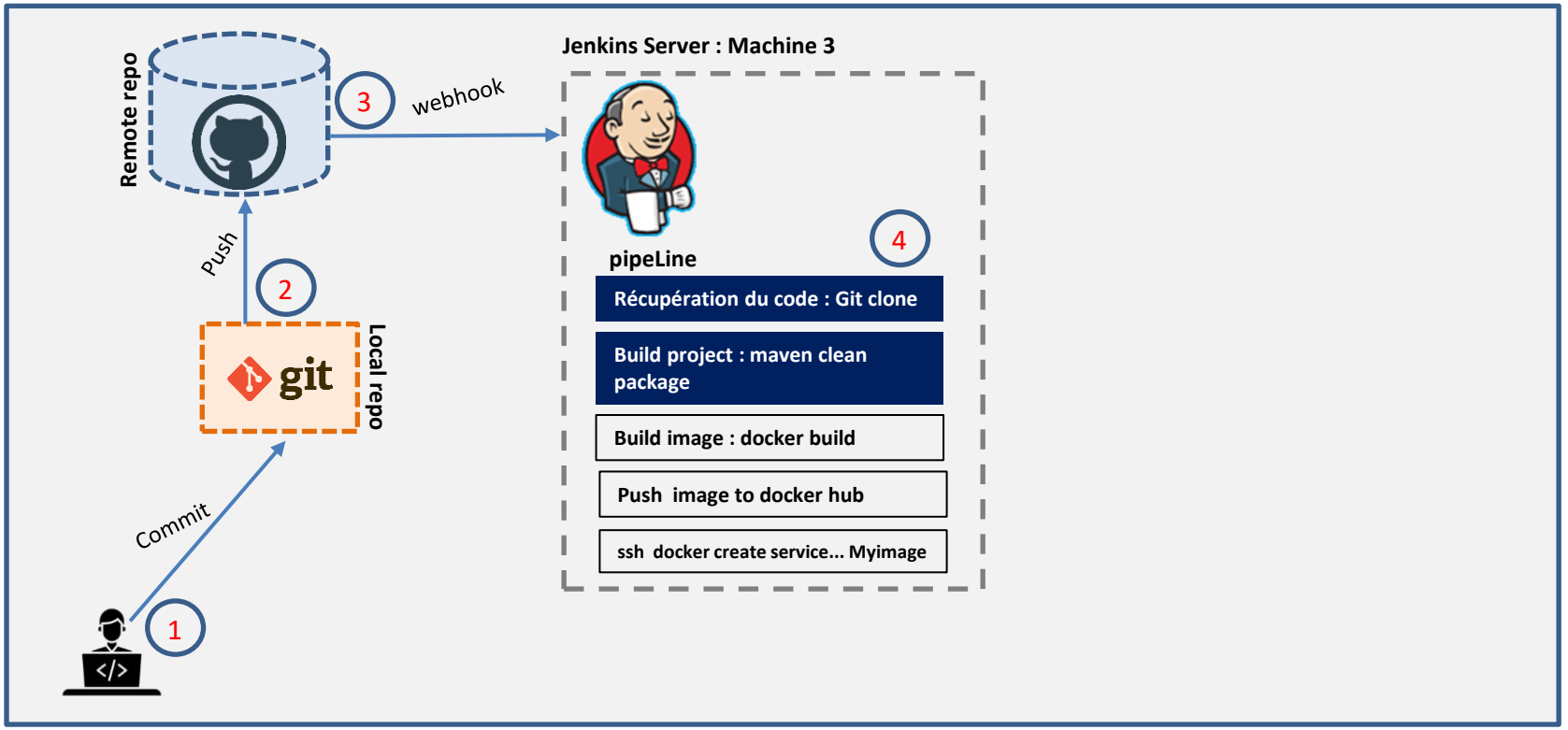


JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

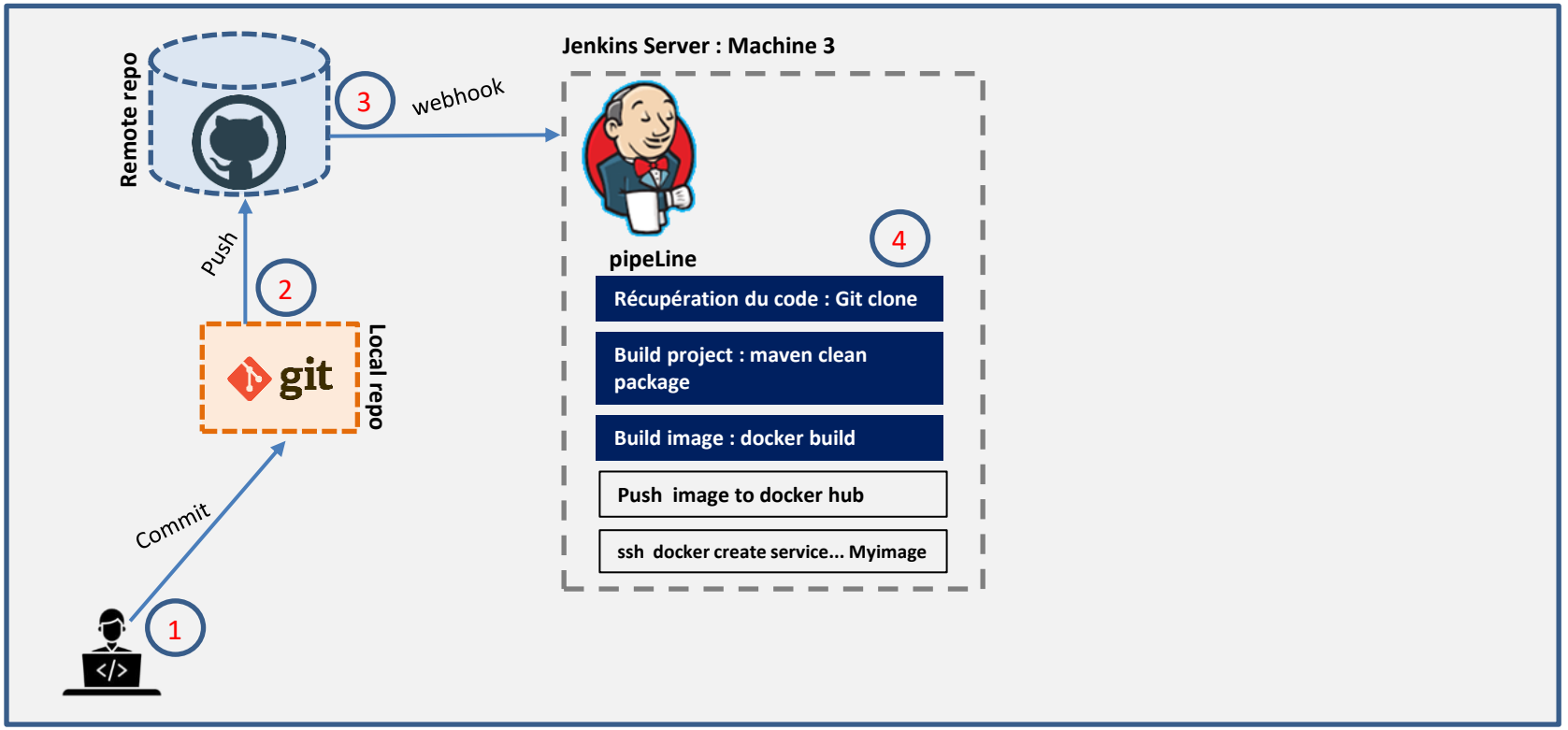


JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

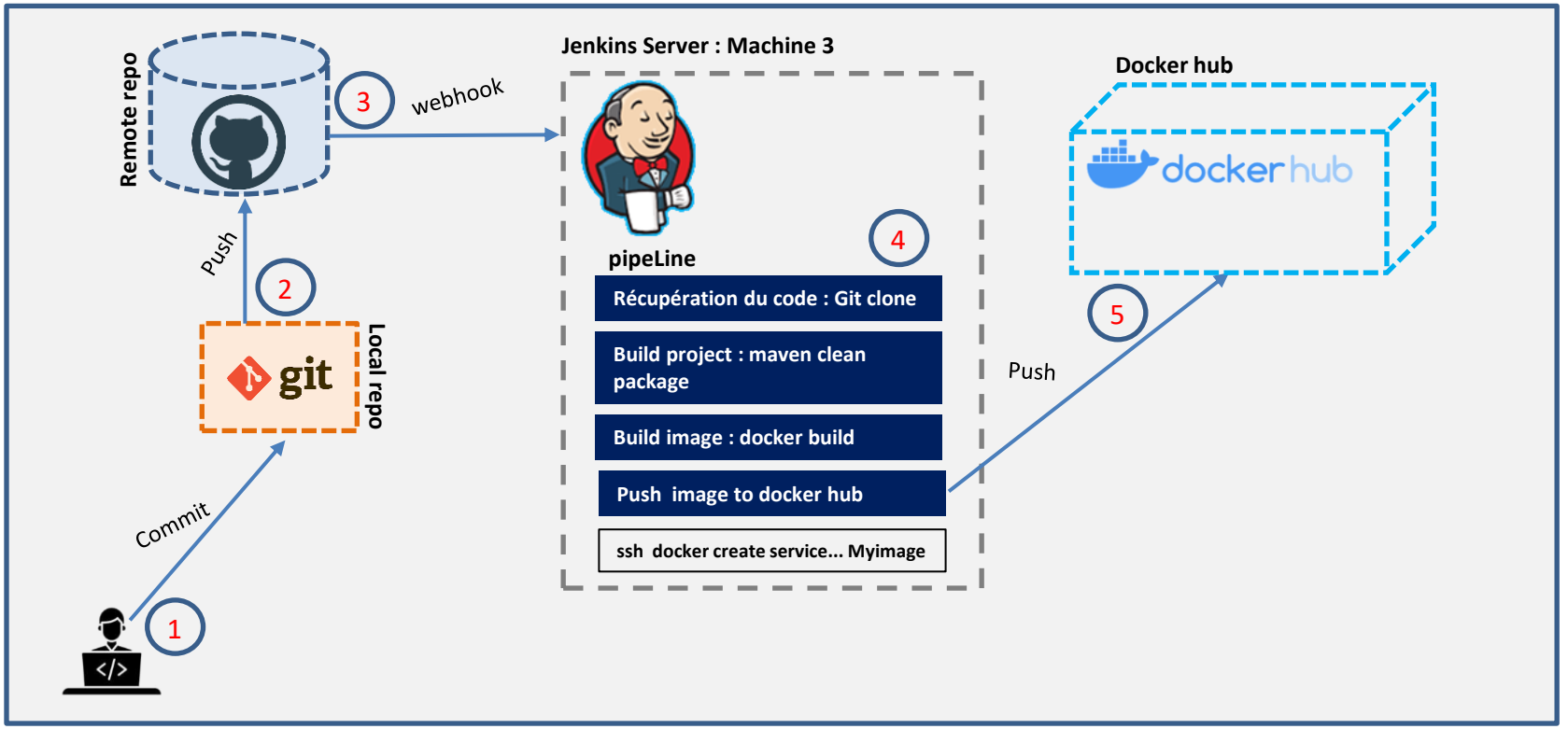


JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

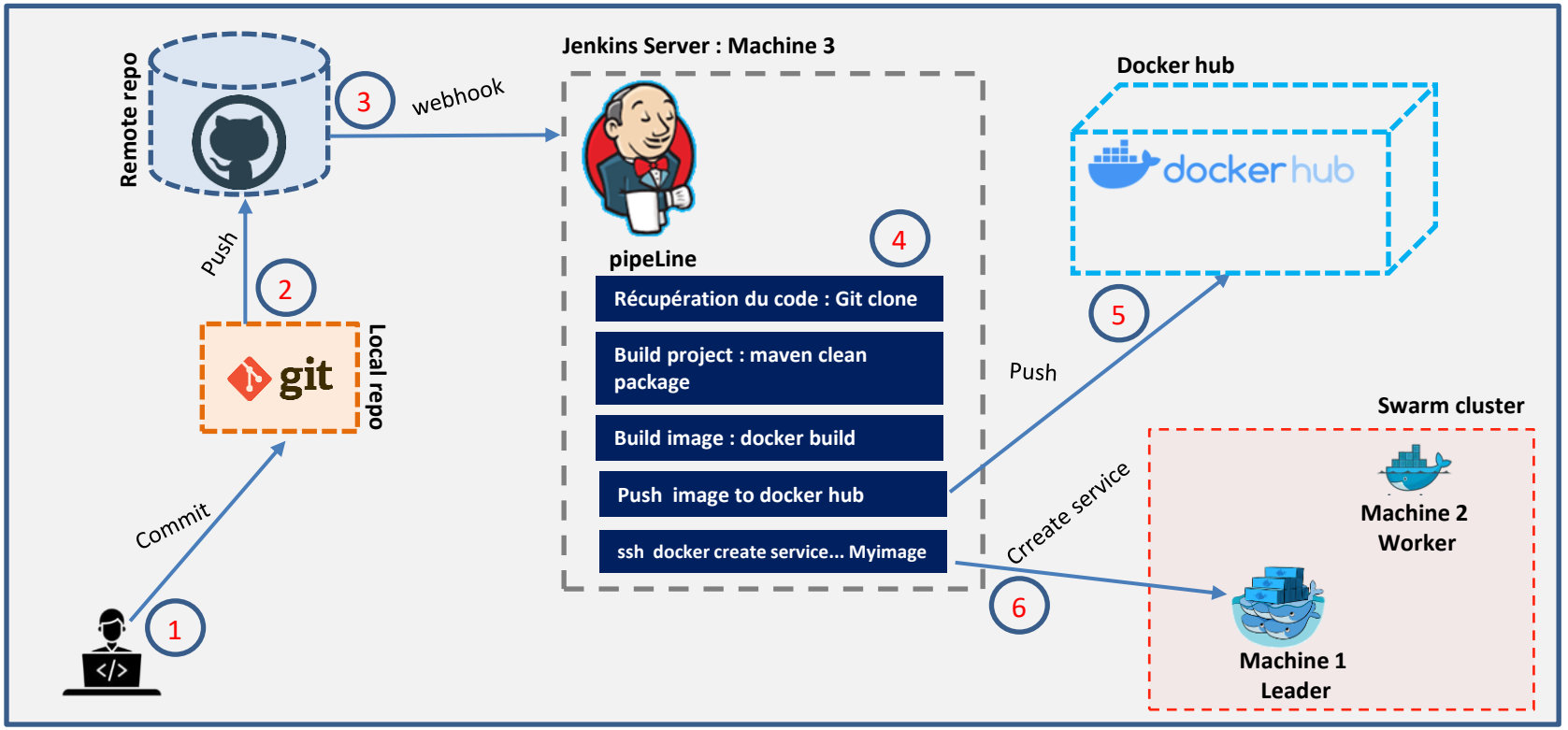


JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

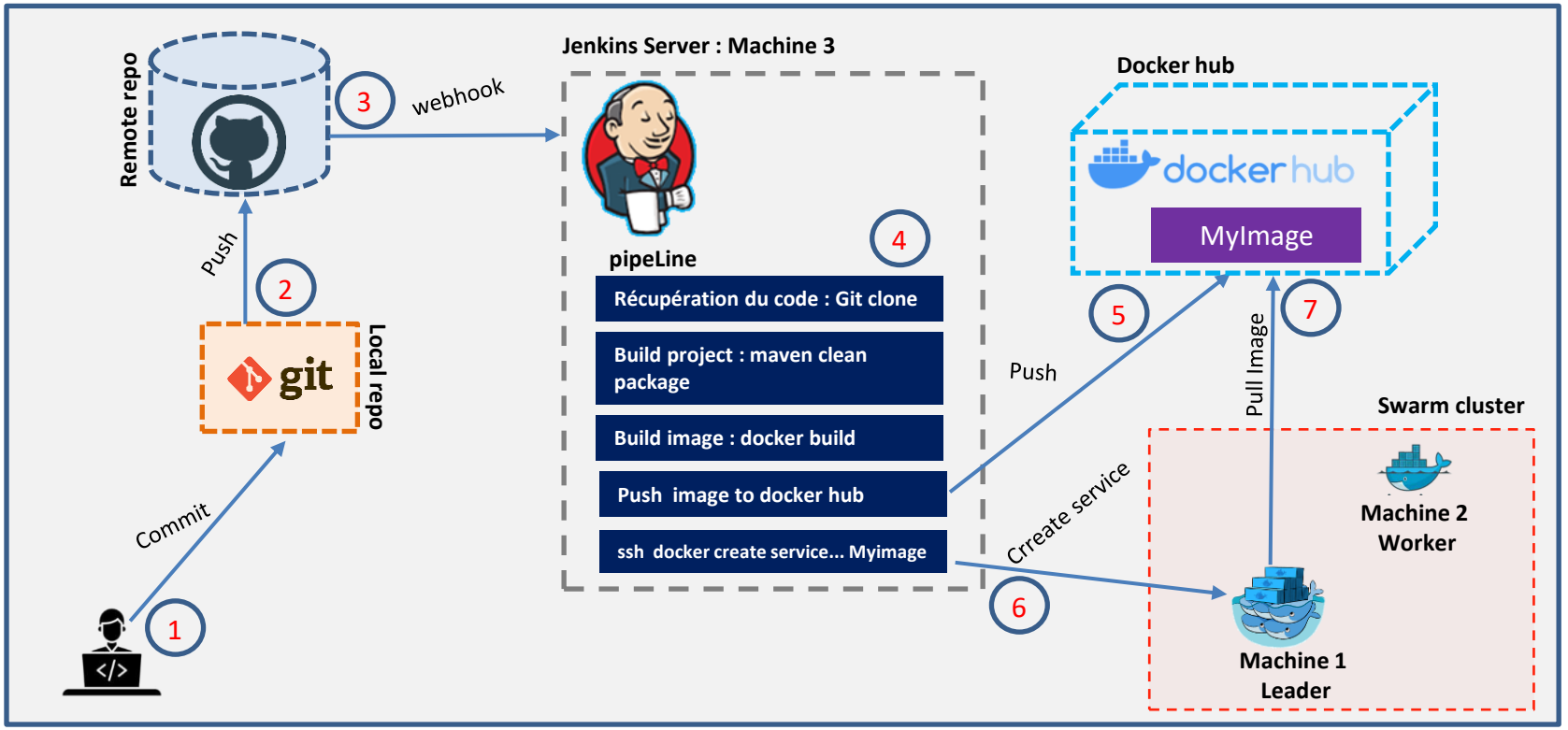


JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

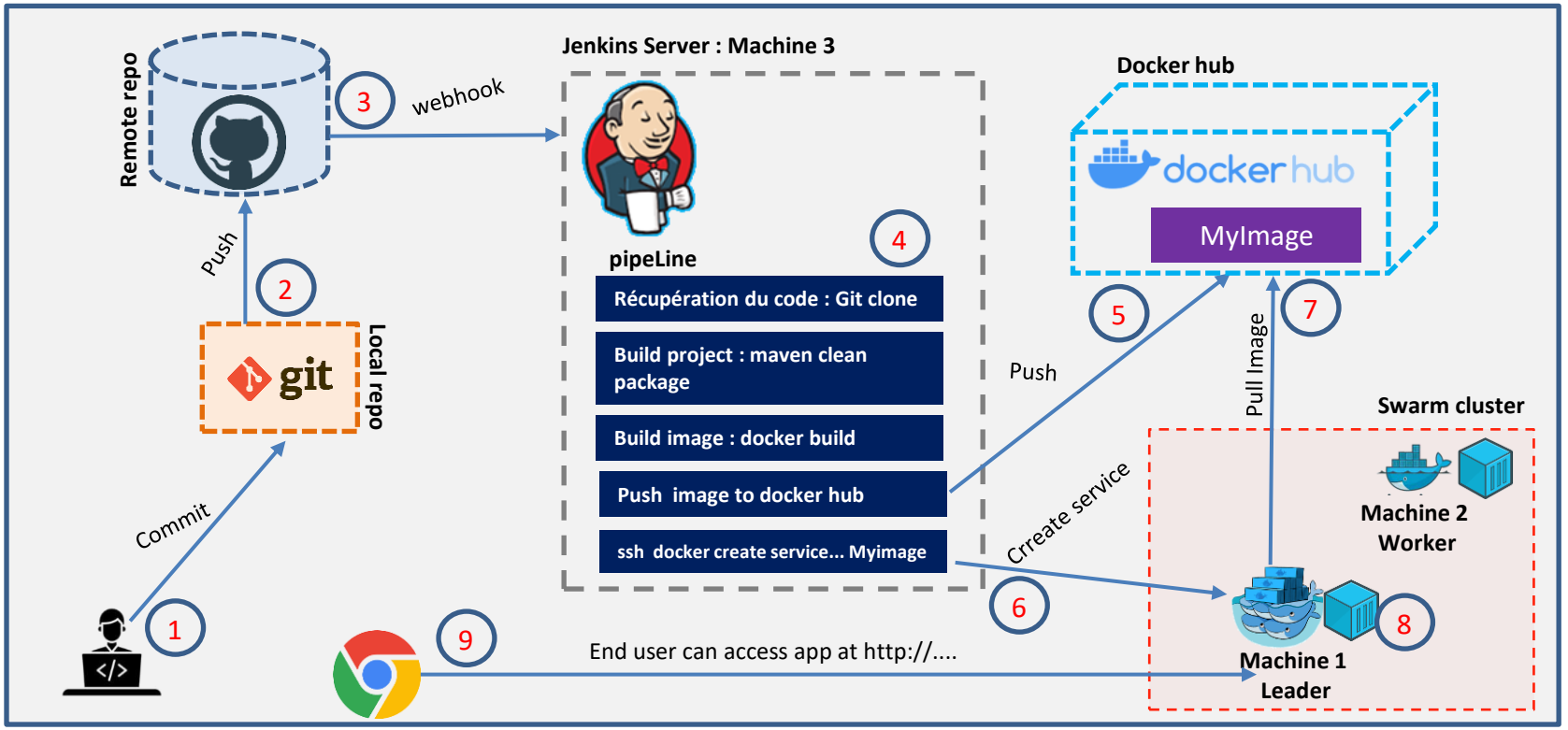


JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM



JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT
TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

- **Objectif de l'atelier**
  Utiliser jenkins pipeline afin de déployer sur un cluster swarm un site web conteneurisé

- **Prérequis de l'atelier**
  - Trois machines virtuelles
    - Deux machines pour le Swarm (cluster de déploiement)
      - swarm**1 (leader)**, swarm**2 (worker)**
    - Une machine pour le serveur jenkins
      - jenkins installé
      - docker installé
      - *usermod –aG docker jenkins* : s'assurer que jenkins fait partie du docker group afin que jenkins job puisse exécuter les commandes docker.

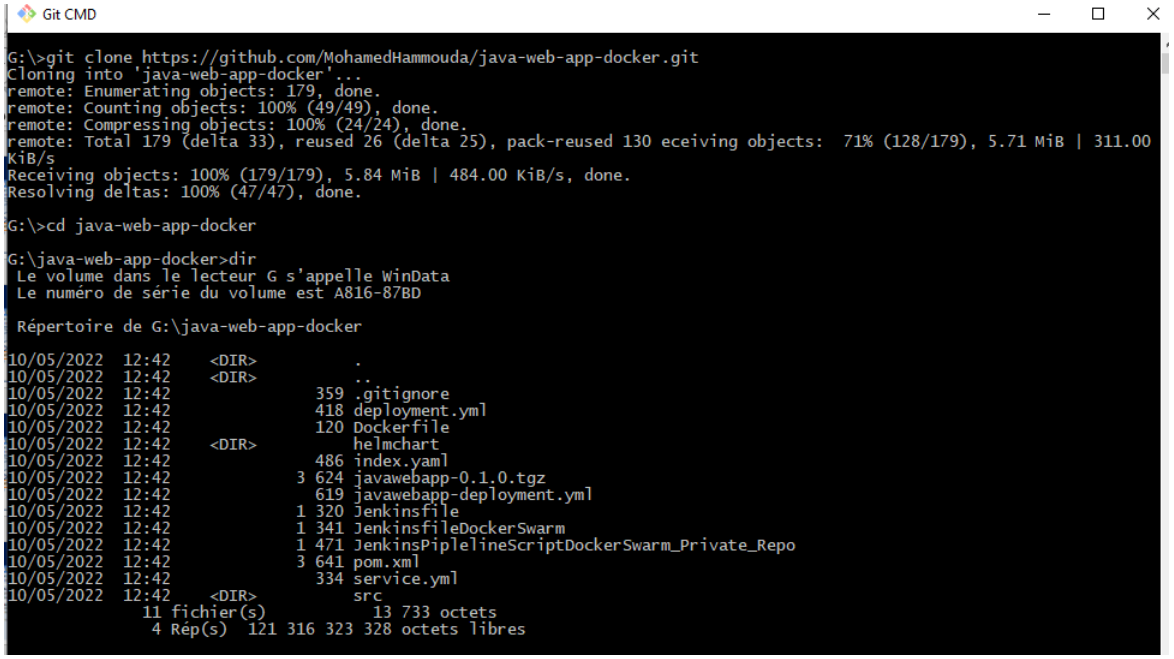
JENKINS-SWARM CI/CD

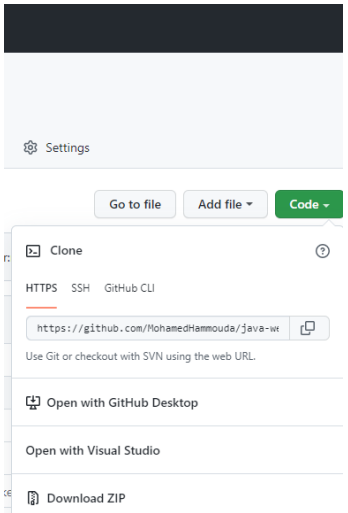
JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

1. Accéder à : https://github.com/MohamedHammouda/java-web-app-docker.git
   Dupliquer le projet dans votre repo distant puis cloner le sur votre machine physique





JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

2

- Initialiser un cluster swarm composé des deux machines virtuelle swarm1 & swarm2
- Au lieu de créer des nouvelles MV, vous pouvez coloner une machine existante contenant docker
  Bouton droit --> Manage--> clone
- Renommer vos machine swarm1 et swarm 2
- Les machines crclonées auront les même **hostnames**, pour ne pas vous induire à l'erreur, vous pouvez modifier le hostname de chaque machine en les appelant respectivement sw1 et sw2

```
Swarm1$  hostnamectl set-hostname sw1
Swarm1$ docker swarm init

Swarm2$  hostnamectl set-hostname sw2
Swarm2$ docker swarm join --token SWMTKN-1-4lc06x4w8pvizc91lynymn2v7a2citpnqwhc2ucigcuakuy3qe-9qvh6p8pwqq73rypnkwnyqhl9 192.168.236.133:2377
```

- S'assurer que le cluster est créé :

```
Swarm1$ docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
pyhu1txtega7k5n7ssue1seo2 *	sw1	Ready	Active	**Leader**	20.10.14
v8u9aigpaa3p0clc2er55xrj2	sw2	Ready	Active		20.10.14

JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

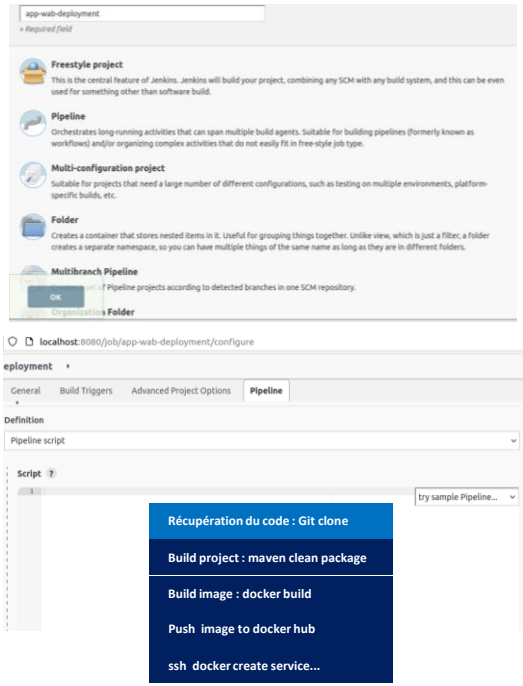
TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

3

- Préparer le pipeline au niveau de jenkins server (machine3 jenkinsServer) par la Création d'un nouveau job pipeline
  new item-->pipeline-->ok-->app-wab-deployment
- Définir la pipeline du projet :

Etape 1 :

```
stage('SCM Checkout'){
        git url: 'https://github.com/MithunTechnologiesDevOps/java-web-app-docker.git',branch: 'master'
      }
```



JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

3

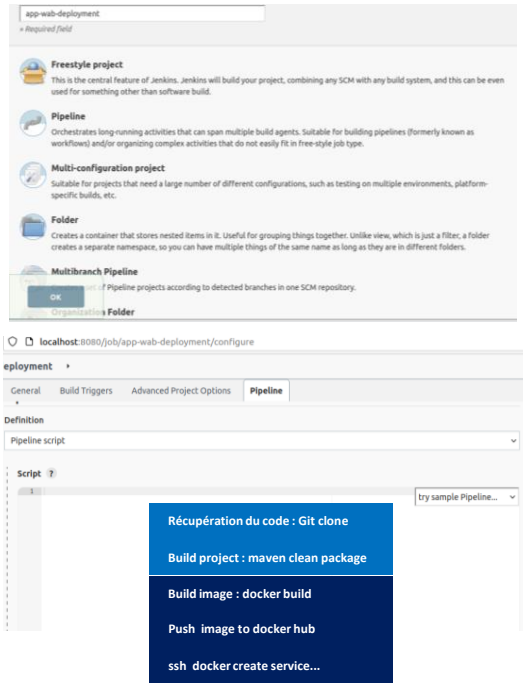
- Préparer le pipeline au niveau de jenkins server (machine3 jenkinsServer) par la Création d'un nouveau job pipeline
  new item-->pipeline-->ok-->app-wab-deployment
- Définir la pipeline du projet :

Etape 1 :

```
stage('SCM Checkout'){
        git url: 'https://github.com/MithunTechnologiesDevOps/java-web-app-
docker.git',branch: 'master'
    }
```

Etape 2 :

```
stage(" Maven Clean Package"){
        def mavenHome =  tool name: "Maven-3.5.6", type: "maven"
        def mavenCMD = "${mavenHome}/bin/mvn"
        sh "${mavenCMD} clean package"
         }
```



JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

3

- Préparer le pipeline au niveau de jenkins server (machine3 jenkinsServer) par la Création d'un nouveau job pipeline
  new item-->pipeline-->ok-->app-wab-deployment
- Définir la pipeline du projet :

Etape 1 :

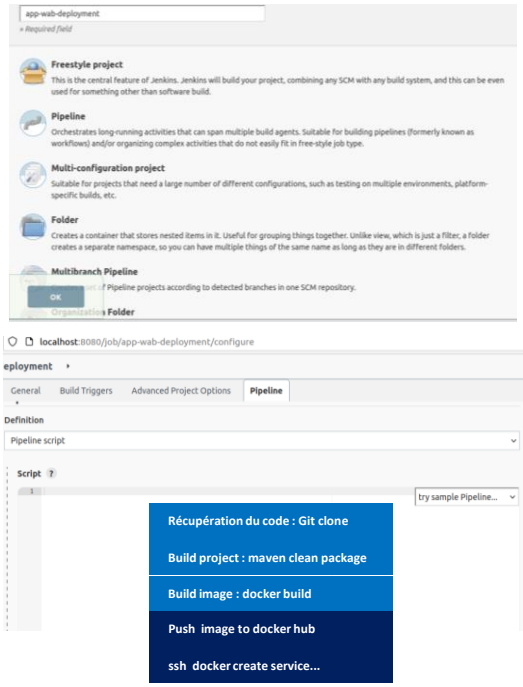
```
stage('SCM Checkout'){
        git url: 'https://github.com/MithunTechnologiesDevOps/java-web-app-
docker.git',branch: 'master'
    }
```

Etape 2 :

```
stage(" Maven Clean Package"){
        def mavenHome =  tool name: "Maven-3.5.6", type: "maven"
        def mavenCMD = "${mavenHome}/bin/mvn"          sh "${mavenCMD} clean package"
         }
```

Etape 3 :

```
 stage('Build Docker Image'){
        sh 'docker build -t mohamedhammouda/java-web-app .'   }
```



JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

3

- Préparer le pipeline au niveau de jenkins server (machine3 jenkinsServer) par la Création d'un nouveau job pipeline
  new item-->pipeline-->ok-->app-wab-deployment
- Définir la pipeline du projet :

Etape 1 :

```
stage('SCM Checkout'){
        git url: 'https://github.com/MithunTechnologiesDevOps/java-web-app-
docker.git',branch: 'master'
    }
```

Etape 2 :

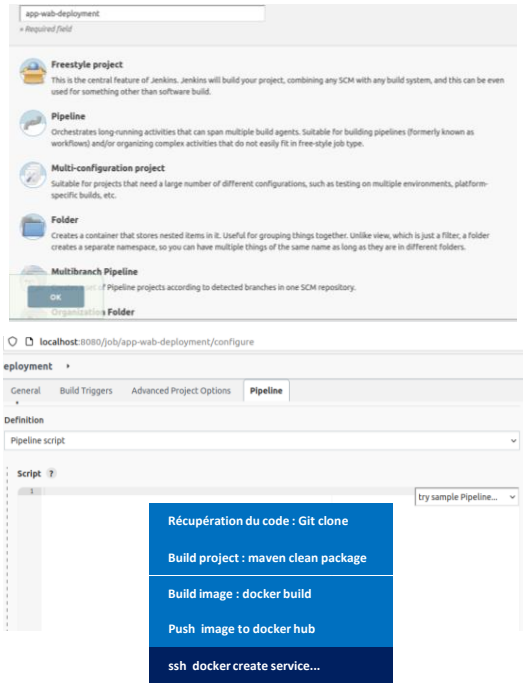
```
stage(" Maven Clean Package"){
        def mavenHome =  tool name: "Maven-3.5.6", type: "maven"
        def mavenCMD = "${mavenHome}/bin/mvn"          sh "${mavenCMD} clean package"
         }
```

Etape 3 :

```
 stage('Build Docker Image'){
        sh 'docker build -t mohamedhammouda/java-web-app .'   }
```

Etape 4 :

```
stage('Push Docker Image'){
        withCredentials([string(credentialsId: 'docker_hub_pwd', variable:
'docker_hub_pwd')]) {
           sh "docker login -u mohamedhammouda -p ${docker_hub_pwd}"          }
        sh 'docker push mohamedhammouda/java-web-app'    }
```



JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

3

- Préparer le pipeline au niveau de jenkins server (machine3 jenkinsServer) par la Création d'un nouveau job pipeline

Etape 1 :

```
stage('SCM Checkout'){
        git url: 'https://github.com/MithunTechnologiesDevOps/java-web-app-
docker.git',branch: 'master'
    }
```

Etape 2 :

```
stage(" Maven Clean Package"){
        def mavenHome =  tool name: "Maven-3.5.6", type: "maven"
        def mavenCMD = "${mavenHome}/bin/mvn"          sh "${mavenCMD} clean package"
         }
```

Etape 3 :

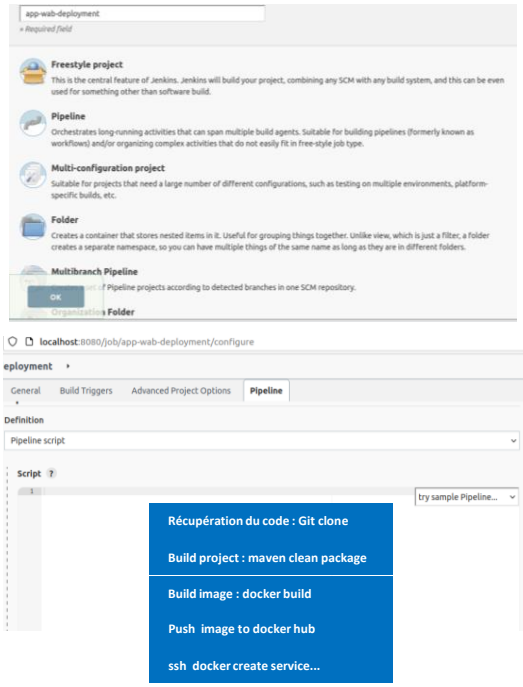
```
 stage('Build Docker Image'){
        sh 'docker build -t mohamedhammouda/java-web-app .'   }
```

Etape 4 :

```
stage('Push Docker Image'){
        withCredentials([string(credentialsId: 'docker_hub_pwd', variable:
'docker_hub_pwd')]) {
           sh "docker login -u mohamedhammouda -p ${docker_hub_pwd}"          }
        sh 'docker push mohamedhammouda/java-web-app'    }
```

Etape 5 :

```
stage('Run Docker Image In Dev Server'){
```



JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

3

Etape 5 :

- L'étape 5 nécessite un accès distant (remote access) entre jenkinsserver et swarm1. créer une paire de clé (privé&pub) sur la machine jenkinsserver

  Jenkinsserver# ssh-keygen -t ed25519 -b 4096 -C "your_email@example.com"

- Deux fichiers sont créés dans le repertoire ~/.ssh

  ls -al ~/.ssh/

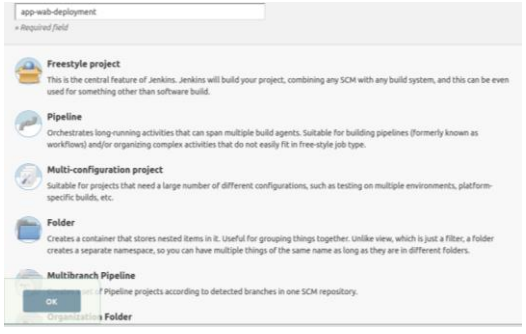
  drwx------ 2 mohamed mohamed 4096 May 9 00:06 .
  drwxr-xr-x 16 mohamed mohamed 4096 May 8 14:40 ..
  -rw------- 1 mohamed mohamed 411 May 8 23:59 id_ed25519
  -rw-r--r-- 1 mohamed mohamed 103 May 8 23:59 id_ed25519.pub

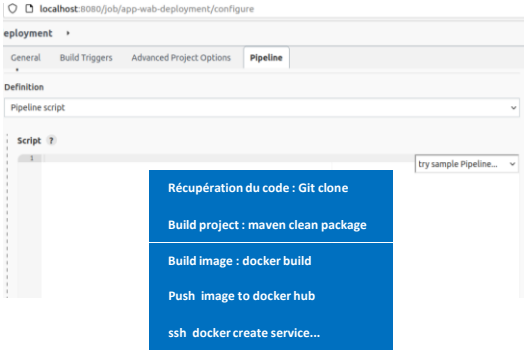
- Sur la machine swarm1(leader) créer un repertoire **.ssh** s'il n'existe pas
- Copier la clé public de la machine jenkinsServer vers la machine swarm1

  Jenkinsserver# scp ~/.ssh/id_ed25519.pub sw1@<ipOfSwarm1> ~/.ssh/uploaded_key.pub

- Vérifier l'existence du fichier uploaded_key.pub sur la machine swarm1. Créer un ficher authorized_key contenant les données du fichier id_e25519. la clé contenue dans ce fichier sera utilisé à chaque fois qu'il aura une connexion ssh

  sw1# cat ~/.ssh/uploaded_key.pub >> ~ /.ssh/authorized_key





JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

3

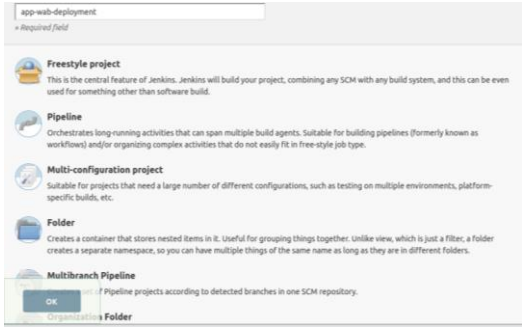
Etape 5 :

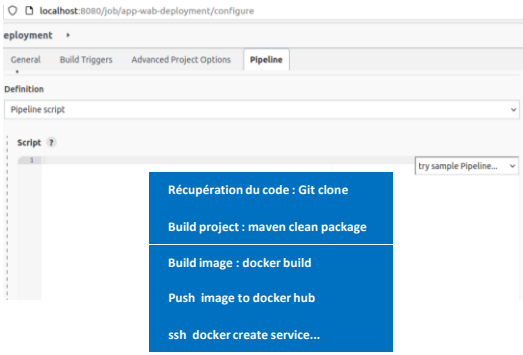
- Octroyer les privileges nécessaires au repertoire .ssh et ses fichiers

  Jenkinsserver# chmod 700 ~/.ssh
  Jenkinsserver# chmod 600 ~/.ssh/*

- Vérifier que vous pouvez maintenant connecter de jenkinsServer à swarm1

  Jenkinsserver# ssh sw1@<ip of Swarm1>





JENKINS-SWARM CI/CD

JENKINS DOCKER SWARM INTEGRATION

CI/CD PIPELINE SCRIPT

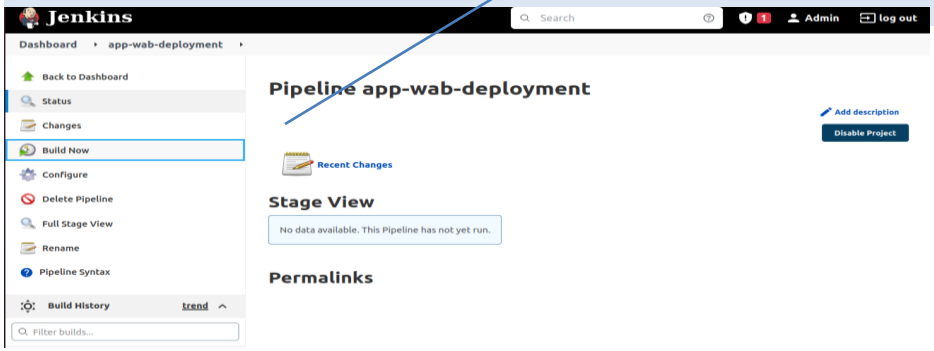
TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM

3

Etape 5 :

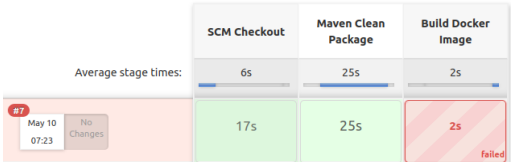
```
stage('Run Docker Image In Dev Server'){
            def dockerRun = ' docker service create  -d -p 8080:8080 --name java-web-app --replicas 2  mohamedhammouda/java-web-app'

        sshagent(['Docker_Swarm_Manager_SSH']) {
          sh 'ssh -o StrictHostKeyChecking=no mohamed@192.168.236.129 docker stop java-web-app || true'
          sh 'ssh  mohamed@192.168.236.129 docker rm java-web-app || true'
          sh 'ssh  mohamed@192.168.236.129 docker rmi -f $(docker images -q) || true'
          sh "ssh  mohamed@192.168.236.129 ${dockerRun}"
        }     }
```



Exécuter manuellement votre pipeline en cliquant sur la commande Builfnow
Il est à noter qu'on pourra configurer l'exécution automatique de la pipeline en utilisant le polling ou bien le webhook

Progression du build...
Echec au niveau de l'étape 3
Pourqupoi?? Consulter la console de la pipeline
Pour moi docker doemon is not yorking



jenkinsServer$ service docker start
Progression du build...
Echec au niveau de l'étape 5
Pourqupoi?? Consulter la console de la