

ATELIER : DEVOPS



Mohamed HAMMOUDA



■ 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 dépôts d'image, aussi bien que des scripts en shell Unix ou batch Windows.

JENKINS

FIRST JOB : New Item: Freestyle project



Jenkins

Dashboard

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

search

?

1

1

Enter an item name

MyFirstJob

» Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

Pipeline

Activ Accé

JENKINS

FIRST JOB : New Item: Freestyle project



Jenkins

Dashboard

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Lockable Resources
- New View

Build Queue

No builds in the queue.

Build Executor Status

- 1 Idle
- 2 Idle

General Source Code Management Build Triggers **Build Environment** Build Post-build Actions

- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM

Build Environment

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published Gradle build scans
- ☐ With Ant

Build

Execute Windows batch command

Command

```
Echo "Hello... Ceci est mon premier Job sous Jenkins" : %date%--%time%
```

See [the list of available environment variables](#)

Advanced...

Save Apply

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,

Active
Accédez

JENKINS

FIRST JOB : Build Now :



The screenshot shows the Jenkins dashboard for 'Project MyFirstJob'. The left sidebar contains links: Back to Dashboard, Status, Changes, Workspace, Build Now (highlighted with a blue bar), Configure, Delete Project, and Rename. Below these is the Build History section with a search bar and a 'trend' button. A hand icon points to the 'Build Now' button. The main content area shows 'Project MyFirstJob' with links to Workspace, Recent Changes, and Permalinks.

The screenshot shows the Jenkins 'Build #1' page for 'Project MyFirstJob'. The top bar shows the breadcrumb 'Dashboard > MyFirstJob > #1'. The main content area features a green checkmark icon and the text 'Build #1 (Oct 23, 2021 6:38:33 PM)'. Below this, it says 'No changes.' and 'Started by user Mohamed Hammouda'. The left sidebar is identical to the previous screenshot, but the 'Build Now' button is not highlighted. A hand icon points to the 'Console Output' link in the sidebar.

Console Output

Démarré par l'utilisateur [Mohamed Hammouda](#)

Running as SYSTEM

Building in workspace C:\Users\Mohamed\AppData\Local\Jenkins\.jenkins\workspace\MyFirstJob
[MyFirstJob] \$ cmd /c call C:\Users\Mohamed\AppData\Local\Temp\jenkins3160368824738058368.bat

C:\Users\Mohamed\AppData\Local\Jenkins\.jenkins\workspace\MyFirstJob>Echo "Hello... Ceci est mon premier Job sous Jenkins" : 23/10/2021--18:38:33,46
"Hello... Ceci est mon premier Job sous Jenkins" : 23/10/2021--18:38:33,46

C:\Users\Mohamed\AppData\Local\Jenkins\.jenkins\workspace\MyFirstJob>exit 0
Finished: SUCCESS

JENKINS

SECOND JOB : Déclenchement automatique des jobs



Jenkins

Dashboard

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Lockable Resources
- New View

Build Queue

No builds in the queue.

Build Executor Status

- 1 Idle
- 2 Idle

General Source Code Management Build Triggers **Build Environment** Build Post-build Actions

- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM

Build Environment

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published Gradle build scans
- ☐ With Ant

Build

Execute Windows batch command

Command

```
echo "Hello.... Ceci est mon deuxième JOB qui sera déclenché automatiquement tout les deux minutes" : %date%--%time%
```

See [the list of available environment variables](#)

Advanced...

Save Apply

JENKINS

SECOND JOB : Déclenchement automatique des jobs



Jenkins

Dashboard

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

General Source Code Management **Build Triggers** Build Environment Build Post-build Actions

Build Triggers

☐ Trigger builds remotely (e.g., from scripts) ?

☐ Build after other projects are built ?

☒ Build periodically ?

Schedule ?

⚠ Do you really mean "every minute" when you say "*****"? Perhaps you meant "H * * * *" to poll once per hour

Would last have run at samedi 23 octobre 2021 19 h 03 WAT; would next run at samedi 23 octobre 2021 19 h 03 WAT.

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE	HOUR	DOM	MONTH	DOW
MINUTE	Minutes within the hour (0–59)			
HOUR	The hour of the day (0–23)			
DOM	The day of the month (1–31)			
MONTH	The month (1–12)			
DOW	The day of the week (0–7) where 0 and 7 are Sunday.			

To specify multiple values for one field, the following operators are available. In the order of precedence,

- * specifies all valid values
- M-N specifies a range of values
- M-N/X or */X steps by intervals of X through the specified range or whole valid range
- A, B, ..., Z enumerates multiple values

Save

Apply

tasks to produce even load on the system, the symbol H (for "hash") should be used wherever possible. For
dozen daily jobs will cause a large spike at midnight. In contrast, using H * * * * would still execute each

Active
Accédez

JENKINS

SECOND JOB : Déclenchement automatique des jobs



Jenkins

Dashboard

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Lockable Resources
- New View

Build Queue

No builds in the queue.

Build Executor Status

1	Idle
2	Idle

Jenkins search ?

Dashboard > MySecondJob

Project MySecondJob
un deuxième job qui sera déclenché automatiquement tout les deux mn

Workspace

Recent Changes

Permalinks

- Last build (#8), 58 sec ago
- Last stable build (#8), 58 sec ago
- Last successful build (#8), 58 sec ago
- Last completed build (#8), 58 sec ago

Build History trend

	find
✓ #8	Oct 23, 2021 7:18 PM
✓ #7	Oct 23, 2021 7:17 PM
✓ #6	Oct 23, 2021 7:16 PM
✓ #5	Oct 23, 2021 7:15 PM
✓ #4	Oct 23, 2021 7:14 PM
✓ #3	Oct 23, 2021 7:13 PM

Informations sur le dernier build réussi

Plusieurs builds relatif au projet MySecondJob déclenchés automatiquement

■ JENKINS

■ JENKINS PIPELINE



Jenkins

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



Jenkins

```
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

Jenkins

search ?

1 2 Mohamed Hammouda

Dashboard ConsoleLuncher

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Build Now](#)

[Configure](#)

[Delete Pipeline](#)

[Full Stage View](#)

[Rename](#)

[Pipeline Syntax](#)

[Build History](#) trend ^

find x

#31

Oct 24 13:38

2 commits

#30

Oct 24 13:36

No Changes

#29

Oct 24 13:34

1 commit

#28

Oct 24 13:33

1 commit

#27

Oct 24 13:31

1 commit

#26

Oct 24

No

Average stage times:
(Average full run time: ~25s)

Declarative: Checkout SCM	Checkout Codebase	Build	Test
5s	8s	4s	1s
3s	4s	8s	1s
3s	9s	9s	
3s	8s	8s failed	
3s	8s	7s	

Active Windows
Accédez aux paramètres pour activer W

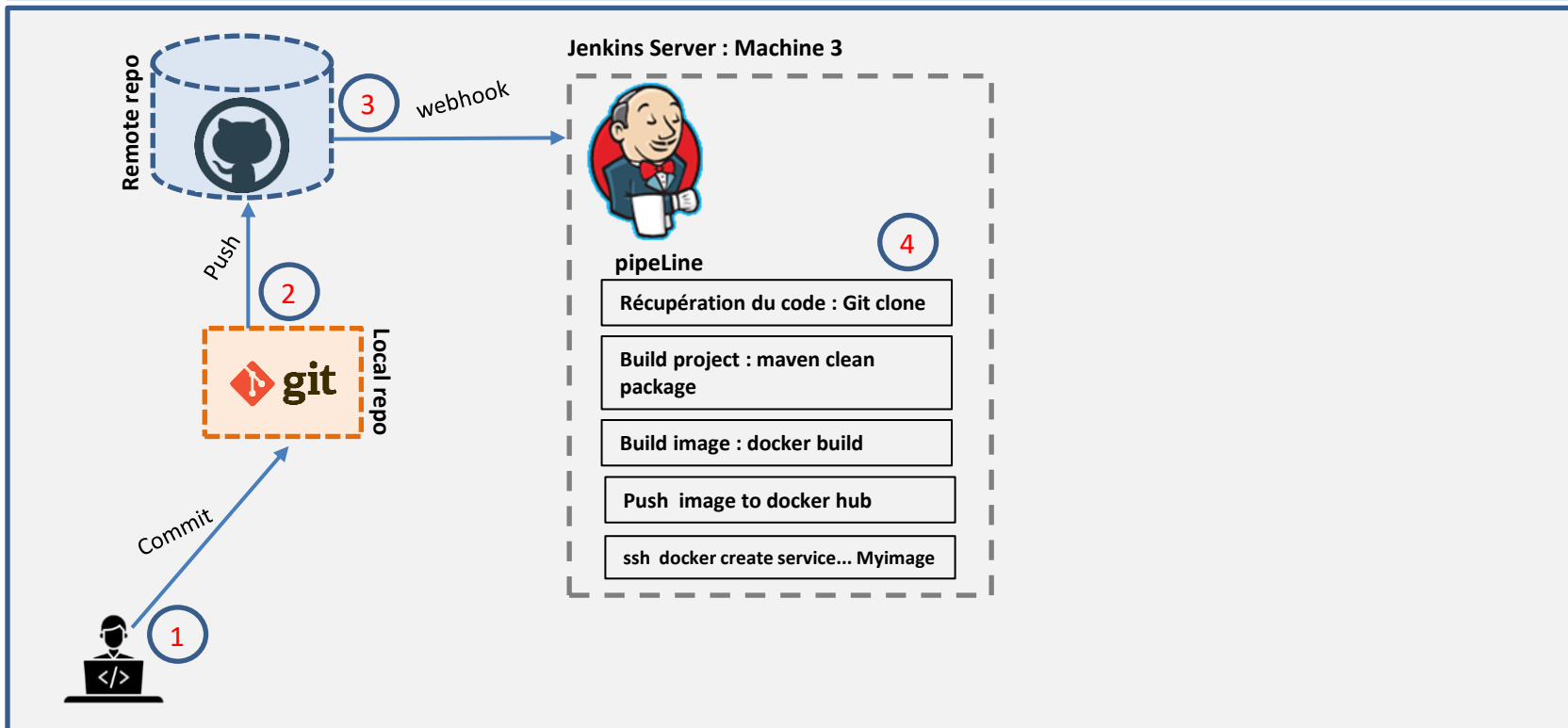
■ JENKINS-SWARM CI/CD

■ JENKINS DOCKER SWARM INTEGRATION



CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM



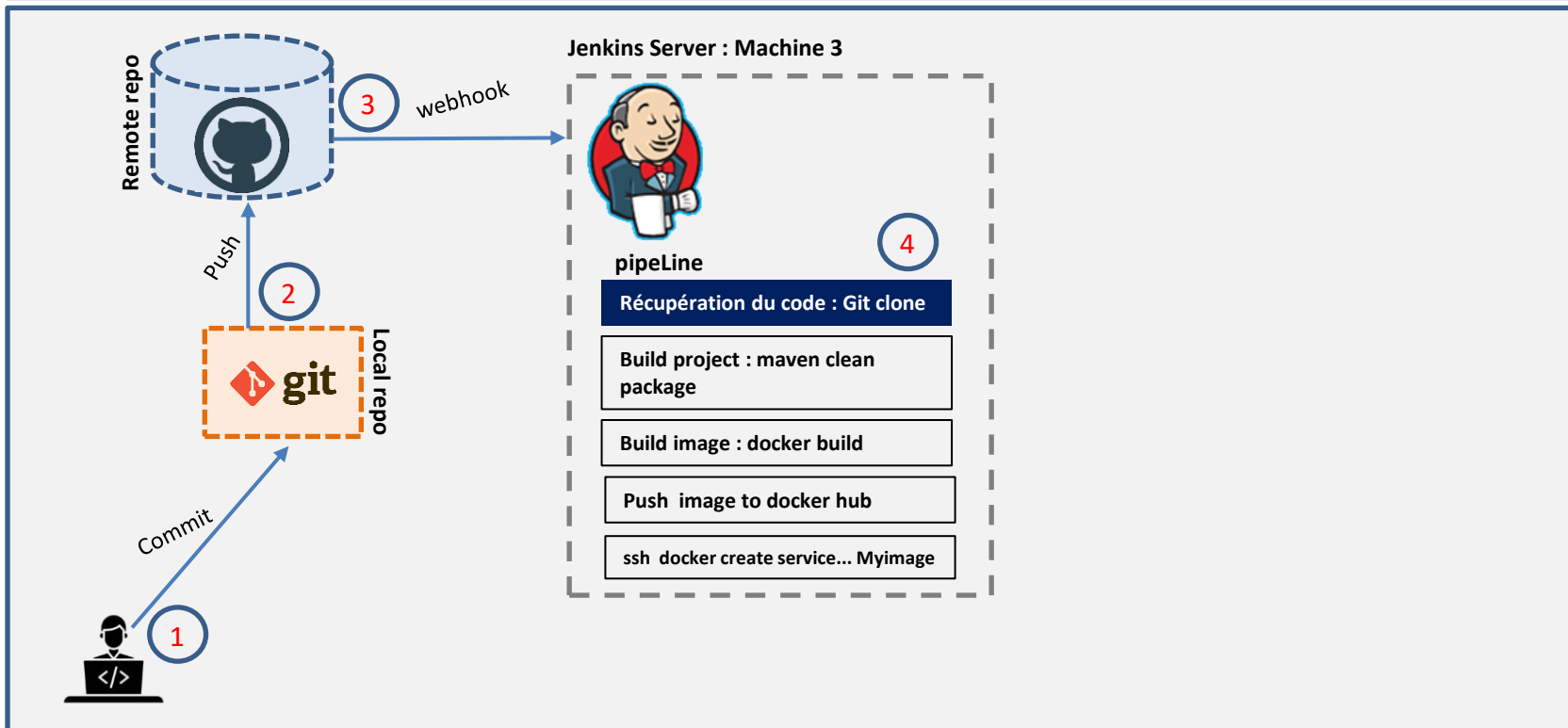
■ JENKINS-SWARM CI/CD

■ JENKINS DOCKER SWARM INTEGRATION



CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM



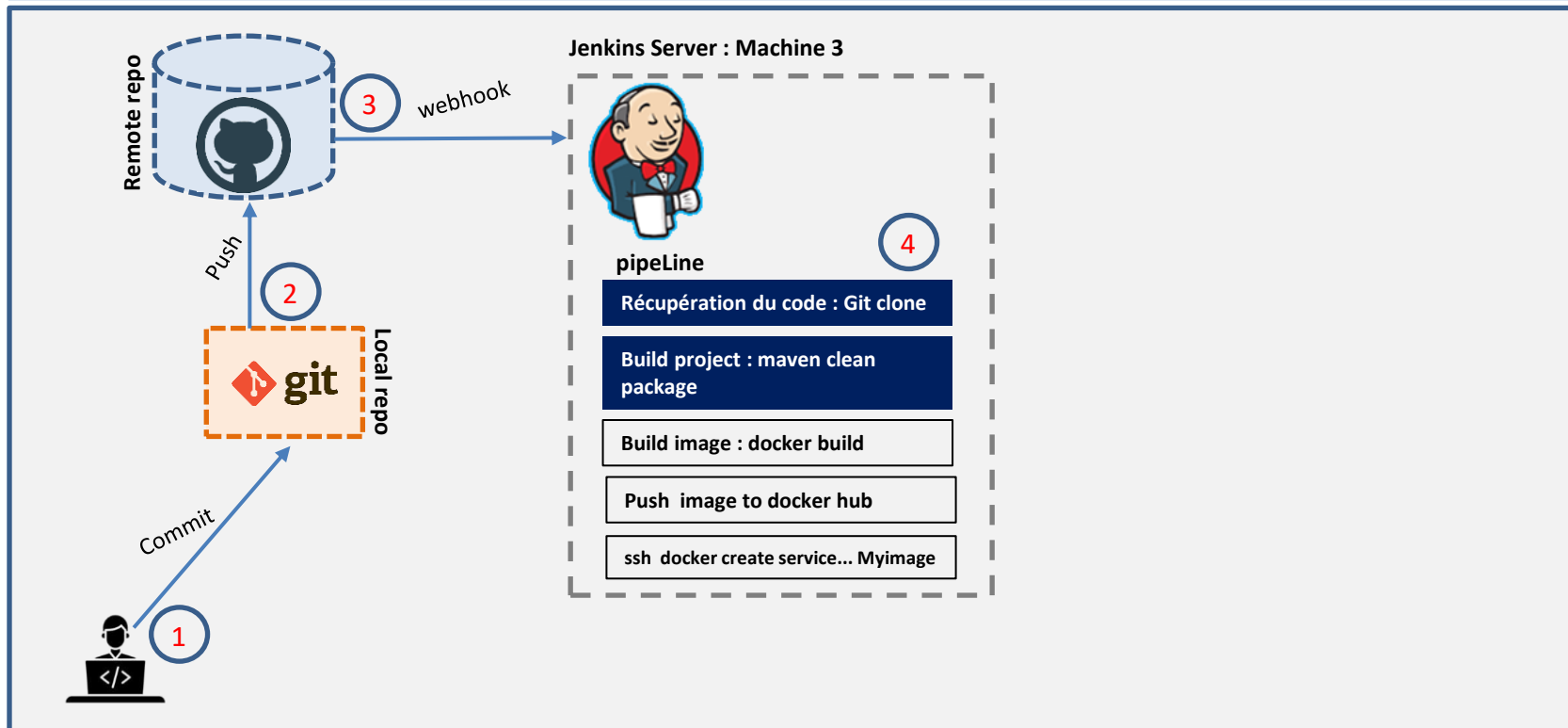
■ JENKINS-SWARM CI/CD

■ JENKINS DOCKER SWARM INTEGRATION



CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM



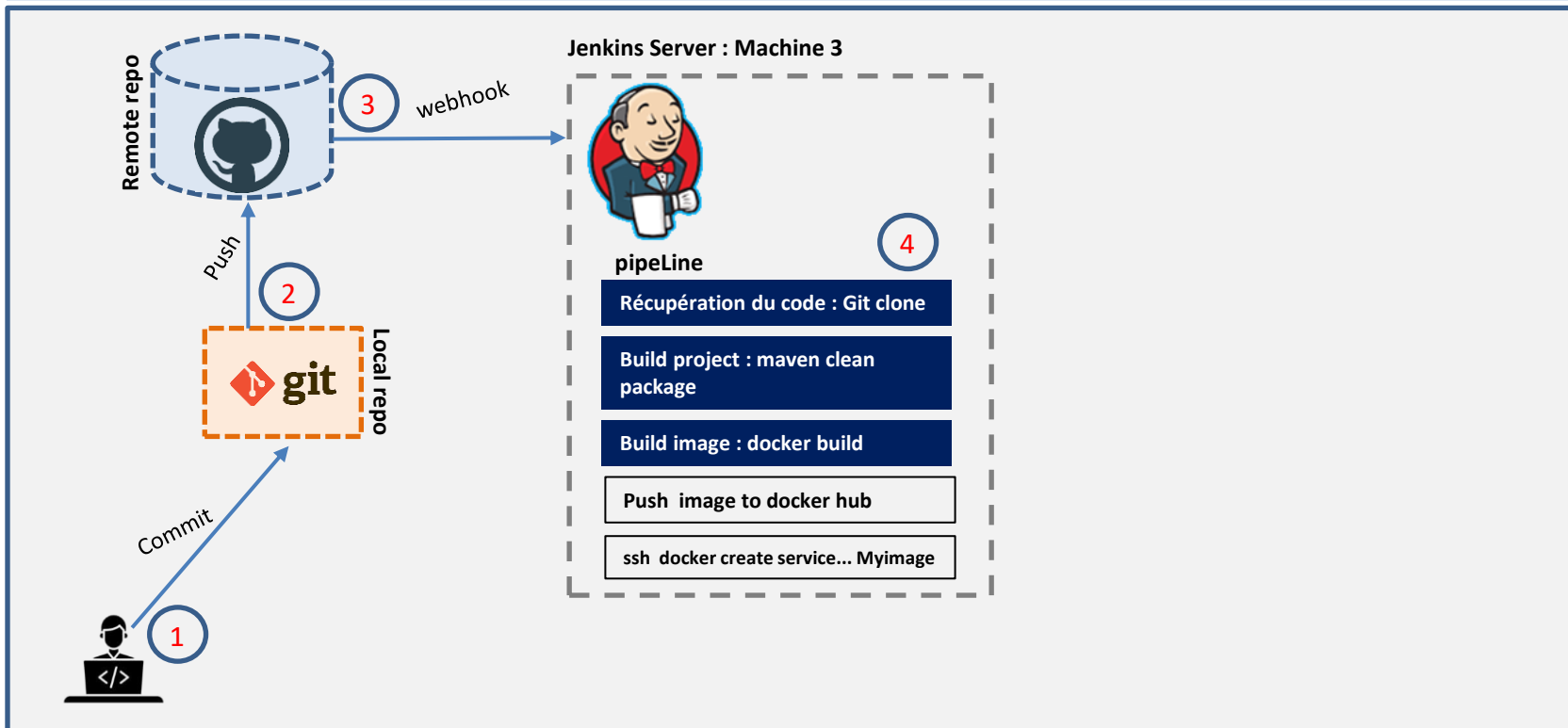
■ JENKINS-SWARM CI/CD

■ JENKINS DOCKER SWARM INTEGRATION



CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM



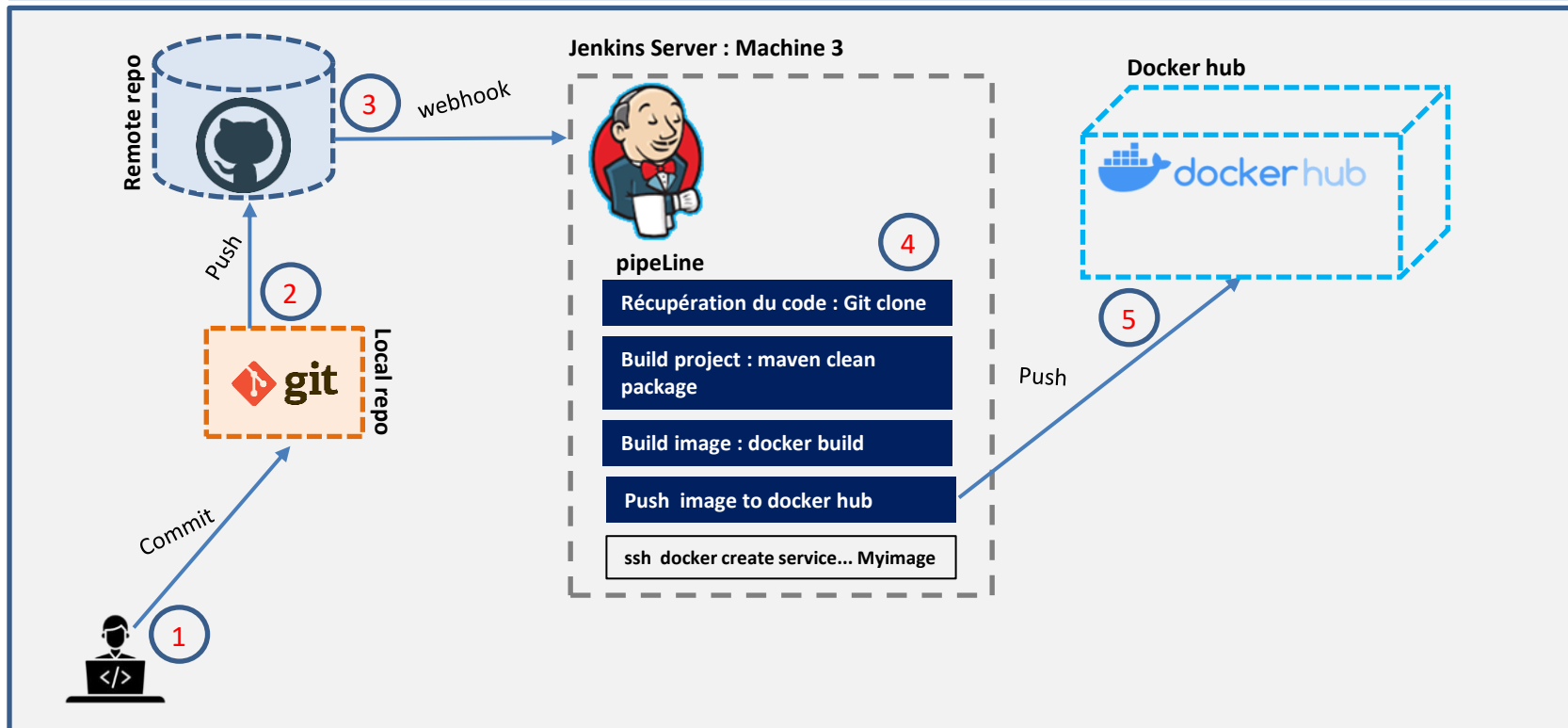
■ JENKINS-SWARM CI/CD

■ JENKINS DOCKER SWARM INTEGRATION



CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM



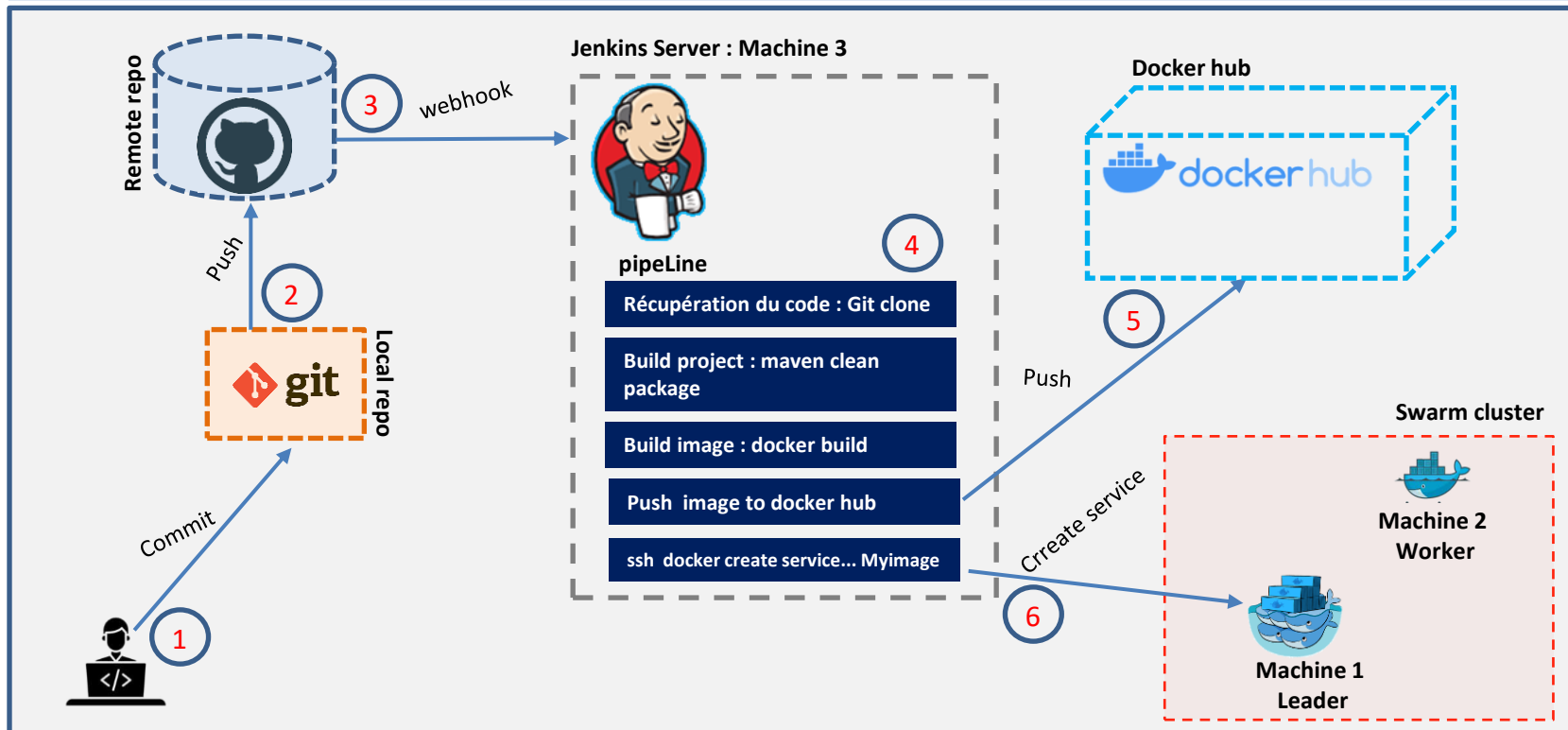
■ JENKINS-SWARM CI/CD

■ JENKINS DOCKER SWARM INTEGRATION



CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM



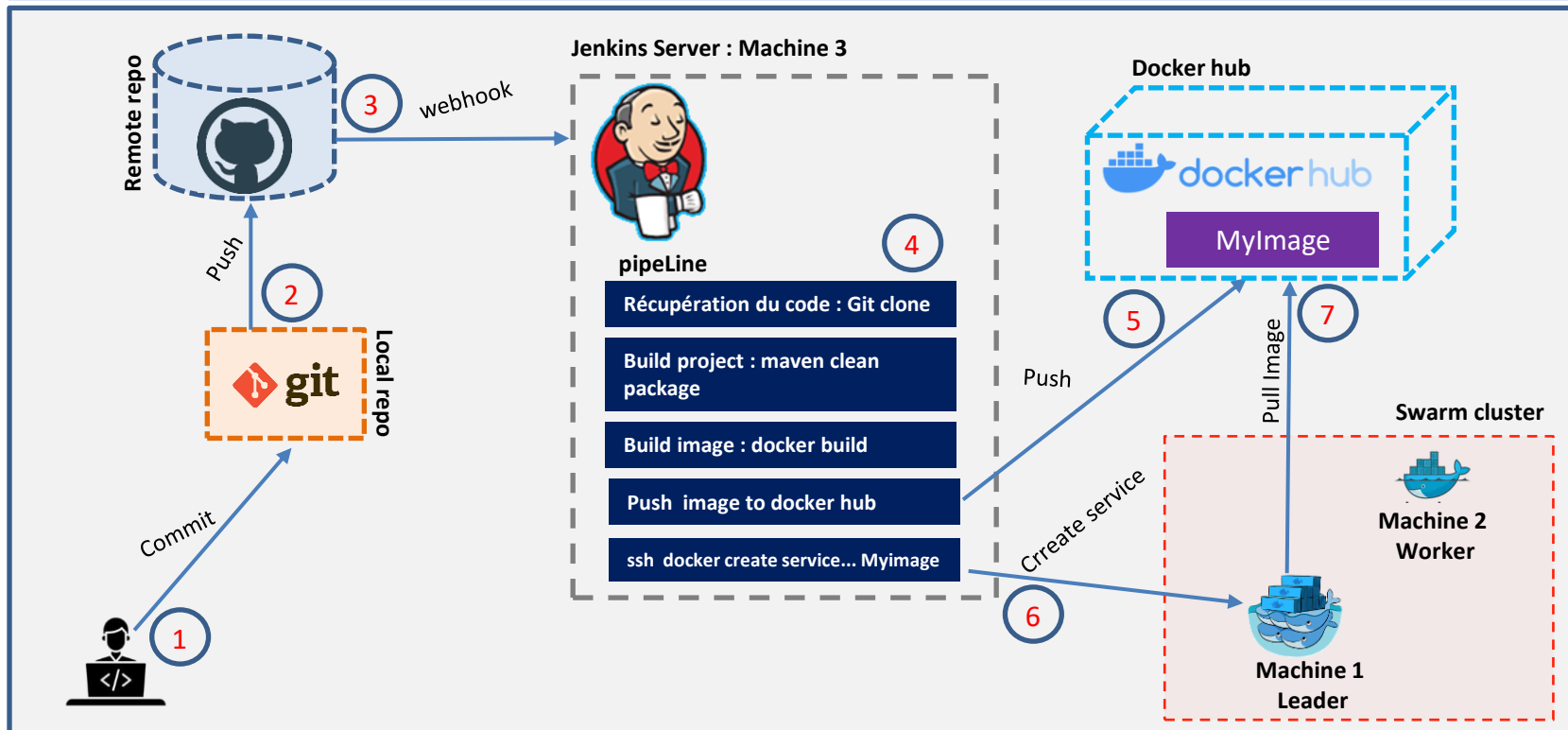
■ JENKINS-SWARM CI/CD

■ JENKINS DOCKER SWARM INTEGRATION



CI/CD PIPELINE SCRIPT

TO BUILD AND DEPLOY A SINGLE SERVICE IN DOCKER SWARM



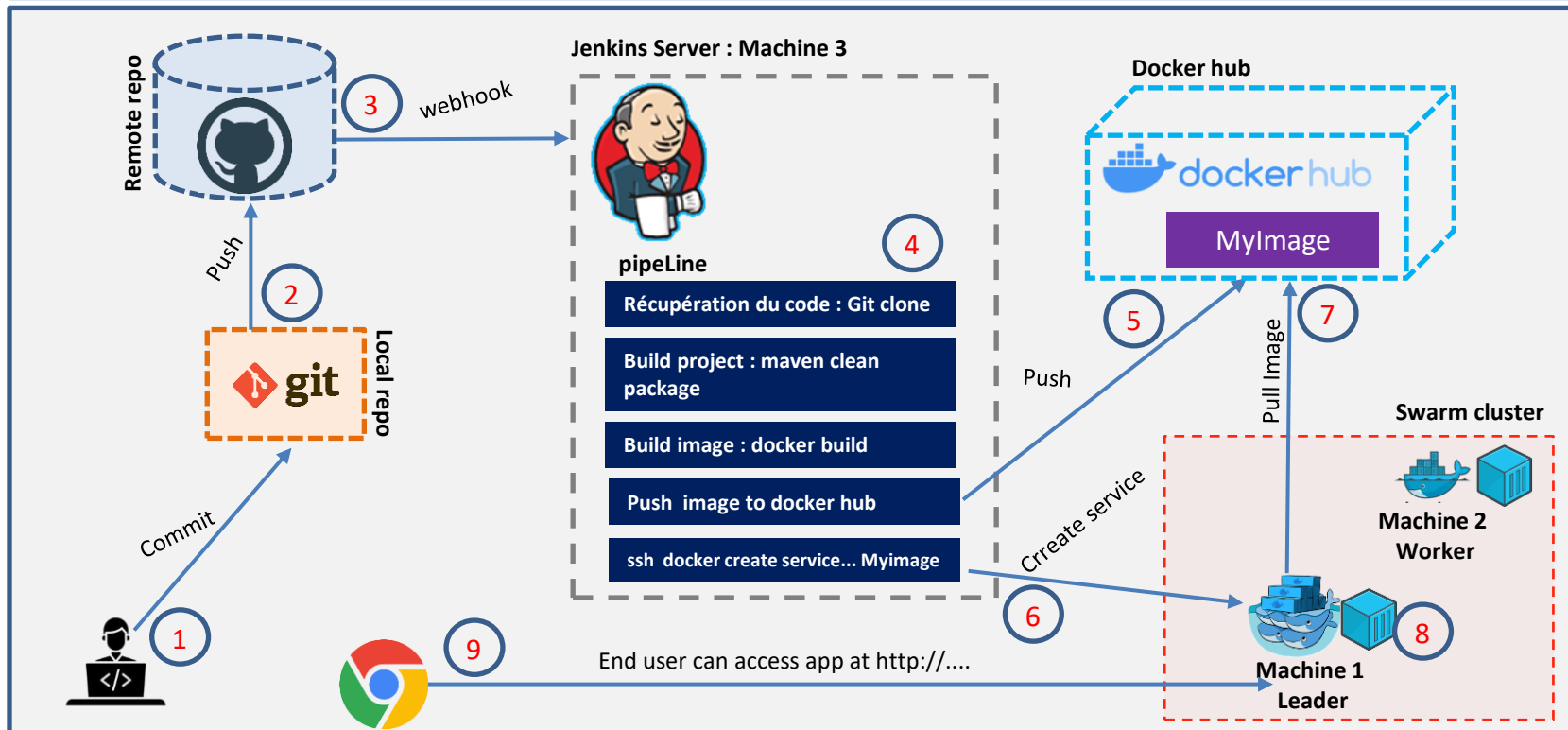
■ JENKINS-SWARM CI/CD

■ 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)
 - swarm1 (**leader**), swarm2 (**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

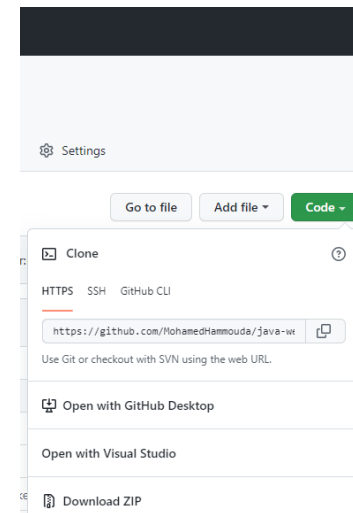
```
Git CMD
G:\>git clone https://github.com/MohamedHammouda/java-web-app-docker.git
Cloning into 'java-web-app-docker'...
remote: Enumerating objects: 179, done.
remote: Counting objects: 100% (49/49), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 179 (delta 33), reused 26 (delta 25), pack-reused 130 eceiving objects: 71% (128/179), 5.71 MiB | 311.00
KiB/s
Receiving objects: 100% (179/179), 5.84 MiB | 484.00 KiB/s, done.
Resolving deltas: 100% (47/47), done.

G:\>cd java-web-app-docker

G:\java-web-app-docker>dir
Le volume dans le lecteur G s'appelle WinData
Le numéro de série du volume est A816-87BD

Répertoire de G:\java-web-app-docker

10/05/2022 12:42 <DIR> .
10/05/2022 12:42 <DIR> ..
10/05/2022 12:42 359 .gitignore
10/05/2022 12:42 418 deployment.yml
10/05/2022 12:42 120 Dockerfile
10/05/2022 12:42 <DIR> helmchart
10/05/2022 12:42 486 index.yml
10/05/2022 12:42 3 624 javawebapp-0.1.0.tgz
10/05/2022 12:42 619 javawebapp-deployment.yml
10/05/2022 12:42 1 320 Jenkinsfile
10/05/2022 12:42 1 341 JenkinsfileDockerSwarm
10/05/2022 12:42 1 471 JenkinsPipelineScriptDockerSwarm_Private_Repo
10/05/2022 12:42 3 641 pom.xml
10/05/2022 12:42 334 service.yml
10/05/2022 12:42 <DIR> src
10/05/2022 12:42 11 fichier(s) 13 733 octets
10/05/2022 12:42 4 Rép(s) 121 316 323 328 octets libres
```



■ 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 cloner 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-9qvh6p8pwqq73rypnkwnyqhI9 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

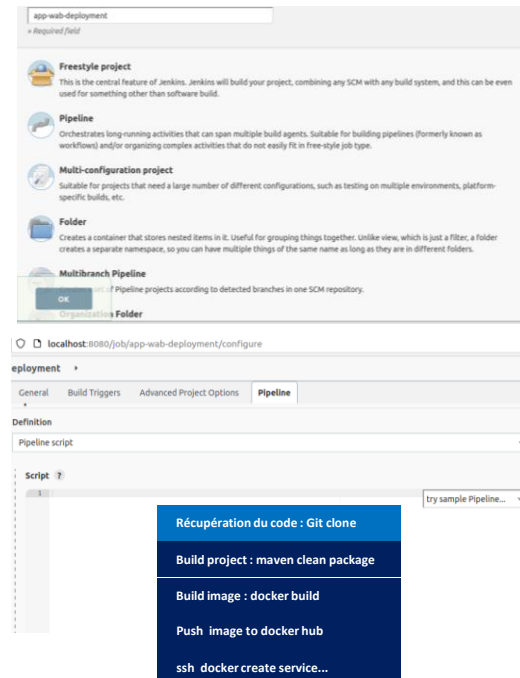
■ JENKINS DOCKER SWARM INTEGRATION



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

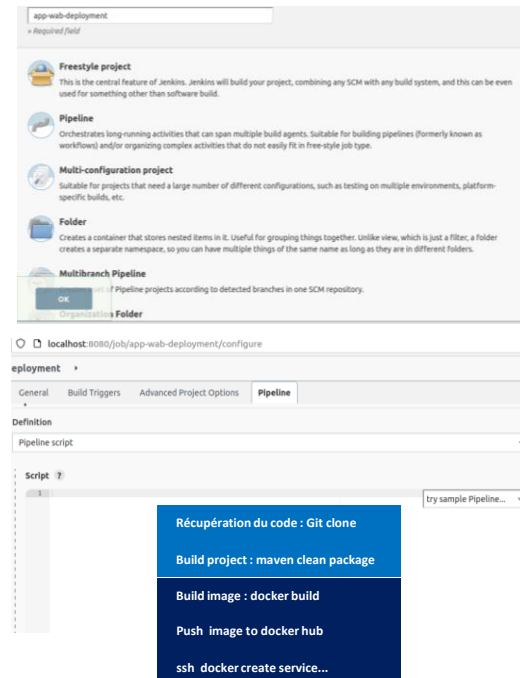
- Préparer le pipeline au niveau de jenkins server (machine3 jenkinsServer) par la Création d'un nouveau job pipeline  
*new item-->pipeline-->ok-->app-wab-deployment*
- Définir la pipeline du projet :

#### 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

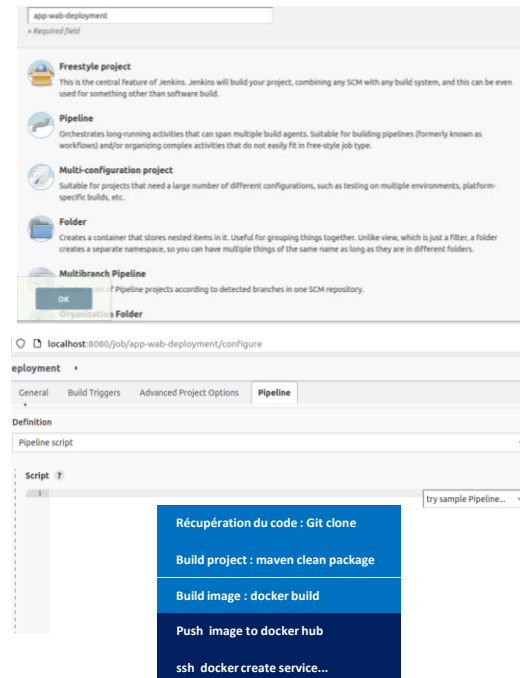
## ■ JENKINS DOCKER SWARM INTEGRATION



### 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

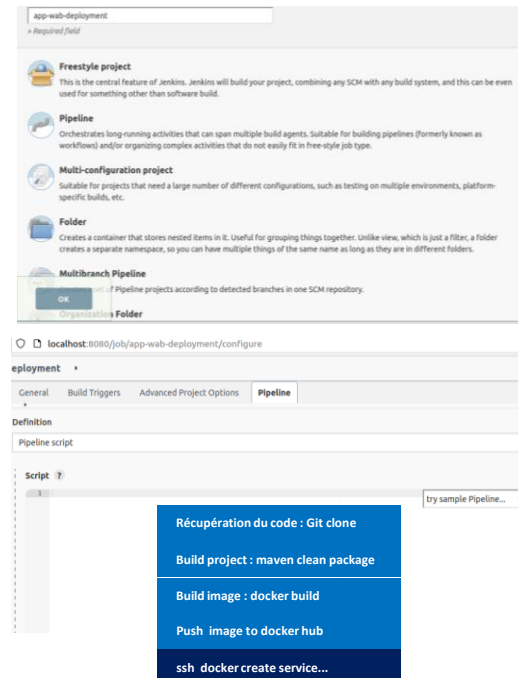
■ JENKINS DOCKER SWARM INTEGRATION



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

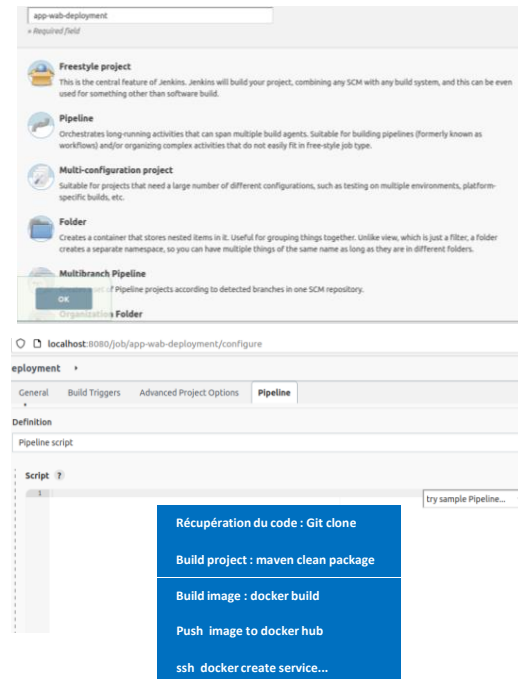
■ JENKINS DOCKER SWARM INTEGRATION



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

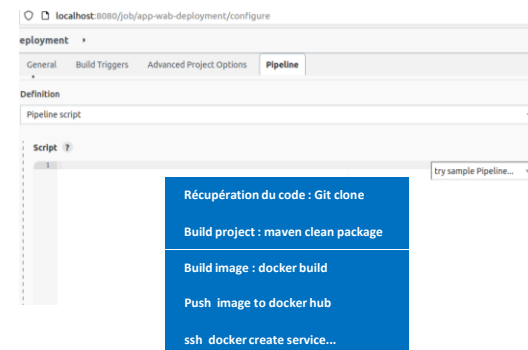
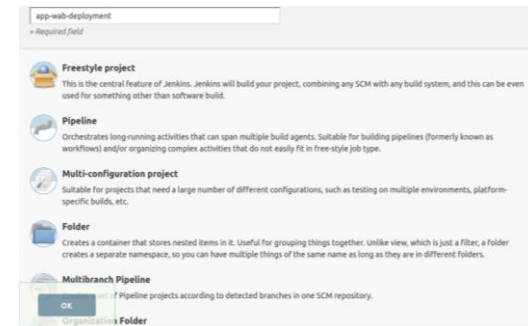
```
ls -al ~/.ssh/
drwx----- 2 mohamed mohamed 4096 May 9 00:06 .
drwxr-xr-x 16 mohamed mohamed 4096 May 8 14:40 ..
-rw----- 1 mohamed mohamed 411 May 8 23:59 id_ed25519
-rw-r--r-- 1 mohamed mohamed 103 May 8 23:59 id_ed25519.pub
```

- 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 fichier 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



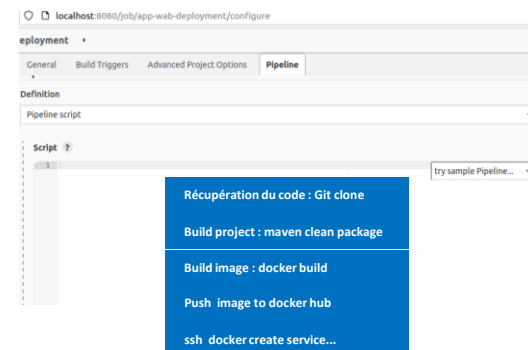
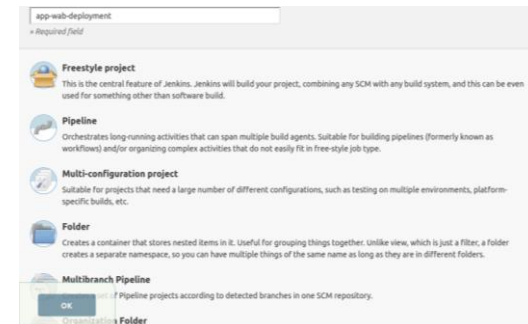
### CI/CD PIPELINE SCRIPT

### 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

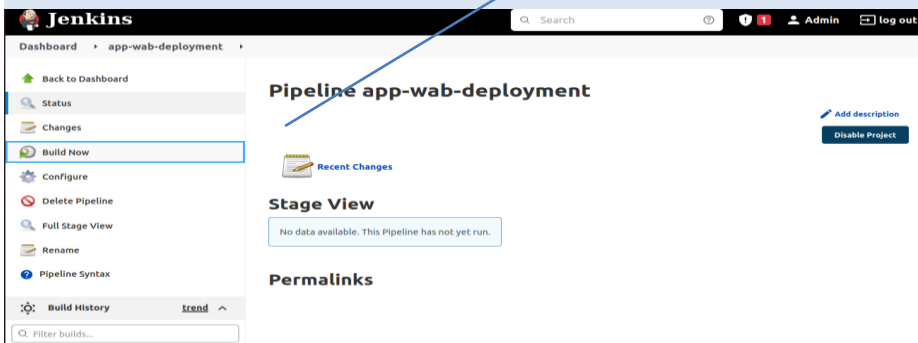
Etape 5 :

```
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 Build now  
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  
Pourquoi?? Consulter la console de la pipeline  
Pour moi docker daemon is not working



|                      | SCM Checkout | Maven Clean Package | Build Docker Image |
|----------------------|--------------|---------------------|--------------------|
| Average stage times: | 6s           | 25s                 | 2s                 |
| May 10 07:23         | No Changes   | 17s                 | 25s                |
|                      |              |                     | 2s failed          |

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

|                      | SCM Checkout | Maven Clean Package | Build Docker Image | Push Docker Image to Dev Server | Run Docker Image in Dev Server |
|----------------------|--------------|---------------------|--------------------|---------------------------------|--------------------------------|
| Average stage times: | 13s          | 25s                 | 4s                 | 2min 14s                        | 13s                            |
| May 10 07:23         | 13s          | 25s                 | 4s                 | 2min 14s                        | 13s                            |
| May 10 07:23         | 17s          | 25s                 | 4s                 | 2min 14s                        | 13s                            |