



# ÉCOLE MOHAMMADIA D'INGÉNIEURS

GÉNIE LOGICIEL / DEVOPS

TP N°2 : PIPELINE CI/CD MULTI-PROJETS — PARTIE  
TOMCAT (DIAPOS 1 À 6)

---

## Déploiement Spring Boot sur Tomcat avec GitLab CI/CD (Version 1.0)

---

*Élèves :*

Sami FAOUZI

*Encadré par :*

Asmaa RETBI

2ème année Génie Informatique

1<sup>er</sup> décembre 2025

## Table des matières

<b>1</b>	<b>Contexte et objectif (jusqu'à la diapo 6)</b>	<b>2</b>
<b>2</b>	<b>Prérequis : installation et démarrage de Tomcat (Diapos 1–2)</b>	<b>2</b>
2.1	Installation et structure du dossier Tomcat (capture attendue) . . . . .	2
2.2	Démarrage et vérification (captures attendues) . . . . .	3
<b>3</b>	<b>Configuration Spring Boot en WAR (Diapos 3.1–3.3 et 4)</b>	<b>5</b>
3.1	Modification du <code>pom.xml</code> : packaging <code>war</code> (Diapo 3.1) . . . . .	5
3.2	Ajout de la dépendance Tomcat en <code>provided</code> (Diapo 3.2) . . . . .	5
3.3	Création de <code>ServletInitializer</code> (Diapo 3.3) . . . . .	6
3.4	Génération du WAR et validation (Diapo 4) . . . . .	6
<b>4</b>	<b>Ajout du stage Deploy sur GitLab (Diapo 5)</b>	<b>7</b>
4.1	Principe . . . . .	7
4.2	Extrait du job <code>deploy-tomcat</code> (à ajouter en bas de <code>.gitlab-ci.yml</code> ) . . .	7
4.3	Point critique (runner) . . . . .	8
<b>5</b>	<b>Exécution du pipeline GitLab CI et test localhost (Diapo 6)</b>	<b>8</b>
5.1	Commit & Push (déclenchement pipeline) . . . . .	8
5.2	Capture GitLab attendue : pipeline SUCCESS . . . . .	8
5.3	Test navigateur attendu : endpoint déployé sur Tomcat . . . . .	8
<b>6</b>	<b>Conclusion (version 1.0)</b>	<b>9</b>

## 1 Contexte et objectif (jusqu'à la diapo 6)

L'objectif de cette première version est de mettre en place la chaîne minimale permettant :

- d'installer et démarrer Apache Tomcat localement ;
- d'adapter une application Spring Boot pour produire un fichier `.war` (déployable sur Tomcat) ;
- d'ajouter un **stage Deploy dans GitLab CI** et de lancer le pipeline ;
- de valider le déploiement par un test navigateur sur `localhost`.

## 2 Prérequis : installation et démarrage de Tomcat (Diapos 1–2)

### 2.1 Installation et structure du dossier Tomcat (capture attendue)

Tomcat a été installé localement. La capture suivante montre la **composition du dossier décompressé / installé**.

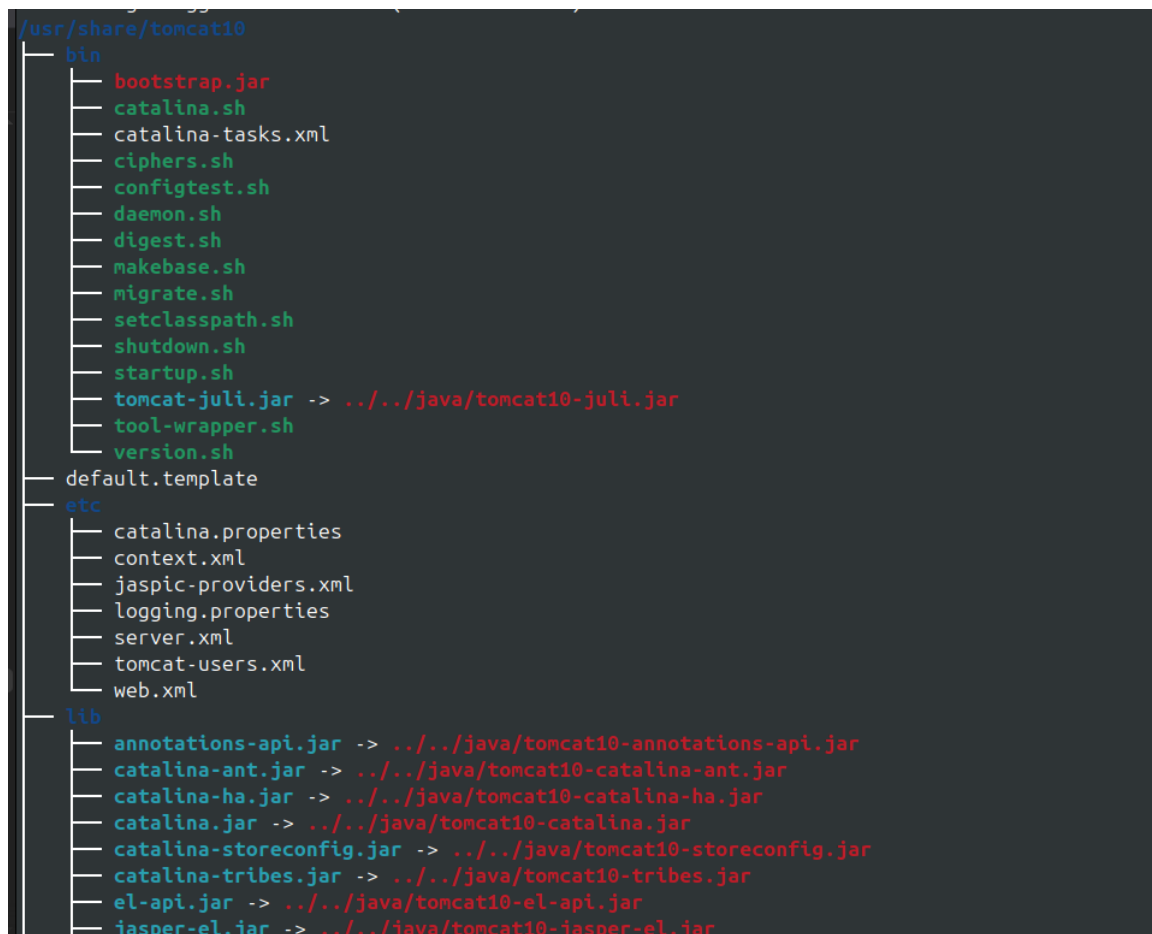


FIGURE 1 – Composition du dossier Tomcat (vue 1).

```

├── tomcat-i18n-ja.jar -> ../../java/tomcat10-i18n-ja.jar
├── tomcat-i18n-ru.jar -> ../../java/tomcat10-i18n-ru.jar
├── tomcat-jdbc.jar -> ../../java/tomcat10-jdbc.jar
├── tomcat-jni.jar -> ../../java/tomcat10-jni.jar
├── tomcat-util.jar -> ../../java/tomcat10-util.jar
├── tomcat-util-scan.jar -> ../../java/tomcat10-util-scan.jar
├── tomcat-websocket.jar -> ../../java/tomcat10-websocket.jar
├── websocket-api.jar -> ../../java/tomcat10-websocket-api.jar
├── websocket-client-api.jar -> ../../java/tomcat10-websocket-client-api.jar
└── logrotate.template

4 directories, 50 files
/etc/tomcat10
├── Catalina
│   └── localhost
├── catalina.properties
├── context.xml
├── jaspic-providers.xml
├── logging.properties
├── policy.d
│   ├── 01system.policy
│   ├── 02debian.policy
│   ├── 03catalina.policy
│   ├── 04webapps.policy
│   └── 50local.policy
├── server.xml
├── tomcat-users.xml
└── web.xml

4 directories, 12 files
/var/lib/tomcat10
├── conf -> /etc/tomcat10
├── lib
├── logs -> ../../log/tomcat10
├── policy
│   └── catalina.policy
├── webapps
│   └── ROOT
└── work -> ../../cache/tomcat10

3 directories, 1 file
sami@FAOUZI:~$

```

FIGURE 2 – Composition du dossier Tomcat (vue 2).

## 2.2 Démarrage et vérification (captures attendues)

Après installation, Tomcat est démarré en tant que service et son état est vérifié.

```
sami@FAOUZI:~$ systemctl status tomcat10
● tomcat10.service - Apache Tomcat 10 Web Application Server
   Loaded: loaded (/usr/lib/systemd/system/tomcat10.service; enabled; preset:~>
   Active: active (running) since Mon 2025-12-01 15:32:15 +01; 19s ago
     Docs: https://tomcat.apache.org/tomcat-10.0-doc/index.html
  Process: 8029 ExecStartPre=/usr/libexec/tomcat10/tomcat-update-policy.sh (c~>
   Main PID: 8033 (java)
    Tasks: 29 (limit: 18757)
  Memory: 246.6M (peak: 265.1M)
     CPU: 7.172s
    CGroup: /system.slice/tomcat10.service
            └─8033 /usr/lib/jvm/default-java/bin/java -Djava.util.logging.conf~>

Dec 01 15:32:16 FAOUZI tomcat10[8033]: At least one JAR was scanned for TLDs ye~>
Dec 01 15:32:16 FAOUZI tomcat10[8033]: Deployment of deployment descriptor [/et~>
Dec 01 15:32:16 FAOUZI tomcat10[8033]: Deploying deployment descriptor [/etc/to~>
Dec 01 15:32:16 FAOUZI tomcat10[8033]: The path attribute with value [/manager]~>
Dec 01 15:32:17 FAOUZI tomcat10[8033]: At least one JAR was scanned for TLDs ye~>
Dec 01 15:32:17 FAOUZI tomcat10[8033]: Deployment of deployment descriptor [/et~>
Dec 01 15:32:17 FAOUZI tomcat10[8033]: Deploying web application directory [/va~>
Dec 01 15:32:17 FAOUZI tomcat10[8033]: At least one JAR was scanned for TLDs ye~>
Dec 01 15:32:17 FAOUZI tomcat10[8033]: Deployment of web application directory ~>
Dec 01 15:32:17 FAOUZI tomcat10[8033]: Server startup in [1834] milliseconds
lines 1-22/22 (END)
```

FIGURE 3 – Vérification côté terminal : Tomcat en cours d'exécution (*service started / active*).

Ensuite, un accès navigateur à <http://localhost:8081> confirme le bon fonctionnement.

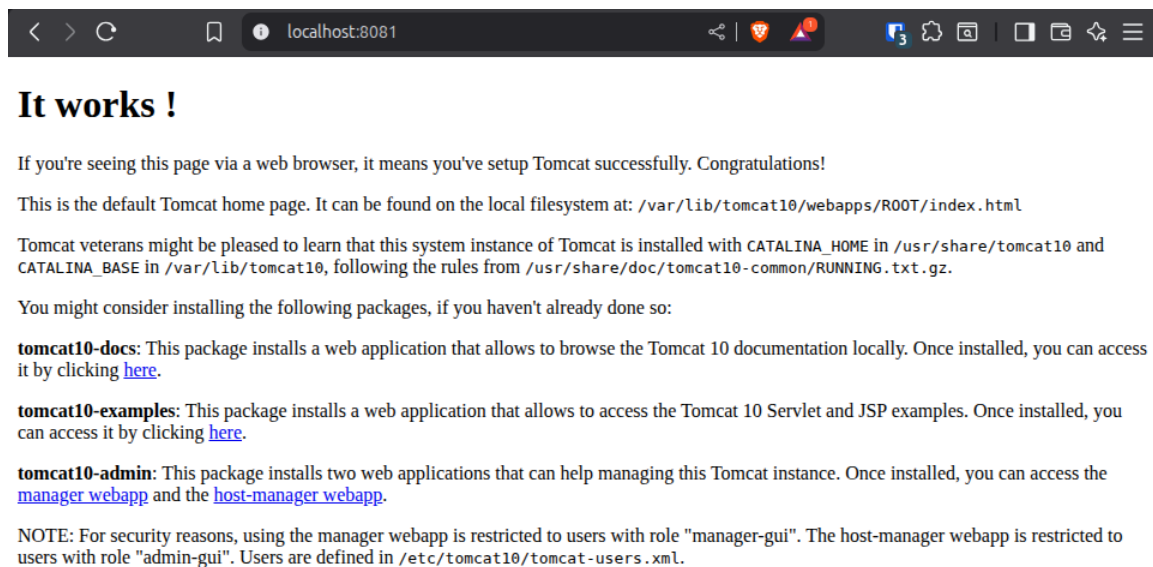
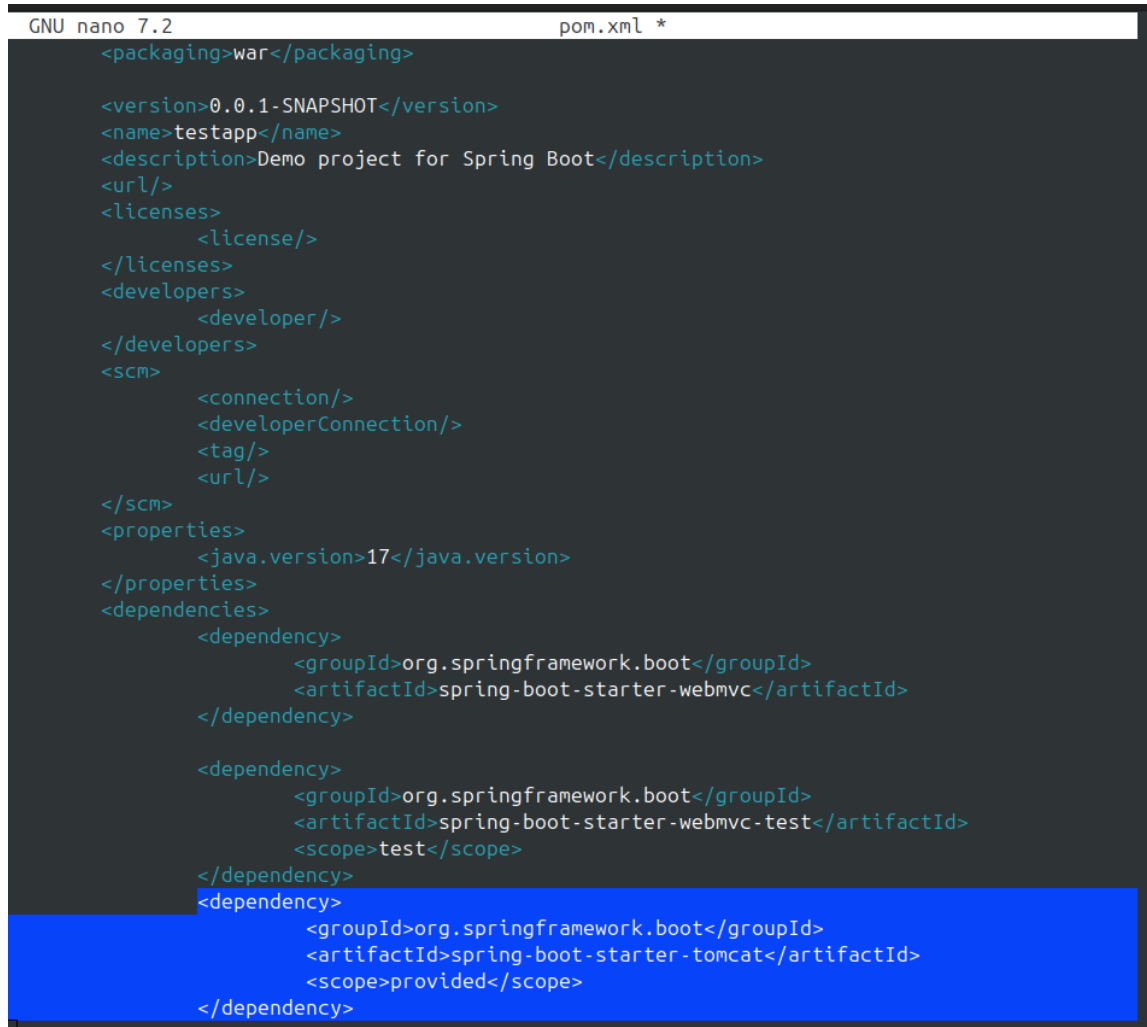


FIGURE 4 – Page d'accueil par défaut Tomcat ([localhost:8081](http://localhost:8081)).

### 3 Configuration Spring Boot en WAR (Diapos 3.1–3.3 et 4)

#### 3.1 Modification du `pom.xml` : `packaging war` (Diapo 3.1)

Le type de `packaging` a été modifié de `jar` vers `war` afin de produire un artefact déployable sur Tomcat.



```
GNU nano 7.2                                pom.xml *
<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>
<name>testapp</name>
<description>Demo project for Spring Boot</description>
<url/>
<licenses>
  <license/>
</licenses>
<developers>
  <developer/>
</developers>
<scm>
  <connection/>
  <developerConnection/>
  <tag/>
  <url/>
</scm>
<properties>
  <java.version>17</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webmvc</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webmvc-test</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

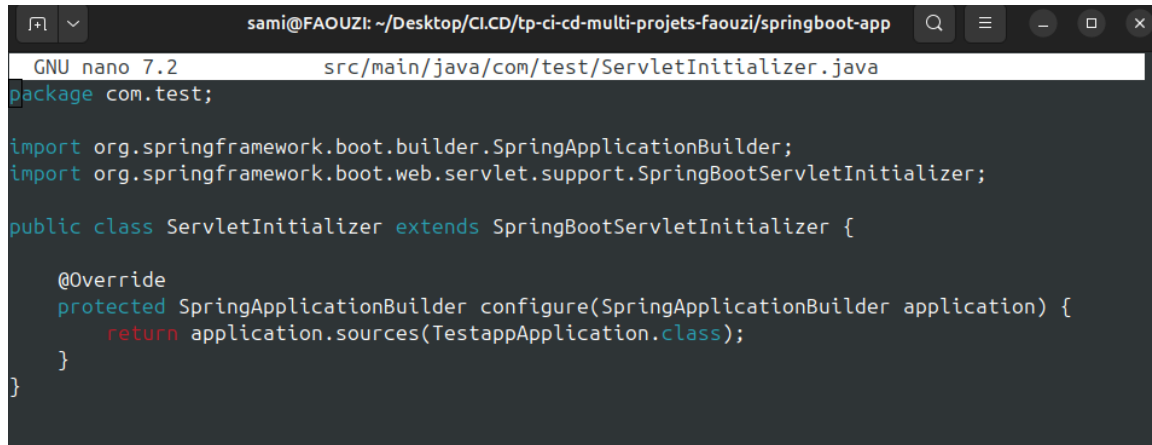
FIGURE 5 – Modification du `pom.xml` : `<packaging>war</packaging>` et dépendance Tomcat en `provided`.

#### 3.2 Ajout de la dépendance Tomcat en `provided` (Diapo 3.2)

Pour un déploiement sur un Tomcat externe, la dépendance `spring-boot-starter-tomcat` est déclarée avec le scope `provided`, car elle sera fournie par le serveur Tomcat au moment de l'exécution.

### 3.3 Création de `ServletInitializer` (Diapo 3.3)

Une classe `ServletInitializer` est ajoutée pour permettre à Spring Boot de s'initialiser correctement dans un conteneur Servlet (Tomcat).



```

GNU nano 7.2 src/main/java/com/test/ServletInitializer.java
package com.test;

import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(TestappApplication.class);
    }

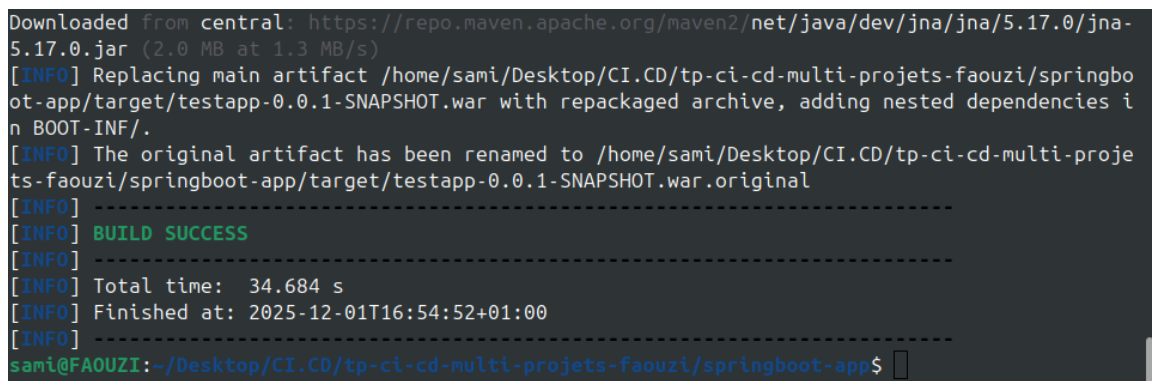
}

```

FIGURE 6 – Création de la classe `ServletInitializer` étendant `SpringBootServletInitializer`.

### 3.4 Génération du WAR et validation (Diapo 4)

La compilation Maven est exécutée avec `-DskipTests`. Deux captures sont attendues : **BUILD SUCCESS** et présence du fichier `.war`.



```

Downloaded from central: https://repo.maven.apache.org/maven2/net/java/dev/jna/jna/5.17.0/jna-5.17.0.jar (2.0 MB at 1.3 MB/s)
[INFO] Replacing main artifact /home/sami/Desktop/CI.CD/tp-ci-cd-multi-projets-faouzi/springboot-app/target/testapp-0.0.1-SNAPSHOT.war with repackaged archive, adding nested dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to /home/sami/Desktop/CI.CD/tp-ci-cd-multi-projets-faouzi/springboot-app/target/testapp-0.0.1-SNAPSHOT.war.original
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 34.684 s
[INFO] Finished at: 2025-12-01T16:54:52+01:00
[INFO] -----
sami@FAOUZI:~/Desktop/CI.CD/tp-ci-cd-multi-projets-faouzi/springboot-app$

```

FIGURE 7 – Console Maven montrant **BUILD SUCCESS**.

```
sami@FAOUZI:~/Desktop/CI.CD/tp-ci-cd-multi-projets-faouzi/springboot-app$ ls -lh target/*.war
-rw-rw-r-- 1 sami sami 19M Dec  1 16:54 target/testapp-0.0.1-SNAPSHOT.war
sami@FAOUZI:~/Desktop/CI.CD/tp-ci-cd-multi-projets-faouzi/springboot-app$ ls -lh target
total 32M
drwxrwxr-x 3 sami sami 4.0K Dec  1 16:54 classes
drwxrwxr-x 3 sami sami 4.0K Dec  1 16:54 generated-sources
drwxrwxr-x 3 sami sami 4.0K Dec  1 16:54 generated-test-sources
drwxrwxr-x 2 sami sami 4.0K Dec  1 16:54 maven-archiver
drwxrwxr-x 3 sami sami 4.0K Dec  1 16:54 maven-status
drwxrwxr-x 4 sami sami 4.0K Dec  1 16:54 testapp-0.0.1-SNAPSHOT
-rw-rw-r-- 1 sami sami 19M Dec  1 16:54 testapp-0.0.1-SNAPSHOT.war
-rw-rw-r-- 1 sami sami 13M Dec  1 16:54 testapp-0.0.1-SNAPSHOT.war.original
drwxrwxr-x 3 sami sami 4.0K Dec  1 16:54 test-classes
sami@FAOUZI:~/Desktop/CI.CD/tp-ci-cd-multi-projets-faouzi/springboot-app$
```

FIGURE 8 – Vérification du fichier `.war` généré dans `target/`.

## 4 Ajout du stage Deploy sur GitLab (Diapo 5)

### 4.1 Principe

Le stage `deploy` est exécuté après le stage `build`. Dans notre cas, le déploiement cible **Tomcat local**, donc il nécessite un **runner de type Shell** sur la machine qui possède Tomcat (sinon, impossible d'accéder à `/var/lib/tomcat10` et `systemctl`).

### 4.2 Extrait du job `deploy-tomcat` (à ajouter en bas de `.gitlab-ci.yml`)

Ci-dessous, l'extrait correspondant à la diapo 5 (adapté Linux/Tomcat10 et runner local) :

```
stages:
  - build
  - test
  - deploy

build-springboot:
  stage: build
  image: maven:3.9.9-eclipse-temurin-21-alpine
  script:
    - cd springboot-app
    - mvn -B clean package -DskipTests
  artifacts:
    paths:
      - springboot-app/target/*.war

deploy-tomcat:
  stage: deploy
```



```
needs:
  - job: build-springboot
    artifacts: true
script:
  - echo "===== Début Deploy Tomcat ====="
  - ls -lh springboot-app/target/*.war
  - sudo cp springboot-app/target/*.war /var/lib/tomcat10/webapps/springboot-ci.war
  - sudo systemctl restart tomcat10
  - echo "===== Fin Deploy Tomcat ====="
only:
  - main
tags:
  - local-shell
```

### 4.3 Point critique (runner)

Pour que `deploy-tomcat` s'exécute, un **Project Runner** doit être enregistré sur GitLab avec le tag `local-shell` et un executor `shell`.

## 5 Exécution du pipeline GitLab CI et test localhost (Diapo 6)

### 5.1 Commit & Push (déclenchement pipeline)

Après modification du `.gitlab-ci.yml`, les commandes suivantes déclenchent le pipeline :

```
git add .
git commit -m "add deploy stage for tomcat"
git push
```

### 5.2 Capture GitLab attendue : pipeline SUCCESS

La capture attendue est : **CI/CD > Pipelines** montrant le pipeline **vert** (build/test/deploy).

**Capture à ajouter :** GitLab **CI/CD > Pipelines** avec pipeline **SUCCESS** (jobs verts).  
 Nom conseillé : `pipeline_success_gitlab.png`

FIGURE 9 – Pipeline GitLab CI/CD en succès (build/test/deploy).

### 5.3 Test navigateur attendu : endpoint déployé sur Tomcat

Après succès du deploy, l'application est testée sur le navigateur via Tomcat :

- URL : <http://localhost:8081/springboot-ci/hello>
- (si l'endpoint diffère, utiliser l'URL réellement implémentée)

**Capture à ajouter :** navigateur sur <http://localhost:8081/springboot-ci/hello>  
Nom conseillé : [test\\_hellolocalhost.png](#)

FIGURE 10 – Test navigateur confirmant le déploiement sur Tomcat.

## 6 Conclusion (version 1.0)

Cette version 1.0 couvre l'avancement jusqu'à la diapo 6 :

- Tomcat installé et opérationnel ([localhost:8081](#)) ;
- application Spring Boot configurée pour produire un [.war](#) (packaging et [ServletInitializer](#)) ;
- ajout du stage [deploy](#) dans GitLab et préparation de l'exécution du pipeline ;
- captures restantes à compléter : **pipeline GitLab SUCCESS** et **test /hello**.