Déploiement d'une application Spring Boot (Docker / Serveur)

29. Le Déploiement (Deployment) : Introduction

Après avoir développé, testé et sécurisé votre application, il est temps de la rendre accessible au monde entier (ou à votre équipe front-end). Contrairement aux anciennes applications Java qui nécessitaient l'installation complexe d'un serveur Tomcat externe, **Spring Boot simplifie radicalement le déploiement**.

Grâce au concept de "*Fat JAR*" (ou *Uber JAR*), Spring Boot empaquette votre code, toutes vos dépendances (bibliothèques), et le serveur web (Tomcat) dans un seul et unique fichier exécutable.

30. Étape 1 : Préparer le fichier final (.jar)

Pour compiler votre projet et générer ce fameux fichier, vous allez utiliser votre outil de build (généralement **Maven**). Ouvrez un terminal à la racine de votre projet (là où se trouve le fichier `pom.xml`) et exécutez la commande suivante :

```
# La commande 'clean' supprime les anciens builds.
# La commande 'package' compile le code, exécute les tests unitaires, et crée le .jar
mvn clean package

# Si vous souhaitez ignorer l'exécution des tests lors de la création du fichier (non recommandé en production) :
mvn clean package -DskipTests
```

Une fois le processus terminé avec le message **BUILD SUCCESS**, Maven aura créé un nouveau dossier nommé `target/`. À l'intérieur, vous trouverez votre fichier exécutable, par exemple : `mystery-backend-0.0.1-SNAPSHOT.jar`.

31. Étape 2 : Déploiement traditionnel sur un serveur (VPS)

Si vous louez un serveur virtuel (VPS chez DigitalOcean, AWS EC2, ou OVH) fonctionnant sous Linux (ex: Ubuntu), le déploiement est très direct. Vous n'avez besoin que d'une seule chose d'installée sur le serveur : **Java (JRE 17)**.

1. Transférez votre fichier `.jar` sur le serveur (via un outil comme *SCP* ou *FileZilla*).
2. Connectez-vous à votre serveur via SSH.
3. Exécutez l'application avec la commande suivante :

```
# Exécution simple (bloque le terminal)
java -jar mystery-backend-0.0.1-SNAPSHOT.jar

# Exécution en arrière-plan (pour que l'API continue de tourner quand vous fermez le
terminal SSH)
nohup java -jar mystery-backend-0.0.1-SNAPSHOT.jar > application.log 2>&1 &
```

> **Bonne pratique pour la production :** Au lieu d'utiliser `nohup`, il est préférable de créer un **Service Systemd** sous Linux. Cela permet à votre application Spring Boot de redémarrer automatiquement si le serveur redémarre ou si l'application plante.

32. Étape 3 : Déploiement moderne avec Docker (La norme de l'industrie)

Aujourd'hui, la meilleure méthode pour déployer une application est de la **"Dockeriser"**. Docker permet de créer un *Conteneur*: une boîte hermétique qui contient votre fichier `.jar` ET la version exacte de Java nécessaire. Cela garantit que votre code fonctionnera de la même manière sur votre ordinateur, sur le serveur de test, et en production.

1. Créer le Dockerfile

À la racine de votre projet (au même niveau que le `pom.xml`), créez un fichier texte nommé exactement `Dockerfile` (sans extension) :

```
# 1. Utiliser une image de base légère contenant Java 17
FROM eclipse-temurin:17-jre-alpine

# 2. Définir le dossier de travail à l'intérieur du conteneur
WORKDIR /app

# 3. Copier le fichier .jar depuis votre dossier target vers le conteneur
# (Pensez à adapter le nom du fichier selon votre projet)
COPY target/mystery-backend-0.0.1-SNAPSHOT.jar app.jar

# 4. Exposer le port sur lequel Tomcat écoute
EXPOSE 8080

# 5. La commande à exécuter au démarrage du conteneur
ENTRYPOINT ["java", "-jar", "app.jar"]
```

2. Construire l'image Docker (Build)

Ouvrez votre terminal et exécutez la commande suivante pour construire l'image Docker (assurez-vous que le service Docker est lancé sur votre machine) :

```
# Le point '.' à la fin indique à Docker de chercher le Dockerfile dans le dossier courant
# '-t' permet de donner un nom (tag) à votre image
docker build -t vanishing-diplomat-api .
```

3. Exécuter le conteneur (Run)

Une fois l'image construite, vous pouvez la lancer n'importe où avec cette commande :

```
# -d : exécute le conteneur en arrière-plan (detached)
# -p 8080:8080 : redirige le port 8080 de votre machine vers le port 8080 du conteneur
# --name : donne un nom identifiable au conteneur en cours d'exécution
docker run -d -p 8080:8080 --name backend-investigation vanishing-diplomat-api
```

Gestion des variables d'environnement (Sécurité) :
En production, vous ne devez **jamais** écrire les mots de passe de vos bases de données ou vos clés secrètes (comme les clés de service Firebase ou les credentials Cloudinary) en dur dans le fichier `application.properties`.

Docker vous permet de passer ces informations de manière sécurisée au moment du lancement :
`docker run -d -p 8080:8080 -e SPRING_DATASOURCE_PASSWORD=MonMotDePasseSecret vanishing-diplomat-api`

Source : https://apcpedagogie.com/deploiement-application-spring-boot · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com