

# Cahier de TP

## « Exploitation des applications Spring »

### Outils utilisés :

- Bonne connexion Internet
- Système d'exploitation recommandé : Linux, MacOS, Windows 10
- JDK8, JDK11
- Editeur XML, .yaml : VSCode par exemple
- Apache JMeter pour solliciter les applications, VisualVM pour observer les processus Java
- PostgreSQL
- Jenkins
- Docker
- Infrastructure Kubernetes

## Atelier 1: Les applications SpringBoot

Cet atelier permet de découvrir 2 applications SpringBoot :

- Une avec le modèle classique de threads et l'utilisation de Tomcat, une base SQL et Java 8
- Une autre avec le modèle Réactif et l'utilisation de Netty, une base MongoDB Embarquée et Java 11

Visualiser la constitution des jars et en particulier le fichier ***META-INF/MANIFEST.MF***

### ***1.1. Démarrage et visualisation des Threads avec Tomcat***

Démarrer la première application

Ouvrir le script JMeter ***LoadDelivery.jmx***

Démarrer le script et observer les threads avec VisualVM

### ***1.2. Démarrage et visualisation des Threads avec Tomcat***

Démarrer la seconde application

Ouvrir le script JMeter ***LoadReactive.jmx***

Démarrer le script et observer les threads avec VisualVM

## Atelier 2: Outils de build

Récupérer les sources des 2 projets

Exécuter pour chaque projet les objectifs suivants :

- Packaging de l'application
- Exécution de l'application
- Génération des infos de build

Pour le projet Maven, générer également une image Docker. Quelle taille fait l'image construite ?

## Atelier 3 : Propriétés de configuration

### 3.1 Surcharge des propriétés de configuration

Dans les sources ou dans le jar, visualiser les propriétés de configuration applicative.

Surcharger la propriété `server.port`

- Via la ligne de commandes
- Via une variable d'environnement

### 3.2 Profils de configuration

Quels sont les profils activés par défaut pour les 2 applications ?

Quels sont les profils définis ?

Modifier via la ligne de commande ou une variable d'environnement, les profils par défaut

Visualisez également la documentation de l'application `delivery-service`

### 3.3 Configuration des traces

Activer l'option **-debug** au démarrage

Modifier la configuration afin de générer un fichier de trace

Modifier le niveau de trace du logger `org.springframework.boot` à `DEBUG` sans l'option **-debug**

## Atelier 4. Déploiement

### 4.1 Mise en service

Modifier le fichier de build afin que l'exécutable généré puisse être mis en service.

Via un fichier de configuration, customiser les variables d'environnement suivantes :

- `JAVA_HOME`, `JAVA_OPTS`
- `RUN_AS_USER`
- Les profils Spring
- Les crédits JDBC

## **4.2 Dockerfile**

Reprendre le fichier Dockerfile présent dans le projet delivery-service et l'améliorer en séparant les couches des librairies Spring, des classes applicatives et en utilisant un utilisateur dédié

## **4.3 Pipeline**

Visualisez le fichier Jenkinsfile, le comprendre et l'exécuter sur une plateforme Jenkins

# **Atelier 5 : Exploitation**

## ***5.1 Configuration de production***

Configuration des traces au format JSON

Désactiver JMX

## ***5.2 Mise en place d'actuator***

Configurer actuator pour visualiser :

- Les informations de santé
- Les informations de l'application
- Les métriques
- Les traces HTTP

Modifier dynamiquement le niveau des logs

## ***5.3 Kubernetes***

Démarrer un cluster Kubernetes

Visualiser les fichiers Kubernetes de delivery-service, les améliorer en fournissant des URLs de probe Kubernetes.

Mettre à jour les config map, le service postgres,

# **Atelier 6 : Spring Cloud avec OAuth2**

Objectifs :

- Architecture où chaque micro-service peut préciser des ACLs par rapport à l'identité du client (scope) et l'utilisateur final
- Utilisation de JWT avec un secret partagé pour limiter le trafic réseau

**Fusionner avec account-service**

## ***1. Mise en place du serveur d'autorisation***

- Reprendre les sources fournis

- Visualiser la configuration
- Effectuer des requêtes pour obtenir un jeton :
  - En tant que browser
  - En tant que service

## **2. Configuration du serveur Gateway**

Ajouter le starter ***cloud-oAuth2***

Reprendre la configuration du serveur de ressource, la visualiser

Configurer les routes de manière adéquate

Utiliser le script *jMeter oAuth2Authenticated* pour valider votre configuration

## **3. ACLs sur members-service**

Configurer *accounts-service* comme *ResourceServer* de la même manière que proxy

Définir les ACLs suivantes :

- Pour accéder à la page home, il faut avoir le rôle ADMIN
- Pour toutes le requête le client oAuth2 doit avoir le scope trust
- Pour toutes les requêtes, l'utilisateur doit avoir le rôle USER

Utiliser le script *jMeter oAuth2WithACLs* pour valider votre configuration