EXTERNALISEZ VOTRE CONFIGURATION SPRING BOOT AVEC KUBERNETES

Philémon Globléhi
Développeur Back-end Java

Le problème

Vous avez des configurations qui changent entre vos environnements (dev, prod) ?
Les développeurs Java stockent souvent ces infos en dur dans le code ou les fichiers properties. Risque :

- Fuites de données sensibles (mots de passe, clés API).
- Deploiements cassés si la config change.
- 1000 versions du même fichier application.properties

La solution Kubernetes

1- ConfigMaps :

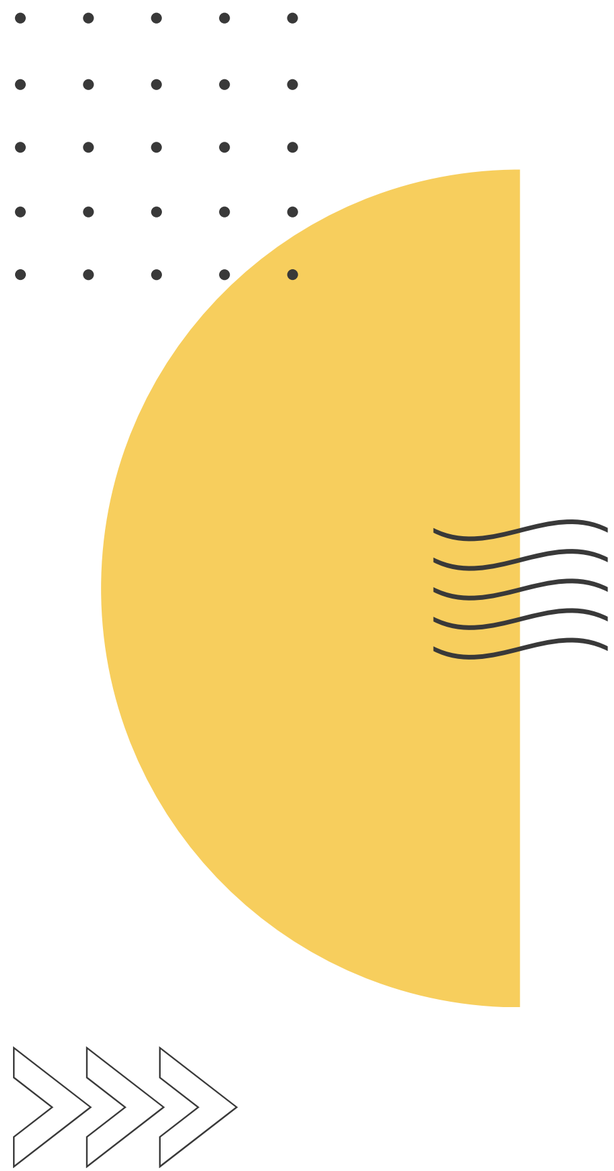
- Stockez vos configurations non sensibles (URLs de services, feature flags).
- Exemple pour Spring Boot :

```yaml
# configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: java-app-config
data:
  application.properties: |
    app.service.url=https://api.prod.com
    app.feature.tag=production
    # ... autres clés non sensibles

```

2- Secrets :

- Pour les données sensibles (mots de passe, tokens).
- Attention : Encodés en base64, pas chiffrés ! Utilisez des outils comme Vault pour plus de sécurité.

```yaml
# secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: db-credentials
type: Opaque
data:
  # Encodé en base64
  username: cHJvZC11c2Vy
  password: cHJvZC1wYXNzd29yZA==

```



Comment les utiliser dans Spring Boot ?

- Montez le ConfigMap/Secret comme volume dans le pod :

```yaml
# deployment.yaml
spec:
  containers:
    - name: java-app
      volumeMounts:
        - name: app-config
          mountPath: "/config"  # Monte le ConfigMap ici
          readOnly: true
      env:  # Injecte les Secrets comme variables d'environnement
        - name: DB_USERNAME
          valueFrom:
            secretKeyRef:
              name: db-credentials
              key: username
        - name: DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: db-credentials
              key: password
  volumes:
    - name: app-config
      configMap:
        name: java-app-config

```

- Dans application.properties, référencez les variables :

```properties
# Utilise les valeurs du ConfigMap (monté dans /config)
app.service.url=${APP_SERVICE_URL}
app.feature.tag=${APP_FEATURE_TAG}

# Utilise les Secrets injectés via variables d'environnement
spring.datasource.username=${DB_USERNAME}
spring.datasource.password=${DB_PASSWORD}
spring.datasource.url=jdbc:mysql://db-host:3306/mydb

```

Comment ça fonctionne ?

- ConfigMap :
  - Le fichier /config/application.properties est lu automatiquement par Spring Boot.
  - Pas besoin de recompiler si app.feature.tag change !
- Secret :
  - Les identifiants sont injectés via variables d'environnement (sécurisé).
  - Jamais stockés dans le code ou l'image Docker.

Pour les Secrets, préférez HashiCorp Vault ou Sealed Secrets (Kubernetes) pour un chiffrement renforcé.

Pourquoi c'est génial ?

- Zéro recompilation quand la config change.
- Séparation claire entre code et configuration.
- Compatible avec Spring Cloud Kubernetes pour du hot reload.

Documentation officielle :

- ConfigMaps: https://kubernetes.io/docs/concepts/configuration/configmap/
- Secrets: https://kubernetes.io/docs/concepts/configuration/secret/

Merci

Philémon Globléhi, Back-end Developer & DevOps Lover
Linkedin: https://www.linkedin.com/in/philemon-globlehi/