04 Konfiguration der Anwendung

Aufgabe

Ändere den Namen der Anwendung, indem du dafür sorgst, dass im Kubernetes-Pod eine Datei **/config/application.properties** erstellt wird, die einen neuen Wert für **spring.application.name** enthält.

Konfigurationsdatei erstellen

Erstelle eine Datei **application.properties** mit z.B. folgendem Inhalt

```
spring.application.name=NextGen Super TodoList App
```

Alternative1: Configmap über *kubectl create* erstellen

Erstelle nun eine Configmap mit dem Namen **backend-app-cm**, die die Datei **application.properties** enthält

```
kubectl create ?? --from-???
```

Alternative2: Configmap von Hand schreiben und in Kubernetes einspielen

Erstelle alternativ eine Datei **configmap.yaml**, anhand dieses Templates.

Die Configmap soll **backend-app-cm** heißen und eine Datei **application.properties** enthalten

Achtung: Die Einrückung des Dateiinhalts in der yaml ist wichtig

Beispielkonfiguration einer anderen Anwendung

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: CONFIGMAP_NAME
data:
  my.file: |
    Lorem ipsum dolor sit amet, consetetur sadipscing elitr

    Lorem ipsum dolor sit amet, consetetur sadipscing elitr
  2nd.file: |
    Franz jagt im komplett verwahrlosten Taxi quer durch Bayern
```

Diese Datei dann in Kubernetes einspielen

```
kubectl ???
```

Prüfung der Configmap

Prüfe nun ob die Configmap in Kubernetes sichtbar ist

```
kubectl get configmap

kubectl describe configmap backend-app-cm
```

Volume zum Kubernetes-Deployment hinzufügen

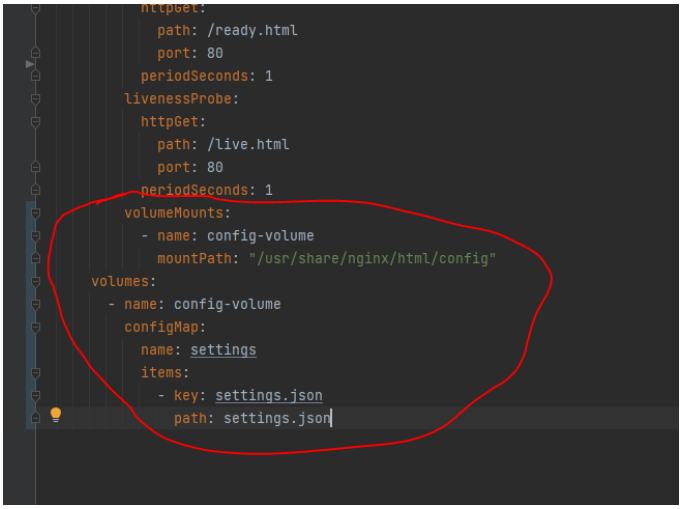
Füge nun der Datei **deployV2.yaml** ein Volume mit dem Namen **config-volume** hinzu, das mit dem Inhalt der Configmap befüllt wird.

Zur Erinnerung:

- Die Configmap heißt **backend-app-cm**
- Wir wollen die Datei **application.properties** aus der Configmap verwenden
- Die Datei soll im Container unter **/config/application.properties** gemountet werden

Beispielkonfiguration einer anderen Anwendung

```
apiVersion: apps/v1
kind: Deployment
...
spec:
  selector:
    matchLabels:
      app: demoapp
  template:
    ...
    spec:
      containers:
        - name: demoapp
          image: k8s-workshop/nginx-app:v2
          ...
          livenessProbe:
            httpGet:
              path: /live.html
              port: 80
            periodSeconds: 1
          volumeMounts:
            - name: config-volume
              mountPath: "/usr/share/nginx/html/config"
      volumes:
        - name: config-volume
          configMap:
            name: settings
            items:
              - key: settings.json
                path: settings.json
```



Änderungen an der Anwendung übernehmen

Übernehme nun die Änderungen an **deployV2.yaml** per kubectl

```
kubectl ??
```

Prüfen der Konfigurationsänderung

Prüfe über die Oberfläche der Anwendung, ob der neue Anwendungsname erscheint. Bedenke dabei, dass die Anwendung etwa 30Sekunden für den Neustart benötigt.

Ändern der Konfiguration

Ändere den Namen der Anwendung nochmals. Am Einfachsten geht das über den folgenden Befehl

```
KUBE_EDITOR="nano" kubectl edit configmap backend-app-cm
```

Die Änderungen kannst du mit **STRG+o** (und anschließend **ENTER**) speichern und den Editor mit **STRG+x** verlassen.

Prüfen der zweiten Konfigurationsänderung

Prüfe über die Oberfläche der Anwendung, ob die geänderte Konfiguration übernommen wurde.

Neustart der Anwendung

Starte die Anwendung neu. Das geht am einfachsten, indem du den Anwendungs-Pod löschst, so dass dieser neu erstellt wird

```
kubectl delete pod backend-server-xxxxx
```

Erst nach dem Neustart der Anwendung wird die neue Konfiguration angezeogen

Erweitern der Configmap

Neben dem Anwendungsnamen, den wir per application.properties-Datei an die Anwendung übergeben, wollen wir auch den Namen der verwendeten Datenbank über die Configmap steuerbar machen.

Editiere dazu unsere Configmap über folgenden Befehl

```
KUBE_EDITOR="nano" kubectl edit configmap backend-app-cm
```

Füge einen zweiten Eintrag in die Configmap ein, der den Schlüssel **DATABASE_NAME** und den Wert **workshopdb_dev** hat.

Die Änderungen kannst du mit **STRG+o** (und anschließend **ENTER**) speichern und den Editor mit **STRG+x** verlassen.

Erweiterte Übung (optional): Anpassen des Deployments

Passe nun die **deployV2.yaml** an, um den eben erstellten Konfigurationswert zu verwenden. Im Deployment gibt es bereits einen Environment-Eintrag für den Datenbank-Namen, dieser soll so angepasst werden, so dass der Wert **workshopdb_dev** nicht mehr hartcodiert ist. Der Wert soll aus dem eben erstellten Eintrag **DATABASE_NAME** der Configmap **backend-app-cm** ausgelesen werden.

Du kannst dich dafür an der deployment.yaml einer anderen Anwendung orientieren, die ihren **FOOTER** über eine configmap bekommt:

```
apiVersion: apps/v1
kind: Deployment
..
spec:
..
  template:
..
    spec:
      containers:
        - name: demoapp
          ...
           env:
             - name: SETTING_X
               value: "ABC"
             - name: FOOTER
               valueFrom:
                 configMapKeyRef:
                   name: settings
                   key: FOOTER
```

Änderungen an der Anwendung übernehmen

Übernehme nun die Änderungen an **deployV2.yaml** per kubectl

```
kubectl ??
```

Prüfe über folgenden Befehl, ob der Datenbank-Name nun aus der Configmap übernommen wird:

```
kubectl describe pod backend-server-xxxxx
```