06 Requests und Limits

Aufgabe

Passe die **deployV2.yaml** so an, dass das die Backend-Anwendung mit folgenden Parametern deployed wird:

- **2** Replicas
- Request
  - **250m** CPU
  - **64Mi** RAM
- Limit
  - **500m** CPU
  - **64Mi** RAM

Beispielkonfiguration einer anderen Anwendung

```
apiVersion: apps/v1
kind: Deployment
metadata:
...
spec:
  replicas: 5
...
    spec:
      containers:
        - name: demoapp
          image: k8s-workshop/nginx-app:v2
...
          livenessProbe:
            httpGet:
              path: /live.html
              port: 80
            periodSeconds: 1
          resources:
            requests:
              memory: "16Mi"
              cpu: "250m"
            limits:
              memory: "16Mi"
              cpu: "500m"
```

Einspielen der Konfiguration

```
kubectl apply -f deployV2.yaml
```

Prüfen der Konfigurationsänderung

Prüfe die Konfigurationsänderung mithilfe der Ausgabe von des noch laufenden *watch -n 1 kubectl get pods* . Können die Backend-Pod gestartet werden?

Debugging

Versuche über folgenden Befehl herauszufinden, warum die Anwendung nicht startet:

```
kubectl describe pod backend-server-XXXX
```

Interessant ist hier vor allem der **State** bzw. **Last State**.

Anpassen der Konfiguration

Passe die Konfiguration so an, dass die Anwendung wieder startet.

Hinweise:

- Spring ist leider relativ speicherhungrig. Du solltest also gleich "in die vollen gehen" und den Speicher so lange immer wieder verdoppeln, bis die Anwendung wieder erfolgreich startet. Passe dazu jeweils Request und Limit an.
- Je nach Leistung deines FAT-Client kann es sein, dass nicht nur der Arbeitsspeicher ein Problem ist. Beobachte daher genau den **State** bzw. **Last State** des Pod, ob weiterhin der Speicher ein Problem ist.
- Durch das neu eingeführte CPU-Limit könnte es sein, dass die Anwendung nun länger braucht als vorher um zu starten. Was musst du tun, damit Kubernetes länger auf den Start der Anwendung wartet.

Übertreibe es mit der Speicherreservierung und den Replicas

Um zu zeigen, dass bei Kubernetes nicht unbedingt der Grundsatz "Viel hilft viel" gilt, versuche folgendes zu konfigurieren:

- Request und Limit ändern auf
  - 3Gi RAM
- 10 Replicas

Wende die Konfiguration über *kubectl apply -f deployV2.yaml* an und beobachte wie sich die Pods im Cluster verhalten.

Prüfe ob du über folgenden Befehl weitere Details erfährst:

```
kubectl describe pod backend-server-XXXX
```

Sinnvolle Konfiguration wiederherstellen

Stelle nun die vorherige Konfiguration mit **2 Replicas, 256Mi** Memory-Request und **256Mi** Memory-Limit wieder her.