Parametrisierte Kubernetes Deployments mit kustomize

In Kubernetes wird mithilfe von YAML-Konfigurationen festgelegt, auf welche Weise Applikationen in der Infrastruktur bereitgestellt werden. Aber was, wenn wir unsere Applikation für mehrere verschiedene Umgebungen konfigurieren möchten? Für jede Umgebung eine einzelne YAML-Konfiguration zu erstellen ist oft mühselige Arbeit. Zum Glück existiert mit Kustomize ein Tool, welches genau diesen Arbeitsschritt automatisieren kann. Wie funktioniert Kustomize? Bevor wir uns mit dieser Frage beschäftigen, schauen wir uns erst einmal an einem Beispiel an, wie eine manuelle Konfiguration für verschiedene Umgebungen aussieht.

In unserem Szenario läuft auf unserer Dev-Umgebung bisher ein Service, welcher über das HTTP-Protokoll auf Port 80 erreichbar ist. Dieser Service soll nun auch auf der Testumgebung erreichbar sein, dort allerdings aus Sicherheitsgründen über HTTPS auf dem Port 443. Um dies umzusetzen, muss für jede Umgebung eine separate Konfiguration angelegt werden:

Konfiguration für die Dev-Umgebung:

```
kind: Service
apiVersion: v1
metadata:
  name: env-anzeige-frontend-https
spec:
  type: LoadBalancer
  selector:
    app: env-anzeige-frontend
  ports:
  - name: http
    port: 80
```

Konfiguration für die Testumgebung mit geänderten Porteinstellungen:

```
kind: Service
apiVersion: v1
metadata:
  name: env-anzeige-frontend-https
spec:
  type: LoadBalancer
  selector:
    app: env-anzeige-frontend
  ports:
  - name: https
    port: 443
```

Um den Service für die Testumgebung zu konfigurieren, wurde die Dev-Konfig kopiert und nur der relevante Datensatz – die Porteinstellungen – geändert. Und genau an diesem Punkt gehen die

Probleme los. Die YAMLs ständig manuell zu kopieren und anzupassen ist auf Dauer sehr zeitaufwändig. Außerdem entstehen durch das ständige „Hin-und-her-kopieren“ verschiedene Konfigurationen, welche größtenteils identisch sind. Dadurch ist im Nachhinein schnell nicht mehr klar, was denn nun der aktuelle Master-Stand ist.

Kustomize kann diese Probleme lösen, indem es die Anpassung bestehender Ressourcen vereinfacht und automatisiert. Dabei arbeitet Kustomize nach einem einfachen Grundkonzept: Es wird nur eine einzige Basiskonfiguration für die Bereitstellung der Anwendung angelegt. Für die einzelnen Umgebungen werden nur die an der Basis durchzuführenden Patches, die sogenannten Overlays, abgelegt.

Schauen wir uns einmal an, wie wir unser Beispiel mit den Porteinstellungen in Kustomize umsetzen können. Als Erstes muss eine *kustomization.yaml* angelegt werden, welche alle benötigten Ressourcen und die zu verwendenden Patches auflistet.

```
#base/kustomization.yaml
resources:
- service.yaml
```

```
#overlays/test/kustomization.yaml
bases:
- ../../base
patches:
- service-ports.yaml
```