Manifest-Dateien

- Grundgerüst Kubernetes Namespace YAML-Datei
- Manifest Dateien anwenden oder löschen
- Grundgerüst Kubernetes Deployment YAML-Datei
- Grundgerüst Kubernetes Service YAML-Datei
- Service Externer-Zugriff Typ
- Grundgerüst Kubernetes Ingress YAML-Datei

Grundgerüst Kubernetes Namespace YAML-Datei

```
apiVersion: v1
kind: Namespace
meta:
  name: einnamespace
  labels:
    # Maschinen auswertbar
    author: phillip
    name: app1
    <key>: <value>
  annotations:
    # Menschen auswertbar
    author: Phillip <mail@phillipunzen.de>
```

Manifest Dateien anwenden oder löschen

Um Manifest-Dateien in Kubernetes anzuwenden oder zu aktualisieren, verwenden wir den folgenden Befehl:

```
kubectl apply -f <Pfad zur Datei>
```

Um Manifest-Dateien aus dem Kubernetes Cluster zu entfernen, verwenden wir den folgenden Befehl:

```
kubectl delete -f <Pfad zur Datei>
```

Grundgerüst Kubernetes Deployment YAML-Datei

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: <deployment name>
  namespace: <namespace angeben>
  annotations:
    author: Phillip <mail@phillipunzen.de>

spec:
  replicas: 3
  strategy:
    type: <Typ> RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  selector:
    matchLabels:
      app: <app-name>
  template:
    metadata:
      labels:
        app: <app-name>
      annotations:
        author: Phillip <mail@phillipunzen.de>
    spec:
      containers:
      - name: <pod name>
        image: <docker-image:tag>
        env:
        - name: KEY
          value: VALUE
        ports:
```

```
    - name: http
      containerPort: <Port im Container>
    resources:
      requests: # Minimum an Hardware
        cpu: "250m" # => 1/4 CPU Kern für den Container
        memory: "256Mi"
      limits:
        cpu: "1000m" # => 1 CPU Kern für den Container
        memory: "512Mi" # => Wenn mehr benötigt, dann wird Container neugestartet
    readinessProbe: # Healthcheck auf HTTP
      httpGet:
        path: /
        port: http
      initialDelaySeconds: 10 # Zeitraum zwischen Checks
    livenessProbe:
      httpGet:
        path: /
        port: http
      initialDelaySeconds: 10 # Zeitraum zwischen Checks
```

Grundgerüst Kubernetes Service YAML-Datei

```
apiVersion: v1
kind: Service
metadata:
  name: <app-name>
  namespace: <namespace name>
  ...
spec:
  selector:
    #Selektor definieren
    app: <app-name>
  ports:
  - name: http
    port: 80
    targetPort: http (targetPort Namen aus dem Deployment)
  type: <Typ>
```

Service Externer-Zugriff Typ

ClusterIp	Innerhalb des Clusters
NodePort	Port wird auf einem Port der Nodes geöffnet
LoadBalancer	Verteilt die Anfragen auf die Pods (Liegt außerhalb des Kubernetes Clusters) => Kostet in der Cloud extra

Grundgerüst Kubernetes Ingress YAML-Datei

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: <Name>
  namespace: <Namespace>
spec:
  ingressClassName: <Name> nginx
  rules:
  - host: <dns name> website.domain.de
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: <app name>
            port:
              name: <Port Name> http
```