Übung: kubectl-CLI und Basis Ressourcen

Das `kubectl` -Kommando stellt, eine der Schaltzentralen des K8s Clusters zur Administration der Ressourcen dar.

In dieser Übung verwenden wir das `kubectl` -Kommando zur Erstellen eines Pods und Services.

Das passiert in einer eigenen Namespace um die Resultate gezielt Darstellen zu können:

In [1]:
```
! kubectl create namespace test
```

```
namespace/test created
```

Erzeugen eines Pod's, hier der Apache Web Server.

Die Option `--restart=Never` erzeugt nur einen Pod. Ansonsten wird ein Deployment erzeugt.

In [2]:
```
! kubectl run apache --image=httpd --restart=Never --namespace test
```

```
pod/apache created
```

Ausgabe der Erzeugten Ergebnisse und die YAML Datei welche den Pod beschreibt:

In [3]:
```
! kubectl get pods,services --namespace test
```

```
NAME         READY   STATUS              RESTARTS   AGE
pod/apache   0/1     ContainerCreating   0          0s
```

In [4]:
```
! kubectl get pod apache -o yaml --namespace test | head -19
```

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2019-05-14T14:07:29Z"
  labels:
    run: apache
  name: apache
  namespace: test
  resourceVersion: "856696"
  selfLink: /api/v1/namespaces/test/pods/apache
  uid: 9bdb68d6-7651-11e9-aaa5-026dcd796e19
spec:
  containers:
  - image: httpd
    imagePullPolicy: Always
    name: apache
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
```

Zu dem Pod `apache` Erzeugen wir einen Service. Dadurch wird der Web Server von aussen sichtbar.

Der Port 80 wird von Kubernetes automatisch auf den nächsten freien Port gemappt.

In [5]:
```
! kubectl expose pod/apache --type="LoadBalancer" --port 80 --namespace test
```

```
service/apache exposed
```

In [6]:
```
! kubectl get service apache -o yaml --namespace test | head -29
```

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2019-05-14T14:07:31Z"
  labels:
    run: apache
  name: apache
  namespace: test
  resourceVersion: "856704"
  selfLink: /api/v1/namespaces/test/services/apache
  uid: 9d1660ac-7651-11e9-aaa5-026dcd796e19
spec:
  clusterIP: 10.100.35.252
  externalTrafficPolicy: Cluster
  ports:
  - nodePort: 32303
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: apache
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer: {}
```

Wir sollten jetzt einen Pod und einen Service apache haben

In [7]:
```
! kubectl get pods,service apache --namespace test
```

```
NAME         READY   STATUS              RESTARTS   AGE
pod/apache   0/1     ContainerCreating   0          3s

NAME             TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/apache   LoadBalancer   10.100.35.252   <pending>     80:32303/TCP   1s
```

Da wir keinen LoadBalancer haben müssen wir mit einem kleinen Shellscript selber die IP des Clusters und der gemappte Port als URL aufbereiten.

Diese Shellscript ist im Script `startsvc` hinterlegt.

In [8]:
```
! kubectl config view -o=jsonpath='{ .clusters[0].cluster.server }' | sed -e 's/https:/http:/' -e "s/6443/$(kubectl
```

```
http://192.168.178.200:32303 (http://192.168.178.200:32303)
```

Zum Aufräumen genügt es den Namespace zu löschen

In [*]:
```
! kubectl delete namespace test
```

```
namespace "test" deleted
```

In [ ]:
```

```