

Übung: Volume und Claims

Zuerst brauchen wir ein `PersistentVolume` worauf die `Claims` zusteuern.

Im nachfolgenden Beispiel wird der `hostPath` als Volume zur Verfügung gestellt:

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: data-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/data"
```

Anschliessend folgen die `Claims` :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: data-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
```


Die Pod selber zeigen auf die Claims :

```
volumes:  
- name: web-storage  
  persistentVolumeClaim:  
    claimName: data-claim
```

Der Pod, muss zuerst aber noch, mittels `volumeMounts` seine Verzeichnisse freigeben:

```
volumeMounts:  
- mountPath: "/usr/local/apache2/htdocs"  
  subPath: htdocs  
  name: "web-storage"
```

Das komplette Beispiel eines Web Server welche seine Dateien auf dem Host ablegt sieht wie folgt aus:

In [1]:  ! cat 09-5-Volume/apache.yaml

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app.kubernetes.io/name: apache
  name: apache
spec:
  containers:
  - image: httpd
    name: apache
    # Volumes im Container
    volumeMounts:
    - mountPath: "/usr/local/apache2/htdocs"
      subPath: htdocs
      name: "web-storage"
  # Volumes in Host
  volumes:
  - name: web-storage
    persistentVolumeClaim:
      claimName: data-claim
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/name: apache
  name: apache
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app.kubernetes.io/name: apache
  type: LoadBalancer
```

```
In [2]: ! kubectl apply -f 09-5-Volume/apache.yaml
```

```
pod/apache created
```

```
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply  
service/apache configured
```

Um mehr als eine leere Seite anzuzeigen, muss jetzt im Verzeichnis `lernkube/data/htdocs` auf dem **lokalen Computer** eine Datei `index.html` angelegt werden. Verwendet dazu Notebook++ u.ä.

Der Inhalt könnte z.B. wie folgt sein:

```
<html>  
  <body>  
    <h1>Hallo Host</h1>  
  </body>  
</html>
```

```
In [3]: ! kubectl config view -o=jsonpath='{ .clusters[0].cluster.server }' | sed -e 's/https:/http:/' -e "s/6443/$(kubectl get svc -n kube-system | grep -o 'Kubernetes' | wc -l)"
```

```
http://192.168.178.200:32483 (http://192.168.178.200:32483)
```

Aufräumen.

Die Datei `lernkube/data/htdocs/index.html` bleibt erhalten, weil auf `PersistentVolume` gespeichert.

```
In [4]: ! kubectl delete -f 09-5-Volume/apache.yaml
```

```
pod "apache" deleted
```

```
service "apache" deleted
```

```
In [ ]: 
```

```
In [ ]: 
```

In []: ▶