

## Übung: Volume und mehrere Container in Pod

Diese Übung Demonstriert wie sich zwei Container innerhalb eines Pods das Verzeichnis `/usr/local/apache2/htdocs` teilen.

Der Container `apache` beinhaltet den Web Server und der Container `file-puller` schreibt alle 30 Sekunden die Datei `index.html` in das Verzeichnis `/usr/local/apache2/htdocs`.

Aus Einfachheitsgründen verwenden wir `emptyDir` als Volume.

### Erläuterungen `emptyDir`

Das `emptyDir`-Volume wird angelegt, wenn ein Pod einem Node zugewiesen wird. Alle Container in dem Pod auf diesem (Worker-)Node können dieses `emptyDir` (einfach ein leeres Verzeichnis) lesen und schreiben.

Der Pfad, mit dem das `emptyDir` innerhalb eines Containers eingehängt wird, kann sich innerhalb der Container des Pods unterscheiden.

Sobald ein Pod von einem Node gelöscht wird, wird auch der Inhalt des `emptyDir` komplett und unwiederbringlich gelöscht. Selbst wenn der gleiche Pod auf dem gleichen Worker Node neu erstellt wird, kann er nicht mehr auf das Volume seines Vorgängers zugreifen. Dies bezieht sich nicht auf den Crash eines Containers des Pods.

Typische Anwendungsfälle für `emptyDir`-Volumes könnten sein:


- Laufzeitdaten einer Applikation (Caches), die bei der Neuerstellung des Pods ohnehin neu erzeugt werden müssten
- Übergabe von (Laufzeit-)Konfigurationsdaten an alle Container innerhalb des Pods

Zuerst schauen wir den Inhalt der YAML Datei an und starten dann die Ressourcen

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

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app.kubernetes.io/name: web
  name: web
spec:
  containers:
    - image: httpd
      name: apache
      # Volumes im Container
      volumeMounts:
        - mountPath: "/usr/local/apache2/htdocs"
          name: "web-storage"
    - image: debian:jessie
      name: file-puller
      # Just spin & wait forever
      command: [ "/bin/bash", "-c", "--" ]
      args: [ "while true; do echo \"<html><body><h1>Hallo es ist $(date)</h1></body></html>\" >/usr/local/apache2/ht
docs/index.html; sleep 30; done;" ]
      # Volumes im Container
      volumeMounts:
        - mountPath: "/usr/local/apache2/htdocs"
          name: "web-storage"
      # Volumes in Host
      volumes:
        - name: "web-storage"
          emptyDir: {}
  ---
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/name: web
  name: web
spec:
  ports:
    - port: 80
      protocol: TCP
```

```
targetPort: 80
selector:
  app.kubernetes.io/name: web
type: LoadBalancer
```

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

```
pod/web created
service/web created
```

Um das Ergebnis anschauen, holen wir uns die IP-Adresse des Loadbalancer und des Services

In [3]:  ! kubectl config view -o=jsonpath='{ .clusters[0].cluster.server }' | sed -e 's/https:/http:/' -e "s/6443/\${kubectl get svc web -o jsonpath='{.spec.type}' | sed -e 's/LoadBalancer/6443/'}"

```
http://192.168.178.200:32139 (http://192.168.178.200:32139)
```

Und geben die Erzeugten Ressourcen aus:

```
In [*]: ! kubectl get pods,services
! echo "\n\nContainer"
! kubectl get pods -o=jsonpath='{range .items[*]}{"\n"}{.metadata.name}{"\t"}{range .spec.containers[*]}{.image}{"",
sort
```

NAME	READY	STATUS	RESTARTS	AGE
pod/apache-7f6c9f4bdd-9j9pf	1/1	Running	0	6d18h
pod/bpmn-backend-79d6d84ff6-rr4w9	1/1	Running	0	32h
pod/bpmn-frontend-6589fd84f7-7pq9z	1/1	Running	0	32h
pod/bpmn-frontend-6589fd84f7-bg789	1/1	Running	0	32h
pod/bpmn-frontend-6589fd84f7-dcf25	1/1	Running	0	32h
pod/bpmn-frontend-6589fd84f7-lnd7z	1/1	Running	0	32h
pod/bpmn-frontend-6589fd84f7-ms64b	1/1	Running	0	32h
pod/camunda-58cc4bd854-pkjzn	1/1	Running	0	32h
pod/debian	1/1	Running	0	6d18h
pod/dind-7999765885-p78j4	1/1	Running	0	6d18h
pod/hello-world-5b85b7d96d-86f2t	1/1	Running	0	6d18h
pod/hello-world-5b85b7d96d-cjcjw	1/1	Running	0	6d18h
pod/hello-world-5b85b7d96d-pdwhs	1/1	Running	0	6d18h
pod/iot-kafka-alert-f4d9d55d7-pthr2	1/1	Running	4	6d18h
pod/iot-kafka-consumer-68b9c4d667-xqtsp	1/1	Running	4	6d18h
pod/iot-kafka-pipe-5f64658ddc-s68cl	1/1	Running	3	6d18h
pod/jenkins-f6677cc74-klxp8	1/1	Running	0	4d7h
pod/jupyter-7bdc6f4699-c855g	1/1	Running	0	6d18h
pod/jupyter-base-79b756f98c-h486h	1/1	Running	0	6d18h
pod/kafka-56867866d7-5h527	1/1	Running	1	6d18h
pod/mlg-5f48ccb64f-k7bhc	1/1	Running	0	6d18h
pod/mosquitto-57d7d687c7-5p24t	1/1	Running	0	6d18h
pod/mqtt-kafka-bridge-69ffb7b995-6x7v8	1/1	Running	0	4d
pod/node-55c78d9b97-f9fq7	1/1	Running	0	6d18h
pod/nodered-75759d85db-b6vvk	1/1	Running	0	6d5h
pod/ubuntu	1/1	Running	0	6d18h
pod/web	0/2	ContainerCreating	0	2s
pod/zookeeper-5449767848-r4g2q	1/1	Running	0	6d18h

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/bpmn-backend	ClusterIP	10.111.68.175	<none>	8090/TCP	32h
service/bpmn-frontend	ClusterIP	10.98.130.72	<none>	80/TCP	32h
service/camunda	NodePort	10.96.171.79	<none>	8080:30686/TCP	32h
service/dind	ClusterIP	None	<none>	2375/TCP	6d18h

service/hello-world	LoadBalancer	10.111.17.152	<pending>	80:30173/TCP	6d18h
service/jenkins	LoadBalancer	10.102.202.107	<pending>	8080:32100/TCP	4d7h
service/jupyter	LoadBalancer	10.97.80.218	<pending>	8888:32288/TCP	6d18h
service/jupyter-base	LoadBalancer	10.96.133.82	<pending>	8888:32188/TCP	6d18h
service/kafka	ClusterIP	None	<none>	9092/TCP	6d18h
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	6d18h
service/mlg	NodePort	10.103.32.174	<none>	8888:32088/TCP	6d18h
service/mosquitto	NodePort	10.102.185.100	<none>	1883:31883/TCP	6d18h
service/node	LoadBalancer	10.102.229.111	<pending>	8080:31906/TCP	6d18h
service/nodered	NodePort	10.100.224.175	<none>	1880:32101/TCP	6d5h
service/web	LoadBalancer	10.98.135.173	<pending>	80:32139/TCP	2s
service/zookeeper	ClusterIP	None	<none>	2181/TCP	6d18h

## Container

```

apache-7f6c9f4bdd-9j9pf:      httpd,
bpmn-backend-79d6d84ff6-rr4w9: misegr/bpmn-backend:latest,
bpmn-frontend-6589fd84f7-7pq9z: misegr/bpmn-frontend:latest,
bpmn-frontend-6589fd84f7-bg789: misegr/bpmn-frontend:latest,
bpmn-frontend-6589fd84f7-dcf25: misegr/bpmn-frontend:latest,
bpmn-frontend-6589fd84f7-lnd7z: misegr/bpmn-frontend:latest,
bpmn-frontend-6589fd84f7-ms64b: misegr/bpmn-frontend:latest,
camunda-58cc4bd854-pkjzn:      camunda/camunda-bpm-platform,
debian: debian:jessie,
dind-7999765885-p78j4:  docker:stable-dind,
hello-world-5b85b7d96d-86f2t: dockercloud/hello-world,
hello-world-5b85b7d96d-cjcjw: dockercloud/hello-world,
hello-world-5b85b7d96d-pdwhs: dockercloud/hello-world,
iot-kafka-alert-f4d9d55d7-pthr2: misegr/iot-kafka-alert,
iot-kafka-consumer-68b9c4d667-xqtsp: misegr/iot-kafka-consumer,
iot-kafka-pipe-5f64658ddc-s68cl: misegr/iot-kafka-pipe,
jenkins-f6677cc74-klxp8: misegr/jenkins,
jupyter-7bdc6f4699-c855g: jupyter/tensorflow-notebook,
jupyter-base-79b756f98c-h486h: jupyter/base-notebook,
kafka-56867866d7-5h527: confluentinc/cp-kafka:latest,
mlg-5f48ccb64f-k7bhc: misegr/jupyter-mlg,
mosquitto-57d7d687c7-5p24t: eclipse-mosquitto,
mqtt-kafka-bridge-69ffb7b995-6x7v8: devicexx/mqtt-kafka-bridge,
node-55c78d9b97-f9fq7: node:8.12.0-alpine,
nodered-75759d85db-b6vvk: nodered/node-red-docker,
ubuntu: ubuntu:14.04,

```

```
web:    httpd, debian:jessie,  
zookeeper-5449767848-r4g2q:    confluentinc/cp-zookeeper:latest,
```

### Aufräumen

```
In [*]: ► ! kubectl delete -f 09-5-Volume/web.yaml
```

```
pod "web" deleted  
service "web" deleted
```

```
In [ ]: ►
```