Übung: Verteilung

In dieser Übung erstellen wir mehrere Pods ab dem gleichen Image mit jeweils einem ReplicaSet, Deployment und Service.

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

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

```
namespace/rs created
```

Erzeugen eines Deployments, hier der das Beispiel von Docker mit einem Web Server welche die aktuelle IP-Adresse ausgibt.

In [2]:
```
! kubectl run apache --image=dockercloud/hello-world --namespace rs
```

```
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/apache created
```

Ausgabe der Erzeugten Ergebnisse und die YAML Datei welche den Erzeugten Ressourcen beschreibt.

Ab `spec.containers` kommt erst der Pod.

In [3]:
```
! kubectl get pod,deployment,replicaset,service --namespace rs
```

```
NAME                              READY   STATUS              RESTARTS   AGE
pod/apache-549b689975-thnp4       0/1     ContainerCreating   0          0s

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/apache      0/1     1            0           0s

NAME                                        DESIRED   CURRENT   READY   AGE
replicaset.extensions/apache-549b689975     1         1         0       0s
```

In [4]:
```
! kubectl get deployment apache -o yaml --namespace rs | head -39
```

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: "2019-05-14T14:09:23Z"
  generation: 1
  labels:
    run: apache
  name: apache
  namespace: rs
  resourceVersion: "856939"
  selfLink: /apis/extensions/v1beta1/namespaces/rs/deployments/apache
  uid: dfee43bb-7651-11e9-aaa5-026dcd796e19
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      run: apache
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        run: apache
    spec:
      containers:
      - image: dockercloud/hello-world
        imagePullPolicy: Always
        name: apache
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
```

Um die ReplicaSet Funktionalität zu Demonstrieren, setzen wir die Anzahl der laufenden Pods auf 3 und schauen uns das Ergebnis an

In [5]:
```
! kubectl --namespace rs scale --replicas=3 deployment/apache
! kubectl get pod,deployment,replicaset,service --namespace rs
```

```
deployment.extensions/apache scaled
NAME                            READY   STATUS              RESTARTS   AGE
pod/apache-549b689975-2r8s7     0/1     ContainerCreating   0          1s
pod/apache-549b689975-ck4hj     0/1     ContainerCreating   0          1s
pod/apache-549b689975-thnp4     1/1     Running             0          3s

NAME                             READY   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/apache     1/3     3            1           3s

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.extensions/apache-549b689975    3         3         1       3s
```

Zu dem **Deployment** `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.

Das Ergebnis schauen wir uns wieder an.

In [6]:
```
! kubectl expose deployment/apache --type="LoadBalancer" --port 80 --namespace rs
! kubectl get pod,deployment,replicaset,service --namespace rs
```

```
service/apache exposed
NAME                             READY   STATUS              RESTARTS   AGE
pod/apache-549b689975-2r8s7      0/1     ContainerCreating   0          2s
pod/apache-549b689975-ck4hj      0/1     ContainerCreating   0          2s
pod/apache-549b689975-thnp4      1/1     Running             0          4s

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/apache      1/3     3            1           4s

NAME                                        DESIRED   CURRENT   READY   AGE
replicaset.extensions/apache-549b689975     3         3         1       4s

NAME             TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/apache   LoadBalancer   10.108.176.226   <pending>     80:31281/TCP   0s
```

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 [7]:
```
! kubectl config view -o=jsonpath='{ .clusters[0].cluster.server }' | sed -e 's/https:/http:/' -e "s/6443/$(kubectl
```

```
http://192.168.178.200:31281 (http://192.168.178.200:31281)
```

Wird der obige Link in mehreren Browsern geöffnet sollte jeweils ein anderer Hostname erscheinen.

Erklärung: Kubernetes verteilt die Anfragen auf die laufenden Pods.

Zum Aufräumen genügt es den Namespace zu löschen

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

```
namespace "rs" deleted
```

In [ ]: