Practical 3 Ex3

Dawid Skraba 19433692

1.

```
dawidskraba@dhcp-892b1ae8 ~ % minikube delete
🔥  Deleting "minikube" in docker ...
🔥  Deleting container "minikube" ...
🔥  Removing /Users/dawidskraba/.minikube/machines/minikube ...
💀  Removed all traces of the "minikube" cluster.
dawidskraba@dhcp-892b1ae8 ~ % minikube start --nodes 2
😄  minikube v1.27.0 on Darwin 12.3 (arm64)
❗  Kubernetes 1.25.0 has a known issue with resolv.conf.
rkaround that should work for most use cases.
❗  For more information, see: https://github.com/kubernet
12135
dawidskraba@dhcp-892b1ae8 ~ % kubectl get nodes
NAME           STATUS     ROLES           AGE   VERSION
minikube       Ready      control-plane   47s   v1.25.0
minikube-m02   NotReady   <none>          19s   v1.25.0
```

minikube-m02 e8d971c433c0	gcr.io/k8s-minikube/kicbase:v0	Running	61187,...	8 seconds agc	
minikube fb8ffda53757	gcr.io/k8s-minikube/kicbase:v0	Running	61155,...	26 seconds ag	

I deleted the cluster and created a new one with two nodes. You can also see that in the docker dashboard that a second container is spun up for the second worker node.

2.

Kind: This specifies the object we are creating. Here it's a Deployment. We could specify other other like a pod to create a pod/deployment to our exact needs.

Replicas: Tells Kubernetes how many instances of the same pod to maintain at any given time.

Replicas.strategy: This field specifies the strategy used to replace old pods by new ones. These can be 'RollingUpdate' and 'recreate'. Recreate forces all pods to be shut down when creating new pod and RollingUpdate just shuts down the old ones and creates a new one in its place.

Spec.template.spec.affinity: this field defines the rules for the scheduler to determine where a pod can be placed. This can be to a node using a label. This can be done by using NodeAffinity. antiAffinity is where we don't want our pod placed on the same node as a pod with the same label.

Spec.template.spec.containers: With this field we specify the image which will build our containers and be placed into pods. We also specify the ports and the grace period for which it waits before shutting down a pod after doing a health check.

3.

```
dawidskraba@Dawids-MacBook-Air-2 ex3 % kubectl apply -f .
deployment.apps/hello created
dawidskraba@Dawids-MacBook-Air-2 ex3 %
```

Created this deployment using the "hello-deployment.yaml"

```
dawidskraba@Dawids-MacBook-Air-2 ex3 % kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP          NODE           NOMINATED NODE   READINESS GATES
hello-7fb88bf7b6-gn6sx   1/1     Running   0          86s   10.244.0.3  minikube       <none>           <none>
hello-7fb88bf7b6-ppg6b   1/1     Running   0          86s   10.244.1.2  minikube-m02   <none>           <none>
dawidskraba@Dawids-MacBook-Air-2 ex3 %
```

We see that the pods are running on two different nodes.

a)

```
dawidskraba@Dawids-MacBook-Air-2 ex3 % kubectl delete deployment hello
deployment.apps "hello" deleted
dawidskraba@Dawids-MacBook-Air-2 ex3 % kubectl get pods
No resources found in default namespace.
```

Deleted deployment and made sure no other pods running.

b)

```
dawidskraba@Dawids-MacBook-Air-2 ex3 % kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
minikube       Ready    control-plane   48m   v1.25.0
minikube-m02   Ready    <none>          48m   v1.25.0
dawidskraba@Dawids-MacBook-Air-2 ex3 % kubectl label nodes minikube-m02 disktype=label1
node/minikube-m02 labeled
```

Here I looked at my nodes and decided to label my second node using the label "disktype=label1", this can be used to reference the node later.

c)

```
dawidskraba@Dawids-MacBook-Air-2 ex3 % kubectl get nodes --show-labels
NAME           STATUS   ROLES           AGE   VERSION   LABELS
minikube       Ready    control-plane   50m   v1.25.0   beta.kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,kubernetes.io/arch=arm64,kubernetes.
ux,minikube.k8s.io/commit=4243041b7a72319b9be7842a7d34b6767bbdac2b,minikube.k8s.io/name=minikube,minikube.k8s.io/primary=true,minikube.k8s.io/updated_
io/version=v1.27.0,node-role.kubernetes.io/control-plane=,node.kubernetes.io/exclude-from-external-load-balancers=
minikube-m02   Ready    <none>          50m   v1.25.0   beta.kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=a
kubernetes.io/os=linux
```

To confirm the change we use the above command and above the highlighted areas we can see that the second node is now labelled.

d)

```
dawidskraba@dhcp-892b1938 ex3 % kubectl apply -f ./hello-deployment_updated.yaml
deployment.apps/newhello created
dawidskraba@dhcp-892b1938 ex3 % kubectl get pods -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP           NODE           NOMINATED NODE   READINESS GATES
newhello-96d86f799-h5d7s    1/1     Running   0          6s    10.244.1.9   minikube-m02   <none>           <none>
newhello-96d86f799-jq69w    1/1     Running   0          6s    10.244.1.8   minikube-m02   <none>           <none>
```

```
spec:
  affinity:
    nodeAffinity: #HERE
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: disktype
            operator: In
            values:
              - label1
```

I made a small change in the affinity tag. Firstly I changed the podAntiAffinity tag inside to the nodeAffinity tag. The nodeAffinity tag specifies the rules used by the scheduler to determine where the pods can be placed. The tag I changed was the podAntiAffinity tag which tells the scheduler not to place the pods on the same nodes if the labels of the pods match. PodAffinity attracts pods to pods and NodeAffinity attracts pods to nodes. We have "hard" rule tag which must be met. Then we simply specify which node we want to place our pods on using our label tag from before.