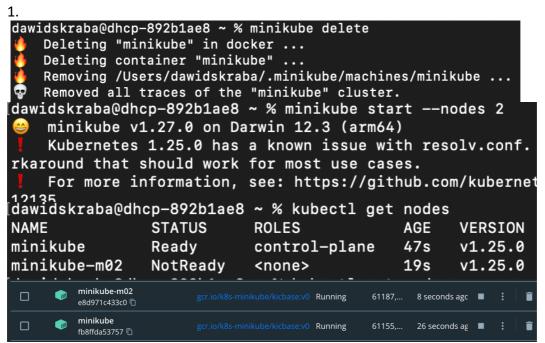
## Practical 3 Ex3 Dawid Skraba 19433692



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.

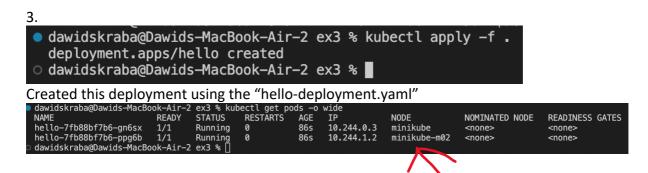
<u>Kind</u>: This specifies the object we are creating. Here it's a Deployment. We could specify other other like a pod to create a pod/deployment to our exact needs.

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

<u>Replicas.strategy</u>: 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.

<u>Spec.template.spec.affinity:</u> 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.

<u>Spec.template.spec.containers:</u> 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.



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
                Ready
 minikube
                         control-plane
                                          48m
                                                v1.25.0
                                                v1.25.0
 minikube-m02
                Ready
                         <none>
                                          48m
▶ 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.

```
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=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.io/os=linux,disktype=label1,kubernetes.io/arch=arm64,beta.kubernetes.i
```

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)

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.