Lab: Minikube

Neha Singh
67
500069028

Minikube has been installed and configured in the environment. Check that it is properly installed, by running the *minikube version* command:

```
Your Interactive Bash Terminal.

$ minikube version
minikube version: v1.8.1
commit: cbda04cf6bbe65e987ae52bb393c10099ab62014
```

Start the cluster, by running the *minikube start* command:

```
$ minikube start --wait=false
* minikube v1.8.1 on Ubuntu 18.04
* Using the none driver based on user configuration
kubectl cluster-info
kubectl get nodes
* Running on localhost (CPUs=2, Memory=2460MB, Disk=145651MB) ...
* OS release is Ubuntu 18.04.4 LTS
* Preparing Kubernetes v1.17.3 on Docker 19.03.6 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
* Launching Kubernetes ...
* Enabling addons: default-storageclass, storage-provisioner
* Configuring local host environment ...
* Done! kubectl is now configured to use "minikube"
```

The cluster can be interacted with using the *kubectl* CLI. This is the main approach used for managing Kubernetes and the applications running on top of the cluster.

Details of the cluster and its health status can be discovered via

```
$ kubectl cluster-info
Kubernetes master is running at https://172.17.0.69:8443
KubeDNS is running at https://172.17.0.69:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
```

To view the nodes in the cluster using

```
 $ kubectl get nodes
 NAME       STATUS     ROLES    AGE   VERSION
 minikube   NotReady   master   5s    v1.17.3
```

Using `kubectl run`, it allows containers to be deployed onto the cluster -

```
$ kubectl create deployment first-deployment --image=katacoda/docker-http-server
deployment.apps/first-deployment created
```

The status of the deployment can be discovered via the running Pods -

```
$ kubectl get pods
NAME                               READY   STATUS              RESTARTS   AGE
first-deployment-666c48b44-mbnz2   0/1     ContainerCreating   0          2s
$ kubectl expose deployment first-deployment --port=80 --type=NodePort
service/first-deployment exposed
```

Once the container is running it can be exposed via different networking options, depending on requirements. One possible solution is NodePort, that provides a

dynamic port to a container.

```
$ kubectl expose deployment first-deployment --port=80 --type=NodePort
service/first-deployment exposed
$ export PORT=$(kubectl get svc first-deployment -o go-template='{{range.spec.ports}}{{if .nodePort}}{{.nod
ort}}{{"\n"}}{{end}}{{end}}')
$ echo "Accessing host01:$PORT"
Accessing host01:30556
$ curl host01:$PORT
<h1>This request was processed by host: first-deployment-666c48b44-mbnz2</h1>
```

The command below finds the allocated port and executes a HTTP request.

```
$ export PORT=$(kubectl get svc first-deployment -o go-template='{{range.spec.ports}}{{if .nodePort}}{{.nod
ort}}{{"\n"}}{{end}}{{end}}')
$ echo "Accessing host01:$PORT"
Accessing host01:30556
$ curl host01:$PORT
<h1>This request was processed by host: first-deployment-666c48b44-mbnz2</h1>
```

Enable the dashboard using Minikube with the command

```
$ minikube addons enable dashboard
* The 'dashboard' addon is enabled
```

Make the Kubernetes Dashboard available by deploying the following YAML definition. This should only be used on Katacoda.

```
$ kubectl apply -f /opt/kubernetes-dashboard.yaml
namespace/kubernetes-dashboard configured
service/kubernetes-dashboard-katacoda created
```

The Kubernetes dashboard allows you to view your applications in a UI. In this deployment, the dashboard has been made available on port *30000* but may take a while to start.

To see the progress of the Dashboard starting, watch the Pods within the *kube-system* namespace using

```
$ kubectl get pods -n kubernetes-dashboard -w
NAME                                         READY   STATUS    RESTARTS   AGE
dashboard-metrics-scraper-7b64584c5c-r5qs5   1/1     Running   0          11s
kubernetes-dashboard-79d9cd965-t7tzq         1/1     Running   0          11s
```