Hello Minikube

This tutorial shows you how to run a sample app on Kubernetes using minikube and Katacoda. Katacoda provides a free, in-browser Kubernetes environment.

> **Note:** You can also follow this tutorial if you've installed minikube locally. See minikube start for installation instructions.

Objectives

- Deploy a sample application to minikube.
- Run the app.
- View application logs.

Before you begin

This tutorial provides a container image that uses NGINX to echo back all the requests.

Create a minikube cluster

1. Click **Launch Terminal**

   Launch Terminal

> **Note:** If you installed minikube locally, run `minikube start`.

2. Open the Kubernetes dashboard in a browser:

```
minikube dashboard
```

3. Katacoda environment only: At the top of the terminal pane, click the plus sign, and then click **Select port to view on Host 1**.

4. Katacoda environment only: Type `30000`, and then click **Display Port**.

Create a Deployment

A Kubernetes *Pod* is a group of one or more Containers, tied together for the purposes of administration and networking. The Pod in this tutorial has only one Container. A Kubernetes *Deployment* checks on the health of your Pod and restarts the Pod's Container if it terminates. Deployments are the recommended way to manage the creation and scaling of Pods.

1. Use the `kubectl create` command to create a Deployment that manages a Pod. The Pod runs a Container based on the provided Docker image.

```
kubectl create deployment hello-node --image=k8s.gcr.io/echoserver:1.4
```

2. View the Deployment:

```
kubectl get deployments
```

   The output is similar to:

```
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
hello-node   1/1     1            1           1m
```

3. View the Pod:

```
kubectl get pods
```

   The output is similar to:

```
NAME                          READY     STATUS    RESTARTS   AGE
hello-node-5f76cf6ccf-br9b5   1/1       Running   0          1m
```

4. View cluster events:

```
kubectl get events
```

5. View the `kubectl` configuration:

```
kubectl config view
```

Note: For more information about `kubectl` commands, see the kubectl overview.

Create a Service

By default, the Pod is only accessible by its internal IP address within the Kubernetes cluster. To make the `hello-node` Container accessible from outside the Kubernetes virtual network, you have to expose the Pod as a Kubernetes *Service*.

1. Expose the Pod to the public internet using the `kubectl expose` command:

```
kubectl expose deployment hello-node --type=LoadBalancer --port=8080
```

   The `--type=LoadBalancer` flag indicates that you want to expose your Service outside of the cluster.

   The application code inside the image `k8s.gcr.io/echoserver` only listens on TCP port 8080. If you used `kubectl expose` to expose a different port, clients could not connect to that other port.

2. View the Service you just created:

```
kubectl get services
```

   The output is similar to:

```
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
hello-node   LoadBalancer   10.108.144.78   <pending>     8080:30369/TCP   21s
kubernetes   ClusterIP      10.96.0.1       <none>        443/TCP          23m
```

On cloud providers that support load balancers, an external IP address would be provisioned to access the Service. On minikube, the `LoadBalancer` type makes the Service accessible through the `minikube service` command.

3. Run the following command:

```
minikube service hello-node
```

4. Katacoda environment only: Click the plus sign, and then click **Select port to view on Host 1**.

5. Katacoda environment only: Note the 5 digit port number displayed opposite to `8080` in services output. This port number is randomly generated and it can be different for you. Type your number in the port number text box, then click Display Port. Using the example from earlier, you would type `30369`.

   This opens up a browser window that serves your app and shows the app's response.

Enable addons

The minikube tool includes a set of built-in addons that can be enabled, disabled and opened in the local Kubernetes environment.

1. List the currently supported addons:

```
minikube addons list
```

   The output is similar to:

```
addon-manager: enabled
dashboard: enabled
default-storageclass: enabled
efk: disabled
freshpod: disabled
gvisor: disabled
helm-tiller: disabled
ingress: disabled
ingress-dns: disabled
logviewer: disabled
metrics-server: disabled
nvidia-driver-installer: disabled
nvidia-gpu-device-plugin: disabled
registry: disabled
registry-creds: disabled
storage-provisioner: enabled
storage-provisioner-gluster: disabled
```

2. Enable an addon, for example, `metrics-server`:

```
minikube addons enable metrics-server
```

The output is similar to:

```
metrics-server was successfully enabled
```

3. View the Pod and Service you just created:

```
kubectl get pod,svc -n kube-system
```

The output is similar to:

```
NAME                                        READY     STATUS    RESTARTS   AGE
pod/coredns-5644d7b6d9-mh9ll                1/1       Running   0          34m
pod/coredns-5644d7b6d9-pqd2t                1/1       Running   0          34m
pod/metrics-server-67fb648c5                1/1       Running   0          26s
pod/etcd-minikube                           1/1       Running   0          34m
pod/influxdb-grafana-b29w8                  2/2       Running   0          26s
pod/kube-addon-manager-minikube             1/1       Running   0          34m
pod/kube-apiserver-minikube                 1/1       Running   0          34m
pod/kube-controller-manager-minikube        1/1       Running   0          34m
pod/kube-proxy-rnlps                        1/1       Running   0          34m
pod/kube-scheduler-minikube                 1/1       Running   0          34m
pod/storage-provisioner                     1/1       Running   0          34m

NAME                            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
service/metrics-server          ClusterIP   10.96.241.45    <none>        80/TCP
service/kube-dns                ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP
service/monitoring-grafana      NodePort    10.99.24.54     <none>        80:30002/TCP
service/monitoring-influxdb     ClusterIP   10.111.169.94   <none>        8083/TCP,8086/
```

4. Disable `metrics-server`:

```
minikube addons disable metrics-server
```

The output is similar to:

```
metrics-server was successfully disabled
```

Clean up

Now you can clean up the resources you created in your cluster:

```
kubectl delete service hello-node
kubectl delete deployment hello-node
```

Optionally, stop the Minikube virtual machine (VM):

```
minikube stop
```