Hello Minikube

This tutorial shows you how to run a sample app on Kubernetes using minikube. The tutorial provides a container image that uses NGINX to echo back all the requests.

Objectives

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

Before you begin

This tutorial assumes that you have already set up `minikube`. See minikube start for installation instructions.

You also need to install `kubectl`. See Install tools for installation instructions.

Create a minikube cluster

```
minikube start
```

Open the Dashboard

Open the Kubernetes dashboard. You can do this two different ways:

Launch a browser | URL copy and paste

If you don't want minikube to open a web browser for you, run the dashboard command with the `--url` flag. `minikube` outputs a URL that you can open in the browser you prefer.

Open a **new** terminal, and run:

```
# Start a new terminal, and leave this running.
minikube dashboard --url
```

Now, switch back to the terminal where you ran `minikube start`.

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.

```
# Run a test container image that includes a webserver
kubectl create deployment hello-node --image=registry.k8s.io/e2e-tes
```

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
```

6. View application logs for a container in a pod.

```
kubectl logs hello-node-5f76cf6ccf-br9b5
```

The output is similar to:

```
I0911 09:19:26.677397       1 log.go:195] Started HTTP server on por
I0911 09:19:26.677586       1 log.go:195] Started UDP server on port
```

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 test image 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 created:

```
kubectl get services
```

The output is similar to:

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

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
```

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:

```
The 'metrics-server' addon is enabled
```

3. View the Pod and Service you created by installing that addon:

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

The output is similar to:

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

NAME                          TYPE        CLUSTER-IP      EXTERNAL-
service/metrics-server        ClusterIP   10.96.241.45    <none>
service/kube-dns              ClusterIP   10.96.0.10      <none>
service/monitoring-grafana    NodePort    10.99.24.54     <none>
service/monitoring-influxdb   ClusterIP   10.111.169.94   <none>
```

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
```

Stop the Minikube cluster

```
minikube stop
```

Optionally, delete the Minikube VM:

```
# Optional
minikube delete
```

If you want to use minikube again to learn more about Kubernetes, you don't need to delete it.

What's next

- Tutorial to *deploy your first app on Kubernetes with kubectl*.
- Learn more about Deployment objects.
- Learn more about Deploying applications.
- Learn more about Service objects.

Feedback

Was this page helpful?

Yes No

Last modified September 15, 2023 at 8:42 AM PST: fix:"Hello Minikube" tutorial objective discrepancy: 'View Application Logs' mentioned but not covered . (#42998) (28a3b6db68)