

Kubernetes Blog Training Partners Community Case Studies Versions ▼ English ▼



Q Search

- ▶ Home
- ▶ Getting started
- ▶ Concepts
- ▶ Tasks
- Tutorials

Hello Minikube

- ▶ Learn Kubernetes Basics
- ▶ Configuration
- ▶ Security
- Stateless Applications

Exposing an External IP Address to Access an Application in a Cluster

Example: Deploying PHP Guestbook application with Redis

- ▶ Stateful Applications
- Services
- ▶ Reference
- ▶ Contribute

Kubernetes Documentation / Tutorials / Stateless Applications / Exposing an External IP Address to Access an Application in a Cluster

Exposing an External IP Address to Access an Application in a Cluster

This page shows how to create a Kubernetes Service object that exposes an external IP address.

Before you begin

- Install kubectl.
- Use a cloud provider like Google Kubernetes Engine or Amazon Web Services to create a Kubernetes cluster. This tutorial creates an external load balancer, which requires a cloud
- Configure kubect1 to communicate with your Kubernetes API server. For instructions, see the documentation for your cloud provider.

Objectives

- Run five instances of a Hello World application.
- Create a Service object that exposes an external IP address.
- Use the Service object to access the running application.

Creating a service for an application running in five pods

1. Run a Hello World application in your cluster:

```
service/load-balancer-example.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
    app.kubernetes.io/name: load-balancer-example
  name: hello-world
spec:
  replicas: 5
  selector:
    matchLabels:
      app.kubernetes.io/name: load-balancer-example
  template:
    metadata:
      labels:
        app.kubernetes.io/name: load-balancer-example
      - image: gcr.io/google-samples/node-hello:1.0
        name: hello-world
        ports:
        - containerPort: 8080
```

kubectl apply -f https://k8s.io/examples/service/load-balancer-example.yaml

The preceding command creates a Deployment and an associated ReplicaSet. The ReplicaSet has five Pods each of which runs the Hello World application.

2. Display information about the Deployment:



kubectl get deployments hello-world kubectl describe deployments hello-world

3. Display information about your ReplicaSet objects:

```
kubectl get replicasets
kubectl describe replicasets
```

4. Create a Service object that exposes the deployment:

```
kubectl expose deployment hello-world --type=LoadBalancer --name=my-service
```

5. Display information about the Service:

```
kubectl get services my-service
```

The output is similar to:

```
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE my-service LoadBalancer 10.3.245.137 104.198.205.71 8080/TCP 54s
```

Note: The type=LoadBalancer service is backed by external cloud providers, which is not covered in this example, please refer to this page for the details.

Note: If the external IP address is shown as <pending>, wait for a minute and enter the same command again.

6. Display detailed information about the Service:

```
kubectl describe services my-service
```

The output is similar to:

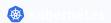
```
Name: my-service
Namespace: default
              app.kubernetes.io/name=load-balancer-example
Labels:
Annotations: <none>
Selector:
             app.kubernetes.io/name=load-balancer-example
Type:
              LoadBalancer
             10.3.245.137
IP:
LoadBalancer Ingress: 104.198.205.71
            <unset> 8080/TCP
NodePort:
             <unset> 32377/TCP
Endpoints: 10.0.0.6:8080,10.0.1.6:8080,10.0.1.7:8080 + 2 more...
Session Affinity: None
Events:
            <none>
```

Make a note of the external IP address (LoadBalancer Ingress) exposed by your service. In this example, the external IP address is 104.198.205.71. Also note the value of Port and NodePort. In this example, the Port is 8080 and the NodePort is 32377.

7. In the preceding output, you can see that the service has several endpoints: 10.0.0.6:8080,10.0.1.6:8080,10.0.1.7:8080 + 2 more. These are internal addresses of the pods that are running the Hello World application. To verify these are pod addresses, enter this command:

```
kubectl get pods --output=wide
```

The output is similar to:



```
... IP
NAME
                NODE
hello-world-2895499144-segjf ... 10.0.2.5 gke-cluster-1-default-pool-e0b8d269-cp
```

8. Use the external IP address (${\tt LoadBalancer}$ Ingress) to access the Hello World application:

```
curl http://<external-ip>:<port>
```

where <external-ip> is the external IP address (LoadBalancer Ingress) of your Service, and <port> is the value of Port in your Service description. If you are using minikube, typing minikube service my-service will automatically open the Hello World application in a browser.

The response to a successful request is a hello message:

Hello Kubernetes!

Cleaning up

To delete the Service, enter this command:

kubectl delete services my-service

To delete the Deployment, the ReplicaSet, and the Pods that are running the Hello World application, enter this command:

kubectl delete deployment hello-world

What's next

Learn more about connecting applications with services.

Feedback

Was this page helpful?





Last modified December 08, 2021 at 6:50 PM PST: Move "Connecting Applications with Services" to tutorials section (ce46f1ca74)

Home

Blog

Training

Case Studies







Partners

Community





