User Guide

David Walter

[2025-09-11 Thu]

Contents

1 Introduction

The Kubernetes External LoadBalancer provides network connectivity to Kubernetes services of type LoadBalancer by assigning external IP addresses and forwarding traffic to service endpoints.
This guide covers installation, configuration, usage, and troubleshooting.

2 Installation

2.1 Prerequisites

Before installing the loadbalancer, ensure you have:

- A Kubernetes cluster (v1.10 or higher)
- A host with access to the Kubernetes API
- A network interface for external IPs
- Sufficient permissions to create RBAC resources

2.2 Installation Methods

2.2.1 Method 1: Direct Deployment

1. Clone the repository:

```
git clone https://github.com/davidwalter0/loadbalancer.git
cd loadbalancer
```

1. Build the binary:

```
go build
```

1. Create the necessary RBAC resources:

```
kubectl apply -f deploy/rbac.yaml
```

1. Deploy the loadbalancer:

```
kubectl apply -f deploy/deployment.yaml
```

2.2.2 Method 2: Using Helm

1. Add the Helm repository:

```
helm repo add loadbalancer https://example.com/charts
helm repo update
```

1. Install the chart:

```
helm install external-lb loadbalancer/external-loadbalancer \
  --set linkdevice=eth0
```

3 Configuration

3.1 Configuration Parameters

The loadbalancer supports the following configuration parameters:

Parameter	Description	Default	Required
`KUBECONFIG`	Path to kubeconfig file	`cluster/auth/kubeconfig`	No
`LINKDEVICE`	Network device for external IPs	-	Yes
`DEBUG`	Enable debug logging	`false`	No
`KUBERNETES`	Use Kubernetes dynamic endpoints	`true`	No

3.2 Configuration Methods

3.2.1 Environment Variables

Set environment variables in the deployment:

```
env:
- name: LINKDEVICE
  value: "eth0"
- name: DEBUG
  value: "true"
```

3.2.2 ConfigMap

Create a ConfigMap and reference it in the deployment:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: loadbalancer-config
data:
  linkdevice: "eth0"
  debug: "true"
---
# In deployment.yaml
env:
- name: LINKDEVICE
  valueFrom:
    configMapKeyRef:
      name: loadbalancer-config
      key: linkdevice
```

3.2.3 Command Line Flags

Pass configuration as command line flags:

```
./loadbalancer --linkdevice=eth0 --debug=true
```

4 Usage

4.1 Creating a LoadBalancer Service

4.1.1 Dynamic IP Allocation

1. Create a Kubernetes service with `type: LoadBalancer` (no `loadBalancerIP` specified):

```
apiVersion: v1
kind: Service
metadata:
  name: example-service
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: example-app
```

1. Apply the service:

```
kubectl apply -f example-service.yaml
```

1. Check the service status:

```
kubectl get service example-service
```

The service should show an external IP automatically assigned by the loadbalancer from the configured IP pool.

4.1.2 Static IP Allocation

To request a specific IP address, set `spec.loadBalancerIP`:

```
apiVersion: v1
kind: Service
metadata:
  name: example-service
spec:
  type: LoadBalancer
  loadBalancerIP: 192.168.0.226  # Request specific IP
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: example-app
```

The IP must be within the configured CIDR range.

4.1.3 Sharing an IP Between Services

Multiple services can share the same IP address on different ports:

```
---
apiVersion: v1
kind: Service
metadata:
  name: web-http
spec:
  type: LoadBalancer
  loadBalancerIP: 192.168.0.226
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: webapp
---
apiVersion: v1
kind: Service
metadata:
  name: web-https
spec:
  type: LoadBalancer
  loadBalancerIP: 192.168.0.226  # Same IP, different port
  ports:
  - port: 443
    targetPort: 8443
  selector:
    app: webapp
```

This is useful for:

- Conserving IP addresses in limited IP pools
- Hosting multiple protocols (HTTP/HTTPS) on the same IP
- Grouping related services

4.2 Service Annotations

The loadbalancer supports the following service annotations:

Annotation	Description	Default
`loadbalancer.example.com/ip`	Request a specific external IP	Auto
`loadbalancer.example.com/internal`	Create an internal-only loadbalancer	`false`

Example:

```
apiVersion: v1
kind: Service
metadata:
  name: example-service
  annotations:
    loadbalancer.example.com/ip: "192.168.1.100"
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: example-app
```

4.3 Accessing Services

Once the service has an external IP, you can access it using:

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

4.4 Viewing Status

To view the status of the loadbalancer:

```
kubectl logs deployment/external-loadbalancer
```

5 Troubleshooting

5.1 Common Issues

5.1.1 No External IP Assigned

Symptoms: Service remains in pending state with no external IP.
Possible causes:

- LoadBalancer not running
- Network interface not found
- RBAC permissions issue

Resolution:

1. Check loadbalancer logs:

   ```
   kubectl logs deployment/external-loadbalancer
   ```

2. Verify the network interface exists:

   ```
   ip addr show
   ```

3. Check RBAC permissions:

   ```
   kubectl auth can-i update services --as=system:serviceaccount:default:loadbalancer-sa
   kubectl auth can-i watch services --as=system:serviceaccount:default:loadbalancer-sa
   ```

5.1.2 Cannot Connect to Service

Symptoms: External IP is assigned but connection refused.
Possible causes:

- Service has no endpoints
- Firewall blocking connection
- Application not listening on port

Resolution:

1. Check if service has endpoints:

   ```
   kubectl get endpoints <service-name>
   ```

2. Verify firewall rules:

   ```
   iptables -L
   ```

3. Check if application is listening:

   ```
   kubectl exec -it <pod-name> -- netstat -lntp
   ```

5.1.3 IP:Port Already Allocated

Symptoms: Service fails with error "IP:port already allocated"
Possible causes:

- Another service is already using the same IP:port combination
- Stale allocation from a crashed service

Resolution:

1. Check which services are using the IP:

   ```
   kubectl get services -o wide | grep <IP-address>
   ```

2. Either:
   - Use a different port on the same IP
   - Use a different IP address
   - Delete the conflicting service

3. Example of fixing the conflict:

   ```
   # If web-http already uses 192.168.0.226:80
   # Use port 443 instead for HTTPS service
   kubectl patch service web-https -p '{"spec":{"loadBalancerIP":"192.168.0.226","ports":[{"port":443}]}}'
   ```

5.1.4 Performance Issues

Symptoms: High latency or connection failures.
Possible causes:

- Insufficient resources
- Network congestion
- Too many connections

Resolution:

1. Check loadbalancer resource usage:

```
kubectl top pod -l app=external-loadbalancer
```

2. Monitor network traffic:

```
iftop -i <interface>
```

3. Scale up resources:

```
kubectl edit deployment external-loadbalancer
# Increase CPU/memory limits
```

5.2 Diagnostic Commands

5.2.1 Check LoadBalancer Status

```
kubectl get pod -l app=external-loadbalancer
kubectl logs deployment/external-loadbalancer
```

5.2.2 Check Service Configuration

```
kubectl get service <service-name> -o yaml
kubectl describe service <service-name>
```

5.2.3 Check Network Configuration

```
ip addr show
ip route
```

5.2.4 Check Connectivity

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

6 Advanced Topics

6.1 High Availability Setup

For production environments, consider running multiple instances of the loadbalancer:

1. Deploy multiple instances:

```
kubectl scale deployment external-loadbalancer --replicas=3
```

2. Use node anti-affinity:

```
affinity:
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: app
          operator: In
          values:
          - external-loadbalancer
      topologyKey: "kubernetes.io/hostname"
```

6.2 Integration with External Systems

6.2.1 Monitoring with Prometheus

1. Enable Prometheus metrics:

```
env:
- name: ENABLE_METRICS
  value: "true"
- name: METRICS_PORT
  value: "9090"
```

2. Create a ServiceMonitor:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: external-loadbalancer
spec:
  selector:
    matchLabels:
      app: external-loadbalancer
  endpoints:
  - port: metrics
```

6.2.2 Logging with Fluentd

Configure Fluentd to collect logs:

```
# fluentd-configmap.yaml
data:
  fluent.conf: |
    <source>
      @type tail
      path /var/log/pods/*external-loadbalancer*/*.log
      pos_file /var/log/fluentd-loadbalancer.pos
      tag kubernetes.loadbalancer
      <parse>
        @type json
      </parse>
    </source>
```

7 Reference

7.1 Command Line Options

Option	Description	Default
`--kubeconfig`	Path to kubeconfig file	cluster/auth/kubeconfig
`--linkdevice`	Network device for external IPs	-
`--debug`	Enable debug logging	false
`--kubernetes`	Use Kubernetes dynamic endpoints	true
`--help`	Show help message	-
`--version`	Show version information	-

7.2 Environment Variables

All command line options have equivalent environment variables with the same name in uppercase.

7.3 Files and Directories

Path	Description
`/etc/loadbalancer`	Default configuration directory
`/var/log/loadbalancer`	Log directory
`/var/run/loadbalancer.pid`	PID file