MetalLB and FRR
A match made in heaven

Federico Paolinelli - Red Hat
Agenda

- MetalLB
- FRR
- MetalLB + FRR
About me 🎩

- Openshift Telco 5G Network team
- Contributed to:
  - KubeVirt
  - SR-IOV Network Operator
  - OVN-Kubernetes
  - CNI plugins
  - Kubernetes
  - MetalLB

@fedepaol
hachyderm.io/@fedepaol
fedepaol@gmail.com
Kubernetes Services
Kubernetes Services
Kubernetes Services

Node 1
**Type: Load Balancer**

Exposes the Service externally using a cloud provider's load balancer. NodePort and ClusterIP Services, to which the external load balancer routes, are automatically created.

```yaml
type: LoadBalancer
status:
  loadBalancer:
    ingress:
      - ip: 203.0.113.100
```
## Load Balancer Service

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbservice</td>
<td>LoadBalancer</td>
<td>10.96.54.189</td>
<td>203.0.113.0</td>
<td>30100:31973/TCP</td>
</tr>
</tbody>
</table>
Load Balancer Service

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbservice</td>
<td>LoadBalancer</td>
<td>10.96.54.189</td>
<td>203.0.113.0</td>
<td>30100:31973/TCP</td>
</tr>
</tbody>
</table>

Node 1

Node 2
Load Balancer Service

Stable IP to reach our application

Load Balancing across the nodes
Let’s move to Bare Metal
### Load Balancer Service

(On bare metal)

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbservice</td>
<td>LoadBalancer</td>
<td>10.96.54.189</td>
<td>&lt;Pending&gt;</td>
<td>30100:31973/TCP</td>
</tr>
</tbody>
</table>
Load Balancer Service
(On bare metal)

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbservice</td>
<td>LoadBalancer</td>
<td>10.96.54.189</td>
<td>&lt;Pending&gt;</td>
<td>30100:31973/TCP</td>
</tr>
</tbody>
</table>

Node 1

Node 2
Enters MetalLB

MetalLB is a load-balancer implementation for bare metal Kubernetes clusters, using standard routing protocols (metallb.universe.tf).
MetalLB is not a load balancer!
Address Assignment

metallb.universe.tf/
Which IPs?
Which IPs?

apiVersion: metallb.io/v1beta1
group: IPAddressPool
metadata:
  name: addresspool-sample1
  namespace: metallb-system
spec:
  addresses:
  - 172.18.0.100-172.18.0.255
Address Advertisement

```
$ curl 192.168.100.11
```

Service:

```yaml
- type: LoadBalancer
  status:
    loadBalancer:
      ingress:
        - ip: 192.168.100.11
```
Two Advertisement Modes

L2  the client and the cluster are in the same local network

BGP  requires interacting with a BGP enabled router
BGP Mode
The primary function of a BGP speaking system is to exchange network reachability information with other BGP systems (BGP RFC)
Service:

type: LoadBalancer
status:
  loadBalancer:
    ingress:
      - ip: 198.51.100.1
BGP Mode

Service:

type: LoadBalancer
status:
  loadBalancer:
    ingress:
      - ip: 198.51.100.1
BGP Mode

Service:

- type: LoadBalancer
- status:
  - loadBalancer:
    - ingress:
      - ip: 198.51.100.1
Service:

```yaml
    type: LoadBalancer
    status:
        loadBalancer:
            ingress:
                - ip: 198.51.100.1
```
BGP Mode

Service:

type: LoadBalancer
status:
  loadBalancer:
    ingress:
      - ip: 198.51.100.1

-> ~ curl 198.51.100.1
BGP Mode

Service:

```
type: LoadBalancer
status:
  loadBalancer:
    ingress:
      - ip: 198.51.100.1
```

Client

→ ~ curl 198.51.100.1

Node 1

Network 198.51.100.1/32
Next Hop 192.168.100.1

192.168.100.1

Node 2

Network 198.51.100.1/32
Next Hop 192.168.100.1

192.168.100.2

Pod

Pod
Service:

- **type**: LoadBalancer
- **status**: loadBalancer:
  - **ingress**: ip: 198.51.100.1

→ `~ curl 198.51.100.1`
BGP Mode

Client

Network 198.51.100.1/32

Next Hop
Router1
Router2

Router1

192.168.100.1

Node 1

Pod
Pod
Service

Node 2

Pod
Pod
Service

192.168.100.2

Client

➜ ~ curl 198.51.100.1

192.168.100.1
BGP Configuration

apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
    name: addresspool-sample1
    namespace: metallb-system
spec:
    addresses:
        - 172.18.0.100-172.18.0.255

apiVersion: metallb.io/v1beta1
kind: BGPPeer
metadata:
    name: peer-sample1
    namespace: metallb-system
spec:
    peerAddress: 10.0.0.1
    peerASN: 64501
    myASN: 64500
    peerPort: 179
    holdTime: "180s"
    keepaliveTime: "180s"
    password: "test"
BGP Configuration

```yaml
apiVersion: metallb.io/v1beta1
class: IPAddressPool
metadata:
  name: addresspool-sample1
  namespace: metallb-system
spec:
  addresses:
    - 172.18.0.100-172.18.0.255
```

```yaml
apiVersion: metallb.io/v1beta1
class: BGPPeer
metadata:
  name: peer-sample1
  namespace: metallb-system
spec:
  peerAddress: 10.0.0.1
  peerASN: 64501
  myASN: 64500
  peerPort: 179
  holdTime: "180s"
  keepaliveTime: "180s"
  password: "test"
```
BGP Mode

- Active / active configuration handled by the external routers
- Extra configuration required to establish BGP sessions
- BFD Support
- Refusing incoming routes
- BGP Peer node selector
- iBGP and eBGP, single and multihop
Architecture
Architecture

- Controller
  - Single Instance
  - Handles the IP pooling and allocation

- Speaker
  - One per node
  - Hostnetworked pod
  - Handles the IP announcement (both L2 and BGP)
Speaker (BGP Native mode)

Speaker container

- Listens for services + MetalLB configuration
- Native BGP implementation in Go
MLB-0001: BGP FRR enablement

Summary

The purpose of this enhancement is to use Free Range Routing (FRR) as an alternative BGP implementation in MetalLB. When directed to, MetalLB will publish prefixes via FRR rather than MetalLB's current built-in BGP implementation.

Motivation

The motivation for this enhancement is to provide an alternative production-ready BGP implementation for use in MetalLB. Overall, this should reduce the effort for adding additional features to the MetalLB project. For example, there are a number of issues in the current backlog that may be addressed by using FRR. Notably:

- Add support for MP BGP encoding for IPv4 and IPv6
- BFD support
- BGP Failover too slow
- OSPF Support
- RIP Support
- Add IPv6 BGP support

FRR is a mature Linux Foundation routing protocol suite based on Quagga that has been used in many production deployments. As such, it has been proven in terms of its maturity, flexibility (as can be seen by the broad range of features it supports), scalability, security, reliability and performance. It also provides detailed logging features to aid debugging.

From [github.com/metallb/metallb/blob/main/design/0001-frr.md](github.com/metallb/metallb/blob/main/design/0001-frr.md)
FRRouting (FRR) is a free and open source Internet routing protocol suite for Linux and Unix platforms. It implements BGP, OSPF, RIP, IS-IS, PIM, LDP, BFD, Babel, PBR, OpenFabric and VRRP, with alpha support for EIGRP and NHRP [...] FRR has its roots in the Quagga project.
FRR Configuration

router bgp 64512
  bgp router-id 10.1.1.254
  neighbor 10.2.2.254 remote-as 64513
  neighbor 10.2.2.254 port 179

  address-family ipv4 unicast
    neighbor 10.2.2.254 activate
    network 172.16.1.10/24
  exit-address-family
FRR Configuration

```
router bgp 64512
  bgp router-id 10.1.1.254
  neighbor 10.2.2.254 remote-as 64513
  neighbor 10.2.2.254 port 179

  address-family ipv4 unicast
    neighbor 10.2.2.254 activate
    network 172.16.1.10/24
  exit-address-family
```
FRR Configuration

```
router bgp 64512
  bgp router-id 10.1.1.254
  neighbor 10.2.2.254 remote-as 64513
  neighbor 10.2.2.254 port 179

  address-family ipv4 unicast
    neighbor 10.2.2.254 activate
    network 172.16.1.10/24
  exit-address-family
```
FRR Configuration

```
router bgp 64512
  bgp router-id 10.1.1.254
  neighbor 10.2.2.254 remote-as 64513
  neighbor 10.2.2.254 port 179

  address-family ipv4 unicast
    neighbor 10.2.2.254 activate
    network 172.16.1.10/24

exit-address-family
```
FRR Configuration - route maps

route-map 10.2.2.254-out permit 2
  match ip address prefix-list with-community
  set community 1111:2222 additive
  on-match next
ip prefix-list with-community permit 172.16.1.10/24

router bgp 64512
  bgp router-id 10.1.1.254
  neighbor 10.2.2.254 remote-as 64513
  neighbor 10.2.2.254 port 179

  address-family ipv4 unicast
    neighbor 10.2.2.254 activate
    neighbor 10.2.2.254 route-map 10.2.2.254-out out
exit-address-family
FRR Configuration - route maps

route-map 10.2.2.254-out permit 2
  match ip address prefix-list with-community
  set community 1111:2222 additive
  on-match next
ip prefix-list with-community permit 172.16.1.10/24

router bgp 64512
  bgp router-id 10.1.1.254
  neighbor 10.2.2.254 remote-as 64513
  neighbor 10.2.2.254 port 179

  address-family ipv4 unicast
    neighbor 10.2.2.254 activate
    neighbor 10.2.2.254 route-map 10.2.2.254-out out
  exit-address-family
FRR Configuration - route maps

```
route-map 10.2.2.254-out permit 2
  match ip address prefix-list with-community
  set community 1111:2222 additive
  on-match next
ip prefix-list with-community permit 172.16.1.10/24
```

```
router bgp 64512
  bgp router-id 10.1.1.254
  neighbor 10.2.2.254 remote-as 64513
  neighbor 10.2.2.254 port 179

  address-family ipv4 unicast
    neighbor 10.2.2.254 activate
    neighbor 10.2.2.254 route-map 10.2.2.254-out out
  exit-address-family
```
FRR Configuration - route maps

route-map 10.2.2.254-out permit 2
  match ip address prefix-list with-community
  set community 1111:2222 additive
  on-match next
  ip prefix-list with-community permit 172.16.1.10/24

router bgp 64512
  bgp router-id 10.1.1.254
  neighbor 10.2.2.254 remote-as 64513
  neighbor 10.2.2.254 port 179

  address-family ipv4 unicast
    neighbor 10.2.2.254 activate
    neighbor 10.2.2.254 route-map 10.2.2.254-out out
  exit-address-family
Speaker BGP mode (FRR)
The workflow
Architecture

Service

BGP Peer

BGPAdv

Kubernetes API

Speaker Pod

Current configuration
Architecture

Service

BGP Peer

BGPAdv

Kubernetes API

Speaker Pod

Code

Data

Go Template

Current configuration

frr-reload.py

frr.conf
The `frr-reload.py` script attempts to update the configuration of running daemons. [...] The script will attempt to retrieve the running configuration from daemons, calculate the delta between that config and the intended one, and execute the required sequence of vtysh commands to enforce the changes.
Architecture

Service

BGP Peer

BGPAdv

Kubernetes API

Speaker Pod

Data

Go Template

Current configuration

frr-reload.py

frr.conf

Code

56
Switching to FRR made easy to implement

- Bidirectional forwarding detection
- VRF support
- IPv6 and Dual Stack support
- (and more to come!)
Challenges
API fitting

MetalLB’s API is not FRR API!

- MetalLB’s focus is on the Service’s IP
- FRR’s focus is on the neighbour
Making sure we don’t break it
Making sure we don’t break it
Making sure we don’t break it
Making sure we don’t break it

- Node selectors
- BGPPeer selectors
- BFD
- Communities
- Local Preferences
- ....

https://flic.kr/p/8RyQBM
Making sure we don’t break it

- Node selectors
- BGPPeer selectors
- BFD
- Communities
- Local Preferences
- ......
Using Kind and FRR to validate MetalLB

*kind* is a tool for running local Kubernetes clusters using Docker container “nodes”.
Using Kind and FRR to validate MetalLB
Using Kind and FRR to validate MetalLB
Using Kind and FRR to validate MetalLB
Using Kind and FRR to validate MetalLB

K8s Configuration (MetalLB + Services)

Node 1

Node 2

Node 3

frr.conf

E2E Tests
Using Kind and FRR to validate MetalLB

K8s Configuration (MetalLB + Services)

Node 1

Node 2

Node 3

E2E Tests

Vtysh show ..

frr.conf

frr.conf
Using Kind and FRR to validate MetalLB

K8s Configuration (MetalLB + Services)

Node 1

Node 2

Node 3

E2E Tests

curl

frr.conf
Using Kind and FRR to validate MetalLB (Multihop)
And it fits in my laptop!
Resources

- Official documentation at [metallb.universe.tf](https://metallb.universe.tf)
- The #metallb slack channel on kubernetes slack
- MetaLLB GitHub [github.com/metallb/metallb](https://github.com/metallb/metallb)
- FRR Routing docs at [frrouting.org](https://frrouting.org)
- FRR Github [github.com/FRRouting/frr](https://github.com/FRRouting/frr)
- FRR Community (slack invite in [frrouting.org/community](https://frrouting.org/community))
A big thanks to the FRR community!
Thanks!

Any questions?

Slides at: speakerdeck.com/fedepaol

@fedepaol
hachyderm.io/@fedepaol
fedepaol@gmail.com