cisco Live!

Let's go

#CiscoLive

The bridge to possible

Integrating Kubernetes into Cisco Networks with BGP and Cilium

Robert Staats – CCIE #3444
Software Engineering Technical Leader

BRKCLD-2999

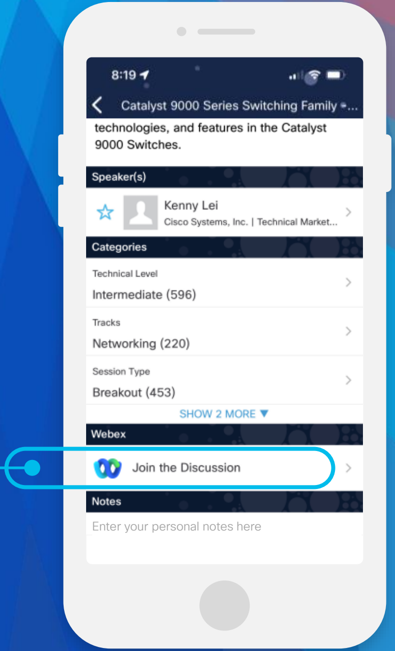
Cisco Webex App

Questions?
Use Cisco Webex App to chat with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until June 9, 2023.


8:19
Catalyst 9000 Series Switching Family ...
technologies, and features in the Catalyst 9000 Switches.
Speaker(s)
Kenny Lei
Cisco Systems, Inc. | Technical Market...
Categories
Technical Level
Intermediate (596)
Tracks
Networking (220)
Session Type
Breakout (453)
SHOW 2 MORE
Webex
Join the Discussion
Notes
Enter your personal notes here

https://ciscolive.ciscoevents.com/ciscolivebot/#BRKCLD-2999

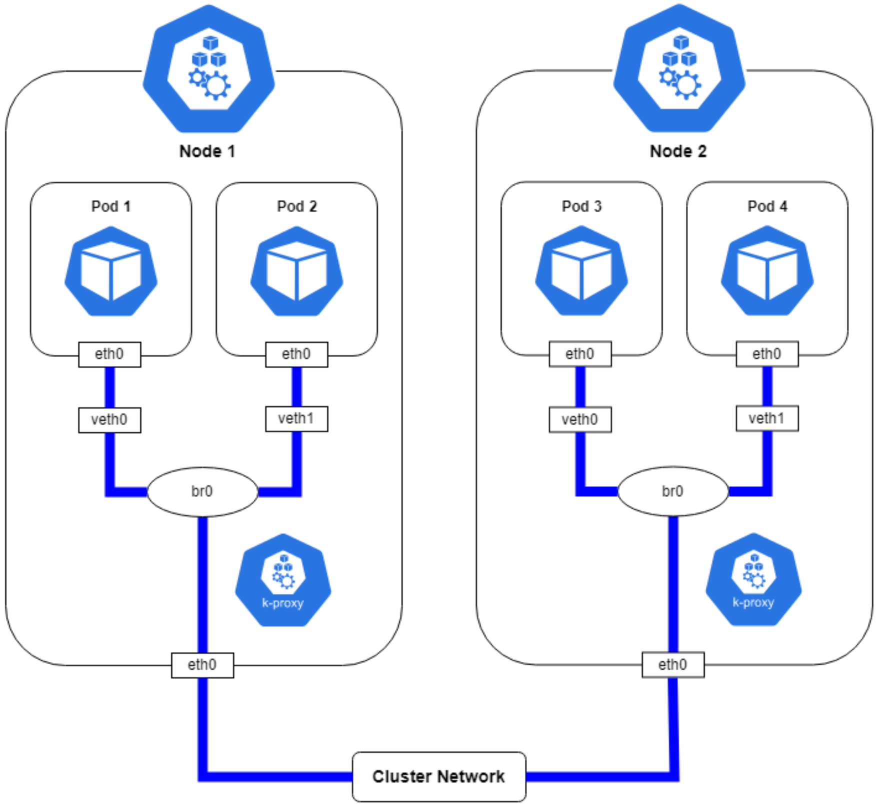
Agenda

- Networking in Kubernetes
- Cilium
- Cilium Installation Considerations
- Cisco BGP Configurations
- Cilium BGP with Kube-Router
- Cilium BGP with BIRD
- Cilium BGP Control Plane

Networking in Kubernetes

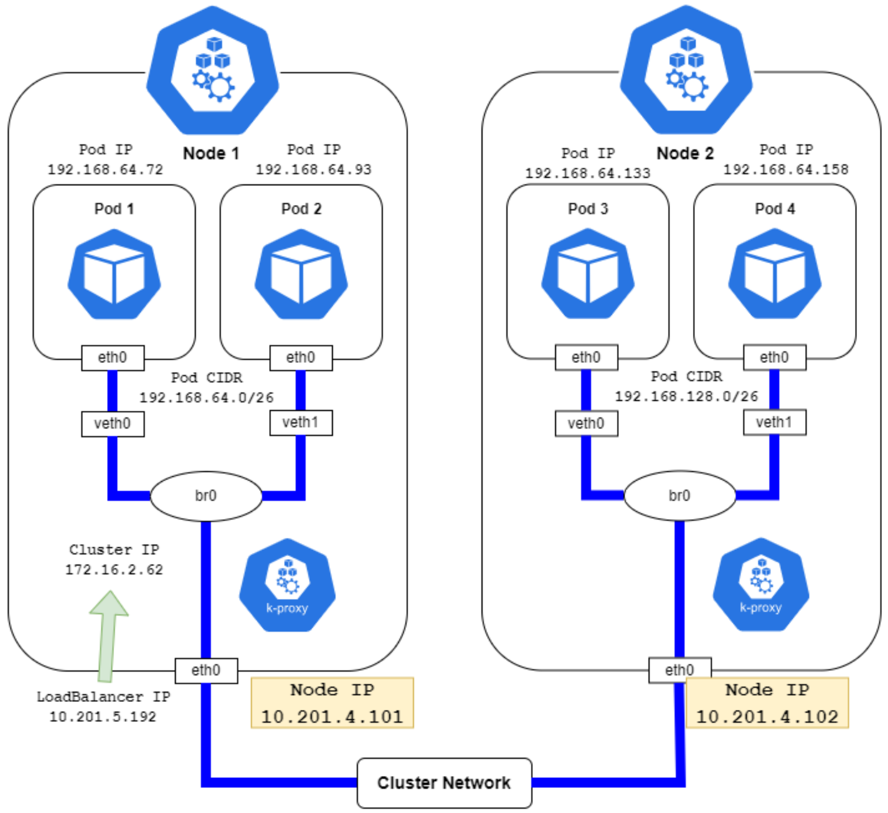
Kubernetes Networking Terms

- Pod is smallest unit of execution in Kubernetes
- Pod consists of one or more containers
- Node is a server that runs the Pods
- Container Network Interface (CNI) plugin provides the network functionality



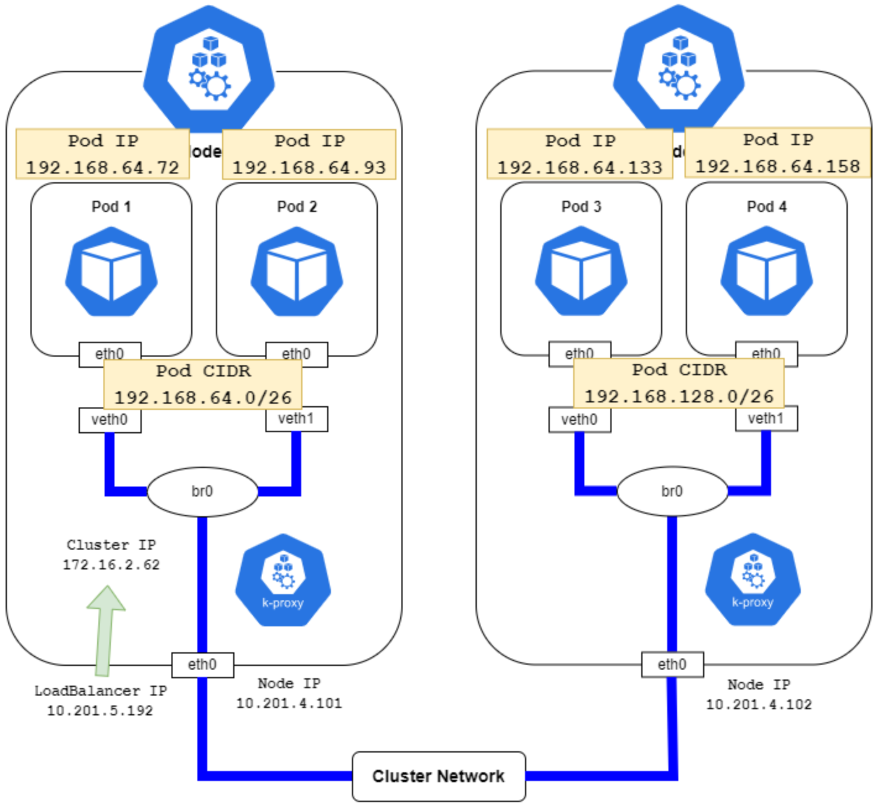
Kubernetes Networking Terms

- Node IPs are manually assigned to each Node.
- These are the IPs that will be used for BGP Peering.



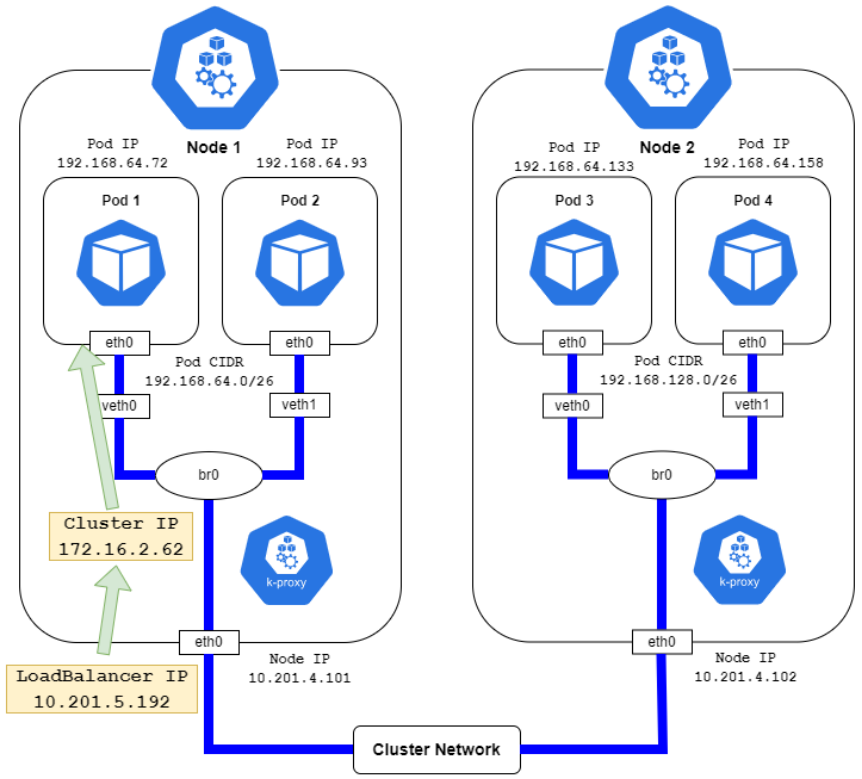
Kubernetes Networking Terms

- Pod IPs are automatically assigned to Pods as they are created.
- The Pod IP is pulled from a pool of IPs assigned to the Node called the Pod CIDR.



Kubernetes Networking Terms

- Cluster IPs are automatically assigned to Services from the Service CIDR Pool.
- Load Balancer IPs are automatically assigned from a Load Balancer Pool.



Container Network Interface (CNI)

- A framework for dynamically configuring networking resources.
- Uses a group of libraries and specifications written in Go.
- The plugin specification defines an interface for configuring the network, provisioning IP addresses, and maintaining connectivity with multiple hosts.

Common Kubernetes CNI Plugins

- Cilium
- Calico
- Flannel
- Weave
- Canal
- Many, many, more can be found here:

  https://github.com/containernetworking/cni

Cilium

What is Cilium?

- CNI plugin for Kubernetes
- Networking, observability, and security solution with an eBPF-based data plane
- Provides a simple flat Layer 3 network with the ability to span multiple clusters in either a native routing or overlay mode
- It is L7-protocol aware and can enforce network policies on L3-L7 using an identity based security model that is decoupled from network addressing

Challenges with iptables

- iptables updates must be made by recreating and updating all rules in a single transaction
- Implements chains of rules as a linked list, so all operations are O(n)
- It’s based on matching IPs and ports, not aware about L7 protocols
- Every time you have a new IP or port to match, rules need to be added and the chain changed
- Has high consumption of resources on Kubernetes

eBPF to the Rescue

- Steve McCanne and Van Jacobson at Lawrence Berkeley Laboratory developed Berkeley Packet Filter (BPF) in 1992
- BPF was designed for network packet filtering with Unix operating systems
- eBPF in Linux began in 2014 with kernel version 3.18
- Fully available since Linux 4.4

What Does eBPF Provide?

- Uses pre-defined hooks into the kernel
- Much more efficient packet filtering.
- Allows for tracing, performance analysis, debugging, security, and more

Cilium Capabilities

- Implements distributed load balancing for traffic between pods and to external services, and is able to fully replace kube-proxy, using efficient hash tables in eBPF allowing for almost unlimited scale
- Supports advanced functionality like integrated ingress and egress gateway, bandwidth management and service mesh, and provides deep network and security visibility and monitoring

Cilium Installation Considerations

Requirements

- Kubernetes must be configured to use CNI
- Linux kernel >= 4.9.17
- Cilium’s kube-proxy replacement depends on the socket-LB feature, which requires a v4.19.57, v5.1.16, v5.2.0 or more recent Linux kernel.
- Linux kernels v5.3 and v5.8 add additional features that Cilium can use to further optimize the kube-proxy replacement implementation

Kube Proxy Replacement

- kube-proxy uses IP Tables
- Cilium can replace kube-proxy to enable the use of eBPF

- Client Source IP Preservation
- Internal Traffic Policy
- Maglev Consistent Hashing
- Direct Server Return (DSR)
- Socket LoadBalancer Bypass in Pod Namespace
- LoadBalancer & NodePort XDP Acceleration
- Session Affinity

NodePort with FHRP & vPC

- When using Cilium’s kube-proxy replacement in conjunction with HSRP and vPC the default configuration can cause issues or unwanted traffic flows.
- You could also choose to set the peer-gateway option on the Nexus vPC config.

```
helm install cilium cilium/cilium \
  --version 1.13.0 \
  --namespace kube-system \
  --set kubeProxyReplacement=strict \
  --set bpf.lbBypassFIBLookup=false
```

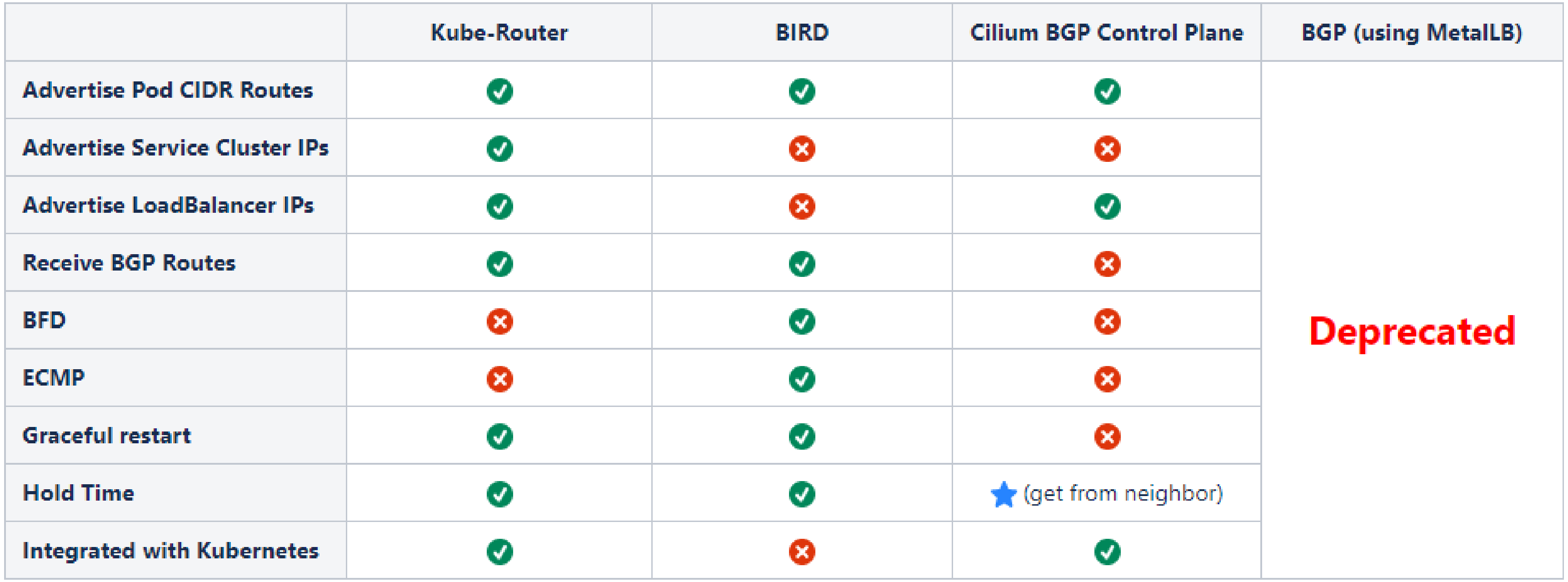
Another option (on Nexus vPC config):

```
vpc domain 1
  peer-gateway
```

Cilium BGP Deployment Options

- Kube-Router - Beta
- BIRD Internet Routing Daemon
- BGP (using MetalLB) - Deprecated
- Cilium BGP Control Plane

Cilium BGP Networking Comparison

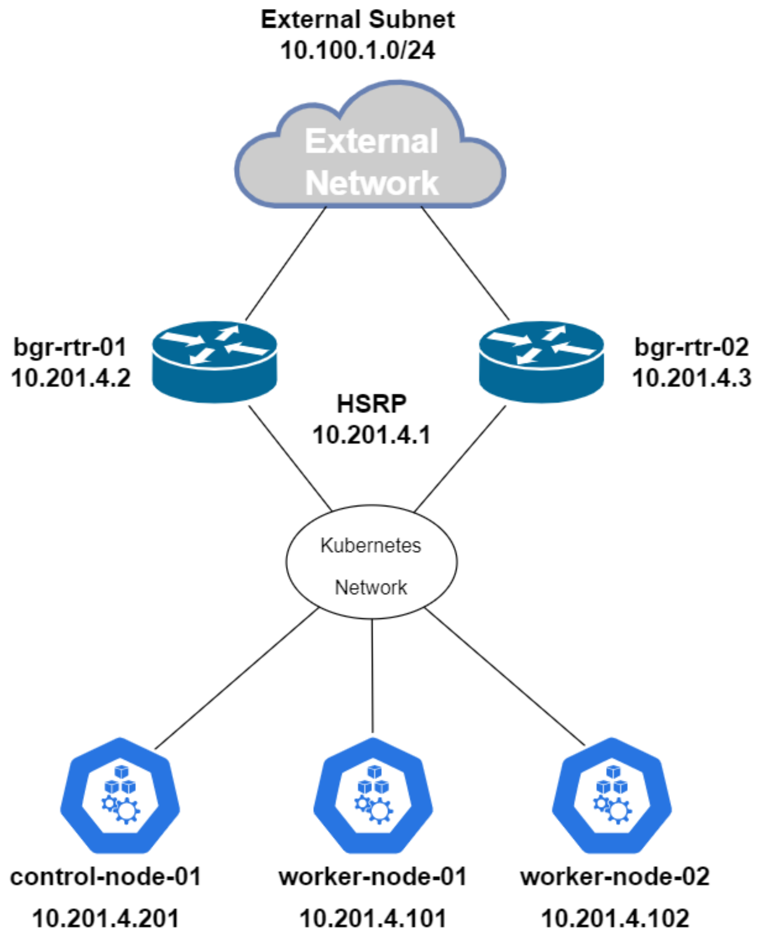


	Kube-Router	BIRD	Cilium BGP Control Plane	BGP (using MetalLB)
Advertise Pod CIDR Routes	✓	✓	✓	
Advertise Service Cluster IPs	✓	✗	✗	
Advertise LoadBalancer IPs	✓	✗	✓	
Receive BGP Routes	✓	✓	✗	
BFD	✗	✓	✗	**Deprecated**
ECMP	✗	✓	✗	
Graceful restart	✓	✓	✗	
Hold Time	✓	✓	★ (get from neighbor)	
Integrated with Kubernetes	✓	✗	✓	

Cisco BGP Configurations

IOS XE

Network Topology



IOS XE

iBGP peering between routers

```
router bgp 65000
 bgp router-id 10.201.4.2
 neighbor 10.201.4.3 remote-as 65000
 neighbor 10.201.4.3 update-source Gi2
 neighbor 10.201.4.101 remote-as 65201
 neighbor 10.201.4.102 remote-as 65201
 neighbor 10.201.4.201 remote-as 65201
!
 address-family ipv4
  network 10.100.1.0 mask 255.255.255.0
  neighbor 10.201.4.3 activate
  neighbor 10.201.4.101 activate
  neighbor 10.201.4.102 activate
  neighbor 10.201.4.201 activate
  maximum-paths 32
 exit-address-family
!
```

IOS XE

eBGP peering to Kubernetes nodes

```
router bgp 65000
 bgp router-id 10.201.4.2
 neighbor 10.201.4.3 remote-as 65000
 neighbor 10.201.4.3 update-source Gi2
 neighbor 10.201.4.101 remote-as 65201
 neighbor 10.201.4.102 remote-as 65201
 neighbor 10.201.4.201 remote-as 65201
!
 address-family ipv4
  network 10.100.1.0 mask 255.255.255.0
  neighbor 10.201.4.3 activate
  neighbor 10.201.4.101 activate
  neighbor 10.201.4.102 activate
  neighbor 10.201.4.201 activate
  maximum-paths 32
 exit-address-family
!
```

IOS XE

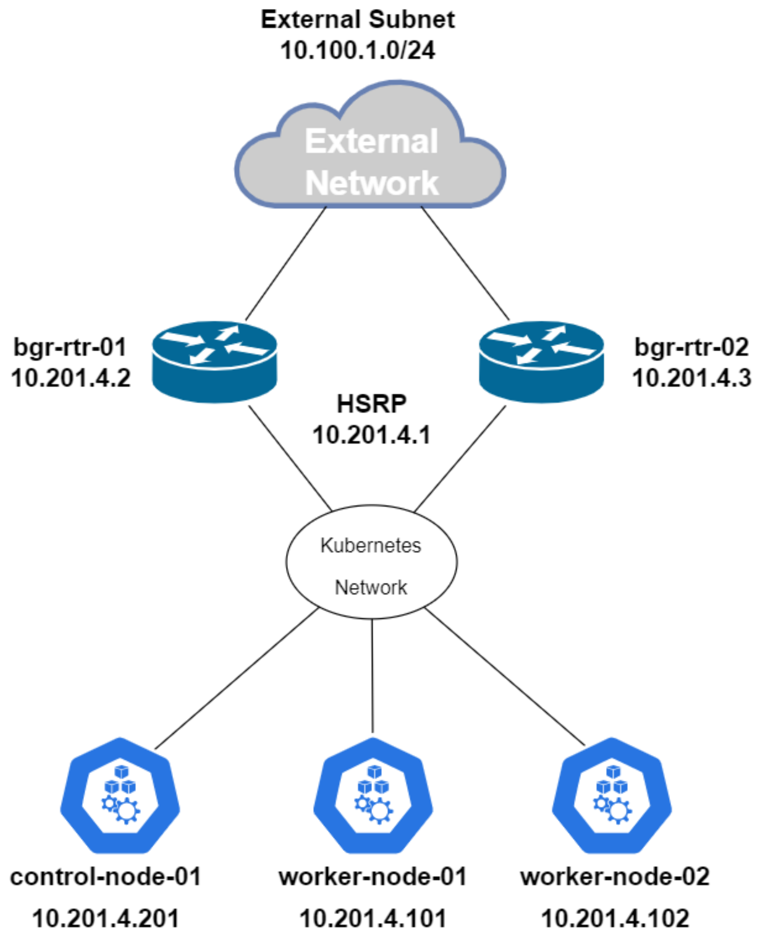
Enabling ECMP with a maximum value of 32 paths

Advertise external route

```
router bgp 65000
 bgp router-id 10.201.4.2
 neighbor 10.201.4.3 remote-as 65000
 neighbor 10.201.4.3 update-source Gi2
 neighbor 10.201.4.101 remote-as 65201
 neighbor 10.201.4.102 remote-as 65201
 neighbor 10.201.4.201 remote-as 65201
!
 address-family ipv4
  network 10.100.1.0 mask 255.255.255.0
  neighbor 10.201.4.3 activate
  neighbor 10.201.4.101 activate
  neighbor 10.201.4.102 activate
  neighbor 10.201.4.201 activate
  maximum-paths 32
 exit-address-family
!
```

NXOS

Network Topology



NXOS

iBGP peering between routers

```
feature bgp

router bgp 65000
  router-id 10.201.4.2
  address-family ipv4 unicast
    network 10.100.1.0/24
    maximum-paths 32
  neighbor 10.201.4.3
    remote-as 65000
    description Peer Router 2
    update-source Ethernet1/2
    address-family ipv4 unicast
  neighbor 10.201.4.101
    remote-as 65201
    description worker-node-01
    address-family ipv4 unicast
  neighbor 10.201.4.102
    remote-as 65201
    description worker-node-02
    address-family ipv4 unicast
  neighbor 10.201.4.201
    remote-as 65201
    description control-node-01
    address-family ipv4 unicast
```

NXOS

eBGP peering to Kubernetes nodes

```
feature bgp

router bgp 65000
  router-id 10.201.4.2
  address-family ipv4 unicast
    network 10.100.1.0/24
    maximum-paths 32
  neighbor 10.201.4.3
    remote-as 65000
    description Peer Router 2
    update-source Ethernet1/2
    address-family ipv4 unicast
  neighbor 10.201.4.101
    remote-as 65201
    description worker-node-01
    address-family ipv4 unicast
  neighbor 10.201.4.102
    remote-as 65201
    description worker-node-02
    address-family ipv4 unicast
  neighbor 10.201.4.201
    remote-as 65201
    description control-node-01
    address-family ipv4 unicast
```

NXOS

Enabling ECMP with a maximum value of 32 paths

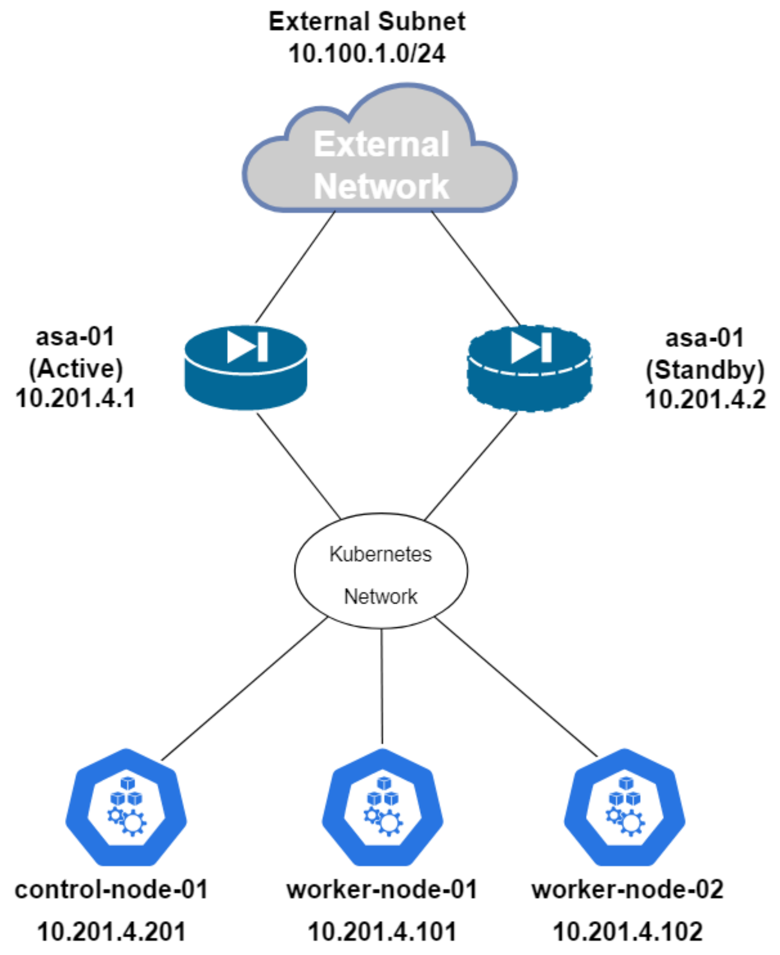
Advertise external route

```
feature bgp

router bgp 65000
  router-id 10.201.4.2
  address-family ipv4 unicast
    network 10.100.1.0/24
    maximum-paths 32
  neighbor 10.201.4.3
    remote-as 65000
    description Peer Router 2
    update-source Ethernet1/2
    address-family ipv4 unicast
  neighbor 10.201.4.101
    remote-as 65201
    description worker-node-01
    address-family ipv4 unicast
  neighbor 10.201.4.102
    remote-as 65201
    description worker-node-02
    address-family ipv4 unicast
  neighbor 10.201.4.201
    remote-as 65201
    description control-node-01
    address-family ipv4 unicast
```

ASA

Network Topology



ASA

Example BGP config

```
router bgp 65000
 address-family ipv4 unicast
  neighbor 10.201.4.101 remote-as 65201
  neighbor 10.201.4.101 activate
  neighbor 10.201.4.102 remote-as 65201
  neighbor 10.201.4.102 activate
  neighbor 10.201.4.201 remote-as 65201
  neighbor 10.201.4.201 activate
  network 10.100.1.0 mask 255.255.255.0
  maximum-paths 8
  no auto-summary
  no synchronization
 exit-address-family
```

ASA

eBGP peering to Kubernetes nodes

```
router bgp 65000
 address-family ipv4 unicast
  neighbor 10.201.4.101 remote-as 65201
  neighbor 10.201.4.101 activate
  neighbor 10.201.4.102 remote-as 65201
  neighbor 10.201.4.102 activate
  neighbor 10.201.4.201 remote-as 65201
  neighbor 10.201.4.201 activate
  network 10.100.1.0 mask 255.255.255.0
  maximum-paths 8
  no auto-summary
  no synchronization
 exit-address-family
```

ASA

Enabling ECMP with a maximum value of 8 paths

Advertise external route

```
router bgp 65000
 address-family ipv4 unicast
  neighbor 10.201.4.101 remote-as 65201
  neighbor 10.201.4.101 activate
  neighbor 10.201.4.102 remote-as 65201
  neighbor 10.201.4.102 activate
  neighbor 10.201.4.201 remote-as 65201
  neighbor 10.201.4.201 activate
  network 10.100.1.0 mask 255.255.255.0
  maximum-paths 8
  no auto-summary
  no synchronization
 exit-address-family
```

Cilium BGP with Kube-Router

	Kube-Router	BIRD	Cilium BGP Control Plane	BGP (using MetalLB)
Advertise Pod CIDR Routes	✓	✓	✓	Deprecated
Advertise Service Cluster IPs	✓	✗	✗	
Advertise LoadBalancer IPs	✓	✗	✓	
Receive BGP Routes	✓	✓	✗	
BFD	✗	✓	✗	
ECMP	✗	✓	✗	
Graceful restart	✓	✓	✗	
Hold Time	✓	✓	★ (get from neighbor)	
Integrated with Kubernetes	✓	✗	✓	

Create the Cluster

- Skipped the installation of the kube-proxy add-on
- Define the Pod Network CIDR because kube-router will pull the pod CIDRs directly from K8s
- This example is using the containerd runtime. Other container runtimes can be used

```
kubeadm init \
  --control-plane-endpoint 10.201.4.201:6443 \
  --cri-socket unix:///run/containerd/containerd.sock \
  --service-cidr 172.16.0.0/20 \
  --pod-network-cidr=192.168.0.0/20 \
  --upload-certs \
  --skip-phases=addon/kube-proxy
```

Deploy Cilium

- tunneling/encapsulation must be disabled in order for routing to be delegated to kube-router
- IPAM mode must be set to kubernetes because kube-router pulls the pod CIDRs directly from K8s

```
helm install cilium cilium/cilium \
  --version 1.13.0 \
  --namespace kube-system \
  --set kubeProxyReplacement=strict \
  --set k8sServiceHost=10.201.4.201 \
  --set k8sServicePort=6443 \
  --set ipam.mode=kubernetes \
  --set tunnel=disabled \
  --set ipv4NativeRoutingCIDR=192.168.0.0/20
```

Load Balancer IP Pool

```
apiVersion:
"cilium.io/v2alpha1"
kind:
CiliumLoadBalancerIPPool
metadata:
  name: "pool-1"
spec:
  cidrs:
  - cidr: "10.201.5.0/24"
  - cidr: "10.201.6.0/24"
  - cidr: "10.201.7.0/24"
```

Apply CiliumLoadBalancerIPPool

```
# kubectl apply -f load-balancer-pool.yml
ciliumloadbalancerippool.cilium.io/pool-1 created
```

Join Worker Nodes

- At this point we have a functional Cluster using Cilium

Download the kube-router DaemonSet template

This is just an example, see the documentation for the latest version.

```
curl -LO https://raw.githubusercontent.com/cloudnativelabs/kube-
router/v1.2/daemonset/generic-kuberouter-only-advertise-
routes.yaml
```

Set Required Args

- Edit the generic-kuberouter-only-advertise-routes.yaml file and edit the args section
- These arguments must be set to these exact values

```
- "--run-router=true"
- "--run-firewall=false"
- "--run-service-proxy=false"
- "--enable-cni=false"
- "--enable-pod-egress=false"
```

Set Optional Args

- Edit the generic-kuberouter-only-advertise-routes.yaml file and edit the args section
- These arguments are optional
- These are the values used for this presentation

```
- "--enable-ibgp=true"
- "--enable-overlay=true"
- "--advertise-cluster-ip=true"
- "--advertise-external-ip=true"
- "--advertise-loadbalancer-ip=true"
```

BGP Peering Args

- Edit the generic-kuberouter-only-advertise-routes.yaml file and edit the args section
- These arguments are optional
- These are the values used for this presentation

ASA (HA Pair)

```
- "--peer-router-ips=10.201.4.1"
- "--peer-router-asns=65000"
- "--cluster-asn=65201‶
```

IOSXR and NXOS (Two routers)

```
- "--peer-router-ips=10.201.4.2,10.201.4.3"
- "--peer-router-asns=65000,65000"
- "--cluster-asn=65201"
```

Deploy Kube-Router

```
# kubectl apply -f generic-kuberouter-only-advertise-routes.yaml
```

Verify the kube-router pods are running

```
# kubectl -n kube-system get pods -o wide -l k8s-app=kube-router
NAME                 READY   STATUS    RESTARTS   AGE   IP             NODE                 NOMINATED NODE   READINESS
GATES
kube-router-fnwpc    1/1     Running   0          11h   10.201.4.102   kube-int-worker-02   <none>           <none>
kube-router-jnp4p    1/1     Running   0          11h   10.201.4.101   kube-int-worker-01   <none>           <none>
kube-router-vchrx    1/1     Running   0          11h   10.201.4.201   kube-int-control-01  <none>           <none>
```

Connect to a kube-router pod

```
# kubectl -n kube-system exec -it kube-router-vchrx bash
```

Show the BGP Neighbors

```
#gobgp neighbor

Peer                AS  Up/Down State       |#Received  Accepted
10.201.4.2      65000 00:05:20 Establ       |        7         1
10.201.4.3      65000 00:03:09 Establ       |        7         1
10.201.4.101    65201 12:57:28 Establ       |        1         1
10.201.4.102    65201 12:57:41 Establ       |        1         1
```

Show the BGP Neighbor Details

gobgp neighbor 10.201.4.2

```
BGP neighbor is 10.201.4.2, remote AS 65000
  BGP version 4, remote router ID 10.201.4.2
  BGP state = ESTABLISHED, up for 00:05:58
  BGP OutQ = 0, Flops = 0
  Hold time is 90, keepalive interval is 30 seconds
  Configured hold time is 90, keepalive interval is 30 seconds

  Neighbor capabilities:
    multiprotocol:
        ipv4-unicast:   advertised and received
    route-refresh:      advertised and received
    extended-nexthop:   advertised
        Local:  nlri: ipv4-unicast, nexthop: ipv6
    graceful-restart:   advertised and received
        Local: restart time 90 sec
            ipv4-unicast
        Remote: restart time 120 sec, restart flag set
            ipv4-unicast
    4-octet-as: advertised and received
    enhanced-route-refresh:     received
    fqdn:       advertised
      Local:
         name: kube-int-control-01, domain:
    cisco-route-refresh:        received
  Message statistics:
                          Sent        Rcvd
    Opens:                  80          10
    Notifications:           5           4
    Updates:                22          45
    Keepalives:           1452        1554
    Route Refresh:           0           1
    Discarded:               0           0
    Total:                1559        1614
  Route statistics:
    Advertised:              5
    Received:                7
    Accepted:                1
```

Show the BGP Routes

```
#gobgp global rib

 Network             Next Hop            AS_PATH                Age        Attrs
*> 10.100.1.0/24       10.201.4.3          65000                  00:12:39   [{Origin: i} {Med: 0}]
*  10.100.1.0/24       10.201.4.2          65000                  00:12:37   [{Origin: i} {Med: 0}]
*> 10.201.5.19/32      10.201.4.201                               00:02:45   [{Origin: i}]
*> 172.16.0.1/32       10.201.4.201                               00:02:45   [{Origin: i}]
*> 172.16.0.10/32      10.201.4.201                               00:02:45   [{Origin: i}]
*> 172.16.6.95/32      10.201.4.201                               00:02:45   [{Origin: i}]
*> 172.16.12.128/32    10.201.4.201                               00:02:45   [{Origin: i}]
*> 192.168.0.0/24      10.201.4.201                               00:02:45   [{Origin: i}]
*> 192.168.1.0/24      10.201.4.101                               00:06:03   [{Origin: i} {LocalPref: 100}]
*> 192.168.2.0/24      10.201.4.102                               00:06:02   [{Origin: i} {LocalPref: 100}]
```

Routing table on Node

Routes shown in red are learned via BGP

```
# ip route
default via 10.201.4.1 dev ens160 proto static
10.100.1.0/24 via 10.201.4.2 dev ens160 proto 17
10.201.4.0/24 dev ens160 proto kernel scope link src 10.201.4.201
192.168.0.0/24 via 192.168.0.121 dev cilium_host src 192.168.0.121
192.168.0.121 dev cilium_host scope link
192.168.1.0/24 via 10.201.4.101 dev ens160 proto 17
192.168.2.0/24 via 10.201.4.102 dev ens160 proto 17
```

BGP Summary on the Router

```
# show ip bgp summary

 […]

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.201.4.3      4        65000      42      44       34    0    0 00:28:32        7
10.201.4.101    4        65201      12      16       34    0    0 00:03:23        6
10.201.4.102    4        65201      11      15       34    0    0 00:02:45        6
10.201.4.201    4        65201      26      28       34    0    0 00:09:20        6
```

BGP Routing Table on Router

`show ip route bgp`

```
[…]
       10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
B         10.201.5.19/32 [20/0] via 10.201.4.201, 00:00:21
                         [20/0] via 10.201.4.102, 00:00:21
                         [20/0] via 10.201.4.101, 00:00:21
      172.16.0.0/32 is subnetted, 4 subnets
B         172.16.0.1 [20/0] via 10.201.4.201, 00:02:33
                     [20/0] via 10.201.4.102, 00:02:33
                     [20/0] via 10.201.4.101, 00:02:33
B         172.16.0.10 [20/0] via 10.201.4.201, 00:02:33
                      [20/0] via 10.201.4.102, 00:02:33
                      [20/0] via 10.201.4.101, 00:02:33
B         172.16.6.95 [20/0] via 10.201.4.201, 00:01:09
                      [20/0] via 10.201.4.102, 00:01:09
                      [20/0] via 10.201.4.101, 00:01:09
B         172.16.12.128 [20/0] via 10.201.4.201, 00:02:33
                        [20/0] via 10.201.4.102, 00:02:33
                        [20/0] via 10.201.4.101, 00:02:33
B      192.168.0.0/24 [20/0] via 10.201.4.201, 00:03:09
B      192.168.1.0/24 [20/0] via 10.201.4.101, 00:02:35
B      192.168.2.0/24 [20/0] via 10.201.4.102, 00:02:33
```

Cilium BGP with BIRD

	Kube-Router	BIRD	Cilium BGP Control Plane	BGP (using MetalLB)
Advertise Pod CIDR Routes	✓	✓	✓	Deprecated
Advertise Service Cluster IPs	✓	✗	✗	
Advertise LoadBalancer IPs	✓	✗	✓	
Receive BGP Routes	✓	✓	✗	
BFD	✗	✓	✗	
ECMP	✗	✓	✗	
Graceful restart	✓	✓	✗	
Hold Time	✓	✓	★ (get from neighbor)	
Integrated with Kubernetes	✓	✗	✓	

Features

- Requires the most configuration
- No integration with Kubernetes

- BGP password support
- BFD
- ECMP
- Graceful restart
- Hold Time

Create the Cluster

- Skipped the installation of the kube-proxy add-on
- Pod Network CIDR is not defined, it will be managed by Cilium
- This example is using the containerd runtime. Other container runtimes can be used

```
kubeadm init \
--control-plane-endpoint 10.201.4.201:6443 \
--cri-socket unix:///run/containerd/containerd.sock \
--service-cidr 172.16.0.0/20 \
--upload-certs \
--skip-phases=addon/kube-proxy
```

Deploy Cilium

- IPAM mode set to cluster-pool so Cilium will handle pod CIDR distribution

```
helm install cilium cilium/cilium \
  --version 1.13.0 \
  --namespace kube-system \
  --set kubeProxyReplacement=strict \
  --set k8sServiceHost=10.201.4.201 \
  --set k8sServicePort=6443 \
  --set ipam.mode=cluster-pool \
  --set ipam.operator.clusterPoolIPv4PodCIDR=192.168.0.0/20 \
  --set ipam.operator.clusterPoolIPv4MaskSize=26 \
  --set tunnel=disabled \
  --set ipv4NativeRoutingCIDR=192.168.0.0/20 \
  --set autoDirectNodeRoutes=true \
  --set bpf.masquerade=true
```

Load Balancer IP Pool

```
apiVersion:
"cilium.io/v2alpha1"
kind:
CiliumLoadBalancerIPPool
metadata:
  name: "pool-1"
spec:
  cidrs:
  - cidr: "10.201.5.0/24"
  - cidr: "10.201.6.0/24"
  - cidr: "10.201.7.0/24"
```

Apply CiliumLoadBalancerIPPool

```
# kubectl apply -f load-balancer-pool.yml
ciliumloadbalancerippool.cilium.io/pool-1 created
```

Join Worker Nodes

- At this point we have a functional Cluster using Cilium
- Now we add BIRD for BGP support

Install BIRD

- Perform the install of each Node
- Install the BIRD repo
- Enable the BIRD service and start it up
- Verify it is running

```
# apt install -y bird2


# systemctl enable bird
# systemctl start bird


# birdc show route
BIRD 2.0.8 ready.
```

Configure BIRD

- Example snippet from /etc/bird/bird.conf of worker 1
- Set Router ID
- Disable generating direct routes
- Set export all for kernel OS to receive BGP routes

```
router id 10.201.4.101;

protocol direct {
  disabled;
}

protocol kernel {
  ipv4 {
    import all;
    export all;
  };
}
```

Configure BIRD

- Add static route for the Pod CIDR
- Add BGP peers

```
protocol static {
  ipv4;
  route 192.168.0.64/26 via "cilium_host";
}


protocol bgp rtr01 {
  description "BGP to rtr-01";
  local 10.201.4.101 as 65201;
  neighbor 10.201.4.1 as 65000;

  ipv4 {
    import filter {accept;};
    export filter {accept;};
  };
}
```

Restart BIRD and check the routing table

```
# systemctl restart bird
```

```
# ip route
default via 10.201.4.1 dev ens160 proto static
10.201.4.0/24 dev ens160 proto kernel scope link src 10.201.4.101
10.100.1.0/24 via 10.201.4.1 dev ens160 proto bird metric 32
192.168.0.0/26 via 10.201.4.201 dev ens160
192.168.0.64/26 via 10.201.4.102 dev ens160
192.168.0.128/26 via 192.168.0.140 dev cilium_host src 192.168.0.140
192.168.0.128/26 dev cilium_host proto bird scope link metric 32
192.168.0.140 dev cilium_host scope link
```

BGP Summary on the Router

```
# show ip bgp summary

 […]

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.201.4.3      4        65000      18      17        6    0    0 00:09:25        4
10.201.4.101    4        65201     136     118        6    0    0 00:09:32        1
10.201.4.102    4        65201      16      18        6    0    0 00:09:33        1
10.201.4.201    4        65201      16      18        6    0    0 00:09:32        1
```

BGP Routing Table on Router

```
# show ip route bgp

[…]

      192.168.0.0/26 is subnetted, 3 subnets
B        192.168.0.0 [20/0] via 10.201.4.201, 00:09:36
B        192.168.0.64 [20/0] via 10.201.4.102, 00:09:36
B        192.168.0.128 [20/0] via 10.201.4.101, 00:09:36
```

BGP Password

- Example snippet from /etc/bird/bird.conf of worker 1
- Add password to bgp peer section for each peer

```
protocol bgp rtr01 {
  description "BGP to rtr-01";
  local 10.201.4.201 as 65201;
  neighbor 10.201.4.2 as 65000;
  password "S@mplePa$$";

  ipv4 {
    import filter {accept;};
    export filter {accept;};
  };
}
```

BGP Password

- Add password to peer on each BGP router

```
router bgp 65000
 neighbor 10.201.4.201 remote-as 65201
 neighbor 10.201.4.201 password S@mplePa$$
 !
 address-family ipv4
  neighbor 10.201.4.201 activate
 exit-address-family
!
```

BFD

Password Options

- none
- simple
- md5
- sha1
- meticulous keyword

```
protocol bfd {
  interface "{{grains['node_mgnt_device']}}" {
    min rx interval 100 ms;
    min tx interval 100 ms;
    idle tx interval 300 ms;
    multiplier 10;
    authentication meticulous keyed md5;
    password "bfdPa$$";
  };

  neighbor 10.201.4.2;
  neighbor 10.201.4.3;
}


protocol bgp rtr01 {
  […]
  bfd on;
}
```

BFD

- Configure key chain
- Configure template
- Apply template to interface
- Enable BFD on BGP neighbor

```
key chain bfd-password
 key 1
  key-string bfdPa$$

bfd-template single-hop bfd-bgp
 interval min-tx 100 min-rx 100 multiplier 10
 authentication meticulous-md5 keychain bfd-
password

interface GigabitEthernet2
 […]
 bfd template bfd-bgp
!

router bgp 65000
 […]
 neighbor 10.201.4.101 fall-over bfd
```

Verify BFD on Worker Node

```
# birdc show bfd sessions
BIRD 2.0.8 ready.
bfd1:
IP address                Interface  State      Since          Interval  Timeout
10.201.4.3                ens160     Up         22:28:46.419     0.300    0.000
10.201.4.2                ens160     Up         22:28:46.419     0.300    0.000
```

Verify BFD on Router

```
8kv-bgp-rtr-01#show bfd summary

                    Session          Up           Down

Total                     2           1              1


8kv-bgp-rtr-01#show bfd neighbors

IPv4 Sessions
NeighAddr                                LD/RD          RH/RS     State     Int
10.201.4.3                             4097/4097        Up        Up        Gi2
10.201.4.101                           4101/4101        Up        Up        Gi2
```

Verify BFD with BGP on Router

```
8kv-bgp-rtr-01#show ip bgp neighbors 10.201.4.101
BGP neighbor is 10.201.4.101,  remote AS 65201, external link
 Fall over configured for session
 BFD is configured. BFD peer is Up. Using BFD to detect fast fallover (single-hop).
 [...]
```

ECMP

- Modify the bird.conf file to enable ECMP
- By default, multipath merging is disabled. If enabled, default value of the limit is 16.

```
protocol kernel {
  ipv4 {
    import all;
    export all;
  };

  merge paths yes limit 2;
}
```

Verify ECMP on Worker Node

```
# ip route
default via 10.201.4.1 dev ens160 proto static
10.100.1.0/24 proto bird metric 32
        nexthop via 10.201.4.2 dev ens160 weight 1
        nexthop via 10.201.4.3 dev ens160 weight 1
10.201.4.0/24 dev ens160 proto kernel scope link src 10.201.4.101
192.168.0.0/26 via 10.201.4.201 dev ens160
192.168.0.64/26 via 10.201.4.102 dev ens160
192.168.0.128/26 via 192.168.0.140 dev cilium_host src 192.168.0.140
192.168.0.128/26 dev cilium_host proto bird scope link metric 32
192.168.0.140 dev cilium_host scope link
```

Graceful Restart

- Enables BGP sessions to be restarted without causing a disruption in the network.
- It works by allowing routers to maintain their established routes even after a session reset or restart.

```
protocol bgp rtr01 {
   [...]
   graceful restart;
}
```

CISCO Live!

Graceful Restart

- Configure each BGP router
- If the **bgp graceful-restart** command has been issued after the BGP session has been established, you must reset by issuing the **clear ip bgp *** command or by reloading the router before graceful restart capabilities will be exchanged.

```
router bgp 65000
 [...]
 bgp graceful-restart
```

Verify Graceful Restart on the Worker Node

```
# birdc show protocols all rtr01
BIRD 2.0.8 ready.
Name       Proto      Table      State  Since          Info
rtr01      BGP        ---        up     23:06:40.965   Established
  Description:    BGP to rtr-01
  BGP state:          Established
    Neighbor address: 10.201.4.2
    Neighbor AS:      65000
    Local AS:         65201
    Neighbor ID:      10.201.4.2
    Local capabilities
      […]
      Graceful restart
        Restart time: 120
        AF supported: ipv4
        AF preserved:
      4-octet AS numbers
      Enhanced refresh
      Long-lived graceful restart
```

Verify Graceful Restart on the Router

```
#show ip bgp neighbors 10.201.4.3 | i Graceful
    Graceful Restart Capability: advertised
  Graceful-Restart is enabled, restart-time 120 seconds, stalepath-time 360 seconds
```

BGP Hold Time

- Default hold time is 180 seconds with a 60 second keepalive.
- Restart BGP session with neighbor to apply new hold time

```
router bgp 65000
[…]
 neighbor 10.201.4.101 timers 10 30
```

BGP Hold Time

- Not needed if set on the upstream router
- BGP will select the values from the peer with the lowest hold time when establishing the BGP session

```
protocol bgp rtr01 {
  description "BGP to rtr-01";
  local 10.201.4.101 as 65201;
  neighbor 10.201.4.2 as 65000;
  bfd on;
  graceful restart;
  hold time 30;
  keepalive time 10;

  ipv4 {
    import filter {accept;};
    export filter {accept;};
  };
}
```

Verify Hold time on the Router

```
8kv-bgp-rtr-01#show ip bgp neighbors  10.201.4.101 | i keepalive
  Last read 00:00:06, last write 00:00:02, hold time is 30, keepalive interval is 10 seconds
  Configured hold time is 30, keepalive interval is 10 seconds
```

BIRD Console

- Use birdc to enter the BIRD console to run commands

```
# birdc
BIRD 2.0.8 ready.
bird>
```

Common Cisco commands in BIRD

Cisco Command	BIRD Command
`show ip route`	`show route [table name]`
`show ip route bgp`	`show route [table name] protocol <protocol_name>`
`show ip route 10.100.0.0 longer-prefixes`	`show route where net ~ 10.100.0.0/16`
`show ip bgp 10.100.1.0`	`show route where net ~ 10.100.0.0/16 all`
`show ip bgp sum`	`show protocols`
`show ip bgp neighbors 10.201.4.101`	`show protocols all <protocol_name>`
`show ip bgp neighbors 10.201.4.101 advertised-routes`	`show route export <protocol_name>`
`clear ip bgp 10.201.4.101`	`reload <protocol_name> [in/out]`
`show ip route summary`	`show route [table name] count`

Cilium BGP Control Plane

	Kube-Router	BIRD	Cilium BGP Control Plane	BGP (using MetalLB)
Advertise Pod CIDR Routes	✓	✓	✓	Deprecated
Advertise Service Cluster IPs	✓	✗	✗	
Advertise LoadBalancer IPs	✓	✗	✓	
Receive BGP Routes	✓	✓	✗	
BFD	✗	✓	✗	
ECMP	✗	✓	✗	
Graceful restart	✓	✓	✗	
Hold Time	✓	✓	★ (get from neighbor)	
Integrated with Kubernetes	✓	✗	✓	

Cilium BGP Control Plane

- Implementation of goBGP
- Only advertises BGP routes
- Currently can not receive BGP routes
- Tightly integrated with Kubernetes and Cilium

Create the Cluster

- Skipped the installation of the kube-proxy add-on
- Pod Network CIDR is not defined, it will be managed by Cilium
- This example is using the containerd runtime. Other container runtimes can be used

```
kubeadm init \
--control-plane-endpoint 10.201.4.201:6443 \
--cri-socket unix:///run/containerd/containerd.sock \
--service-cidr 172.16.0.0/20 \
--upload-certs \
--skip-phases=addon/kube-proxy
```

Enabling BGP Control Plane with Cilium

When set to true the BGP Control Plane Controllers will be instantiated and will begin listening for **CiliumBGPPeeringPolicy** events.

A single flag in the Cilium Agent exists to turn on the BGP Control Plane feature set.

```
--enable-bgp-control-plane=true
```

The BGP Control Plane will only work when IPAM mode is set to "cluster-pool", "cluster-pool-v2beta", and "kubernetes"

Install Cilium

```
helm template cilium cilium/cilium \
  --version 1.13.0 \
  --namespace kube-system \
  --set kubeProxyReplacement=strict \
  --set k8sServiceHost=10.201.4.201 \
  --set k8sServicePort=6443 \
  --set ipam.mode=cluster-pool \
  --set ipam.operator.clusterPoolIPv4PodCIDR=192.168.0.0/20 \
  --set ipam.operator.clusterPoolIPv4MaskSize=26 \
  --set tunnel=disabled \
  --set ipv4NativeRoutingCIDR=192.168.0.0/20 \
  --set autoDirectNodeRoutes=true \
  --set bpf.masquerade=true \
  --set bgpControlPlane.enabled=true \
> cilium-template.yml

kubectl apply -f cilium-template.yml
```

Load Balancer IP Pool

```
apiVersion:
"cilium.io/v2alpha1"
kind:
CiliumLoadBalancerIPPool
metadata:
  name: "pool-1"
spec:
  cidrs:
  - cidr: "10.201.5.0/24"
  - cidr: "10.201.6.0/24"
  - cidr: "10.201.7.0/24"
```

Apply CiliumLoadBalancerIPPool

```
# kubectl apply -f load-balancer-pool.yml
ciliumloadbalancerippool.cilium.io/pool-1 created
```

Join Worker Nodes

- At this point we have a functional Cluster using Cilium

CiliumBGPPeeringPolicy CRD

- All BGP peering topology information is carried in a CiliumBGPPeeringPolicy CRD
- Each CiliumBGPPeeringPolicy defines one or more CiliumBGPVirtualRouter configurations.
- Label is used to assign BGP policy to nodes

```yaml
apiVersion: "cilium.io/v2alpha1"
kind: CiliumBGPPeeringPolicy
metadata:
 name: bgp-peering-policy
spec:
 nodeSelector:
   matchLabels:
     bgp-policy: use-bgp
 virtualRouters:
 - localASN: 65201
   serviceSelector:
     matchExpressions:
     - {key: somekey, operator: NotIn,
values: ['never-used-value’]}
   exportPodCIDR: true
   neighbors:
    - peerAddress: '10.201.4.2/32'
      peerASN: 65000
    - peerAddress: '10.201.4.3/32'
      peerASN: 65000
```

Pod CIDR Announcements

Set exportPodCIDR to true to announce Pod subnets via BGP.

```
apiVersion: "cilium.io/v2alpha1"
kind: CiliumBGPPeeringPolicy
metadata:
 name: bgp-peering-policy
spec:
 nodeSelector:
   matchLabels:
     bgp-policy: use-bgp
 virtualRouters:
 - localASN: 65201
   serviceSelector:
     matchExpressions:
     - {key: somekey, operator: NotIn,
values: ['never-used-value’]}
   exportPodCIDR: true
   neighbors:
    - peerAddress: '10.201.4.2/32'
      peerASN: 65000
    - peerAddress: '10.201.4.3/32'
      peerASN: 65000
```

Service Announcements

By default, virtual routers will not announce services. Virtual routers will announce the ingress IPs of any LoadBalancer services that matches the .serviceSelector of the virtual router.

If you wish to announce ALL services within the cluster, a NotIn match expression with a dummy key and value can be used like:

```
apiVersion: "cilium.io/v2alpha1"
kind: CiliumBGPPeeringPolicy
metadata:
 name: bgp-peering-policy
spec:
 nodeSelector:
   matchLabels:
     bgp-policy: use-bgp
 virtualRouters:
 - localASN: 65201
   serviceSelector:
     matchExpressions:
     - {key: somekey, operator: NotIn,
values: ['never-used-value']}
   exportPodCIDR: true
   neighbors:
    - peerAddress: '10.201.4.2/32'
      peerASN: 65000
    - peerAddress: '10.201.4.3/32'
      peerASN: 65000
```

Apply CiliumBGPPeeringPolicy

```
# kubectl apply -f cilium_bgp_peering_policy.yml
ciliumbgppeeringpolicy.cilium.io/bgp-peering-policy created
```

Add label to nodes to assign the BGP policy

```
# kubectl label nodes kube-int-control-01 bgp-policy=use-bgp

# kubectl label nodes kube-int-worker-01 bgp-policy=use-bgp

# kubectl label nodes kube-int-worker-02 bgp-policy=use-bgp
```

Routing table on Node

No BGP routes

```
# ip route
default via 10.201.4.1 dev ens160 proto static
10.201.4.0/24 dev ens160 proto kernel scope link src 10.201.4.201
192.168.0.0/26 via 10.201.4.101 dev ens160
192.168.0.64/26 via 10.201.4.102 dev ens160
192.168.0.128/26 via 192.168.0.157 dev cilium_host src
192.168.0.157
192.168.0.157 dev cilium_host scope link
```

BGP Summary on the Router

```
# show ip bgp summary

  […]

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.201.4.3      4        65000      20      19       10    0    0 00:11:41        5
10.201.4.101    4        65201      17      22       10    0    0 00:06:46        2
10.201.4.102    4        65201      17      22       10    0    0 00:06:30        2
10.201.4.201    4        65201      17      24       10    0    0 00:06:54        2
```

BGP Routing Table on Router

```
# show ip route bgp

[…]

      10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
B        10.201.5.163/32 [20/0] via 10.201.4.201, 00:04:30
                         [20/0] via 10.201.4.102, 00:04:30
                         [20/0] via 10.201.4.101, 00:04:30
      192.168.0.0/26 is subnetted, 3 subnets
B        192.168.0.0 [20/0] via 10.201.4.101, 00:07:23
B        192.168.0.64 [20/0] via 10.201.4.102, 00:07:07
B        192.168.0.128 [20/0] via 10.201.4.201, 00:07:31
```

Conclusion

Hopefully this provided you with a good understanding of the various BGP deployment options with Cilium and the pros and cons of each option.

Fill out your session surveys!

Attendees who fill out a minimum of four session surveys and the overall event survey will get **Cisco Live-branded socks** (while supplies last)!

Attendees will also earn 100 points in the **Cisco Live Challenge** for every survey completed.

These points help you get on the leaderboard and increase your chances of winning daily and grand prizes

Continue your education

- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

The bridge to possible

Thank you

#CiscoLive

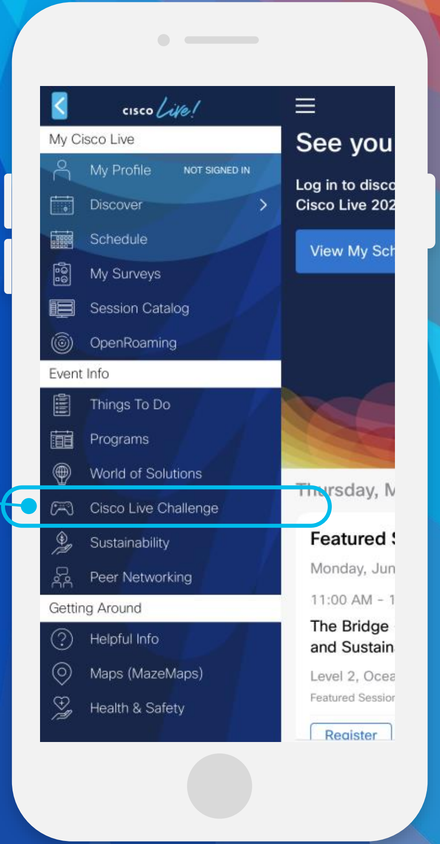
Cisco Live **Challenge**

Gamify your Cisco Live experience!

Get points for attending this session!

How:

1. Open the Cisco Events App.
2. Click on 'Cisco Live Challenge' in the side menu.
3. Click on View Your Badges at the top.
4. Click the + at the bottom of the screen and scan the QR code:



cisco Live!

Let's go

#CiscoLive