Arista EOS and Cilium Technical Brief

Deploying Kubernetes with Arista EOS and Cilium for high performance networking and eBPF security superpowers.

Inside

Overview

Arista CloudEOS for Kubernetes provides a proven network stack for routing on Kubernetes nodes. By using the same EOS code on the Kubernetes nodes as the rest of the network, operators are able to easily manage their entire cluster network with the same tools as their network switches. In addition, CloudEOS provides all the streaming telemetry and analytics as any other EOS device, giving better monitoring and observability for the network state of the Kubernetes node. Arista has partnered with Isovalent to combine CloudEOS with Cilium, the leading provider for application and network security for Kuberntes to provide a complete solution for Kubernetes networking. The combined offering delivers a best-in-class solution for both network and security for Kuberentes clusters built around open standards and without vendor lock-in.

Introduction

As customers deploy their applications onto Kubernetes clusters, many are looking to optimize the performance of network traffic while maintaining operational efficiency and network security. By integrating Arista's CloudEOS with the best-in-class Kubernetes networking provided by Cilium (https://cilium.io), an open source projected maintained by Isovalent, customers are able to directly connect their container workloads to Arista network switches, removing the overhead of encapsulation protocols. In addition, Isovalent provides Kubernetes network security policy allowing administrators to define which pods are allowed to communicate with one another without the use of complex iptables rules.

Customers are further wanting to have the same experience they have in the data center with high performance data center switches running Arista EOS (Extensible operating system). For this use case we put EOS inside of a container as CloudEOS to provide the same functionality, monitoring and analytics as a physical Arista switch allowing the operator to use the same tools to manage their network as their Kubernetes clusters.

This paper describes how to configure Kubernetes networking with Arista EOS and Isovalent's Cilium.

Solution Architecture

Arista's Universal Spine architecture is a high performance layer 3 leaf spine network that provides IP connectivity to the Kubernetes nodes across the data center. Each Kubernetes node runs the CloudEOS container which peers with the top of rack(ToR) switch. Alongside the CloudEOS container the solution provides best of breed Kubernetes network security with Isovalent's Cilium running on each host. Cilium is then configured to run without an overlay and will instead receive BGP routes for each node's container networks directly from the ToR.

Single homed server

In the first scenario a Kubernetes node running Cilium is connected to an Arista ToR switch with a single network connection. BGP is established between the node and the ToR without VXLAN overlay. Once the BGP peering is established, the ToR will see routes from the Kubernetes node. Both CloudEOS and the ToR switch are within the same BGP autonomous system. The leaf switch is running as a BGP route-reflector.

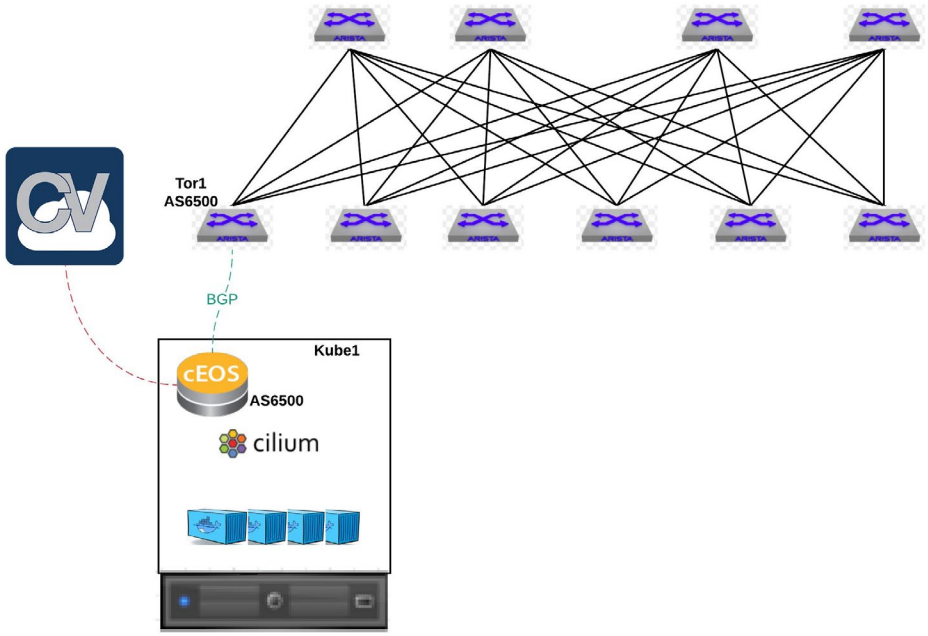

Figure 1: Single homed Kubernetes node

To implement this configuration, the Arista switch must be configured to peer with the CloudEOS pod.

Arista Switch Config

```
maximum-paths 32
neighbor 10.90.224.105 remote-as 65000
neighbor 10.90.224.105 maximum-routes 12000
neighbor 10.90.225.105 route-reflector-client
```

The Kubernetes administrator will also need to apply a single YAML file for Cilium to work in which it will need to be edited to disable all encapsulations so that all routes east/west should follow the routes from BGP.

Cilium deployment based off of Kubernetes version can be found here:

https://cilium.readthedocs.io/en/stable/gettingstarted/k8s-install-default/

The following will need to be changed to disable all encapsulations.

Cilium config - Kube node

cilium.yaml

```
# Encapsulation mode for communication between nodes
# Possible values:
#   - disabled
#   - vxlan (default)
#   - geneve
tunnel: "disabled"
```

Cloud EOS config - Kube node

The CloudEOS config (cloudeos-cilium.yaml) can be found within the Arista public github repo. https://github.com/aristanetworks/cloudeos-k8s

Download and edit the YAML file for your environment.

cloudeos.yaml

```
  env:
  - name: NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
  - name: "BGP_AS"
    value: "65000"
  - name: "INTERFACE_MTU"
    value: "9000"
  - name: "BGP_PEER"
    value: "10.90.224.98"
  - name: "CLOUDVISION_IP"
    value: "10.90.224.168"
  - name: "CNI_PROVIDER"
    value: "cilium"
```

CloudEOS will take any values that are passed from this YAML and then render a configuration to match the single homed server. There are a number of options to configure parameters such as the BGP ASN and peer IP address. Please refer to the Github repo listed above for all of the available options. In this example CloudEOS will use a BGP Autonomous system of 65000, which is configured using an environment variable. For each interface it will use a MTU of 9000. CloudEOS will then start to stream all of its state data the same exact way a data center switch will stream its state data to a CloudVision server at 10.90.224.168. CNI_PROVIDER tells CloudEOS to use the Cilium CNI.

After the CloudEOS container is configured the switch will have a route for the container subnet from the attached node:

```
Leaf1#show ip route bgp

VRF: default

Codes: C - connected, S - static, K - kernel,
       O - OSPF, IA - OSPF inter area, E1 - OSPF external type 1,
       E2 - OSPF external type 2, N1 - OSPF NSSA external type 1,
       N2 - OSPF NSSA external type2, B I - iBGP, B E - eBGP,
       R - RIP, I L1 - IS-IS level 1, I L2 - IS-IS level 2,
       O3 - OSPFv3, A B - BGP Aggregate, A O - OSPF Summary,
       NG - Nexthop Group Static Route, V - VXLAN Control Service,
       DH - DHCP client installed default route, M - Martian,
       DP - Dynamic Policy Route

 B I     10.233.0.0/24 [200/0] via 10.90.224.105, Vlan15
```

CloudEOS uses the 10.233.0.0/24 for any pods which are created through Kubernetes. We can find the 10.233.0.0/24 network within the Kubernetes node's local interfaces. This is where the Cilium CNI comes in to provide the network plumbing for all Kubernetes pods.

```
arista@Kube1:~$ ifconfig cilium_host

cilium_host: flags=4291<UP,BROADCAST,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.233.0.106  netmask 255.255.255.255  broadcast 0.0.0.0
        inet6 fe80::d0a8:b0ff:fee8:4708  prefixlen 64  scopeid 0x20<link>
        ether d2:a8:b0:e8:47:08  txqueuelen 1000  (Ethernet)
        RX packets 17435587  bytes 1445665574 (1.4 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1116  bytes 95020 (95.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

We can then exec into the CloudEOS container that runs within the kube-system namespace to check to see if it is receiving routes from Leaf1. These routes will also show up in CloudVision.

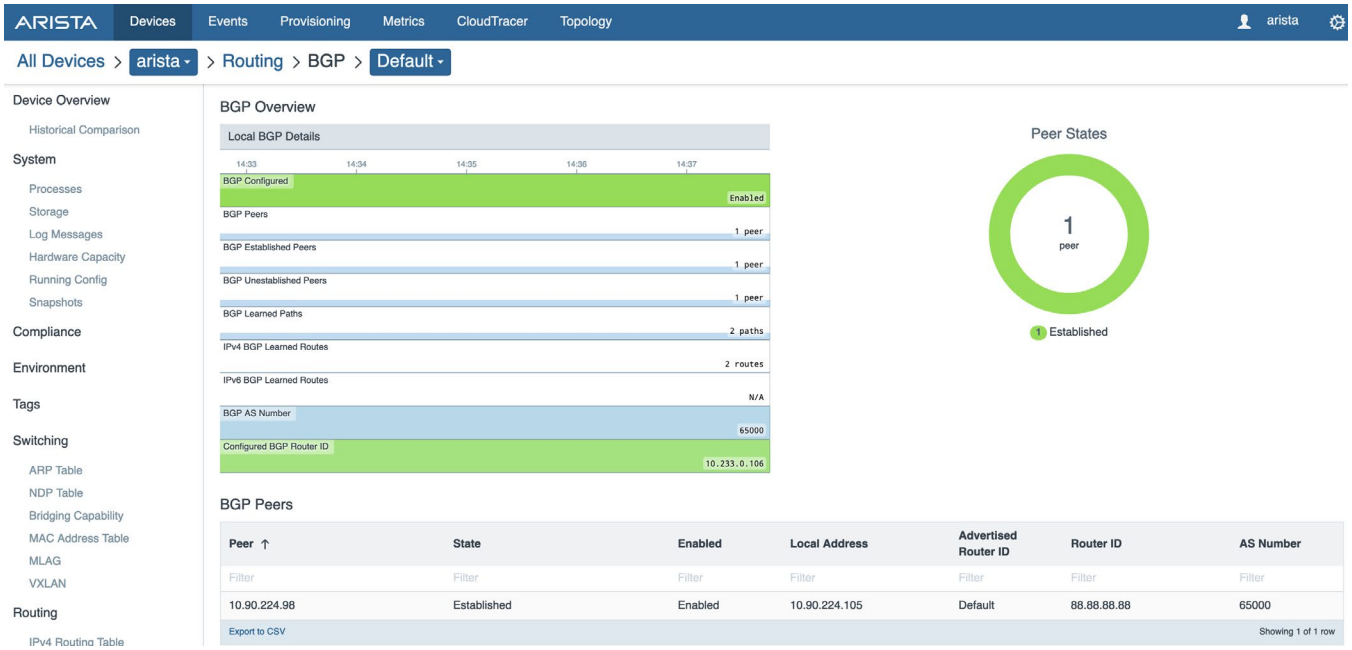

Figure 2: CloudVision view of CloudEOS container routing table

```
arista@Kube1:/home/arista# kubectl get pods -n kube-system
NAME                                READY   STATUS    RESTARTS   AGE
cilium-5vgv9                        1/1     Running   1          40d
cilium-operator-6b9b76785f-7sgvj    1/1     Running   0          17d
cilium-pgbht                        1/1     Running   0          40d
cloudeos-fk94g                      1/1     Running   0          28d
cloudeos-rg87a                       1/1     Running   0          28d
coredns-5644d7b6d9-4wspv            1/1     Running   0          17d
coredns-5644d7b6d9-k2v9n            1/1     Running   7          56d
etcd-arista                         1/1     Running   4          140d
kube-apiserver-arista               1/1     Running   4          140d
kube-controller-manager-arista      1/1     Running   7          140d
kube-proxy-cr99x                    1/1     Running   6          61d
kube-proxy-vtv8z                    1/1     Running   3          140d
kube-scheduler-arista               1/1     Running   6          140d

arista@Kube1:/home/arista# kubectl exec -it cloudeos-fk94g -n kube-system Cli

Kube1>en

Kube1#show ip bgp summary

BGP summary information for VRF default

Router identifier 10.233.0.106, local AS number 65000

Neighbor Status Codes: m - Under maintenance

  Neighbor         V  AS           MsgRcvd   MsgSent  InQ OutQ  Up/Down State   PfxRcd PfxAcc
  10.90.224.98     4  65000              9         9    0    0 00:04:42 Estab   2      2
```

Dual homed server using BGP

For customers looking for redundancy, it is possible to attach the server to two ToR switches. Those ToR switches could be in different BGP autonomous systems or they can be in the same autonomous system. We achieve this through Kubernetes annotations. The Kubernetes administrator simply needs to annotate a node to say that the Kubernetes node will peer via BGP to a peer using the correct BGP autonomous system.

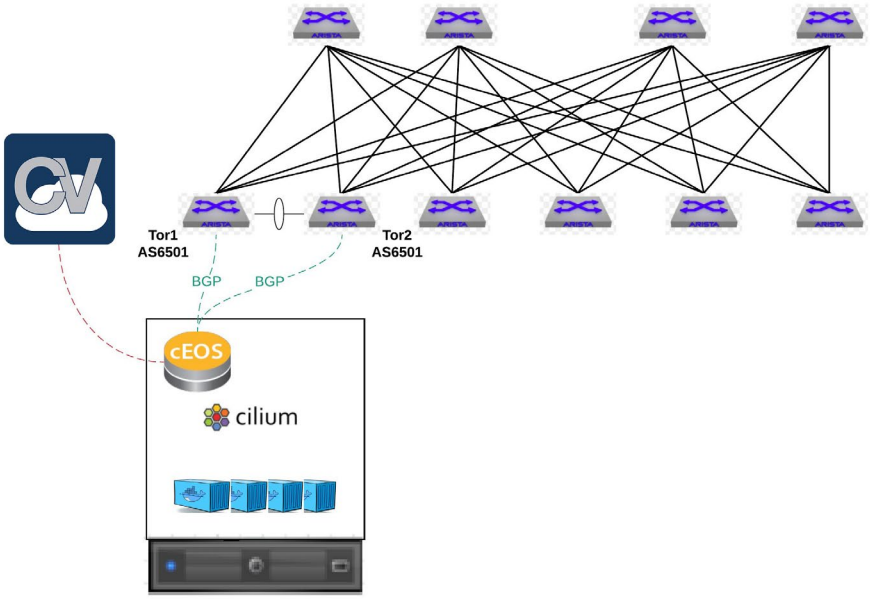

Figure 3: Dual homed Kubernetes server

Arista Switch Config

Tor1

```
router bgp 65001
maximum-paths 32
  neighbor 10.90.224.105 remote-as 65000
  neighbor 10.90.224.105 maximum-routes 12000
  neighbor 10.90.224.106 remote-as 65000
  neighbor 10.90.224.106 maximum-routes 12000
```

Tor2

```
router bgp 65001
maximum-paths 32
  neighbor 10.90.224.105 remote-as 65000
  neighbor 10.90.224.105 maximum-routes 12000
  neighbor 10.90.224.106 remote-as 65000
  neighbor 10.90.224.106 maximum-routes 12000
```

Since Tor is running under BGP AS 65001 in this example we need to create what is referred to as a node annotation. CloudEOS will use the Kubernetes API upon deployment to read the node annotation and render a configuration for BGP.

```
arista@Kube1:~$ kubectl annotate node arista "arista/bgp-remote-as1=65001"
arista@Kube1:~$ kubectl annotate node arista "arista/bgp-remote-as2=65001"
arista@Kube1:~$ kubectl annotate node arista "arista/bgp-peer-ip-1=10.90.224.98"
arista@Kube1:~$ kubectl annotate node arista "arista/bgp-peer-ip-2=10.90.224.99"
```

#cloudeos.yaml

cloudeos.yaml

```
 env:
  - name: NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
  - name: "BGP_AS"
    value: "65000"
  - name: "INTERFACE_MTU"
    value: "9000"
  - name: "CLOUDVISION_IP"
    value: "10.90.224.168"
  - name: "CNI_PROVIDER"
    value: "cilium"
```

Conclusion

By using standard network protocols such as BGP, and best in class data center networking switches, Arista and Isovalent provide a high performance networking solution for Kubernetes clusters. This solution provides operational simplicity and visibility to networking teams deploying Kubernetes in the data center, while supporting advanced features such as eBPF security policies.

For more information, please contact your Arista or Isovalent sales teams.

About Arista

Arista Networks was founded to deliver software-driven cloud networking solutions for large data center storage and computing environments. Arista's award-winning platforms, ranging in Ethernet speeds from 10 to 100 gigabits per second, redefine scalability, agility and resilience. Arista has shipped more than 15 million cloud networking ports worldwide with CloudVision and EOS.

About Isovalent

Founded by long-time leaders in Linux networking and backed by top-tier Silicon Valley investors, Isovalent is a stealth-mode startup that delivers the most advanced Kubernetes networking & security solutions to the most demanding enterprise customers. Isovalent builds and maintains the Cilium and eBPF open source communities and contributes actively to Envoy and Kubernetes, enabling a uniquely powerful open source-centric networking & security solution.

Santa Clara—Corporate Headquarters
5453 Great America Parkway,
Santa Clara, CA 95054

Phone: +1-408-547-5500
Fax: +1-408-538-8920
Email: info@arista.com

Ireland—International Headquarters
3130 Atlantic Avenue
Westpark Business Campus
Shannon, Co. Clare
Ireland

Vancouver—R&D Office
9200 Glenlyon Pkwy, Unit 300
Burnaby, British Columbia
Canada V5J 5J8

San Francisco—R&D and Sales Office 1390
Market Street, Suite 800
San Francisco, CA 94102

India—R&D Office
Global Tech Park, Tower A & B, 11th Floor
Marathahalli Outer Ring Road
Devarabeesanahalli Village, Varthur Hobli
Bangalore, India 560103

Singapore—APAC Administrative Office
9 Temasek Boulevard
#29-01, Suntec Tower Two
Singapore 038989

Nashua—R&D Office
10 Tara Boulevard
Nashua, NH 03062