Antrea BGP Proposal

CRD Design

Draft 1 (Deprecated)

The struct of the CRD:

- Name: **BGPPolicy**
- Spec:
  - `nodeSelector`, required, which is to set the target K8s Nodes.
  - `localASN`, which is the local ASN used by BGP.
  - `advertisements` , required, which has different selectors to select Pod CIDR, Service IPs or Egress IPs, which will be advertised to BGP peers.
  - `bgpPeers`, required, which is the list of BGP peers. Further refinement is possible, given the similarity to the BGP field in the Custom Resource Definition (CRD) for Cillium, as outlined in this link.

Option 1:

For a Kubernetes Node, multiple `BGPPolicies` with distinct `localASN` values can be concurrently applied and take effect. However, if multiple `BGPPolicies` sharing the same `localASN` are applied to a K8s Node, only the initial one will be effective, while the others will be queued. If the currently effective `BGPPolicy` is removed, the first one in the queue will then take effect.

Option 2:

Only one `BGPPolicy` can take effect on a Kubernetes Node, and others serve as alternatives, regardless of the AS number. After an effective `BGPPolicy` applied to a Node is deleted, an alternative BGPPolicy will become effective.

We need more discussion about the selectors in `advertisements`. Currently, we use the following selectors:

- **ClusterIPs**: use serviceSelector and namespaceSelector to select Services, then the ClusterIPs of the selected Services will be advertised.
- **ExternalIPs**: use serviceSelector and namespaceSelector to select Services, then the external IPs of the selected Services will be advertised.

- **LoadBalancerIPs**: use serviceSelector and namespaceSelector to select Services, then the LoadBalancer ingress IPs of the selected Services will be advertised.
- **PodIPs**: boolean, which decides whether to advertise local Pod CIDR.
- **EgressIPs**: boolean, which decides whether to advertise local Egress IPs.

TODO: use selector or boolean for Service and Egress IP selection? For Service IPs, I was thinking of using selector since a custom wants to advertise Service IPs in a certain namespace, according to this link. Another advantage of using selector is that this could provide a flexible way to select Service IPs. For a boolean, it either selects all Service IPs or none.

We thought of these selectors and decided not to use them:

- podSelector + namespaceSelector to select Pod IPs
- the Service CIDR
- serviceSelector + namespaceSelector to select Services, then advertise the Service IPs, like ClusterIP, ExternalIP, and LoadBalancer ingress IPs of them.
- List of namespaced Services.
- the local Pod CIDR.
- ipPoolSelector to select external IP pools used by Egress, then advertise the IPs in these pools
- List of Egress IPs

This is an example of the CR.

```
apiVersion: "antrea.io/v1alpha1"
kind: BGPPolicy
metadata:
  name: test-bgp-policy
spec:
  # Required, select the target K8s Nodes
  nodeSelector:
    matchLabels:
      bgp-policy: web
  # Required, local ASN used by BGP. Two CRs with the same value of the field
cannot be applied to a Node at
  # the same time. CRs with different values of the field can be applied to a
Node at the same time.
  localASN: 60001
  # Required, to select routes to advertise to BGP peers.
  advertisements:
    # Optional, select clusterIPs.
    clusterIPs:
       serviceSelector:
        matchLabels:
          app: web
       namespaceSelector:
```

```
        matchLabels:
          kubernetes.io/metadata.name: prod
    # Optional, select externalIPs.
    externalIPs:
       serviceSelector:
        matchLabels:
          app: web
      namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: prod
    # Optional, select loadBalancerIPs.
    loadBalancerIPs:
       serviceSelector:
        matchLabels:
          app: web
      namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: prod
    # Optional, if setting field to true, local NodeIPAM Pod CIDR will be
advertised. When AntreaIPAM is enabled, IPs allocated by Antrea IPAM will be
also advertised.
    podIPs: false
    # Optional, select Egress IPs.
    egressIPs: false
    #egressIPs:
    #  matchLabels:
    #    app: web

  # Required, list of BGP peers.
  bgpPeers:
    - address: "192.168.1.254"
       port: 179
      asn: 60254
      authSecret: secretname
      multipleHopTTL: 10
      connectRetryTime: 120
      holdTime: 90
      keepAliveTime: 30
      gracefulRestart:
       enabled: true
        restartTimeSeconds: 120
    - address: "192.168.1.253"
       port: 179
      asn: 60253
      authSecret: secretname
      multipleHopTTL: 10
      connectRetryTime: 120
      holdTime: 90
      keepAliveTime: 30
      gracefulRestart:
       enabled: true
```

```
        restartTimeSeconds: 120
```

Draft 2

Name

BGPPolicy

Description

- With `nodeSelector`, apply a BGPPolicy to the select K8s Nodes.
- If multiple BGPPolicies are applied to a Node, only the first one will be effective and in `Active` status, and others serve as alternatives.
- `localASN` of every item in `bgpConfigurations` should be unique.
- Multiple `bgpConfigurations` are necessary for advertising various sets of IPs to different groups of BGP peers.

Prons

- Easy to advertise different sets of IPs to different BGP peers.
- Only selectors and flags are needed to decide whether to advertise selected IPs to a group of BGP peers.
- The implementation becomes more application-oriented as it utilizes Service and Namespace selectors. There is no need to employ certain BGP filters that might require defining CIDRs or IPs for exclusion from a group of BGP peers.
- It might be simpler to implement the controller of the CRD.

Cons

- More resources are needed since multiple `bgpConfigurations` are needed in some cases.
- Address-oriented filtering is not supported when there is a need to exclude one or more specific IPs from a group of BGP peers.

Metadata

Field	Description	Schema
name		string

Spec

Field	Description	Schema	Default
nodeSelector	Select the Nodes to which the BGPPlicy will be applied.	selector	

configurations	The list of BGPConfigurations. Every BGPConfiguration is for a BGP process.	list of BGPConfiguration	

BGPConfiguration

Field	Description	Schema	Default
localASN	The local AS number used by BGP.	integer	
bindAddresses	The addresses where to listen for BGP connections. If this is empty, listen to all IPv4 and IPv6 addresses.	list of string	empty list
listenPort	The listening port used by BGP.	integer	
routerID	RouterID used by BGP. It is only needed when deploying an IPv6 single stack.	string of an IPv4 address	empty string
advertisements	Select the prefixes to advertise to BGP peers.	Advertisements or List of Advertisement	
peers	The list of BGP peers.	list of BGPPeer	

Advertisements

Field	Description	Schema	Default
clusterIPs	Select Services and their ClusterIPs will be advertised.	Service selector and Namespace selector	nil
externalIPs	Select Services and their external IPs will be advertised.	Service selector and Namespace selector	nil
loadbalancerIPs	Select Services and	Service selector and	nil

	their LoadBalancer ingress IPs will be advertised.	Namespace selector	
podIPs	Whether to advertise local Pod CIDR	boolean	false
egressIPs	Whether to advertise local Egress IPs	boolean	false

Optimized Advertisement

Thanks for antonin.bas@gmail.com's suggestion.

Field	**Description**	**Schema**	**Default**
services	Select Services and their LoadBalancer ingress IPs will be advertised.	list of ServiceAdvertisement	empty list
podIPs	Whether to advertise local Pod CIDR	boolean	false
egressIPs	Whether to advertise local Egress IPs	boolean	false

ServiceAdvertisement

serviceSelector	Select Services with selector.	selector	nil
namespaceSelector	Select Services with Namespace selector.	selector	nil
clusterIPs	Whether to advertise clusterIPs.	boolean	false
externalIPs	Whether to advertise externalIPs.	boolean	false
loadbalancerIPs	Whether to advertise loadbalancerIPs.	boolean	false

BGPPeer

Field	Description	Schema	Default
address	IPv4 or IPv6 address of BGP peer	IPv4 or IPv6 string	
port	Port of BGP peer	integer	179
asn	The As number of BGP peer	integer	
gracefulRestartTime	Restart time for BGP graceful restart. If it is set to 0, graceful restart of BGP will be disabled.	integer	120
authSecretRef	BGP password to the peer.	string (name of a K8s Secret)	nil (no password)

BGPPeert

Field	Description	Schema	Default
address	IPv4 or IPv6 address of BGP peer	IPv4 or IPv6 string	
port	Port of BGP peer	integer	179
asn	The As number of BGP peer	integer	
gracefulRestartTime	Restart time for BGP graceful restart. If it is set to 0, graceful restart of BGP will be disabled.	integer	120
authSecretRef	BGP password to the peer.	string (name of a K8s Secret)	nil (no password)

Sample YAML

```
apiVersion: "antrea.io/v1alpha1"
```

```yaml
kind: BGPPolicy
metadata:
  name: test-bgp-policy
spec:
  nodeSelector:
    matchLabels:
      kubernetes.io/hostname: k8s-node-control-plane
  bgpConfigurations:
    - localASN: 60001
      advertisements:
        externalIPs:
          serviceSelector:
            matchLabels:
              app: web
          namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: prod
        egressIPs: true
      bgpPeers:
        - address: "192.168.1.254"
          port: 179
          asn: 60254
          authSecretRef: secretname
    - localASN: 60002
      advertisements:
        clusterIPs:
          serviceSelector:
            matchLabels:
              app: web
          namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: test
        podIPs: true
      bgpPeers:
        - address: "192.168.1.253"
          port: 179
          asn: 60253
```

Sample YAML with Optimize Advertisements

```yaml
apiVersion: "antrea.io/v1alpha1"
kind: BGPPolicy
metadata:
  name: test-bgp-policy
spec:
  nodeSelector:
```

```
    matchLabels:
      kubernetes.io/hostname: k8s-node-control-plane
  bgpConfigurations:
    - localASN: 60001
      advertisements:
        serviceIPs:
          - serviceSelector:
              matchLabels:
                zone: a
              namespaceSelector:
                matchLabels:
                  kubernetes.io/metadata.name: prod
              externalIP: true
              loadBalancerIP: true
          - serviceSelector:
              matchLabels:
                zone: b
              namespaceSelector:
                matchLabels:
                  kubernetes.io/metadata.name: prod
              clusterIP: true
        egressIPs: true
        podIPs: true
      bgpPeers:
        - address: "192.168.1.254"
          port: 179
          asn: 60254
          authSecretRef: secretname
    - localASN: 60002
      advertisements:
        serviceIPs:
          - serviceSelector:
              matchLabels:
                zone: a
              namespaceSelector:
                matchLabels:
                  kubernetes.io/metadata.name: test
              clusterIP: true
        podIPs: true
      bgpPeers:
        - address: "192.168.1.253"
          port: 179
          asn: 60253
```

Name

BGPPolicy

Description

- With `nodeSelector`, apply a BGPPolicy to the select K8s Nodes.
- If multiple BGPPolicies are applied to a Node, only the first one will be effective and in `Active` status, and others serve as alternatives.
- `localASN` is not required. The default value is 65000.
- `BGPFilters` are needed to avoid advertising some sets of IPs to a BGP peer. A `BGPFilter` can be used by multiple `bgpPeer`.

Prons

- Less resources are needed since only a single BGP process is needed.
- The implementation has more fine-grained filters, using BGPFilters, to advertise different sets of IPs to different groups of BGP peers.

Cons

- It is a bit complex to use when using the BGPFilter.
- The implementation might become more complex as an additional `CRD` BGPFilter is required.

Metadata

Field	Description	Schema
name		string

Spec

Field	Description	Schema	Default
localASN	The local AS number used by BGP.	integer	65000
bindAddresses	The addresses where to listen for BGP connections. If this is empty, listen to all IPv4 and IPv6 addresses.	list of string	empty list
listenPort	The listening port used by BGP.	integer	179
nodeSelector	Select the Nodes to which the BGPPlicy will be applied.	selector	

routerID	RouterID used by BGP. It is only needed when deploying an IPv6 single stack.	string of a IPv4 address	empty string
advertisements	Select the prefixes to advertise to BGP peers.	Advertisements	
peers	The list of BGP peers.	list of BGPPeer	

Advertisements

Field	Description	Schema	Default
clusterIPs	Select Services and their ClusterIPs will be advertised.	Service selector and Namespace selector	nil
externalIPs	Select Services and their external IPs will be advertised.	Service selector and Namespace selector	nil
loadbalancerIPs	Select Services and their LoadBalancer ingress IPs will be advertised.	Service selector and Namespace selector	nil
podIPs	Whether to advertise local Pod CIDR	boolean	false
egressIPs	Whether to advertise local Egress IPs	boolean	false

BGPPeer

Field	Description	Schema	Default
address	IPv4 or IPv6 address of BGP peer	IPv4 or IPv6 string	
port	Port of BGP peer	integer	179
asn	The As number of	integer	

	BGP peer		
gracefulRestartTime	Restart time for BGP graceful restart. If it is set to 0, graceful restart of BGP will be disabled.	integer	120
authSecretRef	BGP password to the peer.	string (name of a K8s Secret)	nil (no password)
filters	A list of BGPFilters applied to the peer. These filters are used for preventing some prefixes to be advertised to the peer.	list of BGPFilter	empty list

BGPFilter

TODO

Sample YAML

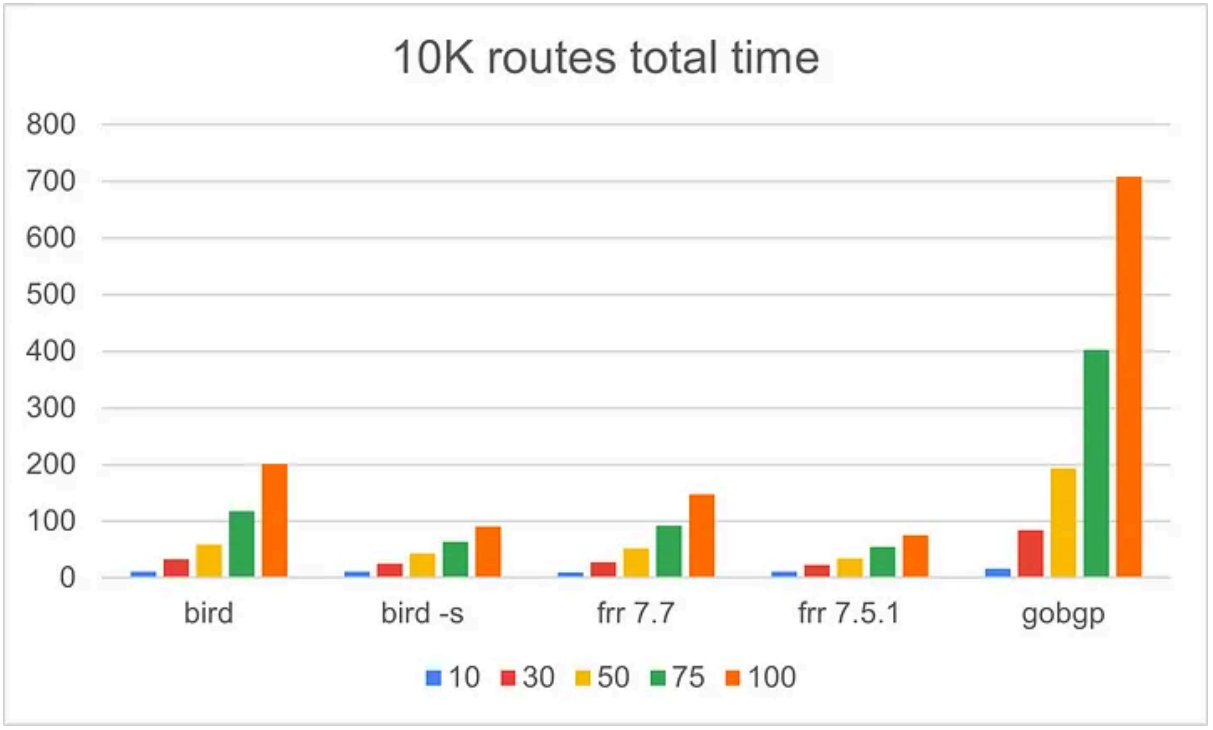
```yaml
apiVersion: "antrea.io/v1alpha1"
kind: BGPPolicy
metadata:
  name: test-bgp-policy
spec:
  nodeSelector:
    matchLabels:
      kubernetes.io/hostname: k8s-node-control-plane
  localASN: 65000
  advertisements:
    externalIPs:
      serviceSelector:
        matchLabels:
          app: web
      namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: prod
    clusterIPs:
      serviceSelector:
        matchLabels:
          app: web
      namespaceSelector:
```

```
      matchLabels:
        kubernetes.io/metadata.name: test
  podIPs: true
  egressIPs: true
  bgpPeers:
    - address: "192.168.1.254"
      port: 179
      asn: 60254
      authSecretRef: secretname
      filters:
          - reject-pod-ips
          - reject-cluster-ips
    - address: "192.168.1.253"
      port: 179
      asn: 60253
      filters:
          - reject-egress-ips
          - reject-external-ips
```

Data Path

We utilize goBGP as our data path, leveraging its maturity as a native Go framework. Notably, it is employed by projects such as Cillium, Kube-router, and others.

In comparison to other BGP implementations, while goBGP may not be the optimal choice, it proves to be adequate for advertising Kubernetes Service IPs and Pod IPs. In the figure below, the prefix routes scale is 10K, with peer numbers of 10, 30, 50, 75, and 100. We could see that the convergence time of goBGP is almost the same with others when there are 10 peers. Our implementation exclusively requires eBGP, and a full mesh of iBGP is unnecessary. Typically, each K8s Node might establish BGP connections with at most two peers. See the benchmark result in this link.



Despite the relatively high resource cost of goBGP, we can mitigate this by disabling certain features to reduce resource consumption.

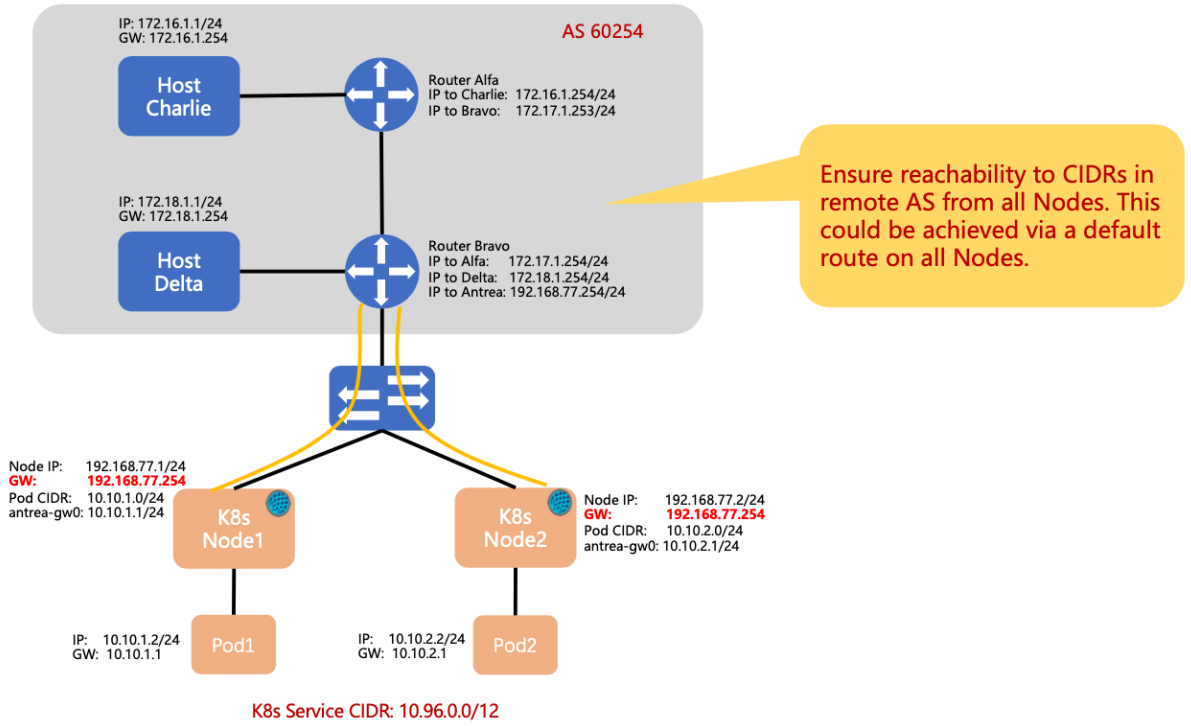
	nos	version	peers	routes per pe	neighbor (s)	elapsed (s)	since first ro	exabgp (s)	total time	max cpu %	max mem (G	flags	date	cores	Mem (GB)	notes	results
2	bird	v2.0.8-52-g8eea3	10	10000	3	2	1	1	00:11	10	0.00933		2021/6/10	16	64	table per neighbor that redistributes to a master table	
3	frr	7.7-dev_git	10	10000	1	3	2	1	00:10	19	0.057		2021/6/10	16	64		
4	frr	7.5.1_git (d64c849b	10	10000	0	3	3	0	00:11	40	0.105		2021/7/27	16	64		
5	gobgp	2.28.0	10	10000	6	4	3	1	00:17	1160	0.077		2021/6/10	16	64		
6	bird	v2.0.8-52-g8eea3	30	10000	3	9	9	0	00:33	101	1.11		2021/6/10	16	64	table per neighbor that redistributes to a master table	
7	bird	v2.0.8-52-g8eea3	30	10000	3	3	3	0	00:26	99	1.11	#NAME?	2021/6/10	16	64	single table	
8	frr	7.7-dev_git	30	10000	5	3	3	0	00:28	65	0.293		2021/6/10	16	64		
9	frr	7.5.1_git (d64c849b	30	10000	0	3	3	0	00:23	67	0.281		2021/7/27	16	64		
10	gobgp	2.28.0	30	10000	4	59	59	0	01:24	1570	0.507		2021/6/10	16	64		

To facilitate a potential switch to another BGP implementation in the future, it is essential to maintain loose coupling between the controller implementation of BGPPolicy and the BGP interface.

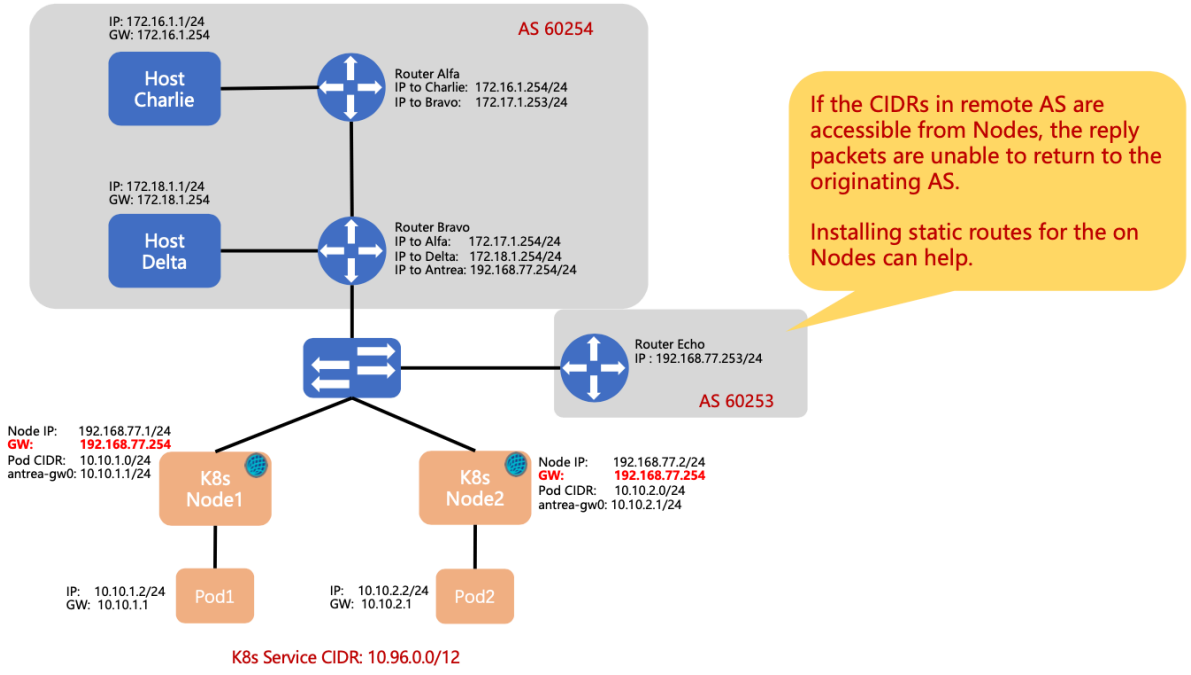
TODO: support more BGP data paths in the future, maybe FRR or Bird.

Requirements

Ensure reachability to CIDRs from outside AS through the default route on all K8s Nodes. Like the following:



If the Kubernetes network is structured like the following, we may not currently plan to provide support. While the issue can be addressed by installing routes learned from BGP peers, this approach may introduce additional effort and complicate the overall design.



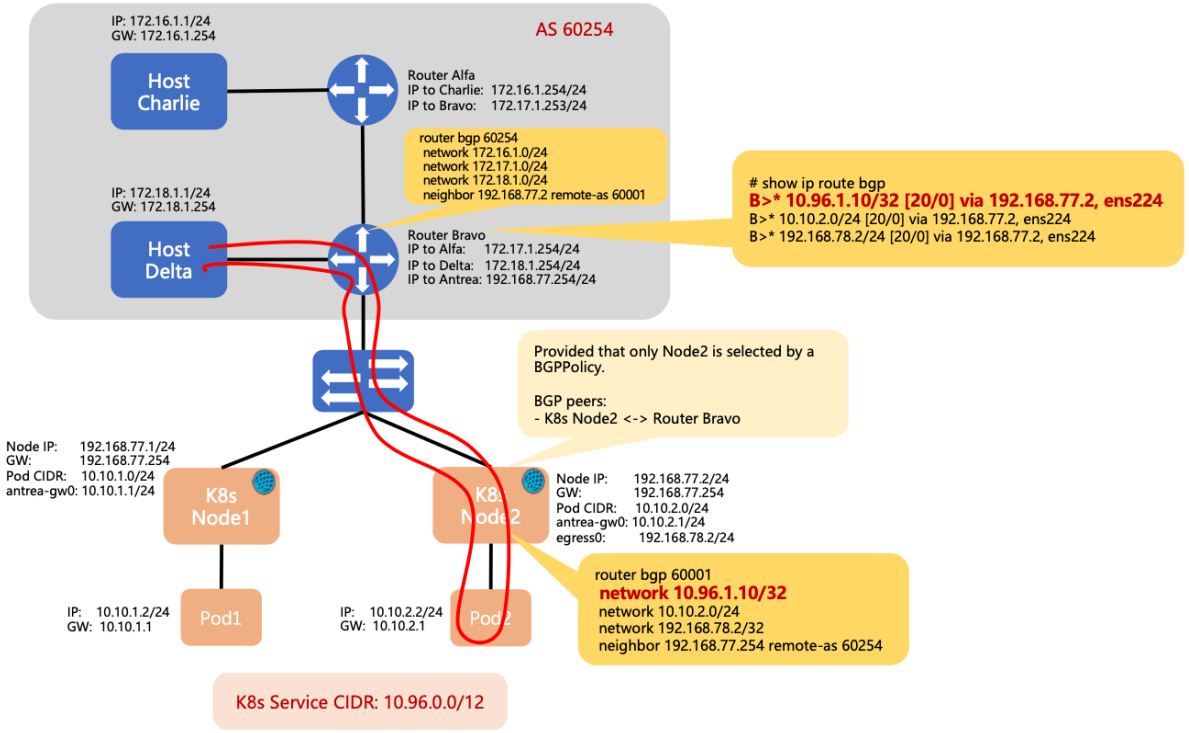
Service IPs can be advertised by multiple Nodes, leading to the installation of multiple equal-cost routes (Equal-cost multi-path, ECMP). However, it is essential to maintain session granularity in load balancing for BGP peers. This implies that

packets from a connection should consistently be forwarded to the same next hop (K8s Node). This is crucial because the initial packet of a connection might undergo SNAT on a specific K8s Node. If subsequent packets are forwarded to a different K8s Node, it could result in a connection broken.

Demo Cases

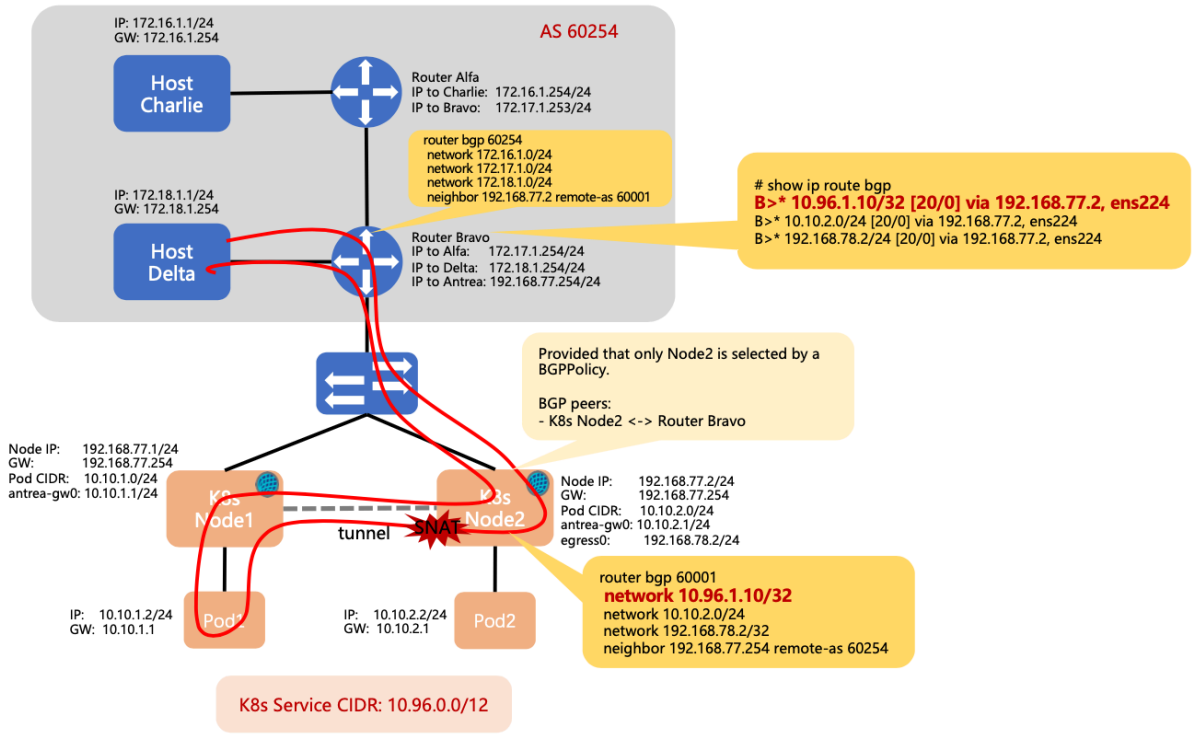
Host Delta -> Service with backend Pod2

Provided that only Node2 is selected by a BGPPolicy.
Service IPs are advertised by Node2, enabling host Delta to access the Services IPs via Node2. Since the Endpoint is just on Node2, SNAT is not needed.



Host Delta -> Service with backend Pod1 (encap)

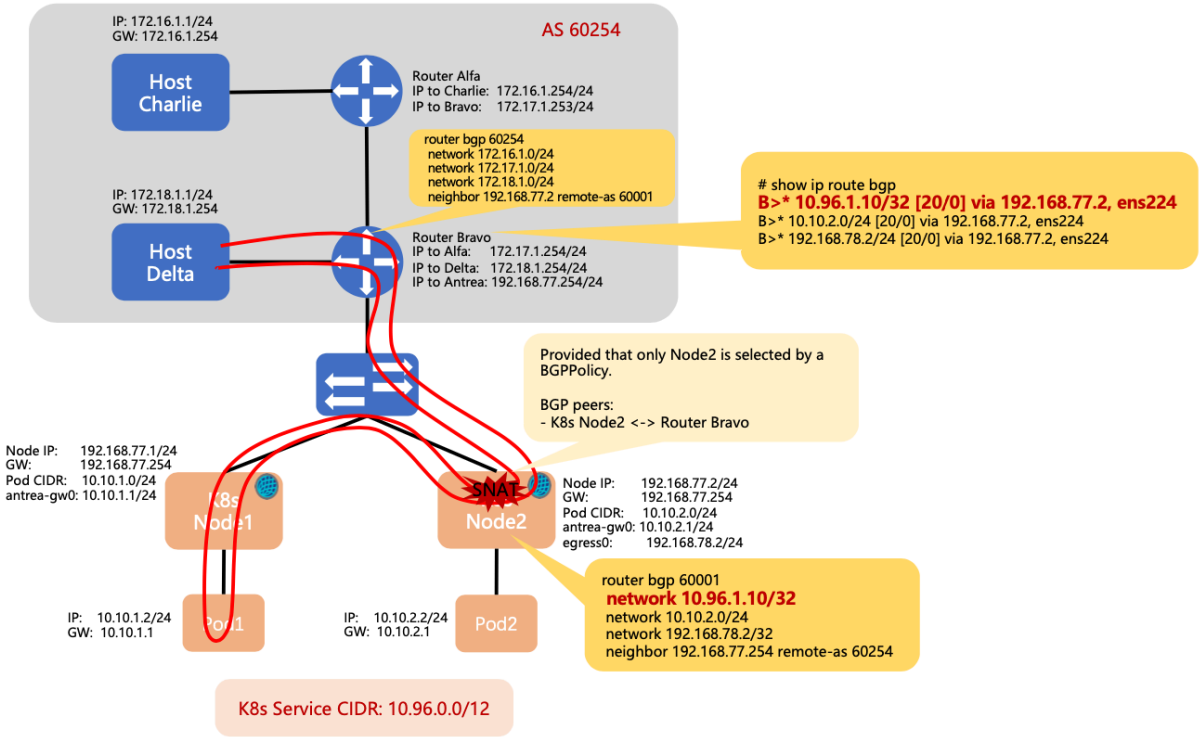
Provided that only Node2 is selected by a BGPPolicy.
Service IPs are advertised by Node2, enabling host Delta to access the Services IPs via Node2. Since the Endpoint is on Node1, SNAT is needed.
If a Service with externalTrafficPolicy Local and no local Endpoints, don't advertise the Service IPs of the Service. If the Service's externalTrafficPolicy is Local and has no local Endpoints on Node2, then the Service IPs will not be advertised on Node2.



Host Delta -> Service with backend Pod1 (noEncap)

Provided that only Node2 is selected by a BGPPolicy.

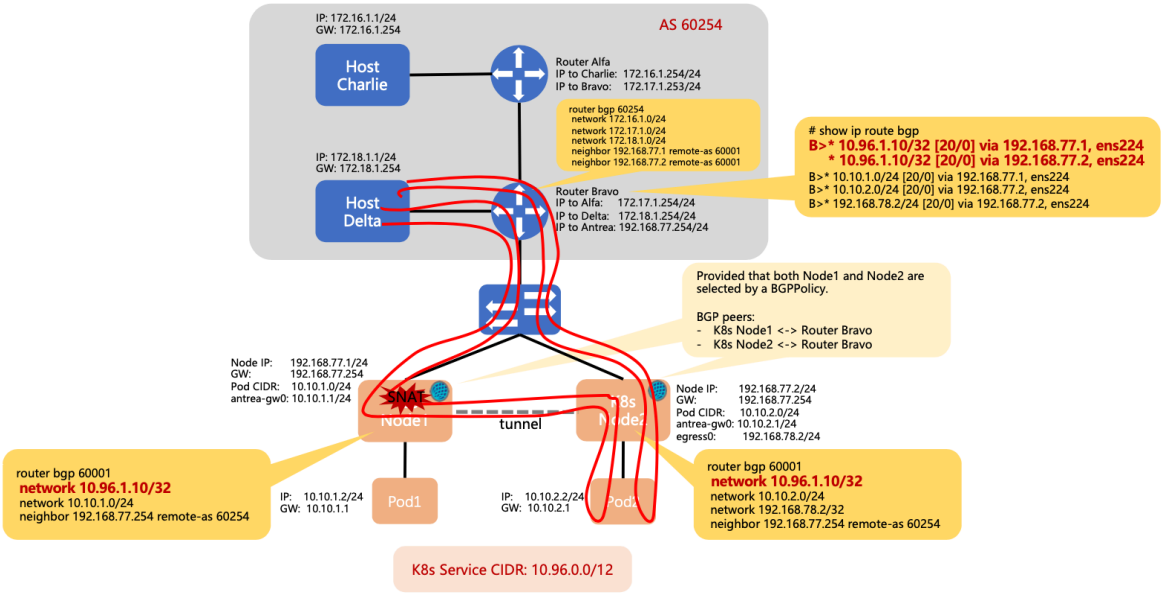
Service IPs are advertised by Node2, enabling host Delta to access the Services IPs via Node2. Since the Endpoint is on Node1, SNAT is needed.



Host Delta -> Service with backend Pod2

Provided that both Node1 and Node2 are selected by a BGPPolicy.

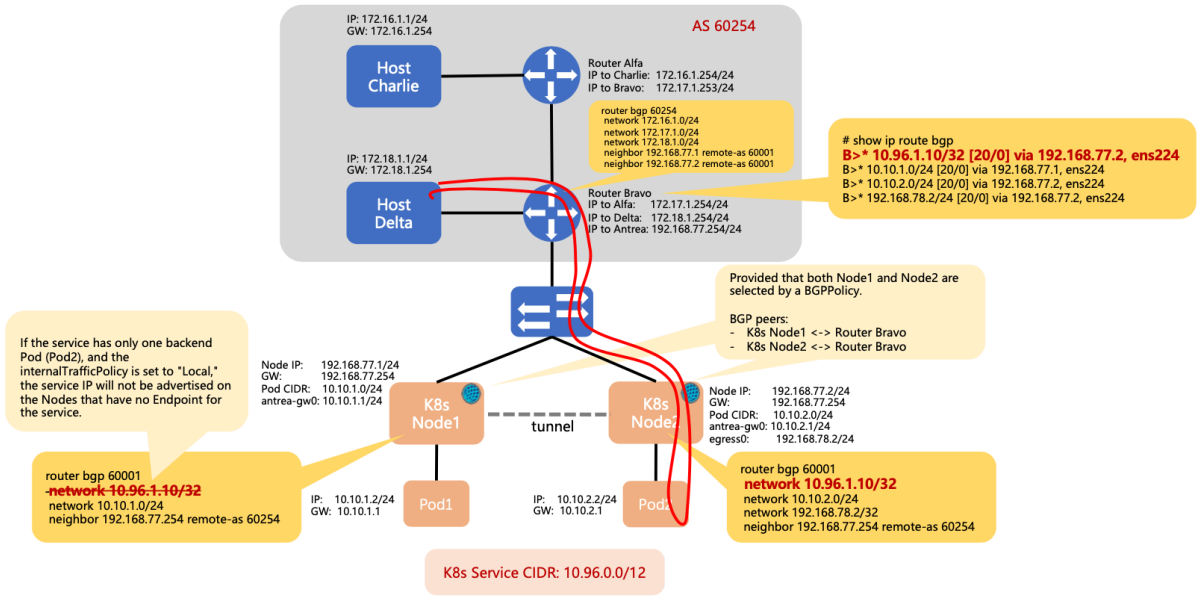
Service IPs are advertised by both Node1 and Node2, enabling host Delta to access the Services IPs through ECMP. Since the Endpoint is on Node2, when the first packet of a connection is forwarded to Node2, SNAT is not needed; when the first packet of a connection is forwarded to Node1, SNAT is needed. Note that, the subsequent packets of a connection should always be forwarded the same Node to avoid connection disruption. This might need some configurations in remote routers, maybe called something like “session sticky”.



Host Delta -> Service with backend Pod2, iTP/eTP == Local

Provided that both Node1 and Node2 are selected by a BGPPolicy.

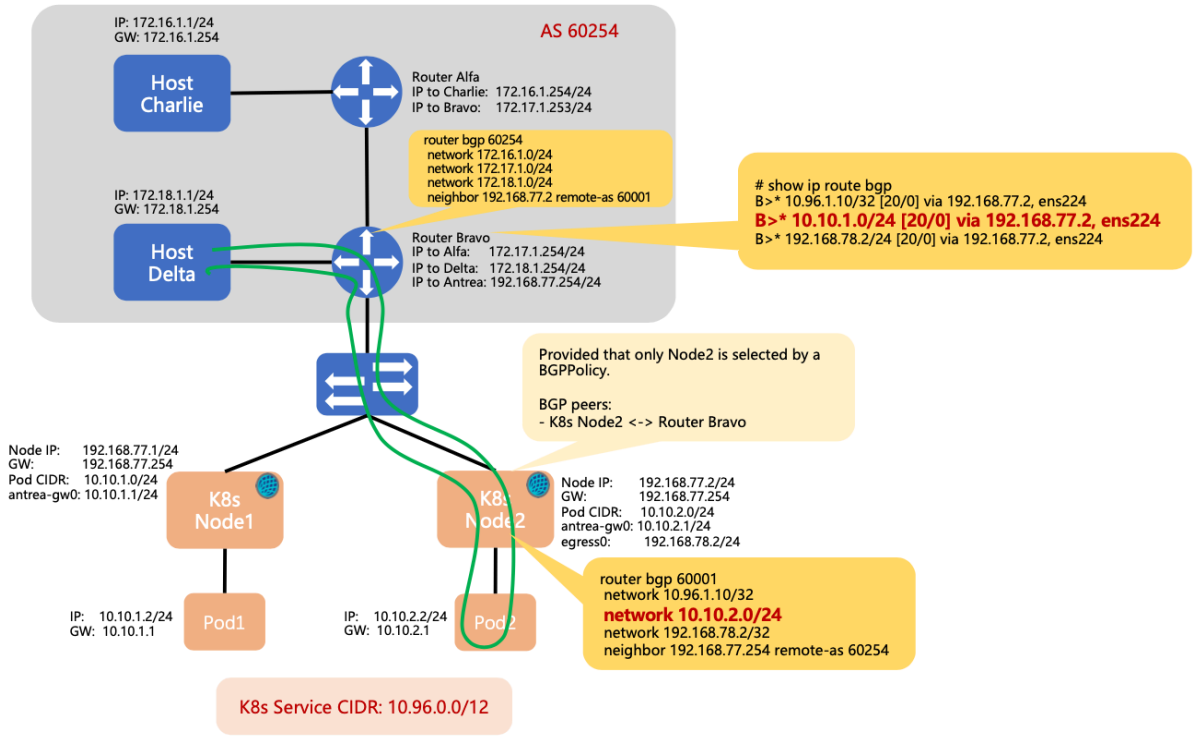
If the Service’s externalTrafficPolicy is Local and has no local Endpoints on Node1, then the Service IPs will not be advertised on Node1. The Service IPs are only advertised on Node2.



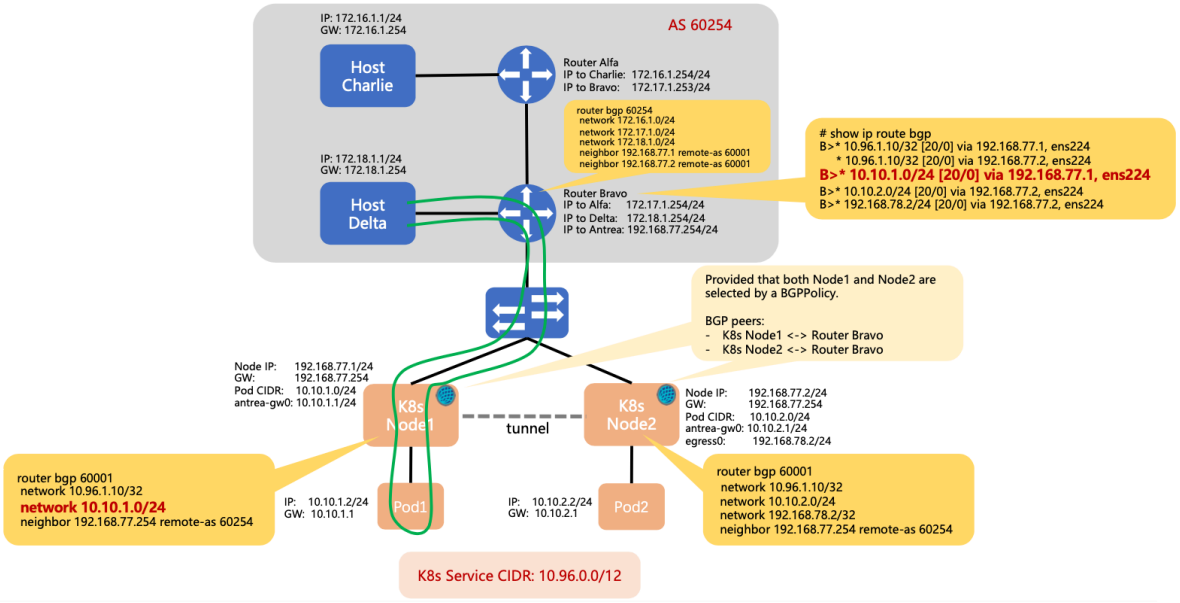
Host Delta -> Pod2

Provided that only Node2 is selected by a BGPPolicy.

Pod CIDR of Node2 is advertised by Node2, enabling host Delta to access the Pod IP via Node2. Like Node2, if Node1 is selected by a BGPPolicy, Pod CIDR of Node1 could be advertised by Node1.



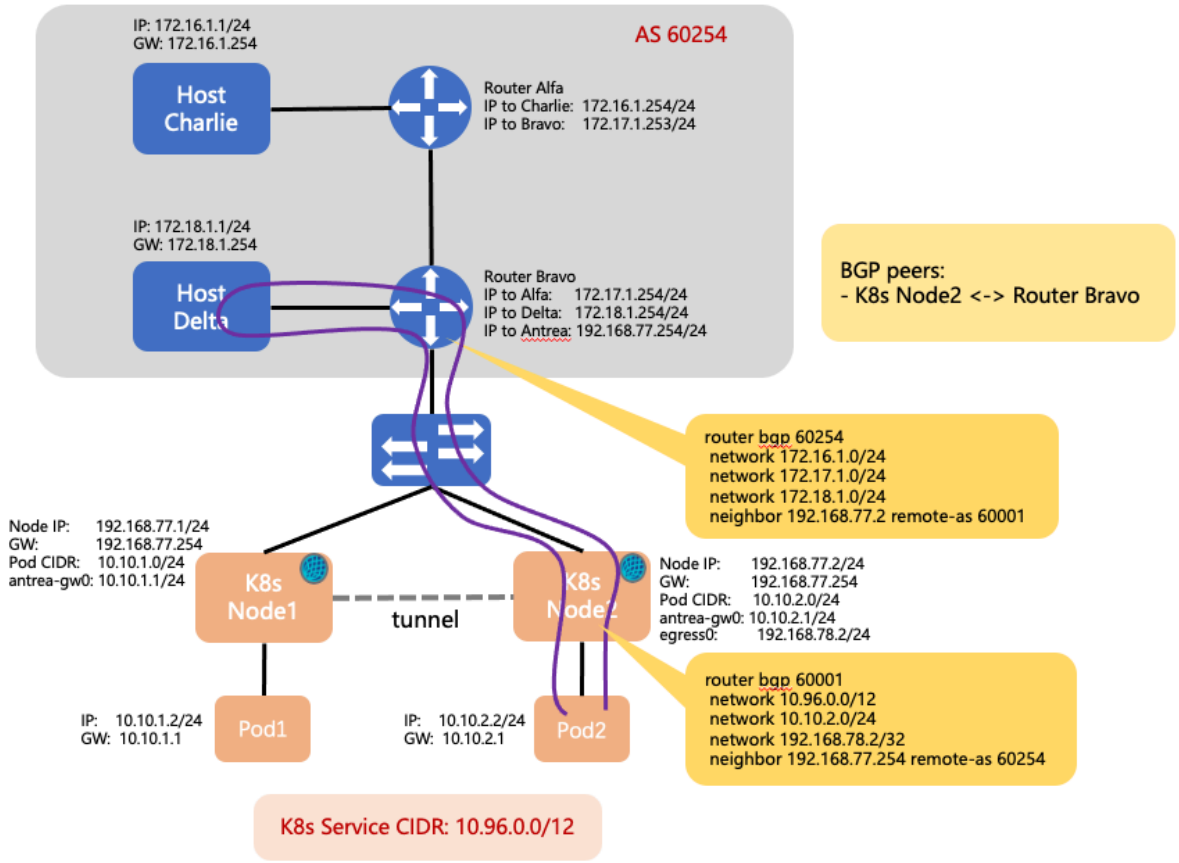
Host Delta -> Pod1



Pod2 -> Egress -> Host Delta

Provided that only Node2 is selected by a BGPPolicy.

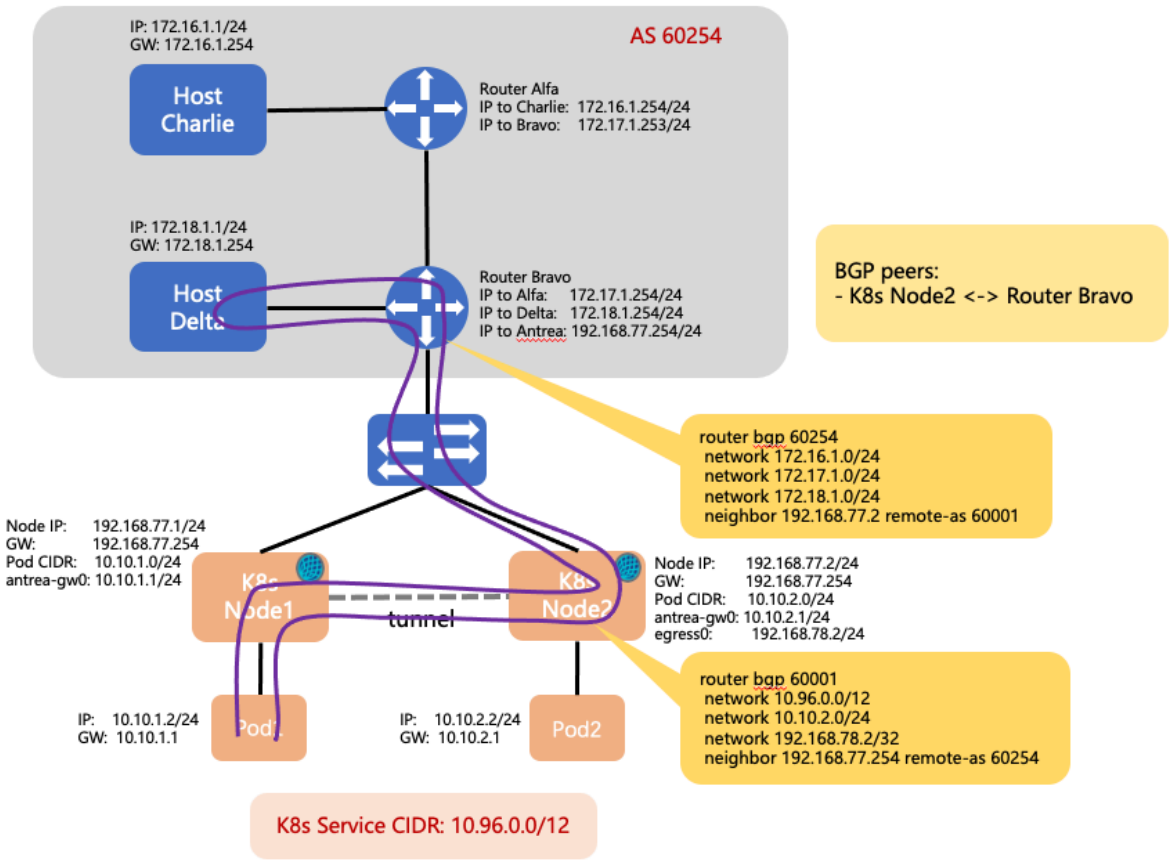
Egress IP on Node2 is advertised by Node2, enabling the reply packets of connection initiated from Pod2 and destined to host Delta can be forwarded back to Node2.



Pod1 -> Egress -> Host Delta

Provided that only Node2 is selected by a BGPPolicy.

Egress IP on Node2 is advertised by Node2, enabling the reply packets of connection initiated from Pod1 and destined to host Delta can be forwarded back to Node2.



Work Break Down

- CRD design
- code
- e2e test
- antctl