Switch CNI

Flannel to Calico

Introduction

When I first provided my Talos cluster, I didn't realize it comes with Flannel as the default CNI. At first, this wasn't a problem, but as I mature using Kubernetes, I quickly found I need more advanced network policies. These are the steps I took to convert a live cluster from Flannel to Calico with no downtime.

Preperation

Since this is a big move and a lot can go wrong, I went through several steps to ensure uptime.

1. Set up a test cluster to mirror your current setup. All the actions we are going to take can first be done there to ensure nothing pops up during conversion.
2. Snapshot all controllers (if possible) or do a backup of etcd

Convert the cluster

The steps are straightforward, and thankfully, Kubernetes is very good at extensibility, so adding dual CNI providers does not break networking.

First, let's create our namespace and set the proper permissions needed by Calico

```
kubectl create namespace tigera-operator
kubectl label namespace tigera-operator pod-security.kubernetes.io/enforce=privileged
```

Next, we will use Helm to install the provider. This can be done with regular manifests, but it will be easier to maintain and upgrade in the future with Helm. You may need to update the version.

```
helm install calico projectcalico/tigera-operator --version v3.29.3 --namespace tigera-operator
```

After some time, we can check to make sure that all pods are running and ready! This took a bit to finish so make sure its done before moving on.

```
kubectl get pods -n calico-system
```

Once that is complete, we can edit the default IP pool used by Calico. This is optional, just note the default is 192.168.0.0/16, which was not ideal for my network. Change the cidr field to what you require.

```
apiVersion: crd.projectcalico.org/v1
kind: IPPool
metadata:
  name: default-ipv4-ippool
spec:
  allowedUses:
  - Workload
  - Tunnel
  blockSize: 26
  cidr: 10.225.0.0/16
  ipipMode: Always
  natOutgoing: true
  nodeSelector: all()
  vxlanMode: Never
```

```
kubectl delete ippools.crd.projectcalico.org default-ipv4-ippool
kubectl apply -f ippool.yaml
```

The cluster should be in a stable state at this point. You will begin to see new pods with the updated IP range. The nodes may also have a secondary IP. This is ok. Now we can patch the controller nodes to remove the Flannel manifest.

```
cluster:
  network:
    cni:
      name: none
  proxy:
    disabled: true
```

```
talosctl patch mc --patch @cnipatch.yaml
```

Perform this step on all controllers. Then we can verify it worked by querying Talos controllers for their built-in manifest files. We do not want to see Flannel list anywhere.

```
talosctl get manifests -n <controller-ip>
```

Sample output:

```
NODE          NAMESPACE     TYPE      ID                          VERSION
192.168.249.4   controlplane   Manifest   00-kubelet-bootstrapping-token   1
192.168.249.4   controlplane   Manifest   01-csr-approver-role-binding     1
192.168.249.4   controlplane   Manifest   01-csr-node-bootstrap            1
192.168.249.4   controlplane   Manifest   01-csr-renewal-role-binding      1
192.168.249.4   controlplane   Manifest   11-core-dns                      1
192.168.249.4   controlplane   Manifest   11-core-dns-svc                  1
192.168.249.4   controlplane   Manifest   11-kube-config-in-cluster        1
```

The final step is to remove the flannel daemonset and configmap

```
kubectl delete daemonset -n kube-system kube-flannel
kubectl delete cm kube-flannel-cfg -n kube-system
```

Conclusion

You may notice that some pods still have the old cidr range. This is mostly okay. I found that networking for the most part still works. Its best to go through the namespaces and do a rollout restart on the various resources and they will pick up the new networking rules.

Revision #1
Created 2 May 2025 21:23:08 by Kory
Updated 2 May 2025 21:48:04 by Kory