Migrate from one IP pool to another

Big picture

Migrate pods from one IP pool to another on a running cluster without network disruption.

Value

Pods are assigned IP addresses from IP pools that you configure in Calico. As the number of pods increase, you may need to increase the number of addresses available for pods to use. Or, you may need to move pods from a CIDR that was used by mistake. Calico lets you migrate from one IP pool to another one on a running cluster without network disruption.

Features

This how-to guide uses the following Calico features:

- IPPool resource

Concepts

IP pools and cluster CIDRs

Calico supports using multiple disjoint IP pool CIDRs within the cluster. However, Kubernetes expects that all pods have addresses within the same cluster CIDR. This means that although it is technically feasible to create an IP pool outside of the cluster CIDR, we do not recommend it. Pods allocated addresses outside of the Kubernetes cluster CIDR will lose network connectivity.

Before you begin…

Verify that you are using Calico IPAM.

If you are not sure which IPAM your cluster is using, the way to tell depends on install method.

- Operator
- Manifest

The IPAM plugin can be queried on the default Installation resource.

```
kubectl get installation default -o go-template --template
```

```
{{.spec.cni.ipam.type}}
```

If your cluster is using Calico IPAM, the above command should return a result of `Calico`.

SSH to one of your Kubernetes nodes and examine the CNI configuration.

```
cat /etc/cni/net.d/10-calico.conflist
```

Look for the entry:

```
"ipam": {
    "type": "calico-ipam"
},
```

If it is present, you are using the Calico IPAM. If the IPAM is not Calico, or the 10-calico.conflist file does not exist, you cannot use these features in your cluster.

Verify orchestrator support for changing the pod network CIDR.

Although Kubernetes supports changing the pod network CIDR, not all orchestrators do. Check your orchestrator documentation to verify.

How to

Migrate from one IP pool to another

Follow these steps to migrate pods from one IP pool to another pool.

If you follow these steps, existing pod connectivity will not be affected. (If you delete the old IP pool before you create and verify the new pool, existing pods will be affected.) When pods are deleted, applications may be temporarily unavailable (depending on the type of application); plan accordingly.

1. Add a new IP pool.
   It is highly recommended that your Calico IP pools are within the Kubernetes cluster CIDR. If pods IPs are allocated from outside of the Kubernetes cluster CIDR, some traffic flows may have NAT applied unnecessarily causing unexpected behavior.
2. Disable the old IP pool.
   Disabling an IP pool only prevents new IP address allocations; it does not affect the networking of existing pods.
3. Delete pods from the old IP pool. This includes any new pods that may have been created with the old IP pool prior to disabling the pool.
4. Verify that new pods get an address from the new IP pool.
5. Delete the old IP pool.

Tutorial

In the following example, we created a Kubernetes cluster using **kubeadm**. But the IP pool CIDR we

configured (192.168.0.0/16) doesn't match the Kubernetes cluster CIDR. Let's change the CIDR to **10.0.0.0/16**, which for the purposes of this example falls within the cluster CIDR.

Let's run `calicoctl get ippool -o wide` to see the IP pool, **default-ipv4-ippool**.

```
    NAME                  CIDR             NAT    IPIPMODE   VXLANMODE
DISABLED
    default-ipv4-ippool   192.168.0.0/16   true   Always     Never
false
```

When we run `calicoctl get wep –all-namespaces`, we see that a pod is created using the default range (192.168.52.130/32).

```
    NAMESPACE     WORKLOAD                    NODE      NETWORKS
INTERFACE
    kube-system   coredns-6f4fd4bdf-8q7zp   vagrant   192.168.52.130/32
cali800a63073ed
```

Let's get started changing this pod to the new IP pool (10.0.0.0/16).

Step 1: Add a new IP pool

We add a new **IPPool** with the CIDR range, **10.0.0.0/16**.

```
    apiVersion: projectcalico.org/v3
    kind: IPPool
    metadata:
      name: new-pool
    spec:
      cidr: 10.0.0.0/16
      ipipMode: Always
      natOutgoing: true
```

Let's verify the new IP pool.

```
calicoctl get ippool -o wide
```

```
    NAME                  CIDR             NAT    IPIPMODE   DISABLED
    default-ipv4-ippool   192.168.0.0/16   true   Always     false
    new-pool              10.0.0.0/16      true   Always     false
```

Step 2: Disable the old IP pool

List the existing IP pool definition.

```
calicoctl get ippool -o yaml > pools.yaml
```

```
    apiVersion: projectcalico.org/v3
```

```
    items:
      - apiVersion: projectcalico.org/v3
        kind: IPPool
        metadata:
          name: default-ipv4-ippool
        spec:
          cidr: 192.0.0.0/16
          ipipMode: Always
          natOutgoing: true
      - apiVersion: projectcalico.org/v3
        kind: IPPool
        metadata:
          name: new-pool
        spec:
          cidr: 10.0.0.0/16
          ipipMode: Always
          natOutgoing: true
```

Edit pools.yaml.

Disable this IP pool by setting: `disabled: true`

```
    apiVersion: projectcalico.org/v3
    kind: IPPool
    metadata:
      name: default-ipv4-ippool
    spec:
      cidr: 192.0.0.0/16
      ipipMode: Always
      natOutgoing: true
      disabled: true
```

Apply the changes.

Remember, disabling a pool only affects new IP allocations; networking for existing pods is not affected.

```
calicoctl apply -f pools.yaml
```

Verify the changes.

```
calicoctl get ippool -o wide
```

```
    NAME                  CIDR             NAT    IPIPMODE   DISABLED
    default-ipv4-ippool   192.168.0.0/16   true   Always     true
    new-pool              10.0.0.0/16      true   Always     false
```

Step 3: Delete pods from the old IP pool

Next, we delete all of the existing pods from the old IP pool. (In our example, coredns is our only pod;

for multiple pods you would trigger a deletion for all pods in the cluster.)

```
kubectl delete pod -n kube-system coredns-6f4fd4bdf-8q7zp
```

Step 4: Verify that new pods get an address from the new IP pool

1. Create a test namespace and nginx pod.

   ```
   kubectl create ns ippool-test
   ```

2. Create an nginx pod.

   ```
   kubectl -n ippool-test create deployment nginx --image nginx
   ```

3. Verify that the new pod gets an IP address from the new range.

   ```
   kubectl -n ippool-test get pods -l app=nginx -o wide
   ```

4. Clean up the ippool-test namespace.

   ```
   kubectl delete ns ippool-test
   ```

Step 5: Delete the old IP pool

Now that you've verified that pods are getting IPs from the new range, you can safely delete the old pool.

```
calicoctl delete pool default-ipv4-ippool
```