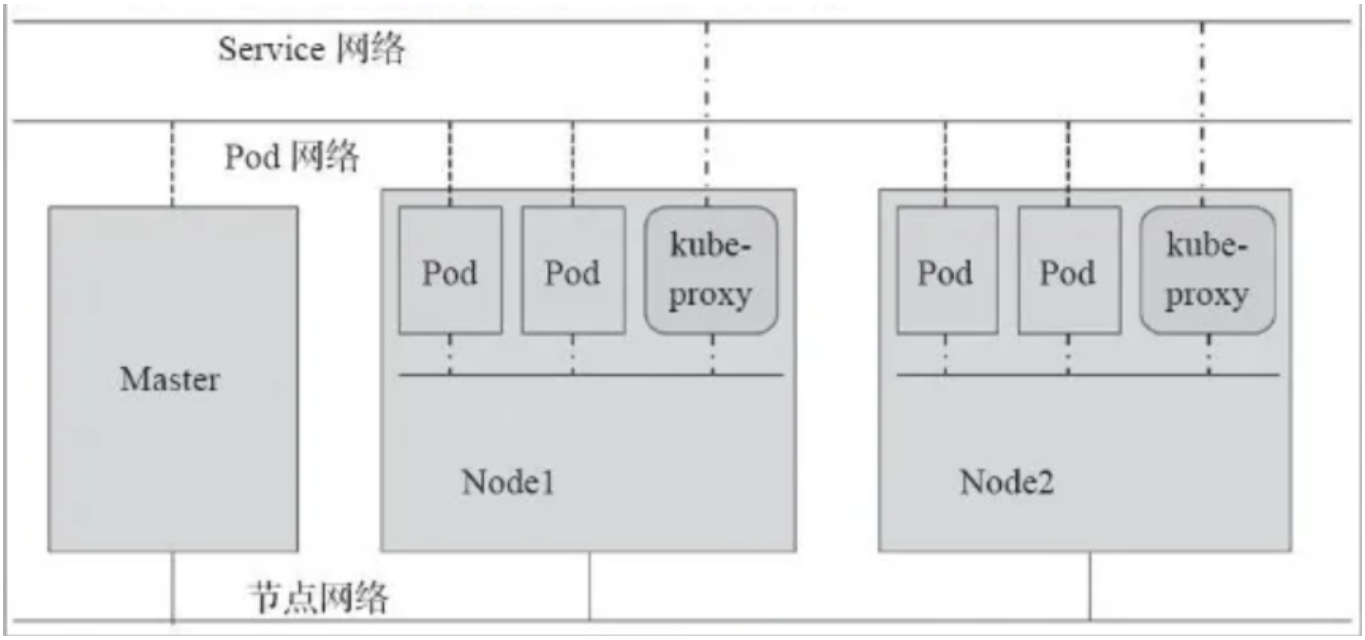
1，k8s网络架构



1.1 k8s存在三种网络，分别是：

- node网络
- service网络
- Pod网络

1.2 对应的三种IP，如下：

- Node IP：Node节点的IP地址，即物理网卡的IP地址。
- Pod IP：Pod的IP地址，即docker容器的IP地址，此为虚拟IP地址。

```
同Service下的pod可以直接根据PodIP相互通信
不同Service下的pod在集群间pod通信要借助于 cluster ip
pod和集群外通信，要借助于node ip
```

- Cluster IP：Service的IP地址，此为虚拟IP地址

```
Cluster IP仅仅作用于Kubernetes Service这个对象，并由Kubernetes管理和分配P地址
Cluster IP无法被ping，他没有一个“实体网络对象”来响应
Cluster IP只能结合Service Port组成一个具体的通信端口，单独的Cluster IP不具备通信的基础，并且他们属于Kubernetes集群这样一个封闭的空间。
在不同Service下的pod节点在集群间相互访问可以通过Cluster IP
```

以上三种ip中，Pod ip、Cluster ip为虚拟网络，在实际节点部署中，借助网络插件来实现，并且三种网络地址互不冲突。

2，生产环境k8s集群网络状况

2.1 节点数量和版本：

Master节点数量	Cpu node节点数量	Gpu node节点数量	Node节点总数
3	18	58	76

2.2 集群安装方式和网络插件

集群采用kubeadm开源工具安装，网络插件为calico

```
[root@k8s-master-01 ~]# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.7",
GitCommit:"132a687512d7fb058d0f5890f07d4121b3f0a2e2", GitTreeState:"clean", BuildDate:"2021-05-
12T12:38:16Z", GoVersion:"go1.15.12", Compiler:"gc", Platform:"linux/amd64"}

[root@k8s-master-01 ~]# kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.7",
GitCommit:"132a687512d7fb058d0f5890f07d4121b3f0a2e2", GitTreeState:"clean", BuildDate:"2021-05-
12T12:40:09Z", GoVersion:"go1.15.12", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.7",
GitCommit:"132a687512d7fb058d0f5890f07d4121b3f0a2e2", GitTreeState:"clean", BuildDate:"2021-05-
12T12:32:49Z", GoVersion:"go1.15.12", Compiler:"gc", Platform:"linux/amd64"}

[root@k8s-master-01 ~]# calicoctl version
Client Version:    v3.17.4
Git commit:        20c6a32a
Cluster Version:   v3.17.4
Cluster Type:      kubespray,bgp,kubeadm,kdd,k8s
```

2.3 现网Pod CIDR 和Cluster CIDR设置范围

```
#获取结果如下：
[root@k8s-master-01 ~]# kubeadm config view |grep Subnet
  podSubnet: 10.233.64.0/18
  serviceSubnet: 10.233.0.0/18


#node pod ip地址分配策略
[root@k8s-master-01 ~]# cat /etc/kubernetes/manifests/kube-controller-manager.yaml
    - --cluster-cidr=10.233.64.0/18  #Pod Ip CIDR范围
    - --cluster-name=cluster.local
    - --cluster-signing-cert-file=/etc/kubernetes/ssl/ca.crt
    - --cluster-signing-key-file=/etc/kubernetes/ssl/ca.key
    - --configure-cloud-routes=false
    - --controllers=*,bootstrapsigner,tokencleaner
    - --kubeconfig=/etc/kubernetes/controller-manager.conf
    - --leader-elect=true
    - --leader-elect-lease-duration=15s
    - --leader-elect-renew-deadline=10s
    - --node-cidr-mask-size=24 #node节点地址分配为24位掩码
#从规划看出，每个Node将从10.233.64.0/18地址段中分配一个24位掩码的地址，即每个node上Pod可用的地址数量为254
个
```

> 可以看到集群安装时规划的Pod Ip CIDR范围为：10.233.64.0/18；Cluster Ip CIDR范围为：10.233.0.0/18

根据Ip地址计算结果如下：

- Cluster Ip:

网络和IP地址计算器

显示网络，广播，第一次和最后一个给定的网络地址:

IP/掩码位: 10 | 233 | 0 | 0 / 18 | 计算 | 清除重算

结果结果:

可用地址:	16382			
掩码:	255	255	192	0
网络:	10	233	0	0
第一个可用:	10	233	0	1
最后可用:	10	233	63	254
广播:	10	233	63	255

- Pod Ip:

IP/掩码位: 10 | 233 | 64 | 0 / 18 | 计算 | 清除重算

结果结果:

可用地址:	16382			
掩码:	255	255	192	0
网络:	10	233	64	0
第一个可用:	10	233	64	1
最后可用:	10	233	127	254
广播:	10	233	127	255

- Cluster Ip可以使用的Ip地址范围为：10.233.0.1~10.233.63.254 共16382个可用地址；集群所需要service ip地址将从该地址池中获取；
- Pod Ip可以使用的Ip地址范围为：10.233.64.1~10.233.127.254，共有16382个可用地址；由于集群安装规划时定义每个node分配一个24位的地址段，第一个可用的地址段为：10.233.64.0/24，最后一个可用的地址段为：10.233.127.0/24，所以在安全情况管下可以支持的node数量为64台！
- Pod Ip地址不足的情况下，可能会导致以下的报错：

  error in getting result from AddNetworkList: failed to allocate for range 0: no IP addresses available in range set: 10.233.64.1-10.233.127.254

3,目前解决方案：

3.1 通过修改Pod Ip CIDR掩码范围来扩大地址可用段：

3.1.1 该种方案是目前所知是风险较低的方案，可以不用重启在网节点上的Pod，如将10.233.64.0/18改为10.233.64.1/16；但是对于本集群不适用：

- 本集群初始化时规划的Pod Ip和Cluster IP 由同一个ip段来划分；
- Pod Ip掩码范围扩大后，会导致Pod Ip地址会和Cluster Ip有重合部分，存在冲突，会导致部分Pod访问出错；

3.2 更改Pod Ip的地址，例如将本集群10.233.64.0/18修改为10.200.0.0/16后，可以最大支持256个24位的地址，即256个node节点。

> calico官方pod ip替换方案:
> https://projectcalico.docs.tigera.io/networking/migrate-pools

步骤如下：

3.2.1 确认您使用的是 Calico IPAM

```
SSH 到您的一个 Kubernetes 节点并检查 CNI 配置

cat /etc/cni/net.d/10-calico.conflist
         "ipam": {
              "type": "calico-ipam"
          },
```

生产节点k8s查询结果如下：

```
"plugins":[
    {
      "datastore_type": "kubernetes",
      "nodename": "k8s-master-01",
      "type": "calico",
      "log_level": "info",
      "log_file_path": "/var/log/calico/cni/cni.log",
      "ipam": {
        "type": "calico-ipam",
        "assign_ipv4": "true",
      }
生产节点使用的是Calico IPAM
```

3.2.2 添加一个新的ip池

注意：强烈建议您的 Calico IP 池位于 Kubernetes 集群 CIDR 内。如果 Pod IP 是从 Kubernetes 集群 CIDR 外部分配的，则某些流量可能会不必要地应用 NAT，从而导致意外行为。

3.2.3 禁用旧 IP 池。

注意：禁用 IP 池只会阻止新的 IP 地址分配；它不会影响现有 pod 的联网。

1. 从旧 IP 池中删除 pod。这包括在禁用池之前可能已使用旧 IP 池创建的任何新 pod。
2. 验证新 pod 是否从新 IP 池中获取地址。
3. 删除旧 IP 池。

3.3 calico官网更改Pod Ip教程示例步骤如下

在以下示例中，使用的kubeadm创建了一个 Kubernetes 集群。但是我们配置的 IP 池 CIDR（192.168.0.0/16）与 Kubernetes 集群 CIDR 不匹配。让我们将 CIDR 更改为10.0.0.0/16，出于本示例的目的，它属于集群 CIDR。

- 运行calicoctl get ippool -o wide以查看 IP 池default-ipv4-ippool。

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

- 当我们运行时calicoctl get wep --all-namespaces，我们看到使用默认范围 (192.168.52.130/32) 创建了一个 pod。

```
命名空间工作负载节点网络接口
kube-system coredns-6f4fd4bdf-8q7zp vagrant 192.168.52.130/32 cali800a63073ed
```

- 开始将此 pod 更改为新的 IP 池 (10.0.0.0/16)。

3.3.1 第 1 步：添加新的 IP 池

- 我们添加一个CIDR 范围为10.0.0.0/16的新IPPool。

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

- 验证新的 IP 池。

```
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
```

3.3.2 第 2 步：禁用旧 IP 池

- 列出现有的 IP 池定义

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

apiVersion: projectcalico.org/v3
```

```
    items:
    - apiVersion: projectcalico.org/v3
      kind: IPPool
      元数据:
        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
```

- 编辑 pools.yaml,通过设置禁用旧的 IP 池：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
```

- 通过apply命令让新的calico yaml文件生效
  禁用池只会影响新的 IP 分配；现有 pod 的网络不受影响。

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

- 验证更改:

```
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
```

3.3.3 第 3 步：从旧 IP 池中删除 pod

接下来，我们从旧 IP 池中删除所有现有的 Pod。（在我们的示例中，coredns是我们唯一的 pod；对于多个 pod，您将触发集群中所有 pod 的删除。）

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

3.3.4 第 4 步：验证新 pod 是否从新 IP 池中获取地址

1. 创建一个测试命名空间和 nginx pod。

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

2. 创建一个 nginx pod。

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

3. 验证新 pod 是否从新范围获取 IP 地址。

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

4. 清理 ippool-test 命名空间。

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

3.3.5 第 5 步：删除旧 IP 池

现在您已经验证 Pod 正在从新范围获取 IP，您可以安全地删除旧池。

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

以上为calico官网提供的操作步骤，但是现网环境比较复杂，节点数量和业务Pod较多，更换Pod Ip地址特别是需要重启业务的情况下，会面临更多的风险问题。

根据以上步骤搭建测试环境测试如下：

4,更换Pod CIDR测试

4.1 测试环境

- 节点数量：1Master+2Node

```
root@k8s-master1:/data# kubectl get nodes
NAME          STATUS   ROLES                  AGE     VERSION
k8s-master1   Ready    control-plane,master   2d14h   v1.20.15
k8s-node1     Ready    <none>                 18h     v1.20.15
k8s-node2     Ready    <none>                 18h     v1.20.15
```

- kubeadm版本：

```
root@k8s-master1:/data# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.15",
GitCommit:"8f1e5bf0b9729a899b8df86249b56e2c74aebc55", GitTreeState:"clean", BuildDate:"2022-01-
19T17:26:37Z", GoVersion:"go1.15.15", Compiler:"gc", Platform:"linux/amd64"}
```

- kubernetes版本

```
root@k8s-master1:/data# kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.15",
GitCommit:"8f1e5bf0b9729a899b8df86249b56e2c74aebc55", GitTreeState:"clean", BuildDate:"2022-01-
19T17:27:39Z", GoVersion:"go1.15.15", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.15",
GitCommit:"8f1e5bf0b9729a899b8df86249b56e2c74aebc55", GitTreeState:"clean", BuildDate:"2022-01-
19T17:23:01Z", GoVersion:"go1.15.15", Compiler:"gc", Platform:"linux/amd64"}
```

- calico版本

```
    root@k8s-master1:/data# calicoctl version
    Client Version:    v3.17.6
    Git commit:        a26c5fc6
    Cluster Version:   v3.17.6
    Cluster Type:      k8s,bgp,kubeadm,kdd
```

> 以上大版本号和生产环境保持一致

4.2安装初始化参数如下：

```
    kubeadm  init  --apiserver-advertise-address=192.168.66.128 --apiserver-bind-port=6443  --
kubernetes-version=v1.20.15  --pod-network-cidr=10.10.0.0/18 --service-cidr=172.31.0.0/18 --
service-dns-domain=cluster.local --image-repository=k8s.io.com/google_containers --ignore-
preflight-errors=swap --token-ttl=0
```

即初始化CIDR设置：

```
*   Pod CIDR:10.10.0.0/18
*   Cluster CIDR:172.31.0.0/18
```

初始化后查看：

```
#kubeadm config view |grep Subnet
 podSubnet: 10.10.0.0/18
  serviceSubnet: 172.31.0.0/18
```

> 计划将 podSubnet: 10.10.0.0/18替换为10.20.0.0/16 地址段

4.3 将 calicoctl 安装为 Kubernetes pod

4.3.1 下载calicoctl yaml文件

```
    $ kubectl apply -f https://docs.projectcalico.org/manifests/calicoctl.yaml
```

4.3.2 修改yaml中image版本为生产环境版本：calico/ctl:v3.17.6，apply生效：

```
# kubectl apply -f calicoctl.yaml

#设置别名：
$ alias calicoctl="kubectl exec -i -n kube-system calicoctl -- /calicoctl "

# calicoctl version
Client Version:    v3.17.6
Git commit:        a26c5fc6
Cluster Version:   v3.17.6
Cluster Type:      k8s,bgp,kubeadm,kdd
```

4.4 添加一个新的ip地址池

4.4.1 备份旧的ip pool yaml文件

```
calicoctl get ippool -o yaml > pool-old.yaml
```

```
#查看当前ippool状态
root@k8s-master1:/data# calicoctl get ippool -owide
NAME                  CIDR           NAT    IPIPMODE   VXLANMODE   DISABLED   SELECTOR
default-ipv4-ippool   10.10.0.0/18   true   Always     Never       false      all()
```

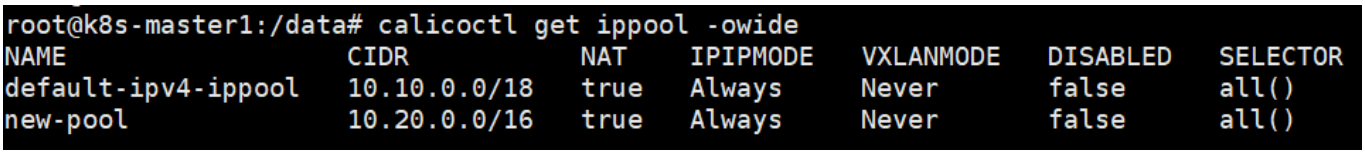
添加一个新的ip pool如下：

```
    calicoctl create -f -<<EOF
    apiVersion: projectcalico.org/v3
    kind: IPPool
    metadata:
      name: new-pool
    spec:
      cidr: 10.20.0.0/16
      ipipMode: Always
      natOutgoing: true
    EOF
```

```
root@k8s-master1:/data# calicoctl create -f -<<EOF
> apiVersion: projectcalico.org/v3
> kind: IPPool
> metadata:
>   name: new-pool
> spec:
>   cidr: 10.20.0.0/16
>   ipipMode: Always
>   natOutgoing: true
> EOF
Successfully created 1 'IPPool' resource(s)
```

4.4.2 目前有2个ip pools,查看如下：

```
root@k8s-master1:/data# calicoctl get ippool -owide
NAME                  CIDR           NAT    IPIPMODE   VXLANMODE   DISABLED   SELECTOR
default-ipv4-ippool   10.10.0.0/18   true   Always     Never       false      all()
new-pool              10.20.0.0/16   true   Always     Never       false      all()
```



4.5 关闭旧的ip pool

4.5.1 本分当前的ip pool文件

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

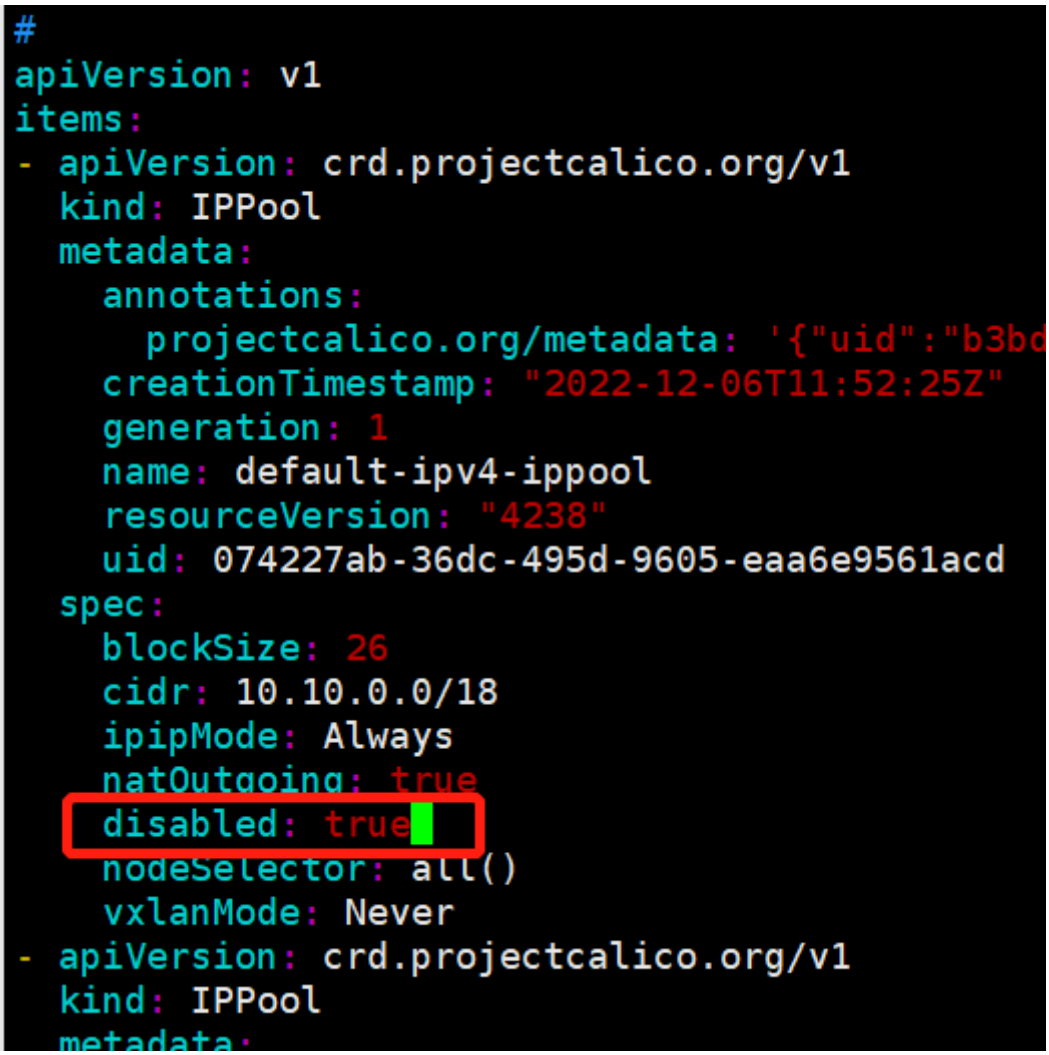
- pool.yaml文件如下：

```
root@k8s-master1:/data# cat pool.yaml
apiVersion: projectcalico.org/v3
items:
- apiVersion: projectcalico.org/v3
  kind: IPPool
  metadata:
    creationTimestamp: "2022-12-06T11:52:25Z"
    name: default-ipv4-ippool
    resourceVersion: "4238"
    uid: 074227ab-36dc-495d-9605-eaa6e9561acd
  spec:
    blockSize: 26
    cidr: 10.10.0.0/18
    ipipMode: Always
    natOutgoing: true
    nodeSelector: all()
    vxlanMode: Never
- apiVersion: projectcalico.org/v3
```

```
    kind: IPPool
    metadata:
      creationTimestamp: "2022-12-09T02:19:48Z"
      name: new-pool
      resourceVersion: "15142"
      uid: 24926f25-1a75-4180-a06d-27ccdd98522f
    spec:
      blockSize: 26
      cidr: 10.20.0.0/16
      ipipMode: Always
      natOutgoing: true
      nodeSelector: all()
      vxlanMode: Never
  kind: IPPoolList
  metadata:
    resourceVersion: "15380"
```

4.5.2 编辑yaml文件，将旧的ip pool即default-ipv4-ippool关闭,添加：disabled:true

```
  # kubectl edit ippool
  apiVersion: v1
  items:
  - apiVersion: crd.projectcalico.org/v1
    kind: IPPool
    metadata:
      annotations:
        projectcalico.org/metadata: '{"uid":"b3bd9d74-4de9-49f0-8338-
cef6aeefd766","creationTimestamp":"2022-12-06T11:52:25Z"}'
      creationTimestamp: "2022-12-06T11:52:25Z"
      generation: 1
      name: default-ipv4-ippool
      resourceVersion: "4238"
      uid: 074227ab-36dc-495d-9605-eaa6e9561acd
    spec:
      blockSize: 26
      cidr: 10.10.0.0/18
      ipipMode: Always
      natOutgoing: true
      disabled: true      #添加此行
      nodeSelector: all()
      vxlanMode: Never
```



> disable default-ipv4-ippool后，只影响新建pod的ip地址分配，不影响已有的pod

4.5.3 查看地址池的变化

```
    # calicoctl get ippool -o wide
    NAME                  CIDR           NAT    IPIPMODE   VXLANMODE   DISABLED   SELECTOR
    default-ipv4-ippool   10.10.0.0/18   true   Always     Never       true       all()
    new-pool              10.20.0.0/16   true   Always     Never       false      all()
```

```
root@k8s-master1:/data# calicoctl get ippool -o wide
NAME                  CIDR           NAT    IPIPMODE   VXLANMODE   DISABLED   SELECTOR
default-ipv4-ippool   10.10.0.0/18   true   Always     Never       true       all()
new-pool              10.20.0.0/16   true   Always     Never       false      all()
```

4.6 更改node节点pod CIDR参数

> 使用新的ip源范围覆盖旧的podCIDR，使用命令如下,如果节点数量较多，需要提前统计规划好node Pod CIDR地址：

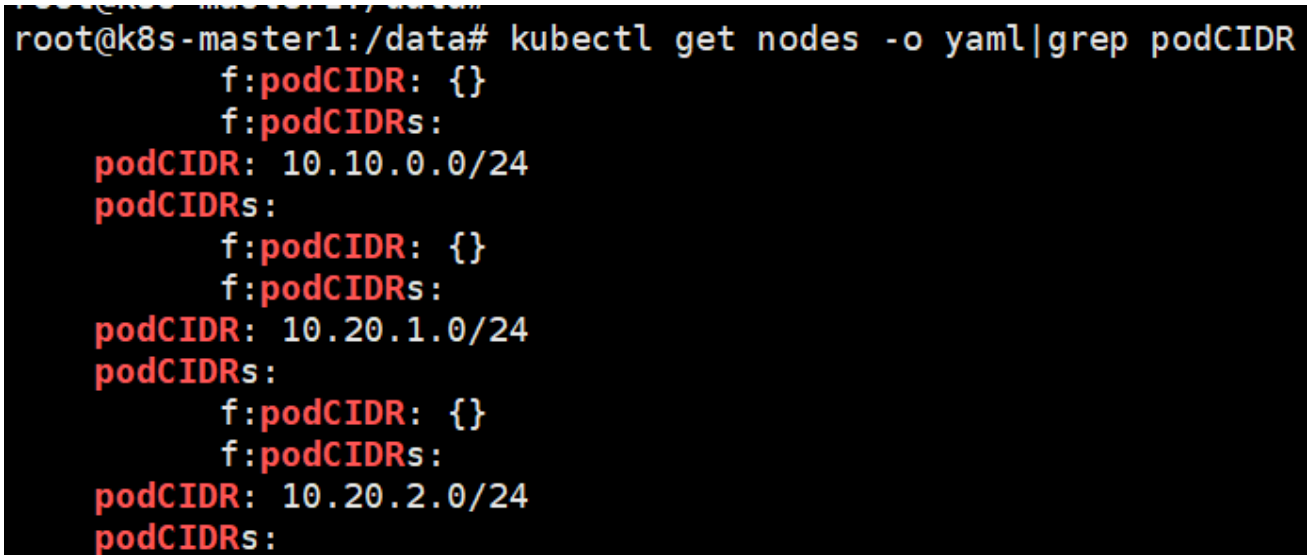
```
    kubectl get no k8s-node1 -o yaml > file1.yaml; sed -i "s~10.10.1.0/24~10.20.1.0/24~"
file1.yaml; kubectl delete no k8s-node1 && kubectl apply -f file1.yaml

    kubectl get no k8s-node2 -o yaml > file2.yaml; sed -i "s~10.10.2.0/24~10.20.2.0/24~"
file2.yaml; kubectl delete no k8s-node2 && kubectl apply -f file2.yaml
```

> 即先导出Node的yaml文件，修改里面pod CIDR;删除Node节点，最后通过修改后的文件将node重新加入

- 修改前：

- 修改后：



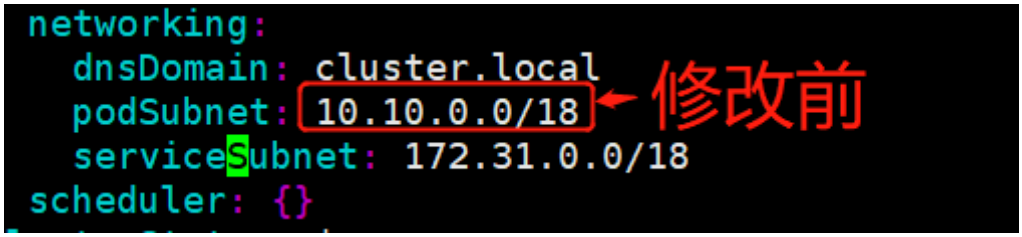
> 由于k8s master节点字段只读，所以无法直接修改k8s master节点node cidr

4.7 在master节点上，修改kubeadm-config ConfigMap 和 kube-controller-manager.yaml文件中的CIDR：

> 编辑 kubeadm-config ConfigMap 并将 podSubnet 更改为新的 IP 范围：

```
kubectl -n kube-system edit cm kubeadm-config
```

- 修改前：



- 修改后：

- 查看kue-system namespace下的pod状况：

```
root@k8s-master1:/data# kubectl get pods -n kube-system -w
NAME                                      READY   STATUS    RESTARTS   AGE
calico-kube-controllers-785f86bc75-ggmc2  1/1     Running   0          2d15h
calico-node-ggszf                         1/1     Running   0          2d15h
calico-node-hsdmf                         1/1     Running   0          2d15h
calico-node-sqbbb                         1/1     Running   0          2d15h
calicoctl                                 1/1     Running   0          38m
coredns-54d67798b7-69425                  1/1     Running   0          2d15h
coredns-54d67798b7-88j87                  1/1     Running   0          2d15h
etcd-k8s-master1                          1/1     Running   0          2d15h
kube-apiserver-k8s-master1                1/1     Running   0          2d15h
kube-controller-manager-k8s-master1       1/1     Running   0          2d15h
kube-proxy-4vxpx                          1/1     Running   0          2d15h
kube-proxy-g5zjv                          1/1     Running   0          2d15h
kube-proxy-smfhn                          1/1     Running   0          2d15h
kube-scheduler-k8s-master1                1/1     Running   0          2d15h
```

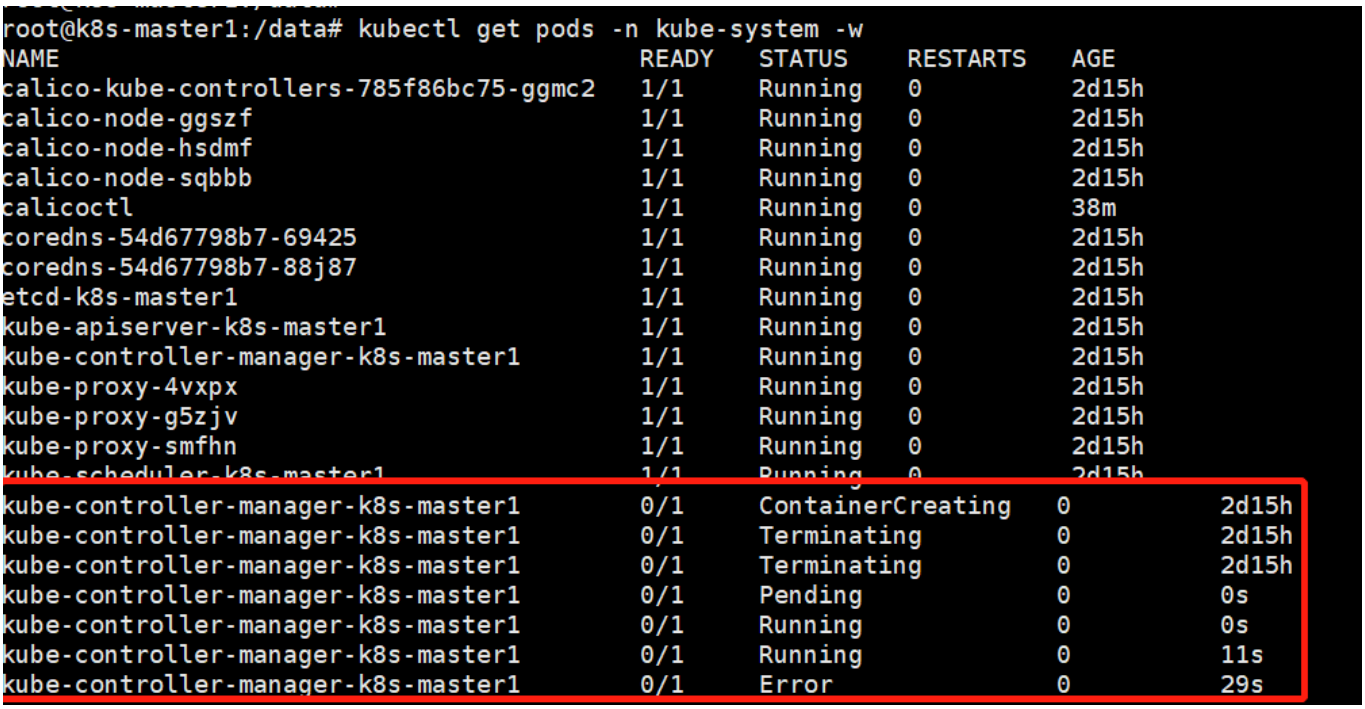
> 更改master节点 /etc/kubernetes/manifests/kube-controller-manager.yaml文件中的--cluster-cidr字段为新的Pod CIDR地址

原文件部分如下：

```
root@k8s-master1:~# cat /etc/kubernetes/manifests/kube-controller-manager.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    component: kube-controller-manager
    tier: control-plane
  name: kube-controller-manager
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-controller-manager
    - --allocate-node-cidrs=true
    - --authentication-kubeconfig=/etc/kubernetes/controller-manager.conf
    - --authorization-kubeconfig=/etc/kubernetes/controller-manager.conf
    - --bind-address=127.0.0.1
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --cluster-cidr=10.10.0.0/18  #修改此字段为10.20.0.0/16
    - --cluster-name=kubernetes
    - --cluster-signing-cert-file=/etc/kubernetes/pki/ca.crt
    - --cluster-signing-key-file=/etc/kubernetes/pki/ca.key
    - --controllers=*,bootstrapsigner,tokencleaner
    - --kubeconfig=/etc/kubernetes/controller-manager.conf
    - --leader-elect=true
    - --port=0
```

```
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
    - --root-ca-file=/etc/kubernetes/pki/ca.crt
    - --service-account-private-key-file=/etc/kubernetes/pki/sa.key
    - --service-cluster-ip-range=172.31.0.0/18
    - --use-service-account-credentials=true
```

> 修改保存后，Kube-system namespace下的kube-controller-manager-k8s-master1 pod会自动删除重建：



```
root@k8s-master1:/data# kubectl get pods -n kube-system -w
NAME                                       READY   STATUS              RESTARTS   AGE
calico-kube-controllers-785f86bc75-ggmc2   1/1     Running             0          2d15h
calico-node-ggszf                          1/1     Running             0          2d15h
calico-node-hsdmf                          1/1     Running             0          2d15h
calico-node-sqbbb                          1/1     Running             0          2d15h
calicoctl                                  1/1     Running             0          38m
coredns-54d67798b7-69425                   1/1     Running             0          2d15h
coredns-54d67798b7-88j87                   1/1     Running             0          2d15h
etcd-k8s-master1                           1/1     Running             0          2d15h
kube-apiserver-k8s-master1                 1/1     Running             0          2d15h
kube-controller-manager-k8s-master1        1/1     Running             0          2d15h
kube-proxy-4vxpx                           1/1     Running             0          2d15h
kube-proxy-g5zjv                           1/1     Running             0          2d15h
kube-proxy-smfhn                           1/1     Running             0          2d15h
kube-scheduler-k8s-master1                 1/1     Running             0          2d15h
kube-controller-manager-k8s-master1        0/1     ContainerCreating   0          2d15h
kube-controller-manager-k8s-master1        0/1     Terminating         0          2d15h
kube-controller-manager-k8s-master1        0/1     Terminating         0          2d15h
kube-controller-manager-k8s-master1        0/1     Pending             0          0s
kube-controller-manager-k8s-master1        0/1     Running             0          0s
kube-controller-manager-k8s-master1        0/1     Running             0          11s
kube-controller-manager-k8s-master1        0/1     Error               0          29s
```

> 多次测试到这个位置发现，修改后kube-controller-manager-k8s-master1这个pod会不断的重启，kube-controller-manager时负责控制器的创建和管理，Node上Pod CIDR的分配也是其负责。

- 检查日志如下：

```
# kubectl logs -f kube-controller-manager-k8s-master1 -n kube-system

日志中错误如下：
F1207 08:49:33.177703       1 controllermanager.go:250] error starting controllers: failed to mark
cidr[10.10.0.0/24] at idx [0] as occupied fo
r node: k8s-master1: cidr 10.10.0.0/24 is out the range of cluster cidr 10.20.0.0/16
goroutine 172 [running]:
```

5，测试结果

5.1 master节点kube-controller-manager配置文件直接修改会导致报错，目前多次测试无法通过，kubernetes官方社区暂时未找到相应的解决方法；

5.2 目前kubernets 官方slack有大量类似问题需求，但是官方没有给出相应的解决方案；

5.3 kubernetes官方从最新的v1.25版本开始支持multiple ClusterCIDRs，可以更改多个cidr段；

5.4 上面的修改方案如果前面可以正常执行，但是最后要重启或删除重建所有获取旧ip pool地址的Pod，生产环境业务较为复杂，存在极大的风险。

5.5 目前来看，使用kubeadm工具安装集群虽然方便，但是部署后集群参数难以修改，建议后续部署采用二进制方案或者开源的kubeasz来部署。

5.6 目前集群搭建缺少长远规划，地址和node port资源正常情况下需要尽可能的扩大范围，无论是calico还是flannel网络插件，默认的CIDR范围都是16位，而该集群安装时初始化修改为18位。