演示：Service创建通路

展示通过kubectl创建service，到ServiceController分配ClusterIP，返回给apiserver，最后通知proxy的流程

信息通路：kubectl->apiserver->ServiceController->apiserver->proxy

流程

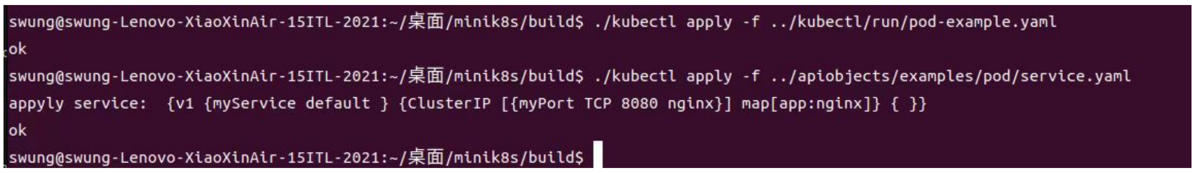
首先通过kubectl创建一个pod

```
➜  build git:(kubelet) ✗ ./kubectl apply -f pod-example.yaml
ok
➜  build git:(kubelet) ✗ cat pod-example.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod
  labels:
    app: nginx
spec:
  restartPolicy: Always
  containers:
    - name: viewer
      image: dplsming/nginx-fileserver:1.0
      ports:
        - containerPort: 80
          hostPort: 8888
      volumeMounts:
        - name: volume
          mountPath: /usr/share/nginx/html/files
    - name: downloader
      image: dplsming/aria2ng-downloader:1.0
      ports:
        - name: nginx
          containerPort: 6800
          hostPort: 6800
        - name: nginx
          containerPort: 6880
          hostPort: 6880
      volumeMounts:
        - name: volume
          mountPath: /data
  volumes:
    - name: volume
      hostPath:
        path: /pod
    - name: nfs-volume
      nfs:
        path: /exports
        server: 192.168.10.1
```

然后再创建一个Service

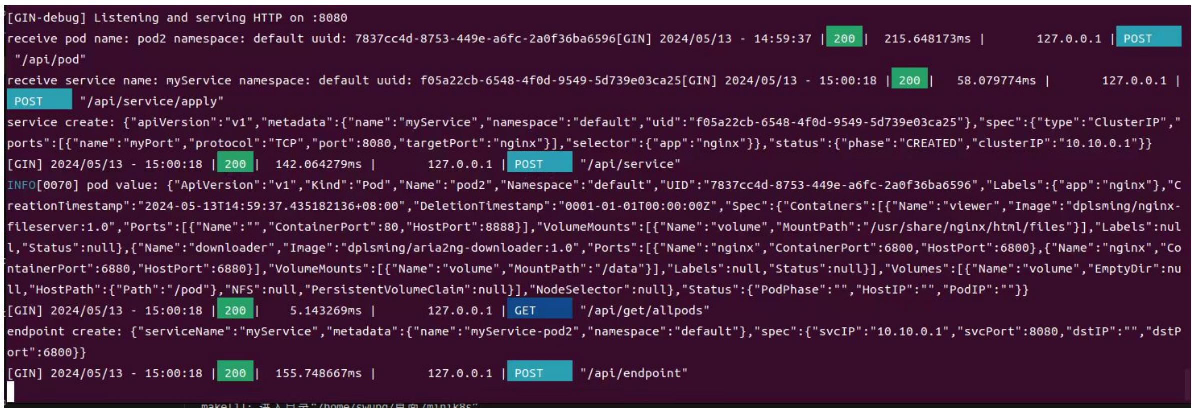
```
→  build git:(kubelet) ✗ ./kubectl apply -f service.yaml
ok
→  build git:(kubelet) ✗ cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: myService
  namespace: default
spec:
  selector:
    app: nginx
  type: ClusterIP
  ports:
    - name: myPort
      protocol: TCP
      port: 8080 # 对外暴露的端口
      targetPort: nginx # 转发的端口的名字，pod对应的端口名字
```

- apiserver收到http请求，将service信息告知ServiceController，并等待ServiceController分配ClusterIP
- ServiceController收到service创建的通知，根据策略，分配ClusterIP，然后筛选满足条件的pod，并生成对应的endpoints信息，将带有分配好的ClusterIP的service信息以及筛选出的endpoints信息回传给apiserver
- apiserver收到带有分配好的ClusterIP的service信息以及筛选出的endpoints信息，将其存入etcd，并发布service创建的通知
- proxy监听到service创建的通知，调用ipvs指令创建service及其对应的endpoints



```
swung@swung-Lenovo-XiaoXinAir-15ITL-2021:~/桌面/minik8s/build$ ./kubectl apply -f ../kubectl/run/pod-example.yaml
ok
swung@swung-Lenovo-XiaoXinAir-15ITL-2021:~/桌面/minik8s/build$ ./kubectl apply -f ../apiobjects/examples/pod/service.yaml
applyy service:  {v1 {myService default } {ClusterIP [{myPort TCP 8080 nginx}] map[app:nginx]} { }}
ok
swung@swung-Lenovo-XiaoXinAir-15ITL-2021:~/桌面/minik8s/build$
```

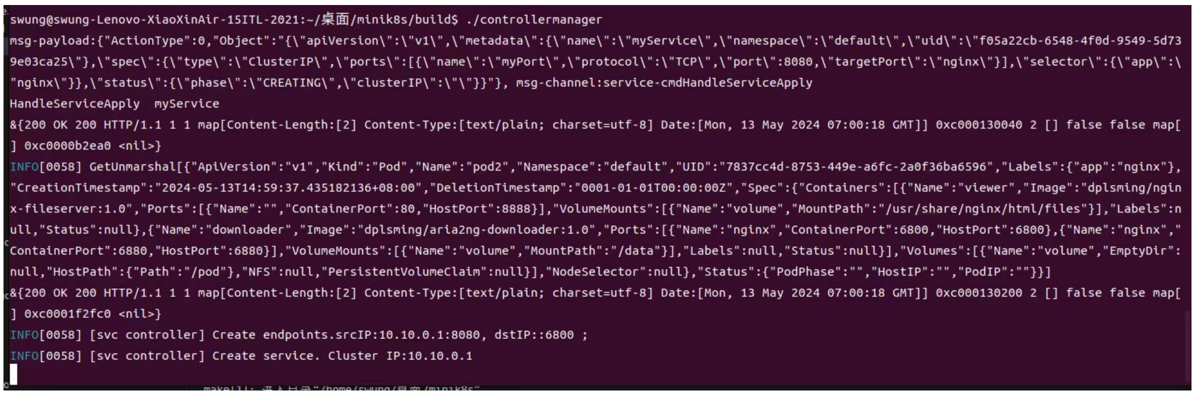
上图为使用kubectl分别创建一个pod和service



```
[GIN-debug] Listening and serving HTTP on :8080
receive pod name: pod2 namespace: default uuid: 7837cc4d-8753-449e-a6fc-2a0f36ba6596[GIN] 2024/05/13 - 14:59:37 | 200 |  215.648173ms |       127.0.0.1 | POST
 "/api/pod"
receive service name: myService namespace: default uuid: f05a22cb-6548-4f0d-9549-5d739e03ca25[GIN] 2024/05/13 - 15:00:18 | 200 |   58.079774ms |       127.0.0.1 |
 POST     "/api/service/apply"
service create: {"apiVersion":"v1","metadata":{"name":"myService","namespace":"default","uid":"f05a22cb-6548-4f0d-9549-5d739e03ca25"},"spec":{"type":"ClusterIP","
ports":[{"name":"myPort","protocol":"TCP","port":8080,"targetPort":"nginx"}],"selector":{"app":"nginx"}},"status":{"phase":"CREATED","clusterIP":"10.10.0.1"}}
[GIN] 2024/05/13 - 15:00:18 | 200 |  142.064279ms |       127.0.0.1 | POST     "/api/service"
INFO[0070] pod value: {"ApiVersion":"v1","Kind":"Pod","Name":"pod2","Namespace":"default","UID":"7837cc4d-8753-449e-a6fc-2a0f36ba6596","Labels":{"app":"nginx"},"C
reationTimestamp":"2024-05-13T14:59:37.435182136+08:00","DeletionTimestamp":"0001-01-01T00:00:00Z","Spec":{"Containers":[{"Name":"viewer","Image":"dplsming/nginx-
fileserver:1.0","Ports":[{"Name":"","ContainerPort":80,"HostPort":8888}],"VolumeMounts":[{"Name":"volume","MountPath":"/usr/share/nginx/html/files"}],"Labels":nul
l,"Status":null},{"Name":"downloader","Image":"dplsming/aria2ng-downloader:1.0","Ports":[{"Name":"nginx","ContainerPort":6800,"HostPort":6800},{"Name":"nginx","Co
ntainerPort":6880,"HostPort":6880}],"VolumeMounts":[{"Name":"volume","MountPath":"/data"}],"Labels":null,"Status":null}],"Volumes":[{"Name":"volume","EmptyDir":nu
ll,"HostPath":{"Path":"/pod"},"NFS":null,"PersistentVolumeClaim":null}],"NodeSelector":null},"Status":{"PodPhase":"","HostIP":"","PodIP":""}}
[GIN] 2024/05/13 - 15:00:18 | 200 |    5.143269ms |       127.0.0.1 | GET      "/api/get/allpods"
endpoint create: {"serviceName":"myService","metadata":{"name":"myService-pod2","namespace":"default"},"spec":{"svcIP":"10.10.0.1","svcPort":8080,"dstIP":"","dstP
ort":6800}}
[GIN] 2024/05/13 - 15:00:18 | 200 |  155.748667ms |       127.0.0.1 | POST     "/api/endpoint"
```

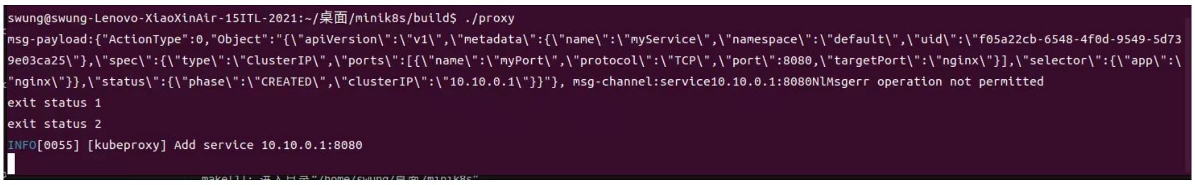
上图显示apiserver首先从kubectl收到“/api/service/apply”的消息，然后将消息告知ServiceController后，ServiceController分配ClusterIP之后，通过“/api/service“再将带有分配好的ClusterIP的service信息回传给apiserver，同时，ServiceController还要筛选满足条件的pod，ServiceController首先通过”/api/get/allpods“从apiserver拿到所有的pod信息，然后进行筛选，并生成对应的endpoints信息，

最后通过"api/endpoint"将筛选出的endpoints信息回传给apiserver



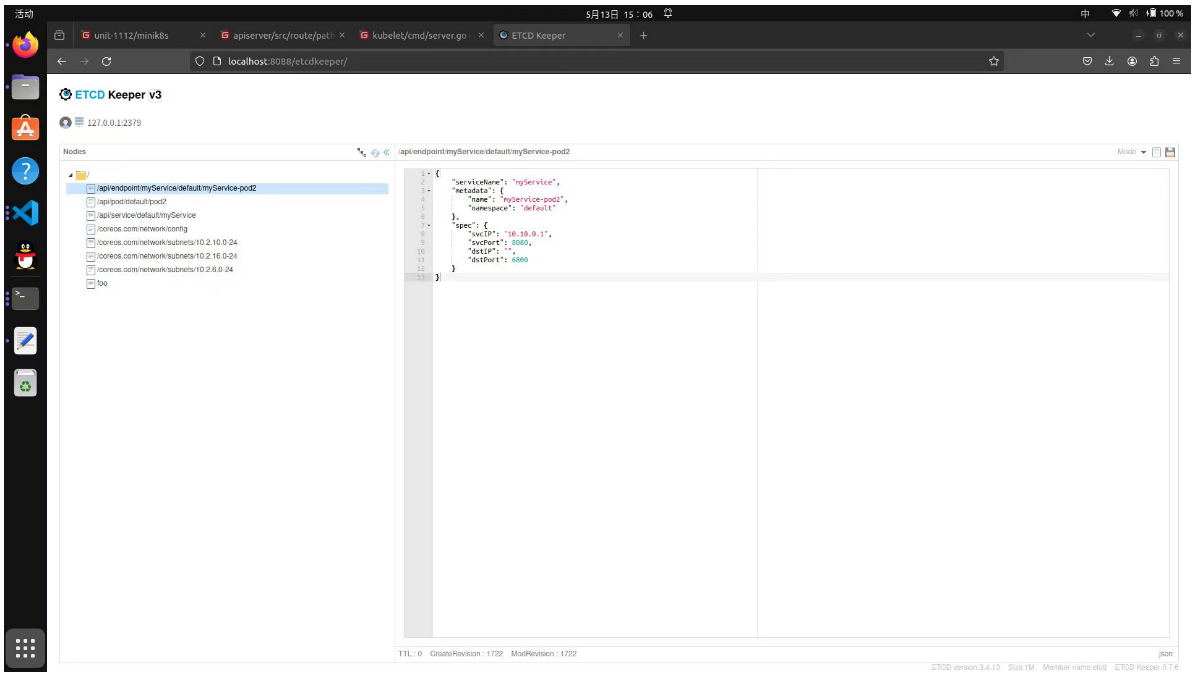
```
swung@swung-Lenovo-XiaoXinAir-15ITL-2021:~/桌面/minik8s/build$ ./controllermanager
msg-payload:{"ActionType":0,"Object":"{\"apiVersion\":\"v1\",\"metadata\":{\"name\":\"myService\",\"namespace\":\"default\",\"uid\":\"f05a22cb-6548-4f0d-9549-5d73
9e03ca25\"},\"spec\":{\"type\":\"ClusterIP\",\"ports\":[{\"name\":\"myPort\",\"protocol\":\"TCP\",\"port\":8080,\"targetPort\":\"nginx\"}],\"selector\":{\"app\":\
"nginx\"}},\"status\":{\"phase\":\"CREATING\",\"clusterIP\":\"\"}}"}, msg-channel:service-cmdHandleServiceApply
HandleServiceApply  myService
&{200 OK 200 HTTP/1.1 1 1 map[Content-Length:[2] Content-Type:[text/plain; charset=utf-8] Date:[Mon, 13 May 2024 07:00:18 GMT]] 0xc000130040 2 [] false false map[
] 0xc0000b2ea0 <nil>}
INFO[0058] GetUnmarshal[{"ApiVersion":"v1","Kind":"Pod","Name":"pod2","Namespace":"default","UID":"7837cc4d-8753-449e-a6fc-2a0f36ba6596","Labels":{"app":"nginx"},
"CreationTimestamp":"2024-05-13T14:59:37.435182136+08:00","DeletionTimestamp":"0001-01-01T00:00:00Z","Spec":{"Containers":[{"Name":"viewer","Image":"dplsming/ngin
x-fileserver:1.0","Ports":[{"Name":"","ContainerPort":80,"HostPort":8888}],"VolumeMounts":[{"Name":"volume","MountPath":"/usr/share/nginx/html/files"}],"Labels":n
ull,"Status":null},{"Name":"downloader","Image":"dplsming/aria2ng-downloader:1.0","Ports":[{"Name":"nginx","ContainerPort":6800,"HostPort":6800},{"Name":"nginx","
ContainerPort":6880,"HostPort":6880}],"VolumeMounts":[{"Name":"volume","MountPath":"/data"}],"Labels":null,"Status":null}],"Volumes":[{"Name":"volume","EmptyDir":
null,"HostPath":{"Path":"/pod"},"NFS":null,"PersistentVolumeClaim":null}],"NodeSelector":null},"Status":{"PodPhase":"","HostIP":"","PodIP":""}}]
&{200 OK 200 HTTP/1.1 1 1 map[Content-Length:[2] Content-Type:[text/plain; charset=utf-8] Date:[Mon, 13 May 2024 07:00:18 GMT]] 0xc000130200 2 [] false false map[
] 0xc0001f2fc0 <nil>}
INFO[0058] [svc controller] Create endpoints.srcIP:10.10.0.1:8080, dstIP::6800 ;
INFO[0058] [svc controller] Create service. Cluster IP:10.10.0.1
```

上图显示ServiceController为新创建的service分配的ClusterIP以及筛选出的满足条件的pod对应创建的endpoint信息



```
swung@swung-Lenovo-XiaoXinAir-15ITL-2021:~/桌面/minik8s/build$ ./proxy
msg-payload:{"ActionType":0,"Object":"{\"apiVersion\":\"v1\",\"metadata\":{\"name\":\"myService\",\"namespace\":\"default\",\"uid\":\"f05a22cb-6548-4f0d-9549-5d73
9e03ca25\"},\"spec\":{\"type\":\"ClusterIP\",\"ports\":[{\"name\":\"myPort\",\"protocol\":\"TCP\",\"port\":8080,\"targetPort\":\"nginx\"}],\"selector\":{\"app\":\
"nginx\"}},\"status\":{\"phase\":\"CREATED\",\"clusterIP\":\"10.10.0.1\"}}"}, msg-channel:service10.10.0.1:8080NlMsgerr operation not permitted
exit status 1
exit status 2
INFO[0055] [kubeproxy] Add service 10.10.0.1:8080
```

上图显示proxy创建对应的service



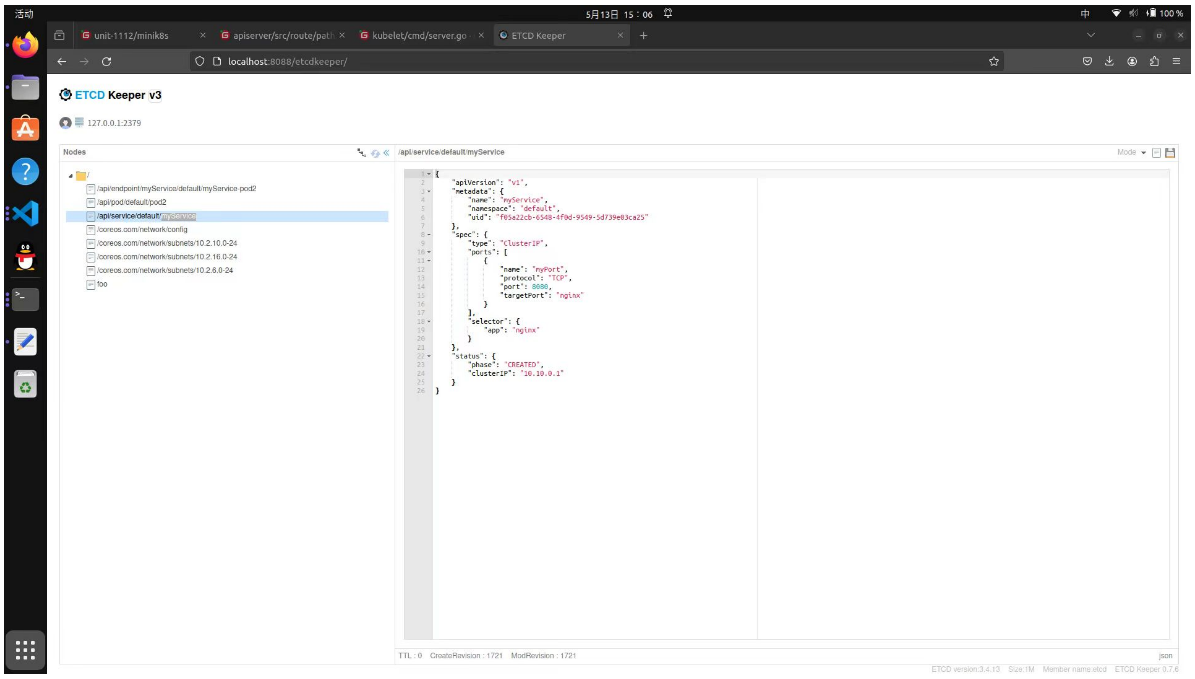
```
/
/api/endpoint/myService/default/myService-pod2
/api/pod/default/pod2
/api/service/default/myService
/coreos.com/network/config
/coreos.com/network/subnets/10.2.10.0-24
/coreos.com/network/subnets/10.2.16.0-24
/coreos.com/network/subnets/10.2.6.0-24
foo
```

/api/endpoint/myService/default/myService-pod2

```
{
    "serviceName": "myService",
    "metadata": {
        "name": "myService-pod2",
        "namespace": "default"
    },
    "spec": {
        "svcIP": "10.10.0.1",
        "svcPort": 8080,
        "dstIP": "",
        "dstPort": 6800
    }
}
```

TTL : 0   CreateRevision : 1722   ModRevision : 1722   json

ETCD version:3.4.13   Size:1M   Member name:etcd   ETCD Keeper 0.7.6



ETCD Keeper v3

127.0.0.1:2379

Nodes

```
/
  /api/endpoint/myService/default/myService-pod2
  /api/pod/default/pod2
  /api/service/default/myService
  /coreos.com/network/config
  /coreos.com/network/subnets/10.2.10.0-24
  /coreos.com/network/subnets/10.2.16.0-24
  /coreos.com/network/subnets/10.2.6.0-24
  foo
```

/api/service/default/myService

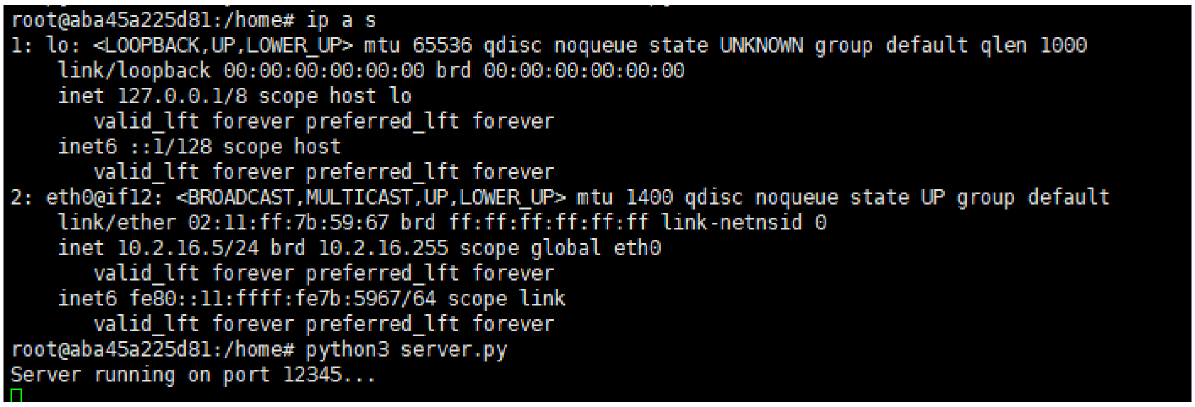
```
{
    "apiVersion": "v1",
    "metadata": {
        "name": "myService",
        "namespace": "default",
        "uid": "f05a22cb-6548-4f0d-9549-5d739e03ca25"
    },
    "spec": {
        "type": "ClusterIP",
        "ports": [
            {
                "name": "myPort",
                "protocol": "TCP",
                "port": 8080,
                "targetPort": "nginx"
            }
        ],
        "selector": {
            "app": "nginx"
        }
    },
    "status": {
        "phase": "CREATED",
        "clusterIP": "10.10.0.1"
    }
}
```

TTL : 0   CreateRevision : 1721   ModRevision : 1721   json

ETCD version:3.4.13   Size:1M   Member name:etcd   ETCD Keeper 0.7.6

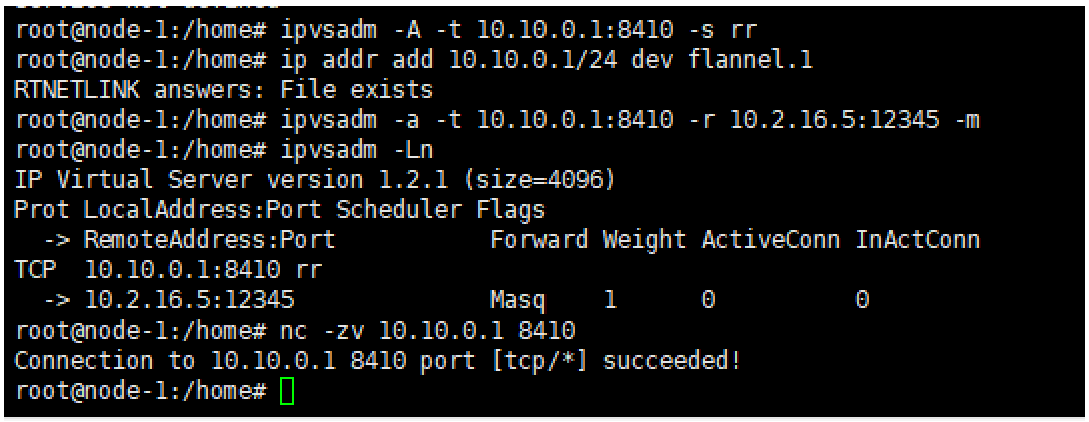
etcd中对应的service和endpoint信息

在服务器中测试ClusterIP

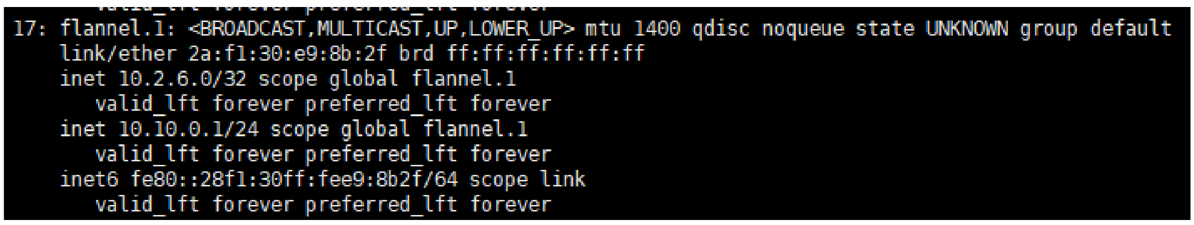


```
root@aba45a225d81:/home# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state UP group default
    link/ether 02:11:ff:7b:59:67 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.2.16.5/24 brd 10.2.16.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::11:ffff:fe7b:5967/64 scope link
       valid_lft forever preferred_lft forever
root@aba45a225d81:/home# python3 server.py
Server running on port 12345...
```

在node3节点上起了一个container，在12345端口上开启一个服务



```
root@node-1:/home# ipvsadm -A -t 10.10.0.1:8410 -s rr
root@node-1:/home# ip addr add 10.10.0.1/24 dev flannel.1
RTNETLINK answers: File exists
root@node-1:/home# ipvsadm -a -t 10.10.0.1:8410 -r 10.2.16.5:12345 -m
root@node-1:/home# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.10.0.1:8410 rr
  -> 10.2.16.5:12345              Masq    1      0          0
root@node-1:/home# nc -zv 10.10.0.1 8410
Connection to 10.10.0.1 8410 port [tcp/*] succeeded!
root@node-1:/home#
```

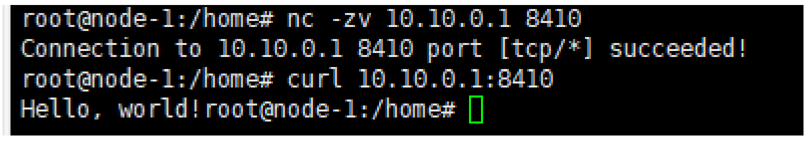


```
17: flannel.1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state UNKNOWN group default
    link/ether 2a:f1:30:e9:8b:2f brd ff:ff:ff:ff:ff:ff
    inet 10.2.6.0/32 scope global flannel.1
       valid_lft forever preferred_lft forever
    inet 10.10.0.1/24 scope global flannel.1
       valid_lft forever preferred_lft forever
    inet6 fe80::28f1:30ff:fee9:8b2f/64 scope link
       valid_lft forever preferred_lft forever
```

在node1节点上手动执行命令绑定ClusterIP到flannel.1虚拟网卡上，并配置endpoint（真正提供服务的节点）

这里就是配置前面node3的容器中开启的服务

执行nc指令和curl指令发现连接成功，得到node3的容器中服务的返回值Hello，world



```
root@node-1:/home# nc -zv 10.10.0.1 8410
Connection to 10.10.0.1 8410 port [tcp/*] succeeded!
root@node-1:/home# curl 10.10.0.1:8410
Hello, world!root@node-1:/home#
```