ZEVENET

Kube-Proxy Deep Dive

Kubernetes Sevilla - Octubre 2019

Let’s talk about…

Intro

Traditional load balancing

Networking in Docker

Networking challenges in Kubernetes

What’s next?

Demo

ZEVENET

Intro

Since 2010, from Zen Load Balancer to Zevenet developing highly available and scalable systems.

- 500 customers
- 60 countries
- 1600 downloads per month

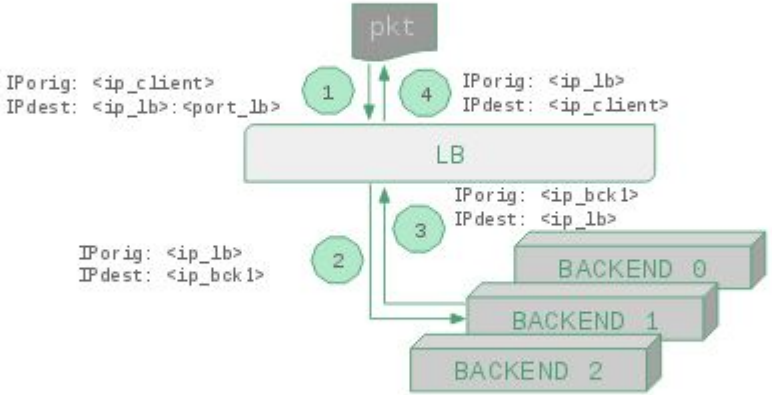
Traditional Load Balancing

Kernel Space

- Routing
- **sNAT**
- DSR

User Space

- Proxy
- DNS



Prerouting

Forward

Postrouting

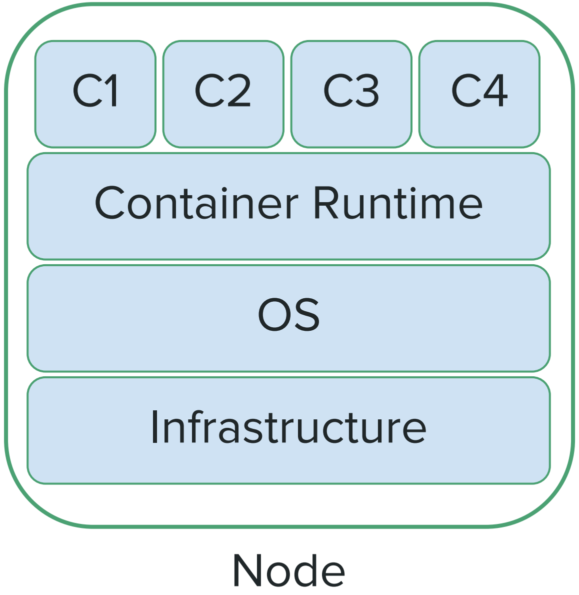
Networking in Docker

Low level networking layer

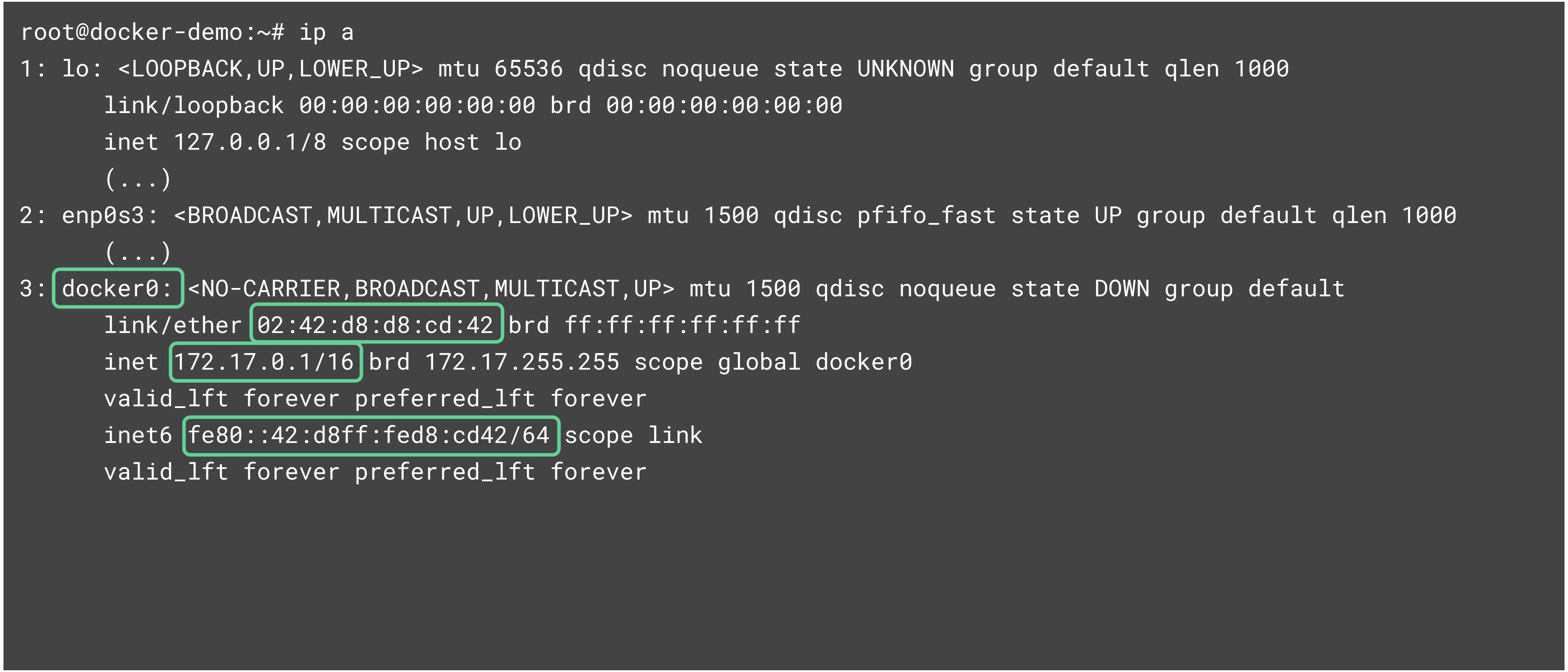
- IPVLAN, MACVLAN, routing, namespaces, iptables

Container networking layer

- Single-host bridge, multi-host, IP-per-container

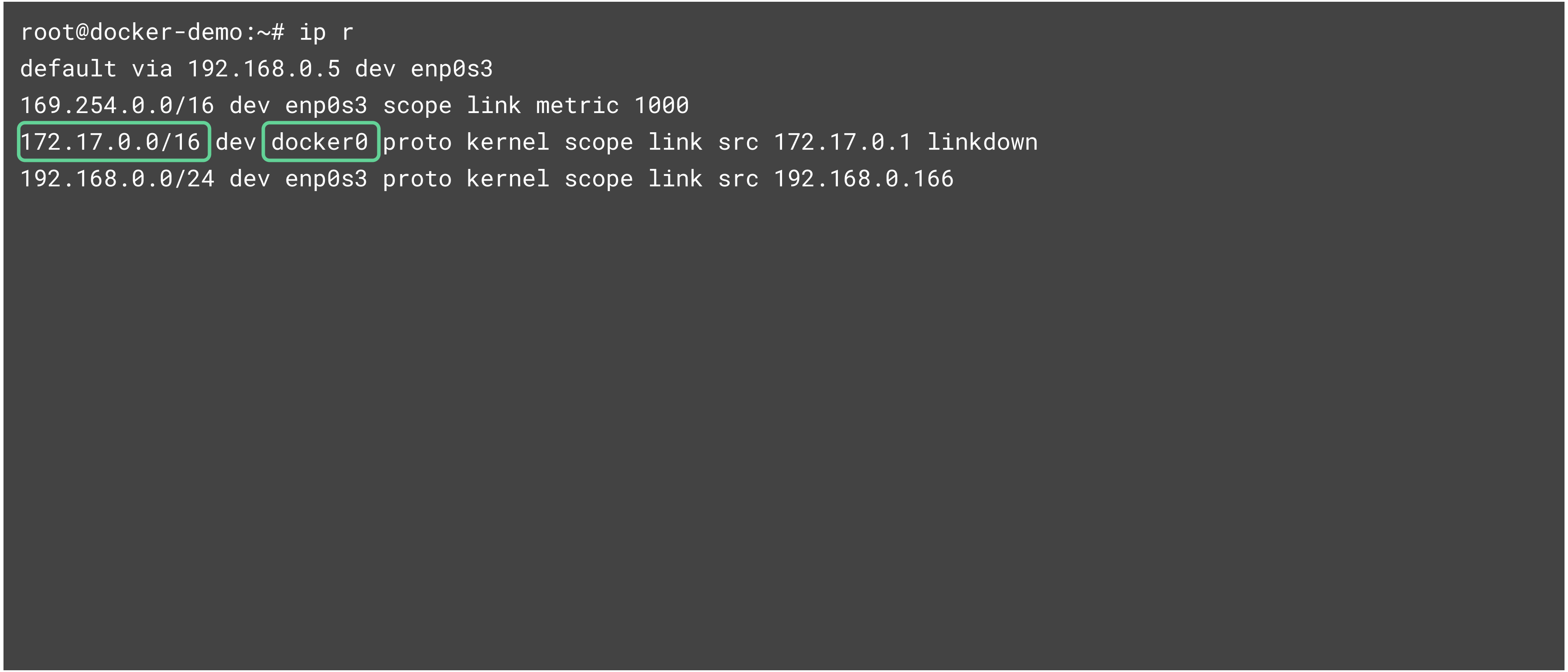


Networking in Docker



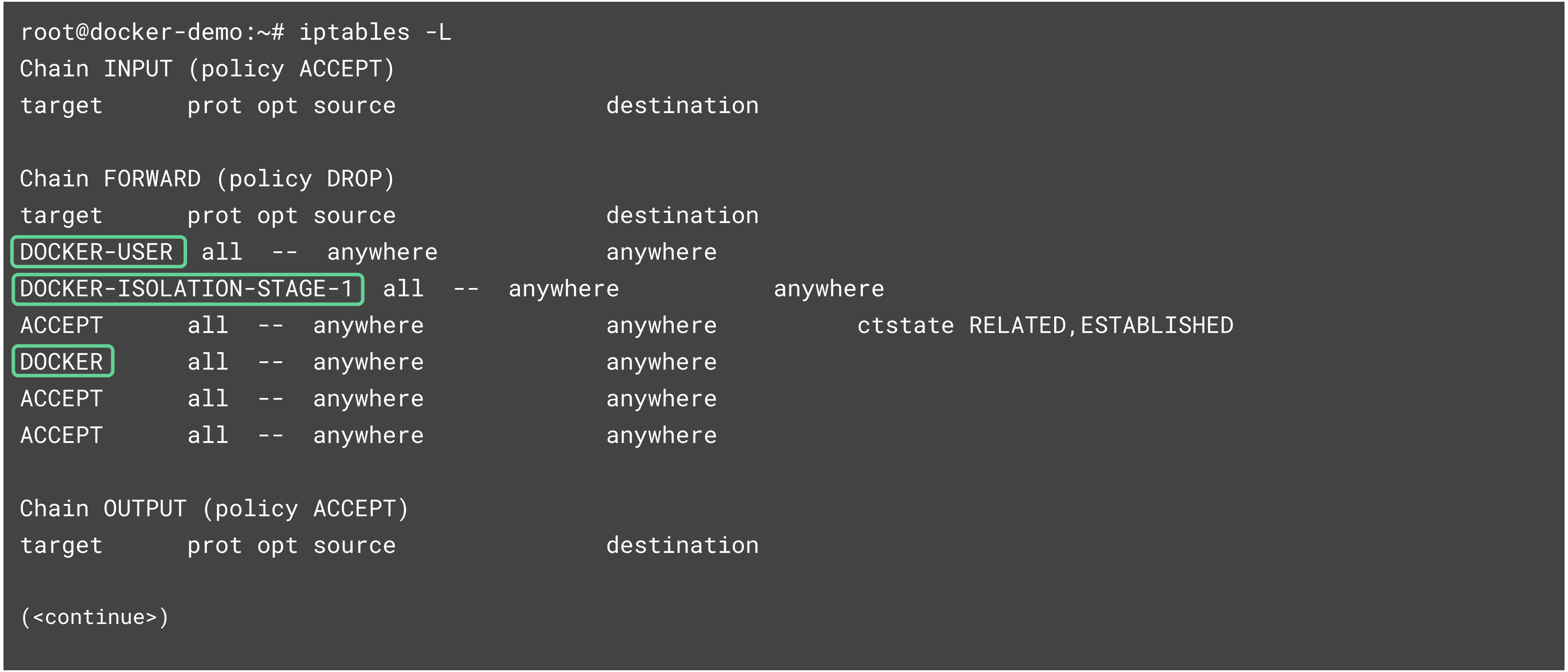
```
root@docker-demo:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
      link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
      inet 127.0.0.1/8 scope host lo
      (...)
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
      (...)
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
      link/ether 02:42:d8:d8:cd:42 brd ff:ff:ff:ff:ff:ff
      inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
      valid_lft forever preferred_lft forever
      inet6 fe80::42:d8ff:fed8:cd42/64 scope link
      valid_lft forever preferred_lft forever
```

Networking in Docker



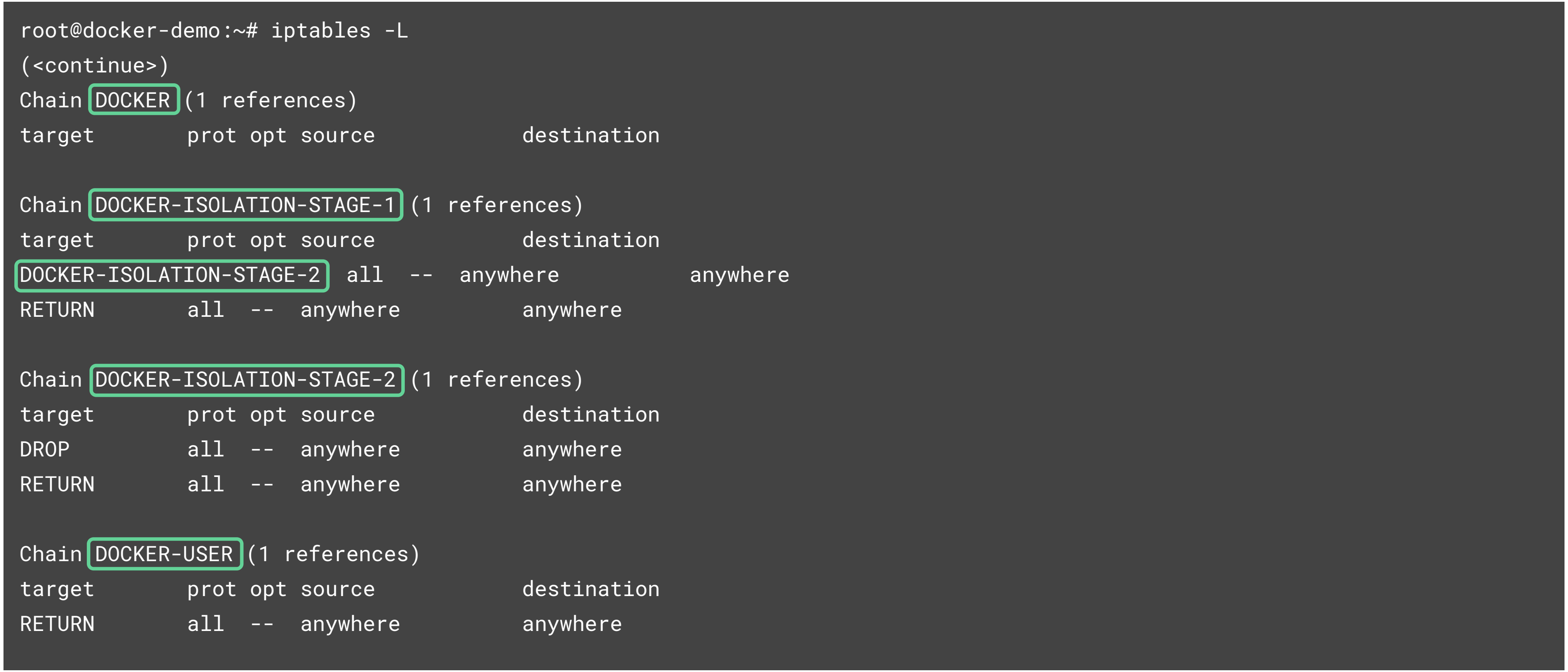
```
root@docker-demo:~# ip r
default via 192.168.0.5 dev enp0s3
169.254.0.0/16 dev enp0s3 scope link metric 1000
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.0.0/24 dev enp0s3 proto kernel scope link src 192.168.0.166
```

Networking in Docker



```
root@docker-demo:~# iptables -L
Chain INPUT (policy ACCEPT)
target      prot opt source               destination

Chain FORWARD (policy DROP)
target      prot opt source               destination
DOCKER-USER  all  --  anywhere            anywhere
DOCKER-ISOLATION-STAGE-1  all  --  anywhere           anywhere
ACCEPT      all  --  anywhere             anywhere            ctstate RELATED,ESTABLISHED
DOCKER      all  --  anywhere             anywhere
ACCEPT      all  --  anywhere             anywhere
ACCEPT      all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target      prot opt source               destination

(<continue>)
```

Networking in Docker

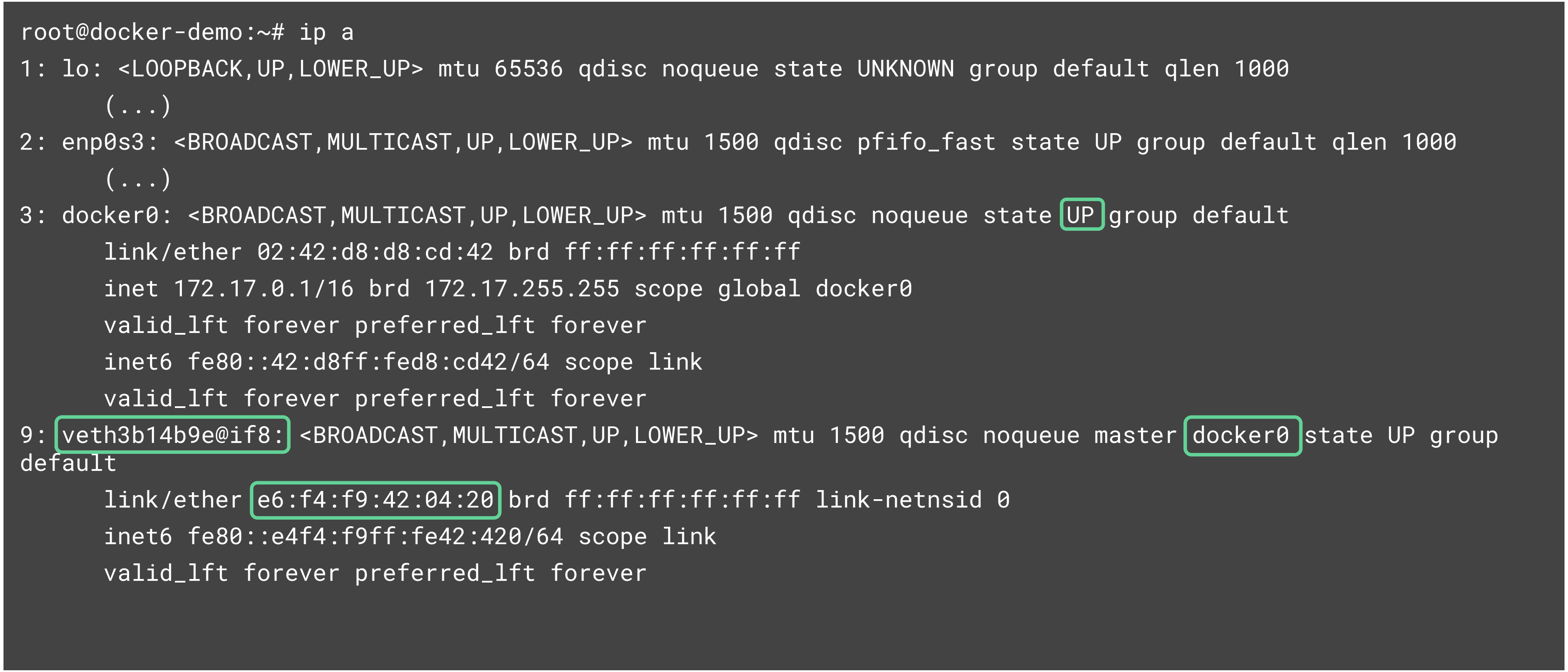


```
root@docker-demo:~# iptables -L
(<continue>)
Chain DOCKER (1 references)
target       prot opt source            destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
target       prot opt source            destination
DOCKER-ISOLATION-STAGE-2  all  --  anywhere          anywhere
RETURN       all  --  anywhere          anywhere

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
target       prot opt source            destination
DROP         all  --  anywhere          anywhere
RETURN       all  --  anywhere          anywhere

Chain DOCKER-USER (1 references)
target       prot opt source            destination
RETURN       all  --  anywhere          anywhere
```

Networking in Docker | IP-per-container

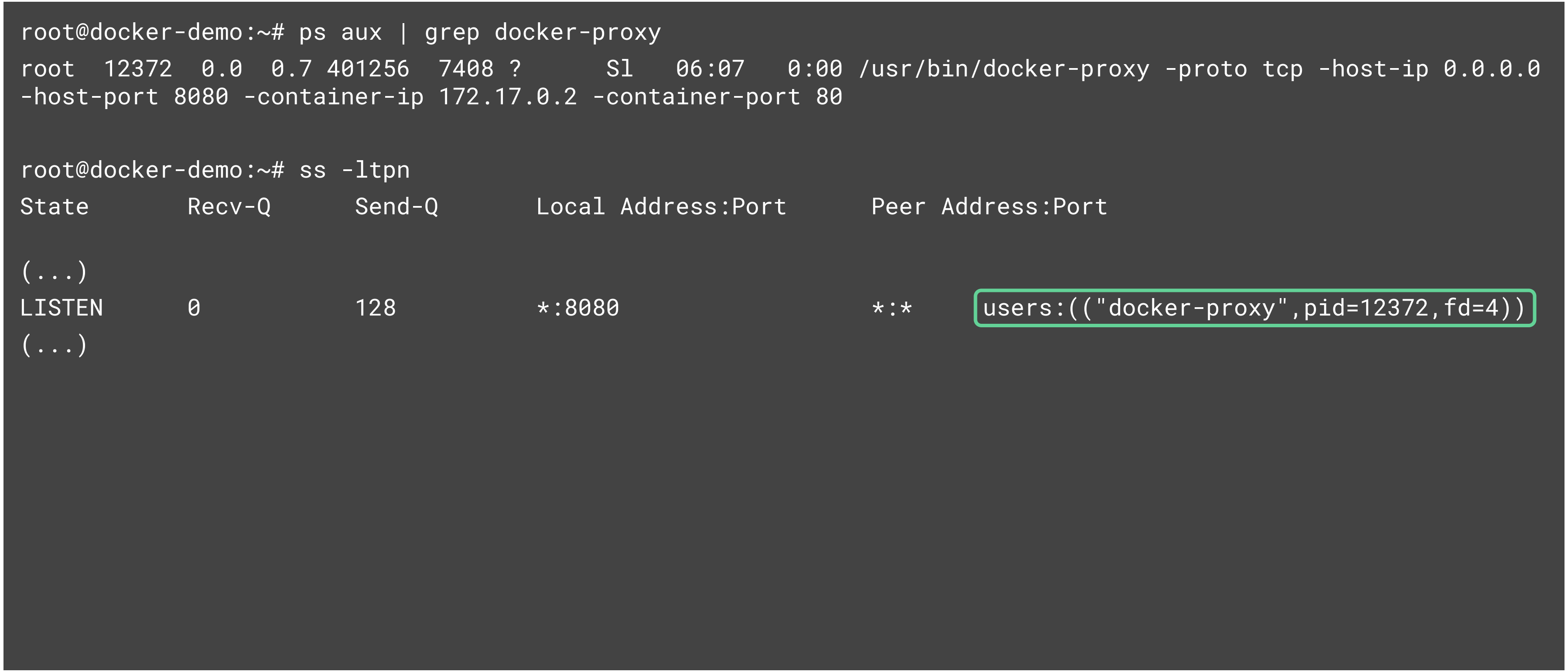
```
root@docker-demo:~# docker run -dit --name my-apache-app \
            -p 8080:80 -v "$PWD":/usr/local/apache2/htdocs/ httpd:2.4
```

Networking in Docker | IP-per-container



```
root@docker-demo:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
      (...)
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
      (...)
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
      link/ether 02:42:d8:d8:cd:42 brd ff:ff:ff:ff:ff:ff
      inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
      valid_lft forever preferred_lft forever
      inet6 fe80::42:d8ff:fed8:cd42/64 scope link
      valid_lft forever preferred_lft forever
9: veth3b14b9e@if8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group
default
      link/ether e6:f4:f9:42:04:20 brd ff:ff:ff:ff:ff:ff link-netnsid 0
      inet6 fe80::e4f4:f9ff:fe42:420/64 scope link
      valid_lft forever preferred_lft forever
```

Networking in Docker | IP-per-container

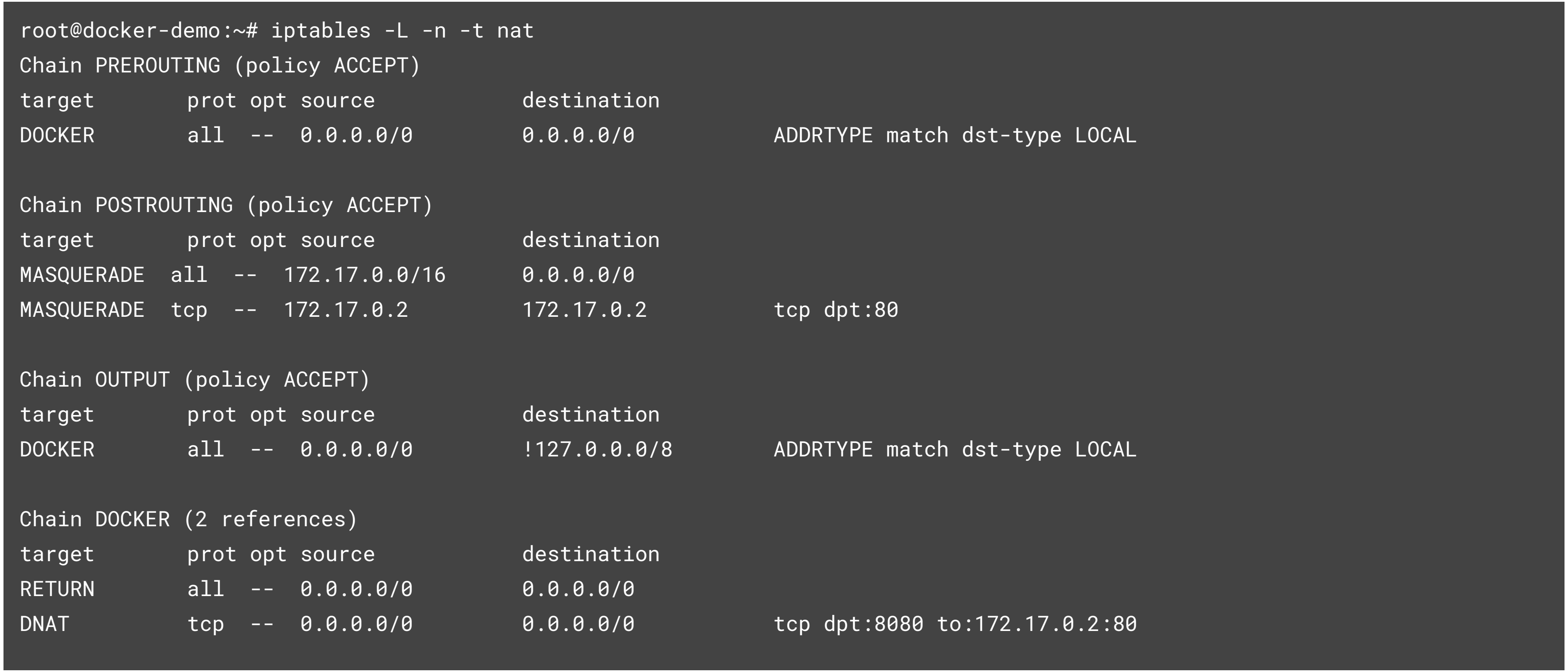


```
root@docker-demo:~# ps aux | grep docker-proxy
root  12372  0.0  0.7 401256  7408 ?      Sl   06:07   0:00 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0
-host-port 8080 -container-ip 172.17.0.2 -container-port 80


root@docker-demo:~# ss -ltpn
State       Recv-Q      Send-Q       Local Address:Port      Peer Address:Port

(...)
LISTEN      0           128          *:8080                  *:*     users:(("docker-proxy",pid=12372,fd=4))
(...)
```

Networking in Docker | IP-per-container

```
root@docker-demo:~# iptables -L
(...)

Chain DOCKER (1 references)
target      prot opt source               destination
ACCEPT      tcp  --  anywhere             172.17.0.2        tcp dpt:http

(...)
```

Networking in Docker | IP-per-container

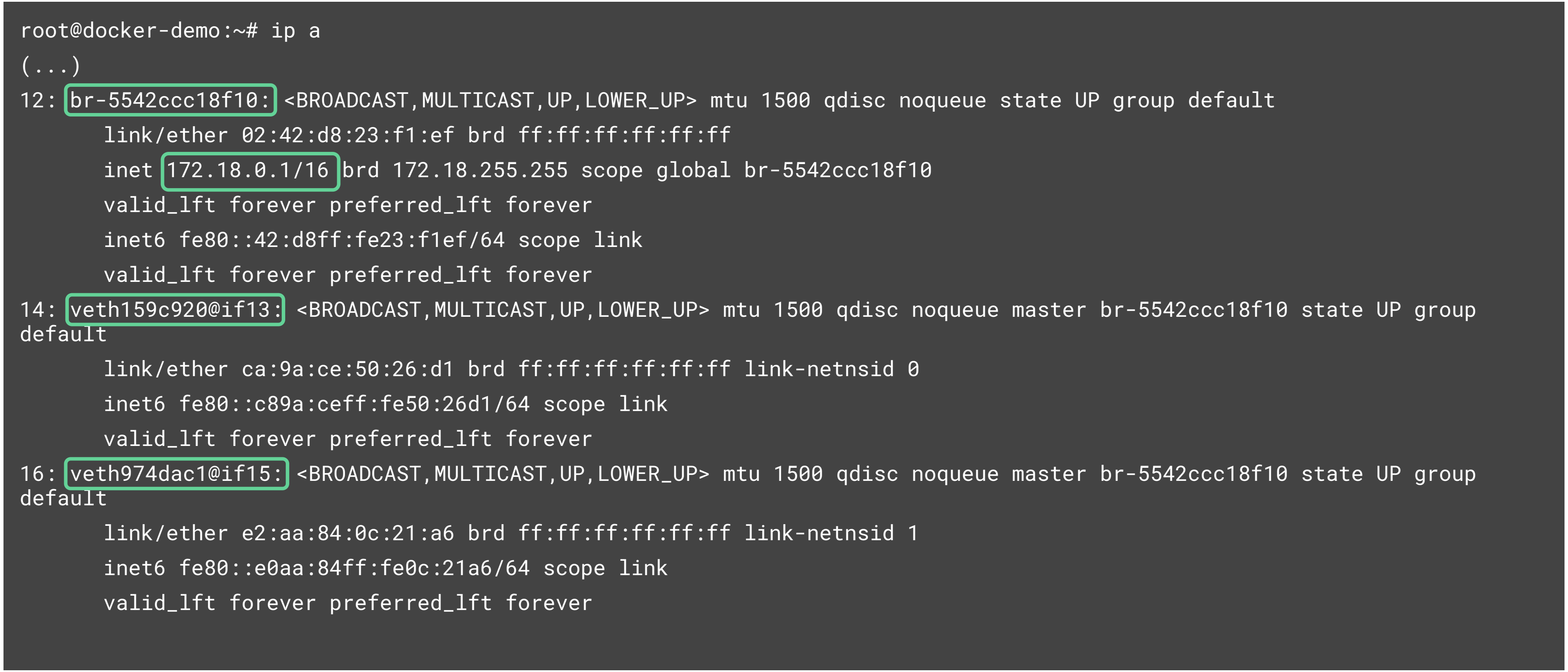


```
root@docker-demo:~# iptables -L -n -t nat
Chain PREROUTING (policy ACCEPT)
target       prot opt source            destination
DOCKER       all  --  0.0.0.0/0         0.0.0.0/0           ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT)
target       prot opt source            destination
MASQUERADE  all  --  172.17.0.0/16      0.0.0.0/0
MASQUERADE  tcp  --  172.17.0.2         172.17.0.2          tcp dpt:80

Chain OUTPUT (policy ACCEPT)
target       prot opt source            destination
DOCKER       all  --  0.0.0.0/0         !127.0.0.0/8        ADDRTYPE match dst-type LOCAL

Chain DOCKER (2 references)
target       prot opt source            destination
RETURN       all  --  0.0.0.0/0         0.0.0.0/0
DNAT         tcp  --  0.0.0.0/0         0.0.0.0/0           tcp dpt:8080 to:172.17.0.2:80
```

Networking in Docker | Single-host-bridge

```
root@docker-demo:~# docker run -dit --name my-apache-app2 -p 8081:80 \
                     -v "$PWD":/usr/local/apache2/htdocs/ httpd:2.4
root@docker-demo:~# docker network create my-net
root@docker-demo:~# docker network connect my-net my-apache-app
root@docker-demo:~# docker network connect my-net my-apache-app2
```

Networking in Docker | Single-host-bridge



```
root@docker-demo:~# ip a
(...)
12: br-5542ccc18f10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
      link/ether 02:42:d8:23:f1:ef brd ff:ff:ff:ff:ff:ff
      inet 172.18.0.1/16 brd 172.18.255.255 scope global br-5542ccc18f10
      valid_lft forever preferred_lft forever
      inet6 fe80::42:d8ff:fe23:f1ef/64 scope link
      valid_lft forever preferred_lft forever
14: veth159c920@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-5542ccc18f10 state UP group
default
      link/ether ca:9a:ce:50:26:d1 brd ff:ff:ff:ff:ff:ff link-netnsid 0
      inet6 fe80::c89a:ceff:fe50:26d1/64 scope link
      valid_lft forever preferred_lft forever
16: veth974dac1@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-5542ccc18f10 state UP group
default
      link/ether e2:aa:84:0c:21:a6 brd ff:ff:ff:ff:ff:ff link-netnsid 1
      inet6 fe80::e0aa:84ff:fe0c:21a6/64 scope link
      valid_lft forever preferred_lft forever
```

Networking in containers

Summary

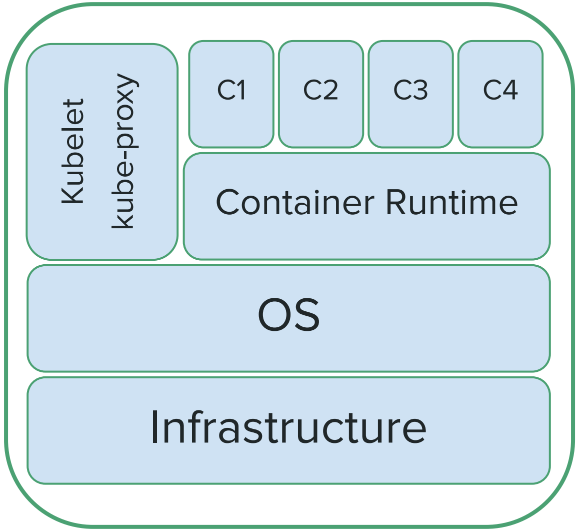
- Private IP network addresses and NATing to outside
- Virtual Interfaces with own MAC address
- Kernel space NATing to forward traffic to containerized services
- User space docker-proxy for layer 7 purposes (not always required)
- Namespaces per container
- Resolves IP-per-container, single-host (bridge) and multi-host (overlay) modes
- DIY load balancing & clustering

ZEVENET

Networking challenges in Kubernetes

Resolve:

- Container orchestration layer
- Clustering
- Service discovery
- Load Balancing
- Automation



Node

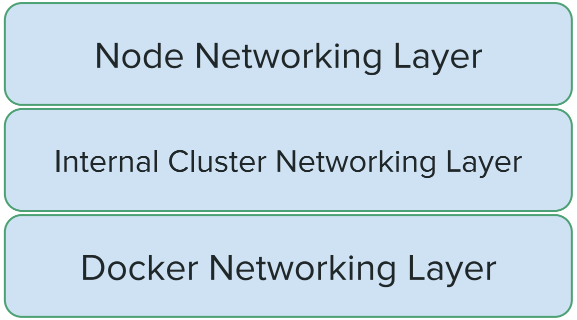
Networking challenges in Kubernetes

Minikube

- Kubernetes in 1 node

Cluster Architecture

- Several masters
- Several nodes



Node Networking Layer

Internal Cluster Networking Layer

Docker Networking Layer

Networking challenges in Kubernetes

Components:

Master Components

- kube-apiserver
- etcd
- kube-scheduler
- kube-controller-manager
- cloud-controller-manager

Node Components

- kubelet
- **kube-proxy**
- container runtime

Addons

- dns
- web ui
- container resource monitoring
- cluster logging

Networking challenges in Kubernetes

kube-proxy in charge of

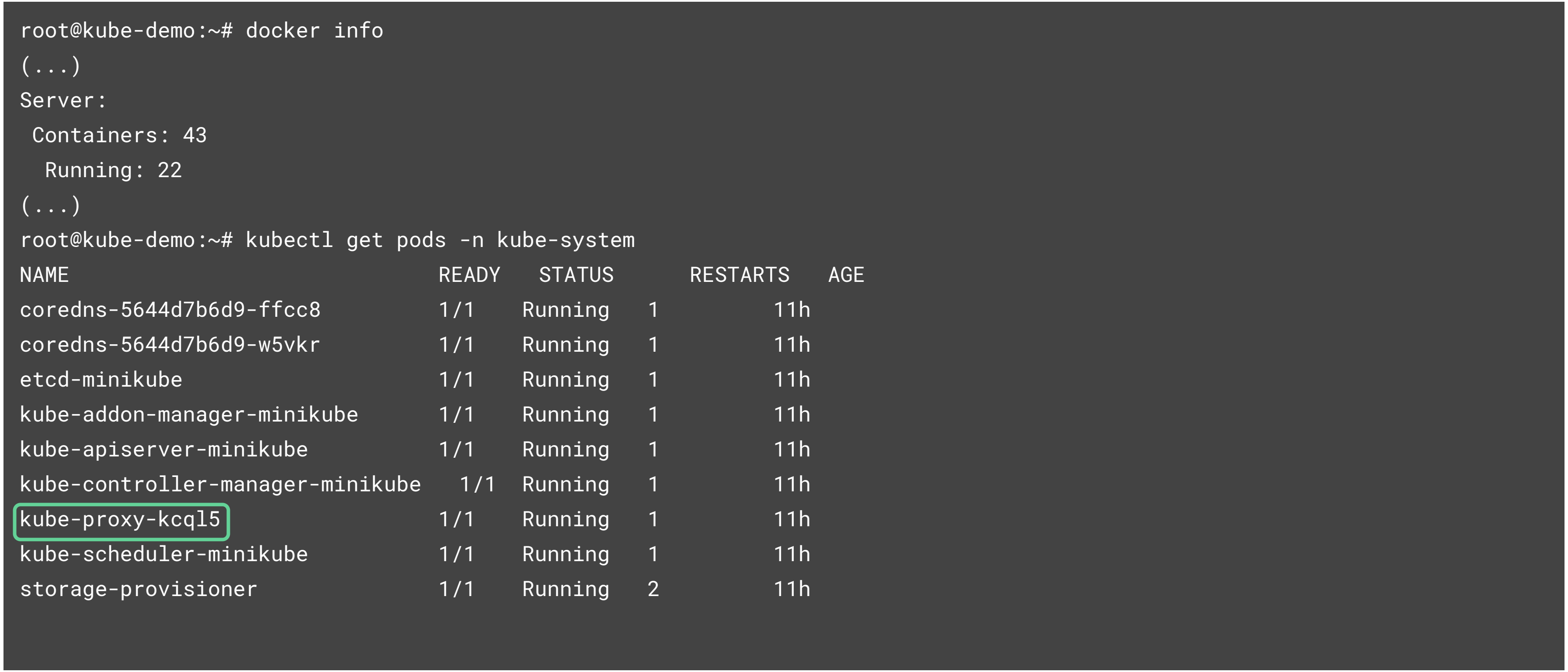
- Allow the communication to the pods from inside or outside the cluster
- Forward the traffic
- Services load balance: different working modes
- Ensure the users network session: connection tracking and persistence
- Access filtering to the services: iptables

Networking challenges in Kubernetes

kube-proxy --proxy-mode <ProxyMode>

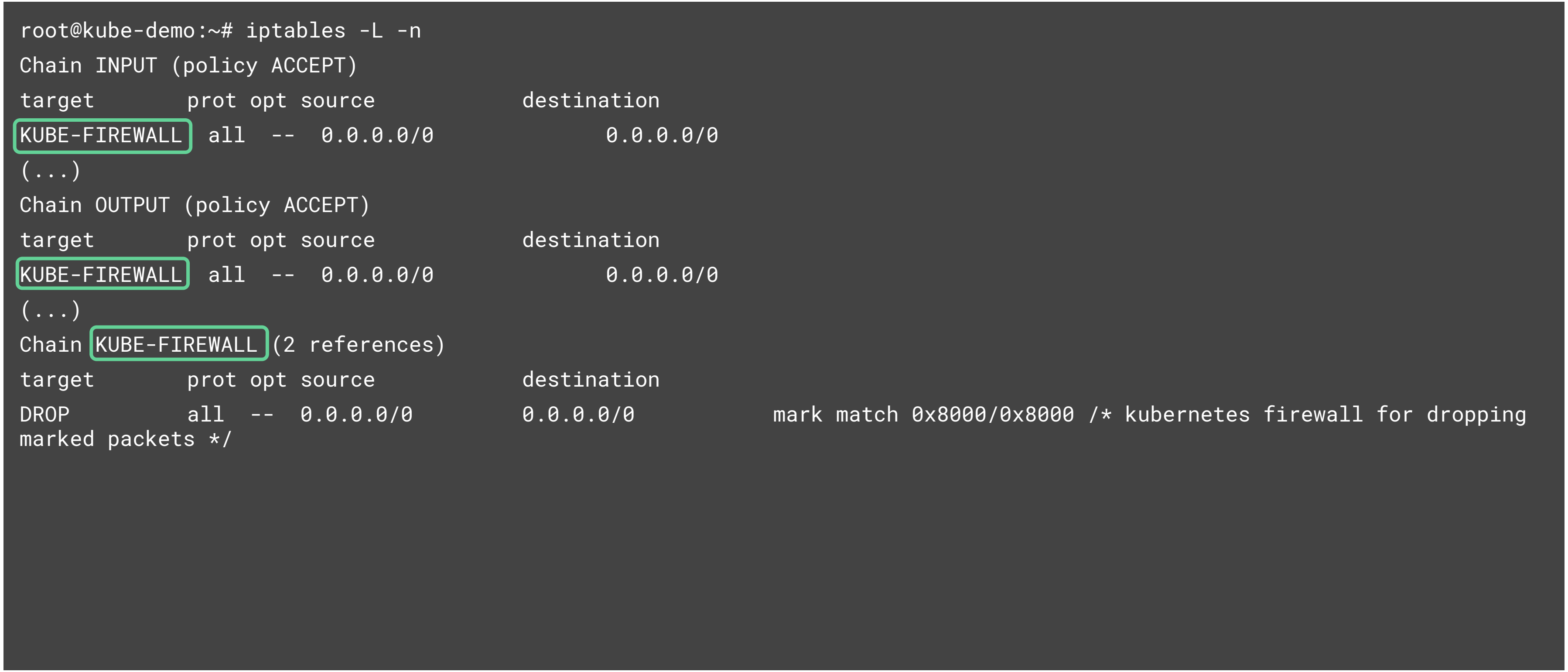
- userspace
  - old and unused
- iptables (default)
  - NAT load balancing, filtering, equal cost & round-robin
  - Based on sequential rules with a complexity of O(n)
- Ipvs
  - NAT & DSR load balancing, more advanced schedulers
  - Kernel process with complexity of O(1), relies on iptables for certain cases

Networking challenges in Kubernetes



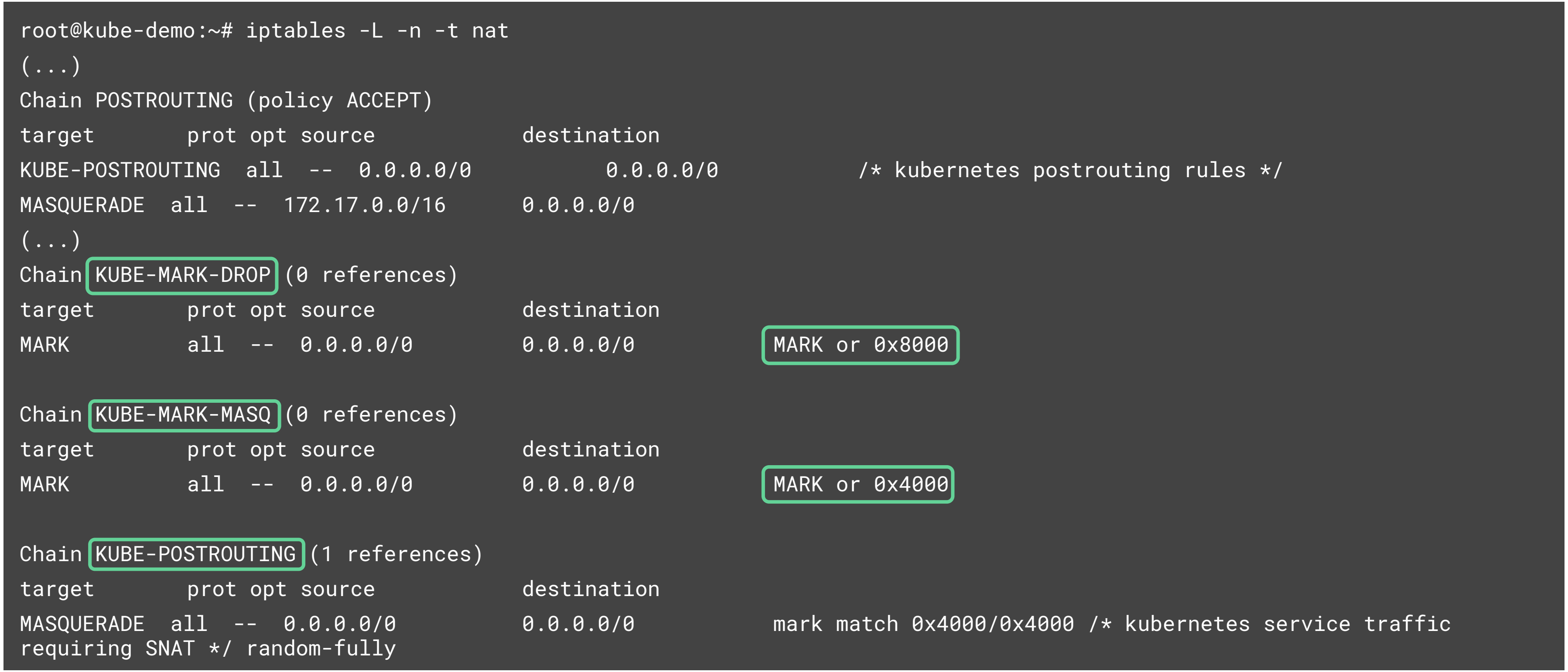
```
root@kube-demo:~# docker info
(...)
Server:
 Containers: 43
  Running: 22
(...)
root@kube-demo:~# kubectl get pods -n kube-system
NAME                             READY   STATUS      RESTARTS   AGE
coredns-5644d7b6d9-ffcc8         1/1    Running   1         11h
coredns-5644d7b6d9-w5vkr         1/1    Running   1         11h
etcd-minikube                    1/1    Running   1         11h
kube-addon-manager-minikube      1/1    Running   1         11h
kube-apiserver-minikube          1/1    Running   1         11h
kube-controller-manager-minikube   1/1  Running   1         11h
kube-proxy-kcql5                 1/1    Running   1         11h
kube-scheduler-minikube          1/1    Running   1         11h
storage-provisioner              1/1    Running   2         11h
```

Networking challenges in Kubernetes



```
root@kube-demo:~# iptables -L -n
Chain INPUT (policy ACCEPT)
target       prot opt source            destination
KUBE-FIREWALL  all  --  0.0.0.0/0             0.0.0.0/0
(...)
Chain OUTPUT (policy ACCEPT)
target       prot opt source            destination
KUBE-FIREWALL  all  --  0.0.0.0/0             0.0.0.0/0
(...)
Chain KUBE-FIREWALL (2 references)
target       prot opt source            destination
DROP         all  --  0.0.0.0/0         0.0.0.0/0           mark match 0x8000/0x8000 /* kubernetes firewall for dropping
marked packets */
```

Networking challenges in Kubernetes



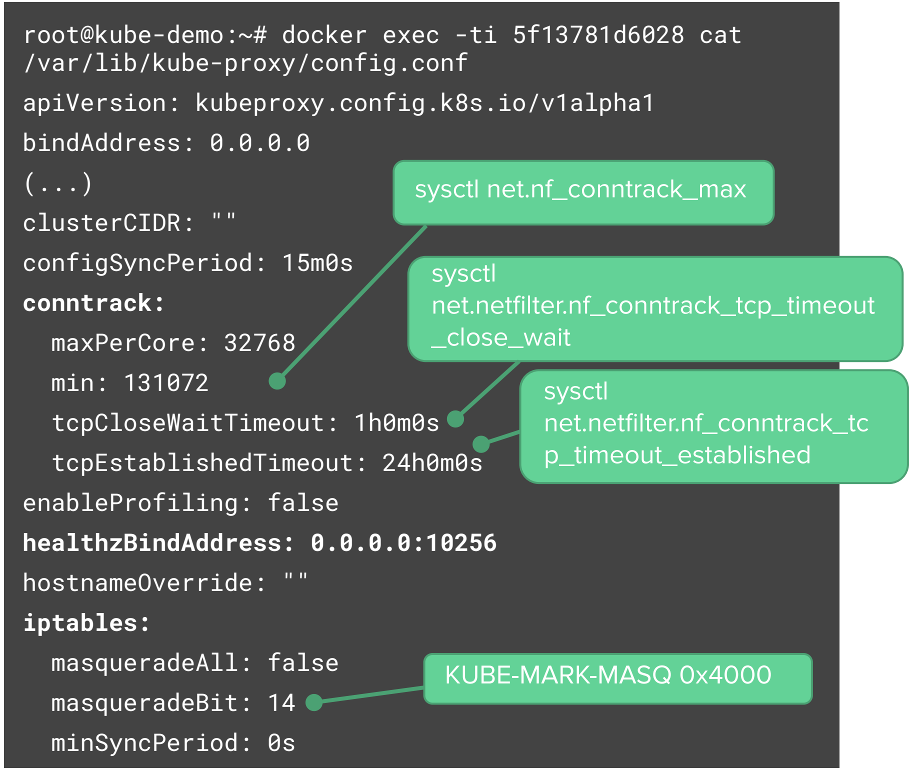
```
root@kube-demo:~# iptables -L -n -t nat
(...)
Chain POSTROUTING (policy ACCEPT)
target       prot opt source            destination
KUBE-POSTROUTING  all  --  0.0.0.0/0           0.0.0.0/0           /* kubernetes postrouting rules */
MASQUERADE  all  --  172.17.0.0/16      0.0.0.0/0
(...)
Chain KUBE-MARK-DROP (0 references)
target       prot opt source            destination
MARK         all  --  0.0.0.0/0         0.0.0.0/0           MARK or 0x8000

Chain KUBE-MARK-MASQ (0 references)
target       prot opt source            destination
MARK         all  --  0.0.0.0/0         0.0.0.0/0           MARK or 0x4000

Chain KUBE-POSTROUTING (1 references)
target       prot opt source            destination
MASQUERADE  all  --  0.0.0.0/0          0.0.0.0/0           mark match 0x4000/0x4000 /* kubernetes service traffic
requiring SNAT */ random-fully
```

Networking challenges in Kubernetes

```
root@kube-demo:~# docker exec -ti 5f13781d6028 cat
/var/lib/kube-proxy/config.conf
apiVersion: kubeproxy.config.k8s.io/v1alpha1
bindAddress: 0.0.0.0
(...)
clusterCIDR: ""
configSyncPeriod: 15m0s
conntrack:
  maxPerCore: 32768
  min: 131072
  tcpCloseWaitTimeout: 1h0m0s
  tcpEstablishedTimeout: 24h0m0s
enableProfiling: false
healthzBindAddress: 0.0.0.0:10256
hostnameOverride: ""
iptables:
  masqueradeAll: false
  masqueradeBit: 14
  minSyncPeriod: 0s
```

```
  syncPeriod: 30s
ipvs:
  excludeCIDRs: null
  minSyncPeriod: 0s
  scheduler: ""
  strictARP: false
  syncPeriod: 30s
kind: KubeProxyConfiguration
metricsBindAddress: 127.0.0.1:10249
mode: ""
nodePortAddresses: null
oomScoreAdj: -999
portRange: ""
udpIdleTimeout: 250ms
winkernel:
  enableDSR: false
  networkName: ""
```

Networking challenges in Kubernetes



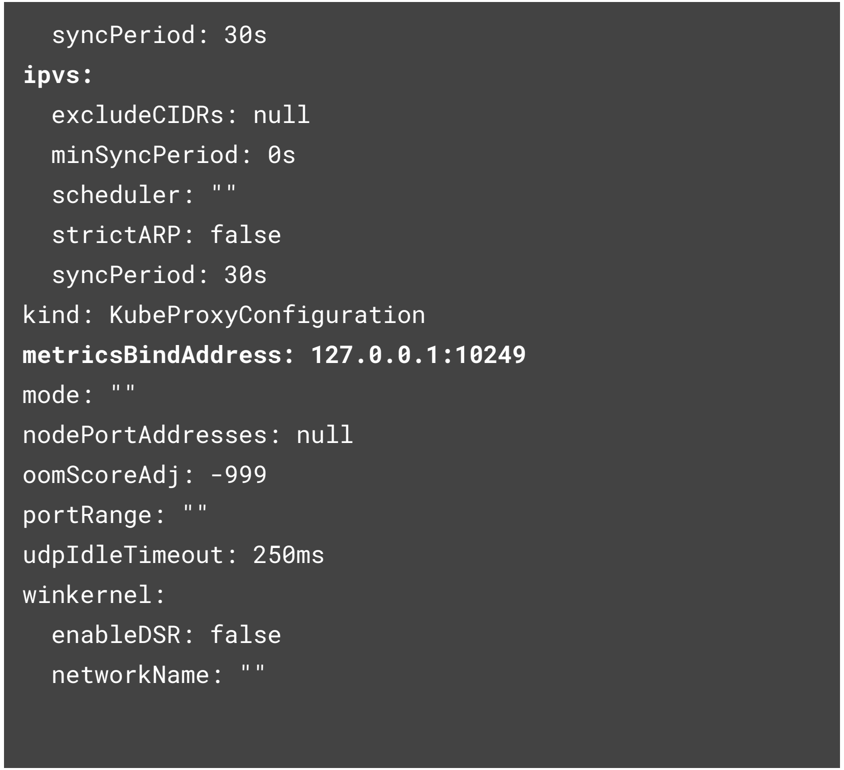
```
root@kube-demo:~# docker exec -ti 5f13781d6028 cat
/var/lib/kube-proxy/config.conf
apiVersion: kubeproxy.config.k8s.io/v1alpha1
bindAddress: 0.0.0.0
(...)
clusterCIDR: ""
configSyncPeriod: 15m0s
conntrack:
  maxPerCore: 32768
  min: 131072
  tcpCloseWaitTimeout: 1h0m0s
  tcpEstablishedTimeout: 24h0m0s
enableProfiling: false
healthzBindAddress: 0.0.0.0:10256
hostnameOverride: ""
iptables:
  masqueradeAll: false
  masqueradeBit: 14
  minSyncPeriod: 0s
```

sysctl net.nf_conntrack_max

sysctl net.netfilter.nf_conntrack_tcp_timeout_close_wait

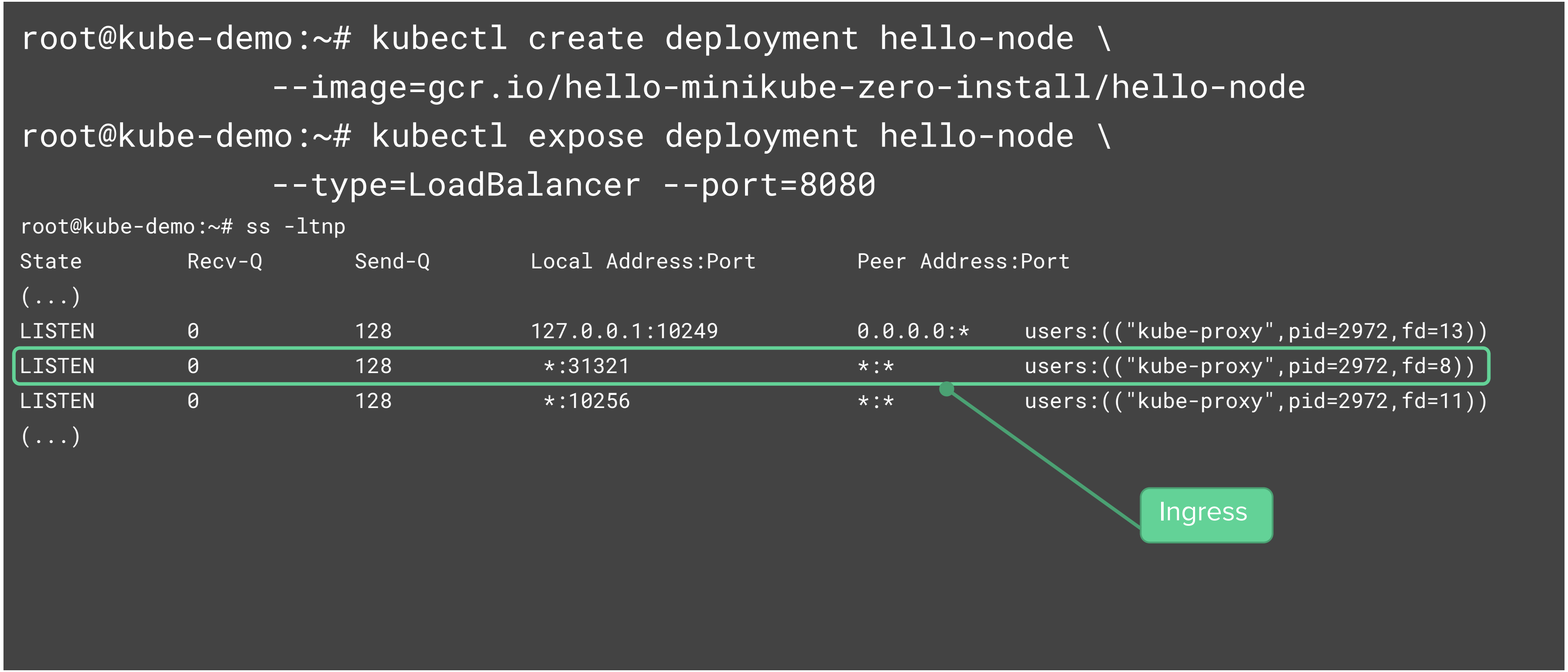
sysctl net.netfilter.nf_conntrack_tcp_timeout_established

KUBE-MARK-MASQ 0x4000



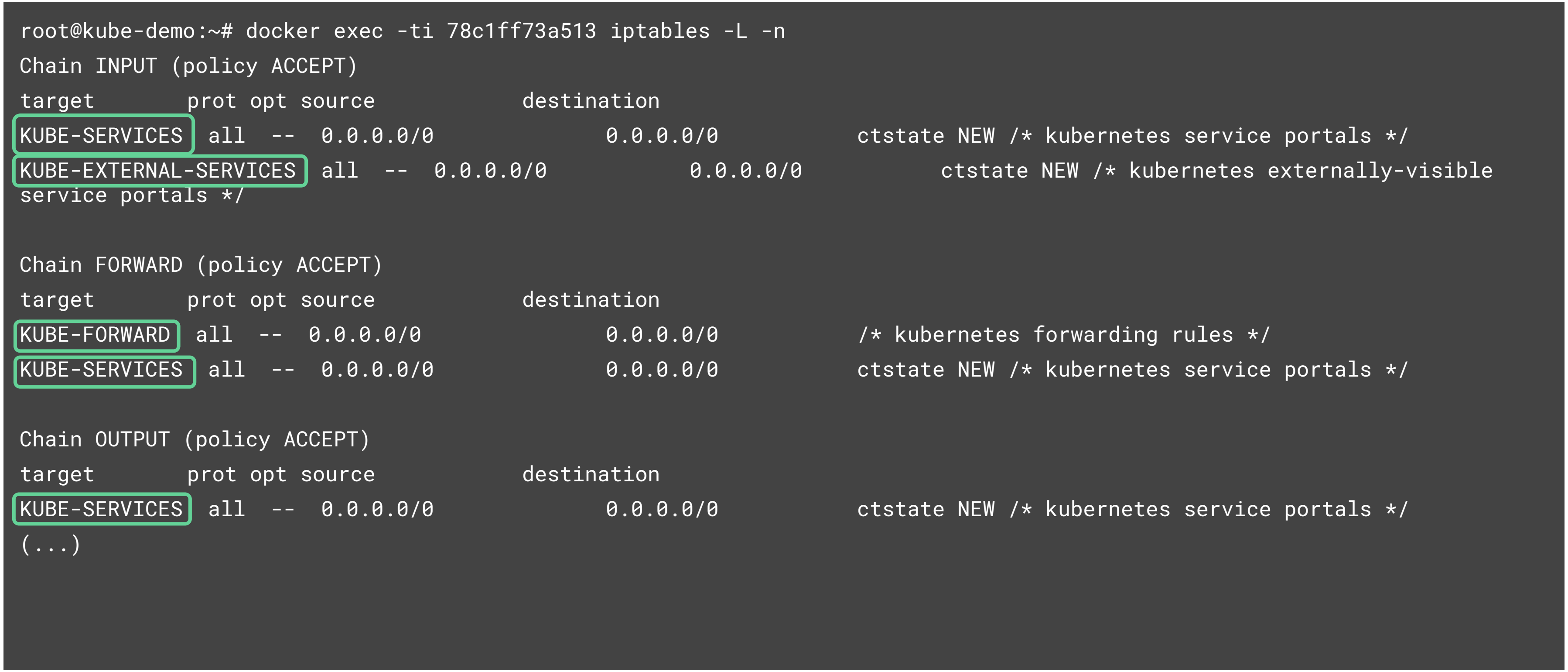
```
  syncPeriod: 30s
ipvs:
  excludeCIDRs: null
  minSyncPeriod: 0s
  scheduler: ""
  strictARP: false
  syncPeriod: 30s
kind: KubeProxyConfiguration
metricsBindAddress: 127.0.0.1:10249
mode: ""
nodePortAddresses: null
oomScoreAdj: -999
portRange: ""
udpIdleTimeout: 250ms
winkernel:
  enableDSR: false
  networkName: ""
```

Networking challenges in Kubernetes



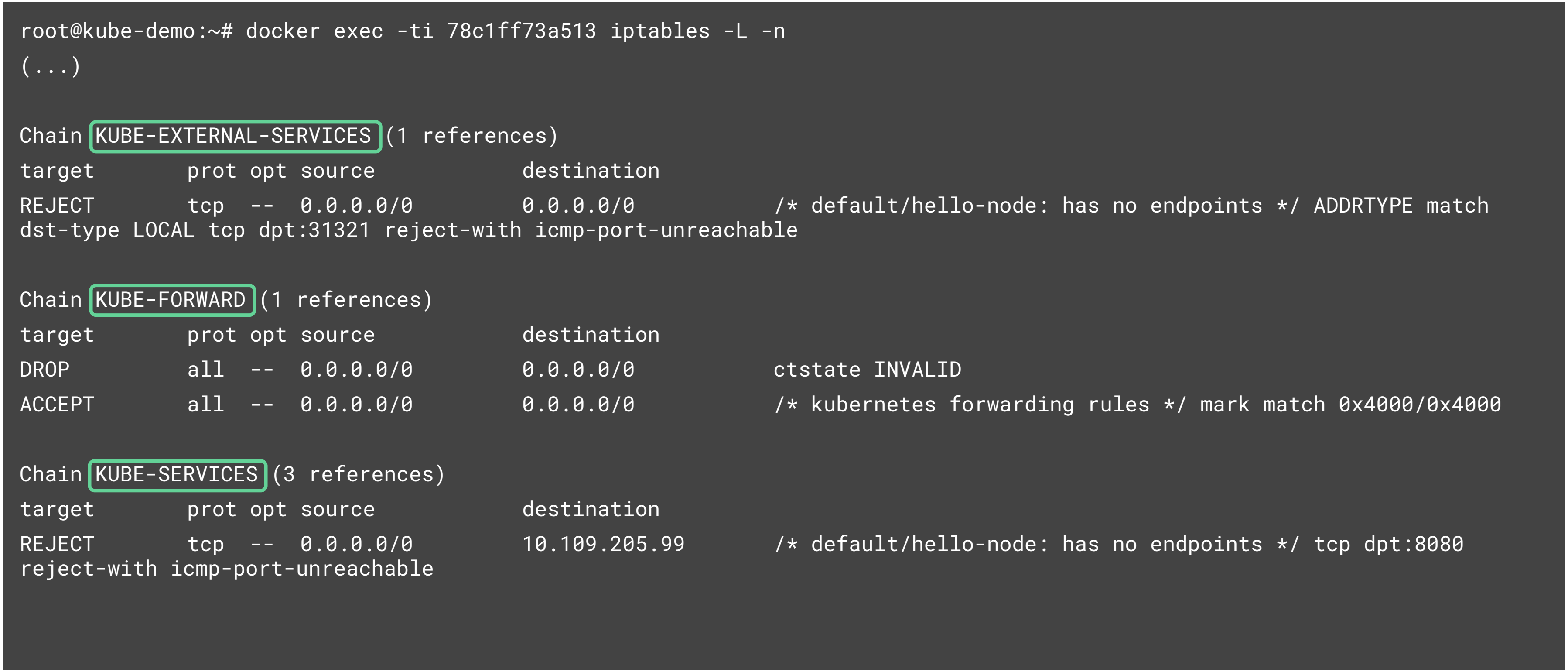
```
root@kube-demo:~# kubectl create deployment hello-node \
                    --image=gcr.io/hello-minikube-zero-install/hello-node
root@kube-demo:~# kubectl expose deployment hello-node \
                    --type=LoadBalancer --port=8080
root@kube-demo:~# ss -ltnp
State         Recv-Q       Send-Q       Local Address:Port        Peer Address:Port
(...)
LISTEN        0            128          127.0.0.1:10249           0.0.0.0:*     users:(("kube-proxy",pid=2972,fd=13))
LISTEN        0            128           *:31321                  *:*           users:(("kube-proxy",pid=2972,fd=8))
LISTEN        0            128           *:10256                  *:*           users:(("kube-proxy",pid=2972,fd=11))
(...)
```

Networking challenges in Kubernetes



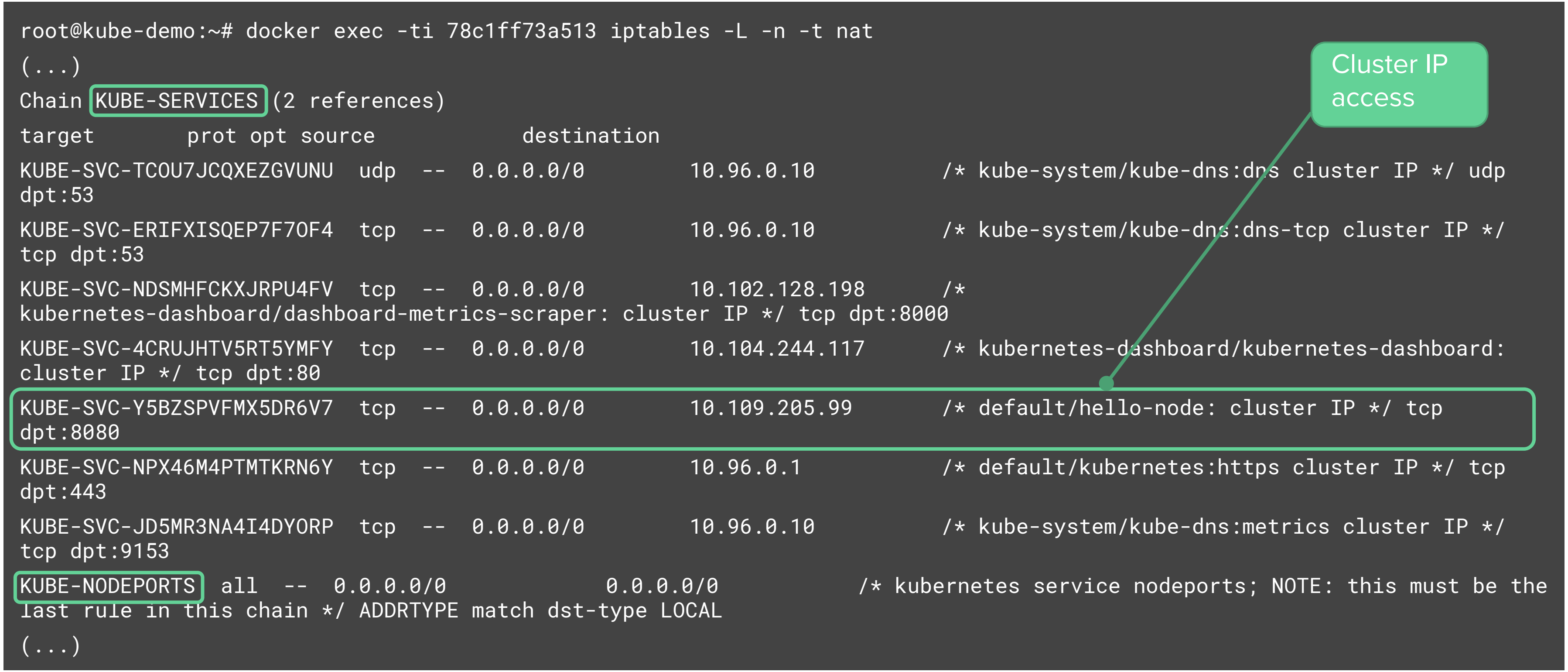
```
root@kube-demo:~# docker exec -ti 78c1ff73a513 iptables -L -n
Chain INPUT (policy ACCEPT)
target       prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0          ctstate NEW /* kubernetes service portals */
KUBE-EXTERNAL-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0          ctstate NEW /* kubernetes externally-visible
service portals */

Chain FORWARD (policy ACCEPT)
target       prot opt source               destination
KUBE-FORWARD  all  --  0.0.0.0/0            0.0.0.0/0          /* kubernetes forwarding rules */
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0          ctstate NEW /* kubernetes service portals */

Chain OUTPUT (policy ACCEPT)
target       prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0          ctstate NEW /* kubernetes service portals */
(...)
```

Networking challenges in Kubernetes



```
root@kube-demo:~# docker exec -ti 78c1ff73a513 iptables -L -n
(...)

Chain KUBE-EXTERNAL-SERVICES (1 references)
target       prot opt source            destination
REJECT       tcp  --  0.0.0.0/0         0.0.0.0/0           /* default/hello-node: has no endpoints */ ADDRTYPE match
dst-type LOCAL tcp dpt:31321 reject-with icmp-port-unreachable

Chain KUBE-FORWARD (1 references)
target       prot opt source            destination
DROP         all  --  0.0.0.0/0         0.0.0.0/0           ctstate INVALID
ACCEPT       all  --  0.0.0.0/0         0.0.0.0/0           /* kubernetes forwarding rules */ mark match 0x4000/0x4000

Chain KUBE-SERVICES (3 references)
target       prot opt source            destination
REJECT       tcp  --  0.0.0.0/0         10.109.205.99       /* default/hello-node: has no endpoints */ tcp dpt:8080
reject-with icmp-port-unreachable
```

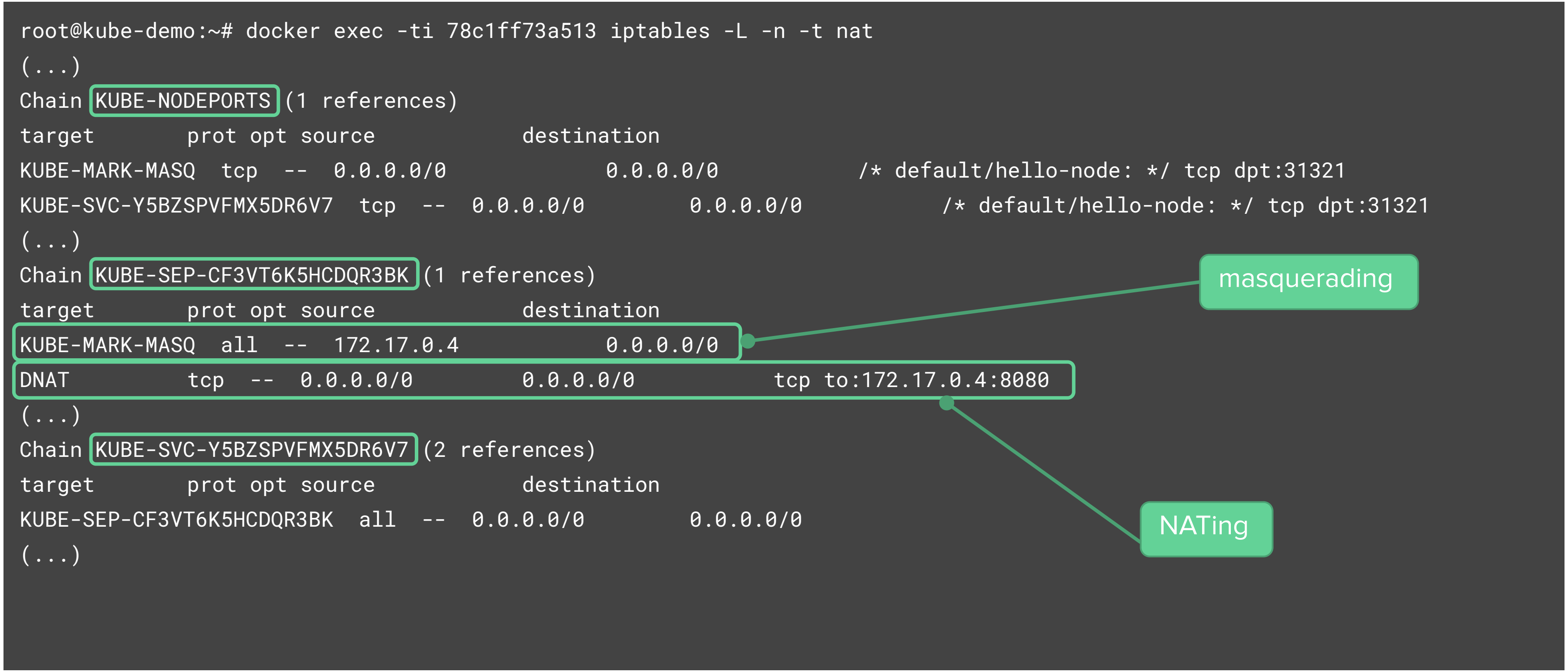
Networking challenges in Kubernetes



```
root@kube-demo:~# docker exec -ti 78c1ff73a513 iptables -L -n -t nat
(...)
Chain KUBE-SERVICES (2 references)
target       prot opt source            destination
KUBE-SVC-TCOU7JCQXEZGVUNU  udp  --  0.0.0.0/0        10.96.0.10          /* kube-system/kube-dns:dns cluster IP */ udp
dpt:53
KUBE-SVC-ERIFXISQEP7F7OF4  tcp  --  0.0.0.0/0        10.96.0.10          /* kube-system/kube-dns:dns-tcp cluster IP */
tcp dpt:53
KUBE-SVC-NDSMHFCKXJRPU4FV  tcp  --  0.0.0.0/0        10.102.128.198      /*
kubernetes-dashboard/dashboard-metrics-scraper: cluster IP */ tcp dpt:8000
KUBE-SVC-4CRUJHTV5RT5YMFY  tcp  --  0.0.0.0/0        10.104.244.117      /* kubernetes-dashboard/kubernetes-dashboard:
cluster IP */ tcp dpt:80
KUBE-SVC-Y5BZSPVFMX5DR6V7  tcp  --  0.0.0.0/0        10.109.205.99       /* default/hello-node: cluster IP */ tcp
dpt:8080
KUBE-SVC-NPX46M4PTMTKRN6Y  tcp  --  0.0.0.0/0        10.96.0.1           /* default/kubernetes:https cluster IP */ tcp
dpt:443
KUBE-SVC-JD5MR3NA4I4DYORP  tcp  --  0.0.0.0/0        10.96.0.10          /* kube-system/kube-dns:metrics cluster IP */
tcp dpt:9153
KUBE-NODEPORTS  all  --  0.0.0.0/0            0.0.0.0/0          /* kubernetes service nodeports; NOTE: this must be the
last rule in this chain */ ADDRTYPE match dst-type LOCAL
(...)
```

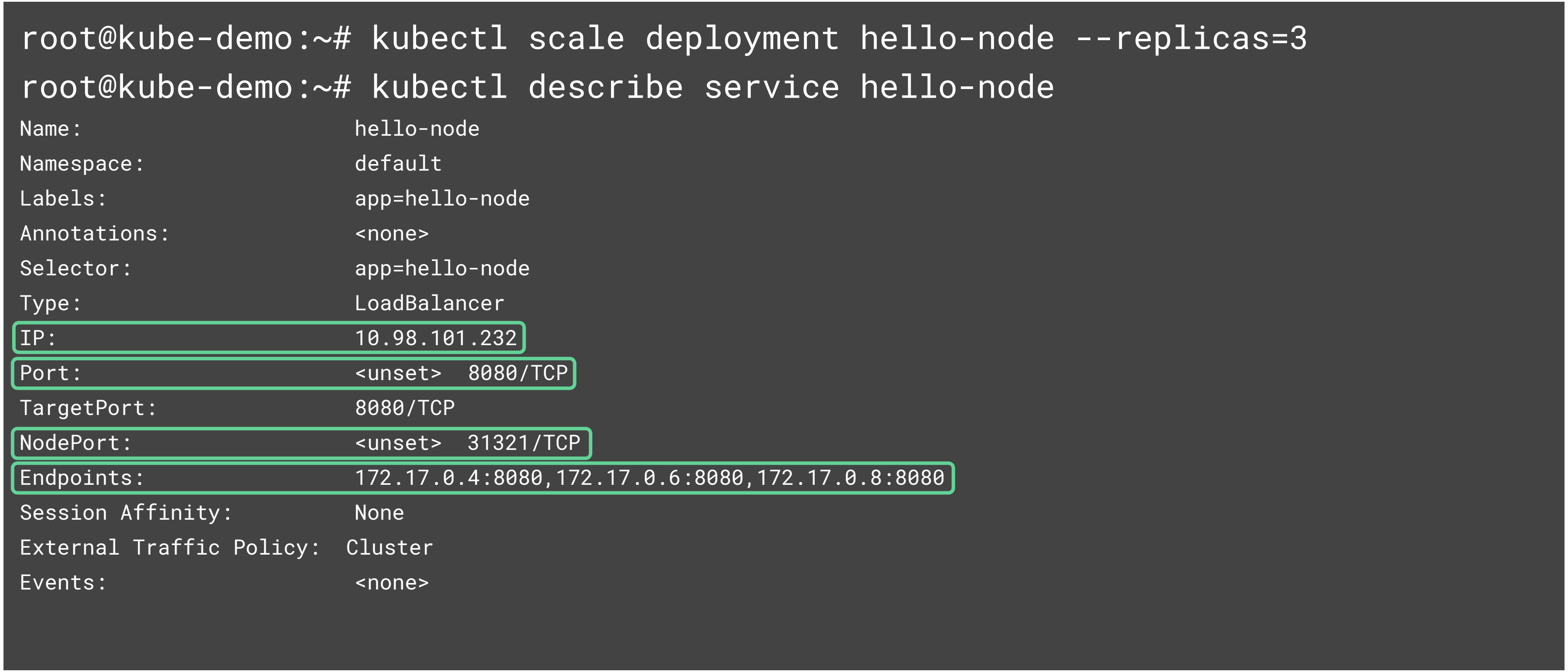
Cluster IP access

Networking challenges in Kubernetes



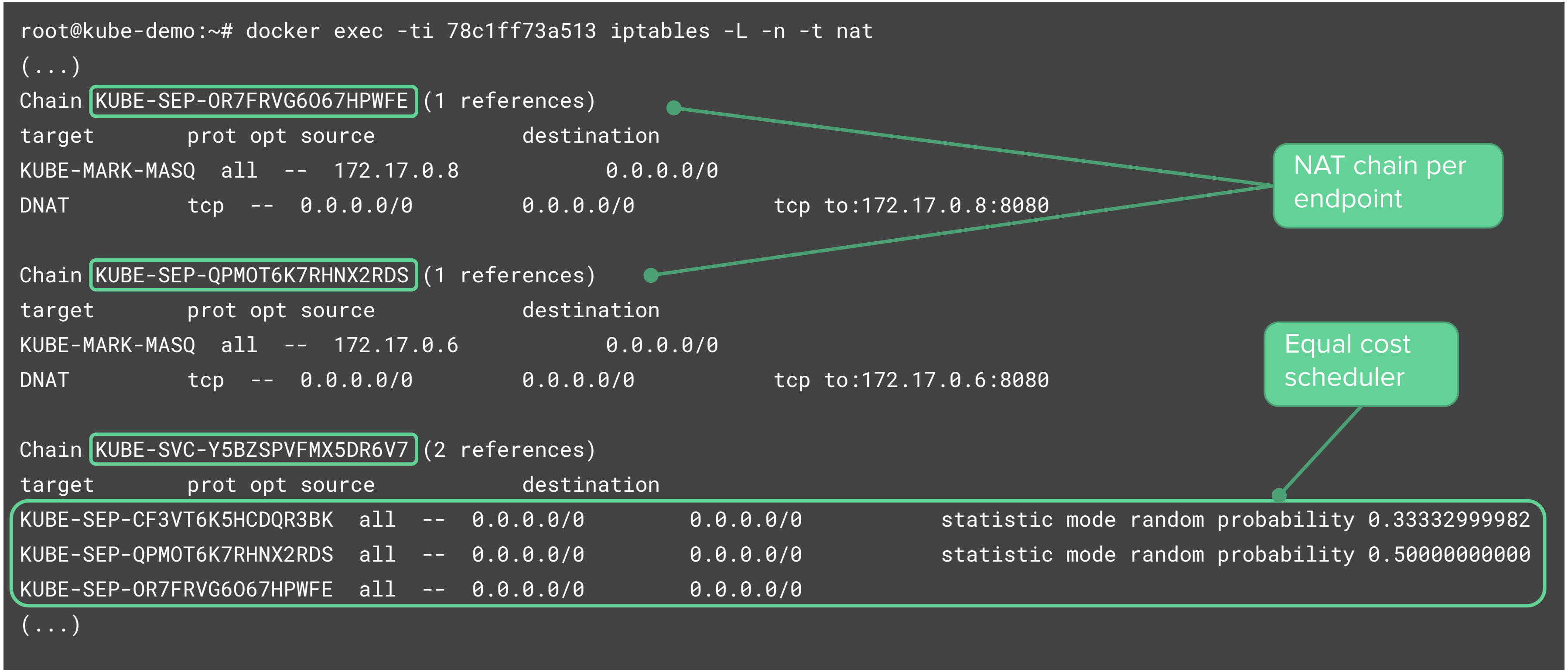
```
root@kube-demo:~# docker exec -ti 78c1ff73a513 iptables -L -n -t nat
(...)
Chain KUBE-NODEPORTS (1 references)
target       prot opt source            destination
KUBE-MARK-MASQ  tcp  --  0.0.0.0/0            0.0.0.0/0           /* default/hello-node: */ tcp dpt:31321
KUBE-SVC-Y5BZSPVFMX5DR6V7  tcp  --  0.0.0.0/0        0.0.0.0/0           /* default/hello-node: */ tcp dpt:31321
(...)
Chain KUBE-SEP-CF3VT6K5HCDQR3BK (1 references)
target       prot opt source            destination
KUBE-MARK-MASQ  all  --  172.17.0.4           0.0.0.0/0
DNAT         tcp  --  0.0.0.0/0         0.0.0.0/0           tcp to:172.17.0.4:8080
(...)
Chain KUBE-SVC-Y5BZSPVFMX5DR6V7 (2 references)
target       prot opt source            destination
KUBE-SEP-CF3VT6K5HCDQR3BK  all  --  0.0.0.0/0        0.0.0.0/0
(...)
```

Networking challenges in Kubernetes



```
root@kube-demo:~# kubectl scale deployment hello-node --replicas=3
root@kube-demo:~# kubectl describe service hello-node
Name:                      hello-node
Namespace:                 default
Labels:                    app=hello-node
Annotations:               <none>
Selector:                  app=hello-node
Type:                      LoadBalancer
IP:                        10.98.101.232
Port:                      <unset>  8080/TCP
TargetPort:                8080/TCP
NodePort:                  <unset>  31321/TCP
Endpoints:                 172.17.0.4:8080,172.17.0.6:8080,172.17.0.8:8080
Session Affinity:          None
External Traffic Policy:  Cluster
Events:                    <none>
```

Networking challenges in Kubernetes



```
root@kube-demo:~# docker exec -ti 78c1ff73a513 iptables -L -n -t nat
(...)
Chain KUBE-SEP-OR7FRVG6O67HPWFE (1 references)
target       prot opt source            destination
KUBE-MARK-MASQ  all  --  172.17.0.8           0.0.0.0/0
DNAT         tcp  --  0.0.0.0/0         0.0.0.0/0           tcp to:172.17.0.8:8080

Chain KUBE-SEP-QPMOT6K7RHNX2RDS (1 references)
target       prot opt source            destination
KUBE-MARK-MASQ  all  --  172.17.0.6           0.0.0.0/0
DNAT         tcp  --  0.0.0.0/0         0.0.0.0/0           tcp to:172.17.0.6:8080

Chain KUBE-SVC-Y5BZSPVFMX5DR6V7 (2 references)
target       prot opt source            destination
KUBE-SEP-CF3VT6K5HCDQR3BK  all  --  0.0.0.0/0        0.0.0.0/0           statistic mode random probability 0.33332999982
KUBE-SEP-QPMOT6K7RHNX2RDS  all  --  0.0.0.0/0        0.0.0.0/0           statistic mode random probability 0.50000000000
KUBE-SEP-OR7FRVG6O67HPWFE  all  --  0.0.0.0/0        0.0.0.0/0
(...)
```

NAT chain per endpoint

Equal cost scheduler

Networking challenges in Kubernetes

Summary

- Plane IP addresses
- Service identified by IP + port
- Dedicated Cluster IP per service
- Dedicated Node Port for service exposure
- Connection tracking, Iptables and proxy for traffic forwarding
- Resolves Clustering and scaling

What's next?

Yet, a lot of kube-proxy corner cases:

- Improve performance and reduce complexity of rules: 4*endpoints
- Support of IPv6 and dual stack
- Load balancing based on mixed protocols or IP only
- Load balance of other protocols FTP, SIP, RDP, SMTP, SYSLOG, LDAP, etc.
- Support of transparent proxy
- Traffic limits per service or endpoints, etc.
- Configurable session persistence
- Better integration with security policies
- Stateful connection tracking replication
- Use of network acceleration techniques

What’s next?

nftlb, the new Zevenet L4 core, based in nftables

https://github.com/zevenet/nftlb

Kubernetes nftlb integration prototype

https://github.com/zevenet/kube-nftlb

Demo

```
root@kube-demo:~# git clone https://github.com/zevenet/kube-nftlb
root@kube-demo:~# ./kube-nftlb/debian_tools_installer.sh
root@kube-demo:~# go get -u github.com/zevenet/kube-nftlb/...
root@kube-demo:~# cd ~/go/src/github.com/zevenet/kube-nftlb/
root@kube-demo:~# kubectl apply -f yaml/give_internet_access_to_pods.yaml
root@kube-demo:~# kubectl apply -f yaml/authentication_system_level_from_pod.yaml
root@kube-demo:~# sh build.sh
root@kube-demo:~# kubectl delete daemonsets -n kube-system kube-proxy
root@kube-demo:~# kubectl apply -f yaml/create_nftlb_as_daemonset.yaml
root@kube-demo:~# kubectl create deployment hello-node \
                           --image=gcr.io/hello-minikube-zero-install/hello-node
root@kube-demo:~# kubectl expose deployment hello-node --type=LoadBalancer --port=8080
root@kube-demo:~# kubectl scale deployment hello-node --replicas=3
root@kube-demo:~# kubectl edit service hello-node
root@kube-demo:~# kubectl delete service hello-node
root@kube-demo:~# kubectl delete deployment hello-node
```

ZEVENET

Happy load balancing!

Zevenet careers
https://www.zevenet.com/careers/