IPVS

This document intends to show users - what is IPVS - difference between IPVS and IPTABLES - how to run kube-proxy in IPVS mode and info on debugging

What is IPVS

IPVS (IP Virtual Server) implements transport-layer load balancing, usually called Layer 4 LAN switching, as part of Linux kernel.

IPVS runs on a host and acts as a load balancer in front of a cluster of real servers. IPVS can direct requests for TCP and UDP-based services to the real servers, and make services of real servers appear as virtual services on a single IP address.

IPVS vs. IPTABLES

IPVS mode was introduced in Kubernetes v1.8, goes beta in v1.9 and GA in v1.11. IPTABLES mode was added in v1.1 and become the default operating mode since v1.2. Both IPVS and IPTABLES are based on `netfilter`. Differences between IPVS mode and IPTABLES mode are as follows:

1. IPVS provides better scalability and performance for large clusters.

2. IPVS supports more sophisticated load balancing algorithms than IPTABLES (least load, least connections, locality, weighted, etc.).

3. IPVS supports server health checking and connection retries, etc.

When IPVS falls back to IPTABLES

IPVS proxier will employ IPTABLES in doing packet filtering, SNAT or masquerade. Specifically, IPVS proxier will use ipset to store source or destination address of traffics that need DROP or do masquerade, to make sure the number of IPTABLES rules be constant, no matter how many services we have.

Here is the table of ipset sets that IPVS proxier used.

set name	members	usage
KUBE-CLUSTER-IP	All service IP + port	Mark-Masq for cases that `masquerade-all=true` or `clusterCIDR` specified
KUBE-LOOP-BACK	All service IP + port + IP	masquerade for solving hairpin purpose
KUBE-EXTERNAL-IP	service external IP + port	masquerade for packages to external IPs
KUBE-LOAD-BALANCER	load balancer ingress IP + port	masquerade for packages to load balancer type service
KUBE-LOAD-BALANCER-LOCAL	LB ingress IP + port with `externalTrafficPolicy=local`	accept packages to load balancer with `externalTrafficPolicy=local`
KUBE-LOAD-BALANCER-FW	load balancer ingress IP + port with `loadBalancerSourceRanges`	package filter for load balancer with `loadBalancerSourceRanges` specified
KUBE-LOAD-BALANCER-SOURCE-CIDR	load balancer ingress IP + port + source CIDR	package filter for load balancer with `loadBalancerSourceRanges` specified
KUBE-NODE-PORT-TCP	nodeport type service TCP port	masquerade for packets to nodePort(TCP)
KUBE-NODE-PORT-LOCAL-TCP	nodeport type service TCP port with `externalTrafficPolicy=local`	accept packages to nodeport service with `externalTrafficPolicy=local`
KUBE-NODE-PORT-UDP	nodeport type service UDP port	masquerade for packets to nodePort(UDP)
KUBE-NODE-PORT-LOCAL-UDP	nodeport type service UDP port with `externalTrafficPolicy=local`	accept packages to nodeport service with `externalTrafficPolicy=local`

IPVS proxier will fall back on IPTABLES in the following scenarios.

1. kube-proxy starts with –masquerade-all=true

If kube-proxy starts with `--masquerade-all=true`, IPVS proxier will masquerade all traffic accessing service Cluster IP, which behaves the same as what IPTABLES proxier. Suppose kube-proxy has flag `--masquerade-all=true` specified, then the IPTABLES installed by IPVS proxier should be like what is shown below.

```
# iptables -t nat -nL

Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes service port

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes service port

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
KUBE-POSTROUTING  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes postrouti

Chain KUBE-MARK-MASQ (2 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            /* kubernetes service traffic
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-LOOP-BACK dst,

Chain KUBE-SERVICES (2 references)
target     prot opt source               destination
KUBE-MARK-MASQ  all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-CLUSTER-I
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-CLUSTER-IP dst,
```

2. Specify cluster CIDR in kube-proxy startup

If kube-proxy starts with `--cluster-cidr=<cidr>`, IPVS proxier will masquerade off-cluster traffic accessing service Cluster IP, which behaves the same as what IPTABLES proxier. Suppose kube-proxy is provided with the cluster cidr `10.244.16.0/24`, then the IPTABLES installed by IPVS proxier should be like what is shown below.

```
# iptables -t nat -nL

Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes service port

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes service port

Chain POSTROUTING (policy ACCEPT)
```

```
target     prot opt source               destination
KUBE-POSTROUTING  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes postrouti

Chain KUBE-MARK-MASQ (3 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            /* kubernetes service traffi
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-LOOP-BACK dst,

Chain KUBE-SERVICES (2 references)
target     prot opt source               destination
KUBE-MARK-MASQ  all  -- !10.244.16.0/24       0.0.0.0/0            match-set KUBE-CLUSTER-I
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-CLUSTER-IP dst,
```

3. Load Balancer type service

For loadBalancer type service, IPVS proxier will install IPTABLES with match of ipset KUBE-LOAD-BALANCER. Specially when service's LoadBalancerSourceRanges is specified or specified externalTrafficPolicy=local, IPVS proxier will create ipset sets KUBE-LOAD-BALANCER-LOCAL/KUBE-LOAD-BALANCER-FW/KUBE-LOAD-BALANCE and install IPTABLES accordingly, which should look like what is shown below.

```
# iptables -t nat -nL

Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes service port

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes service port

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
KUBE-POSTROUTING  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes postrouti

Chain KUBE-FIREWALL (1 references)
target     prot opt source               destination
RETURN     all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-LOAD-BALANCER-S
KUBE-MARK-DROP  all  --  0.0.0.0/0            0.0.0.0/0

Chain KUBE-LOAD-BALANCER (1 references)
target     prot opt source               destination
KUBE-FIREWALL  all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-LOAD-BALANC
```

```
RETURN     all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-LOAD-BALANCER-L
KUBE-MARK-MASQ  all  --  0.0.0.0/0            0.0.0.0/0

Chain KUBE-MARK-DROP (1 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 (2 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            /* kubernetes service traffic
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-LOOP-BACK dst,

Chain KUBE-SERVICES (2 references)
target     prot opt source               destination
KUBE-LOAD-BALANCER  all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-LOAD-B
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-LOAD-BALANCER d
```

4. NodePort type service

For NodePort type service, IPVS proxier will install IPTABLES with match of ipset `KUBE-NODE-PORT-TCP/KUBE-NODE-PORT-UDP`. When specified `externalTrafficPolicy=local`, IPVS proxier will create ipset sets `KUBE-NODE-PORT-LOCAL-TCP/KUBE-NODE-PORT-LOCAL-UDP` and install IPTABLES accordingly, which should look like what is shown below.

Suppose service with TCP type nodePort.

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes service port

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes service port

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
KUBE-POSTROUTING  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes postrouti

Chain KUBE-MARK-MASQ (2 references)
target     prot opt source               destination
MARK       all  --  0.0.0.0/0            0.0.0.0/0            MARK or 0x4000

Chain KUBE-NODE-PORT (1 references)
```

```
target     prot opt source               destination
RETURN     all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-NODE-PORT-LOCAL
KUBE-MARK-MASQ  all  --  0.0.0.0/0            0.0.0.0/0

Chain KUBE-POSTROUTING (1 references)
target     prot opt source               destination
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes service traffic
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-LOOP-BACK dst,

Chain KUBE-SERVICES (2 references)
target     prot opt source               destination
KUBE-NODE-PORT  all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-NODE-PORT-
```

5. Service with externalIPs specified

For service with `externalIPs` specified, IPVS proxier will install IPTABLES with match of ipset `KUBE-EXTERNAL-IP`, Suppose we have service with `externalIPs` specified, IPTABLES rules should look like what is shown below.

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes service port

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes service port

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
KUBE-POSTROUTING  all  --  0.0.0.0/0            0.0.0.0/0            /* kubernetes postrouti

Chain KUBE-MARK-MASQ (2 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            /* kubernetes service traffic
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-LOOP-BACK dst,

Chain KUBE-SERVICES (2 references)
target     prot opt source               destination
KUBE-MARK-MASQ  all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-EXTERNAL-I
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-EXTERNAL-IP dst
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0            match-set KUBE-EXTERNAL-IP dst
```

Run kube-proxy in IPVS mode

Currently, local-up scripts, GCE scripts and kubeadm support switching IPVS proxy mode via exporting environment variables or specifying flags.

Prerequisite

Ensure IPVS required kernel modules (**Notes**: use `nf_conntrack` instead of `nf_conntrack_ipv4` for Linux kernel 4.19 and later)

```
ip_vs
ip_vs_rr
ip_vs_wrr
ip_vs_sh
nf_conntrack_ipv4
```

1. have been compiled into the node kernel. Use

```
grep -e ipvs -e nf_conntrack_ipv4 /lib/modules/$(uname -r)/modules.builtin
```

and get results like the followings if compiled into kernel.

```
kernel/net/ipv4/netfilter/nf_conntrack_ipv4.ko
kernel/net/netfilter/ipvs/ip_vs.ko
kernel/net/netfilter/ipvs/ip_vs_rr.ko
kernel/net/netfilter/ipvs/ip_vs_wrr.ko
kernel/net/netfilter/ipvs/ip_vs_lc.ko
kernel/net/netfilter/ipvs/ip_vs_wlc.ko
kernel/net/netfilter/ipvs/ip_vs_fo.ko
kernel/net/netfilter/ipvs/ip_vs_ovf.ko
kernel/net/netfilter/ipvs/ip_vs_lblc.ko
kernel/net/netfilter/ipvs/ip_vs_lblcr.ko
kernel/net/netfilter/ipvs/ip_vs_dh.ko
kernel/net/netfilter/ipvs/ip_vs_sh.ko
kernel/net/netfilter/ipvs/ip_vs_sed.ko
kernel/net/netfilter/ipvs/ip_vs_nq.ko
kernel/net/netfilter/ipvs/ip_vs_ftp.ko
```

OR

2. have been loaded.

```
# load module <module_name>
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack_ipv4

# to check loaded modules, use
```

```
lsmod | grep -e ip_vs -e nf_conntrack_ipv4
# or
cut -f1 -d " "  /proc/modules | grep -e ip_vs -e nf_conntrack_ipv4
```

Packages such as `ipset` should also be installed on the node before using IPVS mode.

Kube-proxy will fall back to IPTABLES mode if those requirements are not met.

Local UP Cluster

Kube-proxy will run in IPTABLES mode by default in a local-up cluster.

To use IPVS mode, users should export the env `KUBE_PROXY_MODE=ipvs` to specify the IPVS mode before starting the cluster:

```
# before running `hack/local-up-cluster.sh`
export KUBE_PROXY_MODE=ipvs
```

GCE Cluster

Similar to local-up cluster, kube-proxy in clusters running on GCE run in IPTABLES mode by default. Users need to export the env `KUBE_PROXY_MODE=ipvs` before starting a cluster:

```
#before running one of the commands chosen to start a cluster:
# curl -sS https://get.k8s.io | bash
# wget -q -O - https://get.k8s.io | bash
# cluster/kube-up.sh
export KUBE_PROXY_MODE=ipvs
```

Cluster Created by Kubeadm

If you are using kubeadm with a configuration file, you have to add mode: ipvs in a KubeProxyConfiguration (separated by – that is also passed to kubeadm init).

```
...
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
mode: ipvs
...
```

before running

```
kubeadm init --config <path_to_configuration_file>
```

to specify the ipvs mode before deploying the cluster.

Notes If ipvs mode is successfully on, you should see IPVS proxy rules (use `ipvsadm`) like

```
 # ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.0.0.1:443 rr persistent 10800
  -> 192.168.0.1:6443             Masq    1      1          0
```

or similar logs occur in kube-proxy logs (for example, `/tmp/kube-proxy.log` for local-up cluster) when the local cluster is running:

```
Using ipvs Proxier.
```

While there is no IPVS proxy rules or the following logs occurs indicate that the kube-proxy fails to use IPVS mode:

```
Can't use ipvs proxier, trying iptables proxier
Using iptables Proxier.
```

See the following section for more details on debugging.

Debug

Check IPVS proxy rules

Users can use `ipvsadm` tool to check whether kube-proxy are maintaining IPVS rules correctly. For example, we have the following services in the cluster:

```
 # kubectl get svc --all-namespaces
NAMESPACE     NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)         AGE
default       kubernetes   ClusterIP   10.0.0.1     <none>        443/TCP         1d
kube-system   kube-dns     ClusterIP   10.0.0.10    <none>        53/UDP,53/TCP   1d
```

We may get IPVS proxy rules like:

```
 # ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.0.0.1:443 rr persistent 10800
  -> 192.168.0.1:6443             Masq    1      1          0
TCP  10.0.0.10:53 rr
  -> 172.17.0.2:53                Masq    1      0          0
UDP  10.0.0.10:53 rr
  -> 172.17.0.2:53                Masq    1      0          0
```

Why kube-proxy can’t start IPVS mode

Use the following check list to help you solve the problems:

1. Specify proxy-mode=ipvs

Check whether the kube-proxy mode has been set to `ipvs`.

2. Install required kernel modules and packages

Check whether the IPVS required kernel modules have been compiled into the kernel and packages installed. (see Prerequisite)