Konference Kubernetes II.

OpenShift SDN

Jan Dvořák

Praha 24.9.2019

Openshift introduction

- OpenShift Container Platform
- OpenShift Origin (upstream)
- RedHat OpenShift Online and Openshift Dedicated

Network stack

SDN is based on the plugin modules

- ovs-subnet:
  - open flat network
  - no isolation - all pods can communicate with any other endpoint or svc in the cluster
- ovs-multitenant
  - isolation on the namespace level
- ovs-networkpolicy
  - all features from subnet and multitenant plugin plus additional isolation features

Supported 3rd party vendor modules

- Cisco Contiv (™)
- Juniper Contrail (™)
- Nokia Nuage (™)
- Tigera Calico (™)
- VMware NSX-T (™)

Basic entities

Namespaces/Projects

- group of the objects(deployments, statefullsets, pods, services, ingress rules, etc.)
- NetworkPolicy are setup on the namespace level

Nodes

- hosting for pods
- three types: Master, Infra, Compute
- connected between each other, usually in one ip range

Pods

- environment for one or multiple containers
- ensure sharing storage and network stack for multiple containers
- own ip range

Network entities

Services

- Ensure communication to pods
- Without external ip just forward traffic inside the cluster
- **Managed by iptables rules on the nodes**
- Own dns name <name>.<ns>.svc.cluster.local
- supported types: ClusterIP, LoadBalancer, NodePort

Routes

- Publish services outside cluster
- Support for multiple protocols like http, https, tls(with sni) and websocket
- Possible ssl termination, reencryption etc
- Available plugins: haproxy or F5

Ingress

- Publish services outside cluster
- Any protocol not only http
- Managed by special controller
- Possible routing or loadbalancing on L7
- Lots of implementation, usually on haproxy or nginx
- Default kubernetes object (routes are openshift specific)

Egress

- Configure access from the pods to outer world.
- Manage on the namespace layer or by separate controllers
- Outgoing traffic is managed by masquerade on the last node in the path

NetworkPolicy

- Define access policy to the namespace objects (pods,services) based on various criteria
- Default policy is allow any traffic to/from pods/endpoints

Requirements and Tools

Iptables

- egress communication
- **services management**
- additional isolation

OpenvSwitch

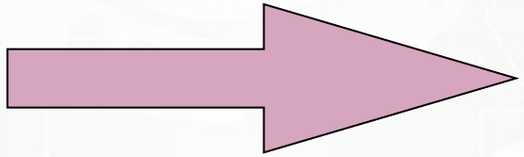
- communication between nodes
- routing traffic to each pod
- network policy implementation
- namespace identification though SDN

Linux namespaces

- isolation inside node
- kernel resource partitioning

Other tools and services

- IPfailover (keepalived),
  HAproxy, DNS, docker regiistry
  etc.



Horizontal Scaling

High Availability

Isolation

Open vSwitch

"Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license. It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols (e.g. NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag). In addition, it is designed to support distribution across multiple physical servers similar to VMware's vNetwork distributed vswitch or Cisco's Nexus 1000V"

source: www.openvswitch.org

Key features:

- OpenFlow support (Openshift 3.11 uses v1.3)

- VXLAN support

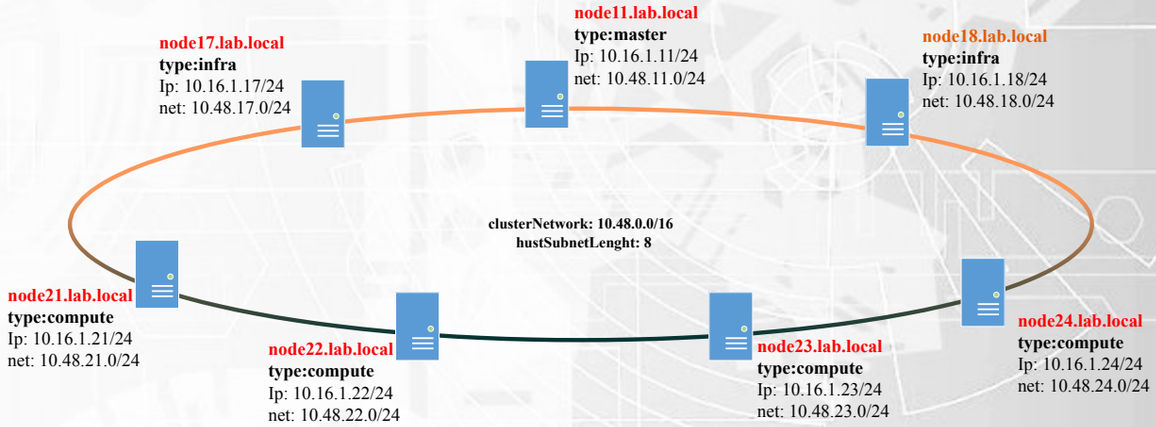
Configuration example:

table=20	priority=100	ip,in_port=5,nw_src=10.48.23.6	actions=load:0x3e2d21->NXM_NX_REG0[]	goto_table:21
table=25	priority=100	ip,nw_src=10.48.23.6	actions=load:0x3e2d21->NXM_NX_REG0[]	goto_table:30
table=40	priority=100	arp,arp_tpa=10.48.23.6	actions=output:5	
table=70	priority=100	ip,nw_dst=10.48.23.6	actions=load:0x3e2d21->NXM_NX_REG1[],load:0x5->NXM_NX_REG2[]	goto_table:80

Flat network concept

- whole cluster network has a one big ip range
- parts of this range are assigned to the nodes based on the hostsubnet definition
- CIDR and hostSubnetLength must be defined during the installation process
- impact on the scalability of the cluster:
  - for example: hostSubnetLength:8 -> max 253 pods on the node

```
# part of the /etc/origin/master/master-config.yaml
.
.
networkConfig:
  clusterNetworks:
  - cidr: 10.48.0.0/16
    hostSubnetLength: 8
  externalIPNetworkCIDRs:
  - 0.0.0.0/0
  networkPluginName: redhat/openshift-ovs-networkpolicy
  serviceNetworkCIDR: 10.49.0.0/16
```



Flat network concept

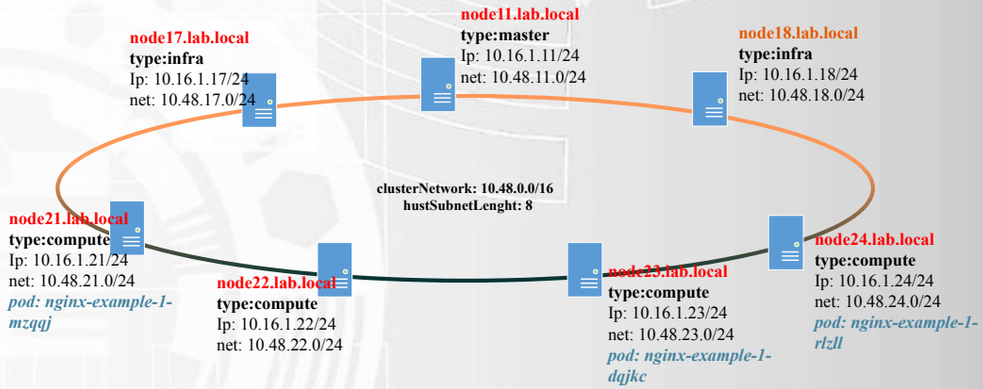
the hostsubnet objects define the ip range for the pods placed on the target node

IPs of the pods are dynamic and assigned throughout the deployment process

```
# oc get hostsubnet
NAME                HOST                HOST IP      SUBNET          EGRESS CIDRS        EGRESS IPS
node11.lab.local    node11.lab.local    10.16.1.11   10.48.11.0/24   []                  []
node17.lab.local    node17.lab.local    10.16.1.17   10.48.17.0/24   [10.16.1.224/27]    [10.16.1.225]
node18.lab.local    node18.lab.local    10.16.1.18   10.48.18.0/24   [10.16.1.224/27]    []
node21.lab.local    node21.lab.local    10.16.1.21   10.48.21.0/24   []                  []
node22.lab.local    node22.lab.local    10.16.1.22   10.48.22.0/24   []                  []
node23.lab.local    node23.lab.local    10.16.1.23   10.48.23.0/24   []                  []
node24.lab.local    node24.lab.local    10.16.1.24   10.48.24.0/24   []                  []
```

```
# oc get pods -o wide
NAME                    READY     STATUS    RESTARTS   AGE       IP            NODE
nginx-example-1-dqjkc   1/1       Running   3          5d        10.48.23.9    node23.lab.local
nginx-example-1-mzqgj   1/1       Running   3          5d        10.48.21.18   node21.lab.local
nginx-example-1-rlzll   1/1       Running   3          4d        10.48.24.9    node24.lab.local
```

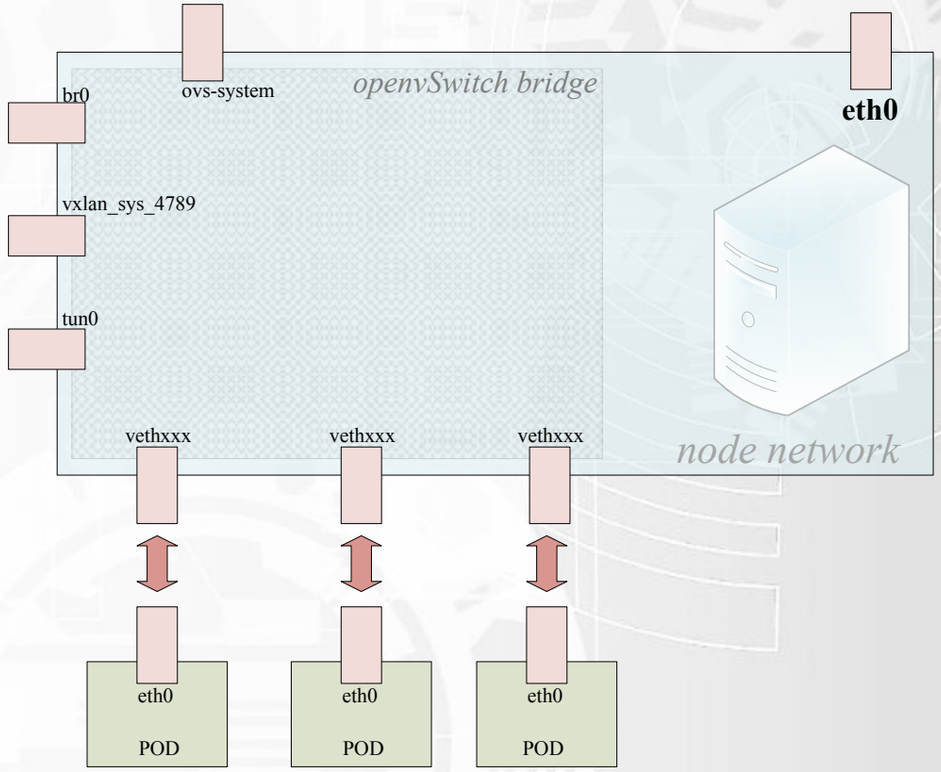
```
# part of the /etc/origin/master/master-config.yaml
.
.
networkConfig:
  clusterNetworks:
  - cidr: 10.48.0.0/16
    hostSubnetLength: 8
  externalIPNetworkCIDRs:
  - 0.0.0.0/0
  networkPluginName: redhat/openshift-ovs-networkpolicy
  serviceNetworkCIDR: 10.49.0.0/16
```



Nodes network devices

- ovs has a different naming convention than os
- eth0 is not part of the bridge!
- tun0 act like a default gateway for pods
- br0 and ovs-system represent the switch
- vxlan_sys_4789 is the adapter for tunneling

```
sh-4.2# ovs-vsctl show
6cc2c284-0c59-4c1b-846f-2e5c6cca6da9
    Bridge "br0"
        fail_mode: secure
        Port "vxlan0"
            Interface "vxlan0"
                type: vxlan
                options: {dst_port="4789", key=flow, remote_ip=flow}
        Port "br0"
            Interface "br0"
                type: internal
        Port "veth6ef56296"
            Interface "veth6ef56296"
        Port "tun0"
            Interface "tun0"
                type: internal
    ovs_version: "2.7.0"
```

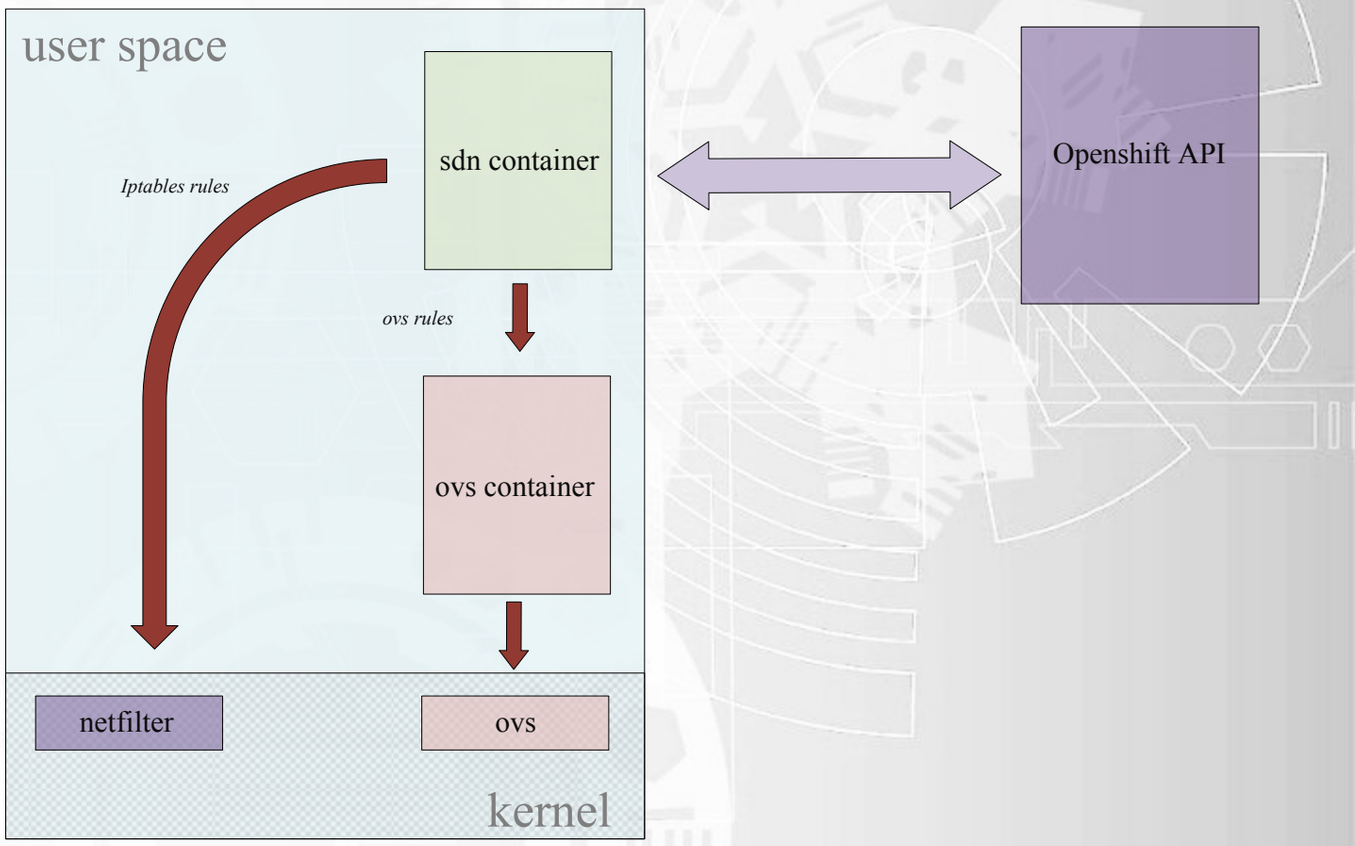


Nodes environment

- two deamonsets are defined in openshift-sdn namespace
- sdn container = controller
- ovs container controls instances of the ovs
- controller act like a operator

kube-proxy:

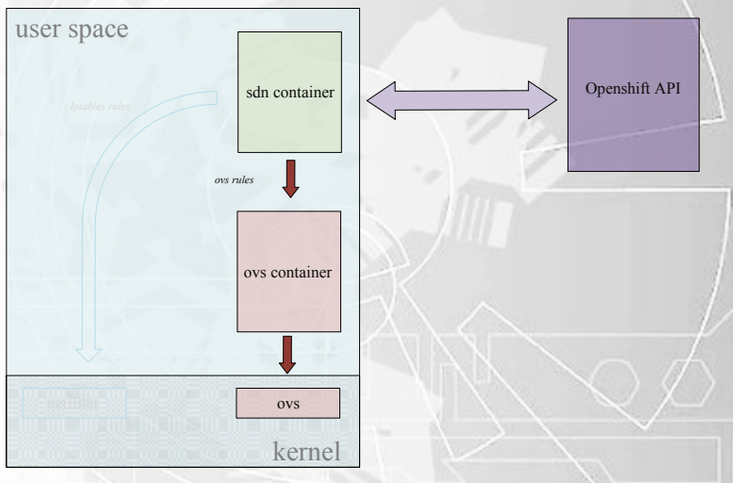
- part of the sdn container
- two mode: userspace or iptables
- provides iptables rule for defined services



New pod deployment

example: *# oc scale dc nginx-example --replicas=3*
deploymentconfig.apps.openshift.io/nginx-example scaled

```
# oc get pods -o wide
NAME                    READY     STATUS    RESTARTS   AGE       IP            NODE                NOMINATED NODE
nginx-example-1-7jv8q   1/1       Running   0          25s       10.48.24.11   node24.lab.local    <none>
nginx-example-1-m6984   1/1       Running   0          25s       10.48.23.11   node23.lab.local    <none>
nginx-example-1-vwnbq   1/1       Running   0          25s       10.48.21.20   node21.lab.local    <none>
```



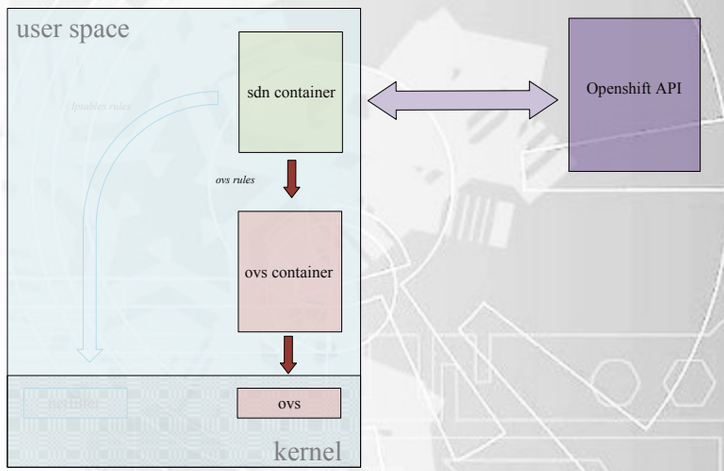
New objects:

- vethxxxx device
- pod eth0 device and ip cfg
- ovs rules for pod access
- pod and containers

New pod deployment

example: *# oc scale dc nginx-example --replicas=3*
deploymentconfig.apps.openshift.io/nginx-example scaled

```
# oc get pods -o wide
NAME                    READY     STATUS    RESTARTS   AGE       IP            NODE                NOMINATED NODE
nginx-example-1-7jv8q   1/1       Running   0          25s       10.48.24.11   node24.lab.local    <none>
nginx-example-1-m6984   1/1       Running   0          25s       10.48.23.11   node23.lab.local    <none>
nginx-example-1-vwnbq   1/1       Running   0          25s       10.48.21.20   node21.lab.local    <none>
```



New objects:

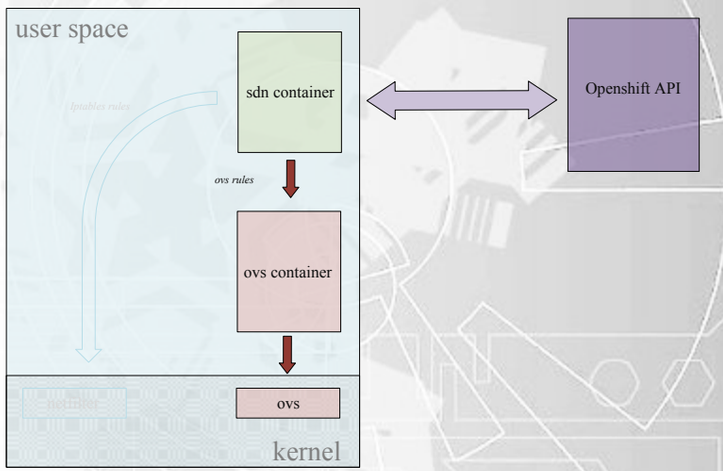
- vethxxxx device
- pod eth0 device and ip cfg
- ovs rules for pod access
- pod and containers

```
# New ovs rules on node21.lab.local

cookie=0x0, table=20, priority=100,arp,in_port=5,arp_spa=10.48.21.20,arp_sha=00:00:0a:30:15:14/00:00:ff:ff:ff:ff actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:21
cookie=0x0, table=20, priority=100,ip,in_port=5,nw_src=10.48.21.20 actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:21
cookie=0x0, table=25, priority=100,ip,nw_src=10.48.21.20 actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:30
cookie=0x0, table=40, priority=100,arp,arp_tpa=10.48.21.20 actions=output:5
cookie=0x0, table=70, priority=100,ip,nw_dst=10.48.21.20 actions=load:0x3e2d21->NXM_NX_REG1[],load:0x5->NXM_NX_REG2[],goto_table:80
```

New pod deployment

example: *# oc scale dc nginx-example --replicas=3*
deploymentconfig.apps.openshift.io/nginx-example scaled

```
# oc get pods -o wide
NAME                    READY     STATUS    RESTARTS   AGE       IP            NODE                NOMINATED NODE
nginx-example-1-7jv8q   1/1       Running   0          25s       10.48.24.11   node24.lab.local    <none>
nginx-example-1-m6984   1/1       Running   0          25s       10.48.23.11   node23.lab.local    <none>
nginx-example-1-vwnbq   1/1       Running   0          25s       10.48.21.20   node21.lab.local    <none>
```



New objects:

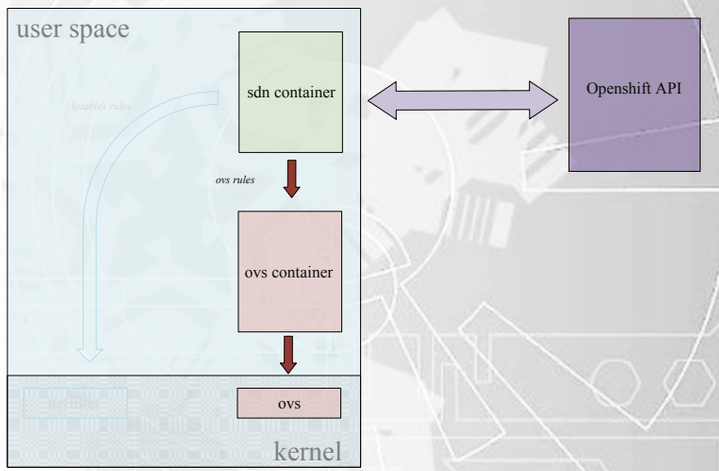
- vethxxxx device
- pod eth0 device and ip cfg
- ovs rules for pod access
- pod and containers

```
# New ovs rules on node23.lab.local

cookie=0x0, table=20, priority=100,arp,in_port=5,arp_spa=10.48.23.11,arp_sha=00:00:0a:30:17:0b/00:00:ff:ff:ff:ff actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:21
cookie=0x0, table=20, priority=100,ip,in_port=5,nw_src=10.48.23.11 actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:21
cookie=0x0, table=25, priority=100,ip,nw_src=10.48.23.11 actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:30
cookie=0x0, table=40, priority=100,arp,arp_tpa=10.48.23.11 actions=output:5
cookie=0x0, table=70, priority=100,ip,nw_dst=10.48.23.11 actions=load:0x3e2d21->NXM_NX_REG1[],load:0x5->NXM_NX_REG2[],goto_table:80
```

New pod deployment

example: *# oc scale dc nginx-example --replicas=3*
deploymentconfig.apps.openshift.io/nginx-example scaled

```
# oc get pods -o wide
NAME                    READY     STATUS    RESTARTS   AGE       IP            NODE                NOMINATED NODE
nginx-example-1-7jv8q   1/1       Running   0          25s       10.48.24.11   node24.lab.local    <none>
nginx-example-1-m6984   1/1       Running   0          25s       10.48.23.11   node23.lab.local    <none>
nginx-example-1-vwnbq   1/1       Running   0          25s       10.48.21.20   node21.lab.local    <none>
```



New objects:

- vethxxxx device
- pod eth0 device and ip cfg
- ovs rules for pod access
- pod and containers

```
# New ovs rules on node24.lab.local

cookie=0x0, table=20, priority=100,arp,in_port=5,arp_spa=10.48.24.11,arp_sha=00:00:0a:30:18:0b/00:00:ff:ff:ff:ff actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:21
cookie=0x0, table=20, priority=100,ip,in_port=5,nw_src=10.48.24.11 actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:21
cookie=0x0, table=25, priority=100,ip,nw_src=10.48.24.11 actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:30
cookie=0x0, table=40, priority=100,arp,arp_tpa=10.48.24.11 actions=output:5
cookie=0x0, table=70, priority=100,ip,nw_dst=10.48.24.11 actions=load:0x3e2d21->NXM_NX_REG1[],load:0x5->NXM_NX_REG2[],goto_table:80
```

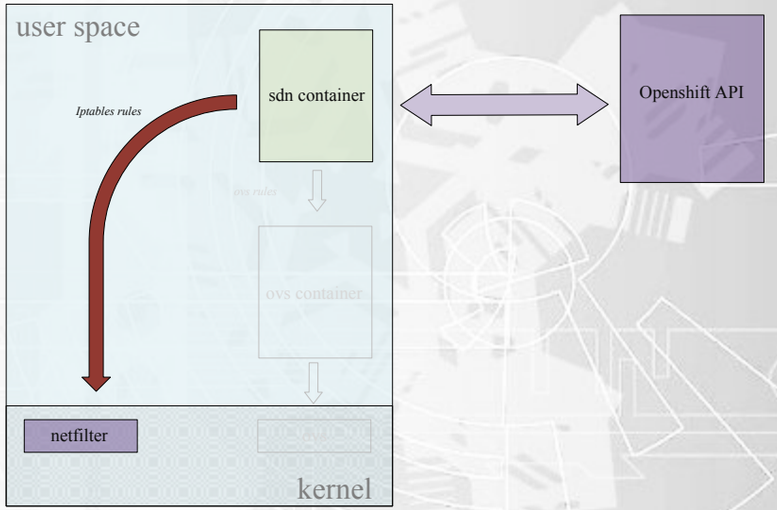
Expose service

example: *# oc expose dc nginx-example*

```
NAME            TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
nginx-example   ClusterIP   10.49.62.91   <NONE>        80/TCP    18m
```

New objects:

- just iptables rules!
- no ip, no ovs rules

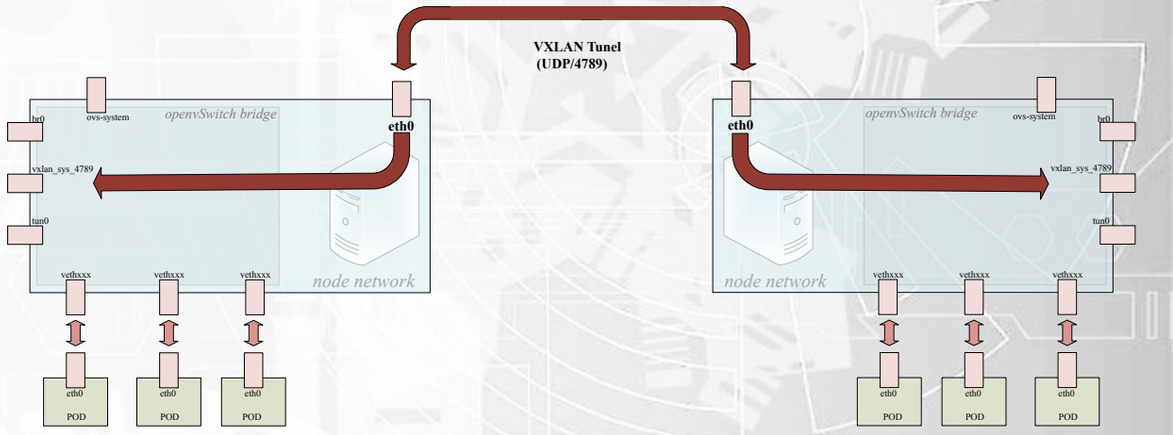



```
# Incoming flow distribuion

-A KUBE-SERVICES ! -s 10.48.0.0/16 -d 10.49.62.91/32 -p tcp -m comment --comment "webapp/nginx-example: cluster IP" -m tcp --dport 80 -j KUBE-MARK-MASQ
-A KUBE-SERVICES -d 10.49.62.91/32 -p tcp -m comment --comment "webapp/nginx-example: cluster IP" -m tcp --dport 80 -j KUBE-SVC-OVPIOQYLPH3A6XTA
-A KUBE-SVC-OVPIOQYLPH3A6XTA -m comment --comment "webapp/nginx-example:" -m statistic --mode random --probability 0.33332999982 -j KUBE-SEP-A3FPCBMX64CTFHWT
-A KUBE-SVC-OVPIOQYLPH3A6XTA -m comment --comment "webapp/nginx-example:" -m statistic --mode random --probability 0.50000000000 -j KUBE-SEP-5IC7UQEZWKI27EZG
-A KUBE-SVC-OVPIOQYLPH3A6XTA -m comment --comment "webapp/nginx-example:" -j KUBE-SEP-74PAZBA2CEDRF737

# Each pod has a own chain!

-A KUBE-SEP-5IC7UQEZWKI27EZG -s 10.48.22.19/32 -m comment --comment "webapp/nginx-example:" -j KUBE-MARK-MASQ
-A KUBE-SEP-5IC7UQEZWKI27EZG -p tcp -m comment --comment "webapp/nginx-example:" -m tcp -j DNAT --to-destination 10.48.22.19:8080

-A KUBE-SEP-74PAZBA2CEDRF737 -s 10.48.23.14/32 -m comment --comment "webapp/nginx-example:" -j KUBE-MARK-MASQ
-A KUBE-SEP-74PAZBA2CEDRF737 -p tcp -m comment --comment "webapp/nginx-example:" -m tcp -j DNAT --to-destination 10.48.23.14:8080

-A KUBE-SEP-A3FPCBMX64CTFHWT -s 10.48.21.23/32 -m comment --comment "webapp/nginx-example:" -j KUBE-MARK-MASQ
-A KUBE-SEP-A3FPCBMX64CTFHWT -p tcp -m comment --comment "webapp/nginx-example:" -m tcp -j DNAT --to-destination 10.48.21.23:8080
```

Virtual extensible LAN (VXLAN)

https://tools.ietf.org/html/rfc7348

- encapsulate ethernet frames within UDP datagrams
- up to 16 million logical networks (as opposed to VLAN => 4094 networks)
- key technology in Openshift SDN
- ensures:
  - communication between hostsubnets and pods
  - **transferring information about openshift namespaces across the cluster**
    - **namespace netid = vxlan vnid !**
- uses udp port 4789
- supported in ovs, possible routing and encapsulation based on the openflow rules




```
# oc get netnamespace | grep client
client                                8078500    []

[root@dl380 ~]# oc get pods -o wide
NAME           IP             NODE
sshd-5-hknr9   10.48.24.27    node24.lab.local

# oc get svc -n webapp
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)
nginx-example   ClusterIP   10.49.145.117   10.16.1.193    8080/TCP
```

```
Frame 1: 124 bytes on wire (992 bits), 124 bytes captured (992 bits)
Ethernet II, Src: RealtekU_e3:e9:fd (52:54:00:e3:e9:fd), Dst: RealtekU_a9:bf:84 (52:54
Internet Protocol Version 4, Src: 10.16.1.24, Dst: 10.16.1.21
User Datagram Protocol, Src Port: 44445, Dst Port: 4789
Virtual eXtensible Local Area Network
    Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 8078500
    Reserved: 0
Ethernet II, Src: 4a:7c:d0:d4:07:57 (4a:7c:d0:d4:07:57), Dst: 0a:58:0a:30:15:20 (0a:58
Internet Protocol Version 4, Src: 10.48.24.27, Dst: 10.48.21.32
Transmission Control Protocol, Src Port: 34794, Dst Port: 8080, Seq: 0, Len: 0
```

Packet flow: pod->service->pods

// on node22.lab.local

tcpdump -nnnv -xx -e -i tun0 'port 52222'

tcpdump: listening on tun0, link-type EN10MB (Ethernet), capture size 262144 bytes

08:58:30.852663 **0a:58:0a:30:16:59 > 6e:93:62:95:75:84**, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 64282, offset 0, flags [DF], proto TCP (6), length 60)

10.48.22.89.52222 > 10.49.145.117.80: Flags [S], cksum 0x832a (correct), seq 3507650740, win 28200, options [mss 1410,sackOK,TS val 15498158 ecr 0,nop,wscale 7], length 0

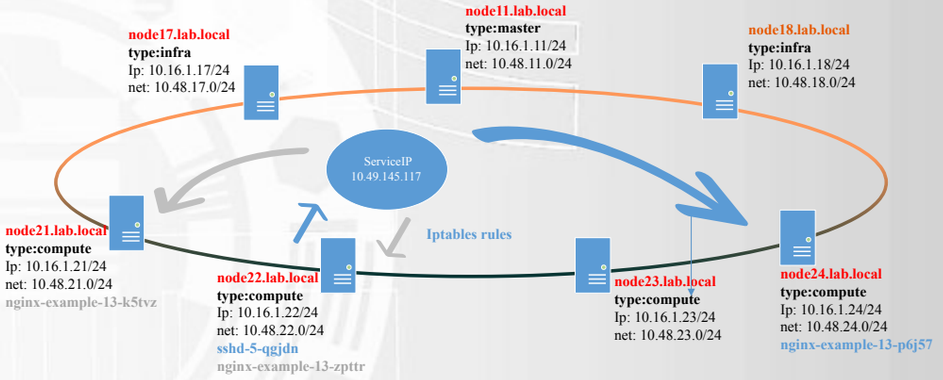
0x0000: 6e93 6295 7584 0a58 0a30 1659 0800 4500

0x0010: 003c fb1a 4000 4006 8372 0a30 1659 0a31

0x0020: 9175 cbfe 0050 d112 80b4 0000 0000 a002

0x0030: 6e28 832a 0000 0204 0582 0402 080a 00ec

0x0040: 7bae 0000 0000 0103 0307



Packet flow: pod->service->pods

// on node22.lab.local

tcpdump -nnnv -xx -e -i tun0 'port 52222'

tcpdump: listening on tun0, link-type EN10MB (Ethernet), capture size 262144 bytes

08:58:30.852663 **0a:58:0a:30:16:59 > 6e:93:62:95:75:84**, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 64282, offset 0, flags [DF], proto TCP (6), length 60)

10.48.22.89.52222 > 10.49.145.117.80: Flags [S], cksum 0x832a (correct), seq 3507650740, win 28200, options [mss 1410,sackOK,TS val 15498158 ecr 0,nop,wscale 7], length 0

0x0000: 6e93 6295 7584 0a58 0a30 1659 0800 4500

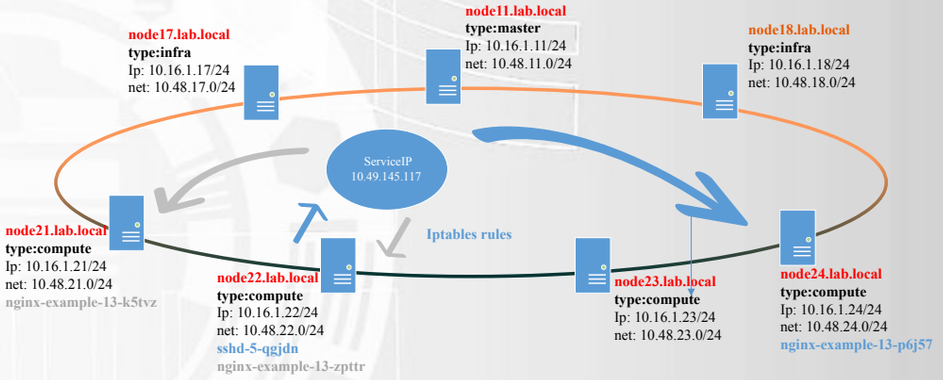
0x0010: 003c fb1a 4000 4006 8372 0a30 1659 0a31

0x0020: 9175 cbfe 0050 d112 80b4 0000 0000 a002

0x0030: 6e28 832a 0000 0204 0582 0402 080a 00ec

0x0040: 7bae 0000 0000 0103 0307

```
# oc get pods -o wide
NAME           READY     STATUS    RESTARTS   AGE       IP             NODE
sshd-5-qgjdn   1/1       Running   1          18h       10.48.22.89    node22.lab.local

# oc get pods -o wide -n webapp
NAME                        READY     STATUS    RESTARTS   AGE       IP             NODE
nginx-example-13-k5tvz      2/2       Running   0          1h        10.48.21.43    node21.lab.local
nginx-example-13-p6j57      2/2       Running   0          1h        10.48.24.87    node24.lab.local
nginx-example-13-zpttr      2/2       Running   0          1h        10.48.22.100   node22.lab.local
```



Packet flow: pod->service->pods

// on node22.lab.local

tcpdump -nnnv -xx -e -i tun0 'port 52222'

tcpdump: listening on tun0, link-type EN10MB (Ethernet), capture size 262144 bytes

08:58:30.852663 **0a:58:0a:30:16:59 > 6e:93:62:95:75:84**, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 64282, offset 0, flags [DF], proto TCP (6), length 60)

10.48.22.89.52222 > 10.49.145.117.80: Flags [S], cksum 0x832a (correct), seq 3507650740, win 28200, options [mss 1410,sackOK,TS val 15498158 ecr 0,nop,wscale 7], length 0

0x0000: 6e93 6295 7584 0a58 0a30 1659 0800 4500

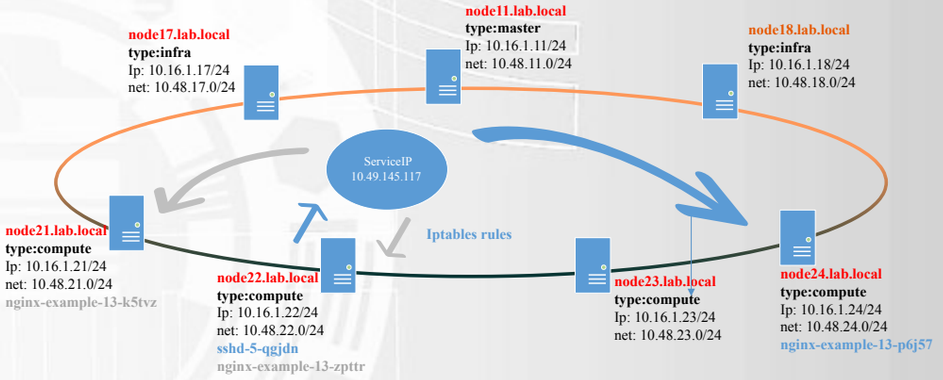
0x0010: 003c fb1a 4000 4006 8372 0a30 1659 0a31

0x0020: 9175 cbfe 0050 d112 80b4 0000 0000 a002

0x0030: 6e28 832a 0000 0204 0582 0402 080a 00ec

0x0040: 7bae 0000 0000 0103 0307

```
1 # oc rsh sshd-5-qgjdn
2 sh-4.2# ncat -p 52222 10.49.145.117 80 -v
3 Ncat: Version 7.50 ( https://nmap.org/ncat )
4 Ncat: Connected to 10.49.145.117:80.
```



Packet flow: pod->service->pods

// on node22.lab.local

tcpdump -nnnv -xx -e -i tun0 'port 52222'

tcpdump: listening on tun0, link-type EN10MB (Ethernet), capture size 262144 bytes

08:58:30.852663 **0a:58:0a:30:16:59 > 6e:93:62:95:75:84**, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 64282, offset 0, flags [DF], proto TCP (6), length 60)

10.48.22.89.52222 > 10.49.145.117.80: Flags [S], cksum 0x832a (correct), seq 3507650740, win 28200, options [mss 1410,sackOK,TS val 15498158 ecr 0,nop,wscale 7], length 0

0x0000: 6e93 6295 7584 0a58 0a30 1659 0800 4500

0x0010: 003c fb1a 4000 4006 8372 0a30 1659 0a31

0x0020: 9175 cbfe 0050 d112 80b4 0000 0000 a002

0x0030: 6e28 832a 0000 0204 0582 0402 080a 00ec

0x0040: 7bae 0000 0000 0103 0307

// on node24.lab.local

tcpdump -nnnv -xx -e -i eth0 'port 52222'

tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes

08:39:51.672196 **6e:93:62:95:75:84 > 0a:58:0a:30:18:57**, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 63, id 20764, offset 0, flags [DF], proto TCP (6), length 60)

10.48.22.89.52222 > 10.48.24.87.8080: Flags [S], cksum 0xe7b2 (correct), seq 3180017406, win 27960, options [mss 1410,sackOK,TS val 14378978 ecr 14378630,nop,wscale 7], length 0

0x0000: 0a58 0a30 1857 6e93 6295 7584 0800 4500

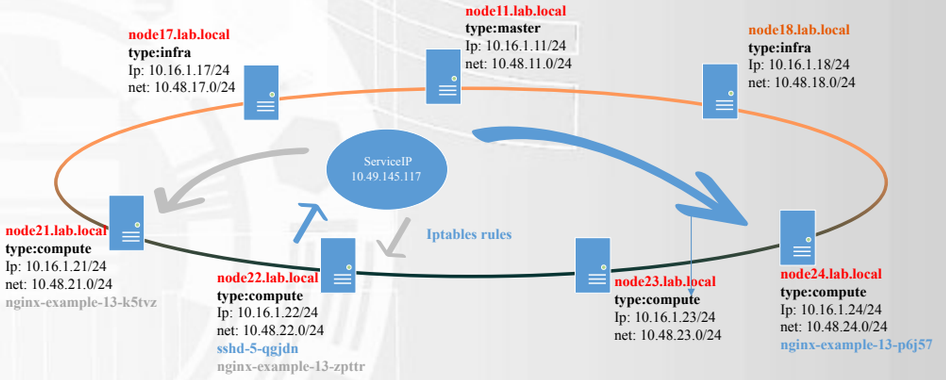
0x0010: 003c 511c 4000 3f06 a790 0a30 1659 0a30

0x0020: 1857 cbfe 1f90 bd8b 36fe 0000 0000 a002

0x0030: 6d38 e7b2 0000 0204 0582 0402 080a 00db

0x0040: 67e2 00db 6686 0103 0307

```
1 //on the pod sshd-5-qgjdn
2
3 sh-4.2# arp -an
4 ? (10.48.21.43) at 0a:58:0a:30:15:2b [ether] on eth0
5 ? (10.48.24.87) at 0a:58:0a:30:18:57 [ether] on eth0          |---- remote pod mac
6 ? (10.48.22.1) at 6e:93:62:95:75:84 [ether] on eth0           |---- tun0 mac
7 ? (10.48.22.100) at 0a:58:0a:30:16:64 [ether] on eth0
```



Packet flow: pod->service->pods

// on node22.lab.local

tcpdump -nnnv -xx -e -i tun0 'port 52222'

tcpdump: listening on tun0, link-type EN10MB (Ethernet), capture size 262144 bytes

08:58:30.852663 **0a:58:0a:30:16:59 > 6e:93:62:95:75:84**, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 64282, offset 0, flags [DF], proto TCP (6), length 60)

10.48.22.89.52222 > 10.49.145.117.80: Flags [S], cksum 0x832a (correct), seq 3507650740, win 28200, options [mss 1410,sackOK,TS val 15498158 ecr 0,nop,wscale 7], length 0

0x0000: 6e93 6295 7584 0a58 0a30 1659 0800 4500

0x0010: 003c fb1a 4000 4006 8372 0a30 1659 0a31

0x0020: 9175 cbfe 0050 d112 80b4 0000 0000 a002

0x0030: 6e28 832a 0000 0204 0582 0402 080a 00ec

0x0040: 7bae 0000 0000 0103 0307

08:39:51.672260 **0a:58:0a:30:18:57 > 0a:58:0a:30:16:59**, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)

10.48.24.87.8080 > 10.48.22.89.52222: Flags [S.], cksum 0x433e (incorrect -> 0xf40f), seq 1638110431, ack 3180017407, win 27960, options [mss 1410,sackOK,TS val 14377877 ecr 14378978,nop,wscale 7], length 0

0x0000: 0a58 0a30 1659 0a58 0a30 1857 0800 4500

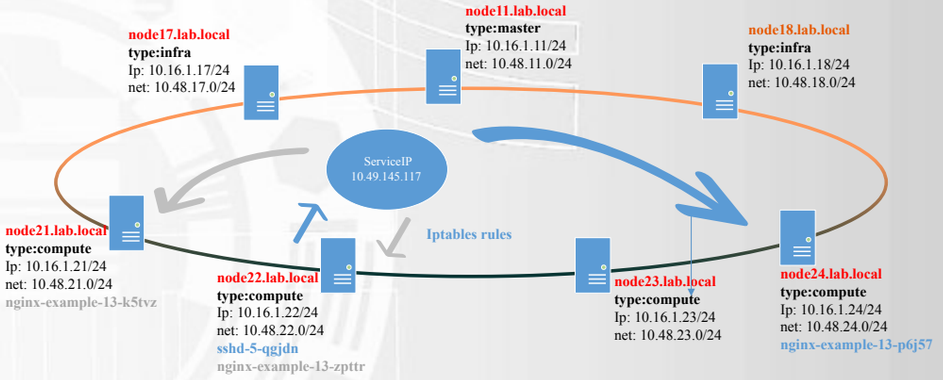
0x0010: 003c 0000 4000 4006 f7ac 0a30 1857 0a30

0x0020: 1659 1f90 cbfe 61a3 94df bd8b 36ff a012

0x0030: 6d38 433e 0000 0204 0582 0402 080a 00db

0x0040: 6395 00db 67e2 0103 0307

```
1 //on the pod sshd-5-qgjdn
2
3 sh-4.2# arp -an
4 ? (10.48.21.43) at 0a:58:0a:30:15:2b [ether] on eth0
5 ? (10.48.24.87) at 0a:58:0a:30:18:57 [ether] on eth0          |---- remote pod mac
6 ? (10.48.22.1) at 6e:93:62:95:75:84 [ether] on eth0           |---- tun0 mac
7 ? (10.48.22.100) at 0a:58:0a:30:16:64 [ether] on eth0
```



Packet flow: pod->service->pods

// on node22.lab.local

tcpdump -nnnv -xx -e -i tun0 'port 52222'

tcpdump: listening on tun0, link-type EN10MB (Ethernet), capture size 262144 bytes

08:58:30.852663 **0a:58:0a:30:16:59 > 6e:93:62:95:75:84**, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, id 64282, offset 0, flags [DF], proto TCP (6), length 60)

10.48.22.89.52222 > 10.49.145.117.80: Flags [S], cksum 0x832a (correct), seq 3507650740, win 28200, options [mss 1410,sackOK,TS val 15498158 ecr 0,nop,wscale 7], length 0

0x0000: 6e93 6295 7584 0a58 0a30 1659 0800 4500

0x0010: 003c fb1a 4000 4006 8372 0a30 1659 0a31

0x0020: 9175 cbfe 0050 d112 80b4 0000 0000 a002

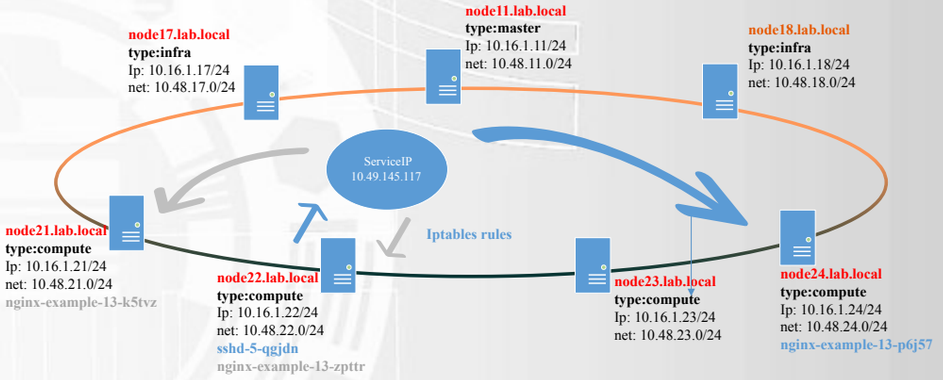
0x0030: 6e28 832a 0000 0204 0582 0402 080a 00ec

0x0040: 7bae 0000 0000 0103 0307

// Wireshark - eth0 node24.lab.local

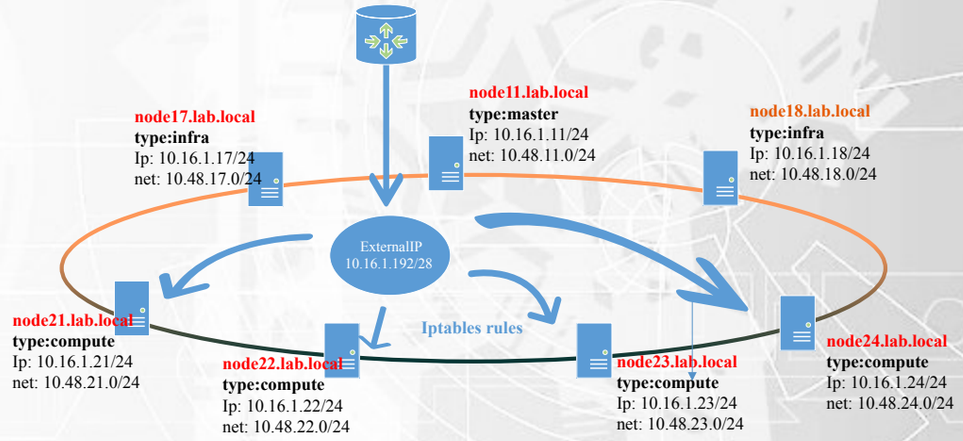
Frame 39: 124 bytes on wire (992 bits), 124 bytes captured (992 bits)
Ethernet II, Src: RealtekU_5e:07:fd (52:54:00:5e:07:fd), Dst: RealtekU_e3:e9:fd (52:54:00:e3:e9:fd)
Internet Protocol Version 4, Src: 10.16.1.22, Dst: 10.16.1.24
User Datagram Protocol, Src Port: 33946, Dst Port: 4789
Virtual eXtensible Local Area Network
  Flags: 0x0800, VXLAN Network ID (VNI)
  Group Policy ID: 0
  VXLAN Network Identifier (VNI): 8078500
  Reserved: 0
Ethernet II, Src: 6e:93:62:95:75:84 (6e:93:62:95:75:84), Dst: 0a:58:0a:30:18:57 (0a:58:0a:30:18:57)
Internet Protocol Version 4, Src: 10.48.22.89, Dst: 10.48.24.87
Transmission Control Protocol, Src Port: 52222, Dst Port: 8080, Seq: 0, Len: 0

```
1 //on the pod sshd-5-qgjdn
2
3 sh-4.2# arp -an
4 ? (10.48.21.43) at 0a:58:0a:30:15:2b [ether] on eth0
5 ? (10.48.24.87) at 0a:58:0a:30:18:57 [ether] on eth0          |---- remote pod mac
6 ? (10.48.22.1) at 6e:93:62:95:75:84 [ether] on eth0           |---- tun0 mac
7 ? (10.48.22.100) at 0a:58:0a:30:16:64 [ether] on eth0
```



External IP and IP Failover

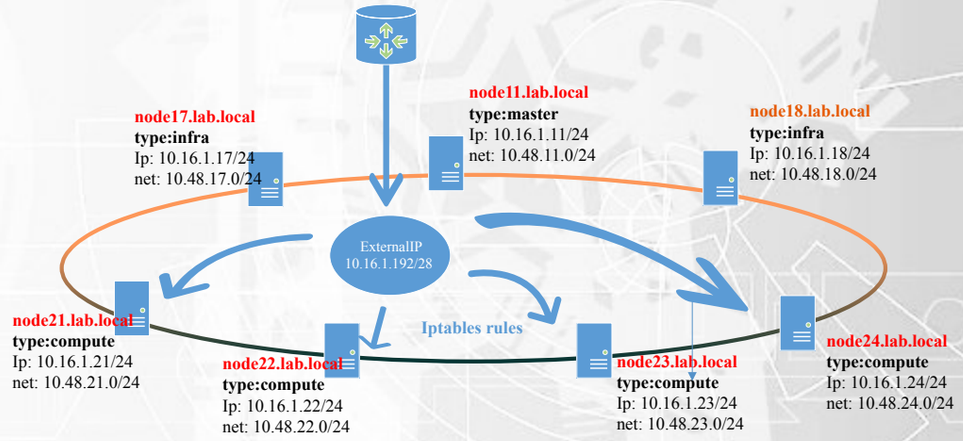
- services can have external ip for access from outter world
- clusterIP type is supported as well
- any protocol can be passed to the pods/containers
- administrator have to configure external ip
- best solution: IP Failover
- openshift manage service external IPs by iptables rules




```
1 ]# oc get svc
2 NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
3 nginx-example   ClusterIP   10.49.145.117   <none>        8080/TCP   2d
```

External IP and IP Failover

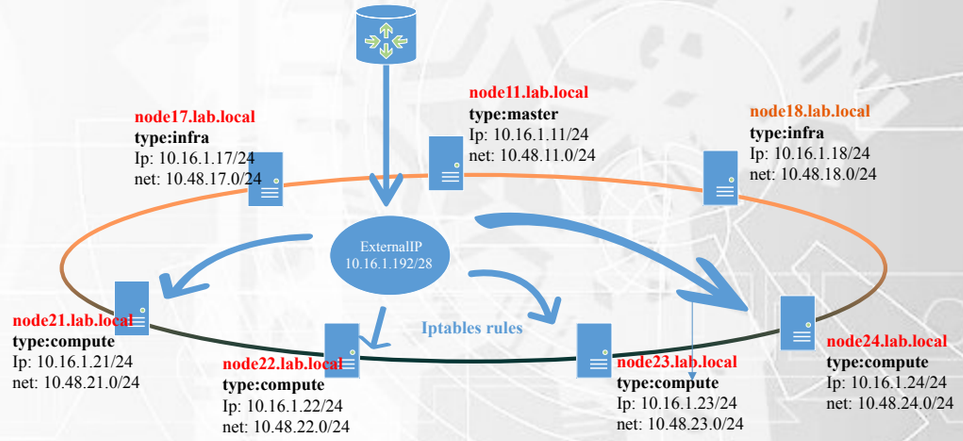
- services can have external ip for access from outter world
- clusterIP type is supported as well
- any protocol can be passed to the pods/containers
- administrator have to configure external ip
- best solution: IP Failover
- openshift manage service external IPs by iptables rules




```
1 ]# oc patch svc nginx-example -p '{"spec":{"externalIPs":["10.16.1.193"]}}'
2 service/nginx-example patched
```

External IP and IP Failover

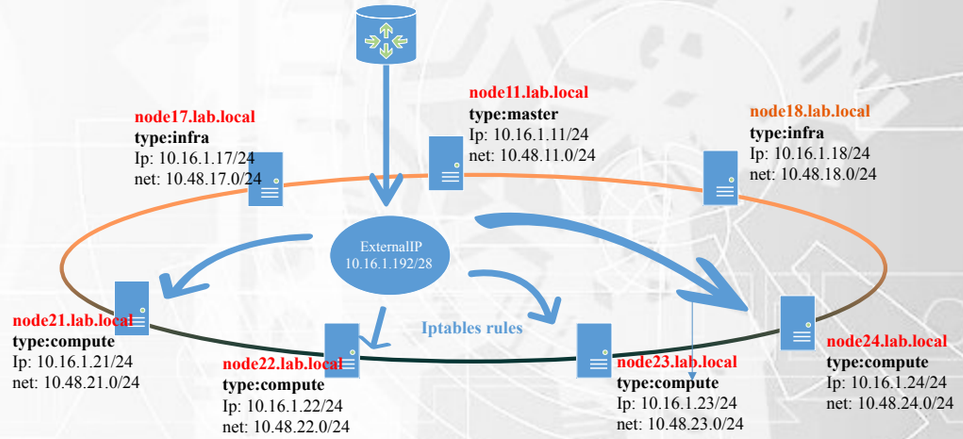
- services can have external ip for access from outter world
- clusterIP type is supported as well
- any protocol can be passed to the pods/containers
- administrator have to configure external ip
- best solution: IP Failover
- openshift manage service external IPs by iptables rules




```
1 # oc get svc
2 NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
3 nginx-example   ClusterIP   10.49.145.117   10.16.1.193   8080/TCP   2d
```

External IP and IP Failover

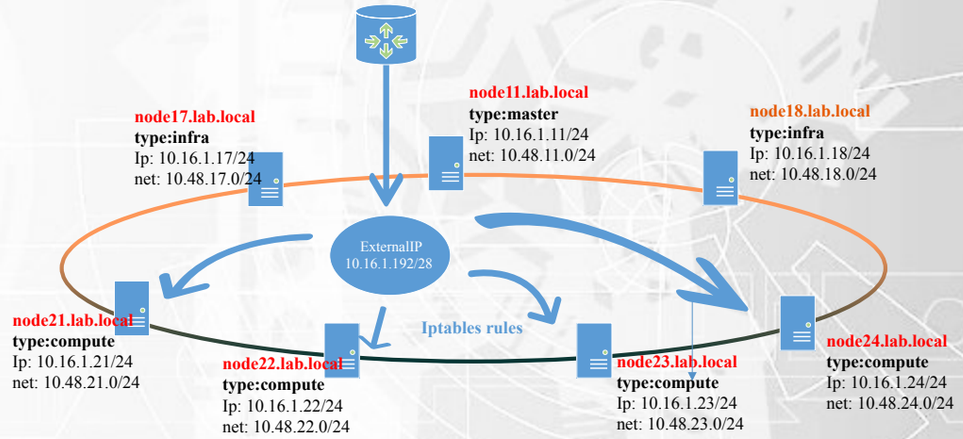
- services can have external ip for access from outter world
- clusterIP type is supported as well
- any protocol can be passed to the pods/containers
- administrator have to configure external ip
- best solution: IP Failover
- openshift manage service external IPs by iptables rules




```
# oc adm ipfailover ipfailover --watch-port=0 --virtual-ips=10.16.1.193 --create --replicas=2
--> Creating IP failover ipfailover ...
    serviceaccount "ipfailover" created
    deploymentconfig.apps.openshift.io "ipfailover" created
--> Success
```

External IP and IP Failover

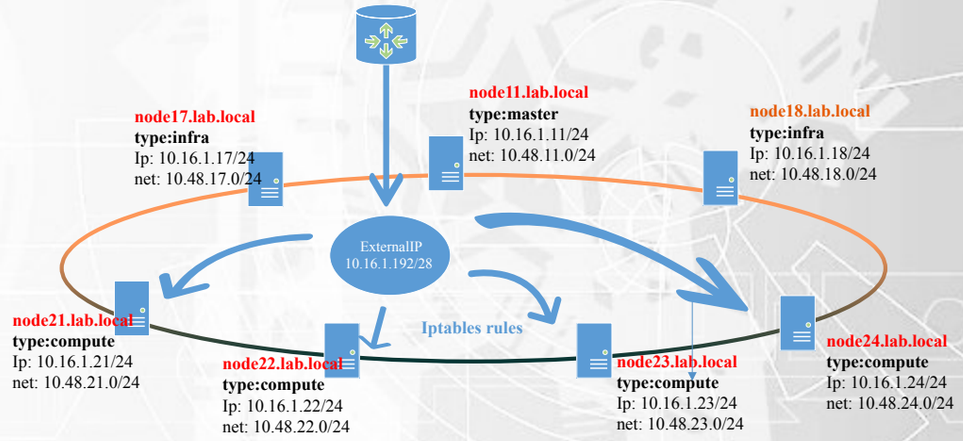
- services can have external ip for access from outter world
- clusterIP type is supported as well
- any protocol can be passed to the pods/containers
- administrator have to configure external ip
- best solution: IP Failover
- openshift manage service external IPs by iptables rules




```
# oc get pods -o wide
NAME                      READY     STATUS    RESTARTS   AGE       IP            NODE                NOMINATED NODE
ipfailover-1-sj22w        1/1       Running   0          11m       10.16.1.21    node21.lab.local    <none>
ipfailover-1-wcvhx        1/1       Running   0          11m       10.16.1.22    node22.lab.local    <none>
nginx-example-1-47mwz     1/1       Running   3          2d        10.48.24.26   node24.lab.local    <none>
nginx-example-1-q56b5     1/1       Running   3          2d        10.48.21.32   node21.lab.local    <none>
nginx-example-1-t27hc     1/1       Running   3          2d        10.48.23.24   node23.lab.local    <none>
```

External IP and IP Failover

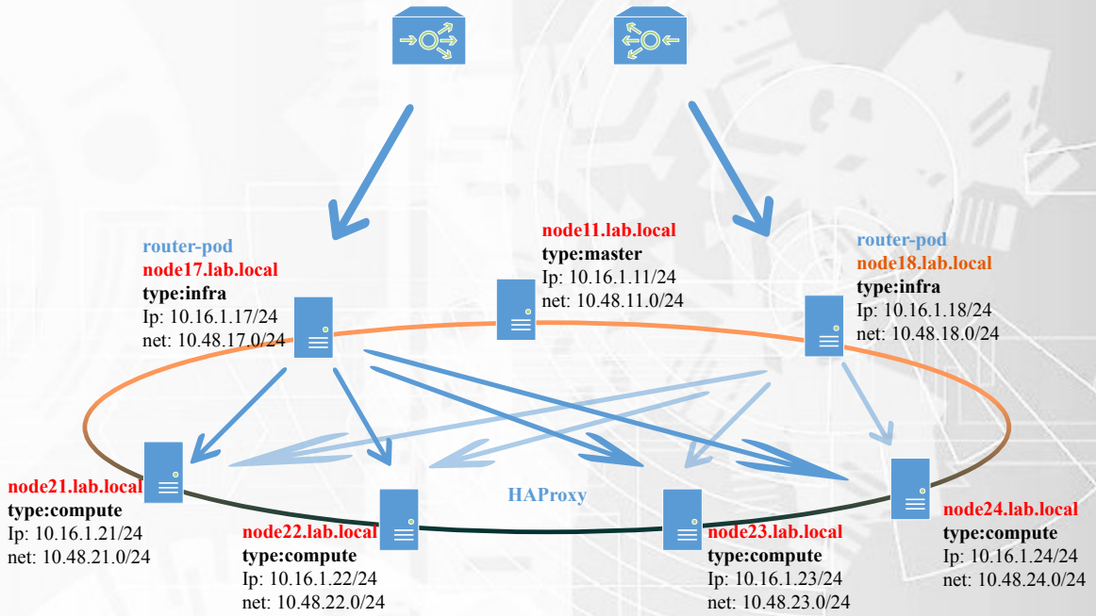
- services can have external ip for access from outter world
- clusterIP type is supported as well
- any protocol can be passed to the pods/containers
- administrator have to configure external ip
- best solution: IP Failover
- openshift manage service external IPs by iptables rules




```
// On node22

# ip a s eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:5e:07:fd brd ff:ff:ff:ff:ff:ff
    inet 10.16.1.22/24 brd 10.16.1.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
    inet 10.16.1.193/32 scope global eth0
       valid_lft forever preferred_lft forever
```

Routes

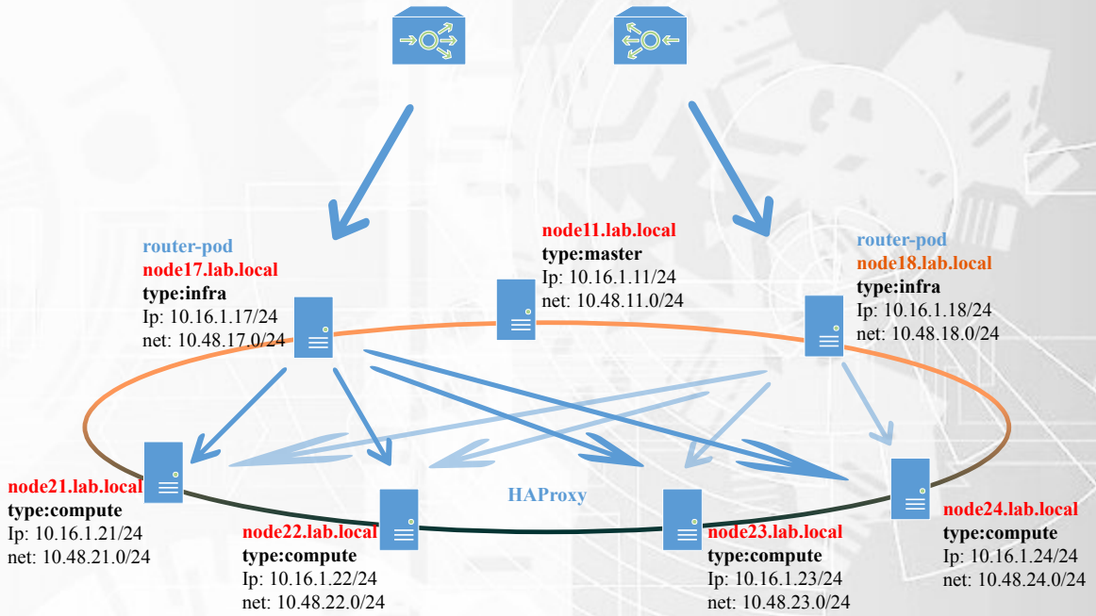
- expose services to outer world
- managed by HAproxy on infra nodes
- run in default namespace like a router pods
- ssl termination, passthrough etc.
- no additional ovs or iptables rules
- custom openshift object




```
# oc get pods -o wide
NAME                       READY     STATUS    RESTARTS   AGE       IP             NODE
docker-registry-1-8lmrt    1/1       Running   0          3m        10.48.18.2     node18.lab.local
registry-console-1-9wh5j   1/1       Running   0          3m        10.48.11.122   node11.lab.local
router-1-bkrgm             1/1       Running   2          22h       10.16.1.18     node18.lab.local
router-1-lbdhg             1/1       Running   2          22h       10.16.1.17     node17.lab.local
```

Routes

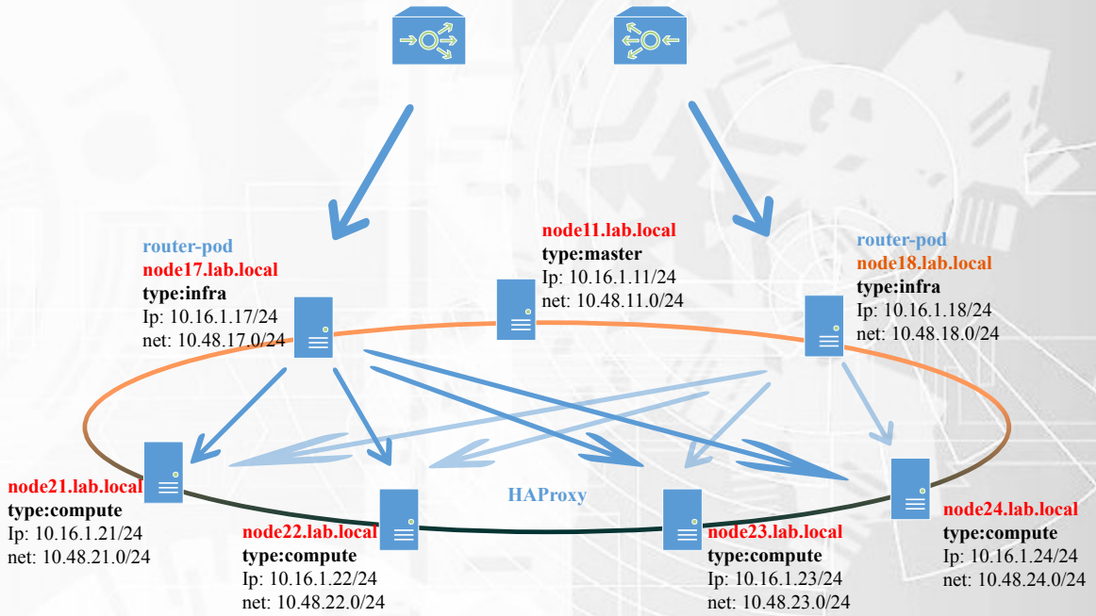
- expose services to outer world
- managed by HAproxy on infra nodes
- run in default namespace like a router pods
- ssl termination, passthrough etc.
- no additional ovs or iptables rules
- custom openshift object




```
# oc get svc
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
ng-ingress-http   ClusterIP   10.49.116.142   10.16.1.193   443/TCP   22h
nginx-example     ClusterIP   10.49.145.117   <none>        80/TCP    4d

# oc expose svc nginx-example
route.route.openshift.io/nginx-example exposed
# oc get route
NAME            HOST/PORT                          PATH      SERVICES        PORT      TERMINATION   WILDCARD
nginx-example   nginx-example-webapp.route.local             nginx-example   8080                    None
```

Routes

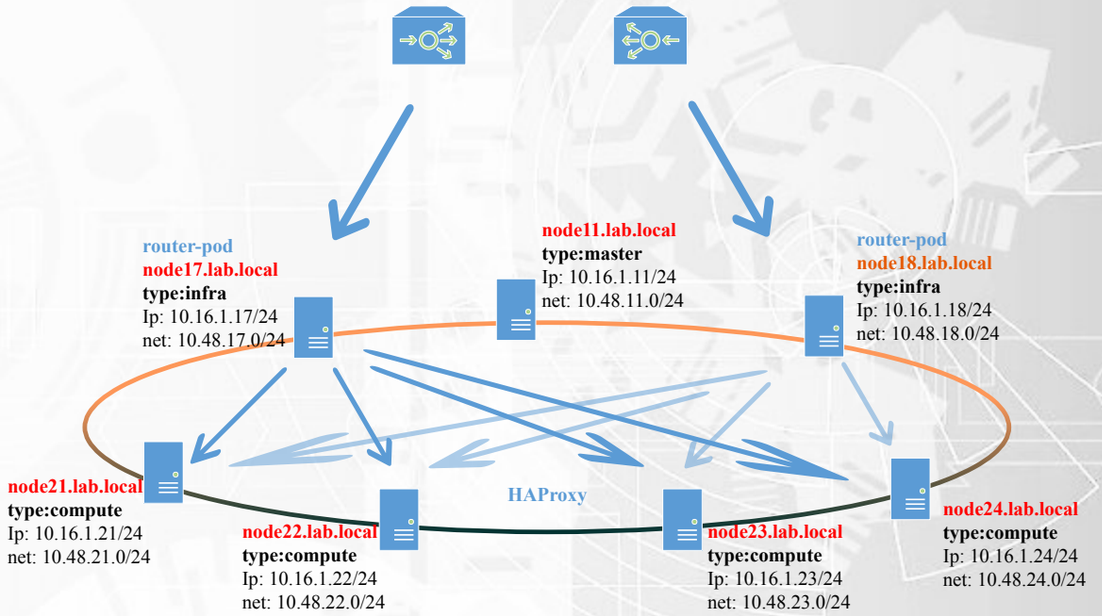
- expose services to outer world
- managed by HAproxy on infra nodes
- run in default namespace like a router pods
- ssl termination, passthrough etc.
- no additional ovs or iptables rules
- custom openshift object




```
# curl http://nginx-example-webapp.route.local -k -v | more
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0* About to connect() to nginx-example-
webapp.route.local port 80 (#0)
*   Trying 10.16.1.17...
* Connected to nginx-example-webapp.route.local (10.16.1.17) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: nginx-example-webapp.route.local
```

Routes

- expose services to outer world
- managed by HAproxy on infra nodes
- run in default namespace like a router pods
- ssl termination, passthrough etc.
- no additional ovs or iptables rules
- custom openshift object




```
1 // haproxy conf file - trimed
2
3 backend be_http:webapp:nginx-example
4   mode http
5  .
6  .
7 server pod:nginx-example-13-k5tvz:nginx-example:10.48.21.44:8080 10.48.21.44:8080
8 server pod:nginx-example-13-zpttr:nginx-example:10.48.22.101:8080 10.48.22.101:8080
9 server pod:nginx-example-13-p6j57:nginx-example:10.48.24.89:8080 10.48.24.89:8080
```

Ingress controllers

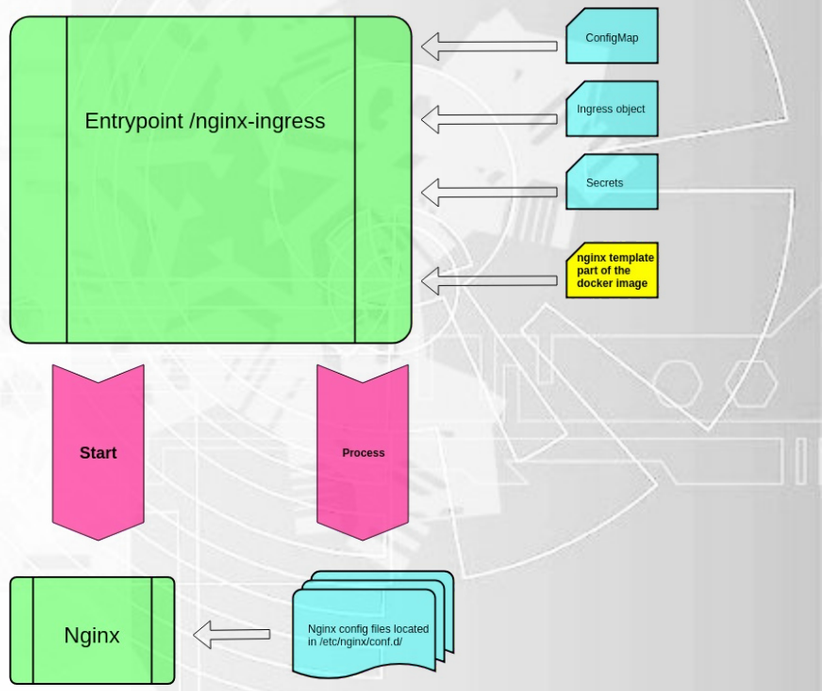
- default kubernetes method for managing of the incoming traffic
- usually controller + reverse proxy (haproxy, nginx)
- many different projects:
  - https://github.com/nginxinc/kubernetes-ingress
  - https://github.com/jcmoraisjr/haproxy-ingress
  - https://github.com/haproxytech/kubernetes-ingress
- usually the same functionality like a route
- deployed on the compute node
- better configuration granularity than route
- support for the forwarding any tcp traffic (low benefit)

```
NAME                      READY     STATUS    RESTARTS   AGE       IP             NODE
ipfailover-1-sj22w        1/1       Running   1          1d        10.16.1.21     node21.lab.local
ipfailover-1-wcvhx        1/1       Running   1          1d        10.16.1.22     node22.lab.local
ng-ingress-http-3-tfsv7   1/1       Running   0          14m       10.48.24.67    node24.lab.local
ng-ingress-http-3-zzpnj   1/1       Running   0          4s        10.48.22.87    node22.lab.local
nginx-example-1-47mwz     1/1       Running   4          3d        10.48.24.28    node24.lab.local
nginx-example-1-q56b5     1/1       Running   4          3d        10.48.21.33    node21.lab.local
nginx-example-1-t27hc     1/1       Running   4          3d        10.48.23.25    node23.lab.local

NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
ng-ingress-http   ClusterIP   10.49.116.142   10.16.1.193   443/TCP   14m
nginx-example     ClusterIP   10.49.145.117   <none>        80/TCP    3d
```

Ingress controllers

- default kubernetes method for managing of the incoming traffic
- usually controller + reverse proxy (haproxy, nginx)
- many different projects:
  - https://github.com/nginxinc/kubernetes-ingress
  - https://github.com/jcmoraisjr/haproxy-ingress
  - https://github.com/haproxytech/kubernetes-ingress
- usually the same functionality like a route
- deployed on the compute node
- better configuration granularity than route
- support for the forwarding any tcp traffic (low benefit)

```
NAME                      READY     STATUS    RESTARTS   AGE       IP             NODE
ipfailover-1-sj22w        1/1       Running   1          1d        10.16.1.21     node21.lab.local
ipfailover-1-wcvhx        1/1       Running   1          1d        10.16.1.22     node22.lab.local
ng-ingress-http-3-tfsv7   1/1       Running   0          14m       10.48.24.67    node24.lab.local
ng-ingress-http-3-zzpnj   1/1       Running   0          4s        10.48.22.87    node22.lab.local
nginx-example-1-47mwz     1/1       Running   4          3d        10.48.24.28    node24.lab.local
nginx-example-1-q56b5     1/1       Running   4          3d        10.48.21.33    node21.lab.local
nginx-example-1-t27hc     1/1       Running   4          3d        10.48.23.25    node23.lab.local

NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
ng-ingress-http   ClusterIP   10.49.116.142   10.16.1.193   443/TCP   14m
nginx-example     ClusterIP   10.49.145.117   <none>        80/TCP    3d
```



Ingress controllers

- default kubernetes method for managing of the incoming traffic
- usually controller + reverse proxy (haproxy, nginx)
- many different projects:
  - https://github.com/nginxinc/kubernetes-ingress
  - https://github.com/jcmoraisjr/haproxy-ingress
  - https://github.com/haproxytech/kubernetes-ingress
- usually the same functionality like a route
- deployed on the compute node
- better configuration granularity than route
- support for the forwarding any tcp traffic (low benefit)

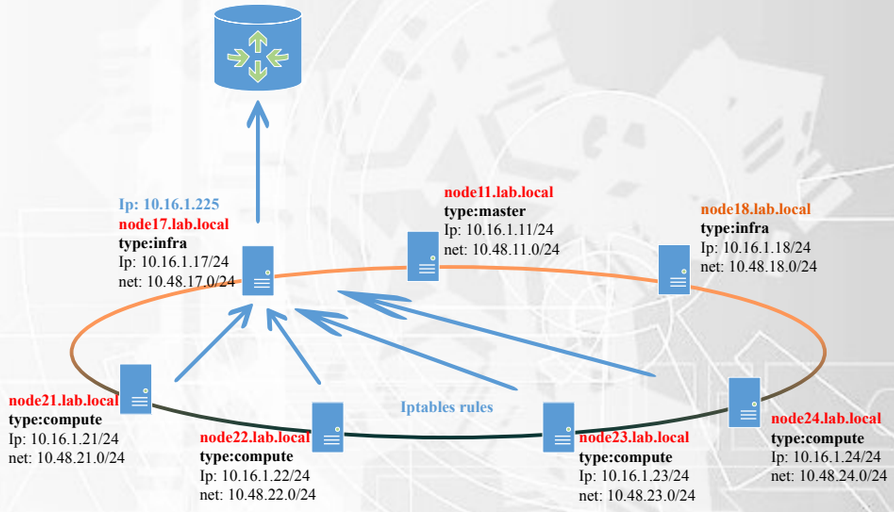
```
NAME                      READY     STATUS    RESTARTS   AGE       IP             NODE
ipfailover-1-sj22w        1/1       Running   1          1d        10.16.1.21     node21.lab.local
ipfailover-1-wcvhx        1/1       Running   1          1d        10.16.1.22     node22.lab.local
ng-ingress-http-3-tfsv7   1/1       Running   0          14m       10.48.24.67    node24.lab.local
ng-ingress-http-3-zzpnj   1/1       Running   0          4s        10.48.22.87    node22.lab.local
nginx-example-1-47mwz     1/1       Running   4          3d        10.48.24.28    node24.lab.local
nginx-example-1-q56b5     1/1       Running   4          3d        10.48.21.33    node21.lab.local
nginx-example-1-t27hc     1/1       Running   4          3d        10.48.23.25    node23.lab.local

NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
ng-ingress-http   ClusterIP   10.49.116.142   10.16.1.193   443/TCP   14m
nginx-example     ClusterIP   10.49.145.117   <none>        80/TCP    3d
```

```
apiVersion: v1
items:
- apiVersion: extensions/v1beta1
  kind: Ingress
  metadata:
    annotations:
      kubernetes.io/ingress.class: ng-ingress-http-webapp-1
    name: ng-ingress-http
    namespace: webapp
  spec:
    rules:
    - host: webapp.lab.local
      http:
        paths:
        - backend:
            serviceName: nginx-example
            servicePort: 80
          path: /
    tls:
    - hosts:
      - webapp.lab.local
      secretName: ng-ingress-default
```

Egress traffic

- without egress cfg openshift SNAT on the last node on path
- egress can be managed:
  - on the namespace level
  - or via special egress controller
- egressnetworkpolicy object on the namespace layer can restrict access outside cluster

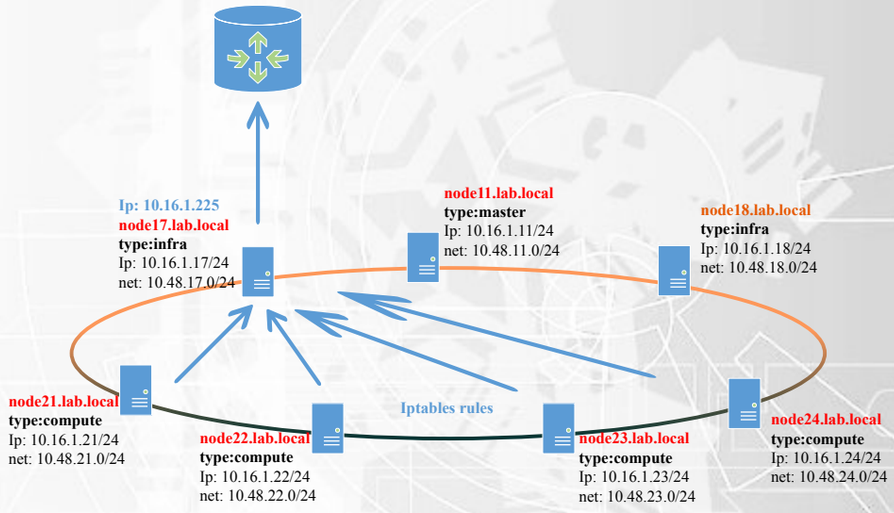
```
# oc rsh nginx-example-1-q56b5
sh-4.2$ curl http://192.168.1.228:8080 -v
* About to connect() to 192.168.1.228 port 8080 (#0)
*   Trying 192.168.1.228...
* Connected to 192.168.1.228 (192.168.1.228) port 8080 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: 192.168.1.228:8080
> Accept: */*
>
```



Egress traffic

- without egress cfg openshift SNAT on the last node on path
- egress can be managed:
  - on the namespace level
  - or via special egress controller
- egressnetworkpolicy object on the namespace layer can restrict access outside cluster

```
# oc rsh nginx-example-1-q56b5
sh-4.2$ curl http://192.168.1.228:8080 -v
* About to connect() to 192.168.1.228 port 8080 (#0)
*   Trying 192.168.1.228...
* Connected to 192.168.1.228 (192.168.1.228) port 8080 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: 192.168.1.228:8080
> Accept: */*
>
```

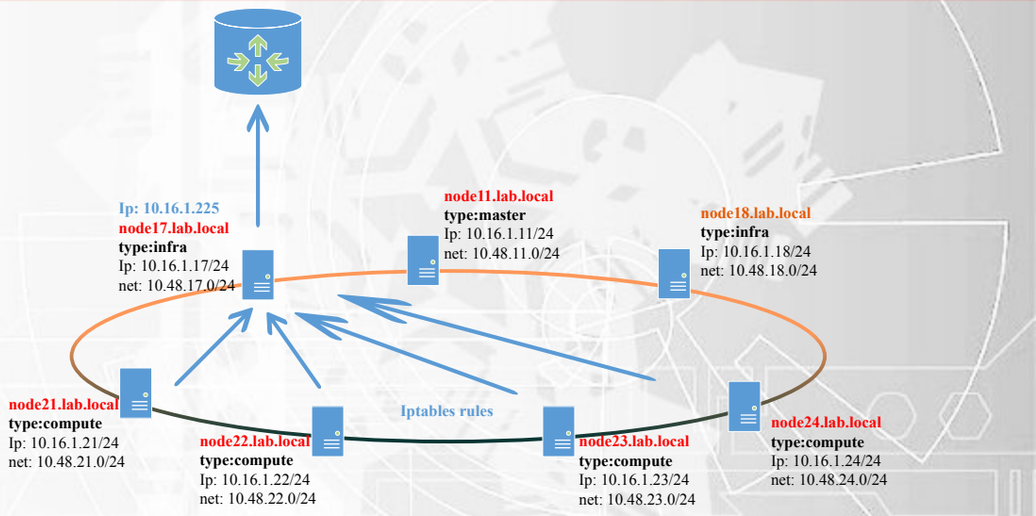



```
# oc get netnamespace
NAME                                NETID      EGRESS IPS
client                              8078500    []
default                             0          []
kube-public                         5916925    []
kube-service-catalog                2275370    []
kube-system                         12682412   []
management-infra                    9772787    []
openshift                           5950122    []
openshift-console                   15973489   []
openshift-infra                     9179195    []
openshift-logging                   8937677    []
openshift-node                      1401652    []
openshift-sdn                       14997392   []
openshift-template-service-broker   5029162    []
openshift-web-console               11599497   []
webapp                              4074785    [10.16.1.225]
```

Egress traffic

- without egress cfg openshift SNAT on the last node on path
- egress can be managed:
  - on the namespace level
  - or via special egress controller
- egressnetworkpolicy object on the namespace layer can restrict access outside cluster

```
# oc rsh nginx-example-1-q56b5
sh-4.2$ curl http://192.168.1.228:8080 -v
* About to connect() to 192.168.1.228 port 8080 (#0)
*   Trying 192.168.1.228...
* Connected to 192.168.1.228 (192.168.1.228) port 8080 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: 192.168.1.228:8080
> Accept: */*
>
```

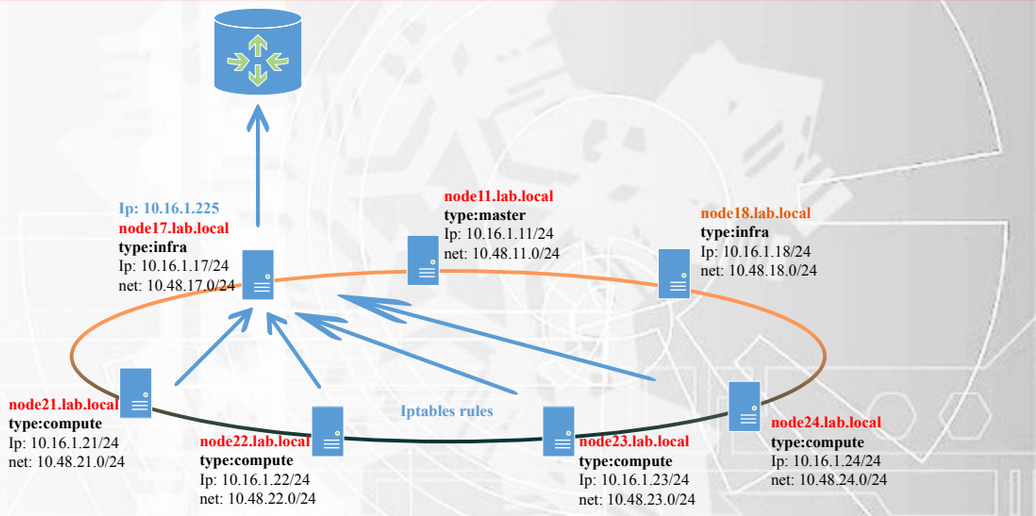



```
# oc get hostsubnet
NAME               HOST               HOST IP      SUBNET          EGRESS CIDRS          EGRESS IPS
node11.lab.local   node11.lab.local   10.16.1.11   10.48.11.0/24   []                    []
node17.lab.local   node17.lab.local   10.16.1.17   10.48.17.0/24   [10.16.1.224/27]      [10.16.1.225]
node18.lab.local   node18.lab.local   10.16.1.18   10.48.18.0/24   [10.16.1.224/27]      []
node21.lab.local   node21.lab.local   10.16.1.21   10.48.21.0/24   []                    []
node22.lab.local   node22.lab.local   10.16.1.22   10.48.22.0/24   []                    []
node23.lab.local   node23.lab.local   10.16.1.23   10.48.23.0/24   []                    []
node24.lab.local   node24.lab.local   10.16.1.24   10.48.24.0/24   []                    []
```

Egress traffic

- without egress cfg openshift SNAT on the last node on path
- egress can be managed:
  - on the namespace level
  - or via special egress controller
- egressnetworkpolicy object on the namespace layer can restrict access outside cluster

```
# oc rsh nginx-example-1-q56b5
sh-4.2$ curl http://192.168.1.228:8080 -v
* About to connect() to 192.168.1.228 port 8080 (#0)
*   Trying 192.168.1.228...
* Connected to 192.168.1.228 (192.168.1.228) port 8080 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: 192.168.1.228:8080
> Accept: */*
>
```

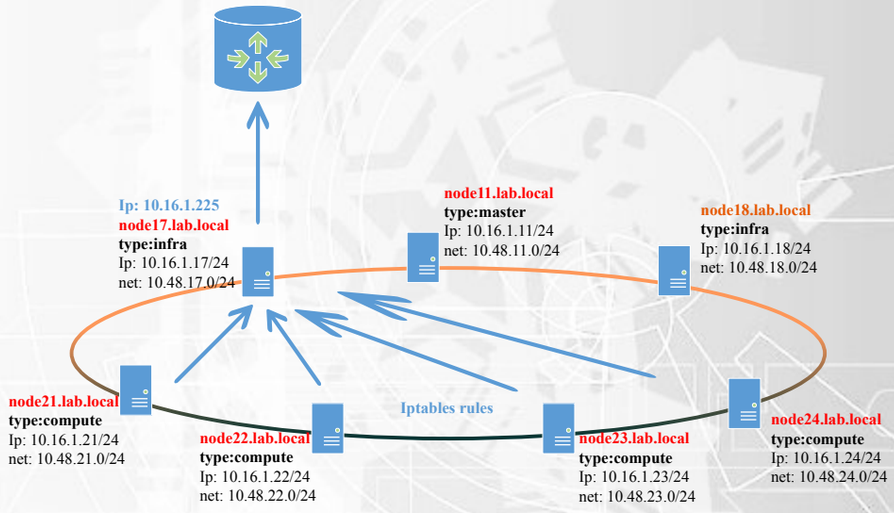



```
[root@node17 ~]# ip a s eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:ae:17:5c brd ff:ff:ff:ff:ff:ff
    inet 10.16.1.17/24 brd 10.16.1.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
    inet 10.16.1.225/24 brd 10.16.1.255 scope global secondary eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::f477:e6a:bee0:e931/64 scope link tentative noprefixroute dadfailed
       valid_lft forever preferred_lft forever
    inet6 fe80::b06c:37a6:c45b:8dc3/64 scope link tentative noprefixroute dadfailed
       valid_lft forever preferred_lft forever
    inet6 fe80::a451:2de2:2752:8af6/64 scope link tentative noprefixroute dadfailed
       valid_lft forever preferred_lft forever
```

Egress traffic

- without egress cfg openshift SNAT on the last node on path
- egress can be managed:
  - on the namespace level
  - or via special egress controller
- egressnetworkpolicy object on the namespace layer can restrict access outside cluster

```
# oc rsh nginx-example-1-q56b5
sh-4.2$ curl http://192.168.1.228:8080 -v
* About to connect() to 192.168.1.228 port 8080 (#0)
*   Trying 192.168.1.228...
* Connected to 192.168.1.228 (192.168.1.228) port 8080 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: 192.168.1.228:8080
> Accept: */*
>
```

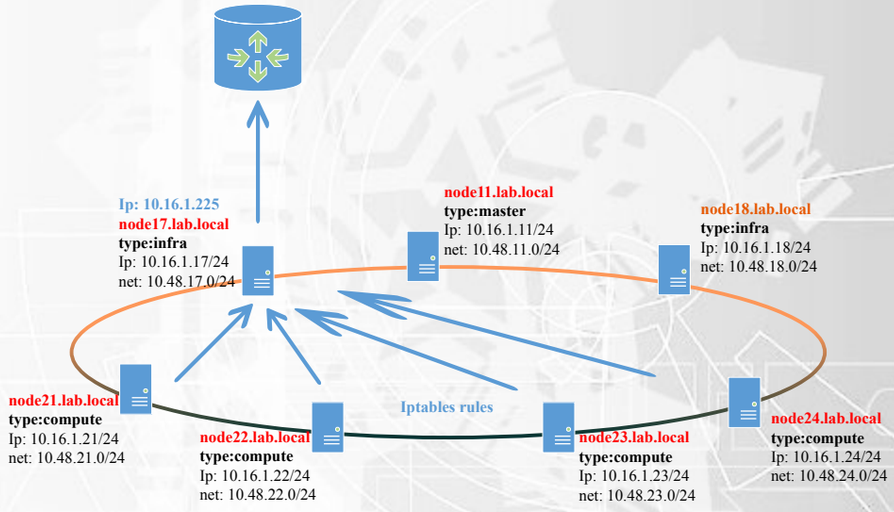



```
# ovs on node17.lab.local
cookie=0x0, table=100, priority=100,ip,reg0=0x3e2d21 actions=set_field:26:b9:8c:5e:a7:89->eth_dst,set_field:0x13e2d20->pkt_mark,goto_table:101
cookie=0x0, table=100, priority=0 actions=goto_table:101
cookie=0x0, table=101, priority=0 actions=output:2
```

Egress traffic

- without egress cfg openshift SNAT on the last node on path
- egress can be managed:
  - on the namespace level
  - or via special egress controller
- egressnetworkpolicy object on the namespace layer can restrict access outside cluster

```
# oc rsh nginx-example-1-q56b5
sh-4.2$ curl http://192.168.1.228:8080 -v
* About to connect() to 192.168.1.228 port 8080 (#0)
*   Trying 192.168.1.228...
* Connected to 192.168.1.228 (192.168.1.228) port 8080 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: 192.168.1.228:8080
> Accept: */*
>
```

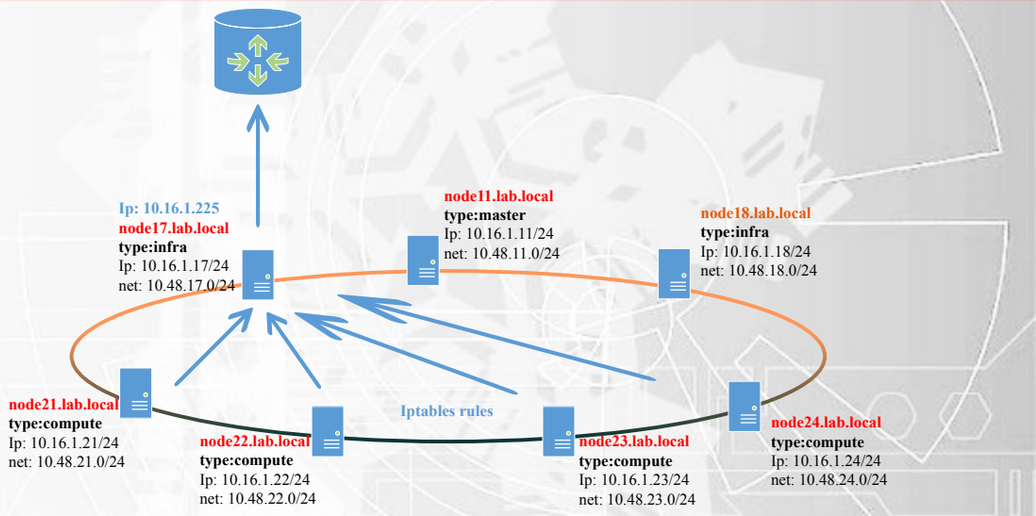



```
1  // IP tables on the node17.lab.local
2
3  -A OPENSHIFT-FIREWALL-ALLOW -d 10.16.1.225/32 -m conntrack --ctstate NEW -j REJECT
4  -A OPENSHIFT-MASQUERADE -s 10.48.0.0/16 -m mark --mark 0x13e2d20 -j SNAT --to-source 10.16.1.225
5
6
7
8
9
10
```

Egress traffic

- without egress cfg openshift SNAT on the last node on path
- egress can be managed:
  - on the namespace level
  - or via special egress controller
- egressnetworkpolicy object on the namespace layer can restrict access outside cluster

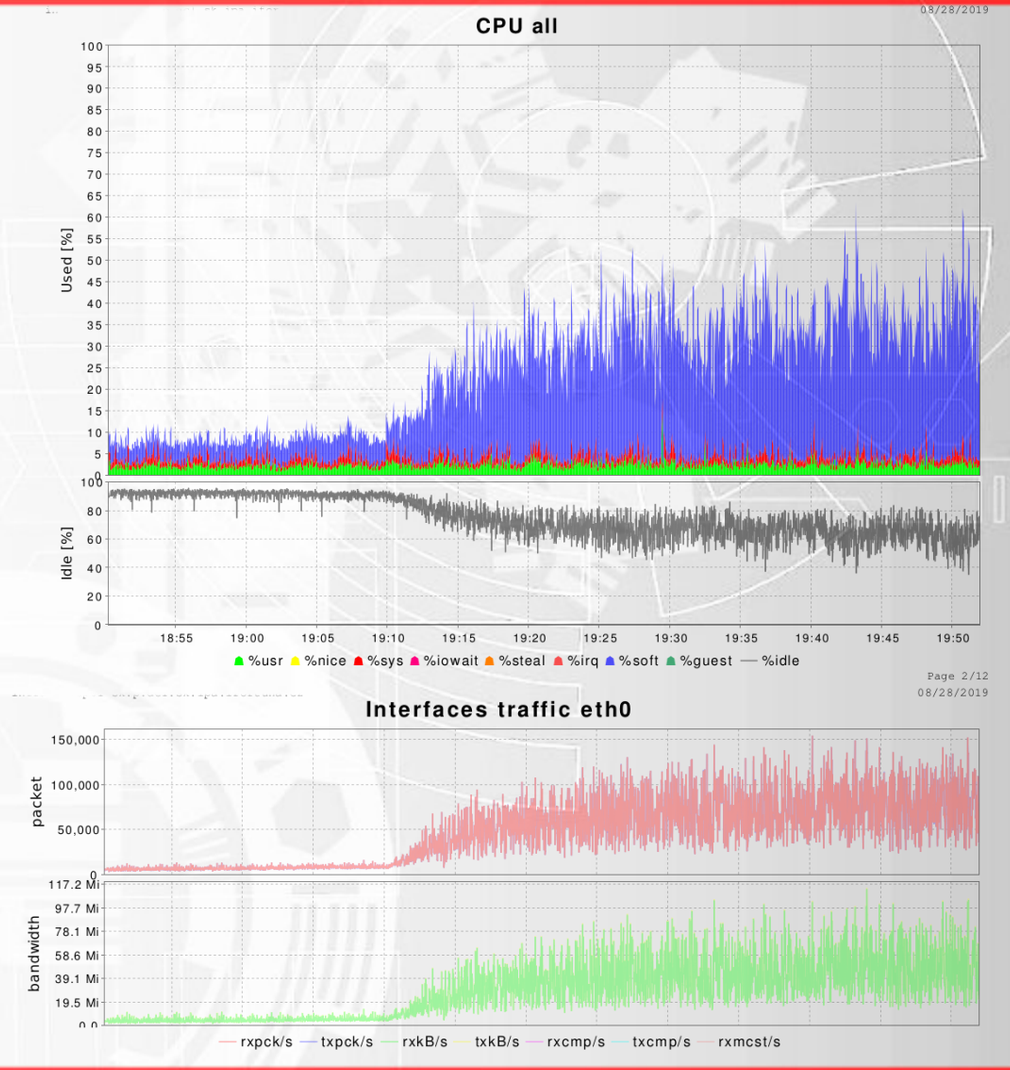
```
# oc rsh nginx-example-1-q56b5
sh-4.2$ curl http://192.168.1.228:8080 -v
* About to connect() to 192.168.1.228 port 8080 (#0)
*   Trying 192.168.1.228...
* Connected to 192.168.1.228 (192.168.1.228) port 8080 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: 192.168.1.228:8080
> Accept: */*
>
```




```
1  // On 192.168.1.228 - remote node
2
3  # nc -l -p 8080 -v
4  Ncat: Version 7.50 ( https://nmap.org/ncat )
5  Ncat: Listening on :::8080
6  Ncat: Listening on 0.0.0.0:8080
7  Ncat: Connection from 10.16.1.225.
8  Ncat: Connection from 10.16.1.225:42076.
9  GET / HTTP/1.1
10 User-Agent: curl/7.29.0
11 Host: 192.168.1.228:8080
12 Accept: */*
```

Openshift SDN Limitation

- network extension beyond the Openshift world
- services rely on the iptables => performance degradation under heavy load
- possible source port collision for traffic behind egress



Troubleshooting: Application administrator

- Usually limited access to the objects in the one or more namespaces
- Basic methodology: side containers
  - some tools need special capability
- Valuable tools: netcat, curl, openssl-tools, bind-tools

```
# oc get pods
NAME                       READY     STATUS    RESTARTS   AGE
ipfailover-1-sj22w         1/1       Running   2          1d
ipfailover-1-wcvhx         1/1       Running   2          1d
ng-ingress-http-3-tfsv7    1/1       Running   1          14h
ng-ingress-http-3-zzpnj    1/1       Running   1          13h
nginx-example-13-k5tvz     2/2       Running   0          10m
nginx-example-13-p6j57     2/2       Running   0          10m
nginx-example-13-zpttr     2/2       Running   0          11m
[root@dl380 ~]# oc rsh -c debugtools nginx-example-13-k5tvz
sh-4.2$
```

```
// Part of the dc nginx-examples
 template:
  metadata:
   creationTimestamp: null
   labels:
    name: nginx-example
   name: nginx-example
  spec:
   containers:
   - image: docker-registry.default.svc:5000/openshift/centos-debug:2.5
     imagePullPolicy: IfNotPresent
     name: debugtools
     resources:
       limits:
         memory: 512Mi
     terminationMessagePath: /dev/termination-log
     terminationMessagePolicy: File
   - image: docker-registry.default.svc:5000/webapp/nginx-example:latest
     imagePullPolicy: IfNotPresent
       initialDelaySeconds: 30
       periodSeconds: 10
       successThreshold: 1
       timeoutSeconds: 3
     name: nginx-example
```

Troubleshooting: System Administrator

- usually access to the all entities and nodes
- basic command: netns -n -t <pid>
- filtering of the namespace traffic based on the vnid (wireshark)
- change kube-proxy verbosity level: /etc/sysconfig/origin-node
- for flow tracing choose the source port
- iptables custom log rules, conntrack
- performance stats (Prometheus, Zabbix etc)
  - metrics:
    - "si" metric (load of the iptables)
    - packets/s
    - load, general cpu stats
    - interrupts
- get a node logs and make own kibana dashboard

```
// sdn-2kwp4 sdn controller on node21.lab.local

# oc logs -f sdn-2kwp4 -n openshift-sdn

I0922 18:09:04.409486   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 --may-exist add-port br0 vethd97fdaf0 -- set Interface vethd97fdaf0 external-ids=sandbox="d0163e78f837620aa5834fc3a05e785838117e54ada21d74038fe3e0956c3112",ip="10.48.21.34"
I0922 18:09:04.422241   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 get Interface vethd97fdaf0 ofport
I0922 18:09:04.430652   12085 ovs.go:159] Executing: ovs-ofctl -O OpenFlow13 bundle br0 - <<
flow add table=20, priority=100, in_port=4, arp, nw_src=10.48.21.34, arp_sha=00:00:0a:30:15:22/00:00:ff:ff:ff:ff, actions=load:8078500->NXM_NX_REG0[], goto_table:21
flow add table=20, priority=100, in_port=4, ip, nw_src=10.48.21.34, actions=load:8078500->NXM_NX_REG0[], goto_table:21
flow add table=25, priority=100, ip, nw_src=10.48.21.34, actions=load:8078500->NXM_NX_REG0[], goto_table:30
flow add table=40, priority=100, arp, nw_dst=10.48.21.34, actions=output:4
flow add table=70, priority=100, ip, nw_dst=10.48.21.34, actions=load:8078500->NXM_NX_REG1[], load:4->NXM_NX_REG2[], goto_table:80
I0922 18:09:04.440251   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 --no-heading --columns=name find interface external-ids:sandbox=d0163e78f837620aa5834fc3a05e785838117e54ada21d74038fe3e0956c3112
I0922 18:09:04.448908   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 --if-exists clear port vethd97fdaf0 qos
I0922 18:09:04.455972   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 --no-heading --columns=_uuid find qos external-ids:sandbox=d0163e78f837620aa5834fc3a05e785838117e54ada21d74038fe3e0956c3112
I0922 18:09:04.464029   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 --if-exists destroy qos
```

Troubleshooting: System Administrator

- usually access to the all entities and nodes
- basic command: netns -n -t <pid>
- filtering of the namespace traffic based on the vnid (wireshark)
- change kube-proxy verbosity level: /etc/sysconfig/origin-node
- for flow tracing choose the source port
- iptables custom log rules, conntrack
- performance stats (Prometheus, Zabbix etc)
  - metrics:
    - "si" metric (load of the iptables)
    - packets/s
    - load, general cpu stats
    - interrupts
- get a node logs and make own kibana dashboard

 # oc scale dc sshd --replicas=4
deploymentconfig.apps.openshift.io/sshd scaled

```
1  // sdn-2kwp4 sdn controller on node21.lab.local
2
3  # oc logs -f sdn-2kwp4 -n openshift-sdn
4
5  I0922 18:09:04.409486   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 --may-exist add-
   port br0 vethd97fdaf0 -- set Interface vethd97fdaf0 external-
   ids=sandbox="d0163e78f837620aa5834fc3a05e785838117e54ada21d74038fe3e0956c3112",ip="10.48.21.3
   4"
6  I0922 18:09:04.422241   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 get Interface
   vethd97fdaf0 ofport
7  I0922 18:09:04.430652   12085 ovs.go:159] Executing: ovs-ofctl -O OpenFlow13 bundle br0 - <<
8  flow add table=20, priority=100, in_port=4, arp, nw_src=10.48.21.34,
   arp_sha=00:00:0a:30:15:22/00:00:ff:ff:ff:ff, actions=load:8078500->NXM_NX_REG0[],
   goto_table:21
9  flow add table=20, priority=100, in_port=4, ip, nw_src=10.48.21.34, actions=load:8078500-
   >NXM_NX_REG0[], goto_table:21
10 flow add table=25, priority=100, ip, nw_src=10.48.21.34, actions=load:8078500->NXM_NX_REG0[],
   goto_table:30
11 flow add table=40, priority=100, arp, nw_dst=10.48.21.34, actions=output:4
12 flow add table=70, priority=100, ip, nw_dst=10.48.21.34, actions=load:8078500->NXM_NX_REG1[],
   load:4->NXM_NX_REG2[], goto_table:80
13 I0922 18:09:04.440251   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 --no-heading --
   columns=name find interface external-
   ids:sandbox=d0163e78f837620aa5834fc3a05e785838117e54ada21d74038fe3e0956c3112
14 I0922 18:09:04.448908   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 --if-exists clear
   port vethd97fdaf0 qos
15 I0922 18:09:04.455972   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 --no-heading --
   columns=_uuid find qos external-
   ids:sandbox=d0163e78f837620aa5834fc3a05e785838117e54ada21d74038fe3e0956c3112
16 I0922 18:09:04.464029   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 --if-exists
   destroy qos
```

Troubleshooting: System Administrator

- usually access to the all entities and nodes
- basic command: netns -n -t <pid>
- filtering of the namespace traffic based on the vnid (wireshark)
- change kube-proxy verbosity level: /etc/sysconfig/origin-node
- for flow tracing choose the source port
- iptables custom log rules, conntrack
- performance stats (Prometheus, Zabbix etc)
  - metrics:
    - "si" metric (load of the iptables)
    - packets/s
    - load, general cpu stats
    - interrupts
- get a node logs and make own kibana dashboard

 # oc expose dc sshd
service/sshd exposed

```
1  // sdn-2kwp4 sdn controller on node21.lab.local
2
3  # oc logs -f sdn-2kwp4 -n openshift-sdn
4
5  I0922 18:13:49.508311   12085 proxy.go:331] hybrid proxy: syncProxyRules start
6  I0922 18:13:49.508419   12085 proxier.go:657] Syncing iptables rules
7  I0922 18:13:49.542291   12085 iptables.go:323] running iptables-save [-t filter]
8  I0922 18:13:49.548216   12085 iptables.go:323] running iptables-save [-t nat]
9  I0922 18:13:49.558073   12085 iptables.go:383] running iptables-restore [-w 5 --noflush --
   counters]
10 I0922 18:13:49.572414   12085 healthcheck.go:235] Not saving endpoints for unknown
   healthcheck "webapp/nginx-example"
11 I0922 18:13:49.572457   12085 proxier.go:631] syncProxyRules took 64.091ms
12 I0922 18:13:49.600197   12085 proxier.go:369] userspace proxy: processing 0 service events
13 I0922 18:13:49.600233   12085 proxier.go:348] userspace syncProxyRules took 27.752555ms
14 I0922 18:13:49.600250   12085 proxy.go:336] hybrid proxy: syncProxyRules finished
15 I0922 18:13:49.600264   12085 bounded_frequency_runner.go:221] sync-runner: ran, next
   possible in 0s, periodic in 30s
16 I0922 18:13:50.133061   12085 ovs.go:161] Executing: ovs-vsctl --timeout=30 get Interface
   vxlan0 options:dst_port
17 I0922 18:13:50.143723   12085 healthcheck.go:98] SDN healthcheck succeeded
18 I0922 18:13:56.494655   12085 config.go:224] Calling handler.OnServiceAdd
19 I0922 18:13:56.494755   12085 proxy.go:331] hybrid proxy: syncProxyRules start
20 I0922 18:13:56.494789   12085 service.go:319] Adding new service port "client/sshd:" at
   10.49.219.251:10022/TCP
21 I0922 18:13:56.494834   12085 proxier.go:657] Syncing iptables rules
22 I0922 18:13:56.522156   12085 config.go:124] Calling handler.OnEndpointsAdd
23 I0922 18:13:56.522230   12085 roundrobin.go:276] LoadBalancerRR: Setting endpoints for
   client/sshd: to [10.48.21.34:10022 10.48.22.39:10022 10.48.23.27:10022 10.48.24.34:10022]
24 I0922 18:13:56.522303   12085 roundrobin.go:100] LoadBalancerRR service "client/sshd:" did
   not exist, created
25 I0922 18:13:56.522386   12085 endpoints.go:234] Setting endpoints for "client/sshd:" to
   [10.48.21.34:10022 10.48.22.39:10022 10.48.23.27:10022 10.48.24.34:10022]
```

Troubleshooting: OpenvSwitch

- check the flows
- log files: /var/log/openvswitch/ovs-vswitchd.log /var/log/openvswitch/ovsdb-server.log
- comments in the okd source code are your friend!
  - *pkg/network/node/ovscontroller.go*
- http://docs.openvswitch.org/en/latest/ref/
- diagnostic commands example:

```
# oc get pods -o wide | grep node21
ipfailover-1-sj22w      1/1       Running   1          1d        10.16.1.21    node21.lab.local
nginx-example-1-q56b5   1/1       Running   4          3d        10.48.21.33   node21.lab.local

# oc get netnamespace | grep -E 'webapp|client'
client                                  8078500    []
webapp                                  4074785    [10.16.1.225]
```

```
// ovs-b2dgt (ovc controller for node21.lab.local)

# oc rsh ovs-b2dgt (ovc controller for node21.lab.local)
sh-4.2# ovs-vsctl --format=table --columns=ofport,name,type,external_ids list interface
ofport name           type     external_ids
------ -------------- -------- ------------------------------------------------------------------------------------------
65534  "br0"          internal {}
1      "vxlan0"       vxlan    {}
2      "tun0"         internal {}
3      "veth935831c3" ""       {ip="10.48.21.33", sandbox="f7939b3884f12521c09b44f7bbeaab70d3ac16f00992392f193d2899ea9f7a6f"}
```

Troubleshooting: OpenvSwitch

- check the flows
- log files: /var/log/openvswitch/ovs-vswitchd.log /var/log/openvswitch/ovsdb-server.log
- comments in the okd source code are your friend!
  - *pkg/network/node/ovscontroller.go*
- http://docs.openvswitch.org/en/latest/ref/
- diagnostic commands example: *ovs-vsctl list-interface*

```
# oc get pods -o wide | grep node21
ipfailover-1-sj22w      1/1       Running   1          1d        10.16.1.21    node21.lab.local
nginx-example-1-q56b5   1/1       Running   4          3d        10.48.21.33   node21.lab.local

# oc get netnamespace | grep -E 'webapp|client'
client                                 8078500    []
webapp                                 4074785    [10.16.1.225]
```

```
// ovs-b2dgt (ovc controller for node21.lab.local)

# oc rsh ovs-b2dgt (ovc controller for node21.lab.local)
sh-4.2# ovs-vsctl --format=table --columns=ofport,name,type,external_ids list interface
ofport name           type     external_ids
------ -------------- -------- ------------------------------------------------------------------------------------------------
65534  "br0"          internal {}
1      "vxlan0"       vxlan    {}
2      "tun0"         internal {}
3      "veth935831c3" ""       {ip="10.48.21.33", sandbox="f7939b3884f12521c09b44f7bbeaab70d3ac16f00992392f193d2899ea9f7a6f"}
```

Troubleshooting: OpenvSwitch

- check the flows
- log files: /var/log/openvswitch/ovs-vswitchd.log /var/log/openvswitch/ovsdb-server.log
- comments in the okd source code are your friend!
  - *pkg/network/node/ovscontroller.go*
- http://docs.openvswitch.org/en/latest/ref/
- diagnostic commands example: *ovs-ofctl -O OpenFlow13 dump-flows br0*

```
# oc get pods -o wide | grep node21
ipfailover-1-sj22w      1/1       Running   1          1d        10.16.1.21    node21.lab.local
nginx-example-1-q56b5   1/1       Running   4          3d        10.48.21.33   node21.lab.local

# oc get netnamespace | grep -E 'webapp|client'
client                                  8078500    []
webapp                                  4074785    [10.16.1.225]
```

```
// ovs-b2dgt (ovc controller for node21.lab.local)

// Output was customized!

sh-4.2# ovs-ofctl -O OpenFlow13 dump-flows br0 | grep 'table=90'
table=90, priority=100,ip,nw_dst=10.48.11.0/24 actions=move:NXM_NX_REG0[]->NXM_NX_TUN_ID[0..31],set_field:10.16.1.11->tun_dst,output:1
table=90, priority=100,ip,nw_dst=10.48.17.0/24 actions=move:NXM_NX_REG0[]->NXM_NX_TUN_ID[0..31],set_field:10.16.1.17->tun_dst,output:1
table=90, priority=100,ip,nw_dst=10.48.18.0/24 actions=move:NXM_NX_REG0[]->NXM_NX_TUN_ID[0..31],set_field:10.16.1.18->tun_dst,output:1
table=90, priority=100,ip,nw_dst=10.48.22.0/24 actions=move:NXM_NX_REG0[]->NXM_NX_TUN_ID[0..31],set_field:10.16.1.22->tun_dst,output:1
table=90, priority=100,ip,nw_dst=10.48.23.0/24 actions=move:NXM_NX_REG0[]->NXM_NX_TUN_ID[0..31],set_field:10.16.1.23->tun_dst,output:1
table=90, priority=100,ip,nw_dst=10.48.24.0/24 actions=move:NXM_NX_REG0[]->NXM_NX_TUN_ID[0..31],set_field:10.16.1.24->tun_dst,output:1
table=90, priority=0 actions=drop

sh-4.2# ovs-ofctl -O OpenFlow13 dump-flows br0 | grep 10.48.21.33
table=20, priority=100,arp,in_port=3,arp_spa=10.48.21.33,arp_sha=00:00:0a:30:15:21/00:00:ff:ff:ff:ff actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:21
table=20, priority=100,ip,in_port=3,nw_src=10.48.21.33 actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:21
table=25, priority=100,ip,nw_src=10.48.21.33 actions=load:0x3e2d21->NXM_NX_REG0[],goto_table:30
table=40, priority=100,arp,arp_tpa=10.48.21.33 actions=output:3
table=70, priority=100,ip,nw_dst=10.48.21.33 actions=load:0x3e2d21->NXM_NX_REG1[],load:0x3->NXM_NX_REG2[],goto_table:80
```

Troubleshooting: OpenvSwitch

- check the flows
- log files: /var/log/openvswitch/ovs-vswitchd.log /var/log/openvswitch/ovsdb-server.log
- comments in the okd source code are your friend!
  - *pkg/network/node/ovscontroller.go*
- http://docs.openvswitch.org/en/latest/ref/
- diagnostic commands example: *ovs-appctl ofproto/trace br0 <options>*

```
# oc get pods -o wide | grep node21
ipfailover-1-sj22w      1/1       Running   1          1d        10.16.1.21    node21.lab.local
nginx-example-1-q56b5   1/1       Running   4          3d        10.48.21.33   node21.lab.local

# oc get netnamespace | grep -E 'webapp|client'
client                                 8078500    []
webapp                                 4074785    [10.16.1.225]
```

```
sh-4.2# ovs-appctl ofproto/trace br0 in_port=2,tcp,nw_src=10.48.22.1,nw_dst=10.48.22.37
Flow: tcp,in_port=2,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=10.48.22.1,nw_dst=10.48.22.37,nw_tos=0

bridge("br0")
-------------
 0. ip,in_port=2,nw_src=10.48.22.1, priority 400
    goto_table:30
30. ip,nw_dst=10.48.22.0/24, priority 200
    goto_table:70
70. ip,nw_dst=10.48.22.37, priority 100
    load:0x7b44a4->NXM_NX_REG1[]
    load:0x7->NXM_NX_REG2[]
    goto_table:80
80. ip,nw_src=10.48.22.1, priority 300
    output:NXM_NX_REG2[]
     -> output port is 7

Final flow: tcp,reg1=0x7b44a4,reg2=0x7,in_port=2,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=10.48.22.1
Megaflow: recirc_id=0,ct_state=-rpl,ip,in_port=2,nw_src=10.48.22.1,nw_dst=10.48.22.37,nw_frag=no
Datapath actions: 4
```

Troubleshooting: OpenvSwitch

- check the flows
- log files: /var/log/openvswitch/ovs-vswitchd.log /var/log/openvswitch/ovsdb-server.log
- comments in the okd source code are your friend!
  - *pkg/network/node/ovscontroller.go*
- http://docs.openvswitch.org/en/latest/ref/
- diagnostic commands example: *ovs-dpctl dump-conntrack*

```
# oc get pods -o wide | grep node21
ipfailover-1-sj22w      1/1       Running   1          1d        10.16.1.21    node21.lab.local
nginx-example-1-q56b5   1/1       Running   4          3d        10.48.21.33   node21.lab.local

# oc get netnamespace | grep -E 'webapp|client'
client                                 8078500    []
webapp                                 4074785    [10.16.1.225]
```

```
sh-4.2# ovs-dpctl dump-conntrack
udp,orig=(src=10.16.1.22,dst=10.16.1.23,sport=59136,dport=4789),reply=(src=10.16.1.23,dst=10.16.1.22,sport=4789,dport=59136)
udp,orig=(src=10.16.1.23,dst=10.16.1.22,sport=38650,dport=4789),reply=(src=10.16.1.22,dst=10.16.1.23,sport=4789,dport=38650)
tcp,orig=(src=10.16.1.11,dst=10.16.1.22,sport=54980,dport=10250),reply=(src=10.16.1.22,dst=10.16.1.11,sport=10250,dport=54980),protoinfo=(state=ESTABLISHED)
tcp,orig=(src=10.48.22.37,dst=10.48.23.25,sport=45890,dport=8080),reply=(src=10.48.23.25,dst=10.48.22.37,sport=8080,dport=45890),protoinfo=(state=TIME_WAIT)
tcp,orig=(src=127.0.0.1,dst=127.0.0.1,sport=47994,dport=43617),reply=(src=127.0.0.1,dst=127.0.0.1,sport=43617,dport=47994),protoinfo=(state=ESTABLISHED)
tcp,orig=(src=127.0.0.1,dst=127.0.0.1,sport=47928,dport=43617),reply=(src=127.0.0.1,dst=127.0.0.1,sport=43617,dport=47928),protoinfo=(state=ESTABLISHED)
udp,orig=(src=10.16.1.22,dst=10.16.1.23,sport=48785,dport=4789),reply=(src=10.16.1.23,dst=10.16.1.22,sport=4789,dport=48785)
udp,orig=(src=10.16.1.22,dst=10.16.1.22,sport=46560,dport=53),reply=(src=10.16.1.22,dst=10.16.1.22,sport=53,dport=46560)
tcp,orig=(src=10.16.1.22,dst=10.16.1.11,sport=52860,dport=8443),reply=(src=10.16.1.11,dst=10.16.1.22,sport=8443,dport=52860),protoinfo=(state=ESTABLISHED)
udp,orig=(src=10.16.1.22,dst=10.16.1.22,sport=48213,dport=53),reply=(src=10.16.1.22,dst=10.16.1.22,sport=53,dport=48213)
```

Further topics

- egress controllers
- tracing iptables rules
- networkpolicy and egressnetworkpolicy
- pod networks

Questions ?

Appendix: Lab configuration

OKD 3.11

Centos 7.6

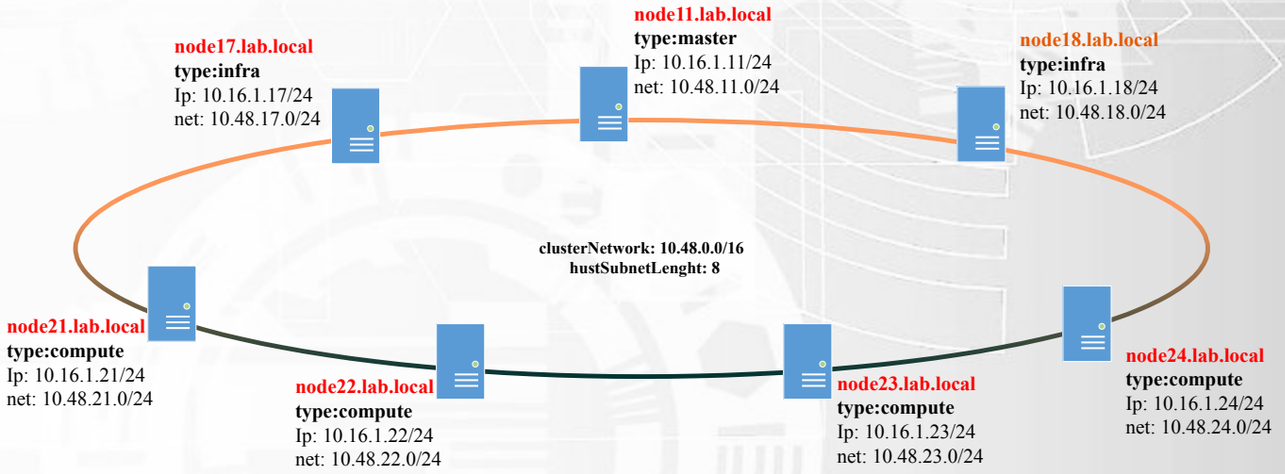
HW: DL380 Gen6 56G RAM

oc get nodes

NAME	STATUS	ROLES	AGE	VERSION
node11.lab.local	Ready	master	10d	v1.11.0+d4cacc0
node17.lab.local	Ready	infra	10d	v1.11.0+d4cacc0
node18.lab.local	Ready	infra	10d	v1.11.0+d4cacc0
node21.lab.local	Ready	compute	10d	v1.11.0+d4cacc0
node22.lab.local	Ready	compute	10d	v1.11.0+d4cacc0
node23.lab.local	Ready	compute	10d	v1.11.0+d4cacc0
node24.lab.local	Ready	compute	9d	v1.11.0+d4cacc0

oc get hostsubnet

NAME	HOST	HOST IP	SUBNET	EGRESS CIDRS	EGRESS IPS
node11.lab.local	node11.lab.local	10.16.1.11	10.48.11.0/24	[]	[]
node17.lab.local	node17.lab.local	10.16.1.17	10.48.17.0/24	[10.16.1.224/27]	[10.16.1.225]
node18.lab.local	node18.lab.local	10.16.1.18	10.48.18.0/24	[10.16.1.224/27]	[]
node21.lab.local	node21.lab.local	10.16.1.21	10.48.21.0/24	[]	[]
node22.lab.local	node22.lab.local	10.16.1.22	10.48.22.0/24	[]	[]
node23.lab.local	node23.lab.local	10.16.1.23	10.48.23.0/24	[]	[]
node24.lab.local	node24.lab.local	10.16.1.24	10.48.24.0/24	[]	[]



Appendix: openflow tables definition for Openshift

// Table 0: initial dispatch based on in_port
// Table 10: VXLAN ingress filtering; filled in by AddHostSubnetRules()
// Table 20: from OpenShift container; validate IP/MAC, assign tenant-id; filled in by setupPodFlows
// Table 21: from OpenShift container; NetworkPolicy plugin uses this for connection tracking
// Table 25: IP from OpenShift container via Service IP; reload tenant-id; filled in by setupPodFlows
// Table 30: general routing
// Table 40: ARP to local container, filled in by setupPodFlows
// Table 50: ARP to remote container; filled in by AddHostSubnetRules()
// Table 60: IP to service from pod
// Table 70: IP to local container: vnid/port mappings; filled in by setupPodFlows
// Table 80: IP policy enforcement; mostly managed by the osdnPolicy
// Table 90: IP to remote container; filled in by AddHostSubnetRules()
// Table 100: egress routing; edited by SetNamespaceEgress()*
// Table 101: egress network policy dispatch; edited by UpdateEgressNetworkPolicy()
// Table 110: outbound multicast filtering, updated by UpdateLocalMulticastFlows()
// Table 111: multicast delivery from local pods to the VXLAN; only one rule, updated by UpdateVXLANMulticastRules()
// Table 120: multicast delivery to local pods (either from VXLAN or local pods); updated by UpdateLocalMulticastFlows()
// Table 253: rule version note

source: pkg/network/node/ovscontroller.go