Nodus Network Policy and OVN Balancer

By:
Kuralamudhan Ramakrishnan (kuralamudhan.ramakrishnan@intel.com) | July 19th, 2021

Acknowledgement:
Srinivasa Addepalli, Ritu Sood
• OVS Conntrack Intro
• OVN ACLs
• OVN Load Balancer
• OVN4NFV Network Policy design
• OVN4NFV OVN Balancer
OVS Conntrack Intro

• OVS uses the Connection Tracking system along with OpenFlow flows.
• OpenFlows match on the state of the TCP, UDP connections
• Important details to understand on the Conntrack Fields
  • ct_state:
    • new: new connection or uncommitted connection
    • Trk: packet is tracked
    • Est: packet in the established connection
OVS Conntrack Intro
(flow #1)
$ ovs-ofctl add-flow br0 \n   "table=0, priority=50, ct_state=-trk, tcp, in_port=veth_l0, actions=ct(table=0)"

(flow #2)
$ ovs-ofctl add-flow br0 \n   "table=0, priority=50, ct_state=+trk+new, tcp, in_port=veth_l0, actions=ct(commit),veth_r0"

(flow #3)
$ ovs-ofctl add-flow br0 \n   "table=0, priority=50, ct_state=-trk, tcp, in_port=veth_r0, actions=ct(table=0)"

(flow #4)
$ ovs-ofctl add-flow br0 \n   "table=0, priority=50, ct_state=+trk+est, tcp, in_port=veth_r0, actions=veth_l0"

(flow #5)
$ ovs-ofctl add-flow br0 \n   "table=0, priority=50, ct_state=+trk+est, tcp, in_port=veth_l0, actions=veth_r0"
## TCP Segment

<table>
<thead>
<tr>
<th>TCP Segment</th>
<th>ct_state(flow#)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connection Setup</strong></td>
<td></td>
</tr>
<tr>
<td>192.168.0.2 → 10.0.0.2 [SYN] Seq=0</td>
<td>-trk(#1) then +trk+new(#2)</td>
</tr>
<tr>
<td>10.0.0.2 → 192.168.0.2 [SYN, ACK] Seq=0 Ack=1</td>
<td>-trk(#3) then +trk+est(#4)</td>
</tr>
<tr>
<td>192.168.0.2 → 10.0.0.2 [ACK] Seq=1 Ack=1</td>
<td>-trk(#1) then +trk+est(#5)</td>
</tr>
<tr>
<td><strong>Data Transfer</strong></td>
<td></td>
</tr>
<tr>
<td>192.168.0.2 → 10.0.0.2 [ACK] Seq=1 Ack=1</td>
<td>-trk(#1) then +trk+est(#5)</td>
</tr>
<tr>
<td>10.0.0.2 → 192.168.0.2 [ACK] Seq=1 Ack=2</td>
<td>-trk(#3) then +trk+est(#4)</td>
</tr>
<tr>
<td>Connection Teardown</td>
<td></td>
</tr>
<tr>
<td>192.168.0.2 → 10.0.0.2 [FIN, ACK] Seq=2 Ack=1</td>
<td>-trk(#1) then +trk+est(#5)</td>
</tr>
<tr>
<td>10.0.0.2 → 192.168.0.2 [FIN, ACK] Seq=1 Ack=3</td>
<td>-trk(#3) then +trk+est(#4)</td>
</tr>
<tr>
<td>192.168.0.2 → 10.0.0.2 [ACK] Seq=3 Ack=2</td>
<td>-trk(#1) then +trk+est(#5)</td>
</tr>
</tbody>
</table>
OVN ACL Intro

• OVN uses the OVS+Conntrack to implement ACLs
• ACLs configured, there are new entries in the logical flow table in the stages switch_in_pre_acl, switch_in_acl, switch_out_pre_acl, and switch_out_acl.
• Let create following rules with OVN ACLs
  • Allow incoming ICMP requests and associated return traffic.
  • Allow incoming SSH connections and associated return traffic.
  • Drop other incoming IP traffic.
• OVN ACLs
• For Example:- Pod with name “acl” has the port default_acl-66d888699-sfs26 created by ovn4nfv

```
ovn-nbctl acl-add ovn4nfvk8s-default-nw 1001 'outport == "default_acl-66d888699-sfs26" && ip && tcp && tcp.dst == 22' allow-related
ovn-nbctl acl-add ovn4nfvk8s-default-nw 1001 'outport == " default_acl-66d888699-sfs26 " && icmp' drop
```
Network Policy -> OVN ACLS

Ingress by OVN ACL

```
ovn-nbctl acl-add to-lport 1001 'outport == portGroupName
priority match= "ip4.src == 172.17.0.0/16 && ip4.src != 172.17.1.0/24"' allow-related
```

Egress by OVN ACL

```
ovn-nbctl acl-add to-lport 1001 'outport == portGroupName
priority match= "ip4.dst == 10.0.0.0/24"' allow-related
```
OVN ACLs to implement Network policy – Update NPs
OVN ACLs to implement Network policy – delete NPs
OVN Load Balancer

- OVN Load Balancer provides a hash-based load balancing mechanism, which can be used on logical switches or logical routers:
  - Used on logical router
    - Can only be used on gateway router
    - Centralized (rather than distributed)
  - Used in logical switch
    - Distributed
    - OVN Load Balancer can be used in client logical switch
- Sample here:
  
  ```
  # uuid=`ovn-nbctl create load_balancer vips:10.254.10.10="192.168.100.10,192.168.100.11```
  
  # ovn-nbctl set logical_switch ls2 load_balancer=$uuid
  # ovn-nbctl get logical_switch ls2 load_balancer
  # ovn-nbctl ls-lb-list ls2
  
<table>
<thead>
<tr>
<th>UUID</th>
<th>LB</th>
<th>PROTO</th>
<th>VIP</th>
<th>IPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>a19bece1-52bf-4555-89f4-257534c0b9d9</td>
<td>tcp/udp</td>
<td>10.254.10.10</td>
<td>192.168.100.10,192.168.100.11</td>
<td></td>
</tr>
</tbody>
</table>
OVN Load Balancer to implement SVC/Endpoint – Update SVC
OVN Load Balancer to implement SVC/Endpoint – delete SVC

Admin

deploy

controller

init

Service
listener

OVN Client
Load Balancer

create Load Balancer for both TCP and UDP protocol along with session affinity

get the SVC loadbalancer VIPs and UUID based on the SVC Session Affinity
decide the TCP/UDP LB or TCP/UDP Session Load Balancer

Delete SVC VIP and backends to the TCP/UDP LB or to the TCP/UDP Session LB

USER

delete svc

update delete svc

handle svc delete

DeleteLoadBalancerVIP

return
Nodus deployment Model

Network C

192.168.10.2

Network Physical Functions

172.16.2.10

Network B

192.168.10.2

Network Physical Functions

172.16.2.10

Pod B

172.16.2.12
192.16.10.3
10.254.10.3

ovs br

Interface 3
Interface 2
Interface 1

Pod A

172.16.2.14
192.16.10.4
10.254.10.4

ovs br

Interface 3
Interface 2
Interface 1

switch

10.254.10.1

Router

Network A

Internet