OVN-Kubernetes (OVN-K)

Jing Yan

2024.07.06

Agenda

- Background
- What is OVN-K?
- OVN-K in k8s
- Production env: k8s cluster with OVN-K
- Network topologies: k8s cluster with OVN-K
- OVN-K’s features
- Typical network data path
  - Pod-to-Pod - same node; Pod-to-Pod - different nodes; Pod-to-ClusterIP; Pod/External-to-NodePort
- Typical use-cases
  - Namespaces Isolation; Namespace-Namespace-ClusterIP Isolation

Background (1)

- Flannel
  - Encapsulation and routing protocols: VXLAN
  - Encryption: IPsec
  - **(Cons) Network management: No network rules/policies/ACLs**
  - Feel free to check the following topics via internal GitLab 03_Flannel&VXLAN_Research-20240629
    - Deployment Process of Flannel
    - The Workflow for Flannel to Assign an IP to a Pod
    - Pod-to-Pod Communication on the Same Node
    - Pod-to-Pod Communication Across Nodes
    - GitLab link: https://gitlab.metastonecorp.com/yanjing/cloudresearch
- Calico
  - Encapsulation and routing protocols: IP-in-IP, VXLAN, BGP
  - Encryption: WireGuard
  - Network management: Policy management and ACLs
  - **(Cons) No multicast support**

Background (2)

- **Red Hat OpenShift (1 cluster contains at max 2000 nodes) [1]**
  - **OpenShift-SDN, OVN-Kubernetes, Istio**
- Amazon Elastic Kubernetes Service (EKS) [2]
  - Amazon VPC CNI Plugin, **Calico**, AWS App Mesh
- Google Kubernetes Engine (GKE) [3]
  - Google Cloud VPC Native Cluster, **Calico**, Istio
- Azure Kubernetes Service (AKS) [4]
  - Azure CNI Plugin, **Calico**, Azure Service Mesh
- IBM Cloud Kubernetes Service [5]
  - **Calico**, VPC Native Networking, Istio
- Oracle Container Engine for Kubernetes (OKE) [6]
  - VPC Native Networking, **Calico**, Service Mesh
- Rancher (by SUSE) [7]
  - Flannel, **Calico**, Canal, Istio
- VMware Tanzu Kubernetes Grid [8]
  - NSX-T, Antrea, **Calico**, Istio

[1] https://access.redhat.com/products/openshift
[2] https://aws.amazon.com/eks/
[3] https://cloud.google.com/kubernetes-engine?hl=en#the-most-scalable-and-fully-automated-kubernetes-service
[4] https://azure.microsoft.com/en-us/products/kubernetes-service
[5] https://www.ibm.com/products/kubernetes-service
[6] https://www.oracle.com/sg/cloud/cloud-native/container-engine-kubernetes/
[7] https://www.rancher.com/
[8] https://tanzu.vmware.com/kubernetes-grid

What is OVN-K [1]?

CNI network plugin for k8s

- Uses OVN on OVS as the abstraction to manage network traffic flows on the node
- Creates logical network topologies
    - Logical switches, routers, ports, acls (network policies), load balancers etc.
- Uses the Geneve (Generic Network Virtualization Encapsulation) protocol to create an overlay network between nodes.

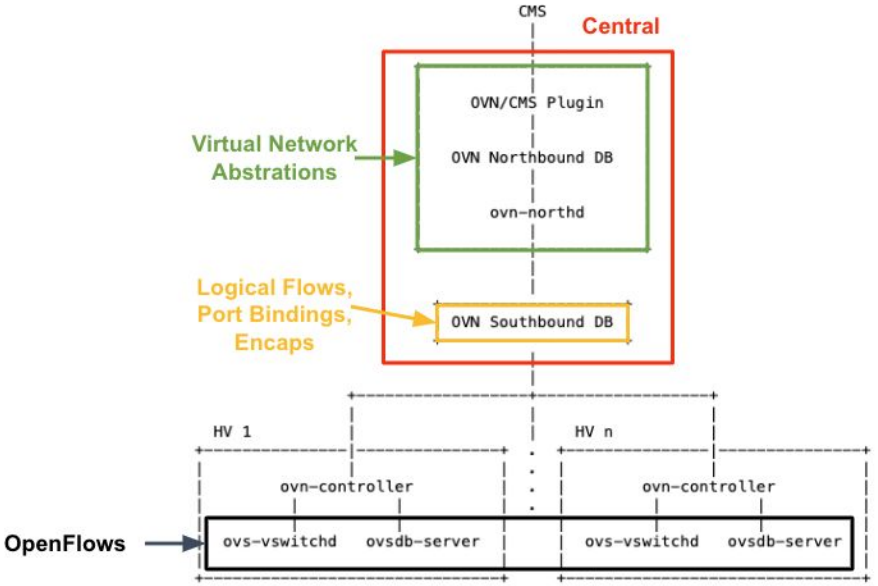


Figure: ovn architecture [2]

OVN-K in k8s

Control Plane Components

- ovnkube-master: Central management, monitors k8s API, translates events into OVN logical elements, manages network topology and IPAM.
- nbdb: Stores logical entities created by ovnkube-master.
- northd: Converts entities into logical flows, inserts into sbdb.
- sbdb: Stores logical flows created by northd.

Node Components

- ovnkube-node: Called as a CNI plugin from kubelet/Containerd, creates virtual interfaces for pods, programs OpenFlows and IPtables for services.
- ovn-controller: Converts logical flows into OpenFlows, programs onto OVS.

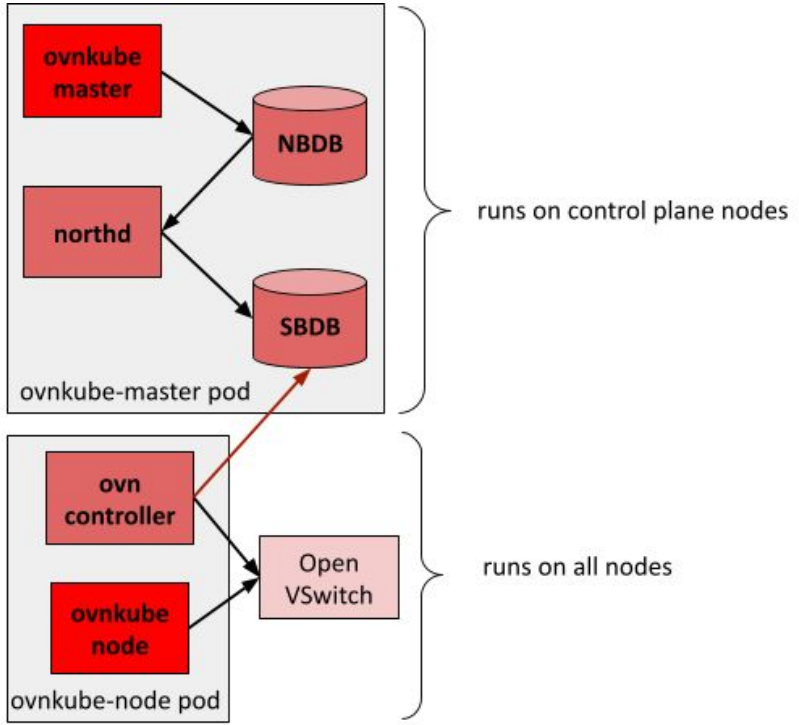


Figure: OVN-K Legacy Architecture [1]

Production env: k8s cluster with OVN-K (1)

```
root@k8stestbed:~# kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   14h
root@k8stestbed:~# kubectl get nodes -A -o wide
NAME                STATUS   ROLES           AGE   VERSION   INTERNAL-IP
ovn-control-plane   Ready    control-plane   14h   v1.29.2   172.18.0.3
ovn-worker          Ready    <none>          14h   v1.29.2   172.18.0.2
ovn-worker2         Ready    <none>          14h   v1.29.2   172.18.0.4
```

```
root@k8stestbed:~# kubectl get pod -A -o wide
NAMESPACE            NAME                                         READY   STATUS    RESTARTS        AGE   IP           NODE
TES
kube-system          coredns-76f75df574-dhxwc                     1/1     Running   0               14h   10.244.1.6   ovn-control-plane
kube-system          coredns-76f75df574-njnbv                     1/1     Running   0               14h   10.244.1.4   ovn-control-plane
kube-system          etcd-ovn-control-plane                       1/1     Running   0               14h   172.18.0.3   ovn-control-plane
kube-system          kube-apiserver-ovn-control-plane             1/1     Running   0               14h   172.18.0.3   ovn-control-plane
kube-system          kube-controller-manager-ovn-control-plane    1/1     Running   0               14h   172.18.0.3   ovn-control-plane
kube-system          kube-scheduler-ovn-control-plane             1/1     Running   0               14h   172.18.0.3   ovn-control-plane
local-path-storage   local-path-provisioner-7577fdbbfb-jhm98      1/1     Running   0               14h   10.244.1.5   ovn-control-plane
ovn-kubernetes       ovnkube-db-5898476556-vbfvc                  2/2     Running   0               14h   172.18.0.3   ovn-control-plane
ovn-kubernetes       ovnkube-identity-5f9795db4f-d7cpn            1/1     Running   0               14h   172.18.0.3   ovn-control-plane
ovn-kubernetes       ovnkube-master-85d9f47587-ssjqt              2/2     Running   0               14h   172.18.0.3   ovn-control-plane
ovn-kubernetes       ovnkube-node-6x6gf                           3/3     Running   1 (4h39m ago)   14h   172.18.0.4   ovn-worker2
ovn-kubernetes       ovnkube-node-hrds2                           3/3     Running   1 (4h39m ago)   14h   172.18.0.2   ovn-worker
ovn-kubernetes       ovnkube-node-xq9cw                           3/3     Running   1 (4h38m ago)   14h   172.18.0.3   ovn-control-plane
ovn-kubernetes       ovs-node-87xzx                               1/1     Running   0               14h   172.18.0.4   ovn-worker2
ovn-kubernetes       ovs-node-g4x4j                               1/1     Running   0               14h   172.18.0.3   ovn-control-plane
ovn-kubernetes       ovs-node-nqklg                               1/1     Running   0               14h   172.18.0.2   ovn-worker
```

Production env: k8s cluster with OVN-K (2)

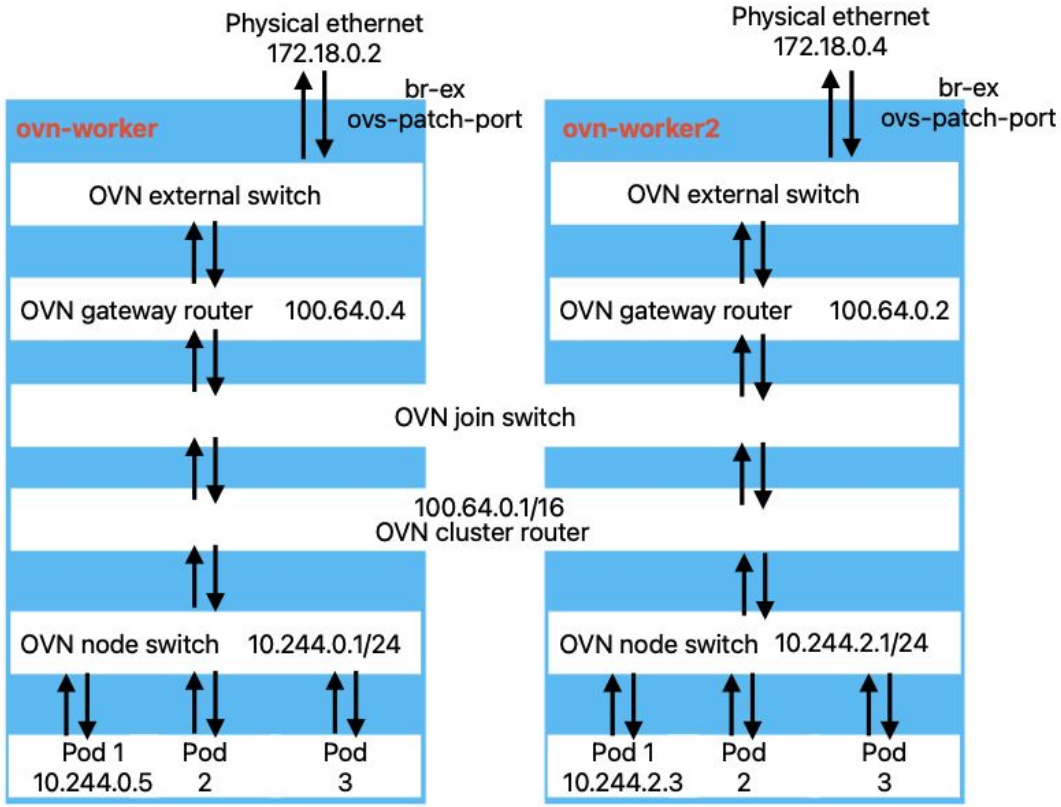
Logical entities created by ovnkube-master, store in nbdb

```
root@k8stestbed:~# ovn-nbctl --db=tcp:172.18.0.3:6641 show | egrep 'switch |router '
switch 2bf131c8-170e-4d2d-b183-34e384d3e90c (ext_ovn-worker2)
switch f90f31c8-296c-4701-a722-215aa244c6ba (ext_ovn-control-plane)
switch 17e8f245-041e-4eaa-847e-7509655ffae1 (join)
switch d4768955-dc1a-4d08-8198-cca5808d79ce (ovn-control-plane)
switch aaec2c32-ab84-478e-b8aa-7fbd28560117 (ovn-worker)
switch 2b4dccff-60b8-4b84-8067-40f77d009736 (ovn-worker2)
switch 1d56a86b-9e48-40a3-bf36-34fe40916b15 (ext_ovn-worker)
router 86a0a658-8fef-409d-a6f9-7f6e75939c11 (GR_ovn-worker2)
router 9c8f39bc-a21a-4ec1-a3fc-c1973232dd40 (ovn_cluster_router)
router 860c6e35-d020-4e10-aa0b-84a45157fbf2 (GR_ovn-worker)
router eac3dcdb-2bb4-4722-ae49-3873b27a71f4 (GR_ovn-control-plane)
```

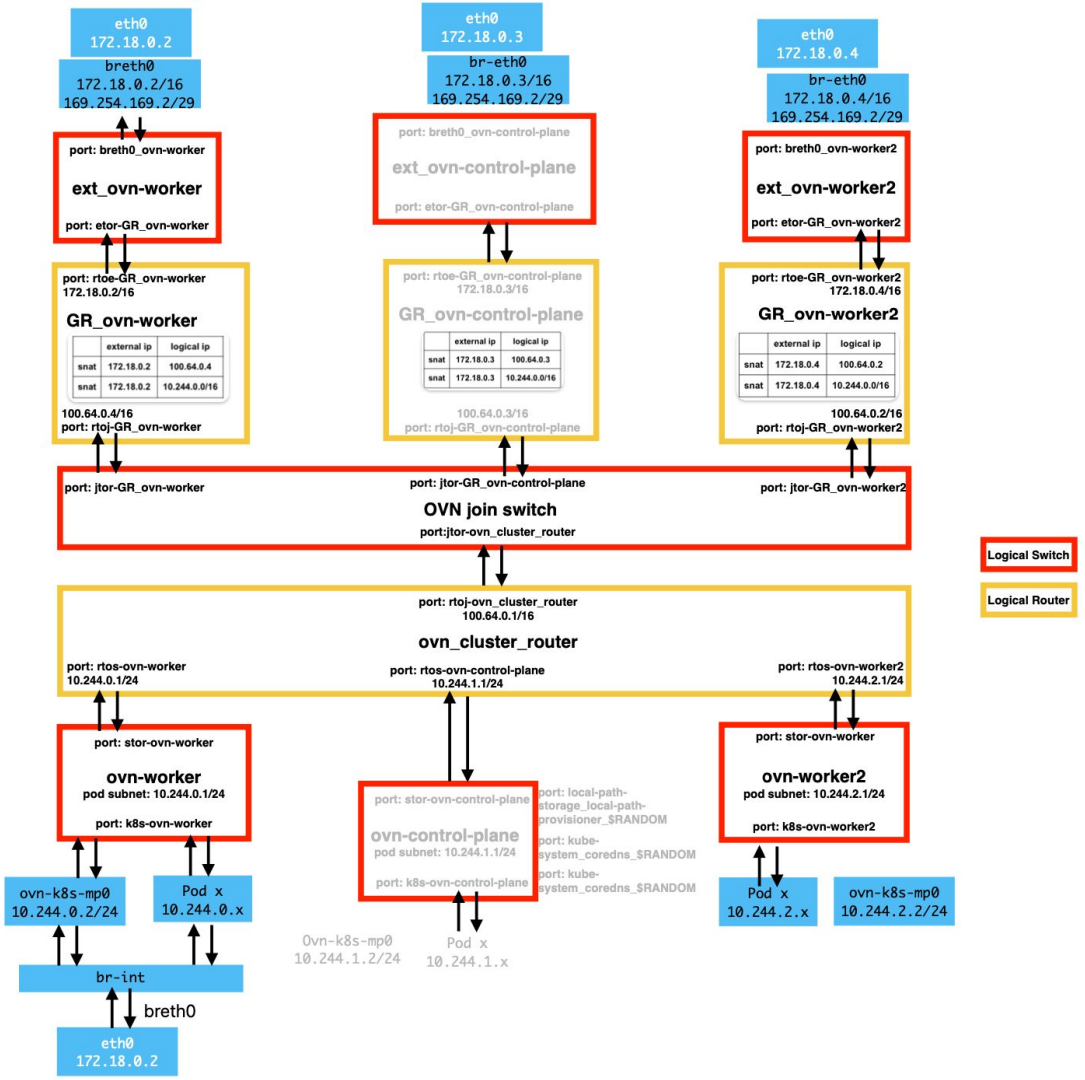
Logical flows created by northd, store in sbdb

```
root@k8stestbed:~# ovn-sbctl --db=tcp:172.18.0.3:6642 show | grep "Chassis " -A3
Chassis "c2791bbc-f3c4-491b-8134-2e11d2b431d5"
    hostname: ovn-control-plane
    Encap geneve
        ip: "172.18.0.3"
--
Chassis "31af130b-04ce-4fe0-904d-5a4024f4624d"
    hostname: ovn-worker
    Encap geneve
        ip: "172.18.0.2"
--
Chassis "1af56874-cfaa-4d0d-8057-fd54d96d29aa"
    hostname: ovn-worker2
    Encap geneve
        ip: "172.18.0.4"
```

Network topologies: k8s cluster with OVN-K (1)



Network topologies: k8s cluster with OVN-K (2)

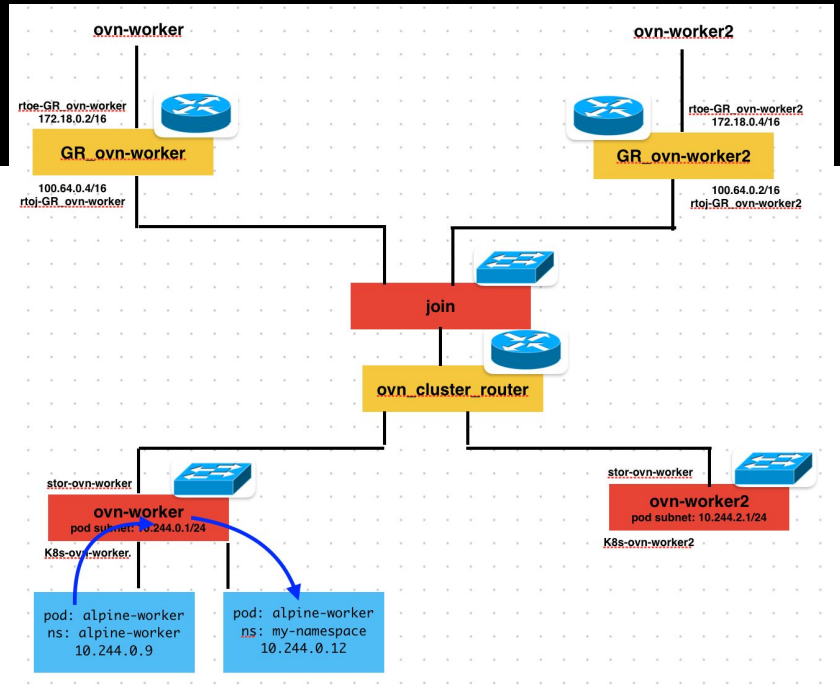


OVN-K’s features

- **(Cons) Assigning a specific subnet to a namespace – Not supported**
  - OVN-Kubernetes typically handles IP address management at the node level rather than the namespace level.
- Network security control – Supported
  - **Could be used for Isoation (Isolate tenants, allow/deny traffic to namespaces, etc.)**
- Cluster egress control – Supported
  - EgressIP, EgressService, EgressQoS, EgressGateway
- Infrastructure security control – Supported
  - Node indentity
- Multiple networking – Supported
  - A K8s pod can have more than one network interface, multiple network policies, multiple VTEP when k8s nodes have multiple SR-IOV adapters.
- Multicast – Supported
  - data could be delivered to multiple IP addresses
- Hardware Acceleration – Supported

Network data path (1): Pod-to-Pod - same node

```
root@k8stestbed:~/acl# kubectl get pods -A -o wide | grep alpine | grep -v worker2
alpine-worker        alpine-worker                        1/1     Running   0              6h41m   10.244.0.9
my-namespace         alpine-worker                        1/1     Running   0              5h11m   10.244.0.12
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl exec -it -n alpine-worker alpine-worker -- ping -c1 10.244.0.12
PING 10.244.0.12 (10.244.0.12): 56 data bytes
64 bytes from 10.244.0.12: seq=0 ttl=64 time=0.528 ms

--- 10.244.0.12 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.528/0.528/0.528 ms
```



Use-cases (1): Namespaces Isolation

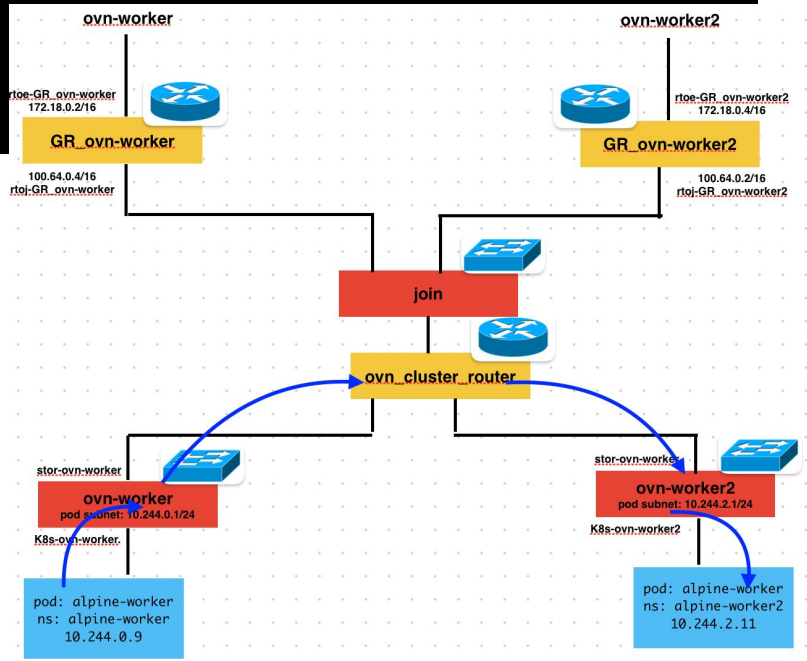
```
root@k8stestbed:~/acl# kubectl get pods -A -o wide | grep alpine | grep -v worker2
alpine-worker        alpine-worker                              1/1     Running   0              6h41m   10.244.0.9
my-namespace         alpine-worker                              1/1     Running   0              5h11m   10.244.0.12
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl exec -it -n alpine-worker alpine-worker -- ping -c1 10.244.0.12
PING 10.244.0.12 (10.244.0.12): 56 data bytes
64 bytes from 10.244.0.12: seq=0 ttl=64 time=0.528 ms

--- 10.244.0.12 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.528/0.528/0.528 ms
```

```
root@k8stestbed:~/acl# cat namesapce-isolation.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-ingress-from-alpine-worker
  namespace: my-namespace
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              namespace: alpine-worker
```

```
root@k8stestbed:~/acl# kubectl apply -f namesapce-isolation.yaml
networkpolicy.networking.k8s.io/deny-ingress-from-alpine-worker created
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl exec -it -n alpine-worker alpine-worker -- ping -c1 10.244.0.12
PING 10.244.0.12 (10.244.0.12): 56 data bytes
^C
--- 10.244.0.12 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
command terminated with exit code 1
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl delete -f namesapce-isolation.yaml
networkpolicy.networking.k8s.io "deny-ingress-from-alpine-worker" deleted
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl exec -it -n alpine-worker alpine-worker -- ping -c1 10.244.0.12
PING 10.244.0.12 (10.244.0.12): 56 data bytes
64 bytes from 10.244.0.12: seq=0 ttl=64 time=0.503 ms

--- 10.244.0.12 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.503/0.503/0.503 ms
```

Network data path (2): Pod-to-Pod - different nodes

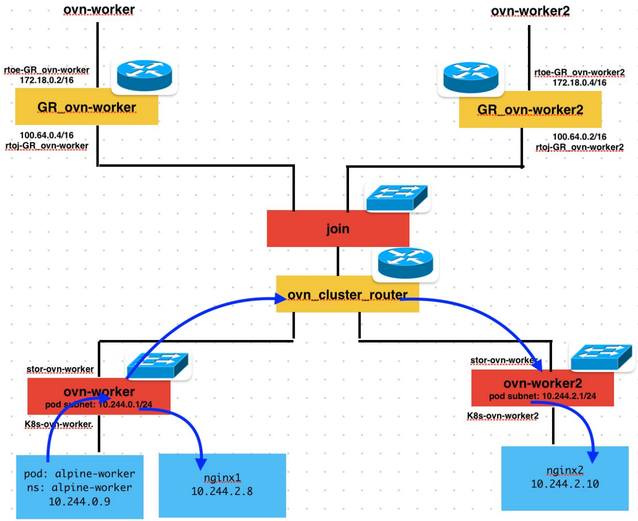
```
root@k8stestbed:~/acl# kubectl get pods -A -o wide | grep alpine | grep -v "my-namespace"
alpine-worker        alpine-worker                              1/1     Running   0               8h      10.244.0.9
alpine-worker2       alpine-worker                              1/1     Running   0               8h      10.244.2.11
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl exec -it -n alpine-worker alpine-worker -- ping -c1 10.244.2.11
PING 10.244.2.11 (10.244.2.11): 56 data bytes
64 bytes from 10.244.2.11: seq=0 ttl=63 time=1.463 ms

--- 10.244.2.11 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.463/1.463/1.463 ms
```



Network data path (3): Pod-to-ClusterIP

```
root@k8stestbed:~/acl# kubectl get svc -A -o wide| grep nginx
nginx-namespace   nginx-service   ClusterIP   10.96.130.76   <none>        80/TCP                  8h     app=nginx
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl get pods -A -o wide | grep nginx
nginx-namespace      nginx-deployment-7c79c4bf97-5bzgd           1/1     Running   0               8h      10.244.2.10   ovn-worker2
nginx-namespace      nginx-deployment-7c79c4bf97-htdxp           1/1     Running   0               8h      10.244.0.8    ovn-worker
nginx-namespace      nginx-deployment-7c79c4bf97-zfqpb           1/1     Running   0               8h      10.244.1.8    ovn-control-plane
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl get pods -A -o wide | grep alpine | grep -v "my-namespace" | grep -v "worker2"
alpine-worker        alpine-worker                               1/1     Running   0               8h      10.244.0.9    ovn-worker
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl exec -it -n alpine-worker alpine-worker -- wget --spider http://10.96.130.76
Connecting to 10.96.130.76 (10.96.130.76:80)
remote file exists
```

```
@k8stestbed:~/acl# ovn-sbctl --db=tcp:172.18.0.3:6642 lflow-list | grep 10.96.130.76
ble=5 (lr_in_defrag       ), priority=100   , match=(ip && ip4.dst == 10.96.130.76), action=(ct_dnat;)
ble=7 (lr_in_dnat         ), priority=120   , match=(ct.new && !ct.rel && ip4 && ip4.dst == 10.96.130.76
ark(backends=10.244.0.8:80,10.244.1.8:80,10.244.2.10:80; force_snat);)
ble=5 (lr_in_defrag       ), priority=100   , match=(ip && ip4.dst == 10.96.130.76), action=(ct_dnat;)
ble=7 (lr_in_dnat         ), priority=120   , match=(ct.new && !ct.rel && ip4 && ip4.dst == 10.96.130.76
ark(backends=10.244.0.8:80,10.244.1.8:80,10.244.2.10:80; force_snat);)
ble=5 (lr_in_defrag       ), priority=100   , match=(ip && ip4.dst == 10.96.130.76), action=(ct_dnat;)
ble=7 (lr_in_dnat         ), priority=120   , match=(ct.new && !ct.rel && ip4 && ip4.dst == 10.96.130.76
ark(backends=10.244.0.8:80,10.244.1.8:80,10.244.2.10:80; force_snat);)
```



Use-cases (3): Namespace-ClusterIP Isolation

```
root@k8stestbed:~/acl# kubectl get svc -A -o wide| grep nginx
nginx-namespace   nginx-service   ClusterIP   10.96.130.76   <none>        80/TCP                   8h    app=nginx
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl get pods -A -o wide | grep nginx
nginx-namespace      nginx-deployment-7c79c4bf97-5bzgd          1/1     Running   0              8h      10.244.2.10   ovn-worker2
nginx-namespace      nginx-deployment-7c79c4bf97-htdxp          1/1     Running   0              8h      10.244.0.8    ovn-worker
nginx-namespace      nginx-deployment-7c79c4bf97-zfqpb          1/1     Running   0              8h      10.244.1.8    ovn-control-plane
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl get pods -A -o wide | grep alpine | grep -v "my-namespace" | grep -v "worker2"
alpine-worker        alpine-worker                              1/1     Running   0              8h      10.244.0.9    ovn-worker
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl exec -it -n alpine-worker alpine-worker -- wget --spider http://10.96.130.76
Connecting to 10.96.130.76 (10.96.130.76:80)
remote file exists
```

```
root@k8stestbed:~/acl# cat ns-svc-isolation.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-specific-ip-blocks
  namespace: nginx-namespace
spec:
  podSelector:
    matchLabels:
      app: nginx
  policyTypes:
  - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              namespace: alpine-worker
```

```
root@k8stestbed:~/acl# kubectl apply -f ns-svc-isolation.yaml
networkpolicy.networking.k8s.io/deny-specific-ip-blocks created
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl exec -it -n alpine-worker alpine-worker -- wget --spider http://10.96.130.76
Connecting to 10.96.130.76 (10.96.130.76:80)
^Ccommand terminated with exit code 130
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl delete -f ns-svc-isolation.yaml
networkpolicy.networking.k8s.io "deny-specific-ip-blocks" deleted
root@k8stestbed:~/acl#
root@k8stestbed:~/acl# kubectl exec -it -n alpine-worker alpine-worker -- wget --spider http://10.96.130.76
Connecting to 10.96.130.76 (10.96.130.76:80)
remote file exists
```

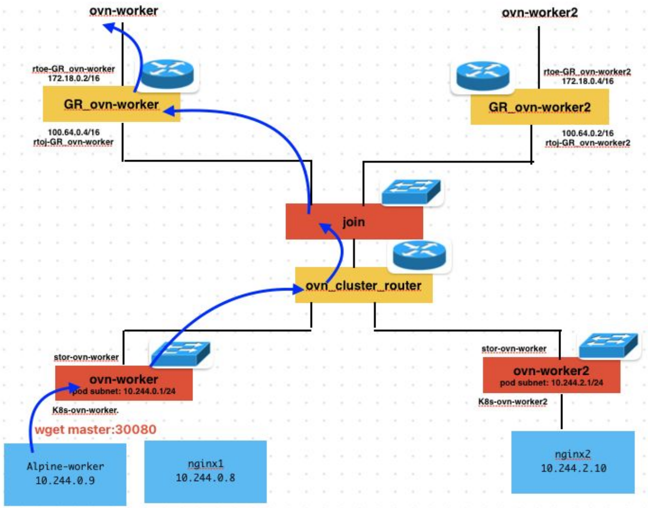
Network data path (4): Pod-to-NodePort (1)

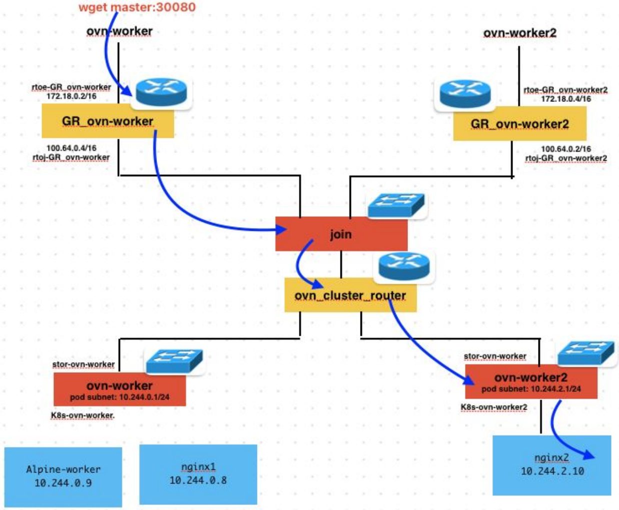
```
root@k8stestbed:~# kubectl get deployments -A | grep nginx
nginx-namespace      nginx-deployment         3/3     3            3           15h
root@k8stestbed:~#
root@k8stestbed:~# kubectl get svc -A -o wide | grep nginx
nginx-namespace   nginx-service   NodePort    10.96.29.186   <none>        80:30080/TCP
root@k8stestbed:~#
root@k8stestbed:~# kubectl get nodes -A -o wide
NAME                STATUS   ROLES           AGE   VERSION   INTERNAL-IP   EXTERNAL-IP
ovn-control-plane   Ready    control-plane   34h   v1.29.2   172.18.0.3    <none>
ovn-worker          Ready    <none>          34h   v1.29.2   172.18.0.2    <none>
ovn-worker2         Ready    <none>          34h   v1.29.2   172.18.0.4    <none>
root@k8stestbed:~#
root@k8stestbed:~# wget --spider http://172.18.0.3:30080
Spider mode enabled. Check if remote file exists.
--2024-07-11 09:33:55--  http://172.18.0.3:30080/
Connecting to 172.18.0.3:30080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 615 [text/html]
Remote file exists and could contain further links,
but recursion is disabled -- not retrieving.
```

```
root@k8stestbed:~/nginx# kubectl exec -it alpine-worker -n alpine-worker -- wget --spider http://172.18.0.2:30080
Connecting to 172.18.0.2:30080 (172.18.0.2:30080)
remote file exists
```

Network data path (4): Pod/External-to-NodePort (2)

```
root@k8stestbed:~# ovn-sbctl --db=tcp:172.18.0.3:6642 lflow-list | grep 30080
  table=7 (lr_in_dnat         ), priority=120  , match=(ct.new && !ct.rel && ip4 && ip4.dst == ^NODEIP_IPv4_0 && tcp && tcp.dst == 30080),
10.244.1.8:80,10.244.2.10:80; force_snat);)
  table=7 (lr_in_dnat         ), priority=120  , match=(ct.new && !ct.rel && ip4 && ip4.dst == ^NODEIP_IPv4_0 && tcp && tcp.dst == 30080),
10.244.1.8:80,10.244.2.10:80; force_snat);)
  table=7 (lr_in_dnat         ), priority=120  , match=(ct.new && !ct.rel && ip4 && ip4.dst == ^NODEIP_IPv4_0 && tcp && tcp.dst == 30080),
10.244.1.8:80,10.244.2.10:80; force_snat);)
  table=6 (ls_in_pre_stateful ), priority=120  , match=(reg0[2] == 1 && ip4.dst == ^NODEIP_IPv4_0 && tcp.dst == 30080), action=(reg1 = ^NOD
  table=13(ls_in_lb           ), priority=120  , match=(ct.new && ip4.dst == ^NODEIP_IPv4_0 && tcp.dst == 30080), action=(reg0[1] = 0; ct_l
  table=6 (ls_in_pre_stateful ), priority=120  , match=(reg0[2] == 1 && ip4.dst == ^NODEIP_IPv4_0 && tcp.dst == 30080), action=(reg1 = ^NOD
  table=13(ls_in_lb           ), priority=120  , match=(ct.new && ip4.dst == ^NODEIP_IPv4_0 && tcp.dst == 30080), action=(reg0[1] = 0; ct_l
  table=6 (ls_in_pre_stateful ), priority=120  , match=(reg0[2] == 1 && ip4.dst == ^NODEIP_IPv4_0 && tcp.dst == 30080), action=(reg1 = ^NOD
  table=13(ls_in_lb           ), priority=120  , match=(ct.new && ip4.dst == ^NODEIP_IPv4_0 && tcp.dst == 30080), action=(reg0[1] = 0; ct_l
```





Thank you!