

OpenShift introduction

wji@ 2022/09



Info

[OpenShift content hub \(internal\)](#)

<https://redhat.hightspot.com/spots/5fda4a94a4dfa0396bab090e?list=5b646a6e1279585581196a55&overview=false>

- [What's New in OpenShift 4.11](#)
- [OpenShift New Feature Enablement Presentation – Logging 5.5 – Features](#)
- [OpenShift New Feature Enablement - Logging 5.5 - Common Labels](#)
- [OpenShift New Feature Enablement - Logging 5.5 - Separate Indices](#)
- [OpenShift New Feature Enablement Presentation – Logging 5.5 - Vector changes](#)
- [OpenShift New Feature Enablement Presentation – Logging 5.5 - Loki Integration](#)
- [OpenShift 4.11 Logging Release notes](#)

[CNV 简单 demo 演示文档](#)

[Openshift 和 Docker 的命令汇总](#)

[Openshift 命令与 yaml 文件的匹配表](#)

[Red Hat OpenShift Container Platform Cheat Sheet](#)

Resources

Of course there is much more information outside the above mentioned highlights.

Here are a few resources to visit for your next steps with Red Hat OpenShift Data Foundation 4.11:

- Red Hat OpenShift Data Foundation 4.11 [landing page](#) on The Source
- Disaster recovery for Red Hat OpenShift workloads [deck](#) and [video](#).
- What's New with Red Hat OpenShift Data Foundation 4.11 [deck](#) and [video](#).
- Red Hat OpenShift Data Foundation [product information](#).
- Red Hat OpenShift Data Foundation official [documentation](#).



console.redhat.com

The screenshot shows the Red Hat Hybrid Cloud Console interface at <https://console.redhat.com/openshift/create/datacenter>. The left sidebar has a red box around the 'Clusters' link under the 'OpenShift' section. The main content area shows 'Create an OpenShift cluster' with tabs for 'Cloud' and 'Datacenter' (which also has a red box around it). Below the tabs, the 'Assisted Installer' method is selected, indicated by a red box around the 'Create cluster' button.

Thanks my manager lijin@ and mentor ngu@ for huge help.

Minimum hardware requirements

- Control plane nodes: At least 4 CPU cores, 16.00 GiB RAM, and 100 GB disk size.
- Workers: At least 2 CPU cores, 8.00 GiB RAM, and 100 GB disk size.

Install OpenShift with the Assisted Installer

- 1 Cluster details
- 2 Host discovery
- 3 Networking
- 4 Review and create

Cluster details

Cluster name *

Base domain *

 example.com

All DNS records must be subdomains of this base and include the cluster name. This cannot be changed after cluster installation. The full cluster address will be:
[Cluster Name].example.com

OpenShift version *

 OpenShift 4.10.24

Install single node OpenShift (SNO)
SNO enables you to install OpenShift using only one host.

Edit pull secret ⓘ

Use arm64 CPU architecture ⓘ
Make sure all the hosts are using arm64 CPU architecture.

Hosts' network configuration

DHCP server Static network configuration

Additional Requirements

- Enabled CPU virtualization support in BIOS (Intel-VT / AMD-V) on all nodes
- Each worker node requires an additional 360 MiB of memory and 2 CPUs
- Each control plane node requires an additional 150 MiB of memory and 4 CPUs
- OpenShift Data Foundation (recommended for full functionality) or another persistent storage service

Using the [ocp-configure](#) project to deploy

ocp-configure

This project can install OCP(openshift) in beaker.

How to use

```
→ ocp-configure git:(main) ./getocp.py
usage: getocp.py [-h] [-n {1,3,6} | -r DESTROY DESTROY]

THE OCP INSTALLITION TOOLS

optional arguments:
  -h, --help            show this help message and exit
  -n {1,3,6}, --install {1,3,6}
                        Install the number of ceph hosts
  -r DESTROY DESTROY, --destroy DESTROY DESTROY
                        Destroy the number of ceph hosts

Any questions, you can mail me <wji@redhat.com>
Install: ./getocp.py -n 3
Destroy: ./getocp.py -r yes-i-realy-realy-want-to-destroy-cluster <number>
```



AGENDA

What is OpenShift?

- Containers
- Container Native Virtualization / Virtual Machines
- How to install cluster with the [ocp-configure](#) tool?

OpenShift Storage!

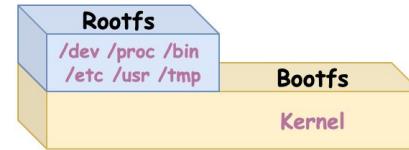
- How to start VM with hostpath / nfs / [ceph](#) storage?

OpenShift [Network](#)!

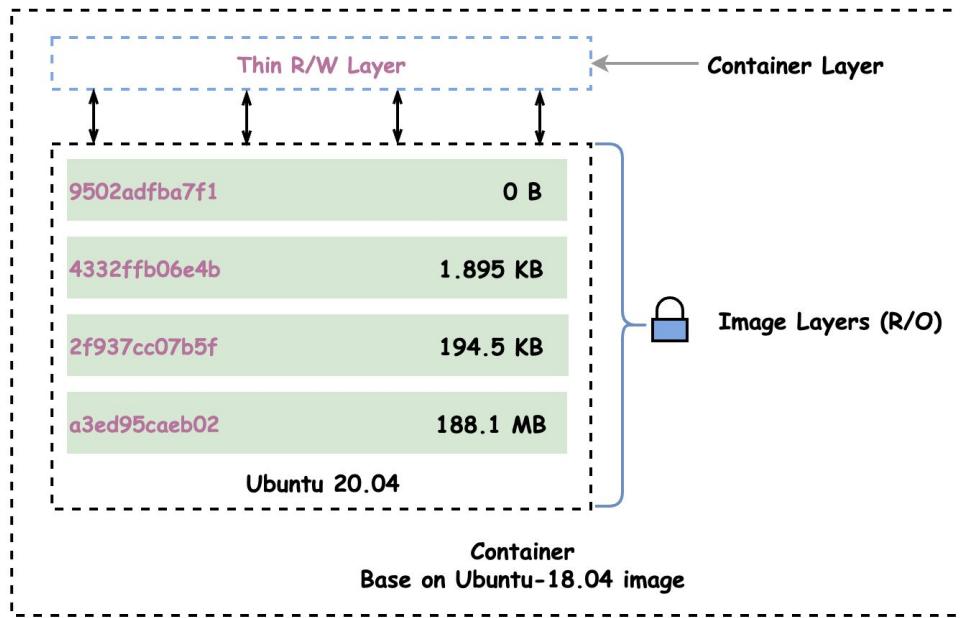
- What differents between SoftwareDefinedNetwork and OpenVirtualNetwork?



What is OpenShift?

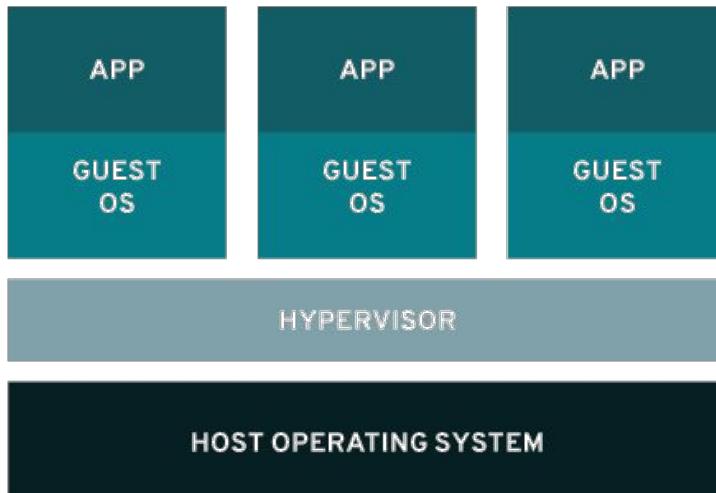


What is Containers?

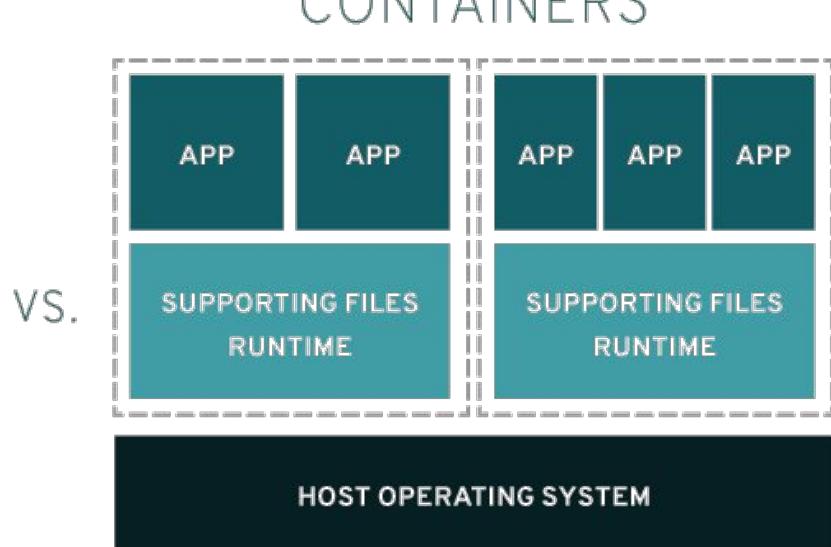


Containers vs VMs

VIRTUALIZATION



CONTAINERS



VS.

What is OpenShift?

Red Hat® OpenShift® is a [Kubernetes](#) distribution—a commercialized software product derived from an open source project. Red Hat OpenShift and Kubernetes are both [container orchestration](#) software, but Red Hat OpenShift is packaged as a downstream [enterprise open source](#) platform—meaning it's undergone additional testing and contains additional features not available from the Kubernetes open source project.

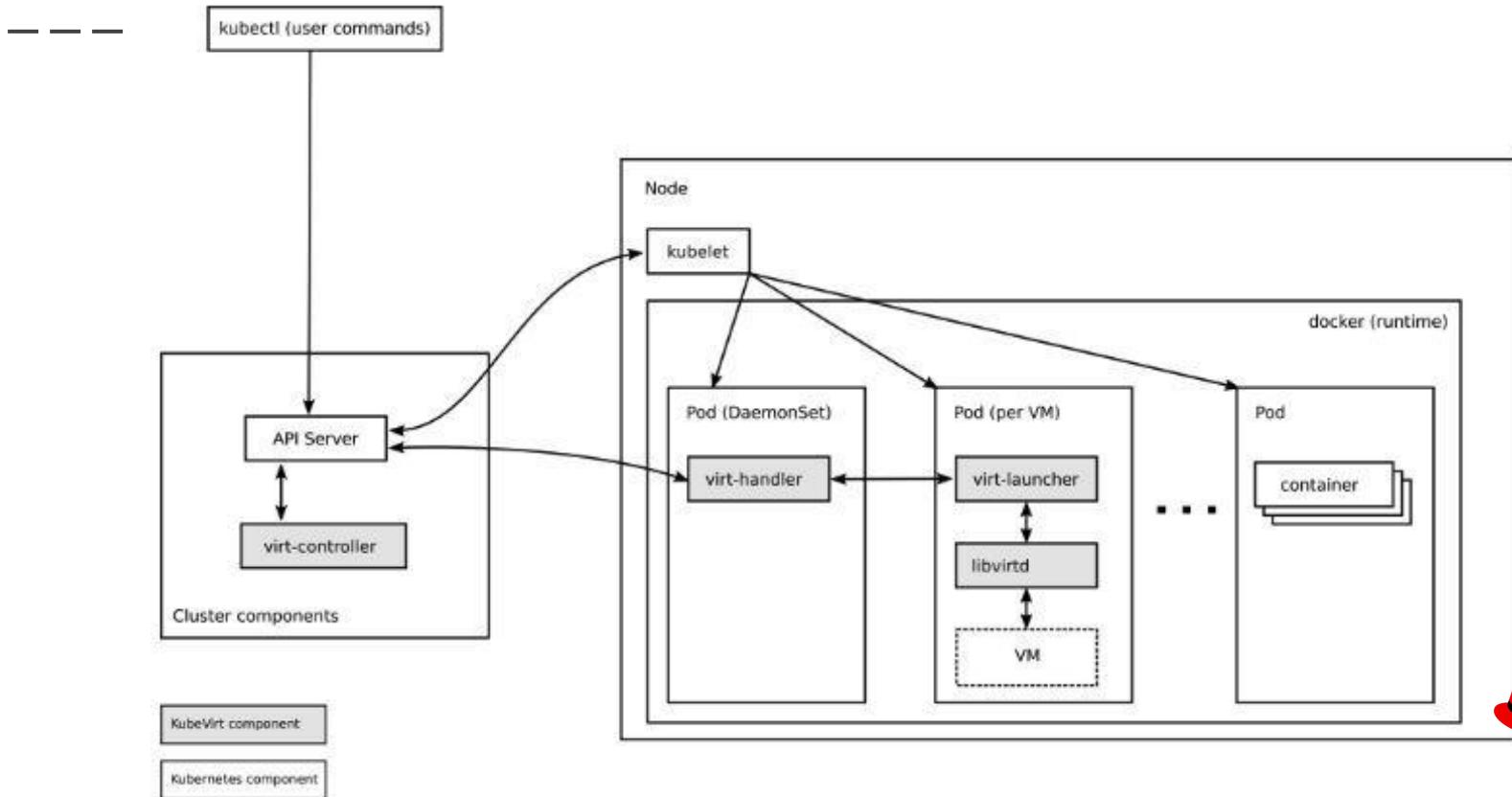
红帽 OpenShift 和 Kubernetes 都负责管理一组容器(称为[集群](#))。每个集群都分为 2 个部分:控制平面和工作节点。容器在工作节点中运行, 而每个工作节点都有自己的 [Linux 操作系统](#)。控制平面负责维护集群的整体状态(例如运行什么 应用以及使用哪些容器 镜像), 而工作节点则负责实际的计算工作。

尽管 Kubernetes 无所不能, 但用户仍然需要整合其他 组件, 例如网络、入口和负载平衡、存储、监控、日志记录等。在以 Kubernetes 为 核心的前提下, 红帽 OpenShift 之所以还要提供这些组件, 因为就在于[单靠 Kubernetes 是不够的](#)。

作为一个万能的容器平台, 红帽 OpenShift 绝不仅仅是一种软件产品。它是实施 [DevOps](#) 文化的关键所在 - 在应用的整个生命周期中, 实现日常运维任务的自动化和环境的标准化。



OpenShift Component!

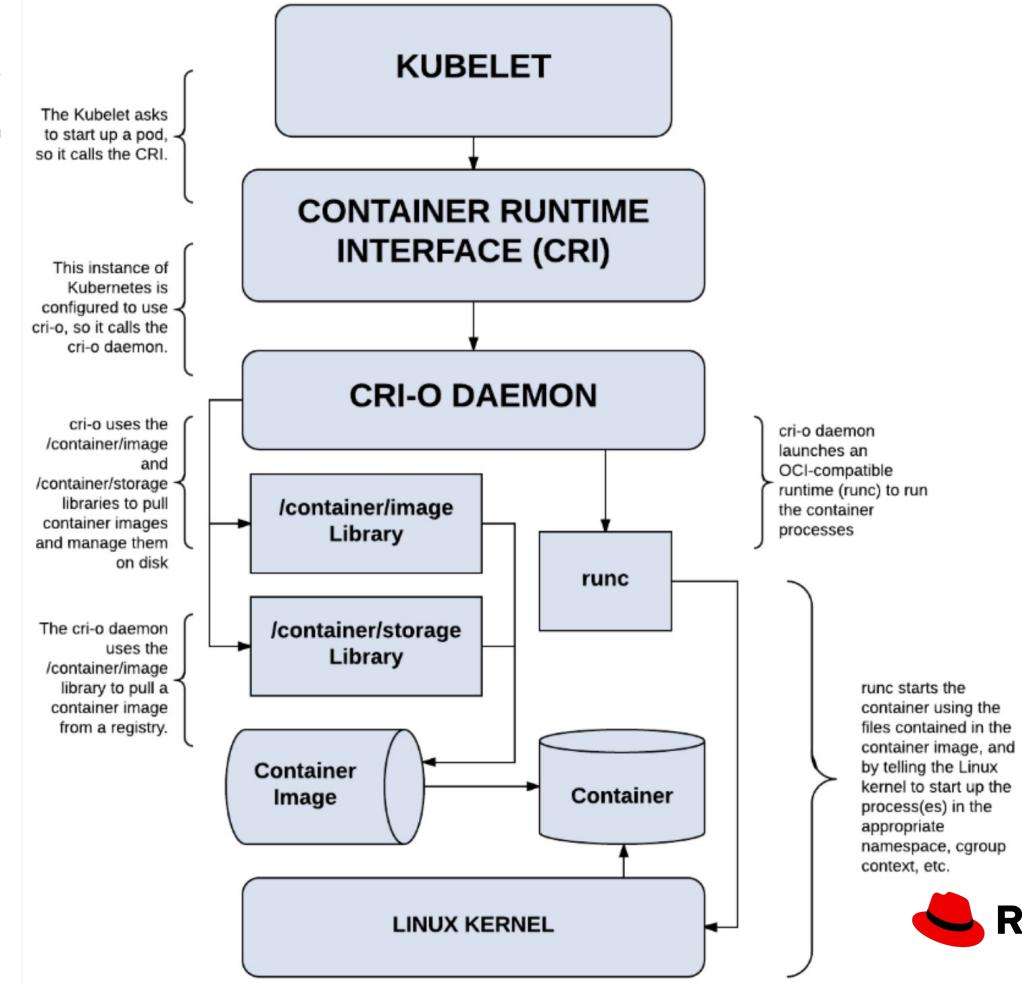


OpenShift Component!

———
CRI-O 是一个开源的、社区驱动的容器引擎。其主要目标是取代 Docker 服务作为 Kubernetes 实施的容器引擎，例如 OpenShift Container Platform。

Open Container Initiative OCI 于 2015 年 6 月 22 日由 Docker、CoreOS 和其他容器行业的领导者发起。OCI 目前包含两个规范：运行时规范 (runtime-spec) 和图像规范 (image-spec)。

runc 是一个 CLI 工具，用于根据 OCI 规范在 Linux 上生成和运行容器。



Virtualized control plane on OpenShift

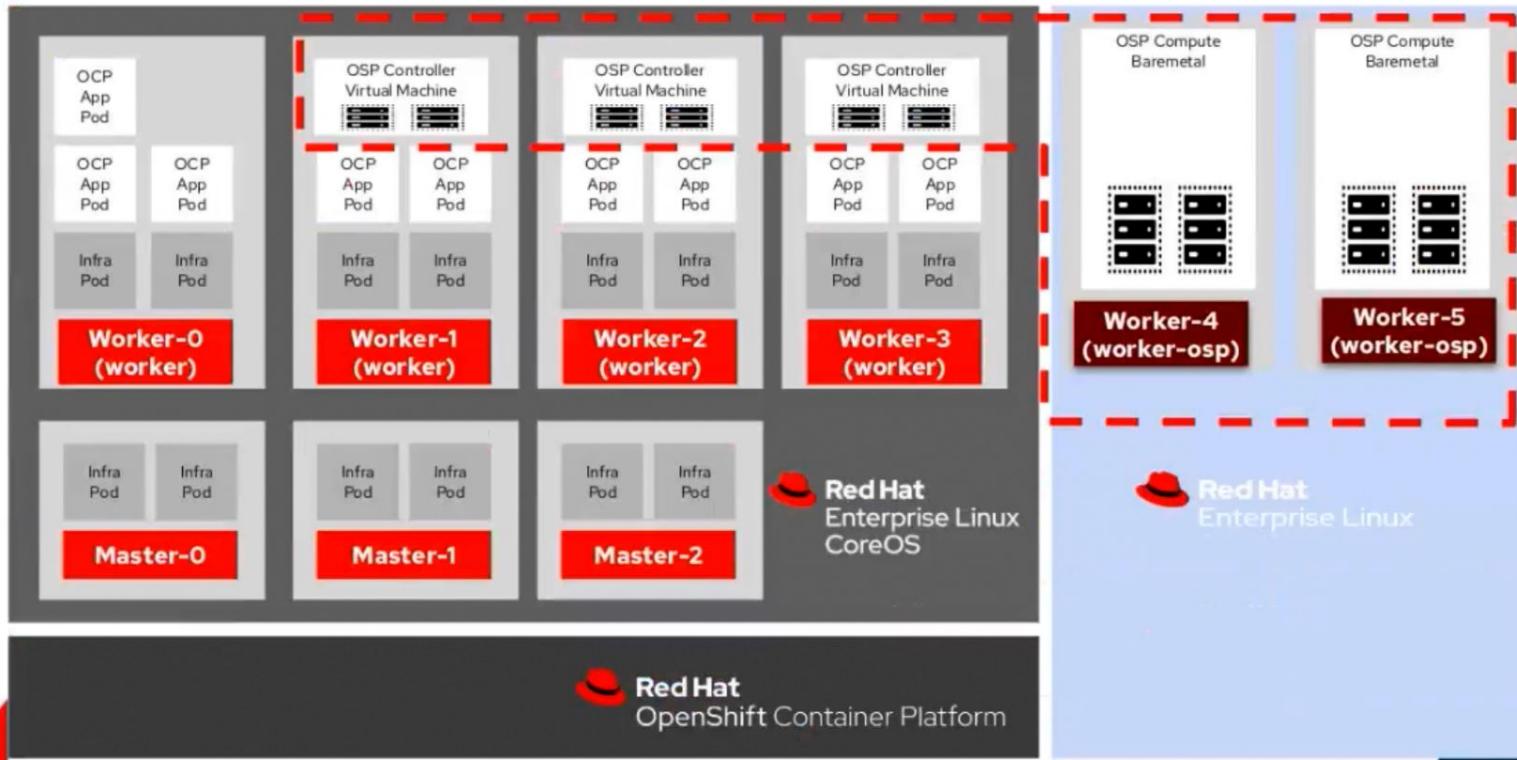
Virtualization

Tech Preview in RHOSP 16.2

CONFIDENTIAL



OpenStack Platform

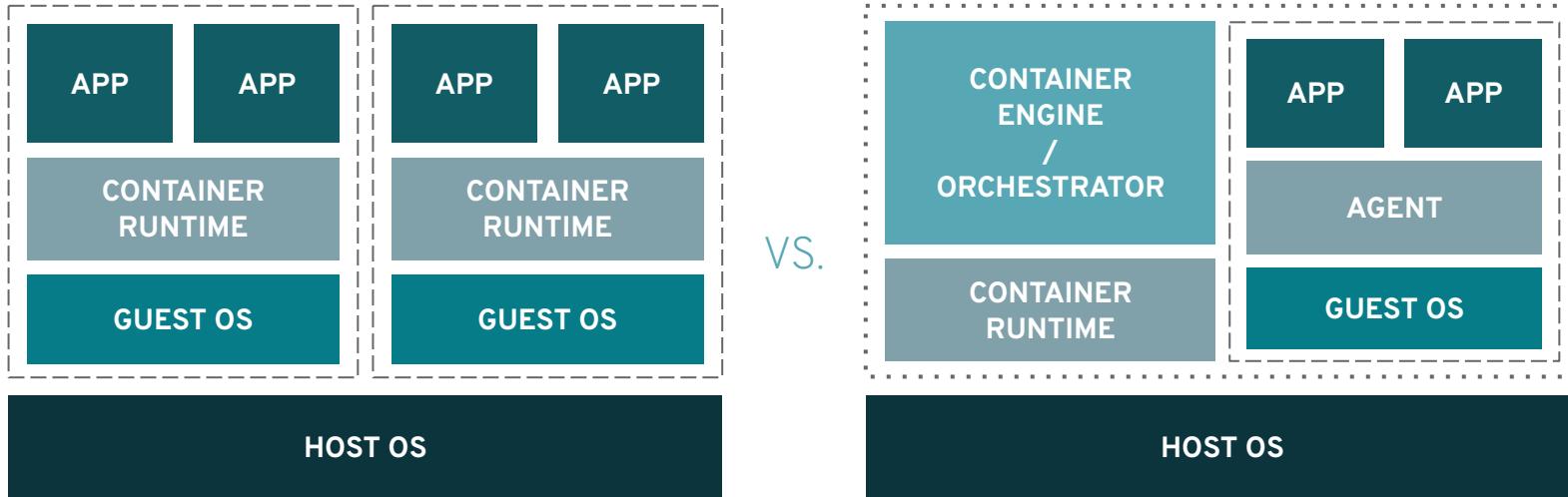


Red Hat

What is OpenShift? - CNV part

- Upstream: Kubevirt
 - <https://kubevirt.io/>
- Brief Introduction [KubeVirt 2 minutes Intro](#)
 - "We just built the qemu/libvirt daemen or relevant into a container to support running the VM on OpenShift env." from ycui@
 - "我们只是将 qemu / libvirt daemen 或相关内容构建到容器中, 以支持在 OpenShift env 上运行 VM 。" from ycui@

Containers + VMs vs VM-based Containers



What is OpenShift? - CNV part

OCP and CNV have something in common, they are both containers. And the difference is CNV, which creates a virtual machine inside a container. **The screenshot on the right shows the details.**

Since they are all containers, they can be managed by scheduling. The next page will show the process.

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar menu is visible, showing options like 'Pod', '部署', '部署配置', etc. The main content area shows a terminal session titled 'virt-launcher-wjiservices-kvnrt' with the status 'Running'. The terminal window displays the following command output:

```
sh-4.4# virsh list
 Id  Name          State
 ----
 1  default_wjiservices   running

sh-4.4# cd /etc/libvirt/qemu
sh-4.4# ls
default_wjiservices.xml
sh-4.4#
sh-4.4# cd /var/log/libvirt/qemu/
sh-4.4# ls
default_wjiservices.log
sh-4.4#
sh-4.4# ip addr show dev k6t-eth0
4: k6t-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc link/ether 02:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.1/24 brd 10.0.2.255 scope global k6t-eth0
        valid_lft forever preferred_lft forever
        inet6 fe80::ff:fe00:0/64 scope link
            valid_lft forever preferred_lft forever
sh-4.4# curl http://10.0.2.2
It works
sh-4.4# curl http://localhost
It works
sh-4.4#
```



```
/usr/bin/qemu-system-x86_64 \
-machine accel=kvm \
-name sandbox-6c2056050a9fa63d726711e01002f0497c2f9ad839fb424eb6e63f0b0c7e39fc \
-uuid c28e839a-26e3-425c-8c90-b86caece4d65 \
-machine q35,accel=kvm,kernel_irqchip \
-cpu host \
-qmp unix:/run/vc/vm/6c2056050a9fa63d726711e01002f0497c2f9ad839fb424eb6e63f0b0c7e39fc/qmp.sock,server,nowait \
-m 2048M,slots=10,maxmem=8630M \
-device pci-bridge,bus=pcie.0,id=pci-bridge-0,chassis_nr=1,shpc=on,addr=2,romfile= \
-device virtio-serial-pci,disable-modern=false,id=serial0,romfile= \
-device virtconsole,chardev=charconsole0,id=console0 \
-chardev socket,id=charconsole0,path=/run/vc/vm/6c2056050a9fa63d726711e01002f0497c2f9ad839fb424eb6e63f0b0c7e39fc/console.sock,server,nowait \
-device virtio-scsi-pci,id=scsi0,disable-modern=false,romfile= \
-object rng-random,id=rng0,filename=/dev/urandom \
-device virtio-rng-pci,rng=rng0,romfile= \
-device vhost-vsock-pci,disable-modern=false,vhostfd=3,id=vsock-2862916059,guest-cid=2862916059,romfile= \
-chardev socket,id=char-a9ef5c8cd2d0989c,path=/run/vc/vm/6c2056050a9fa63d726711e01002f0497c2f9ad839fb424eb6e63f0b0c7e39fc/vhost-fs.sock \
-device vhost-user-fs-pci,chardev=char-a9ef5c8cd2d0989c,tag=kataShared,romfile= \
-netdev tap,id=network-0,vhost=on,vhostfds=4,fds=5 \
-device driver=virtio-net-pci,netdev=network-0,mac=2e:7a:6b:f2:f6:74,disable-modern=false,mq=on,vectors=4,romfile= \
-global kvm-pit.lost_tick_policy=discard \
-vga none \
-no-user-config \
-nodefaults \
-nographic \
-daemonize \
-object memory-backend-file,id=dimm1,size=2048M,mem-path=/dev/shm,share=on \
-numa node,memdev=dimm1 \
-kernel /usr/lib/modules/5.7.11-200.fc32.x86_64/vmlinuz \
-initrd /var/cache/kata-containers/osbuilder-images/5.7.11-200.fc32.x86_64/fedora-kata-5.7.11-200.fc32.x86_64.initrd \
-append tsc=reliable no_timer_check rcupdate.rcu_expedited=1 i8042.direct=1 i8042.dumbkbd=1 i8042.nopnp=1 i8042.noaux=1 \
noreplace-smp reboot=k console=hvc0 console=hvc1 iommu=off cryptomgr.notests net.ifnames=0 pci=lastbus=0 quiet panic=1 nr_cpus=4 \
agent.use_vsock=true scsi_mod.scan=none \
-pidfile /run/vc/vm/6c2056050a9fa63d726711e01002f0497c2f9ad839fb424eb6e63f0b0c7e39fc.pid \
-smp 1,cores=1,threads=1,sockets=4,maxcpus=4
```

Monitor Port

Console Port

Agent Port

Pod Image

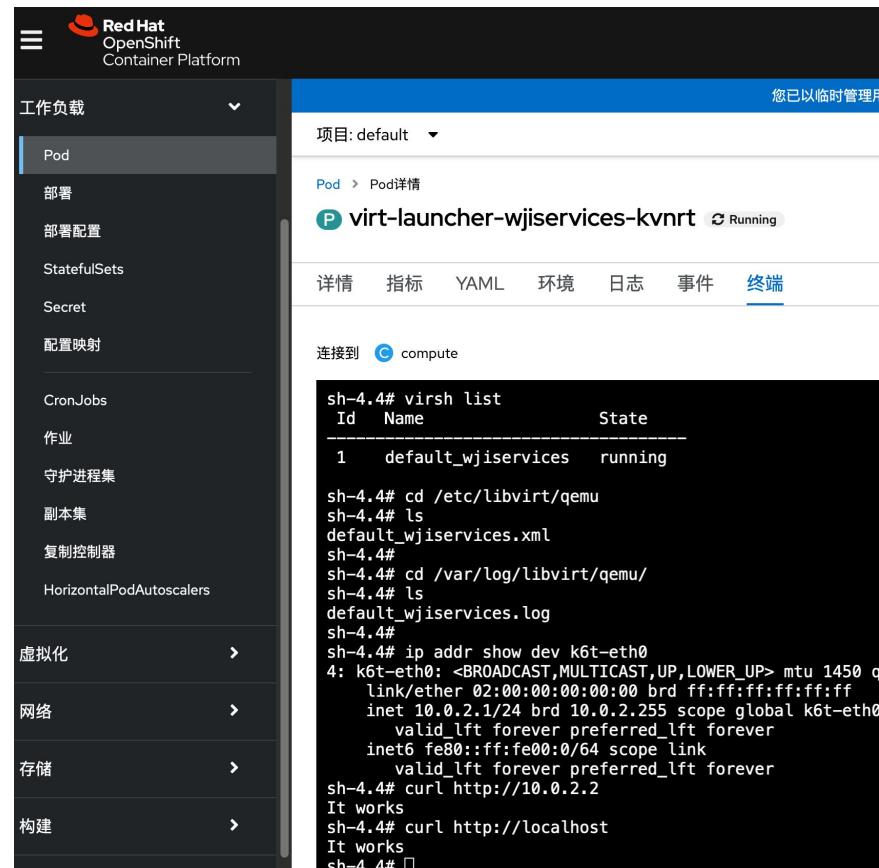
Pod Network

Guest OS

What is OpenShift? - CNV part

OCP and CNV have something in common, they are both containers. And the difference is CNV, which creates a virtual machine inside a container. The screenshot on the right shows the details.

Since they are all containers, they can be managed by scheduling. **The next page will show the process.**

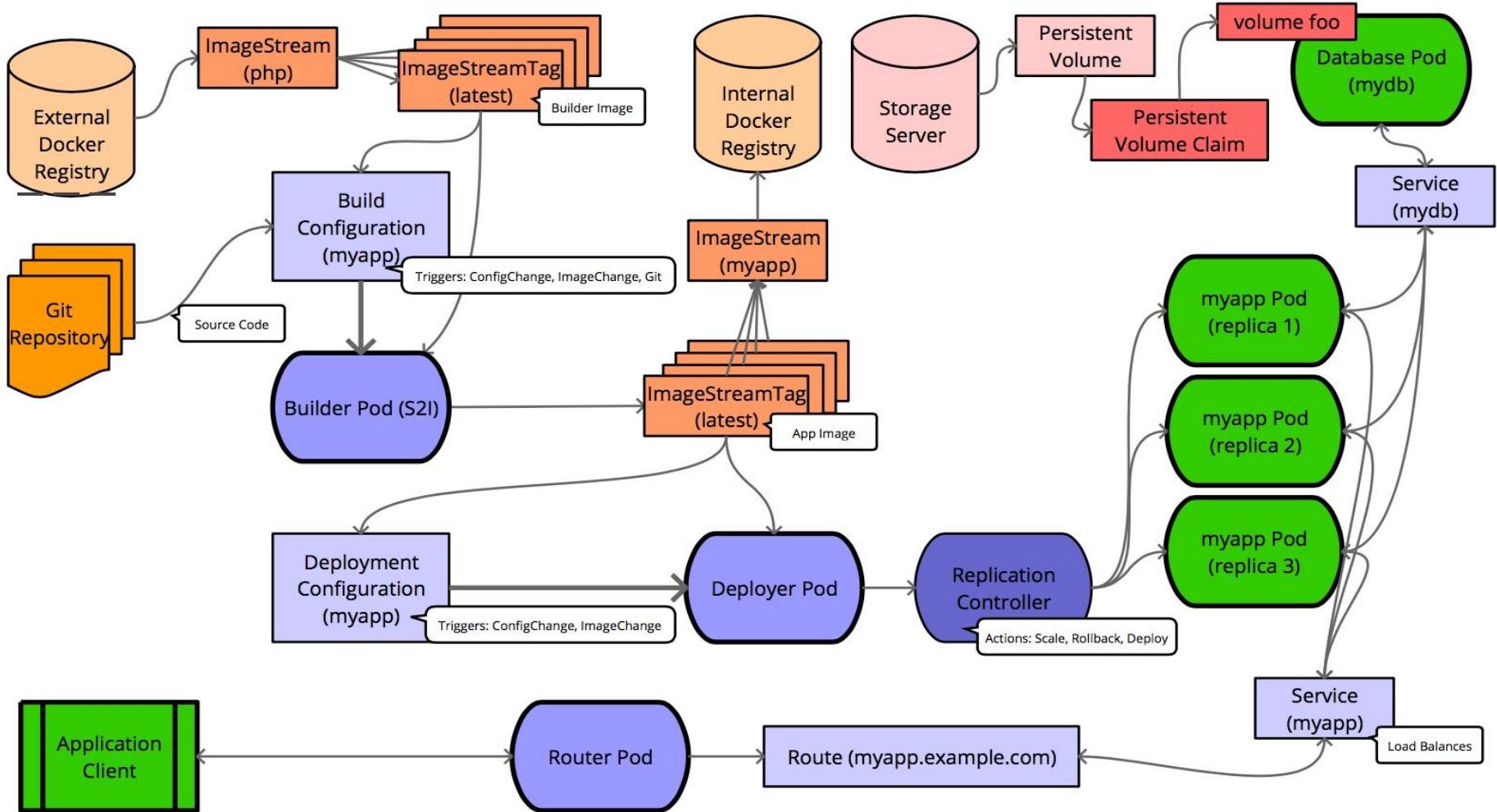


The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar menu is visible, showing options like 'Pod', '部署' (Deployment), '部署配置' (Deployment Configuration), 'StatefulSets', 'Secret', '配置映射' (ConfigMap), 'CronJobs', '作业' (Jobs), '守护进程集' (DaemonSets), '副本集' (ReplicaSets), '复制控制器' (ReplicaController), and 'HorizontalPodAutoscalers'. The main content area is titled 'Pod > Pod详情' and shows a pod named 'virt-launcher-wjiservices-kvnrt' in the 'default' project, status 'Running'. Below this, there are tabs for '详情' (Details), '指标' (Metrics), 'YAML', '环境' (Environment), '日志' (Logs), '事件' (Events), and '终端' (Terminal). A terminal window is open, connected to a node labeled 'compute'. The terminal output shows the user running 'virsh list' to see the running VM 'default_wjiservices', then navigating to '/etc/libvirt/qemu' and '/var/log/libvirt/qemu/' to check logs for 'default_wjiservices'. Finally, the user runs curl commands to test connectivity to port 10.0.2.2, with the response 'It works'.

```
sh-4.4# virsh list
Id  Name          State
1   default_wjiservices  running

sh-4.4# cd /etc/libvirt/qemu
sh-4.4# ls
default_wjiservices.xml
sh-4.4#
sh-4.4# cd /var/log/libvirt/qemu/
sh-4.4# ls
default_wjiservices.log
sh-4.4#
sh-4.4# ip addr show dev k6t-eth0
4: k6t-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc link/ether 02:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
    link/ether 02:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.1/24 brd 10.0.2.255 scope global k6t-eth0
        valid_lft forever preferred_lft forever
        inet6 fe80::ff:fe00:0/64 scope link
            valid_lft forever preferred_lft forever
sh-4.4# curl http://10.0.2.2
It works
sh-4.4# curl http://localhost
It works
sh-4.4#
```





CNV demo presentation

- I. Cluster Deployment
 - II. Cluster Login
 - III. Storage configuration
 - IV. Virtual Machine Management
 - V. Virtual machine more operations
- Practical part: service provisioning

OPENShift VIRTULIZATION

CONSOLE.REDHAT.COM

<https://gitlab.cee.redhat.com/wji/learningmd/-/blob/master/CNV/README.md>



OpenShift Storage!

OpenShift Storage! - pv storage

这里会展示三类存储：

- 手动创建的 PV：
nfs / iscsi / local disk / local folder
- 使用 Local Storage Operator 自动创建的 PV
local disk
- 使用 StorageClass 自动创建的 PV
nfs / iscsi / local / ceph / aws / azure / swift



OpenShift Storage! - pv storage - nfs

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: example
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: slow
  nfs:
    path: /tmp
    server: 172.17.0.2
```

OpenShift Storage! - pv storage - iscsi

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: iscsi-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  iscsi:
    targetPortal: 10.16.154.81:3260
    iqn: iqn.2014-12.example.server:storage.target00
    lun: 0
    fsType: 'ext4'
```

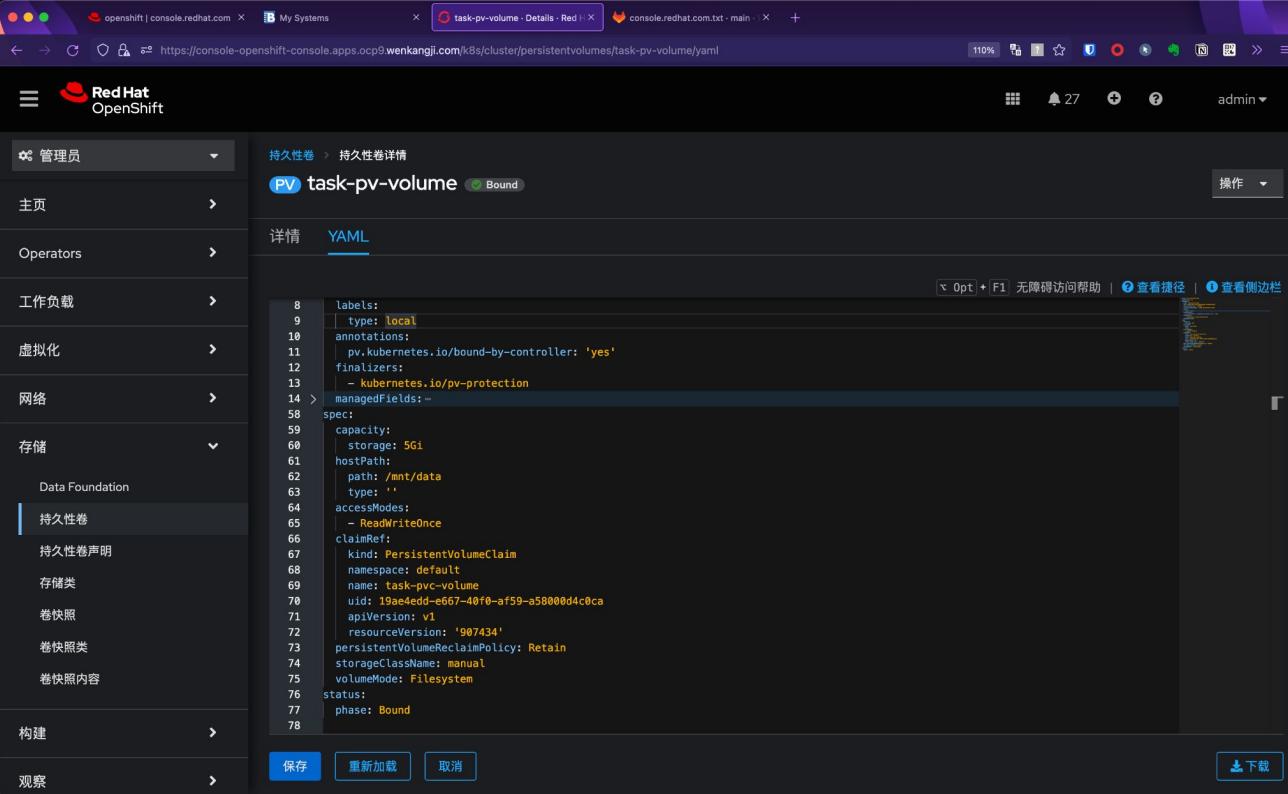


OpenShift Storage! - pv storage - local without LSO - howto

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: example-pv-filesystem
spec:
  capacity:
    storage: 100Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage
  local:
    path: /dev/xvdf
  nodeAffinity:
    required:
      nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/hostname
            operator: In
            values:
              - example-node
```



OpenShift Storage! - pv storage - local without LSO - effect



The screenshot shows the Red Hat OpenShift web console interface. The left sidebar is collapsed, and the main area displays the details of a PersistentVolume named "task-pv-volume". The PV is currently "Bound". The "YAML" tab is selected, showing the following configuration:

```
8  labels:
9    type: local
10   annotations:
11     pv.kubernetes.io/bound-by-controller: 'yes'
12   finalizers:
13     - kubernetes.io/pv-protection
14   managedFields:-
15
16 spec:
17   capacity:
18     storage: 5Gi
19   hostPath:
20     path: /mnt/data
21     type: ''
22   accessModes:
23     - ReadWriteOnce
24   claimRef:
25     kind: PersistentVolumeClaim
26     namespace: default
27     name: task-pvc-volume
28     uid: 19ae4edd-e667-4f10-af59-a58000d4c0ca
29     apiVersion: v1
30     resourceVersion: '907434'
31     persistentVolumeReclaimPolicy: Retain
32   storageClassName: manual
33   volumeMode: Filesystem
34   status:
35     phase: Bound
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
```

At the bottom of the screen, there are buttons for "保存" (Save), "重新加载" (Reload), "取消" (Cancel), and "下载" (Download).



OpenShift Storage! - Local Volume with LSO - GUI

The screenshot shows the OpenShift Local Storage Operator (LSO) interface. On the left, a dark sidebar menu includes '管理员' (Administrator), '主页' (Home), 'Operators' (selected), 'OperatorHub', '安装的 Operators' (selected), '工作负载' (Workloads), '虚拟化' (Virtualization), '概述' (Overview), '虚拟机器' (Virtual Machines), and '模板' (Templates). The main content area has a blue header bar with the message: '您已以临时管理用户身份登录。更新集群 OAuth 配置以允许其他人登录。' (You have logged in as a temporary management user. Update your cluster's OAuth configuration to allow others to log in.) Below this, it says '项目: openshift-local-storage' and '安装的 Operators > Operator 详情'. Under 'Local Storage' (由Red Hat提供的4.11.0-202208020235), there are tabs for '详情' (Details), 'YAML', '订阅' (Subscription), '事件' (Events), '所有实例' (All Instances), 'Local Volume' (selected), 'Local Volume Set', and 'Local Volume Discovery'. A large 'Local Volumes' section displays the message '未找到操作对象' (No operation objects found) and '操作对象是用来定义应用程序行为的声明性组件' (Operation objects are used to define application behavior declaratively). A blue button at the bottom right says '创建Local Volume' (Create Local Volume).



OpenShift Storage! - Local Volume with LSO - CLI

```
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
  name: local-block
  namespace: openshift-local-storage
spec:
  nodeSelector:
    nodeSelectorTerms:
      - matchExpressions:
          - key: cluster.ocs.openshift.io/openshift-storage
            operator: In
            values:
              - ""
  storageClassDevices:
    - storageClassName: localblock
  volumeMode: Block
  devicePaths:
    - /dev/disk/by-path/pci-0000:07:00.0
    - /dev/disk/by-path/pci-0000:07:00.0
    - /dev/disk/by-path/pci-0000:07:00.0
```



OpenShift Storage! - hostpath storageclass

您已以临时管理用户身份登录。更新集群 OAuth 配置以允许其他人登录。

项目: openshift-cnv ▾

安装的 Operators > Operator 详情

OpenShift Virtualization
由Red Hat提供的4.10.2

操作 ▾

详情 YAML 订阅 事件 所有实例 OpenShift Virtualization Deployment HostPathProvisioner deployment

HostPathProvisioners

创建HostPathProvisioner

名称	种类 (Kind)	状态	标签	最后更新
HPP hostpath-provisioner	HostPathProvisioner	Condition: Available	没有标签	2022年8月15日 14:06



HPP hostpath-provisioner

操作 ▾

详情

YAML

Opt + F1 无障碍访问帮助 | 查看捷径 | 查看侧边栏

```
1 apiVersion: hostpathprovisioner.kubevirt.io/v1beta1
2 kind: HostPathProvisioner
3 
4   metadata:
5     annotations:
6       kubectl.kubernetes.io/last-applied-configuration: >
7         {"apiVersion":"hostpathprovisioner.kubevirt.io/v1beta1","kind":"HostPathProvisioner","metadata":{"annotations":{},"name":"hostpath-provisioner"},"spec":{"imagePullPolicy":"IfNotPresent",
8       creationTimestamp: '2022-08-15T06:06:49Z'
9       finalizers:
10      - finalizer.delete.hostpath-provisioner
11      generation: 1835
12      managedFields:-
13        name: hostpath-provisioner
14        resourceVersion: '8865658'
15        uid: 43c7e97f-b8e7-45f4-80ba-2b7c86df9e95
16      spec:
17        imagePullPolicy: IfNotPresent
18        storagePools:
19          - name: sno
20            path: /var/hpvolumes
21            pvcTemplate:
22              accessModes:
23                - ReadWriteOnce
24              resources:
25                requests:
26                  storage: 50Gi
27              storageClassName: localblock-sc
28              volumeMode: Block
29            workload:
30              nodeSelector:
31                kubernetes.io/os: linux
32            status:
33              conditions:
34                - lastHeartbeatTime: '2022-08-19T05:44:00Z'
```

保存

重新加载

取消



OpenShift Storage! - nfs storageclass

```
# helm repo add nfs-subdir-external-provisioner https://kubernetes-sigs.github.io/nfs-subdir-external-provisioner/
# helm install nfs-subdir-external-provisioner nfs-subdir-external-provisioner/nfs-subdir-external-provisioner \
--set nfs.server=<THE NFS SERVER> \
--set nfs.path=<THE NFS EXPORTFS>
# oc patch storageclasses nfs-client -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

The screenshot shows the OpenShift Storage Classes interface. At the top, there's a navigation bar with icons for home, notifications (16), help, and user 'kube:admin'. Below the header is a blue banner with the message: '您已以临时管理用户身份登录。更新集群 OAuth 配置以允许其他人登录。' (You have logged in as a temporary management user. Update the cluster OAuth configuration to allow other users to log in.)

The main area is titled '存储类' (Storage Classes). It features a search bar with '名称' (Name) and '按名称搜索...' (Search by name...). A blue button labeled '创建存储类' (Create Storage Class) is located in the top right.

名称	置备程序	重新声明策略
SC nfs-storage – 默认	k8s-sigs.io/nfs-subdir-external-provisioner	Delete



OpenShift Storage! - nfs storageclass

存储类 > 存储类详情

SC nfs-storage

[详情](#) [YAML](#)

存储类详情

名称 nfs-storage	重新声明策略 Delete
标签 没有标签	编辑 编辑
注解 1注解	默认类 真
置备程序 k8s-sigs.io/nfs-subdir-external-provisioner	卷绑定模式 Immediate
创建于 2022年7月14日 16:18	
所有者 没有所有者	



OpenShift Storage! - ceph storageclass

What is ceph?

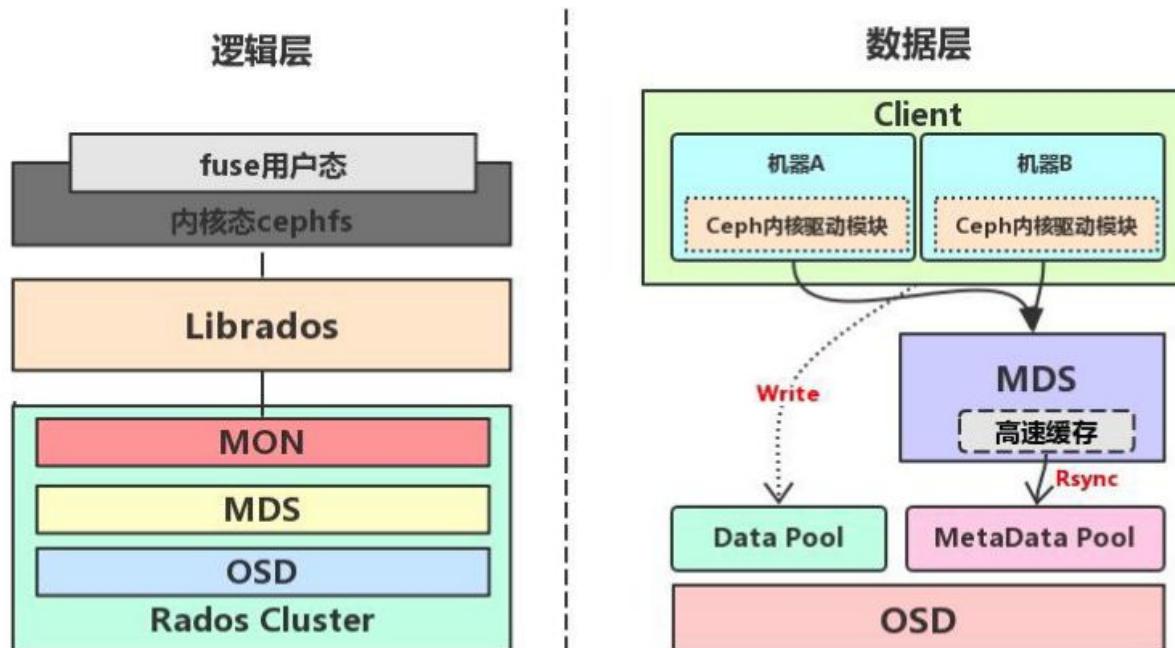
- A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

How to install ceph?

- Ceph can be installed both manually and automatically, and the automatic way is achieved through service specifications.

OpenShift Storage! - ceph storageclass

CephFS



Using the [ceph_configure_autogen](#) project to deploy

ceph_configure_autogen

The rhel87 is more recommended, rhel9 works but uses the rhel8 package. Only x86 platform is supported.

[ceph_configure_autogen installation guide](#)

```
pools: 1 pools, 32 pgs
objects: 0 objects, 0 B
usage: 0 B used, 0 B / 0 B avail
pgs: 100.000% pgs unknown
      32 unknown.

progress:
Updating grafana deployment (+1 -> 1) (0s)
[.....]
Global Recovery Event (0s)
[.....]

[ceph: root@ceph-node01 /]# echo 'cluster is loading, please wait...'
cluster is loading, please wait...
[ceph: root@ceph-node01 /]# ceph -s
cluster:
  id: fed4db16-f52e-11ec-981a-5254006c3c11
  health: HEALTH_OK

services:
  mon: 3 daemons, quorum ceph-node01,ceph-node02,ceph-node03 (age 88s)
    mgr: ceph-node01.vdjceu(active, since 3m), standbys: ceph-node02.hveokj, ceph-node03.lwbjqn
    osd: 3 osds: 3 up (since 17s), 3 in (since 31s)

data:
  pools: 2 pools, 33 pgs
  objects: 0 objects, 0 B
  usage: 15 MiB used, 30 GiB / 30 GiB avail
  pgs: 133 active+clean

Progress:
Updating prometheus deployment (+1 -> 1) (0s)
[.....]
```



ceph_configure_autogen installation guide

文件 编辑 查看 插入 格式 工具 扩展程序 帮助 上次修改是在 7 天前进行的

20% Arial

Step0: Check environment likes RAM & Disk

Step1: Install three nodes to deploy ceph storage

Step2: Register nodes to RHOCP subscribe

Step3: Preflight before installation using Ansible

Step4: Install ceph storage in ceph-node01

4-1: Generate /etc/hosts file to ceph-node01

4-2: Generate ceph.yaml to ceph-node01

4-3: Modify option/next.sh and install it to ceph-node01

4-4: Execute /home/next.sh in ceph-node01

Step5: Wait for coffee time and then check your cluster

Step6: Access ceph in our Physical host

Enjoy it!!!

Step0: Check environment likes RAM & Disk

```
[root@hpe-ml350gen9-01 ~]# hostname  
hpe-ml350gen9-01.hpe2.lab.eng.bos.redhat.com  
[root@hpe-ml350gen9-01 ~]# free -mh  
total used free shared buff/cache available  
Mem: 46Gi 726Mi 42Gi 12Mi 3.3Gi 45Gi  
Swap: 23Gi 0B 23Gi  
[root@hpe-ml350gen9-01 ~]#  
[root@hpe-ml350gen9-01 ~]#  
[root@hpe-ml350gen9-01 ~]# df -hT  
Filesystem Type Size Used Avail Use% Mounted on  
devtmpfs devtmpfs 24G 0 24G 0% /dev  
tmpfs tmpfs 24G 0 24G 0% /dev/shm  
tmpfs tmpfs 24G 9.7M 24G 1% /run  
tmpfs tmpfs 24G 0 24G 0% /sys/fs/cgroup  
/dev/mapper/rhel_hpe-ml350gen9-01-root xfs 70G 4.9G 66G 7% /  
/dev/sda1 xfs 1004M 217M 788M 22% /boot  
/dev/mapper/rhel_hpe-ml350gen9-01-home xfs 464G 3.8G 461G 1% /home  
tmpfs tmpfs 4.7G 0 4.7G 0% /run/user/0  
[root@hpe-ml350gen9-01 ~]#  
[root@hpe-ml350gen9-01 ~]#
```



OpenShift Storage! - ceph storageclass

The screenshot shows the Red Hat OpenShift web interface. The left sidebar has a navigation menu with the following items:

- 管理员
- 主页
- Operators
- 工作负载
- 虚拟化
- 网络
- 存储 → Data Foundation (highlighted with a red box)
- 构建
- 观察
- 计算
- 用户管理
- 管理

The main content area shows a blue header bar with the text "您已以临时管理用户身份登录。更新集群 OAuth 配置以允许其他人登录。" and the project name "openshift-storage". Below this, the title "创建存储系统" is displayed, followed by the sub-instruction "创建存储系统以代表您的 OpenShift Data Foundation 系统及其所有必要的存储和计算资源。".

The main content area is divided into two columns:

- 左侧列 (步骤):**
 - 1 后端存储
 - 2 连接详情 (highlighted with a blue circle)
 - 3 查看并创建
- 右侧列 (连接详情):**

外部存储系统元数据
上传帮助程序脚本

At the bottom of the main content area, there is a note: "下载 `ceph-external-cluster-details-exporter.py` 脚本并在 RHCS 集群上运行, 然后在外部存储系统元数据字段上传结果 (JSON)。" followed by a download link "下载脚本".



OpenShift Storage! - ceph storageclass

ocp 与 ceph 的联动, 后续需要配置的有:

1、下载 ceph-external-cluster-details-exporter.py 文件

2、创建存储池

```
[root@ceph-node01 ~]# ceph osd pool create myPool 32
pool 'myPool' created
[root@ceph-node01 ~]# rbd pool init myPool
[root@ceph-node01 ~]# ceph osd pool application enable myPool rbd
```

3、生成配置文件

```
[root@ceph-node01 ~]# python3 ceph-external-cluster-details-exporter.py --upgrade
[root@ceph-node01 ~]# python3 ceph-external-cluster-details-exporter.py --rbd-data-pool-name myPool
[{"name": "rook-ceph-mon-endpoints", "kind": "ConfigMap", "data": {"data": "ceph-node01=10.16.217.232:6789"...}}]
```

4、上传到 ocp 上进行调用

略



OpenShift Storage! - ceph storageclass

The image shows two side-by-side screenshots of the Red Hat OpenShift Container Platform web console.

Left Screenshot: The user is viewing the details of an **OCS External StorageCluster**. The cluster is named **ocs-external-storagecluster** and is managed by the **ocs-external-storagecluster-storagesystem**. The status is **Ready**. The interface includes tabs for **详情**, **YAML**, **资源**, and **事件**. The **StorageCluster 概述** section shows the name, namespace (**openshift-storage**), and other metadata like annotations and labels. The sidebar on the left lists various OpenShift components and operators.

Right Screenshot: The user is viewing the details of an **ocs-external-storagecluster-storagesystem**. The system is named **ocs-external-storagecluster-storagesystem** and is an **Object** type. It is currently an **对象服务** and has **数据弹性**. The **容量分解** section indicates there is no available usage data. The **性能** section shows a chart with a single data point at approximately 1.5. The sidebar on the left lists various storage-related components and systems.



OpenShift Storage! - more storageclass

The screenshot shows the OpenShift Storage interface. On the left, there's a sidebar with the following navigation items:

- 存储
- Data Foundation
- 持久性卷
- 持久性卷声明
- 存储类** (highlighted)
- 卷快照
- 卷快照类
- 卷快照内容

Below the sidebar, the main content area has a title "置备程序 *". A dropdown menu is open, showing the following options:

- 选择置备程序
- Select Provisioner
- kubernetes.io/no-provisioner
- kubernetes.io/aws-ebs
- kubernetes.io/gce-pd
- kubernetes.io/glusterfs
- kubernetes.io/cinder
- kubernetes.io/azure-file
- kubernetes.io/azure-disk
- kubernetes.io/quobyte
- kubernetes.io/vsphere-volume
- kubernetes.io/portworx-volume
- kubernetes.io/scaleio
- kubernetes.io/storageos
- kubevirt.io/hostpath-provisioner



OpenShift Network!

Technology Highlights Comparison

OpenShift SDN	OVN Kubernetes
veth pairs	veth pairs
OVS bridge	OVS bridge
Central controller / host-ipam	Central controller / host-ipam
VXLAN tunnels	Geneve tunnels
OVS flows for NetworkPolicy	OVS flows for NetworkPolicy
IPTables for services	OVN LBs for services
IPTables for NAT	OVS for NAT



OpenShift Network!

What different between sdn, ovn and ovs?

ovs 是 open vswitch 的缩写, 可以通过备注里的方法编译一个 ovs 出来。

sdn 是单纯用 ovs 搭建起来的网络, 需要配合其他组件比如 dhcp 等配合使用。

ovn 是使用的 ovn 来搭建的网络, ovn 下层用的是 ovs 但是它包含了很多功能, 比如 dhcp 这样就不需要其他组件了。

What is controller and how ovs works?

sdn 使用控制器比如 openstack 的 opendaylight 来控制 ovs 网络。

ovn 里边的控制器比如 openshift-network-operate 给 ovs 创建网络。

What connection between OVN and Geneve tunnel?

geneve本身跟功能关系不大, 它的作用是用来 传输, 连接不同 ovn 节点。而隧道使用 Geneve的网络性能会高于VxLAN。

不同 ovn 节点上的数据要通信的话就得通过 geneve, 同一个 ovn 节点上的数据不需要走 geneve

What is flow table betwwen OVN and SDN?

这个 flow table 是 OVS 的核心, 并且比传统的交换机具有更强大的功能。

OpenShift Network! - Try it!

```
root@edge-qe-per740-01:~/dotfiles
...qe-per740-01:~/dotfiles (ssh) #1 ...t@edge-qe-per740-01:~ (ssh) #2 ...t@edge-qe-per740-01:~ (ssh) #3 ...t@edge-qe-per740-01:~ (ssh) #4 ...t@edge-qe-per740-01:~ (ssh) #5 ...
age.

Starting install...

Domain is still running. Installation may be in progress.
Waiting 1 minute for the installation to complete.
^CDomain install interrupted.
Installation aborted at user request
[root@edge-qe-per740-01 dotfiles]# ovs-vsctl show
56088842-2b6c-4856-800b-bb0cdxfc4f2d3
    Bridge helloworld
        Port helloworld
            Interface helloworld
                type: internal
        Port "vnet3"
            Interface "vnet3"
        Port "vnet1"
            Interface "vnet1"
        Port "vnet2"
            Interface "vnet2"
        Port "vnet0"
            Interface "vnet0"
    ovs_version: "2.7.14"
[root@edge-qe-per740-01 dotfiles]# virsh list
   Id   Name      State
-----
  1   ovs-node01  running
  2   ovs-node02  running
  3   ovs-node03  running
  4   ovs-node04  running
[root@edge-qe-per740-01 dotfiles]# bash testvm.sh 4 helloworld
```



OpenShift Network! SDN part

默认情况下, Docker 网络使用仅使用主机虚机网桥 bridge, 主机内的所有容器都连接至该网桥。

连接到此桥的所有容器都可以彼此通信, 但不能与不同主机上的容器通信。

为了支持跨集群的容器之间的通信, OpenShift 容器平台使用了软件定义的网络(SDN)方法。

软件定义的网络是一种网络模型, 它通过**几个网络层的抽象来管理网络服务**。

SDN 将**处理流量的软件**(称为控制平面)和**路由流量的底层机制**(称为数据平面)解耦。SDN 支持**控制平面和数据平面**之间的通信。

cluster network 由 OpenShift SDN 建立和维护, 它使用 Open vSwitch 创建 overlay 网络, master 节点不能通过集群网络访问容器, 除非 master 同时也为 node 节点。



OpenShift Network! SDN part - plugin

在 OpenShift 中, 可以为 pod 网络配置三个 SDN 插件:

- ovs-subnet: 默认插件, 子网提供了一个 flat pod 网络, 其中每个 pod 可以与其他 pod 和 service 通信。
- ovs-multitenant: 该为 pod 和服务提供了额外的隔离层。当使用此插件时, 每个 project 接收一个唯一的虚拟网络 ID (VLANID), 该 ID 标识来自属于该 project 的 pod 的流量。通过使用 VLANID, 来自不同 project 的 pod 不能与其他 project 的 pod 和 service 通信。
- ovs-network policy: 此插件允许管理员使用 NetworkPolicy 对象定义自己的隔离策略。

OpenShift Network! SDN part - plugin - ovs-subnet

在 OpenShift 中, 可以为 pod 网络配置三个 SDN 插件:

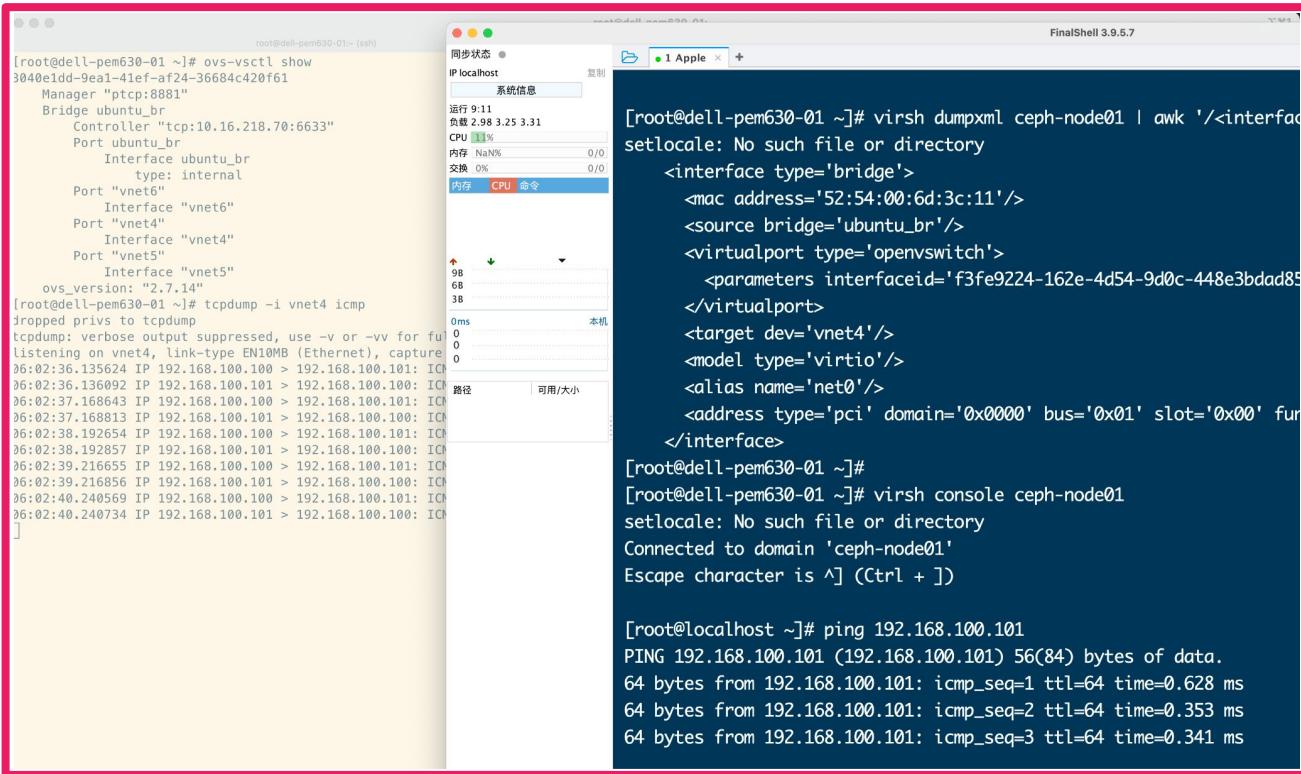
- ovs-subnet: 默认插件, 子网提供了一个 flat pod 网络, 其中每个 pod 可以与其他 pod 和 service 通信。

```
[root@dell-pem630-01 ~]# ovs-vsctl add-br br
[root@dell-pem630-01 ~]# virsh edit <VMnode>
<interface type='bridge'>
    <source bridge='br'/>
    <virtualport type='openvswitch'>
        <parameters
interfaceid='f3fe9224-162e-4d54-9d0c-448e3bdad851'/'>
        </virtualport>
    </interface>
```

```
[root@dell-pem630-01 ~]# ovs-vsctl show
3040e1dd-9ea1-41ef-af24-36684c420f61
    Bridge br
        Port br
        Interface br
            type: internal
        Port "vnet6"
        Interface "vnet6"
        Port "vnet4"
        Interface "vnet4"
        Port "vnet5"
        Interface "vnet5"
    ovs_version: "2.7.14"
```



OpenShift Network! SDN part - plugin - ovs-subnet



The screenshot shows a terminal window with several tabs open. The main tab displays command-line output for network configuration and monitoring:

```
[root@dell-pem630-01 ~]# ovs-vsctl show
3040e1dd-9ea1-41ef-af24-36684c420f61
  Manager "ptcp:8881"
  Bridge ubuntu_br
    Controller "tcp:10.16.218.70:6633"
    Port ubuntu_br
      Interface ubuntu_br
        type: internal
    Port vnet6
      Interface vnet6
    Port "vnet4"
      Interface "vnet4"
    Port "vnet5"
      Interface "vnet5"
  ovs_version: "2.7.14"
[root@dell-pem630-01 ~]# tcpdump -i vnet4 icmp
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full
listening on vnet4, link-type EN10MB (Ethernet), capture
06:02:36.135624 IP 192.168.100.100 > 192.168.100.101: ICMP
06:02:36.136092 IP 192.168.100.101 > 192.168.100.100: ICMP
06:02:37.168643 IP 192.168.100.100 > 192.168.100.101: ICMP
06:02:37.168813 IP 192.168.100.101 > 192.168.100.100: ICMP
06:02:38.192654 IP 192.168.100.100 > 192.168.100.101: ICMP
06:02:38.192857 IP 192.168.100.101 > 192.168.100.100: ICMP
06:02:39.216655 IP 192.168.100.100 > 192.168.100.101: ICMP
06:02:39.216856 IP 192.168.100.101 > 192.168.100.100: ICMP
06:02:40.240569 IP 192.168.100.100 > 192.168.100.101: ICMP
06:02:40.240734 IP 192.168.100.101 > 192.168.100.100: ICMP
]

[...]
```

Below the terminal, a system monitor window is visible, showing CPU usage at 11% and memory usage at 3B.

The right side of the screen shows a terminal window titled "FinalShell 3.9.5.7" with the following commands and outputs:

```
[root@dell-pem630-01 ~]# virsh dumpxml ceph-node01 | awk '/<interface>'
setlocale: No such file or directory
<interface type='bridge'>
  <mac address='52:54:00:6d:3c:11' />
  <source bridge='ubuntu_br' />
  <virtualport type='openvswitch'>
    <parameters interfaceid='f3fe9224-162e-4d54-9d0c-448e3bdad85' />
  </virtualport>
  <target dev='vnet4' />
  <model type='virtio' />
  <alias name='net0' />
  <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0' />
</interface>
[root@dell-pem630-01 ~]#
[root@dell-pem630-01 ~]# virsh console ceph-node01
setlocale: No such file or directory
Connected to domain 'ceph-node01'
Escape character is ^] (Ctrl + ])

[root@localhost ~]# ping 192.168.100.101
PING 192.168.100.101 (192.168.100.101) 56(84) bytes of data.
64 bytes from 192.168.100.101: icmp_seq=1 ttl=64 time=0.628 ms
64 bytes from 192.168.100.101: icmp_seq=2 ttl=64 time=0.353 ms
64 bytes from 192.168.100.101: icmp_seq=3 ttl=64 time=0.341 ms
```



OpenShift Network! SDN part - plugin - ovs-multitenant

在 OpenShift 中, 可以为 pod 网络配置三个 SDN 插件:

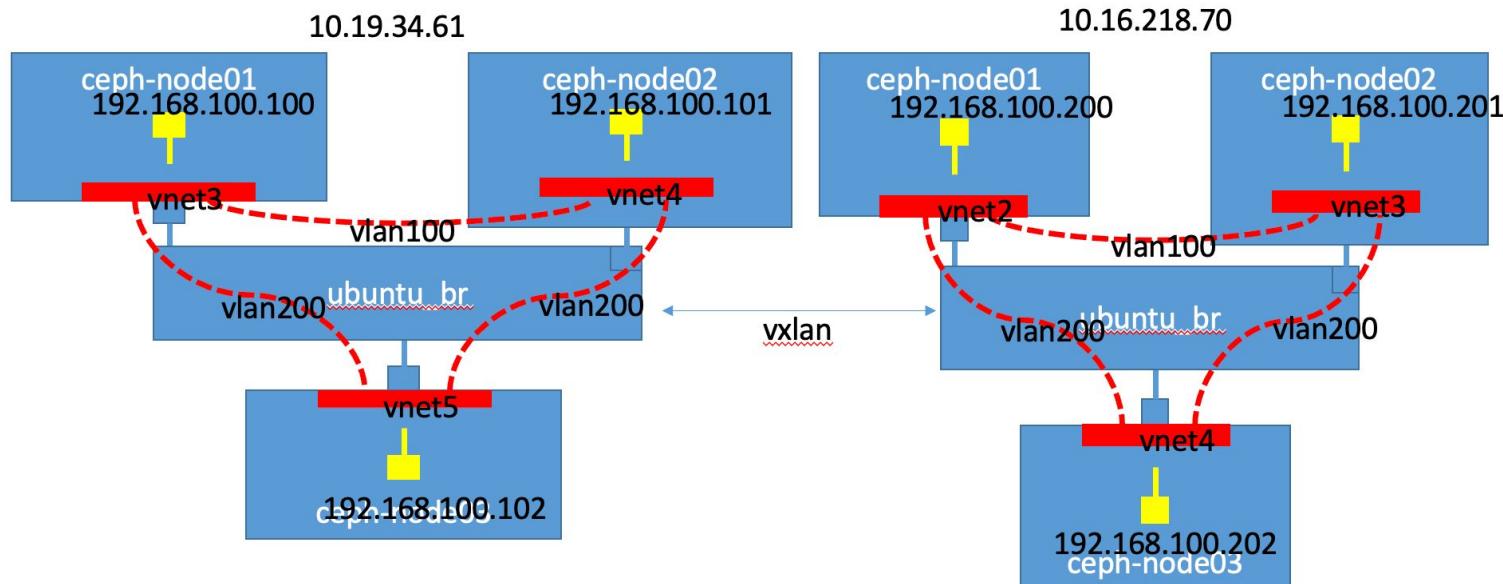
- ovs-multitenant: 该为 pod 和服务提供了额外的隔离层。当使用此插件时, 每个 project 接收一个惟一的虚拟网络 ID (VLANID), 该 ID 标识来自属于该 project 的 pod 的流量。通过使用 VLANID, 来自不同 project 的 pod 不能与其他 project 的 pod 和 service 通信。

```
/usr/share/openvswitch/scripts/ovs-ctl start  
ovs-vsctl add-br ubuntu_br  
ovs-vsctl set Port vnet2 tag=100  
ovs-vsctl set Port vnet3 tag=100  
ovs-vsctl set Port vnet4 tag=200
```

```
ovs-vsctl add-port ubuntu_br \  
    vxlan -- set interface vxlan type=vxlan options:remote_ip=10.19.34.61 option:key=1  
ovs-vsctl set Port vxlan trunks=100,200
```



OpenShift Network! SDN part - plugin - ovs-multitenant



OpenShift Network! SDN part - plugin - ovs-multitenant

```
[root@dell-pem630-01 ~]# ovs-vsctl show  
64b60993-3196-4278-aeae-07e6c3479377  
  Bridge ubuntu_br  
    Port "vnet4"  
      tag: 200  
      Interface "vnet4"  
    Port ubuntu_br  
      Interface ubuntu_br  
        type: internal  
    Port "vnet3"  
      tag: 100  
      Interface "vnet3"  
    Port vxlan  
      trunks: [100, 200]  
      Interface vxlan  
        type: vxlan  
        options: {key="1", remote_ip="10.19.34.61"}  
    Port "vnet2"  
      tag: 100  
      Interface "vnet2"  
  ovs_version: "2.7.14"  
[root@dell-pem630-01 ~]# ovs-ofctl dump-flows ubuntu_br
```

```
[root@cloud-qe-20 ~]# ovs-vsctl show  
0795f011-6ff9-49c4-876a-d79808e44a03  
  Bridge ubuntu_br  
    Port vxlan  
      Interface vxlan  
        type: vxlan  
        options: {key="1", remote_ip="10.16.218.70"}  
    Port "vnet3"  
      tag: 100  
      Interface "vnet3"  
    Port "vnet5"  
      tag: 200  
      Interface "vnet5"  
    Port "vnet4"  
      tag: 100  
      Interface "vnet4"  
    Port ubuntu_br  
      Interface ubuntu_br  
        type: internal  
  ovs_version: "2.7.14"  
[root@cloud-qe-20 ~]# ovs-ofctl dump-flows ubuntu_br
```



OpenShift Network! SDN part - plugin - ovs-network policy

在 OpenShift 中, 可以为 pod 网络配置三个 SDN 插件:

- ovs-network policy: 此插件允许管理员使用 NetworkPolicy 对象定义自己的隔离策略。



OpenShift Network! SDN part - plugin - ovs-network policy

The screenshot shows the Red Hat OpenShift Container Platform interface. The left sidebar has sections like '概述', '项目', '搜索', 'API Explorer', '事件', 'Operators', '工作负载', '虚拟化', '网络' (selected), '服务', '路由', 'Ingresses', and '网络策略'. The main area is titled '创建网络策略' (Create Network Policy) and shows a YAML editor with the following content:

```
1 apiVersion: networking.k8s.io/v1
2 kind: NetworkPolicy
3 metadata:
4   name: ''
5   namespace: default
6 spec:
7   podSelector: {}
8   policyTypes: []
```

The screenshot shows the configuration page for a 'my-policy' network policy in the 'default' project. It includes fields for '策略名称' (Name), 'Pod 选择器' (Pod Selector), '策略类型' (Type), and sections for 'Ingress' and 'Egress' rules.

策略名称 *: my-policy

Pod 选择器: 如果没有提供 pod 选择器，策略将应用到命名空间中的所有 pod。
编辑 pod 选择器 显示此策略要应用到的受影响 pod 的预览

策略类型: 选择默认入口和出口拒绝规则
 拒绝所有 ingress 流量 拒绝所有 egress 流量

Ingress: 删除所有 添加 ingress 规则
添加要应用到所选 pod 的 ingress 规则。如果网络流量数据至少匹配一条规则，则允许来自 pod 的流量。

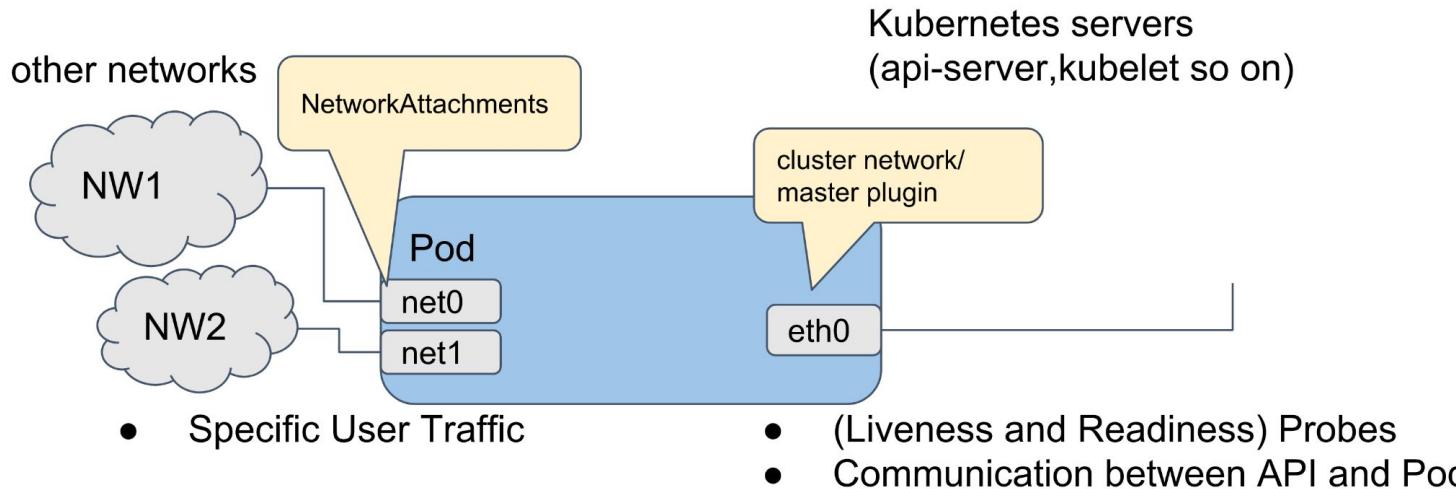
Egress: 删除所有 添加 egress 规则
添加要应用到所选 pod 的 egress 规则。如果网络流量数据至少匹配一条规则，则允许到 pod 的流量。



OpenShift Network! Multus CNI plugin

在 Kubernetes 中，容器联网由实现了 Container Network Interface (CNI) 的网络插件负责。

OpenShift Container Platform 使用 Multus CNI 插件来实现对 CNI 插件的链接。在集群安装过程中，您要配置 default pod 网络。默认网络处理集群中的所有一般网络流量。您可以基于可用的 CNI 插件定义额外网络，并将一个或多个这种网络附加到 pod。您可以根据需要为集群定义多个额外网络。这可让您灵活地配置提供交 换或路由等网络功能的 pod。



The screenshot shows a Red Hat OpenShift Container Platform interface on the left and a CNI documentation page on the right.

Left Side (OpenShift UI):

- Header:** Red Hat OpenShift Container Platform
- Navigation:** 管理员 (Administrator), 主页 (Home), 概述 (Overview), 项目 (Projects), 搜索 (Search), API Explorer, 事件 (Events), Operators, 工作负载 (Workloads), 虚拟化 (Virtualization), 网络 (Network), 服务 (Services), 路由 (Routes), Ingresses, 网络策略 (Network Policies), NetworkAttachmentDefinitions (highlighted).
- Content:** Create Network Attachment Definition dialog. It contains a YAML editor with the following code:

```
1 apiVersion: k8s.cni.cncf.io/v1
2 kind: NetworkAttachmentDefinition
3 metadata:
4   name: example
5   namespace: default
6 spec:
7   config: {}
```

A red arrow points from the "config: {}" line to the "Example configuration" section on the right.

Right Side (CNI Documentation):

- Header:** CNI | Home Documentation Plugins
- Section:** static IP address management plugin
- List:** Main, vlan plugin, bridge plugin, host-device, ipvlan plugin, macvlan plugin, ptp plugin, win-bridge plugin, win-overlay plugin, meta, bandwidth plugin, firewall plugin, Port-mapping plugin, Source based routing plugin, tuning plugin, vrf plugin.
- Section:** On this page: Overview, Example configuration, Example L2-only configuration, Network configuration reference, Interface configuration arguments reference.
- Section:** Example configuration
- Code:**

```
{  
  "cniVersion": "0.3.1",  
  "name": "mynet",  
  "type": "bridge",  
  "bridge": "mynet0",  
  "isDefaultGateway": true,  
  "forceAddress": false,  
  "ipMasq": true,  
  "hairpinMode": true,  
  "ipam": {  
    "type": "host-local",  
    "subnet": "10.10.0.0/16"  
  }  
}
```
- Section:** Example L2-only configuration
- Code:**

```
{  
  "cniVersion": "0.3.1",  
  "name": "mynet",  
  "type": "bridge",  
  "bridge": "mynet0",  
  "ipam": {}  
}
```
- Section:** Network configuration reference
- List:**
 - **name** (string, required): the name of the network.
 - **type** (string, required): "bridge".
 - **bridge** (string, optional): name of the bridge to use.
 - **isGateway** (boolean, optional): Sets isGateway address.
 - **isDefaultGateway** (boolean, optional): Sets isDefault route. Defaults to false.
 - **forceAddress** (boolean, optional): Indicates if a value has been changed. Defaults to false.

管理员

主页

概述

项目

搜索

API Explorer

事件

Operators

工作负载

虚拟化

网络

服务

路由

Ingresses

网络策略

NetworkAttachmentDefinitions

存储

构建

观察

您已以临时管理用户身份登录。更新集群 OAuth 配置以允许其他人登录。

项目: default

Network Attachment Definitions > Network Attachment Definition详情

NAD bridge

操作 ▾

Details YAML

Opt + F1 无障碍访问帮助 | 查看捷径 | 查看侧边栏

```
1 apiVersion: k8s.cni.cncf.io/v1
2 kind: NetworkAttachmentDefinition
3 metadata:
4   creationTimestamp: '2022-08-11T02:41:24Z'
5   generation: 1
6   managedFields:-
16   name: bridge
17   namespace: default
18   resourceVersion: '338796'
19   uid: 2d3cf7c5-9cb1-4354-a1f6-31852f9fb9c8
20 spec:
21   config: >-
22     { "cniVersion": "0.3.1", "name": "mynet", "type": "bridge", "bridge": "mynet0", "isDefaultGateway": true, "forceAddress": false, "ipMasq": true, "hairpinMode": true, "ipam": { "type": "host-local", "subnet": "10.10.0.0/16" } }
```

保存

重新加载

取消



下载

```

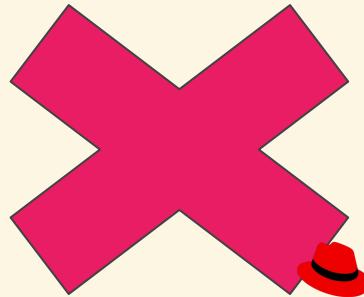
rabbit@rabbit-mbp:~ % ~ oc get vmi -o wide
NAME          AGE     PHASE   IP           NODENAME
rhel9-bridge  4m40s  Running  10.128.2.28  dhcp16-217-244.hpe2.lab.eng.bos.redhat.com
rhel9-bridge2 11m    Running  10.130.1.58  ocp-node01
rhel9-vlan-1  34m    Scheduling
rhel9-vlan-2  28m    Scheduling
vm-fedora-hw-rs4cbks 71m    Running  10.129.2.19  dhcp16-217-225.hpe2.lab.eng.bos.redhat.com
vm-fedora-hw-rsgvbkq  71m    Running  10.128.2.19  dhcp16-217-244.hpe2.lab.eng.bos.redhat.com
vm-fedora-hw-rsmzw2v  71m    Running  10.131.0.68  dhcp16-217-158.hpe2.lab.eng.bos.redhat.com
READY   LIVE-MIGRATABLE PAUSED
True    True
True    True
False
False
True    False
True    False
True    False
→ ~ oc get network-attachment-definition bridge
^C
→ ~
→ ~ oc get network-attachment-definition
NAME          AGE
bridge        79m
l2bridge      79m
vlan          77m
→ ~ oc describe network-attachment-definition bridge
Name:          bridge
Namespace:    default
Labels:        <none>
Annotations:  <none>
API Version: k8s.cni.cncf.io/v1
Kind:         NetworkAttachmentDefinition
Metadata:
  Creation Timestamp: 2022-08-11T02:41:24Z
  Generation:       1
  Managed Fields:
    API Version:  k8s.cni.cncf.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:spec:
        .
        f:config:
          Manager:      Mozilla
          Operation:   Update
          Time:        2022-08-11T02:41:24Z
          Resource Version: 338796
          UID:        2d3cf7c5-9cb1-4354-a1f6-31852f9fb9c8
      Spec:
        Config: { "cniVersion": "0.3.1", "name": "mynet", "type": "bridge", "bridge": "mynet0", "isDefaultGateway": true, "forceAddress": false, "ipMasq": true, "hairpinMode": true, "ipam": { "type": "host-local", "subnet": "10.10.0.0/16" } }
        Events:  <none>
→ ~

```



Red Hat

```
virtctl console rhel9-bridge          virtctl console rhel9-bridge3
[root@rhel9-bridge ~]# ip addr           [root@rhel9-bridge3 ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
  qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
  inetc6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc fq_codel state UP group
  default qlen 1000
    link/ether 02:ca:4d:00:00:05 brd ff:ff:ff:ff:ff:ff
    altname enp1s0
    inet 10.0.2.2/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
      valid_lft 86312276sec preferred_lft 86312276sec
    inetc6 fe80::ca:4dff:fe00:5/64 scope link
      valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
  default qlen 1000
    link/ether 02:ca:4d:00:00:06 brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 10.10.0.2/16 brd 10.10.255.255 scope global dynamic noprefixroute eth1
      valid_lft 86312276sec preferred_lft 86312276sec
    inetc6 fe80::c7a4:7eaa:70d4:9de7/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
[root@rhel9-bridge ~]#  [root@rhel9-bridge3 ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
  qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
  inetc6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc fq_codel state UP group
  default qlen 1000
    link/ether 02:ca:4d:00:00:0d brd ff:ff:ff:ff:ff:ff
    altname enp1s0
    inet 10.0.2.2/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
      valid_lft 86312705sec preferred_lft 86312705sec
    inetc6 fe80::ca:4dff:fe00:d/64 scope link
      valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
  default qlen 1000
    link/ether 02:ca:4d:00:00:0e brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 10.10.0.3/16 brd 10.10.255.255 scope global dynamic noprefixroute eth1
      valid_lft 86312705sec preferred_lft 86312705sec
    inetc6 fe80::bdcd:916d:1993:3df5/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
[root@rhel9-bridge3 ~]#
```



Red Hat

OpenShift Network! Multus CNI plugin - Problem Solved

Run a vm on specific host and volumes?

nodeSelector:

kubernetes.io/hostname: hp-dl380g10-06.lab.eng.pek2.redhat.com

annotations:

kubevirt.io/provisionOnNode: hp-dl380g10-06.lab.eng.pek2.redhat.com



```
virctl console rhe19-bridge
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1450 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 02:ca:4d:00:00:05 brd ff:ff:ff:ff:ff:ff
    altname enp1s0
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:ca:4d:00:00:06 brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 10.10.0.2/16 brd 10.10.255.255 scope global dynamic rtt_mtu 1450
        valid_lft 86310638sec preferred_lft 86310638sec
    inet6 fe80::c7a4:7eaa:70d4:9de7/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[root@rhe19-bridge ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1450 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 02:ca:4d:00:00:05 brd ff:ff:ff:ff:ff:ff
    altname enp1s0
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:ca:4d:00:00:06 brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 10.10.0.2/16 brd 10.10.255.255 scope global dynamic rtt_mtu 1450
        valid_lft 86310328sec preferred_lft 86310328sec
    inet6 fe80::c7a4:7eaa:70d4:9de7/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[root@rhe19-bridge ~]# useradd rabbit
[root@rhe19-bridge ~]# echo 1 | passwd --stdin rabbit
Changing password for user rabbit.
passwd: all authentication tokens updated successfully.
[root@rhe19-bridge ~]# python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080) ...
10.10.0.3 - - [11/Aug/2022 00:54:17] "GET / HTTP/1.1" 200 -
```



```
virctl console rhe19-bridge2
[root@rhe19-bridge2 ~]# curl http://10.10.0.2:8080
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/REC-HTML4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href=".bash_history">.bash_history</a></li>
<li><a href=".bash_logout">.bash_logout</a></li>
<li><a href=".bash_profile">.bash_profile</a></li>
<li><a href=".bashrc">.bashrc</a></li>
<li><a href=".cshrc">.cshrc</a></li>
<li><a href=".lesshtc">.lesshtc</a></li>
<li><a href=".ssh">.ssh</a></li>
<li><a href=".tcshrc">.tcshrc</a></li>
</ul>
<hr>
</body>
</html>
```



The screenshot shows a Red Hat OpenShift Container Platform interface on the left and a CNI documentation page on the right.

Left Side (OpenShift UI):

- Header:** Red Hat OpenShift Container Platform
- Navigation:** 管理员 (Administrator), 主页 (Home), 概述 (Overview), 项目 (Projects), 搜索 (Search), API Explorer, 事件 (Events), Operators, 工作负载 (Workloads), 虚拟化 (Virtualization), 网络 (Network), 服务 (Services), 路由 (Routes), Ingresses, 网络策略 (Network Policies), NetworkAttachmentDefinitions (highlighted).
- Content:** Create Network Attachment Definition dialog. It contains a YAML editor with the following code:

```
1 apiVersion: k8s.cni.cncf.io/v1
2 kind: NetworkAttachmentDefinition
3 metadata:
4   name: example
5   namespace: default
6 spec:
7   config: {}
```

A red arrow points from the "config: {}" line to the "Example configuration" section on the right.

Right Side (CNI Documentation):

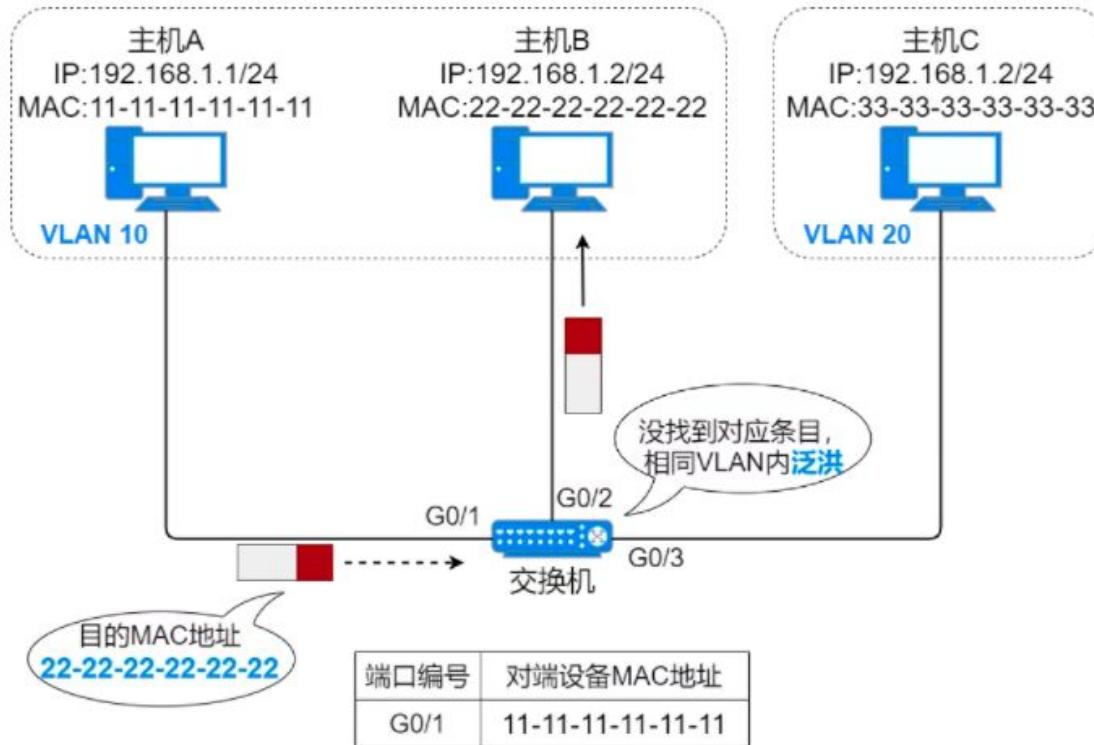
- Header:** CNI | Home Documentation Plugins
- Section:** static IP address management plugin
- List:** Main, vlan plugin, bridge plugin, host-device, ipvlan plugin, macvlan plugin, ptp plugin, win-bridge plugin, win-overlay plugin, meta, bandwidth plugin, firewall plugin, Port-mapping plugin, Source based routing plugin, tuning plugin, vrf plugin.
- Section:** On this page: Overview, Example configuration, Example L2-only configuration, Network configuration reference, Interface configuration arguments reference.
- Section:** Example configuration
- Code:**

```
{  
  "cniVersion": "0.3.1",  
  "name": "mynet",  
  "type": "bridge",  
  "bridge": "mynet0",  
  "isDefaultGateway": true,  
  "forceAddress": false,  
  "ipMasq": true,  
  "hairpinMode": true,  
  "ipam": {  
    "type": "host-local",  
    "subnet": "10.10.0.0/16"  
  }  
}
```
- Section:** Example L2-only configuration
- Code:**

```
{  
  "cniVersion": "0.3.1",  
  "name": "mynet",  
  "type": "bridge",  
  "bridge": "mynet0",  
  "ipam": {}  
}
```
- Section:** Network configuration reference
- List:**
 - **name** (string, required): the name of the network.
 - **type** (string, required): "bridge".
 - **bridge** (string, optional): name of the bridge to use.
 - **isGateway** (boolean, optional): Sets isGateway address.
 - **isDefaultGateway** (boolean, optional): Sets isDefault route. Defaults to false.
 - **forceAddress** (boolean, optional): Indicates if a value has been changed. Defaults to false.

Maybe Deeper...
(flow table)

Openvswitch: Traditional Switch



Openvswitch: Flow Table

```
R2#show ip route
```

```
S 192.168.1.0/24 [1/0] via 192.168.5.2
C 192.168.2.0/24 is directly connected, FastEthernet1/0
C 192.168.5.0/24 is directly connected, Serial2/0
```

```
Switch>show mac address
Mac Address Table
```

Vlan	Mac Address	Type	Ports
---	---	---	---
1	0050.0f40.e0a6	DYNAMIC	Fa0/1
1	0090.0c02.cb01	DYNAMIC	Fa0/2

```
root@controller:~# ovs-ofctl dump-flows br-int
```

```
NXST_FLOW reply (xid=0x4):
cookie=0xbfcbcc1f0281737c, table=0, n_packets=885,
n_bytes=873, idle_age=0, hard_age=65534, priority=2, in_port=1
actions=drop
```



ovs-ofctl 命令是对流表的操作，包括对流表的增, 删, 改, 查等命令。

简单来说流表类似于交换机的MAC地址表, 路由器的路由表, 是ovs交换机指挥流量转发的表。

```

root@intel-whitley-08 ~]# ovs-ofctl dump-flows helloworld
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=6260.500s, table=0, n_packets=0, n_bytes=0, idle_age=6260, dl_src=01:00:00:00:00:00/01:00:00:00:00:00 actions=drop
ns=drop
cookie=0x0, duration=6253.143s, table=0, n_packets=0, n_bytes=0, idle_age=6253, dl_dst=01:80:c2:00:00:00/ff:ff:ff:ff:f0 actions=drop
ns=drop
cookie=0x0, duration=6228.068s, table=0, n_packets=0, n_bytes=0, idle_age=6705, priority=0 actions=resubmit(,1)
cookie=0x0, duration=5983.760s, table=1, n_packets=0, n_bytes=0, idle_age=5983, priority=99,in_port=1 actions=resubmit(,2)
cookie=0x0, duration=5922.761s, table=1, n_packets=0, n_bytes=0, idle_age=5922, priority=99,in_port=2,vlan_tci=0x0000 actions=move
d_vlan_vid:20,resubmit(,2)
cookie=0x0, duration=5922.761s, table=1, n_packets=0, n_bytes=0, idle_age=5922, priority=99,in_port=3,vlan_tci=0x0000 actions=move
d_vlan_vid:30,resubmit(,2)
cookie=0x0, duration=5922.761s, table=1, n_packets=0, n_bytes=0, idle_age=5922, priority=99,in_port=4,vlan_tci=0x0000 actions=move
d_vlan_vid:30,resubmit(,2)
cookie=0x0, duration=6041.462s, table=1, n_packets=0, n_bytes=0, idle_age=6041, priority=0 actions=drop
cookie=0x0, duration=5297.433s, table=2, n_packets=0, n_bytes=0, idle_age=5297, actions=learn(table=10,NXM_OF_VLAN_TCI[0..11],NX
M_OF_ETH_DST[]>NXM_OF_ETH_SRC[],load:NXM_OF_IN_PORT[]->NXM_NX_REG0[0..15]),resubmit(,3)
cookie=0x0, duration=2489.922s, table=3, n_packets=0, n_bytes=0, idle_age=2489, priority=99,dl_dst=01:00:00:00:00:00/01:00:00:00
:00:00 actions=resubmit(,4)
cookie=0x0, duration=2542.035s, table=3, n_packets=0, n_bytes=0, idle_age=2542, priority=50 actions=resubmit(,10),resubmit(,4)
cookie=0x0, duration=1611.709s, table=4, n_packets=0, n_bytes=0, idle_age=1611, reg0=0x1 actions=output:1
cookie=0x0, duration=1421.022s, table=4, n_packets=0, n_bytes=0, idle_age=1607, reg0=0x2 actions=strip_vlan,output:2
cookie=0x0, duration=1421.022s, table=4, n_packets=0, n_bytes=0, idle_age=1607, reg0=0x3 actions=strip_vlan,output:3
cookie=0x0, duration=1421.022s, table=4, n_packets=0, n_bytes=0, idle_age=1607, reg0=0x4 actions=strip_vlan,output:4
cookie=0x0, duration=1410.863s, table=4, n_packets=0, n_bytes=0, idle_age=1410, priority=50,reg0=0 actions=output:1
cookie=0x0, duration=1410.863s, table=4, n_packets=0, n_bytes=0, idle_age=1410, priority=99,reg0=0,dl_vlan=20 actions=output:1,s
trip_vlan,output:2
cookie=0x0, duration=1410.863s, table=4, n_packets=0, n_bytes=0, idle_age=1410, priority=99,reg0=0,dl_vlan=30 actions=output:1,s
trip_vlan,output:3,output:4
cookie=0x0, duration=4732.239s, table=10, n_packets=0, n_bytes=0, idle_age=4732, vlan_tci=0x0014/0x0fff,dl_dst=50:00:00:00:00:01
actions=load:0x1->NXM_NX_REG0[0..15]
cookie=0x0, duration=4359.807s, table=10, n_packets=0, n_bytes=0, idle_age=4359, vlan_tci=0x0014/0x0fff,dl_dst=50:00:00:00:00:02
actions=load:0x2->NXM_NX_REG0[0..15]
cookie=0x0, duration=2428.931s, table=10, n_packets=0, n_bytes=0, idle_age=2428, hard_age=1946, vlan_tci=0x0014/0x0fff,dl_dst=f0
:00:00:00:00:01 actions=load:0x1->NXM_NX_REG0[0..15]
cookie=0x0, duration=2074.948s, table=10, n_packets=0, n_bytes=0, idle_age=2074, hard_age=2059, vlan_tci=0x0014/0x0fff,dl_dst=90
:00:00:00:00:01 actions=load:0x2->NXM_NX_REG0[0..15]
cookie=0x0, duration=653.407s, table=10, n_packets=0, n_bytes=0, idle_age=653, hard_age=129, vlan_tci=0x001e/0x0fff,dl_dst=10:00
:00:00:00:00:01 actions=load:0x1->NXM_NX_REG0[0..15]
cookie=0x0, duration=322.833s, table=10, n_packets=0, n_bytes=0, idle_age=322, vlan_tci=0x001e/0x0fff,dl_dst=20:00:00:00:00:01 a
ctions=load:0x4->NXM_NX_REG0[0..15]

```

Our switch design will consist of five main flow tables, each of which implements one stage in the switch pipeline:

Table 0 Admission control

Table 1 VLAN input processing

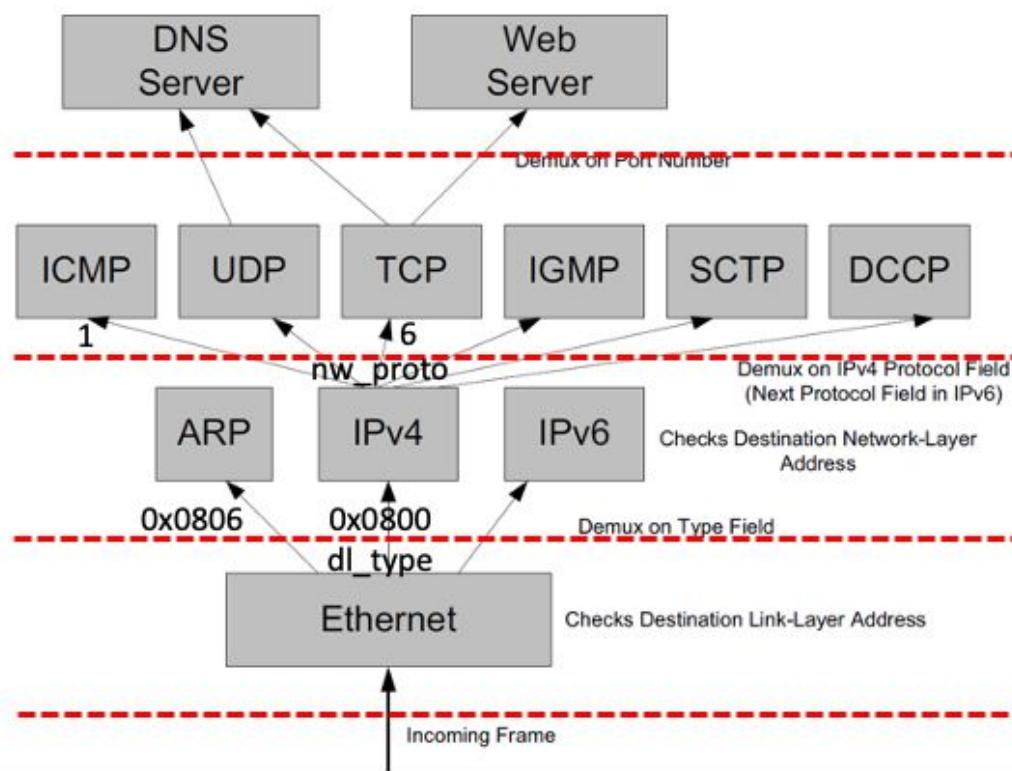
Table 2 Learn source MAC and VLAN for ingress port

Table 3 Look up learned port for destination MAC and VLAN

Table 4 Output processing



Openvswitch: Flow Table - Match Field



icmp_type=type
icmp_code=code

tp_src=port
tp_dst=port

arp_sha=xx:xx:xx:xx:xx:xx nw_src=ip[/netmask]
arp_tha=xx:xx:xx:xx:xx:xx nw_dst=ip[/netmask]

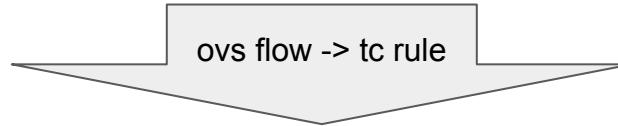
dl_src=xx:xx:xx:xx:xx:xx/xx:xx:xx:xx:xx:xx
dl_dst=xx:xx:xx:xx:xx:xx/xx:xx:xx:xx:xx:xx
dl_vlan=vlan

in_port=port

OvS-TC offload

A sample of TC offload:

```
recirc_id(0),in_port(3),eth(src=24:8a:07:a5:28:02, dst=24:8a:07:a5:28:01), eth_type(0x0  
800) actions:2
```



```
tc filter add dev ens1f0_1 ingress protocol ip chain 0 prio 3 flower dst_mac  
24:8a:07:a5:28:01 src_mac 24:8a:07:a5:28:02 action mirred egress redirect dev  
ens1f0_0
```

SmartNIC读取tc rules,并根据这些规则,在NIC
内部的e-switch中进行转发处理,完成后,将统
计信息反馈给TC subsystem

Openvswitch: Flow Table - Actions

对于Flow Table的管理, 由ovs-ofctl来控制

add-flow *switch* *flow*

mod-flows *switch* *flow*

del-flows *switch* [*flow*]

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie
--------------	----------	----------	--------------	----------	--------

Table 1: Main components of a flow entry in a flow table.

Match Field

Actions

Actions:

output:port 和 output:NXM_NX_REG0[16..31]

enqueue:port:queue

mod_vlan_vid:vlan_vid

strip_vlan

mod_dl_src:mac 和 mod_dl_dst:mac

mod_nw_src:ip 和 mod_nw_dst:ip

mod_tp_src:port 和 mod_tp_dst:port

set_tunnel:id

resubmit([port],[table])

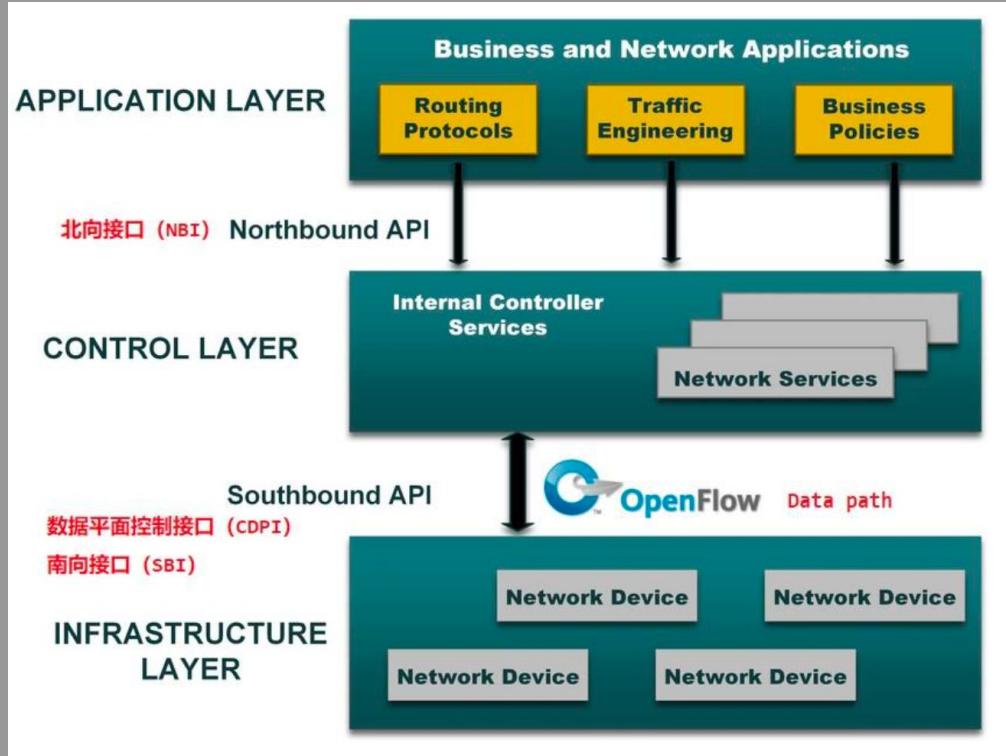
move:src[start..end]→dst[start..end]

load:value→dst[start..end]

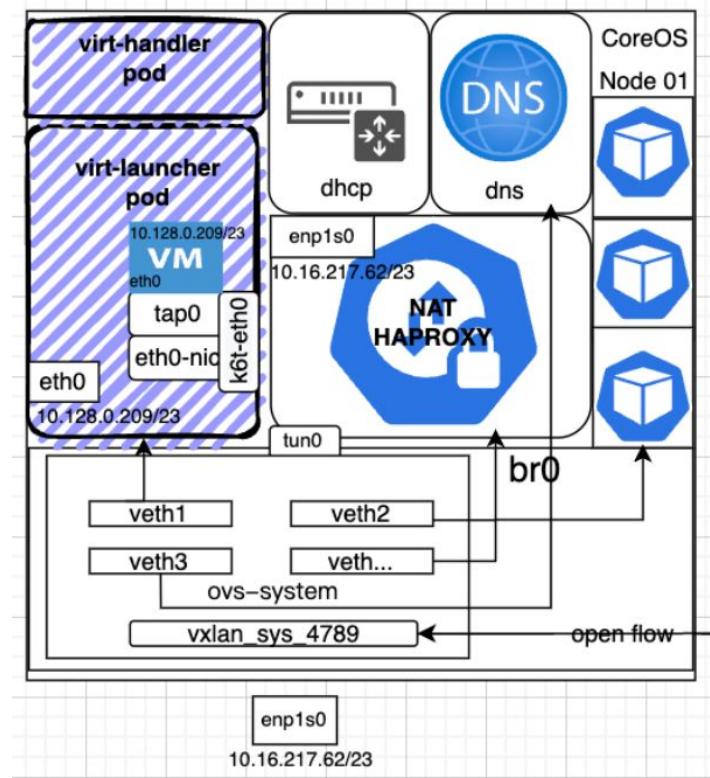
learn(argument[,argument]...)

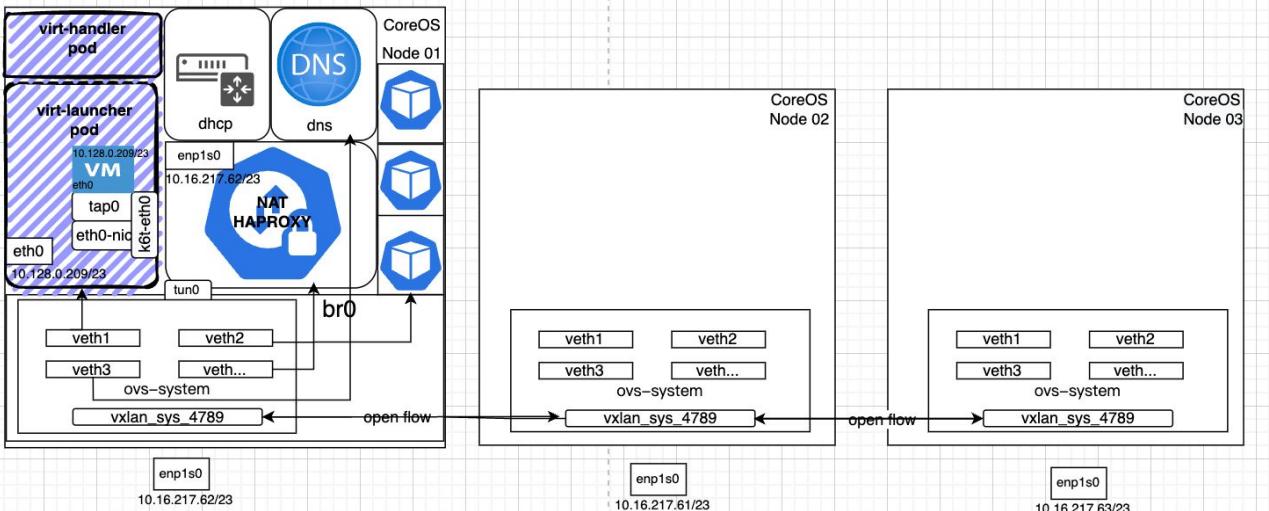


CONTROL LAYER



OpenShift Network! SDN part





Kubernetes Pod SDN

10.128.0.0/14

default via 10.16.217.254 dev enp1s0 proto static metric 100
 10.16.216.0/23 dev enp1s0 proto kernel scope link src 10.16.217.62 metric 100
 10.128.0.0/14 dev tun0 scope link
 172.30.0.0/16 dev tun0

Kubernetes Service SDN

NodePort

172.30.0.0/16

NodePort



此网页正使用大量能耗。将其关闭可能提高Mac的响应速度。

不安全 - dell-pem630-01.dell.lab.eng.bos.redhat.com

OPEN DAYLIGHT Topology Logout (admin)

Topology

Nodes Yang UI Yang Visualizer Yangman

```
[root@dell-pem630-01 ~]# ovs-vsctl show
a42dac02-fe91-4dc1-b4d5-3c6ad2f65305
  Bridge "vlan1_br"
    Controller "tcp:127.0.0.1:6633"
      is_connected: true
    Port "vnet12"
      Interface "vnet12"
    Port patch-eth
      Interface patch-eth
        type: patch
        options: {peer=patch-tap}
    Port "vlan1_br"
      Interface "vlan1_br"
        type: internal
  Bridge ubuntu_br
    Controller "tcp:127.0.0.1:6633"
      is_connected: true
    Port "vnet10"
      Interface "vnet10"
    Port patch-tap
      Interface patch-tap
```

Controls Reload

host:52:54:00:6d:3c:11
host:52:54:00:6d:3c:13
openflow:103942692919363
openflow:129831458859087
host:52:54:00:6d:3c:12
host:52:54:00:6d:3c:12

 Red Hat

OpenShift Network! OVN part

OVN (Open Virtual Network) 是一系列的守护程序，它将虚拟网络配置翻译成 OpenFlow，并将其安装到 Open vSwitch 中。

OVN 比 Open vSwitch 提供了更高层次的抽象，它与模拟路由器和模拟交换机一起工作而不仅仅是 flows。

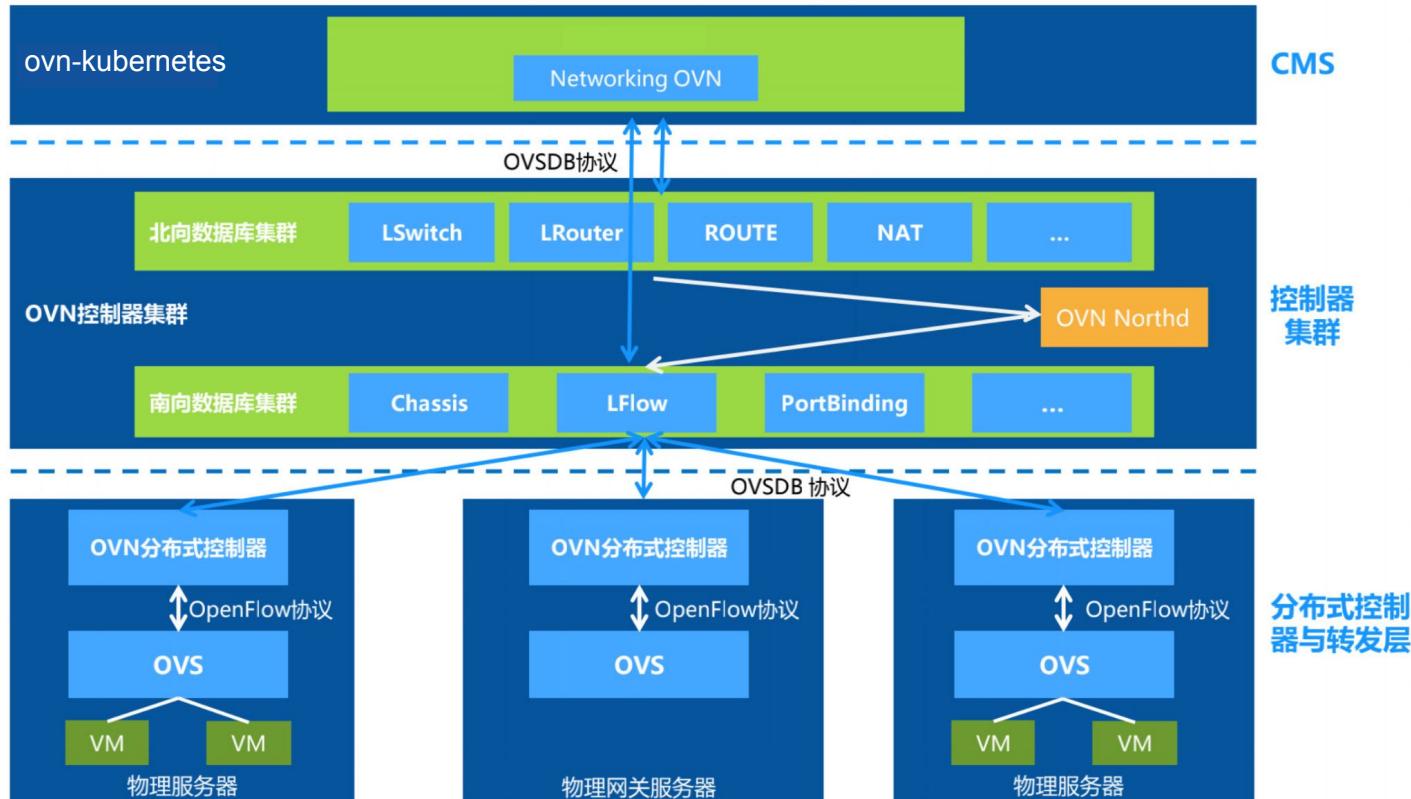
OVN旨在被云计算管理软件（CMS）所使用。OVN提供的一些高级功能包括：

- Distributed virtual routers
- Distributed logical switches
- Access Control Lists
- DHCP
- DNS server

与 Open vSwitch 一样，OVN是用独立于平台的C编写的。**OVN完全在用户空间中运行，因此不需要安装内核模块。**



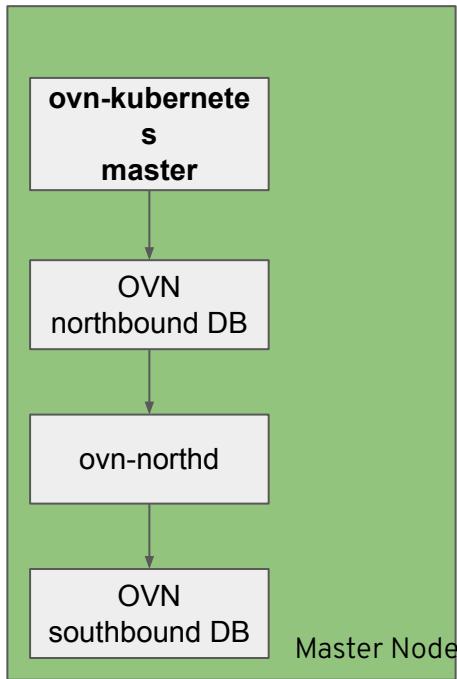
OpenShift Network! OVN part - Architecture



OVN-Kubernetes

- CNI 网络插件从 ocp (GA from 4.6)/kubernetes
 - <https://github.com/ovn-org/ovn-kubernetes>
 - Started by the OVN/OVS communities
 - 使用 OVN on OVS 作为抽象来管理节点上的网络流量
 - 使用 Geneve(通用网络虚拟化封装)协议, 而不是 VXLAN(由OpenShift SDN使用)来创建节点之间的覆盖 overlay 网络。
 - Creates logical network topologies
 - Logical switches, routers, ports, acls (network policies), load balancers etc..
 - 不像 SDN 那样需要 kube-proxy(作为 pod 和 ocp 之间的连接渠道)
 - 允许将网络管理抽象到多节点

OVN-Kubernetes Architecture: Master



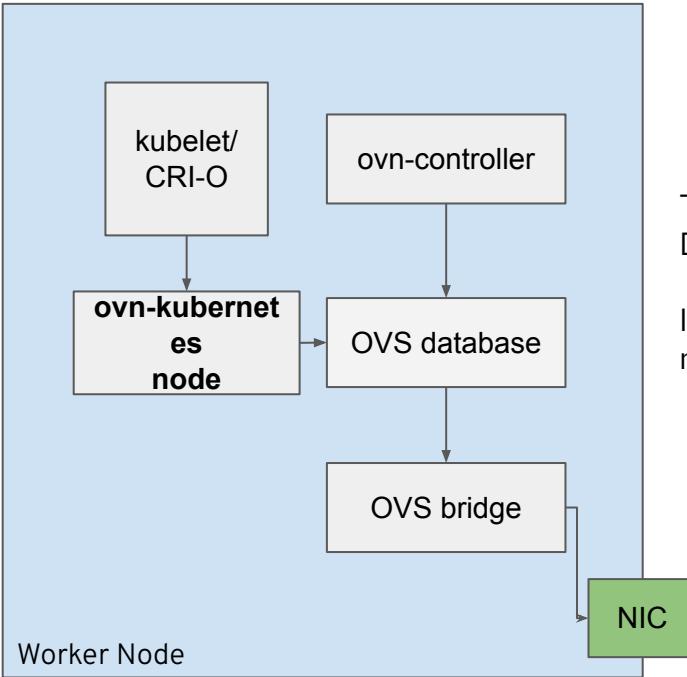
- **OVN-Kubernetes Master**
 - OVN Kubernetes **component**
 - **Central process** that **watches** for cluster events (Pods, Namespaces, Services, Endpoints, NetworkPolicy)
 - **Translates** cluster events into OVN logical network elements **stored** in nbdb
 - **Tracks** Kube-API state
 - **Manages** pod **subnet allocation** to nodes
- **OVN-northd**
 - **Native** OVN component
 - Process that **converts** northbound DB network representation to the lower-level logical flows that are stored in southbound DB

These processes are **started** by the CNO via a **Daemonset** -> `ovnkube-master`

They can be seen running on an OCP cluster in the `openshift-ovn-kubernetes` namespace within the `ovnkube-master` pod as the following containers

- `[northd, nbdb, sbdb, ovnkube-master]`

OVN-Kubernetes Architecture: Node



- **Open vSwitch**

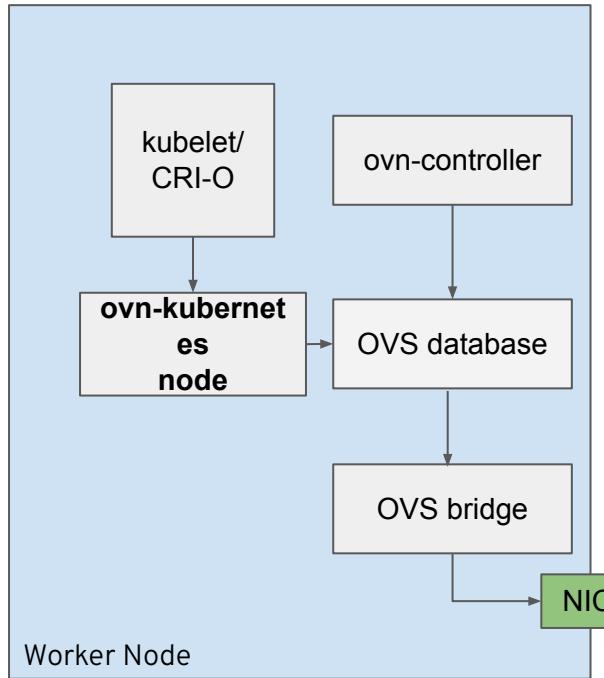
- **Pushes** the new “edge of network” to the hypervisor
- Multilayer software switch used to **implement** openflow rules
- Datapath used by containers

The OVS process is run via systemd on the host but is managed by the CNO via a Daemonset for log tailing and other monitoring reasons-> **ovs-node**

It can be seen running on an OCP cluster in the **openshift-ovn-kubernetes** namespace within the **ovs-node** pods as the following containers

- **[ovs-daemons]**

OVN-Kubernetes Architecture: Node



- **OVN-Controller**
 - OVN component
 - Watches sbdb
 - Matches OVS “physical” ports to OVN logical ports
 - Reads logical flows from sbdb, translates them into OpenFlow flows and sends them to the worker node’s OVS daemon
- **OVN-Kubernetes node**
 - Called as CNI plugin (just an executable) from kubelet/CRI-O
 - Digests IPAM annotation written by ovn-kubernetes master
 - Sets up firewall rules and routes for HostPort and Service access from node
 - Creates OVS port on bridge, moves it into pod network namespace, sets IP details/QoS
 - Deletes entities when pods die

These processes are also started by the CNO via a Daemonset->ovnkube-node. They can be seen running on an OCP cluster in the openshift-ovn-kubernetes namespace within the ovnkube-node pod as the following containers

- [ovn-controller, ovnkube-node]

OpenShift Network! OVN part

```
→ ~ oc get pods -A | grep -i operator | grep -i dns
openshift-dns-operator                               dns-operator-69db46cc47-9bwdp
                                                    2/2      Running     0
                                                    27h

→ ~ oc get pods -A | grep -i operator | grep -i dhcp
→ ~ oc get all -n openshift-ovn-kubernetes

NAME          READY   STATUS    RESTARTS   AGE
pod/ovnkube-master-hw7gw  6/6     Running   0          26h
pod/ovnkube-master-nnhsr  6/6     Running   0          27h
pod/ovnkube-master-xdjxp  6/6     Running   0          27h
pod/ovnkube-node-4cmwt   5/5     Running   0          26h
pod/ovnkube-node-58dk8   5/5     Running   0          26h
pod/ovnkube-node-m7ngq   5/5     Running   0          27h
pod/ovnkube-node-nnmxx   5/5     Running   0          27h
pod/ovnkube-node-rpvw9   5/5     Running   0          26h
pod/ovnkube-node-tjqnq   5/5     Running   0          27h

NAME           TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
service/ovn-kubernetes-master  ClusterIP  None           <none>        9102/TCP      27h
service/ovn-kubernetes-node    ClusterIP  None           <none>        9103/TCP,9105/TCP 27h
service/ovnkube-db             ClusterIP  None           <none>        9641/TCP,9642/TCP 27h

NAME          DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR          AGE
daemonset.apps/ovnkube-master 3         3         3       3           3           beta.kubernetes.io/os=linux, node-role.kubernetes.io/master= 27h
daemonset.apps/ovnkube-node    6         6         6       6           6           beta.kubernetes.io/os=linux 27h
→ ~
```



OpenShift Network! OVN part - ovn-controller

```
rabbit@rabbit-mbp:~
```

```
sh-4.4# ovs-vsctl list Open_vSwitch
_uuid          : 9d426064-2bde-468a-b7f4-0e3baebf8500
bridges        : [810a8430-f347-4868-80e5-afdf41fc4822, d1961230-e621-443f-8d23-8f09bc9c4967]
cur_cfg        : 73
datapath_types : [netdev, system]
datapaths      : {system=42d42d91-8e74-4d57-808d-f7c5147236b4}
db_version     : "8.3.0"
dpdk_initialized: false
dpdk_version   : "DPDK 21.11.0"
external_ids   : {hostname=dhcp16-213-124.lab2.eng.bos.redhat.com, ovn-bridge-mappings="physnet:br-ex", ovn-enable-lflow
-cache="true", ovn-encap-ip="10.16.213.124", ovn-encap-type=geneve, ovn-memlimit-lflow-cache-kb="1048576", ovn-monitor-all="t
rue", ovn-ofctrl-wait-before-clear="0", ovn-openflow-probe-interval="180", ovn-remote="ssl:10.16.213.124:9642,ssl:10.16.213.1
35:9642,ssl:10.16.213.190:9642", ovn-remote-probe-interval="180000", rundir="/var/run/openvswitch", system-id="16447c61-da38-
4157-802d-8fbfbca4c950"}
iface_types    : [bareudp, erspan, geneve, gre, gtpu, internal, ip6erspan, ip6gre, lisp, patch, stt, system, tap, vxlan]
manager_options: []
next_cfg       : 73
other_config   : {vlan-limit="0"}
ovs_version    : "2.17.3"
ssl            : 62f66b35-26d8-4caa-8a35-9b3dfb668b5f
statistics     : {}
system_type    : rhcos
system_version: "4.11"
sh-4.4# timed out waiting for input: auto-logout
sh-4.4# timed out waiting for input: auto-logout

Removing debug pod ...
→ ~
→ ~
```



Red Hat

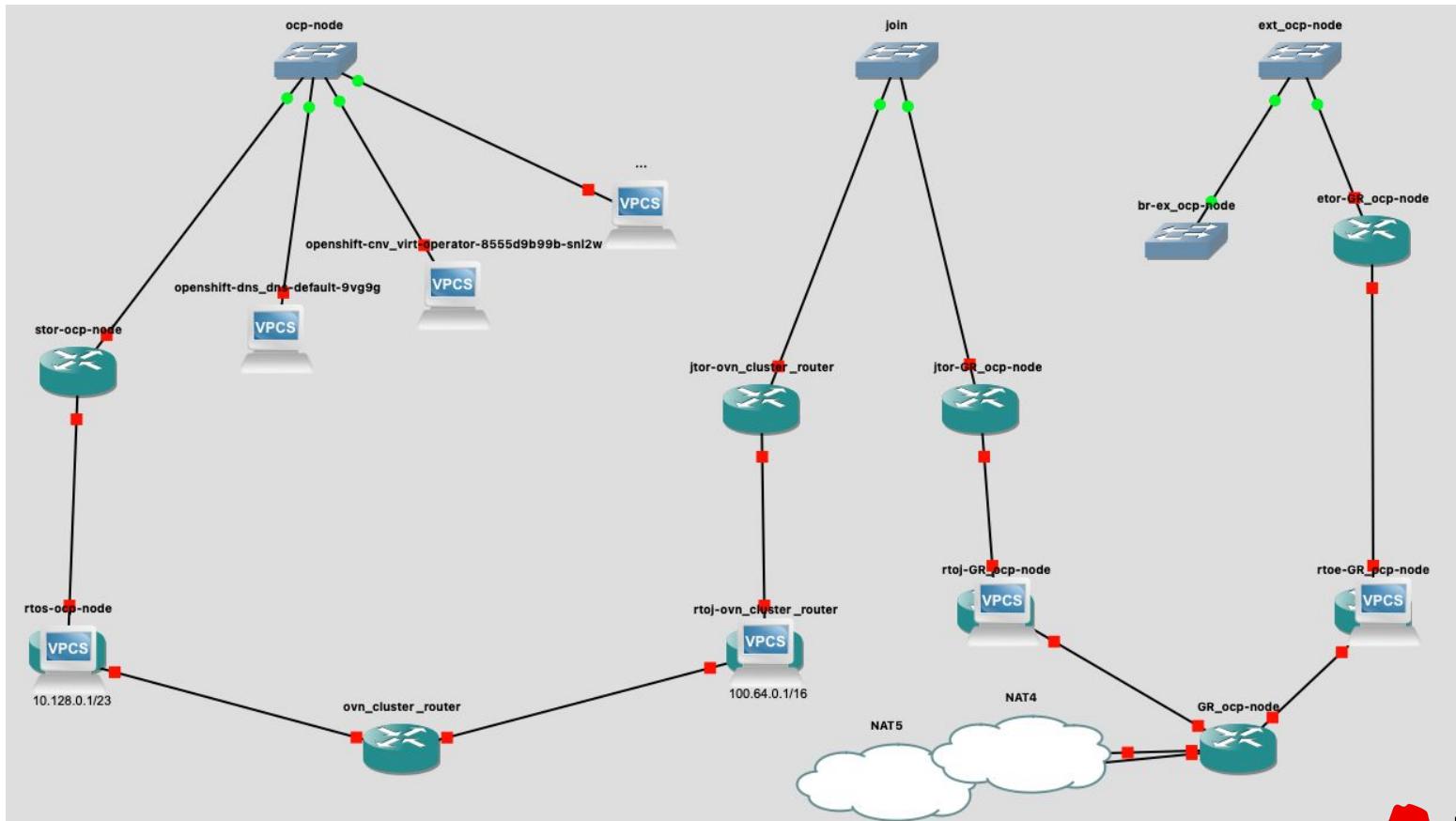
OpenShift Network! OVN part

```
oc --insecure-skip-tls-verify rsh ovnkube-master-9x78k
sh-4.4# ovn-nbctl show | more
switch 6f9ac7c4-8e1b-4ebc-9cad-2076de9a1140 (ovn-node)
  port stor-ocp-node
    type: router
    router-port: rtos-ocp-node
  port openshift-ingress-operator_ingress-operator-6457f4c5c6-qclkk
    addresses: ["0a:58:0a:80:00:20 10.128.0.32"]
  port openshift-service-ca_service-ca-6f9b4d848b-z796x
    addresses: ["0a:58:0a:80:00:1a 10.128.0.26"]
  port openshift-cnv_cluster-network-addons-operator-5797d6fd4f-k24j2
    addresses: ["0a:58:0a:80:00:61 10.128.0.97"]
  port openshift-monitoring_grafana-594bbd9f5c-79c7m
    addresses: ["0a:58:0a:80:00:48 10.128.0.72"]
  port openshift-monitoring_kube-state-metrics-6bf9f95b547-4pkg4
    addresses: ["0a:58:0a:80:00:40 10.128.0.64"]
  port openshift-cnv_hyperconverged-cluster-cli-download-6f775b5ffc-ntn56
    addresses: ["0a:58:0a:80:00:60 10.128.0.96"]
  port openshift-cnv_virt-handler-jv9sl
    addresses: ["0a:58:0a:80:00:80 10.128.0.128"]
  port openshift-machine-api_machine-api-operator-54585b6c7f-shhp6
    addresses: ["0a:58:0a:80:00:25 10.128.0.37"]
  port openshift-apiserver_apiserver-6b7d9b86fc-qmbnh
    addresses: ["0a:58:0a:80:00:2e 10.128.0.46"]
  port openshift-monitoring_prometheus-operator-764b888598-wq76c
    addresses: ["0a:58:0a:80:00:23 10.128.0.35"]
```

北 向网络定义逻辑流（人读）

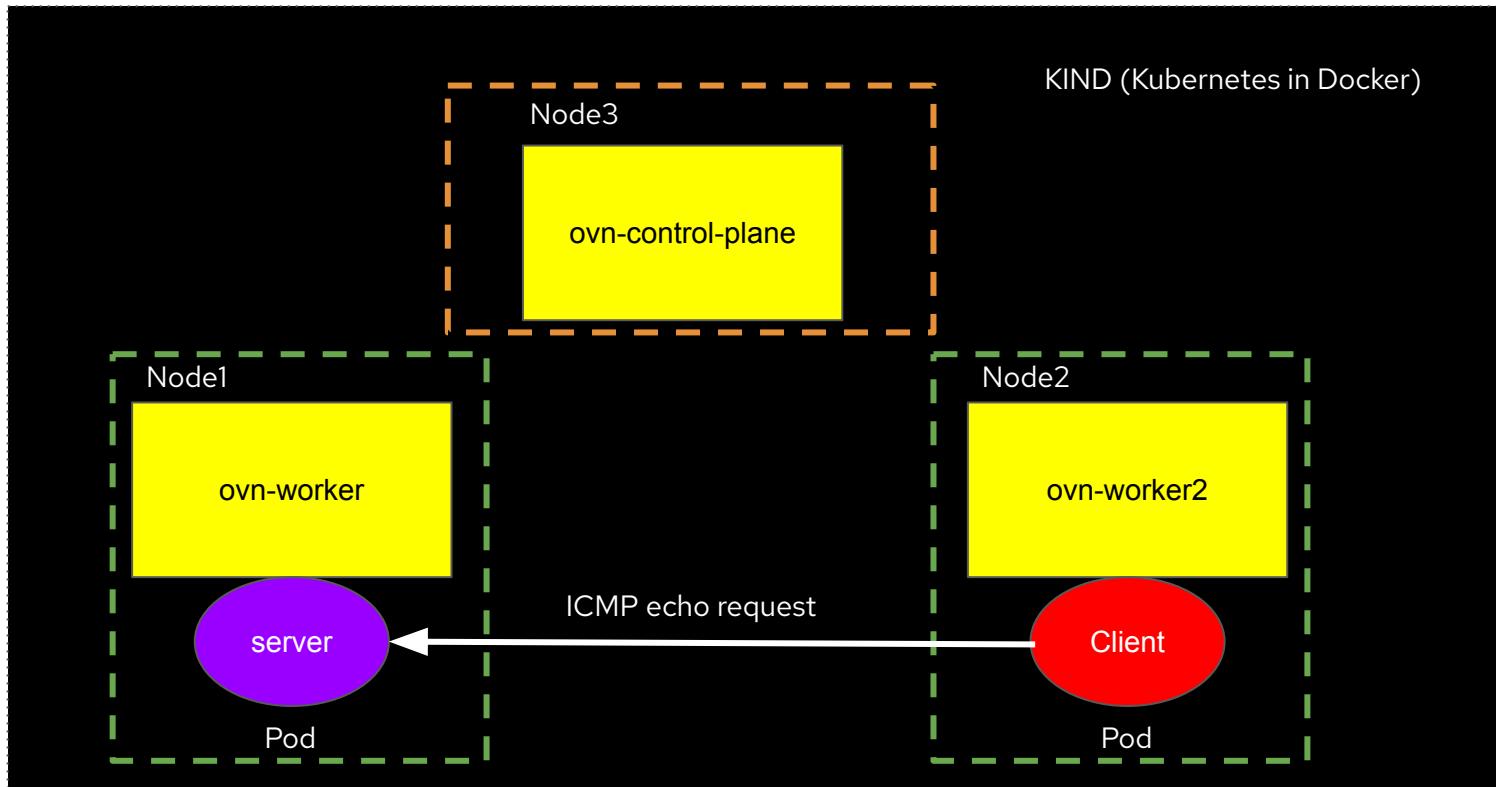
```
oc --insecure-skip-tls-verify rsh ovnkube-master-9x78k
sh-4.4# ovn-sbctl lflow-list | more
Datapath: "GR_ocp-node" (abf90e07-e65e-45ef-9424-9eb0f6c0429b) Pipeline: ingress
  table=0 (lr_in_admission      ), priority=100 , match=(vlan.present || eth.src[40]), action=(drop)
  table=0 (lr_in_admission      ), priority=50   , match=(eth.dst == 0a:58:64:40:00:02 && inport == "rtoj-GR_ocp-node"), action=(xreg0[0..47] = 0a:58:64:40:00:02; next)
  table=0 (lr_in_admission      ), priority=50   , match=(eth.dst == 52:54:00:6c:3c:0c && inport == "rtoe-GR_ocp-node"), action=(xreg0[0..47] = 52:54:00:6c:3c:0c; next)
  table=0 (lr_in_admission      ), priority=50   , match=(eth.mcast && inport == "rtoe-GR_ocp-node"), action=(xreg0[0..47] = 52:54:00:6c:3c:0c; next)
  table=0 (lr_in_admission      ), priority=50   , match=(eth.mcast && inport == "rtoj-GR_ocp-node"), action=(xreg0[0..47] = 0a:58:64:40:00:02; next)
  table=1 (lr_in_lookup_neighbor), priority=110 , match=(inport == "rtoe-GR_ocp-node" && arp.spa == 10.16.216.0/23 && arp.tpa == 10.16.217.64 && arp.op == 1), action=(reg9[2] = lookup_arp(inport, arp.spa, arp.sha); reg9[3] = 1; next)
  table=1 (lr_in_lookup_neighbor), priority=110 , match=(inport == "rtoj-GR_ocp-node" && arp.spa == 100.64.0.0/16 && arp.tpa == 100.64.0.2 && arp.op == 1), action=(reg9[2] = lookup_arp(inport, arp.spa, arp.sha); reg9[3] = 1; next)
  table=1 (lr_in_lookup_neighbor), priority=100 , match=(arp.op == 2), action=(reg9[2] = lookup_arp(inport, arp.spa, arp.sha); reg9[3] = 1; next)
  table=1 (lr_in_lookup_neighbor), priority=100 , match=(inport == "rtoe-GR_ocp-node" && arp.spa == 10.16.216.0/23 && arp.op == 1), action=(reg9[2] = lookup_ar
```

南 向网络表达 OpenFlow（机器）



```
[root@hpe-sl2x160zg6-02 ~]# kubectl get nodes
NAME           STATUS  ROLES          AGE   VERSION
ovn-control-plane  Ready   control-plane  16h   v1.24.0
ovn-worker      Ready   <none>        16h   v1.24.0
ovn-worker2     Ready   <none>        16h   v1.24.0
[root@hpe-sl2x160zg6-02 ~]# kubectl get pods --all-namespaces
NAMESPACE       NAME              READY  STATUS    RESTARTS  AGE
default         client            1/1    Running  0          33m
default         server            1/1    Running  0          34m
kube-system     coredns-6d4b75cb6d-7bnpc  1/1    Running  0          16h
kube-system     coredns-6d4b75cb6d-dhgjf  1/1    Running  0          16h
kube-system     etcd-ovn-control-plane  1/1    Running  0          16h
kube-system     kube-apiserver-ovn-control-plane  1/1    Running  0          16h
kube-system     kube-controller-manager-ovn-control-plane  1/1    Running  2 (15h ago)  16h
kube-system     kube-scheduler-ovn-control-plane  1/1    Running  2 (15h ago)  16h
local-path-storage local-path-provisioner-9cd9bd544-gn56b  1/1    Running  0          16h
ovn-kubernetes  ovnkube-db-5c6757846-s64b6  2/2    Running  0          15h
ovn-kubernetes  ovnkube-master-7f9fc84984-bftdx  2/2    Running  0          15h
ovn-kubernetes  ovnkube-node-d4nng  3/3    Running  0          15h
ovn-kubernetes  ovnkube-node-dgzjx  3/3    Running  0          15h
ovn-kubernetes  ovnkube-node-ksk6r  3/3    Running  0          15h
ovn-kubernetes  ovs-node-jnj6b  1/1    Running  0          15h
ovn-kubernetes  ovs-node-kbccf  1/1    Running  0          15h
ovn-kubernetes  ovs-node-ml2tj  1/1    Running  0          15h
[root@hpe-sl2x160zg6-02 ~]# kubectl get pods -n default
NAME    READY  STATUS    RESTARTS  AGE
client  1/1    Running  0          33m
server  1/1    Running  0          34m
[root@hpe-sl2x160zg6-02 ~]# kubectl get pods -n default -o wide
NAME    READY  STATUS    RESTARTS  AGE   IP          NODE    NOMINATED NODE  READINESS GATES
client  1/1    Running  0          33m  10.244.1.3  ovn-worker2  <none>  <none>
server  1/1    Running  0          34m  10.244.0.11  ovn-worker  <none>  <none>
[root@hpe-sl2x160zg6-02 ~]#
```

Goal - trace the packet flow



OVN logical entities

The logical components are stored in ovn-nbdb (showing only necessary components):

```
[root@ovn-control-plane ~]# ovn-nbctl show
switch 459ebbb9-2cb5-488f-8b1e-f35cda235957 (ovn-control-plane)
  port kube-system_coredns-6d4b75cb6d-7bnpc
    addresses: ["0a:58:0a:f4:02:04 10.244.2.4"]
  port stor-ovn-control-plane
    type: router
    router-port: rtos-ovn-control-plane
  port local-path-storage_local-path-provisioner-9cd9bd544-gn56b
    addresses: ["0a:58:0a:f4:02:05 10.244.2.5"]
  port kube-system_coredns-6d4b75cb6d-dhg5f
    addresses: ["0a:58:0a:f4:02:03 10.244.2.3"]
  port k8s-ovn-control-plane
    addresses: ["da:af:4d:0c:48:a5 10.244.2.2"]

switch 08adf0fb-4fee-474f-a010-0b7eef718806 (ovn-worker)
  port stor-ovn-worker
    type: router
    router-port: rtos-ovn-worker
  port k8s-ovn-worker
    addresses: ["36:1e:87:09:fe:90 10.244.0.2"]
  port default_server
    addresses: ["0a:58:0a:f4:00:0b 10.244.0.11"]

switch d000f3a1-7e84-4055-a326-2c3b989e52ba (ovn-worker2)
  port stor-ovn-worker2
    type: router
    router-port: rtos-ovn-worker2
  port default_client
    addresses: ["0a:58:0a:f4:01:03 10.244.1.3"]
  port k8s-ovn-worker2
    addresses: ["7e:18:c5:1f:9f:ba 10.244.1.2"]

router 9dd99282-8d69-4bf0-a06e-5abc95fd4191 (ovn_cluster_router)
  port rtos-ovn-control-plane
    mac: "0a:58:0a:f4:02:01"
    networks: ["10.244.2.1/24"]
    gateway chassis: [fb39d137-8727-418b-b3ce-2dd461ad50f]
  port rtoj-ovn_cluster_router
    mac: "0a:58:64:40:00:01"
    networks: ["100.64.0.1/16"]
  port rtos-ovn-worker
    mac: "0a:58:0a:f4:00:01"
    networks: ["10.244.0.1/24"]
    gateway chassis: [50758e69-6f60-4bad-91b3-9edc585b8f9]
  port rtos-ovn-worker2
    mac: "0a:58:0a:f4:01:01"
    networks: ["10.244.1.1/24"]
    gateway chassis: [d3d2ba94-a8a9-4832-a895-a611dc5992e]
```

OVN-KIND cluster

 outport

 inport

 logical switch

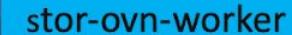
ovn-controlplane

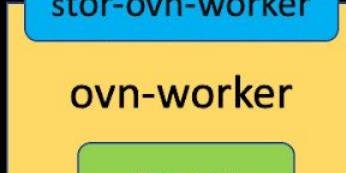
 rtos-ovn-control-plane

 ovn-cluster-router

 rtos-ovn-worker2

ovn-worker

 stor-ovn-worker

 ovn-worker

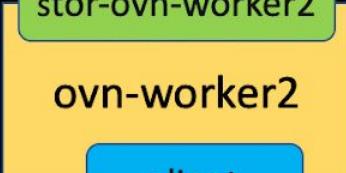
 server

 Server
10.244.2.
3

 conntrack

ovn-worker-2

 stor-ovn-worker2

 ovn-worker2

 client

 Client
10.244.0.
4

 conntrack

OVN Packet Processing

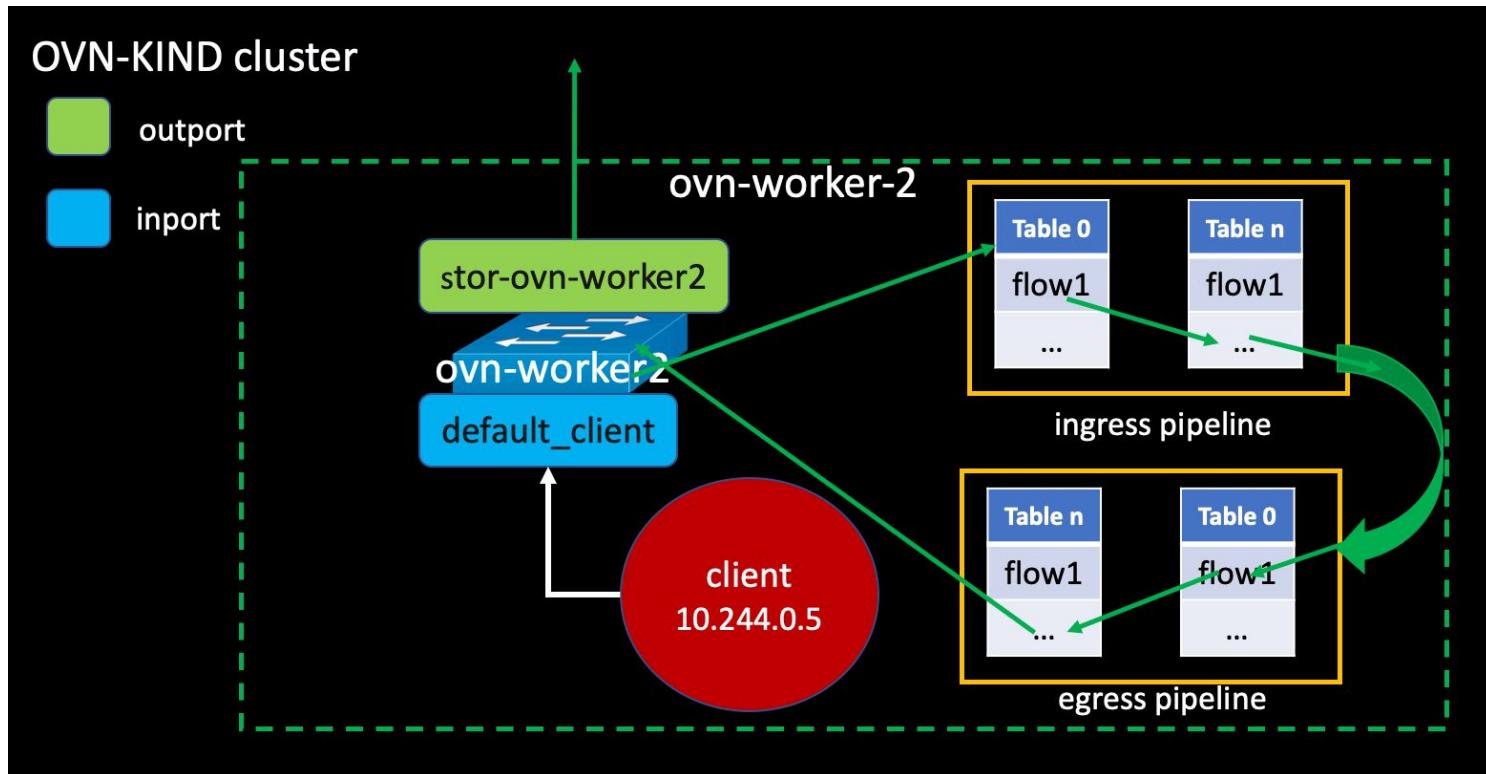
- OVN 有一套复杂的方法来处理数据包
- 它有两条 pipeline, 数据包通过这些管道进行 progress
 - Ingress
 - Egress
- OVN中的数据包首先进入 Ingress pipeline, 然后再进入 Egress pipeline。
 - 它将与 logical flow rules defined in ovn-sbdb Logical_Flows 进行比较, 用于该数据通路 datapath 的 ingress/egress pipeline。
 - 然后找到匹配的流 flow 并执行该动作。

```
/* The two purposes for which ovn-northd uses OVN logical datapaths. */
enum ovn_datapath_type {
    DP_SWITCH, /* OVN logical switch. */
    DP_ROUTER /* OVN logical router. */
};
```

```
/* The two pipelines in an OVN logical flow table. */
enum ovn_pipeline {
    P_IN, /* Ingress pipeline. */
    P_OUT /* Egress pipeline. */
};
```

```
[root@ovn-control-plane ~]# ovn-sbctl lflow-list
Datapath: "ovn-worker2" (684a68ba-7d4c-4bac-9360-de8b47e52dd7) Pipeline: ingress
table=0 (ls_in_port_sec_12), priority=50, match=(inport == "default_client" && eth.src ==
{0a:58:0a:f4:00:05}), action=(next;
table=1 (ls_in_port_sec_ip), priority=90 , match=(inport == "default_client" && eth.src ==
0a:58:0a:f4:00:05 && ip4.src == {10.244.0.5}), action=(next;
table=19(ls_in_l2_lkup), priority=50 , match=(eth.dst == 0a:58:0a:f4:00:01), action=(output
= "stor-ovn-worker2"; output;)
```

OVN Packet Processing



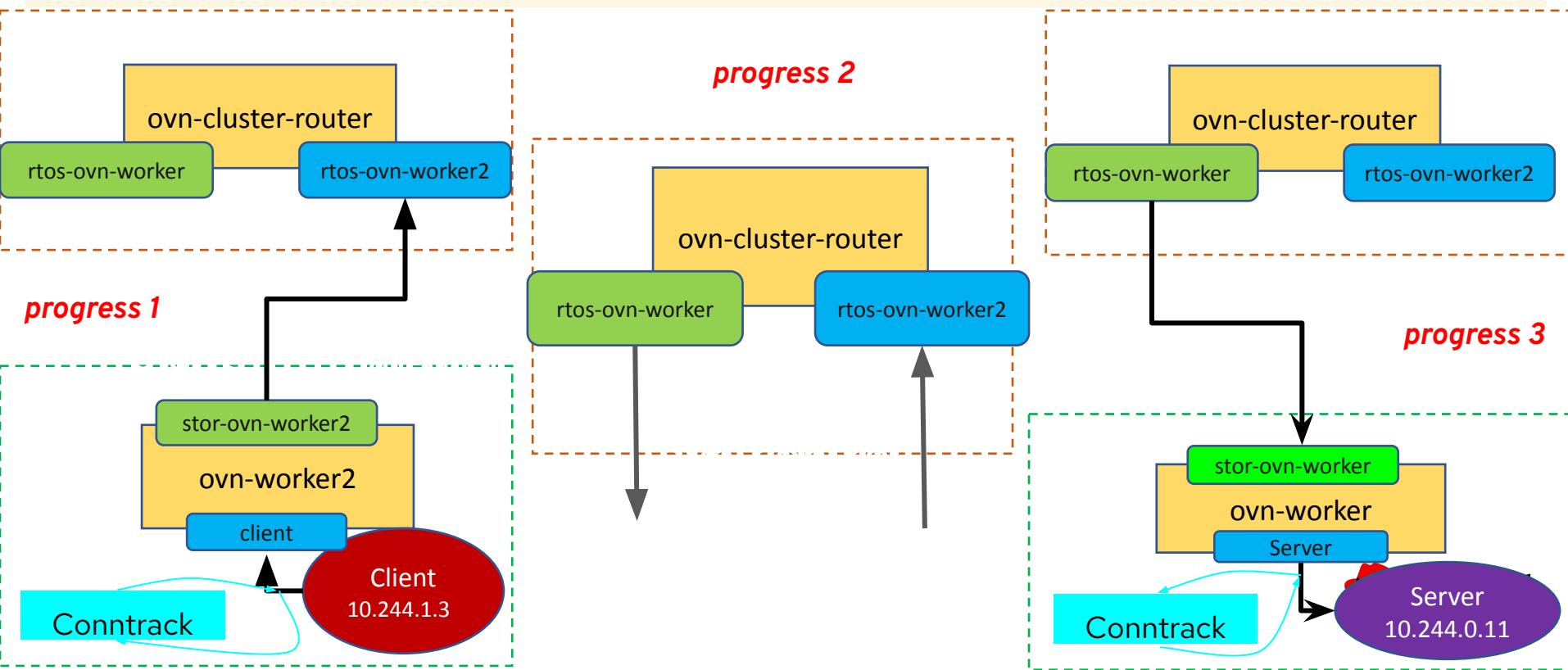
Ovn-Trace

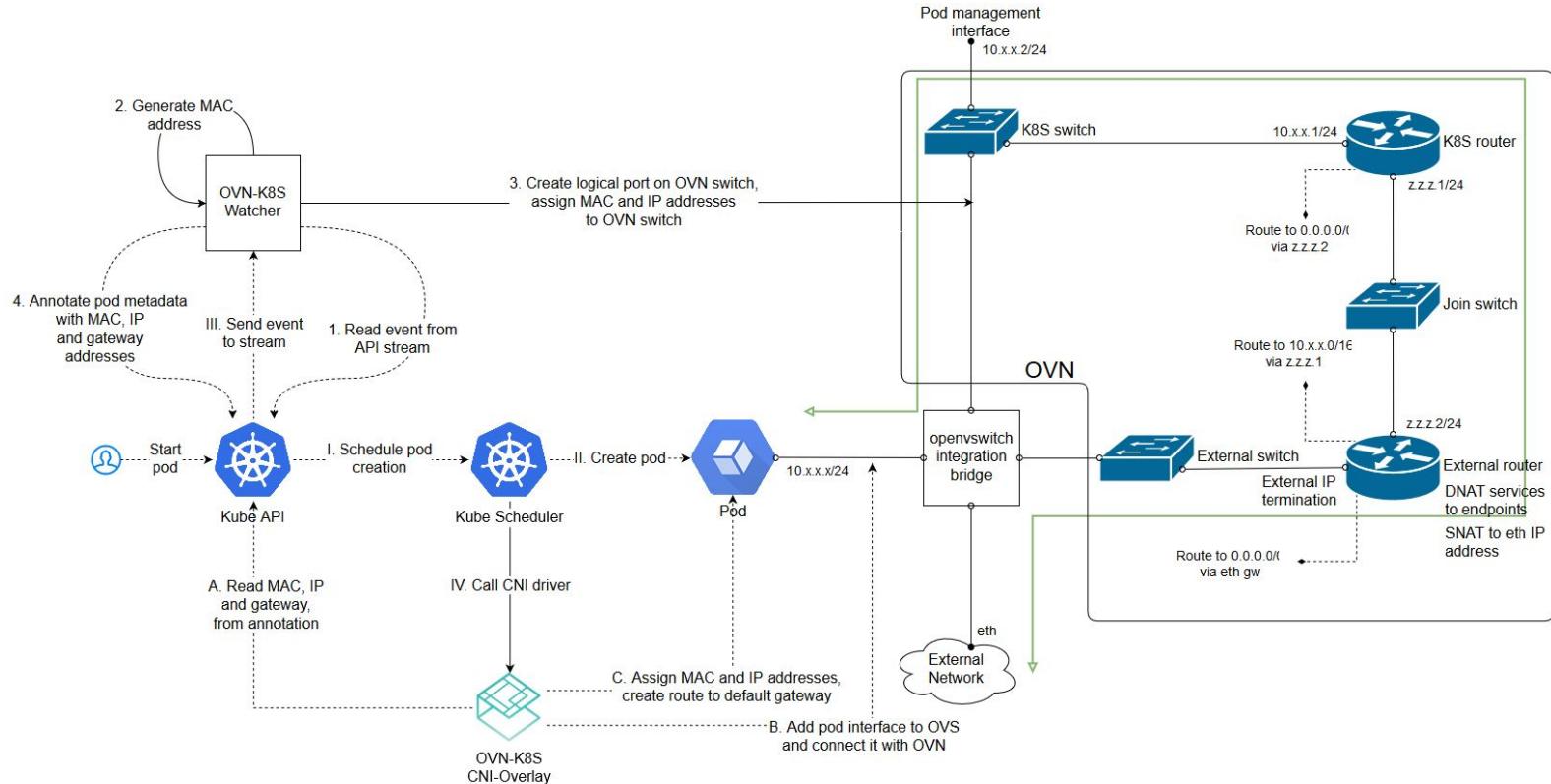
- 要想通过 OVN 追踪一个数据包，需要一个名为 ovn-trace 的工具。
 - 它模拟了 OVN 逻辑拓扑中的数据包转发场景
- A typical ovn-trace command:

```
[root@ovn-control-plane ~]# ovn-trace --ct new --ovs ovn-worker2  
'inport=="default_client" && eth.dst==0a:58:0a:f4:01:01 && eth.src==0a:58:0a:f4:01:03&&  
ip4.src==10.244.1.3 && ip4.dst==10.244.0.11 && ip.ttl==64 && tcp && tcp.src==80'
```

- **--ovs**
 - 使得 ovn-trace 试图获得并显示对应于每个 OVN 逻辑流的 Openflow flows, i.e connects to ovn-sbdb
- **import**
 - 要开始追踪数据包的端口
- **eth.dst and eth.src**
 - The destination of the next layer two hop
 - Here the MAC's relate to: default_client -> rtos-ovn-worker2
- **ip4.src and ip4.dst**
 - 客户端和服务器 pods 的 IP地址 分别为

```
[root@ovn-control-plane ~]# ovn-trace --ct new --ovs ovn-worker2 'inport=="default_client" && eth.dst==0a:58:0a:f4:01:01 && eth.src==0a:58:0a:f4:01:03
&& ip4.src==10.244.1.3 && ip4.dst==10.244.0.11 && ip.ttl==64 && tcp && tcp.src==8080' | grep -e ingress -e egress
ingress(dp="ovn-worker2", inport="default_client")
egress(dp="ovn-worker2", inport="default_client", outport="stor-ovn-worker2")
ingress(dp="ovn_cluster_router", inport="rtos-ovn-worker2")
egress(dp="ovn_cluster_router", inport="rtos-ovn-worker2", outport="rtos-ovn-worker")
ingress(dp="ovn-worker", inport="stor-ovn-worker")
egress(dp="ovn-worker", inport="stor-ovn-worker", outport="default_server")
[root@ovn-control-plane ~]#
```





↔ Traffic flow

I - IV Kubernetes pod creation flow

→ Web API call

1 - 4 Network settings generation flow

→ Non-web action

A - C Network settings application flow

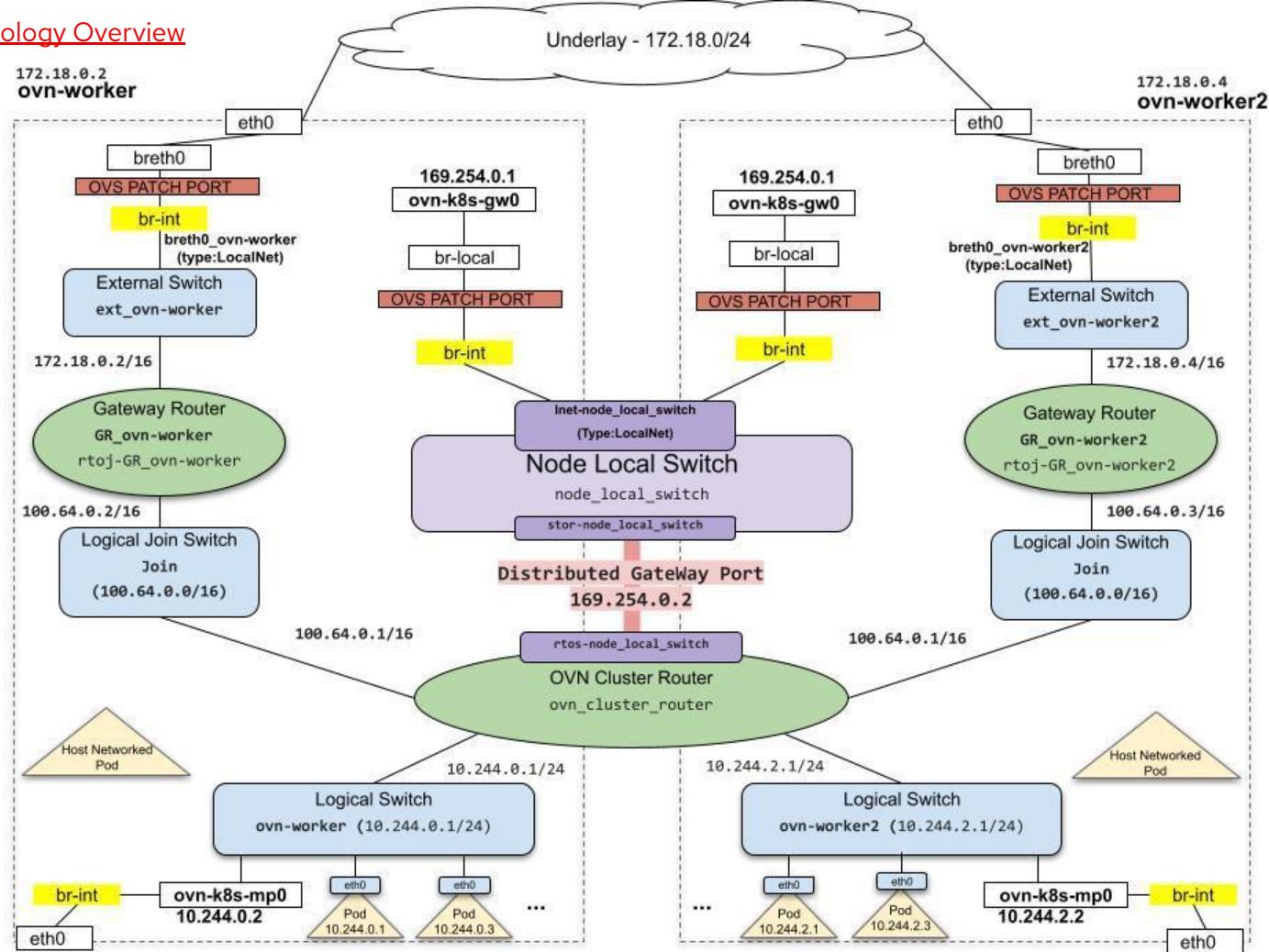
→ Route

● Host visible interface

○ Virtual interface

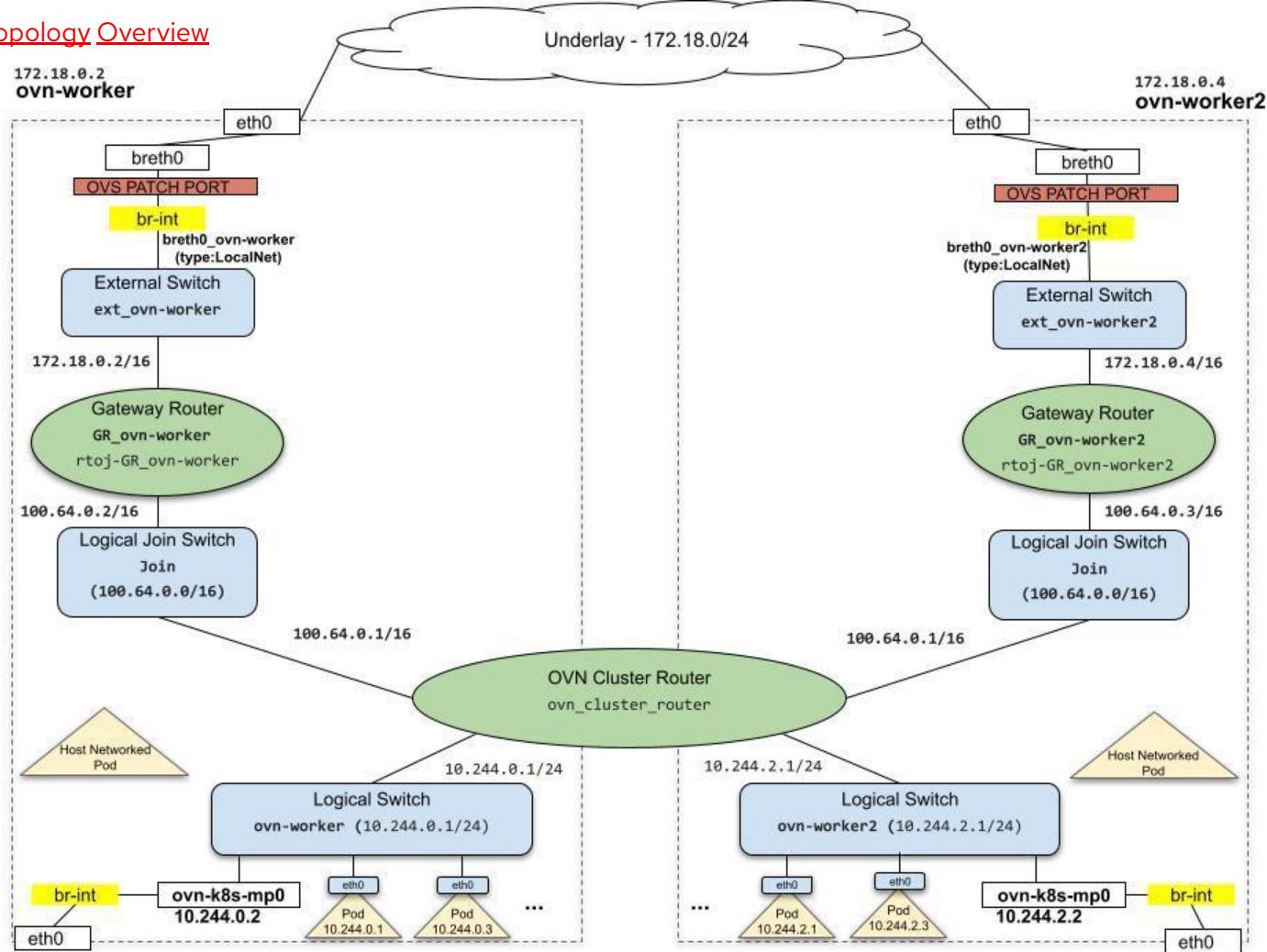
Local GW Topology Overview

Updated:
06/25/2021



Shared GW Topology Overview

Updated:
06/25/2021



Q & A

Reference

[KubeVirt User-Guide](#)

[OpenShift Container Platform 4.11 Documentation](#)

[OVN](#)

[How to use Open Virtual Networking with Kubernetes](#)

[OVN-Kubernetes-presentation-OCT0-October6th2020](#)

Reference - Learning day

[CNV Demo](#)

[Openshift/Container Training and Info](#)

[Openshift sandboxed container](#)

<https://docs.google.com/presentation/d/1U-KSJBPaMPp65v4RqDyFwqSMg9yegiG5/edit#slide=id.p6>

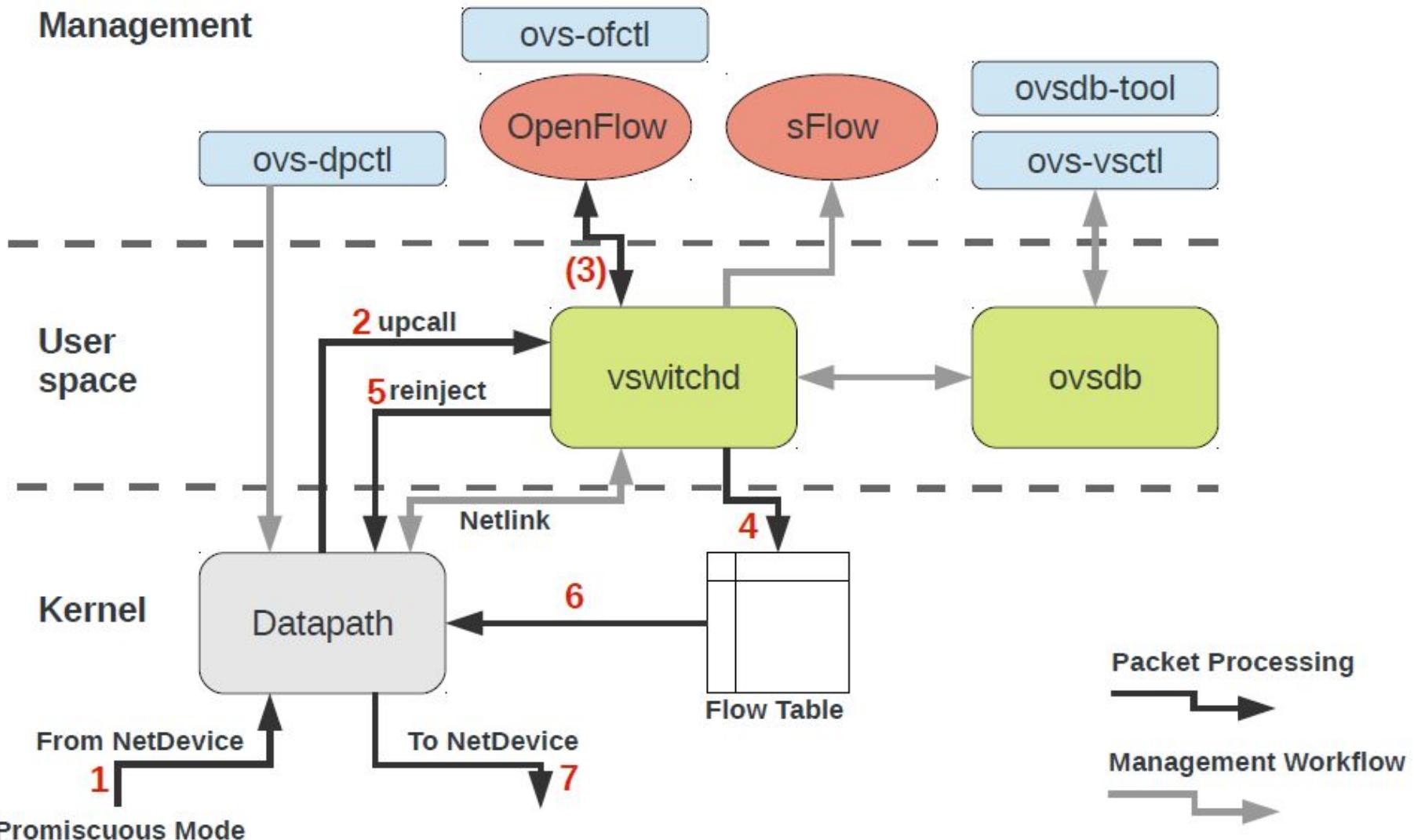
https://docs.google.com/presentation/d/1Xsyboav89hv-Xg3f6IuF9W-f5eR8_MYxZwyWK4Ra8RY/edit#slide=id.g91c10be5e6_0_87

OVS

OVS

Open vSwitch supports the following features:

- Visibility into inter-VM communication via NetFlow, sFlow(R), IPFIX, SPAN, RSPAN, and GRE-tunneled mirrors
- LACP (IEEE 802.1AX-2008)
- Standard 802.1Q VLAN model with trunking
- Multicast snooping
- IETF Auto-Attach SPBM and rudimentary required LLDP support
- BFD and 802.1ag link monitoring
- STP (IEEE 802.1D-1998) and RSTP (IEEE 802.1D-2004)
- Fine-grained QoS control
- Support for HFSC qdisc
- Per VM interface traffic policing
- NIC bonding with source-MAC load balancing, active backup, and L4 hashing
- OpenFlow protocol support (including many extensions for virtualization)
- IPv6 support
- Multiple tunneling protocols (GRE, VXLAN, STT, and Geneve, with IPsec support)
- Remote configuration protocol with C and Python bindings
- Kernel and user-space forwarding engine options
- Multi-table forwarding pipeline with flow-caching engine
- Forwarding layer abstraction to ease porting to new software and hardware platforms



OVS db

ovs - db schema

```
[root@edge-qe-per740-01 ~]# ovsdb-client list-dbs
```

Open_vSwitch

```
[root@edge-qe-per740-01 ~]# ovsdb-client list-tables Open_vSwitch
```

Table

Contro

Bridge

Queue

IPFIX

Net

Open

- 00S

Port

slow

SSI

Flow Sa

View_Sample

Flow Table

Row Table Interface

Interface AutoAttach

Reader Manager

```
[root@edge-qe-per740-01 ~]# cat /etc/openvswitch/conf.db | sed -n '2p' | jq | more
{
  "cksum": "3374030633 22987",
  "name": "Open_vSwitch",
  "version": "7.14.0",
  "tables": {
    "Controller": {
      "columns": {
        "is_connected": {
          "ephemeral": true,
          "type": "boolean"
        },
        "connection_mode": {
          "type": {
            "min": 0,
            "key": {
              "type": "string",
              "enum": [
                "set",
                [
                  "in-band",
                  "out-of-band"
                ]
              ]
            }
          }
        },
        "local_gateway": {
          "type": {
            "min": 0,
            "key": "string"
          }
        },
        "enable_async_messages": {
          "type": {
            "min": 0,
            "key": "boolean"
          }
        },
        "other_config": {
          "type": {
            "max": "unlimited",
            "min": 0,
            "key": "string",
            "value": "string"
          }
        }
      }
    }
  }
}
```

```
[root@edge-qe-per740-01 ~]# ovsdb-client get-schema Open_vSwitch | jq | more
{
  "cksum": "3374030633 22987",
  "name": "Open_vSwitch",
  "tables": {
    "AutoAttach": {
      "columns": {
        "mappings": {
          "type": {
            "key": {
              "maxInteger": 16777215,
              "minInteger": 0,
              "type": "integer"
            },
            "max": "unlimited",
            "min": 0,
            "value": {
              "maxInteger": 4095,
              "minInteger": 0,
              "type": "integer"
            }
          }
        },
        "system_description": {
          "type": "string"
        },
        "system_name": {
          "type": "string"
        }
      }
    },
    "Bridge": {
      "columns": {
        "auto_attach": {
          "type": {
            "key": {
              "refTable": "AutoAttach",
              "type": "uuid"
            },
            "min": 0
          }
        },
        "controller": {
          "type": {
            "key": {
              "refTable": "Controller",
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```

```
root@edge-qe-per740-01:~ (ssh) root@edge-qe-per740-01:~ # ovsdb-client dump
AutoAttach table
_uuid mappings system_description system_name
-----
Bridge table
_uuid auto_attach controller datapath_id datapath_type datapath_version external_ids fail_mode flood_vlans flow_tables ipfix mc
ast_snooping_enable mirrors name netflow other_config ports protocols rstp_enable rstp_status sflow status stp_enable
-----
-----  
b61e977d-3251-4e54-a287-d126d6746ae5 [] [] "00007e971eb6544e" "" <unknown> {} [] [] {} [] fa
lse [] "br2" [] {} [36361a0e-e3a8-43ec-a088-b74e78d6e386, 4a2bc419-f9af-4abb-8c6c-8f6d843ab500, b9324926-fb18-4f3b-9084-1cfdf7736
d50, cc0dd740-af32-43b4-b644-1fe742599f3e, f7f390bc-a7eb-43df-807f-97f50fa37e25] [] false {} [] {} false
4ba0cfda-db16-4eaf-a0d5-9fe10add644 [] [] "0000dacfa04ba4e" "" <unknown> {} [] [] {} [] fa
lse [] "br1" [] {} [5e8a1f44-b8ac-42f0-a926-869888f7299d, 76b7656f-3768-4177-87bf-5c60dc006b98, 8053a6b0-582d-4951-8d38-12684913b
b8f, ce380ea6-3a64-484f-b72c-ba134b01b241, df50366b-9cc1-4dcf-b4e9-ef072a49ef6d] [] false {} [] {} false
Controller table
_uuid connection_mode controller_burst_limit controller_rate_limit enable_async_messages external_ids inactivity_probe is_connected local_gateway local_ip local_netm
ask max_backoff other_config role status target
-----
Flow_Sample_Collector_Set table
_uuid bridge external_ids id ipfix
-----
Flow_Table table
_uuid external_ids flow_limit groups name overflow_policy prefixes
-----
IPFIX table
_uuid cache_active_timeout cache_max_flows external_ids obs_domain_id obs_point_id other_config sampling targets
-----
Interface table
_uuid admin_state bfd bfd_status cfm_fault cfm_fault_status cfm_flap_count cfm_health cfm_mpid cfm_remote_mpids cfm_remote_opstate dup
lex error external_ids ifin
dex ingress_policing_burst ingress_policing_rate lacp_current link_resets link_speed link_state lldp mac mac_in_use mtu mtu_request name ofport ofport_r
equest options other_config statistics
status type
-----
```

OVS - db CRUD

```
ovs-vsctl add-br helloworld  
ip link add first_br type veth peer name first_if  
ip link set first_br up  
ip link set first_if up  
ovs-vsctl add-port helloworld first_br
```

list table

```
ovs-vsctl list Bridge
```

list table [record]

```
ovs-vsctl list Bridge helloworld
```

get table record [column[:key]]

```
ovs-vsctl get Bridge helloworld ports
```

set table record column[:key]=value

```
ovs-vsctl set Port first_br tag=100
```

add table record column [key]=value

```
ovs-vsctl add Port first_br trunks 110
```

remove table record column key

```
ovs-vsctl remove Port first_br trunks 100
```

clear table record column

```
ovs-vsctl clear Port first_br trunks
```