KubeVirt

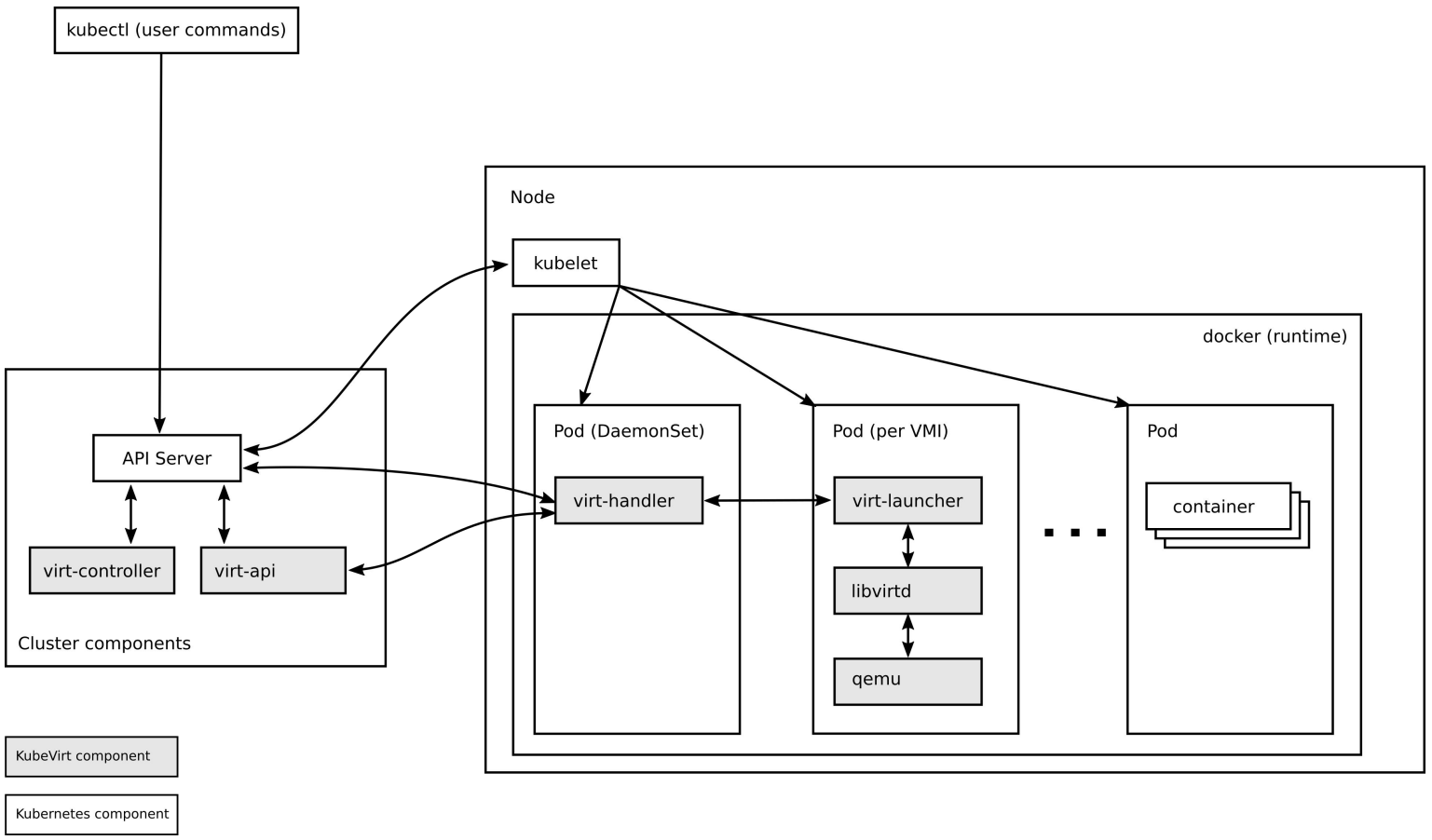
2020.8.2

https://github.com/actor168

KubeVirt整体架构

virt-api：接收API请求
virt-controller：KubeVirt控制器，控制对vmi的创建、删除等管理
virt-handler：Node上的常驻进程，接收virt-launcher消息，并监听virt-launcher状态
virt-launcher：运行libvirtd进程，承载真正vm运行，和vm同生命周期

创建VM时各组件交互过程
1、virt-api 创建VMI CRD对象
2、virt-controller监听到VMI创建时，会根据VMI配置生成pod spec文件，通知virt-handler创建virt-launcher pods
3、virt-controller发现virt-launcher pod创建完毕后，更新VMI CRD状态
4、virt-handler监听到VMI状态变更，通信virt-launcher去创建虚拟机，并负责虚拟机生命周期管理



KubeVirt支持能力

资源管理
CPU/Mem/Disk-支持超配

生命周期管理
支持创建、删除、查询、停止、暂停、重命名、热迁移**7**种API和vmctl操作

其他功能
支持VNC和串行接口
支持K8s DNS集成
支持容器网络集成、网络策略配置
支持虚拟机副本集、DaemonSet等
支持vmctl命令行操作
支持虚拟机模板
支持x86和q35（win）指令集模拟
支持cpu槽拓扑配置
支持HugePage
支持Windows Virtio驱动
支持GPU直通
支持健康检查、监控等

优点：
1、原生支持K8s实现了灵活的资源调度，比如节点驱逐、副本集、DaemonSet等
2、原生支持了K8S容器网络、CNI
3、原生支持K8S存储，挂载方式和类型多样
4、libvirtd进程被封装进了virt-launcher容器，无需担心主机底层依赖配置问题
5、原生支持K8s RBAC对虚机的操纵权限控制

缺点：
1、调用libvirt的能力受VMI CRD实现的限制，如果CRD尚未实现某些能力，导致已有的libvirtd能力将不可用
2、网络不支持固定ip分配
3、cloud-init支持不完善，仅支持NoCloud注入，不支持Metaserver动态注入和修改
4、网络接入的能力有限，受限于K8S CNI规范下支持的类型
5、镜像管理不方便，需要安装VDI，命令行操作，用户使用门槛高

KubeVirt网络

自定义两个概念，Backend和Frontend

- Backend--指Pod使用的网络模式
  - 采用Pod模式（K8s默认网络）
  - 采用Multus模式（多网卡）
  - 采用Genie模式
- Frontend--指虚拟网卡和Pod网络的连接方式
  - 采用Pod内的Bridge网络模式
    - 将vm对应的虚拟网卡vnet0桥接到libvirt提供的k6t-eth0网桥上
    - k6t-eth0通过virt-launcher容器veth网卡对外连接访问
  - 采用slirp/sriov/masquerade方式

- 如何实现的虚机外对内的访问？
  - Service暴露
  - Pod直连

```yaml
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: testvm
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/size: small
        kubevirt.io/domain: testvm
    spec:
      domain:
        devices:
          disks:
          - disk:
              bus: virtio
            name: rootfs
          - disk:
              bus: virtio
            name: cloudinit
          interfaces:
          - name: default
            bridge: {}
        resources:
          requests:
            memory: 64M
      networks:
      - name: default
        pod: {}
      volumes:
        - name: rootfs
          containerDisk:
            image: kubevirt/cirros-registry-disk-demo
        - name: cloudinit
          cloudInitNoCloud:
```

KubeVirt存储

- Pod->VM
  - 支持virtio/sata挂载
- 宿主->Pod->VM
  - 宿主到virt-launcher采用容器存储挂载的方式
- 性能考虑?
  - 存在两层挂载，性能不佳

- PV
- dataVolume
- containerDisk
- hostDisk
- ...

```
<disk type='file' device='disk'>
  <driver name='qemu' type='qcow2' cache='none'/>
  <source file='/var/run/kubevirt-ephemeral-disks/disk-data/rootfs/disk.qcow2' index='2'/>
  <backingStore type='file' index='3'>
    <format type='qcow2'/>
    <source file='/var/run/kubevirt/container-disks/disk_0.img'/>
    <backingStore/>
  </backingStore>
  <target dev='vda' bus='virtio'/>
  <alias name='ua-rootfs'/>
  <address type='pci' domain='0x0000' bus='0x03' slot='0x00' function='0x0'/>
</disk>
<disk type='file' device='disk'>
  <driver name='qemu' type='raw' cache='none'/>
  <source file='/var/run/kubevirt-ephemeral-disks/cloud-init-data/default/testvm/noCloud.iso' index='1'/>
  <backingStore/>
  <target dev='vdb' bus='virtio'/>
  <alias name='ua-cloudinit'/>
  <address type='pci' domain='0x0000' bus='0x04' slot='0x00' function='0x0'/>
</disk>
<controller type='usb' index='0' model='none'>
  <alias name='usb'/>
</controller>
<controller type='virtio-serial' index='0'>
  <alias name='virtio-serial0'/>
  <address type='pci' domain='0x0000' bus='0x02' slot='0x00' function='0x0'/>
</controller>
<controller type='sata' index='0'>
  <alias name='ide'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x1f' function='0x2'/>
</controller>
```

KubeVirt网络1-Bridge

- VM使用K8s给Pod分配的IP地址，可以直接与其他容器、VM直通，可复用K8S内置DNS

```
# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
    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,UP,LOWER_UP> mtu 1376 qdisc pfifo_fast qlen 1000
    link/ether 12:d5:eb:b1:95:57 brd ff:ff:ff:ff:ff:ff
    inet 10.32.0.12/24 brd 10.32.0.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::10d5:ebff:feb1:9557/64 scope link tentative flags 08
       valid_lft forever preferred_lft forever
```

vm

```
    <interface type='bridge'>
      <mac address='12:d5:eb:b1:95:57'/>
      <source bridge='k6t-eth0'/>
      <model type='virtio'/>
      <mtu size='1376'/>
      <alias name='ua-default'/>
      <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
    </interface>
```

```
# cat /etc/resolv.conf
search default.svc.cluster.local
nameserver 10.96.0.10
```

```
[root@testvm /]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    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
2: k6t-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1376 qdisc noqueue state UP group default
    link/ether 12:d5:eb:fd:b9:52 brd ff:ff:ff:ff:ff:ff
    inet 169.254.75.10/32 brd 169.254.75.10 scope global k6t-eth0
       valid_lft forever preferred_lft forever
3: vnet0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1376 qdisc pfifo_fast master k6t-eth0 state UNKNOWN group default qlen 1000
    link/ether fe:d5:eb:b1:95:57 brd ff:ff:ff:ff:ff:ff
34: eth0@if35: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1376 qdisc noqueue master k6t-eth0 state UP group default
    link/ether 12:d5:eb:fd:b9:52 brd ff:ff:ff:ff:ff:ff link-netnsid 0
```

virt-launcher

KubeVirt网络2-Pod+NAT

VM之间需要进行NAT转发，方可互访

```
<interface type='bridge'>
  <mac address='02:00:00:a5:93:1c'/>
  <source bridge='k6t-eth0'/>
  <target dev='vnet0'/>
  <model type='virtio'/>
  <mtu size='1376'/>
  <alias name='ua-default'/>
  <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
</interface>
```

```
sh-5.0# brctl show
bridge name     bridge id               STP enabled     interfaces
k6t-eth0                8000.964e303a4203       no              k6t-eth0-nic
                                                        vnet0
```

virt-launcher

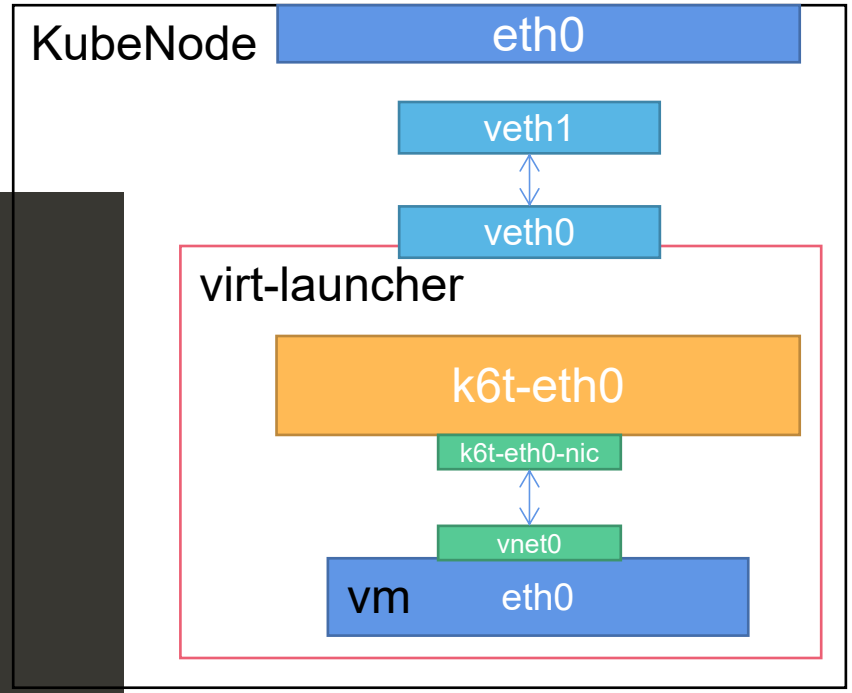
```
sh-5.0# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: k6t-eth0-nic: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue master k6t-eth0 state UNKNOWN group default
    link/ether 96:4e:30:3a:42:03 brd ff:ff:ff:ff:ff:ff
3: k6t-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1376 qdisc noqueue state UP group default
    link/ether 96:4e:30:3a:42:03 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
4: vnet0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1376 qdisc pfifo_fast master k6t-eth0 state UNKNOWN group default qlen 1000
    link/ether fe:00:00:a5:93:1c brd ff:ff:ff:ff:ff:ff
34: eth0@if35: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1376 qdisc noqueue state UP group default
    link/ether 76:07:9f:9f:c5:b4 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.32.0.12/24 brd 10.32.0.255 scope global eth0
       valid_lft forever preferred_lft forever
```

virt-launcher

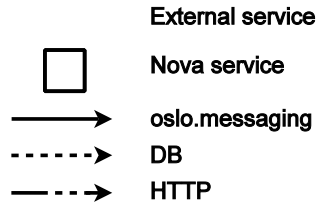
```
# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1376 qdisc pfifo_fast qlen 1000
    link/ether 02:00:00:a5:93:1c brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.2/24 brd 10.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::ff:fea5:931c/64 scope link
       valid_lft forever preferred_lft forever
```

testvm

```
[root@testvm /]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.32.0.1       0.0.0.0         UG    0      0        0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U     0      0        0 k6t-eth0
10.32.0.0       0.0.0.0         255.255.255.0   U     0      0        0 eth0
```



OpenStack Nova

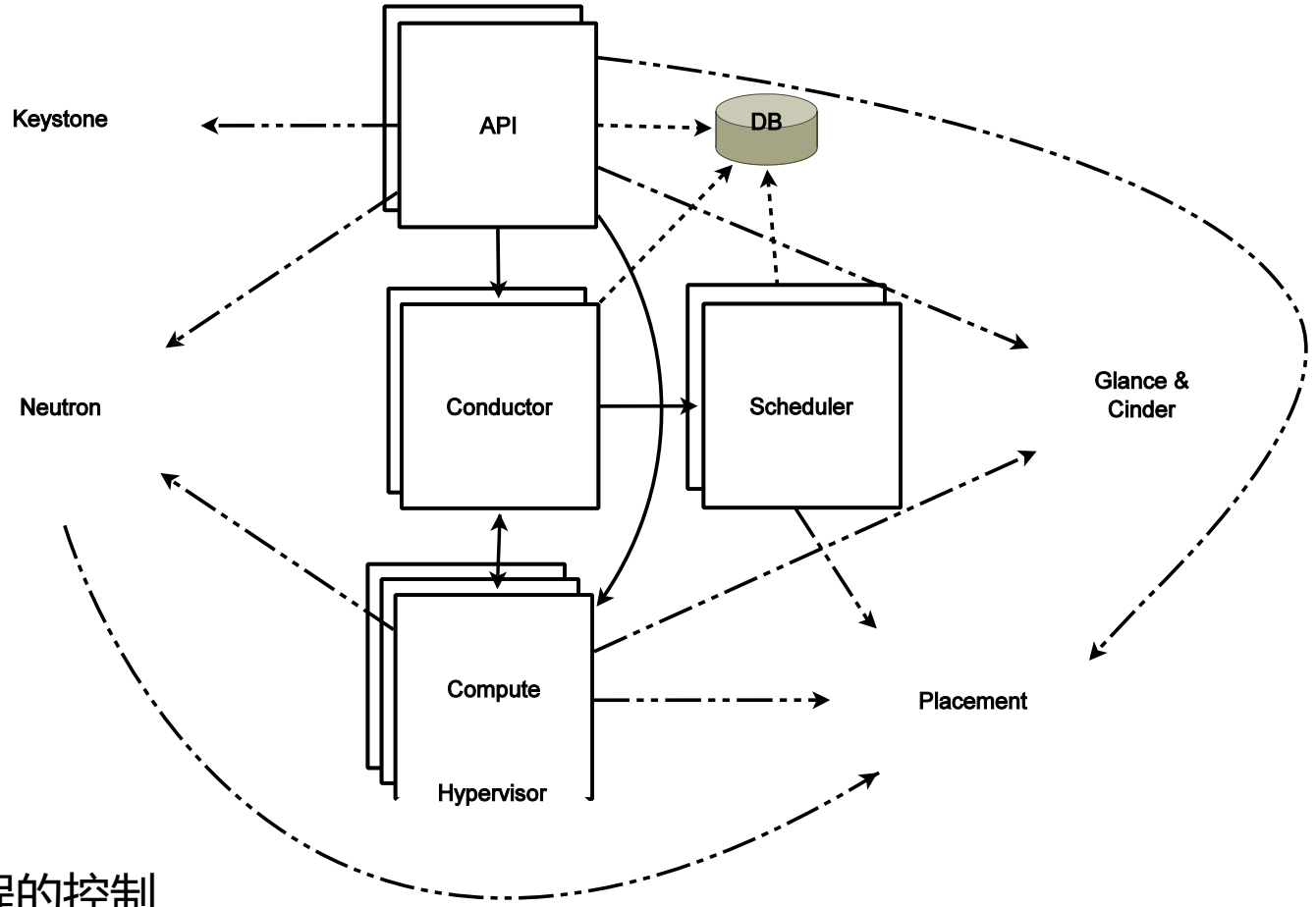


资源管理
CPU/Mem/Disk-支持超配、租户级资源管控

Nova API接口
接口丰富，支持40+种操作，功能完善

其他特性
支持安全组、多IP绑定、规格管理、密钥管理
元数据管理、聚合、配额、标签、事件汇报、
VNC/Spice/RDP/串口

生命周期管理
增删改查、维护迁移、快照、备份、
Resize、重建、重置、锁定等



计算各组件介绍
nova-api：对外暴露Restful服务接口
nova-conductor：代理访问数据库，并且完成复杂流程的控制
nova-scheduler：提供虚拟机的调度服务，分配算法：①过滤②权重③随机
nova-compute：虚拟机生命周期管理，通过调用Libvirt API
nova-novncproxy：novnc 访问虚拟机代理