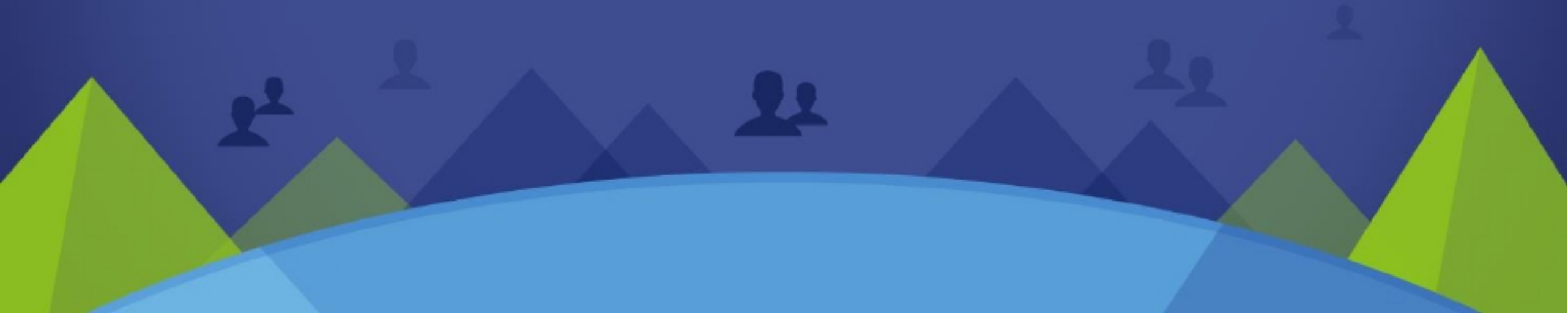
OpenStack Nova 计算服务

哈尔滨工业大学（威海）计算机科学与技术学院 朴学峰



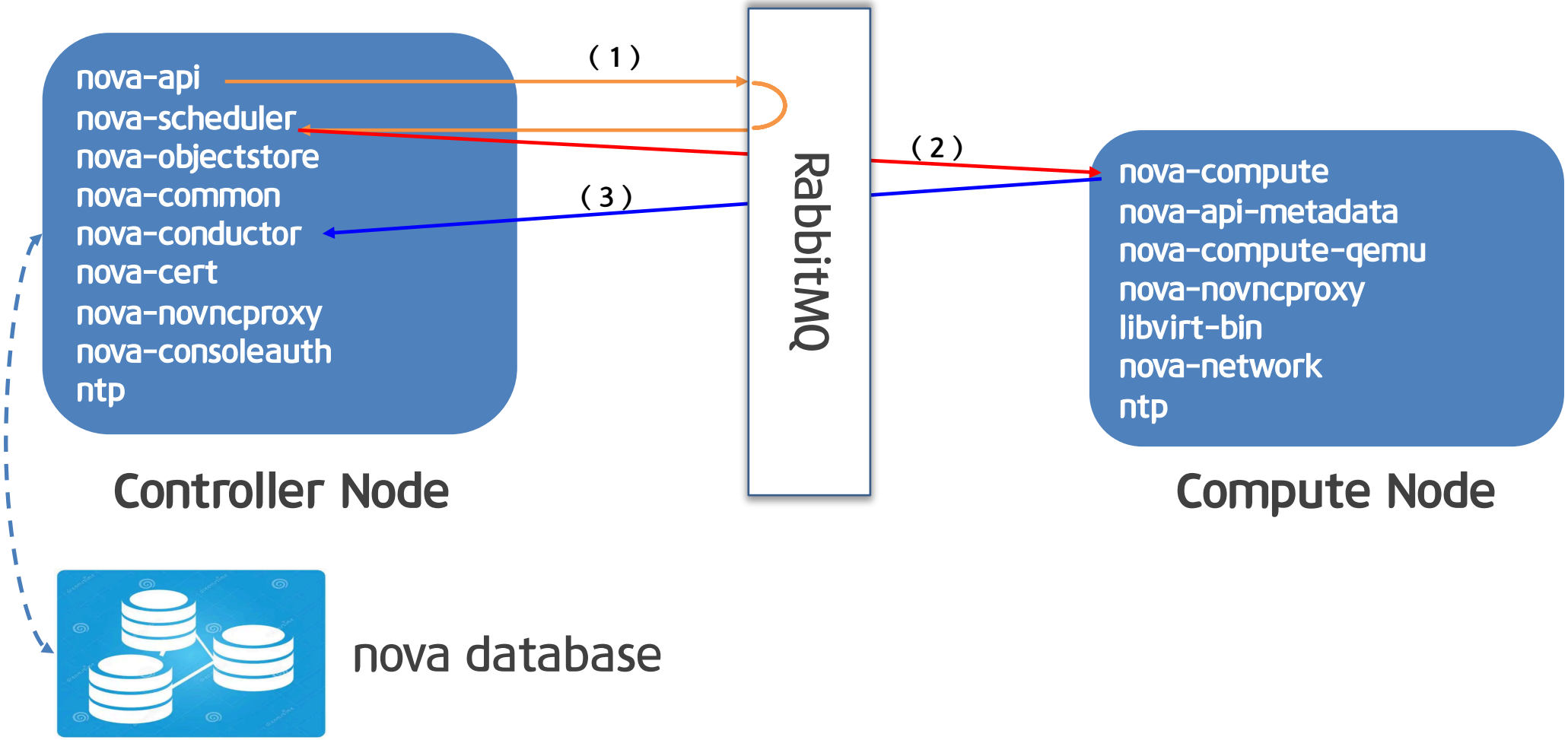
内容导航
CONTENTS

1. 安装 OpenStack 计算服务控制节点服务
2. 安装 OpenStack 计算软件包
3. 配置数据库
4. 配置 OpenStack 计算服务
5. 使用 OpenStack 身份认证服务配置计算
6. 停止和启动 Nova 服务

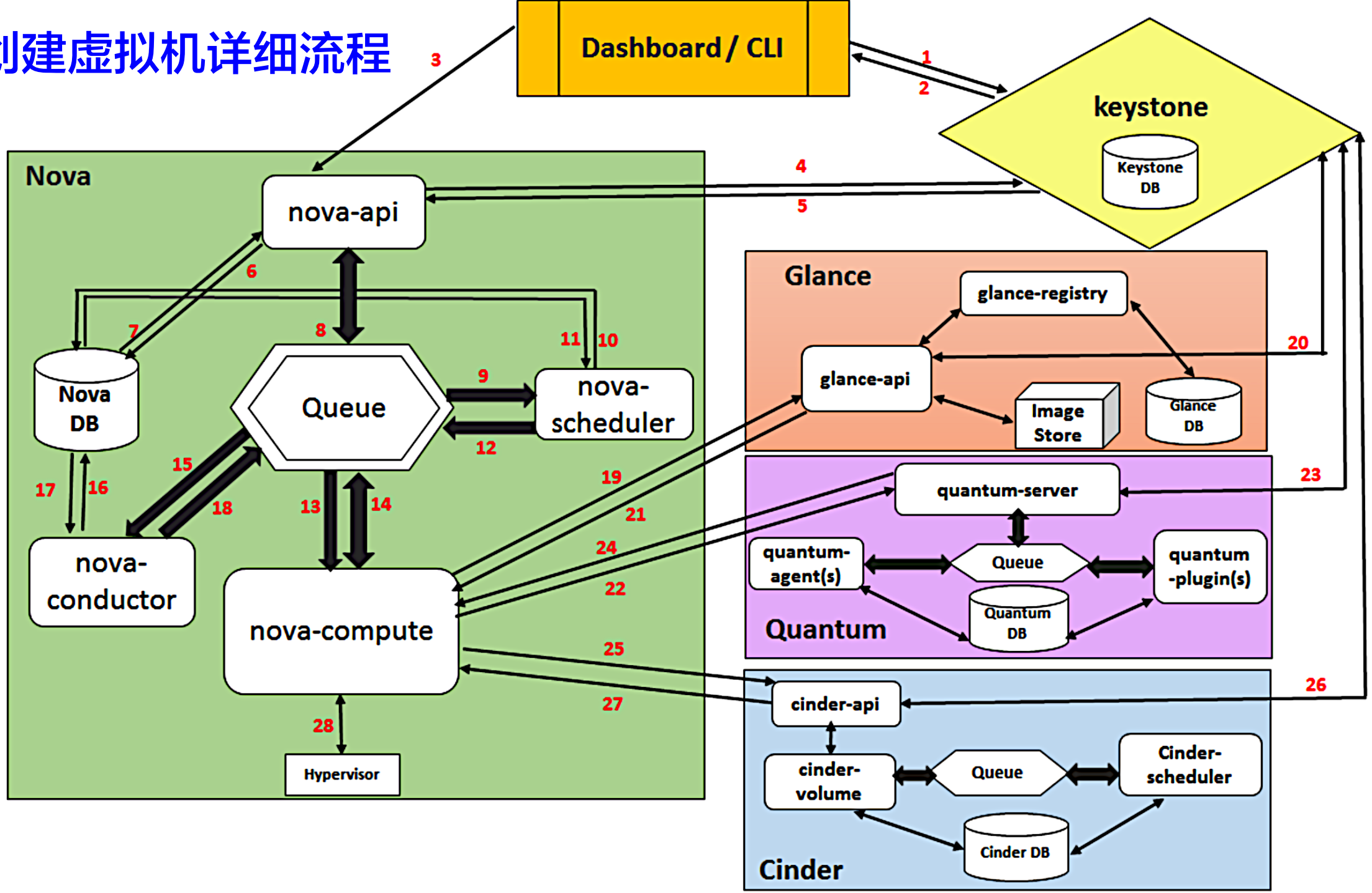
Nova 是 openstack 最核心的服务，负责维护和管理云环境的计算资源，同时管理虚拟机的生命周期

1. 客户向 nova-api 发送请求：我需要创建一个虚拟机；
2. nova-api 对请求进行响应处理，向 RabbitMQ 发送了一条消息：“让 Scheduler 创建一个虚拟机”；
3. nova-scheduler 监听 rabbitMQ 获取到 nova-api 发给它的消息，然后执行调度算法，从若干计算节点中选出合适的计算节点 A（物理节点）；
4. nova-scheduler 向 rabbitMQ 发送一条消息：“在计算节点 A 上创建这个虚拟机”；
5. 计算节点 A 的 nova-compute 从 rabbitMQ 中获取到 nova-scheduler 发给它的消息，然后在本节点的 Hypervisor 上启动虚拟机。
6. 在虚拟机创建的过程中，nova-compute 如果需要查询或更新数据库信息，会通过消息队列向 nova-conductor 发送消息，nova-conductor 负责数据库访问。

实践中需要 Controller Node + Compute Node 完成


nova-api
nova-scheduler
nova-objectstore
nova-common
nova-conductor
nova-cert
nova-novncproxy
nova-consoleauth
ntp
(1)
(3)
RabbitMQ
(2)
nova-compute
nova-api-metadata
nova-compute-qemu
nova-novncproxy
libvirt-bin
nova-network
ntp
Controller Node
Compute Node
nova database

创建虚拟机详细流程


Dashboard / CLI
keystone
Keystone DB
Nova
nova-api
Nova DB
Queue
nova-scheduler
nova-conductor
nova-compute
Hypervisor
Glance
glance-registry
glance-api
Image Store
Glance DB
Quantum
quantum-server
quantum-agent(s)
Queue
quantum -plugin(s)
Quantum DB
Cinder
cinder-api
cinder-volume
Queue
Cinder-scheduler
Cinder DB
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

1. 界面或命令行通过RESTful API向keystone获取认证信息。
2. keystone通过用户请求认证信息，正确后生成token返回给对应的认证请求。
3. 界面或命令行通过RESTful API向nova-api发送一个创建虚拟机的请求（携带token）。
4. nova-api接受请求后向keystone发送认证请求，查看token是否为有效用户。
5. keystone验证token是否有效，如有效则返回有效的认证和对应的角色（注：有些操作需要有角色权限才能操作）。
6. 通过认证后nova-api检查创建虚拟机参数是否有效合法后和数据库通讯。
7. 当所有的参数有效后初始化新建虚拟机的数据库记录。
8. nova-api通过rpc.call向nova-scheduler请求是否有创建虚拟机的资源(Host ID)。
9. nova-scheduler进程侦听消息队列，获取nova-api的请求。
10. nova-scheduler通过查询nova数据库中计算资源的情况，并通过调度算法计算符合虚拟机创建需要的主机。

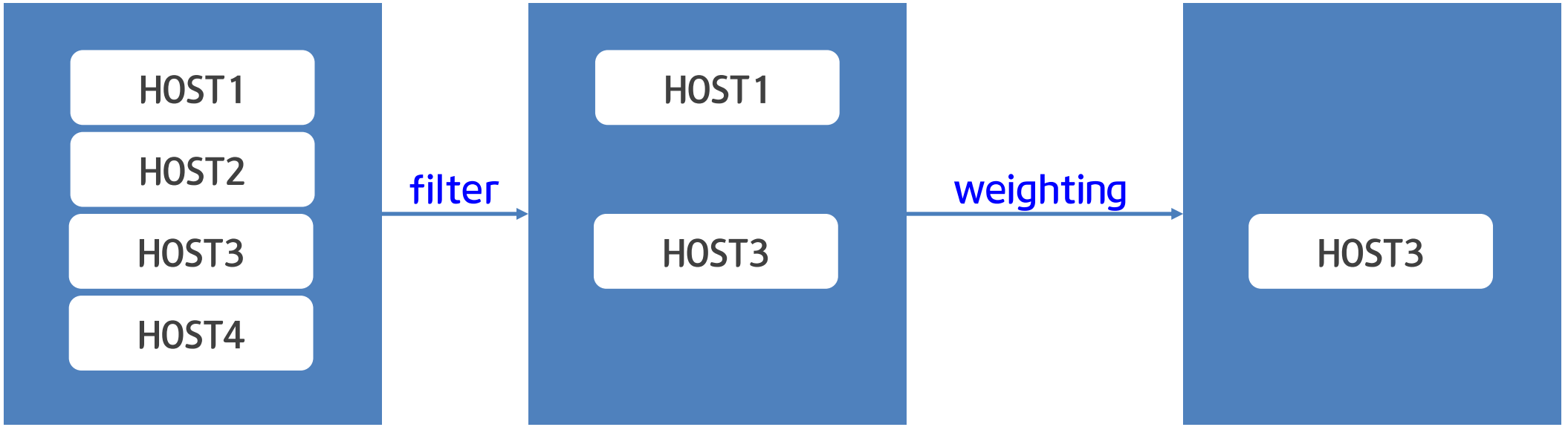
11. 对于有符合虚拟机创建的主机，nova-scheduler更新数据库中虚拟机对应的物理主机信息。
12. nova-scheduler通过rpc.cast向nova-compute发送对应的创建虚拟机请求的消息。
13. nova-compute会从对应的消息队列中获取创建虚拟机请求的消息。
14. nova-compute通过rpc.call向nova-conductor请求获取虚拟机消息。
15. nova-conductor从消息队队列中拿到nova-compute请求消息。
16. nova-conductor根据消息查询虚拟机对应的信息。
17. nova-conductor从数据库中获得虚拟机对应信息。
18. nova-conductor把虚拟机信息通过消息的方式发送到消息队列中。
19. nova-compute从对应的消息队列中获取虚拟机信息消息。
20. nova-compute通过keystone的RESTfull API拿到认证的token，并通过HTTP请求glance-api获取创建虚拟机所需要镜像。

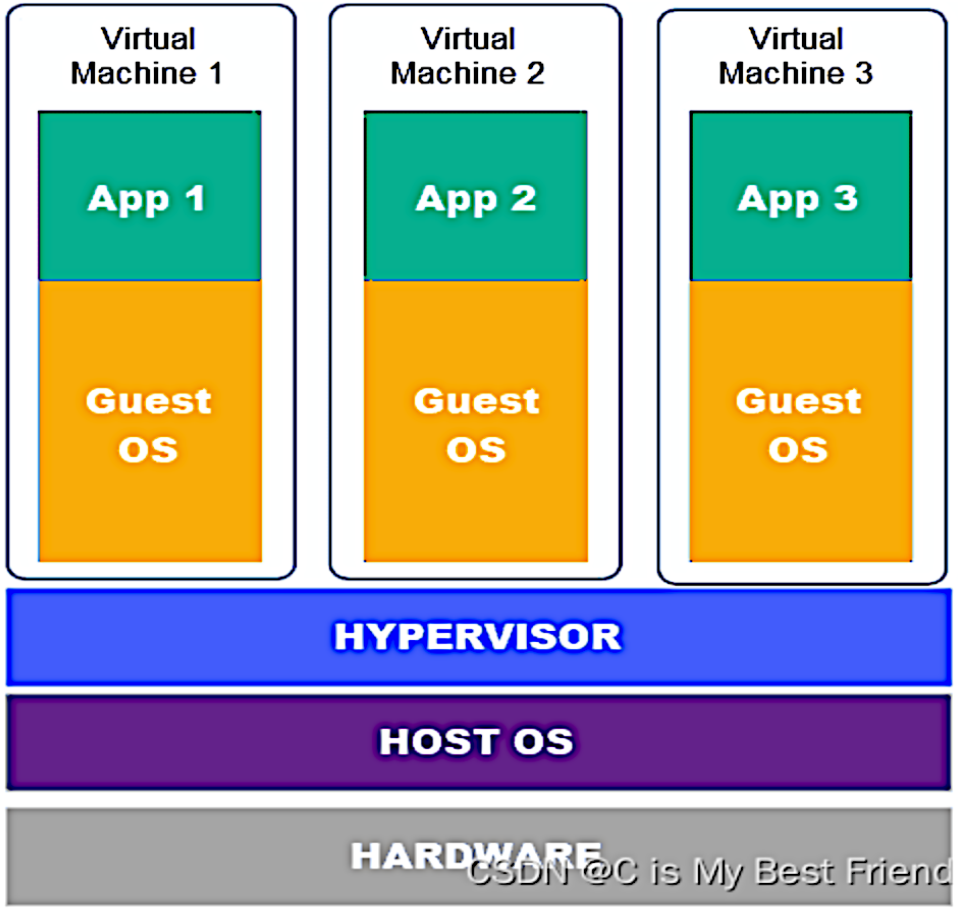
21. glance-api向keystone认证token是否有效，并返回验证结果。
22. token验证通过，nova-compute获得虚拟机镜像信息(URL)。
23. nova-compute通过keystone的RESTfull API拿到认证k的token，并通过HTTP请求neutron-server获取创建虚拟机所需要的网络信息。
24. neutron-server向keystone认证token是否有效，并返回验证结果。
25. token验证通过，nova-compute获得虚拟机网络信息。
26. nova-compute通过keystone的RESTfull API拿到认证的token，并通过HTTP请求cinder-api获取创建虚拟机所需要的持久化存储信息。
27. cinder-api向keystone认证token是否有效，并返回验证结果。
28. token验证通过，nova-compute获得虚拟机持久化存储信息。 nova-compute根据instance的信息调用配置的虚拟化驱动来创建虚拟机。

Nova Scheduler的作用就是进行云主机的调度，即选择哪一个物理主机创建虚拟机，调度过程分为两步：

（1）过滤：根据虚拟机资源配置情况和各个主机的实际情况，过滤掉一些不符合条件的主机

（2）权重：进行过滤后，对剩余的主机进行权重计算，根据权重选择最优物理主机



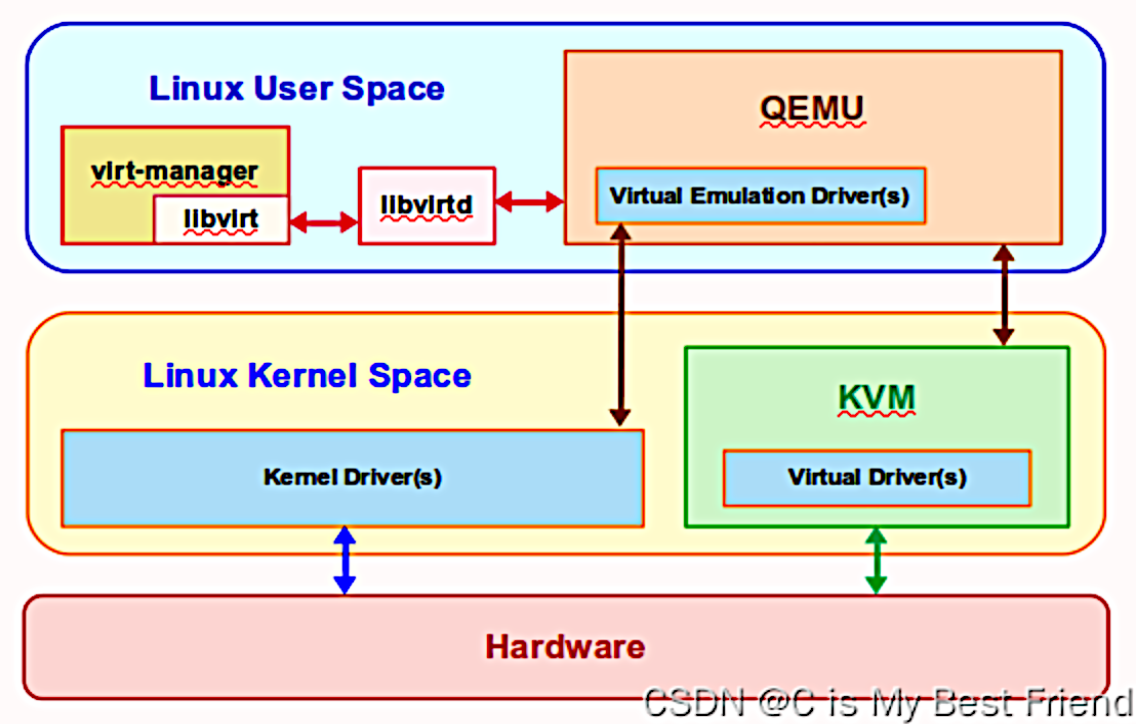


虚拟机层次结构示意图

Hypervisor（超级监督者）是介于虚拟机和宿主机之间，负责对宿主机资源进行虚拟化（virtual machine monitor:VMM），并将虚拟化的资源提供给虚拟机。分为两类：

1. Hypervisor 位于操作系统内核空间，可以直接访问宿主机的硬件资源。代表有 Microsoft Hyper-V、Linux KVM（Kernal-based Virtual Machine）、VMWare ESXi。
2. Hypervisor 位于用户空间，需要通过宿主机的操作系统才能访问硬件资源。代表有 QEMU、Oracle Virtual Box。

Quick EMUlator： A generic and open source machine emulator and virtualizer
一个通用的开源计算机仿真器和虚拟器



QEMU有两种工作模式

1. QEMU 与 Linux 内核交互，从而获取虚拟化的资源。
2. QEMU 与 Linux KVM 交互，仅负责传递交互信息，而不发挥Hypervisor的实际功能。

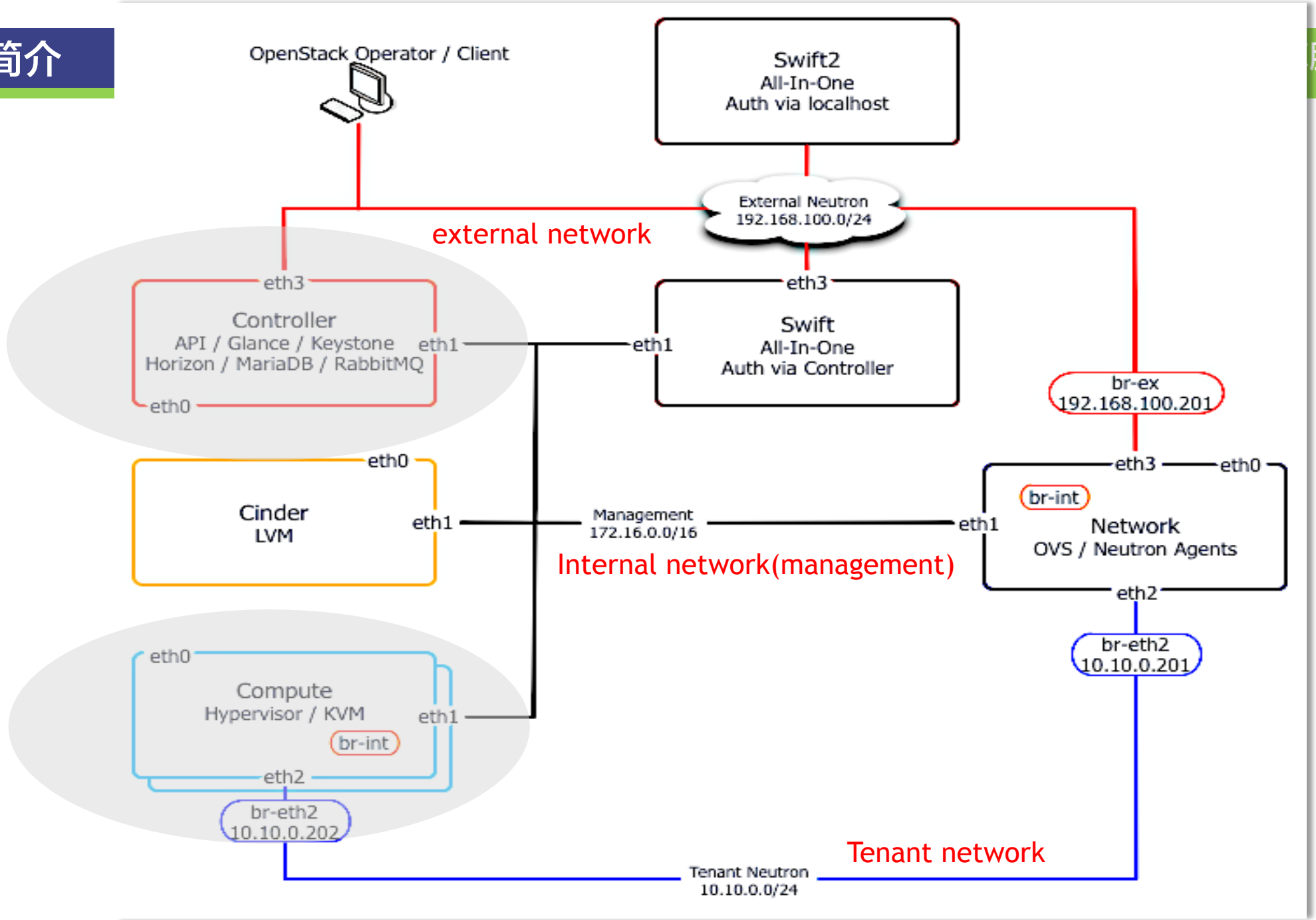
第一种工作模式是用于不需要与硬件交互的场景，第二种工作模式用于需要与硬件交互的场景。

- libvirt 是管理虚拟化平台的开源 API，后台程序和管理工具。可以支持多种虚拟平台
- libvirtd 是 libvirt 的守护进程，提供了一系列在虚拟机之间进行资源调度的接口，这些接口底层调用了 QEMU
- virt-manager 是 libvirt 的图形化界面，在 virt-manager 上的所有操作，都会被映射到 libvirt，libvirt 又会被映射到 QEMU

从虚拟机分层视觉来看，virt-manager、libvirt、QEMU 与 KVM 都属于 Hypervisor 层次。前三者处于用户空间，KVM 处于内核空间

▸▸▸ 支持的虚拟化平台

- LXC ： lightweight Linux container system
- OpenVZ ：lightweight Linux container system
- Kernel-based Virtual Machine/QEMU (KVM) ： open-source hypervisor for Linux and SmartOS
- Xen ： Bare-Metal hypervisor
- User-mode Linux (UML) ：paravirtualized kernel
- VirtualBox ： hypervisor by Oracle (formerly by Sun) for Windows, Linux, Mac OS X, and Solaris
- VMware ESX and GSX ： hypervisors for Intel hardware
- VMware Workstation and Player ： hypervisors for Windows and Linux
- Hyper-V ： hypervisor for Windows by Microsoft
- PowerVM ： hypervisor by IBM for AIX, Linux and IBM i
- Parallels Workstation ： hypervisor for Mac by Parallels IP Holdings GmbH
- Bhyve ： hypervisor for FreeBSD 10


OpenStack Operator / Client
Swift2
All-In-One
Auth via localhost
External Neutron
192.168.100.0/24
external network
eth3
Controller
API / Glance / Keystone
Horizon / MariaDB / RabbitMQ
eth1
eth0
eth1
eth3
Swift
All-In-One
Auth via Controller
br-ex
192.168.100.201
eth0
Cinder
LVM
eth1
Management
172.16.0.0/16
Internal network(management)
eth1
eth3
eth0
br-int
Network
OVS / Neutron Agents
eth2
br-eth2
10.10.0.201
eth0
Compute
Hypervisor / KVM
br-int
eth1
eth2
br-eth2
10.10.0.202
Tenant network
Tenant Neutron
10.10.0.0/24

OpenStack 控制节点服务 for Nova

1. nova-scheduler：响应运行实例的请求，调度器选择某个服务器
2. nova-api：nova 接口服务
3. nova-conductor: 负责计算服务的数据库交互
4. nova-objectstore: 文件存储服务
5. nova-common: 通用 python 库，支持所有的 OpenStack 环境
6. nova-cert: nova 证书管理服务，用来授权给 nova
7. ntp: 为多节点之间的时间同步，networking time protocol
8. nova-novncproxy：用于控制台代理服务
9. nova-consoleauth：用于控制台的授权验证，授权控制台代理提供的用户令牌。此服务必须运行用于控制台代理工作。

OpenStack 控制节点服务 for Nova

1. vagrant ssh controller

2. $sudo apt-get update
3. $sudo apt-get install nova-api nova-conductor nova-scheduler nova-objectstore nova-cert nova-novncproxy nova-consoleauth

OpenStack 计算节点服务 for Nova

1. nova-compute: 运行虚拟机实例的核心服务
2. nova-api-metadata: nova 元数据前端 api，用来在运行多主机 nova 网络中为计算服务提供源数据下载功能
3. nova-compute-qemu: 在计算服务节点上提供 QEMU 服务。只有在不支持硬件虚拟化的环境下才会使用（在本实践中用到）.
4. nova-novncproxy
5. libvirt-bin： 虚拟机管理服务
6. nova-network： 虚拟机网络管理服务，包括IP地址管理、网络模型管理、DHCP功能
7. ntp：network time protocol

OpenStack 计算节点服务 for Nova

1. vagrant ssh compute
2. $sudo apt-get update
3. $sudo apt-get install nova-compute nova-api-metadata nova-compute-qemu nova-novncproxy libvirt-bin nova-network ntp

NTP 配置

修改 /etc/ntp.conf 文件
server ntp.ubuntu.com
server 127.127.1.0
server 127.127.1.0 stratum 10
$ sudo service ntp restart

OpenStack 支持多种数据库，如内置的SQLite（默认），MySQL和 Postgres数据库。
SQLite 只用于测试环境，在实际生产环境中不使用。

要使用 OpenStack 计算服务，首先需要保证有一个名为 nova 的后台数据库。

# 创建 nova 数据库，添加名为 nova的数据库用户，并赋予 OpenStack 计算服务所需要的权限
$ vagrant ssh controller
$ mysql –u root –popenstack
$ create database nova;
$ grant all privileges on nova.* to ‘nova’@’%’ identified by ‘openstack’;
$ grant all privileges on nova.* to ‘nova’@’localhost’ identified by ‘openstack’;

配置数据库服务器的连接地址。/etc/nova/nova.conf

sql_connection=mysql://nova:openstack@172.16.0.200/nova

1. 配置 /etc/nova/nova.conf

- 需要在控制节点和计算节点上分别配置 /etc/nova/nova.conf 文件
- 配置内容，参考下一页

配置 /etc/nova/nova.conf

需要在控制节点和计算节点上配置 /etc/nova/nova.conf 文件

```
[DEFAULT]
dhcpbridge_flagfile=/etc/nova/nova.conf #指定dhcpbridge 服务的配置文件位置
dhcpbridge=/usr/bin/nova-dhcpbridge #指定dhcpbridge 服务的路径
logdir=/var/log/nova #记录所有服务的日志，需要root权限
state_path=/var/lib/nova  #主机上Nova 用于维护运行着的服务状态的路径
lock_path=/var/lock/nova  #Nova 写锁文件的路径
root_wrap_config=/etc/nova/rootwrap.conf  #指定帮助脚本路径
verbose=True  #设置是否显示更多的信息
```

```
use_syslog = True#把日志发送给syslog日志消息来源
syslog_log_facility = LOG_LOCAL0  #指定使用哪个日志消息来源
#解析文件的位置，包括nova-api 服务的paste.deploy配置
api_paste_config=/etc/nova/api-paste.ini
enabled_apis=ec2,osapi_compute,metadata   #指定默认使用的API

# Libvirt and Virtualization
libvirt_use_virtio_for_bridges=True #设置是否使用桥接驱动 virtio
connection_type=libvirt #指定桥接类型
#设置虚拟化模式。Qemu是软件虚拟化，底层需要运行virtual box. 其他选项有kvm, xen
libvirt_type=qemu
```

```
# Database
sql_connection=mysql://nova:openstack@172.16.0.200/nova #设置数据库连接地址

# Messaging
rabbit_host=172.16.0.200 #通知OpenStack 服务到哪里去查找 rabbitmq 消息队列服务

# EC2 API Flags
ec2_host=172.16.0.200 #设置 nova-api 服务的外部IP地址
ec2_dmz_host=172.16.0.200 #设置 nova-api 服务的内部IP地址
ec2_private_dns_show_ip=True   #设置是否显示IP地址
```

```
# Network settings
network_api_class=nova.network.neutronv2.api.API   #设置网络服务API类
neutron_url=http://192.168.100.200:9696 #设置网络服务地址
neutron_auth_strategy=keystone     #设置验证策略
neutron_admin_tenant_name=service #设置网络服务管理权限租户名
neutron_admin_username=neutron  #设置网络服务管理权限用户名
neutron_admin_password=neutron  #设置网络服务管理权限密码
#设置网络服务验证地址
neutron_admin_auth_url=http://192.168.100.200:5000/v2.0
```

#设置Nova服务安全过滤使用 vif 插件（防火墙）
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver

#设置用来创建以太网设备的驱动
linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver
security_group_api=neutron #设置安全API的分类名
firewall_driver=nova.virt.firewall.NoopFirewallDriver #设置防火墙驱动
#设置SSL验证时使用的证书文件路径
neutron_ca_certificates_file=/etc/ssl/certs/ca.pem
service_neutron_metadata_proxy=true #表示用作验证元数据代理的计算节点
neutron_metadata_proxy_shared_secret=foo #设置元数据代理使用的密文

#Metadata

#设置元数据节点地址

metadata_host = 192.168.100.200

metadata_listen = 192.168.100.200

metadata_listen_port = 8775

Cinder

volume_driver=nova.volume.driver.ISCSIDriver #设置卷驱动器

volume_api_class=nova.volume.cinder.API #设置卷接口类型

iscsi_helper=tgtadm

iscsi_ip_address=172.16.0.200 #设置iscsi IP 地址

```
# Images
#指定用glance 服务来管理镜像
image_service=nova.image.glance.GlanceImageService
glance_api_servers=192.168.100.200:9292 #设置glance服务服务器地址

# Scheduler
scheduler_default_filters=AllHostsFilter #指定调度器可向所有计算主机发送请求

# Auth
auth_strategy=keystone #指定使用keystone 验证所有服务
#设置keystoneEc2 的地址
keystone_ec2_url=https://192.168.100.200:5000/v2.0/ec2tokens
```

```
# NoVNC
novnc_enabled=true #为实例启用vnc 服务
novncproxy_host=192.168.100.200 #设置vnc 代理服务器IP地址
novncproxy_base_url=http://192.168.100.200:6080/vnc_auto.html
novncproxy_port=6080
#指定nova 服务使用XVP VNC 控制台代理端口、IP地址、URL
xvpvncproxy_port=6081
xvpvncproxy_host=192.168.100.200
xvpvncproxy_base_url=http:// 192.168.100.200:6081/console
#指定VNC服务器代理客户端地址、监听端口
vncserver_proxyclient_address=192.168.100.200
vncserver_listen=0.0.0.0
```

#身份认证配置
[keystone_authtoken]
auth_uri = http://192.168.100.200/35357/v2.0/
identity_uri = http://192.168.100.200:5000
admin_tenant_name=service
admin_user=nova
admin_password=nova

需要告知OpenStack 计算服务如何使用身份认证服务来认证用户和服务

需要在【 计算节点 】和【 控制节点 】上进行以下操作

1. 在计算节点上安装 Python-keystone 软件包

   $ sudo apt-get install python-keystone

2. 修改 /etc/nova/nova.conf 配置文件
3. 修改 /etc/nova/api-pase.ini 配置文件
4. 确保 OpenStack 身份认证服务运行后，重启 OpenStack 计算服务

1. 修改 /etc/nova/nova.conf 配置文件

```
[DEFAULT]
 api_paste_config=/etc/nova/api-paste.ini
 auth_strategy=keystone
 keystone_ec2_url=https://192.168.100.200:5000/v2.0/ec2tokens

[keystone_authtoken]
 admin_tenant_name=service
 admin_user=nova
 admin_password=nova
 identity_uri=https://192.168.100.200:35357/v2.0
 insecure=true
```

2. 修改 /etc/nova/api-paste.ini 配置文件

```
[filter:keystonecontext]
paste.filter_factory = nova.api.auth:NovaKeystoneContext.factory

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
```

- 控制节点上的相关服务

1. nova-api
2. nova-objectstore
3. nova-scheduler
4. nova-conductor
5. nova-cert
6. nova-novncproxy
7. nova-consoleauth

- 计算节点上的相关服务

1. nova-compute
2. nova-network
3. nova-api-metadata
4. nova-novncproxy
5. libvirt-bin

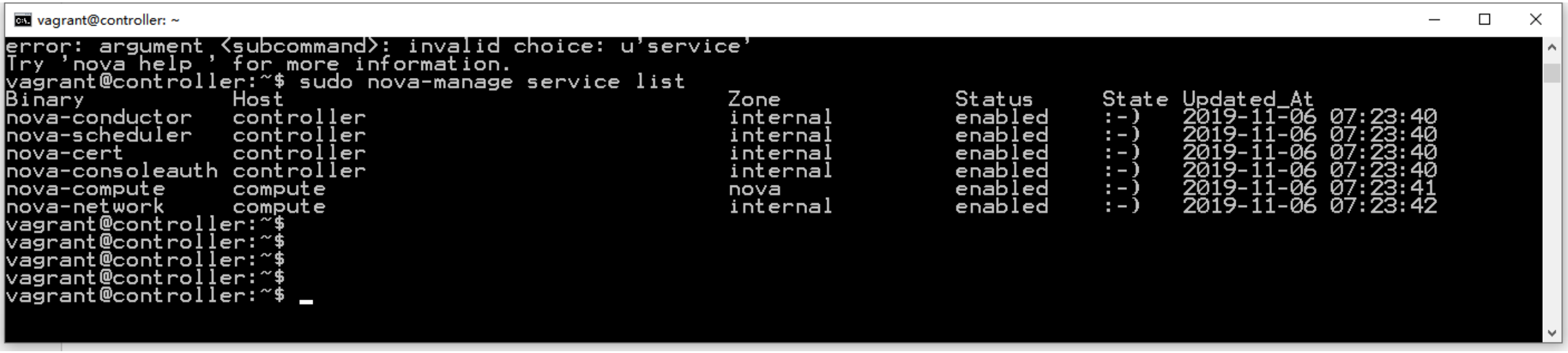
先安装 nova 客户端工具

```
$ sudo apt-get install update
$ sudo apt-get install python-novaclient
```

在控制节点上执行以下命令列出各种 nova服务和对应的状态

```
$ sudo nova-manage service list
```

:-) 表示正常
XXX 表示异常



检查镜像服务是否启动，查看进行信息及确认使用端口 9292

```
$ ps –ef | grep glance
$ netstat –ant | grep 9292.*LISTEN
```

```
vagrant@controller:~$
vagrant@controller:~$
vagrant@controller:~$ ps -ef | grep glance
glance    1137     1  0 06:42 ?        00:00:00 /usr/bin/python /usr/bin/glance-registry
glance    1144     1  0 06:42 ?        00:00:01 /usr/bin/python /usr/bin/glance-api
glance    1451  1137  0 06:42 ?        00:00:00 /usr/bin/python /usr/bin/glance-registry
glance    1737  1144  0 06:42 ?        00:00:00 /usr/bin/python /usr/bin/glance-api
vagrant   4631  3267  0 08:05 pts/0    00:00:00 grep --color=auto glance
vagrant@controller:~$ netstat -ant | grep 9292.*LISTEN
tcp        0      0 0.0.0.0:9292            0.0.0.0:*               LISTEN
vagrant@controller:~$
vagrant@controller:~$
vagrant@controller:~$
vagrant@controller:~$
vagrant@controller:~$
```

检查 rabbitmq 服务是否正常

$ sudo rabbitmqctl status

vagrant@controller: ~

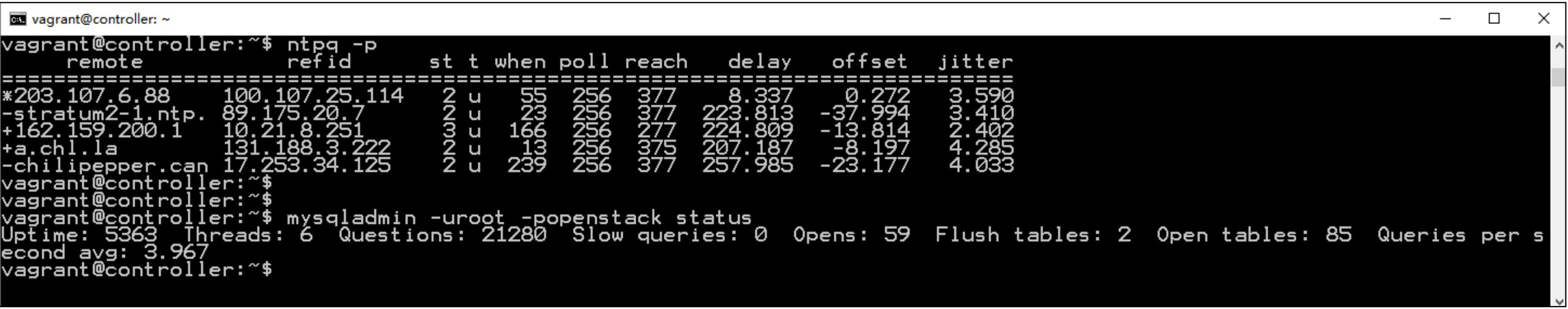
```
vagrant@controller:~$
vagrant@controller:~$ sudo rabbitmqctl status
Status of node rabbit@controller ...
[{pid,2233},
 {running_applications,[{rabbit,"RabbitMQ","3.2.4"},
                        {os_mon,"CPO  CXC 138 46","2.2.14"},
                        {mnesia,"MNESIA  CXC 138 12","4.11"},
                        {xmerl,"XML parser","1.3.5"},
                        {sasl,"SASL  CXC 138 11","2.3.4"},
                        {stdlib,"ERTS  CXC 138 10","1.19.4"},
                        {kernel,"ERTS  CXC 138 10","2.16.4"}]},
 {os,{unix,linux}},
 {erlang_version,"Erlang R16B03 (erts-5.10.4) [source] [64-bit] [smp:2:2] [async-threads:30] [kernel-poll:true]\n"},
 {memory,[{total,63444512},
          {connection_procs,953704},
          {queue_procs,436368},
          {plugins,0},
          {other_proc,13612968},
          {mnesia,114256},
          {mgmt_db,0},
          {msg_index,46936},
          {other_ets,793704},
          {binary,26288952},
          {code,16526072},
          {atom,594537},
          {other_system,4077015}]},
 {vm_memory_high_watermark,0.4},
 {vm_memory_limit,839575142},
 {disk_free_limit,50000000},
 {disk_free,37593616384},
 {file_descriptors,[{total_limit,924},
                    {total_used,19},
                    {sockets_limit,829},
                    {sockets_used,17}]},
 {processes,[{limit,1048576},{used,291}]},
 {run_queue,0},
```

检查 NTP 服务，返回与NTP服务器通信的相关信息

$ ntpq –p

检查数据库服务，返回数据库的统计数据

$ mysqladmin –uroot –popenstack status



THANKS