OpenStack

云计算基础

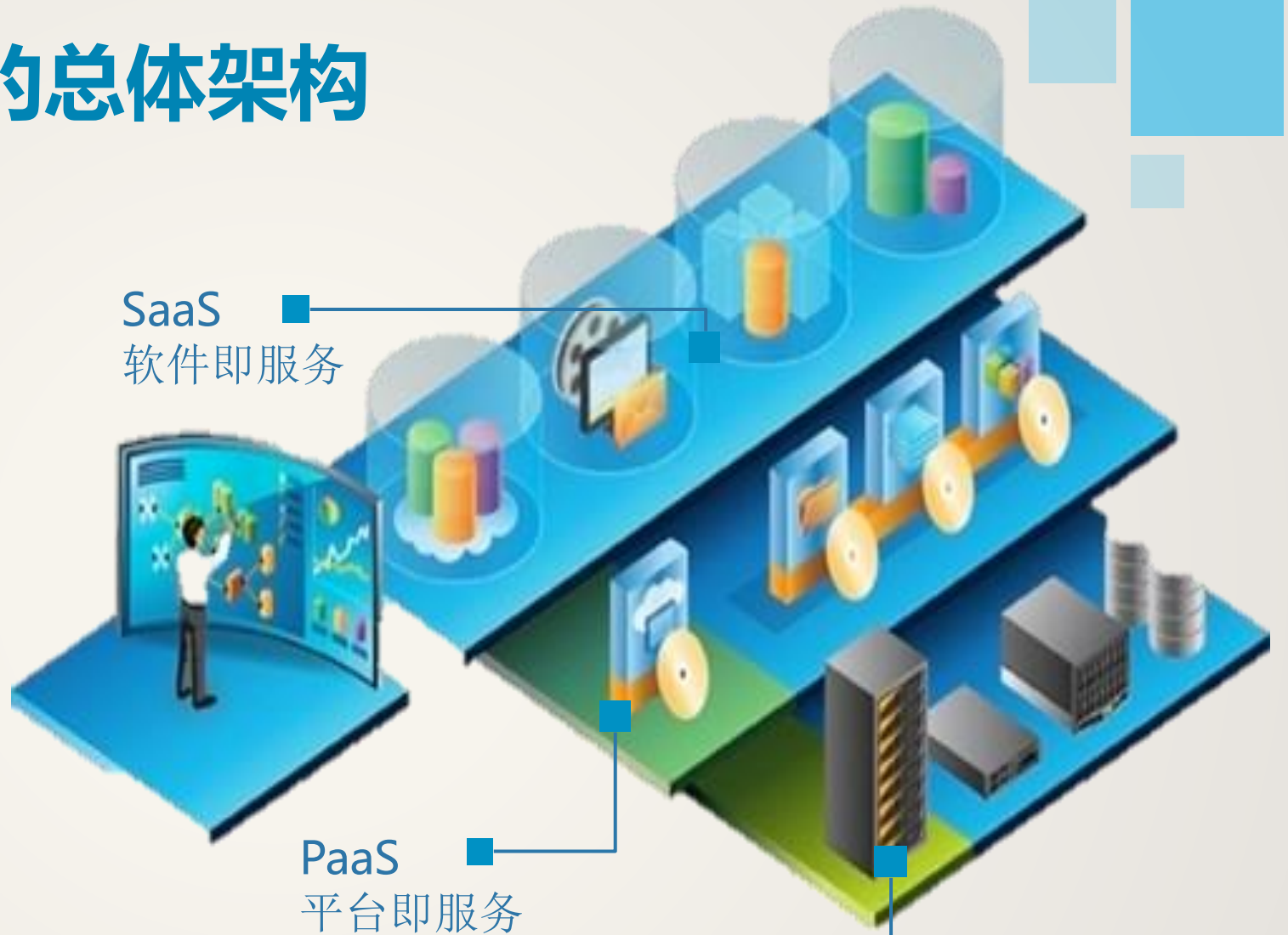
云计算的总体架构

IaaS 基础实施即服务

Linux、BSD、Windows...

OpenStack

基础实施



SaaS
软件即服务

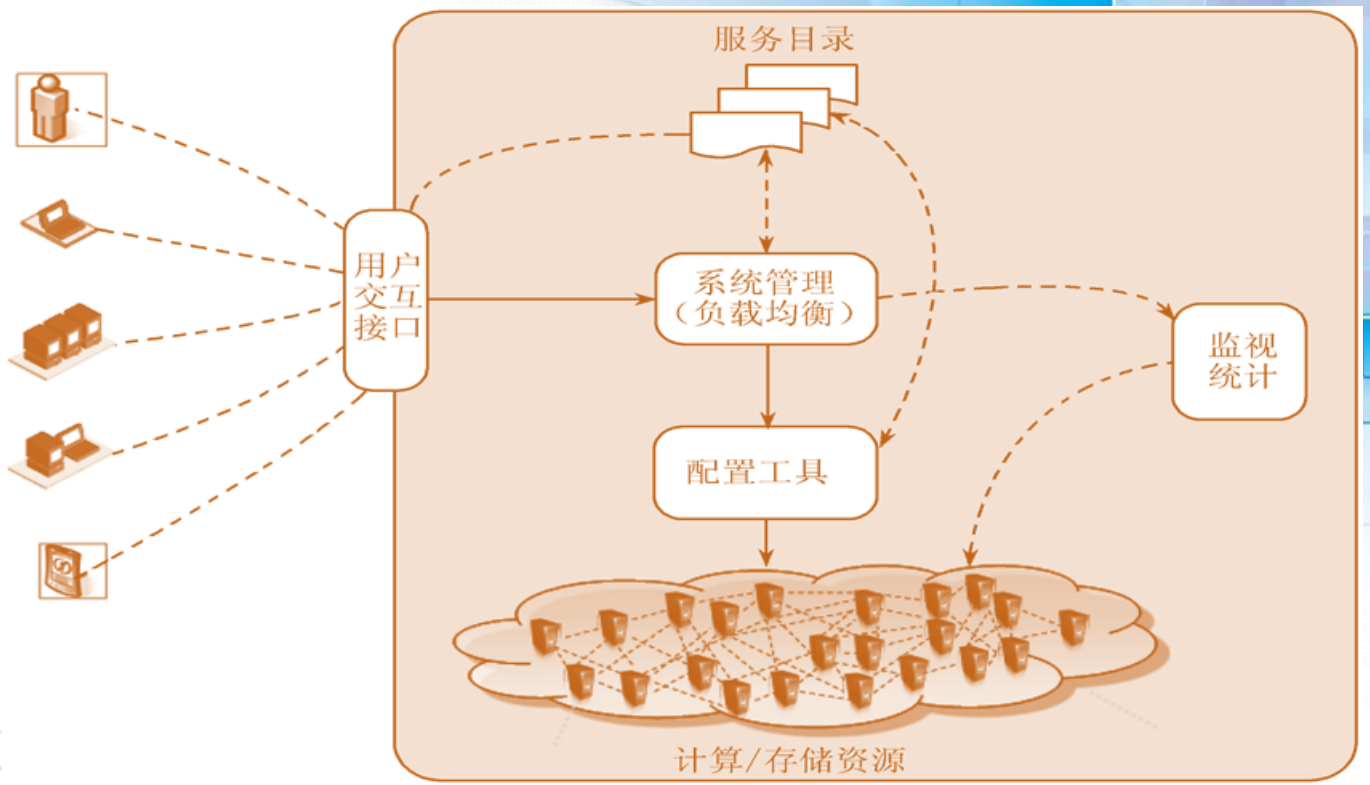
PaaS
平台即服务

OpenStack

IaaS
基础设施即服务

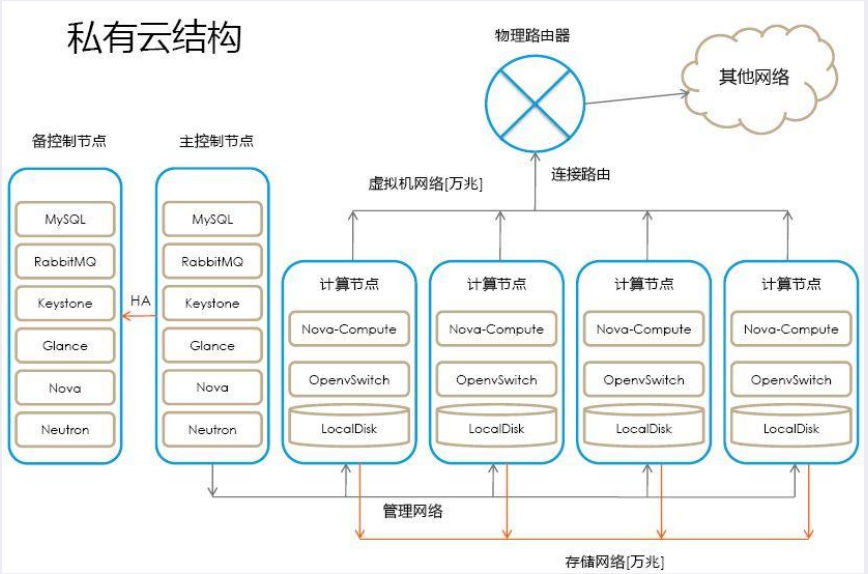
云计算定义

NIST（美国国家标准与技术研究院）：云计算是一个模型，用来实现对已经配置好的计算资源(计算/存储/网络 /应用程序/各种服务)的高效/方便/按需的访问，这些计算资源可以快速的获取和释放，同时管理成本极低，而且与提供商的沟通成本基本为零。



案例

小米的 OpenStack 私有云



CentOS+OpenStack的系统，RDO方式安装，有四个机房，2000+VM，4500+物理机内核；机器的配置主要为：50T内存、1200T虚拟磁盘、480T块存储、120T对象存储

- 可用度达到99.99%：运行16个月，2次故障;
- 目前使用率：平均40%（物理机利用率），1虚拟12;
- 覆盖度：小米所有产品线;
- 业务类型：开发，测试，线上（线下70%）。

云计算平台的系统架构

任务实现

节点部署服务示意图

- 控制节点 组件

  MySQL

  RabbitMQ，Memcache

  Keystone

  Glance；Nova；Neutron

- 实例节点 组件

  Nova；Neutron

- 实例节点 组件

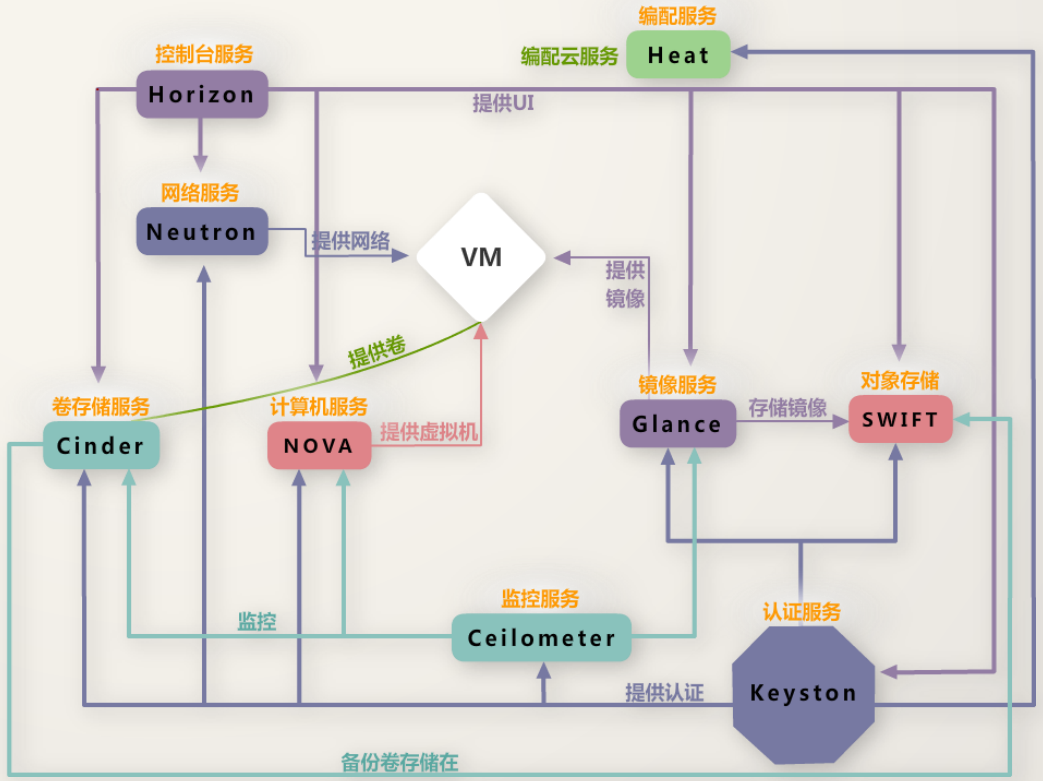
  Swift；

Keystone管理认证用户

1

KeyStone是OpenStack基础支持服务，keystone有如下作用:

- 管理用户和权限
- 认证（Authentication）和鉴权（Authorization）
- 维护OpenStack Services的Endpoint



1 keystone管理认证用户

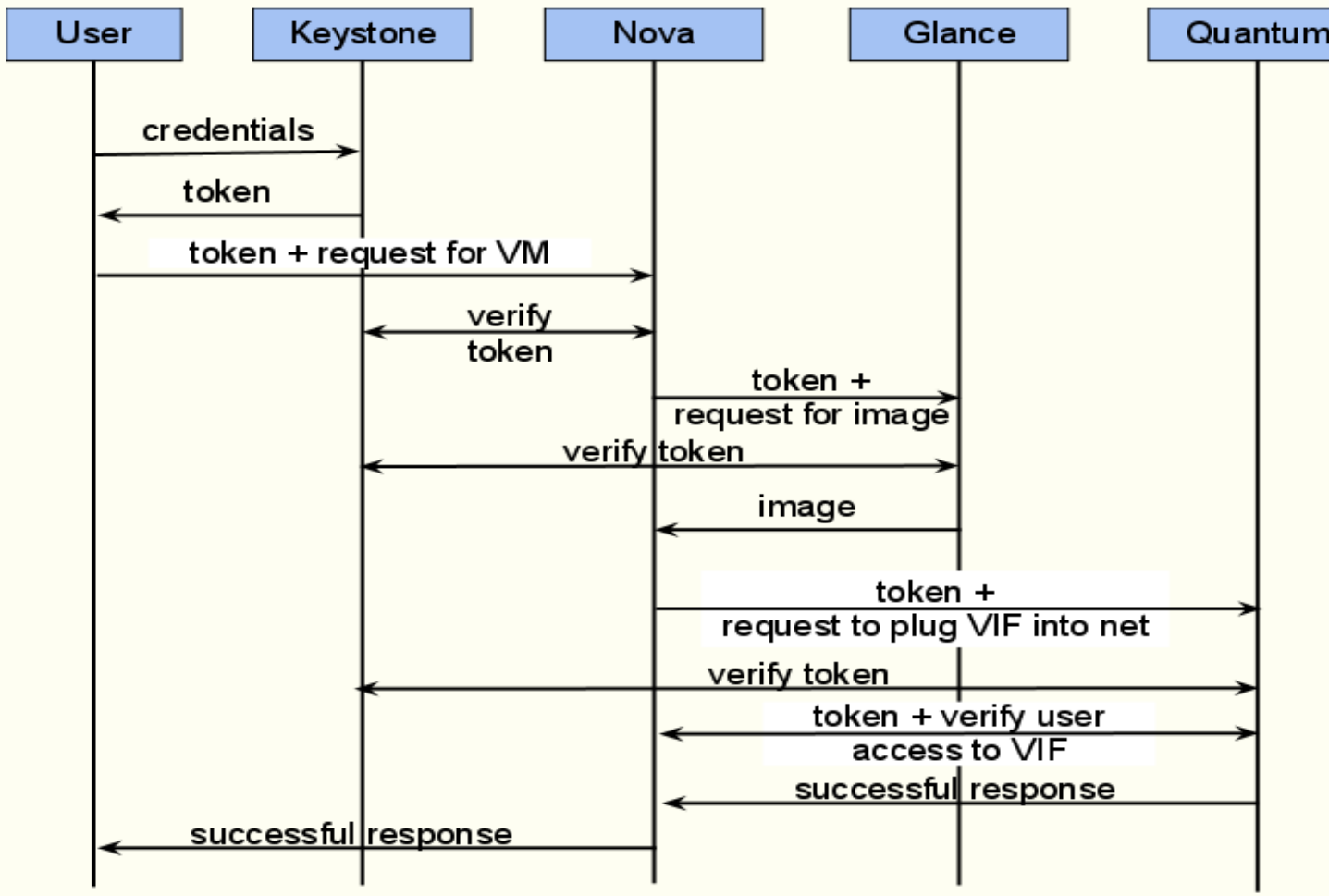
认证服务流程

1. User提供用户名密码请求登录；Keystone验证成功返回token给User；

2. User使用token请求Nova提供服务，Nova会去Keystone查询User Token的有效性；

3. Nova确认token有效后，
   3.1 Nova使用该token请求Glance服务；同前，Glance会向Keystone验证令牌有效性；确认有效后返回image给Nova；
   3.2 Nova使用该token请求Neutron服务；Nuetron向Keystone验证token有效性，确认后提供网络服务；

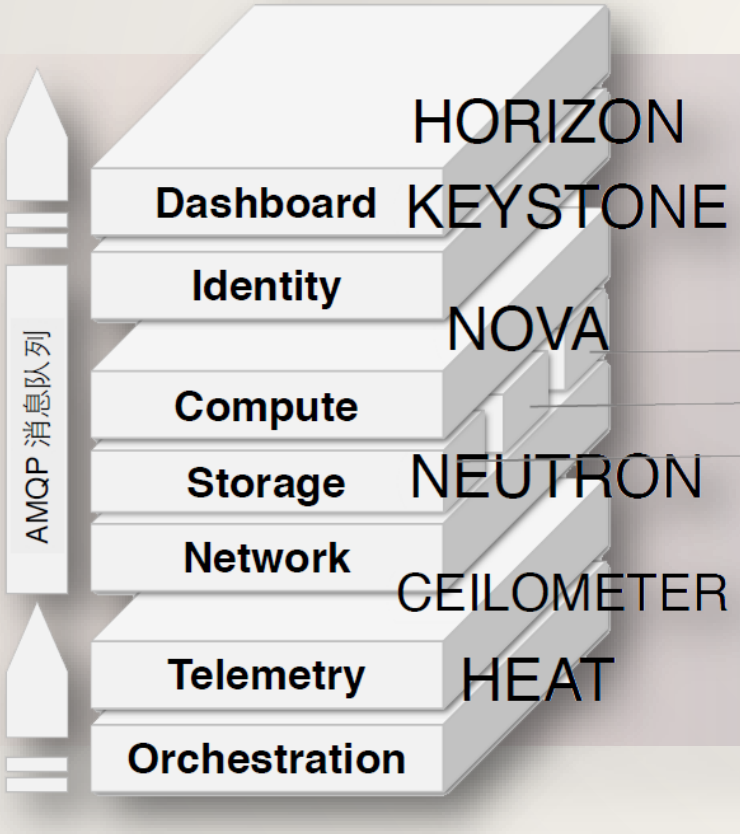
4. Nova反馈操作结果至用户。



服务申请认证机制流程

1 镜像服务Glance

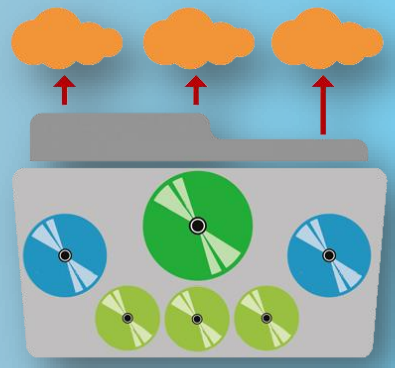
云计算环境下的解决方案



云环境下的解决方案：

GLANCE 镜像

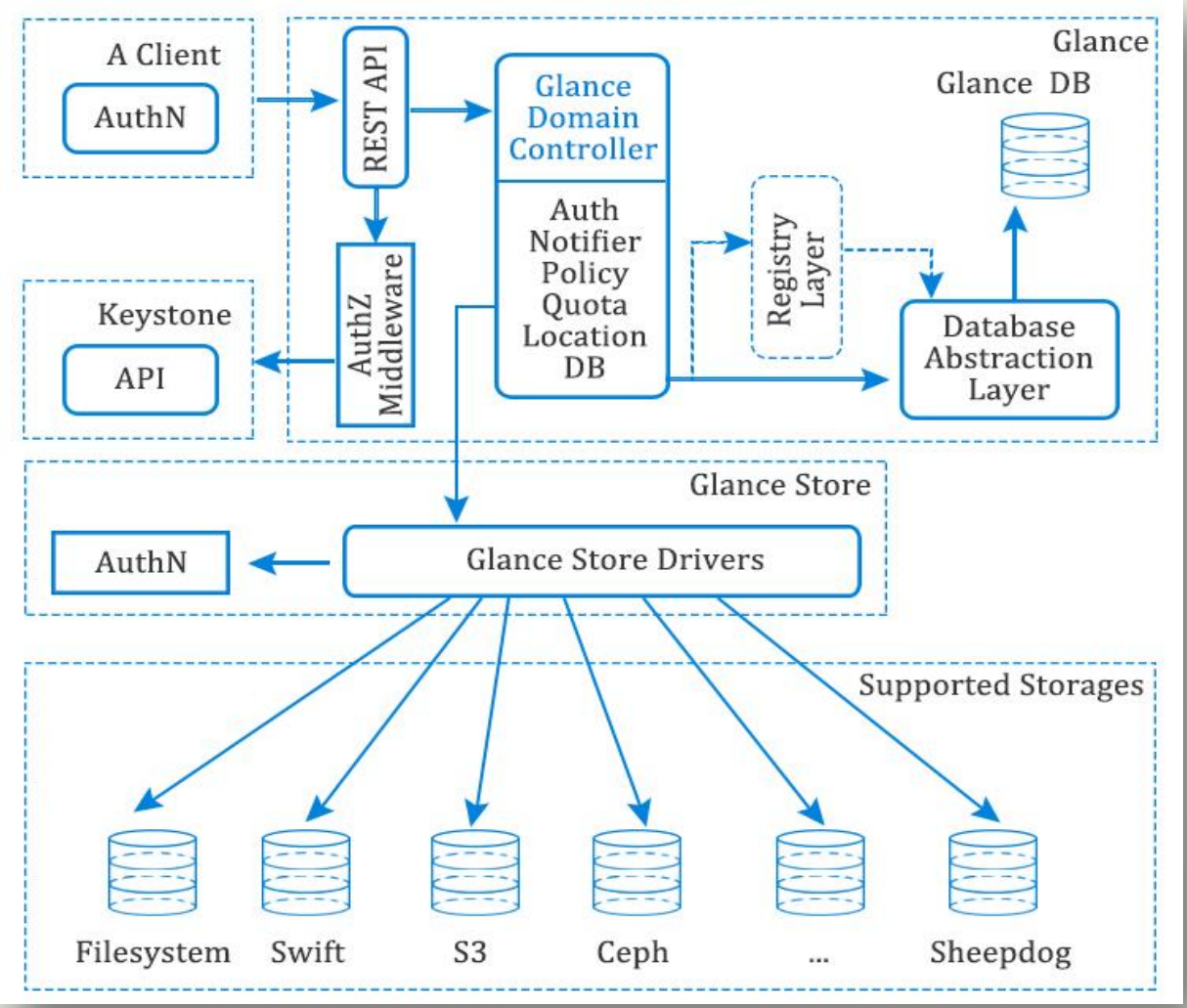
1. Glance用于实现发现、注册、获取虚拟机镜像和元数据；



2. Glance镜像数据支持存储多种的存储系统，可以是简单文件系统、对象存储系统等，确保镜像文件安全可靠。

1 学习镜像服务

Glance服务



Glance的组成：

1.Glance API 是后台进程，提供REST API服务（查询Image、获取Image、存储Image）；

2. Glance Registry是系统后台服务进程，负责Image的元数据（Image的大小、类型等）；

systemctl restart openstack-glance-api.service openstack-glance-registry.service

3.Database Image的元数据保存在database中；

4. Glance Store 将Image存放在后端，后端可用的存储方式可以是：Swift、Amazon S3、Cinde、本地文件系统等等方式；

Glance 架构图

1

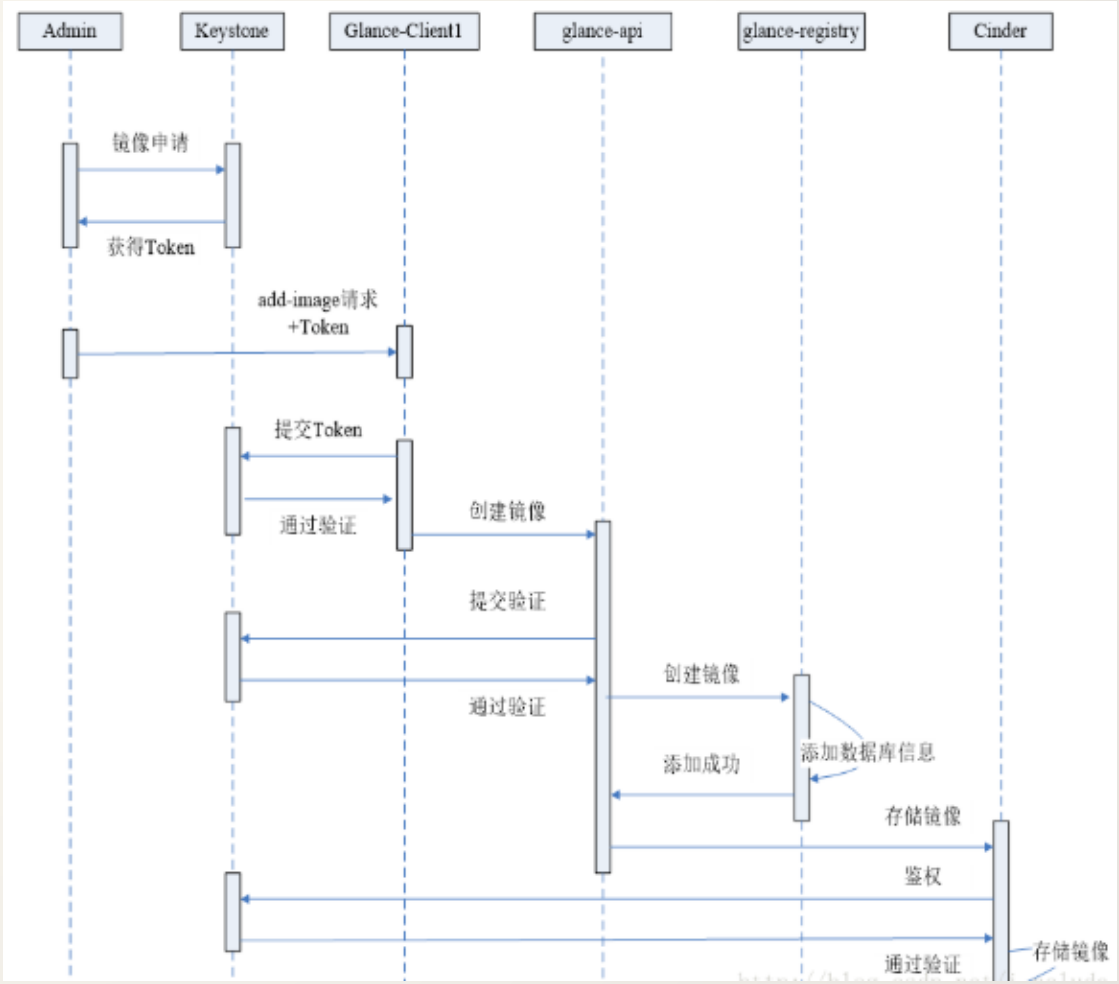
Glance认证管理流程

认证服务流程

1. Admin提供用户名密码请求登录；Keystone验证成功返回token给Admin；

2. Admin使用token请求Glance提供服务，Glance会去Keystone查询Token的有效性；

3. Glance确认token有效后，
   3.1 Glance使用该token请求Glance-api服务创建镜像；同前，Glance会向Keystone验证令牌有效性；确认有效后请求glance-registry服务；
   3.2 Glance使用该token请求Cinder服务；Cinder向Keystone鉴权，确认后存储该镜像；



服务申请认证机制流程

1 组件配置通用模式

安装XXX服务需要以下步骤：

1. 创建XXX数据库；
2. 在keystone 上面注册XXX；
3. 创建XXX服务的API 端点；
4. 安装XXX相关软件;
5. 配置XXX.conf；
6. 启动XXX服务。

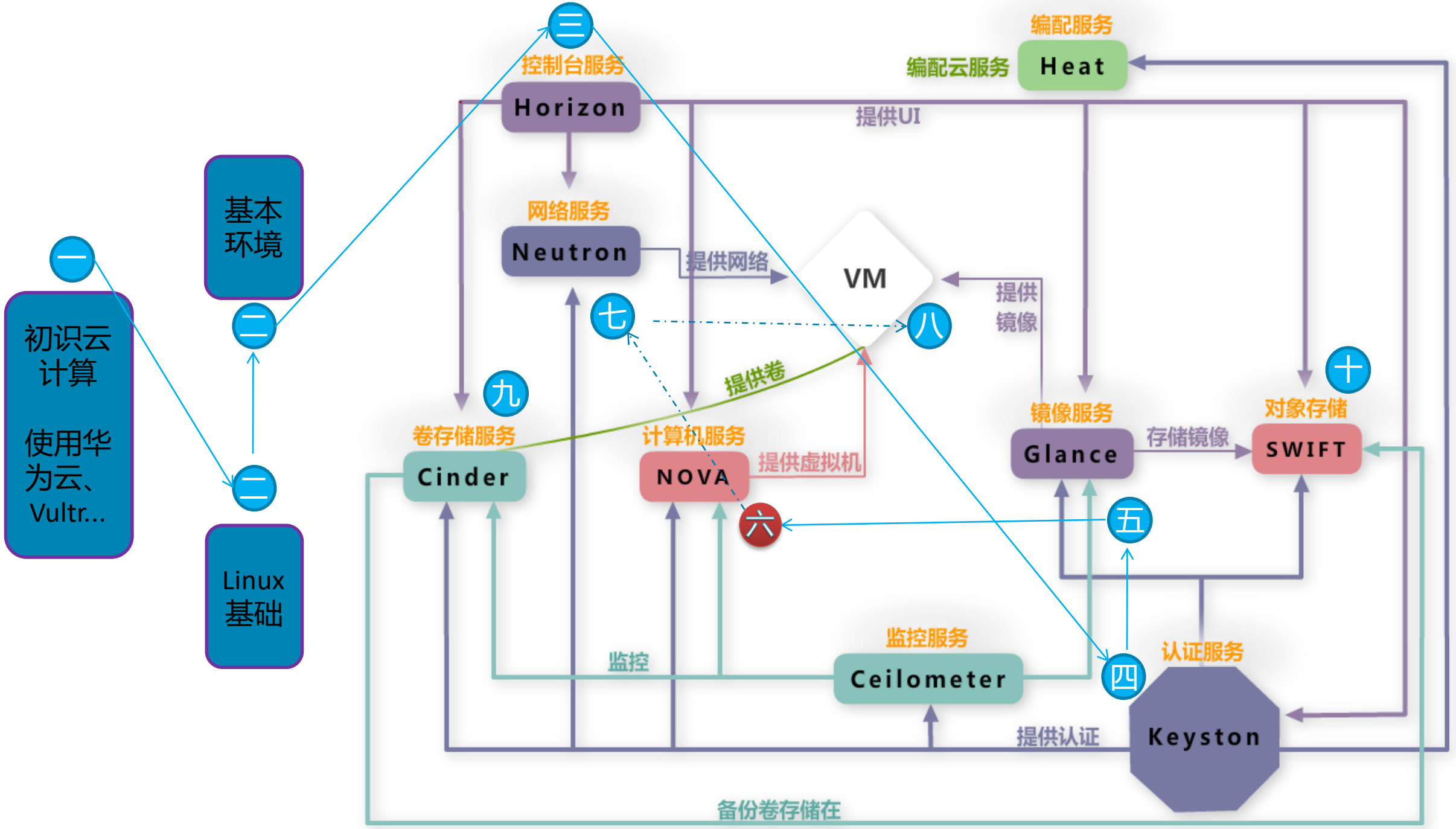
XXX服务主配置文件存放在/etc/XXX，名为/XXX-*.conf，在配置文件中需要配置相应参数。

项目6

第七章 计算服务Nova

学习目标

- 了解虚拟化技术
- 了解计算服务(Nova)的相关概念
- 理解计算服务(Nova)的工作原理


一
初识云计算
使用华为云、Vultr...
二
基本环境
二
Linux基础
三
控制台服务
Horizon
编配服务
编配云服务
Heat
提供UI
网络服务
Neutron
提供网络
VM
七
八
提供镜像
九
提供卷
卷存储服务
Cinder
计算机服务
NOVA
提供虚拟机
六
镜像服务
Glance
存储镜像
对象存储
SWIFT
十
五
监控服务
监控
Ceilometer
四
认证服务
Keyston
提供认证
备份卷存储在

1 OpenStack项目回顾

What is OpenStack?

Rackspace（swift）和NASA（nova）共同发起的开源项目，是一系列软件开源项目的组合。

即 基础设施资源管理平台（IaaS, 云操作系统，可以管理硬件池，根据需求提供资源）。

A版本： Swift + Nova

SWIFT	Object Storage	对象存储
NOVA	Compute	计算*

1 OpenStack项目

OpenStack的技术性能

OpenStack是开源云操作系统，Python语言编写，
有命令行CLI，REST Api， Web UI管理界面

1. A Swift + Nova

2. B + Glance

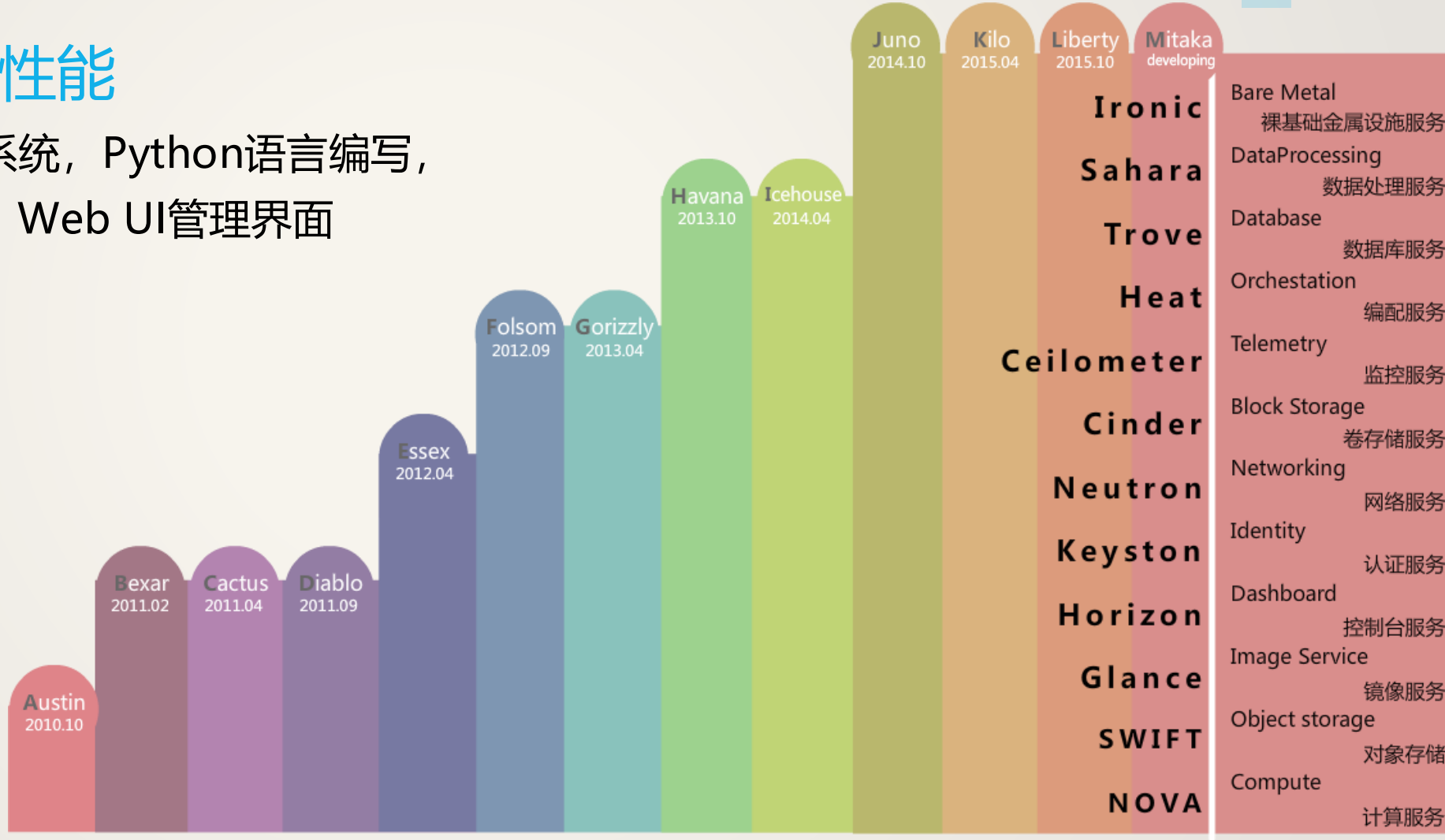
3. C 增加虚拟化技术支持

……

2018 发布Rocky，

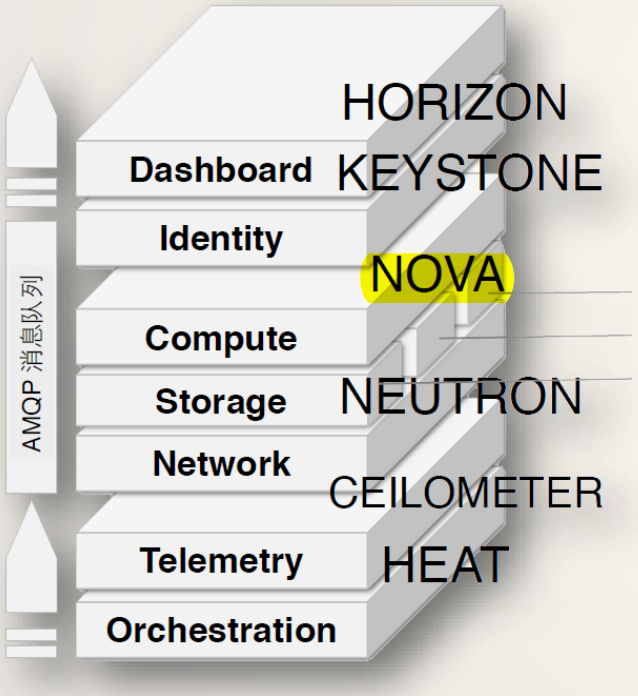
含 裸金属、容器管理、

高性能计算等等



学习计算服务

概述



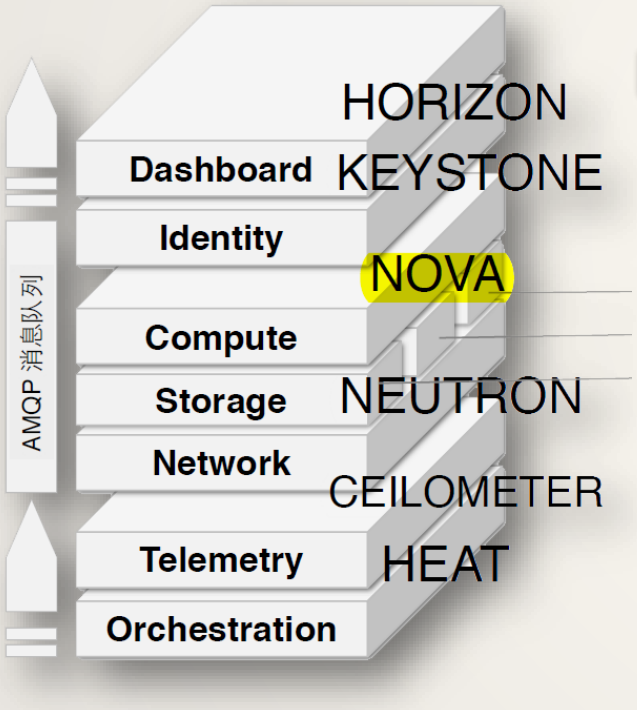
NOVA是OpenStack最核心的服务。负责管理云环境下的计算资源。

其功能包括

- 运行虚拟机实例
- 管理虚拟机生命周期

学习计算服务

概述



- Nova 不是虚拟化软件，仅是一个框架。
  它与运行在主机操作系统上的**虚拟化软件**交互的驱动程序，并基于Web的程序应用接口(API)来提供功能的使用。
- OpenStack最开始的开源项目就是Nova，它提供的软件可以控制基础设施即服务(IaaS)云计算平台，和Amazon EC2云服务器有一定程度相似。

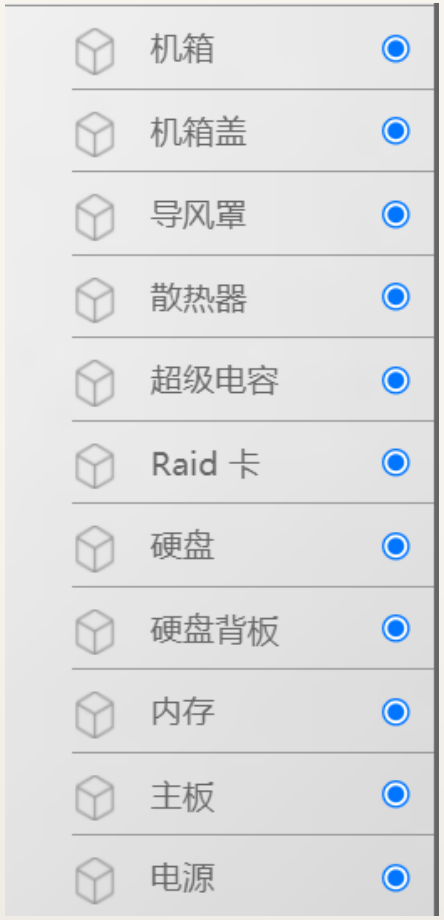
云计算==虚拟化？

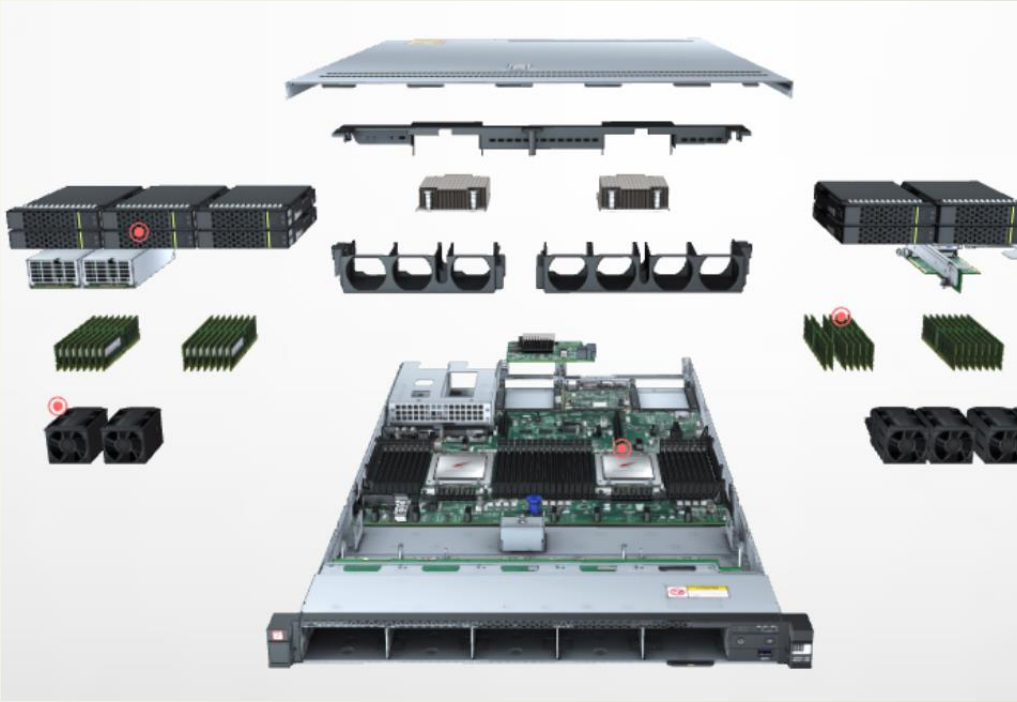
虚拟化操作

物理服务器

https://support-it.huawei.com/server-3D/index_zh.html

物理服务器





虚拟化操作

虚拟化

虚拟化是云计算的基础

虚拟化技术：一台物理服务器上跑多台虚拟机，虚拟机运行独立操作系统，在操作系统上运行多个服务。

1 to N

N to 1 ?

虚拟化操作

虚拟化优势

虚拟化是云计算的基础

虚拟化技术：一台物理服务器上跑单个操作系统，运行多个服务。

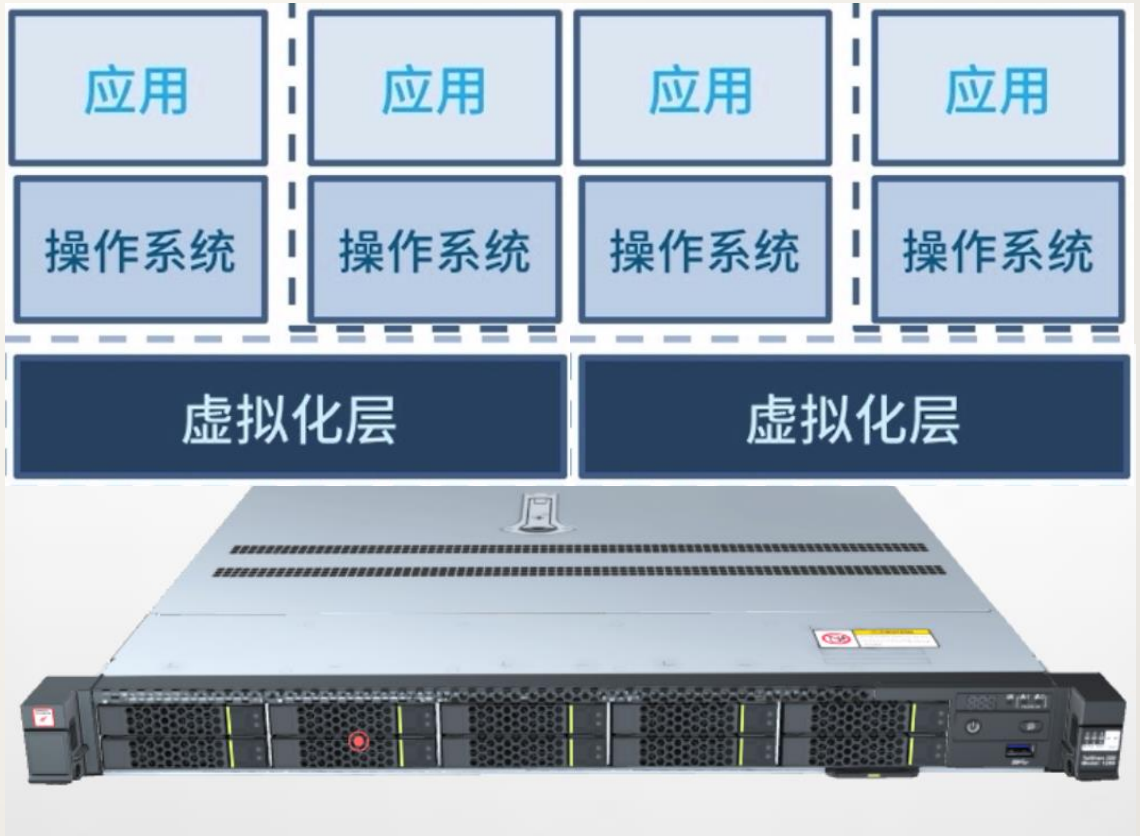
资源利用率？

虚拟化操作

虚拟化优势

虚拟化是云计算的基础

虚拟化技术：一台物理服务器上跑多台虚拟机，虚拟机共享物理机的CPU、内存等资源；逻辑上虚拟机相互隔离。资源利用率 ？



虚拟化操作

虚拟化优势

虚拟化是云计算的基础

虚拟化技术：一台物理服务器上跑多台虚拟机；

虚拟机共享物理机的CPU、内存等资源；

逻辑上虚拟机相互隔离。

虚拟化操作

虚拟化类型

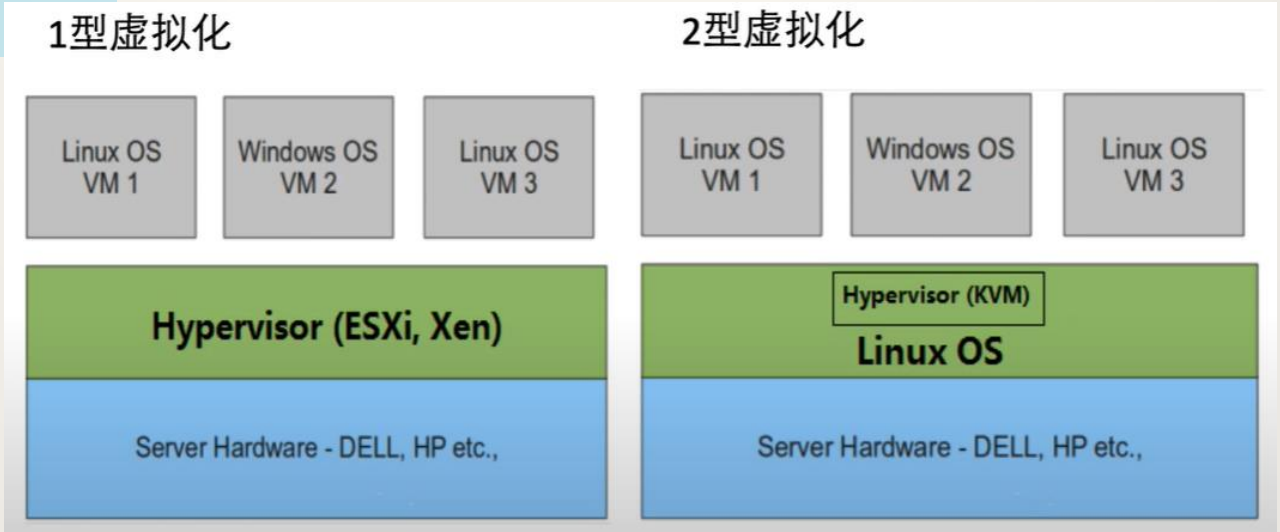
虚拟化是云计算的基础

物理机称为宿主机（Host）； 虚拟机称为客户机（Guest）

1

实现虚拟化的程序称为 Hypervisor，虚拟化分类：

1型虚拟化； 2型虚拟化



1

虚拟化操作

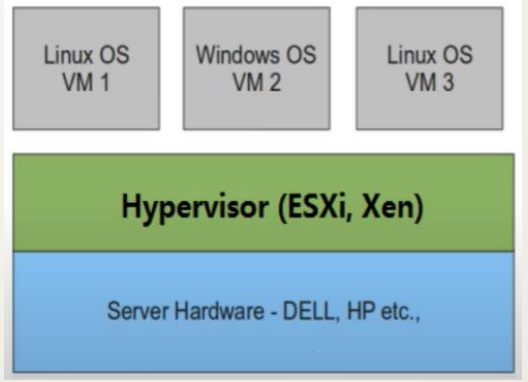
虚拟化架构

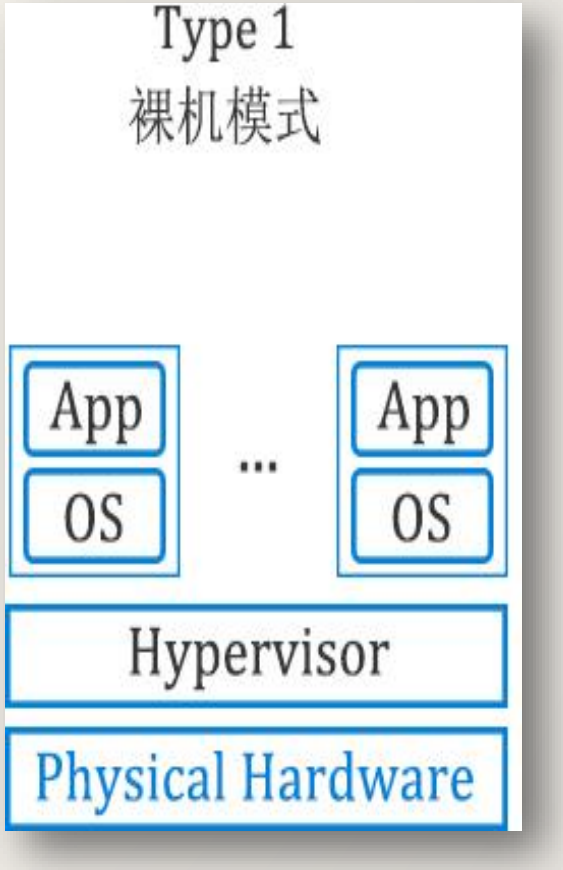
1 型虚拟化

Hypervisor安装在物理机上，虚拟机直接运行在Hypervisor上；

由虚拟化软件提供全仿真的硬件环境，该类型叫作“裸金属”；

Hypervisor一般是定制化的Linux。





1 虚拟化操作

虚拟化架构

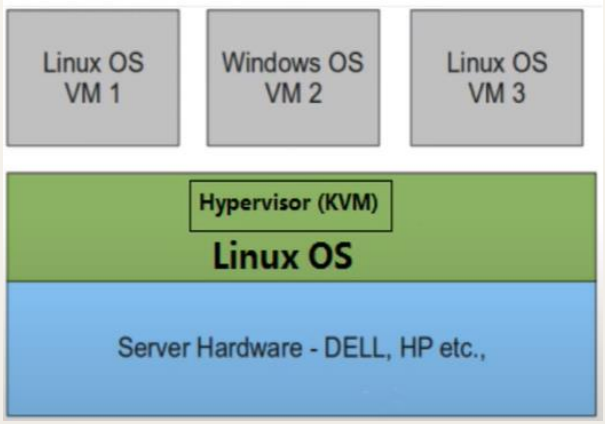
2 型虚拟化

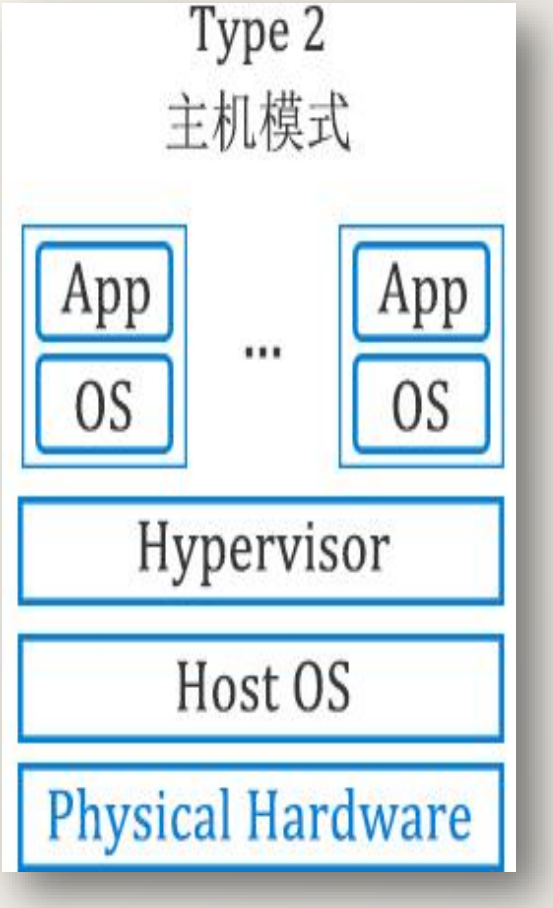
物理机常规安装操作系统：Redhat、Windows等；

Hypervisor作为操作系统上的应用程序；

虚拟机运行在Hypervisor上。

例：KVM、VirtualBox等





1 虚拟化操作

虚拟化架构介绍

性能：1型虚拟机对硬件进行优化，性能高于2型虚拟机

灵活性：2型虚拟机基于普通炒作系统，更灵活；

2型虚拟机支持嵌套（KVM运行KVM....）

1

虚拟化操作

虚拟化原理

常见虚拟化软件

KVM

XEN

VMWARE

VirtualBox

……

虚拟机管理工具参数众多，很难使用；

没有统一的编程接口来管理它们；

没有统一的方式来方便地定义虚拟机。

1 虚拟化操作

虚拟化原理

- Libvirt

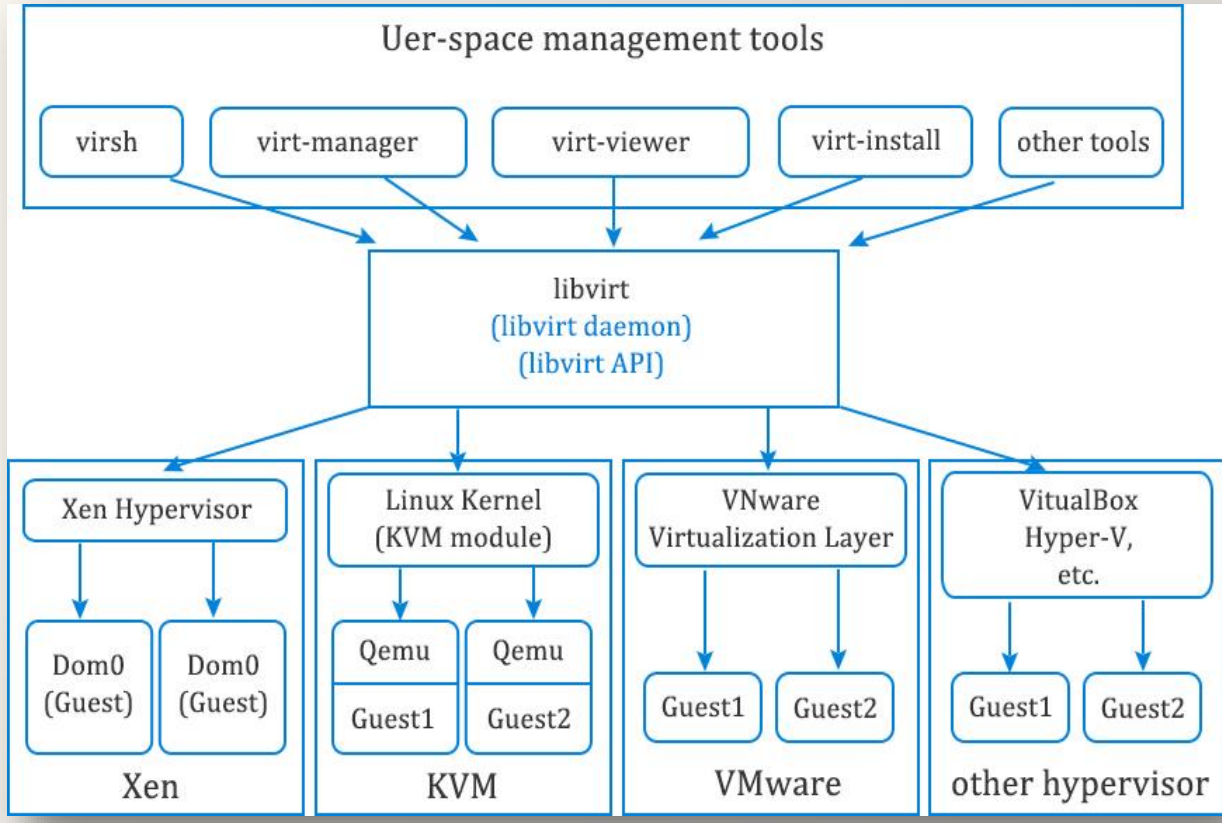
Libvirt是KVM、Xen等的管理工具，其作用是提供单一的接口管理多种不同的虚拟化和 Hypervisor；

组成：libvirtd、API lib、virsh CLI

libvirtd：接受处理API请求，服务进程；

API lib：开发则工具，用于开发virt-manager等管理工具；

virsh CLI：命令行工具。



1

虚拟化操作

虚拟化原理

Libvirt的主要功能：

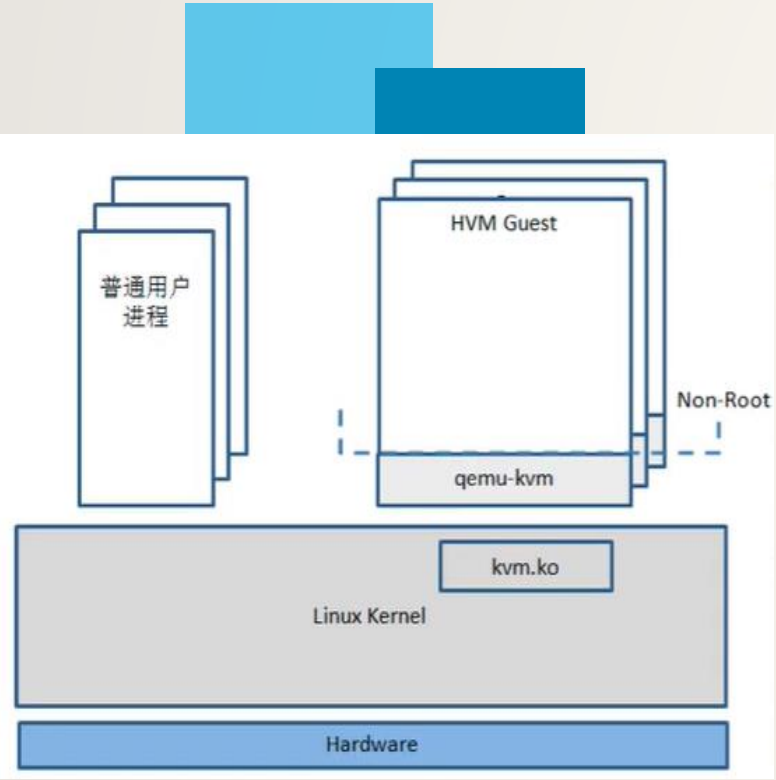
1） 虚拟机管理

- 虚拟机生命周期管理，比如：启动、停止、暂停、保存、恢复和迁移。

- 支持多种设备操作，包括：磁盘、网卡、内存和CPU 添加、删除。

Nova 调用 libvirt 管理 QEMU/KVM 等虚机

虚拟化原理



以KVM为例

1. KVM虚拟化需要CPU硬件支持（BIOS打开虚拟化支持）；

2. 每个虚拟机其实是Linux操作系统的一个进程 ， 同其他系统进程一样被Linux调度；

```
[root@controller ~]# ps -ef|grep qe
root      4177     1  0 Mar10 ?        00:00:00 /usr/bin/qemu-ga --method=virtio-serial
-read,guest-file-write,guest-file-seek,guest-file-flush,guest-exec,guest-exec-status -F/
qemu     22345     1  2 Mar24 ?        07:55:29 /usr/libexec/qemu-kvm -name guest=instar
in-96-instance-00000095/master-key.aes -machine pc-i440fx-rhel7.6.0,accel=tcg,usb=off,du
-bdcd-d3afc07df4e7 -smbios type=1,manufacturer=RDO,product=OpenStack Compute,version=18.
Virtual Machine -no-user-config -nodefaults -chardev socket,id=charmonitor,fd=80,server
ice piix3-usb-uhci,id=usb,bus=pci.0,addr=0x1.0x2 -drive file=/dev/sdb,format=raw,if=none
-blk-pci,scsi=off,bus=pci.0,addr=0x4,drive=drive-virtio-disk0,id=virtio-disk0,bootindex=
net0,mac=fa:16:3e:8c:56:96,bus=pci.0,addr=0x3 -add-fd set=2,fd=85 -chardev pty,id=charse
ablet,id=input0,bus=usb.0,port=1 -vnc 0.0.0.0:1 -k en-us -device cirrus-vga,id=video0,bu
evateprivileges=deny,spawn=deny,resourcecontrol=deny -msg timestamp=on
root     25498 25459  0 16:34 pts/0    00:00:00 grep --color=auto qe
```

虚拟化原理

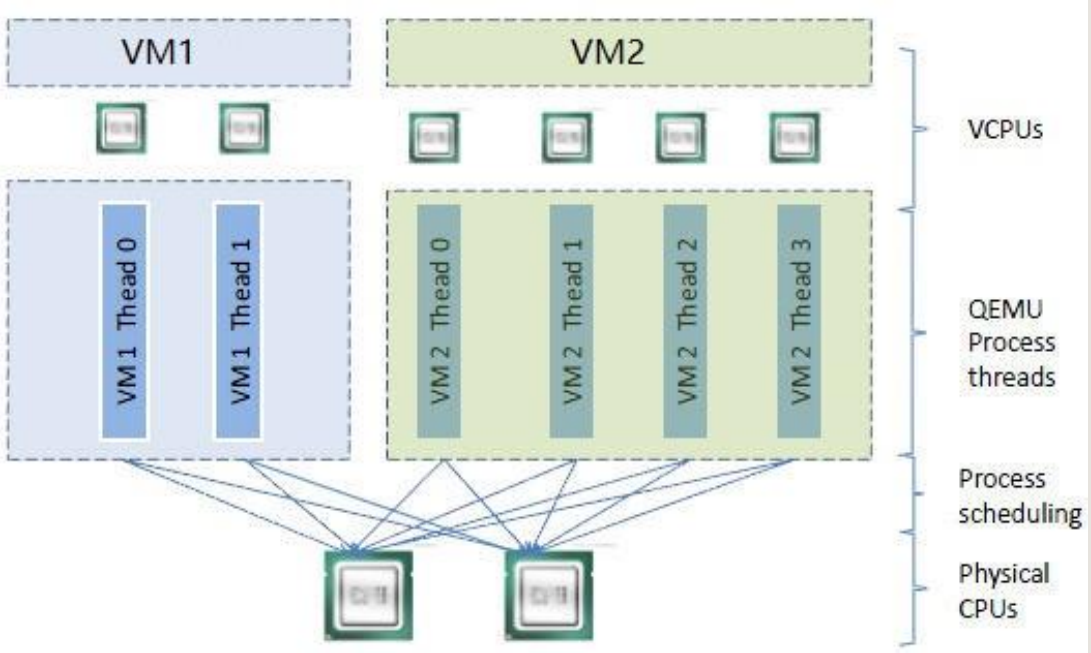
1

3. 虚拟机中每个虚拟CPU对应qemu的一个线程；

虚拟机可模拟的CPU数可大于物理机拥有的CPU数，在虚拟化技术中叫做 超配

生产环境中是否允许超配需要根据虚拟机的整体负载来定

例：VM1有2个vCPU; VM2有4个vCPU, 则VM1、VM2分别对应2个和4个线程。



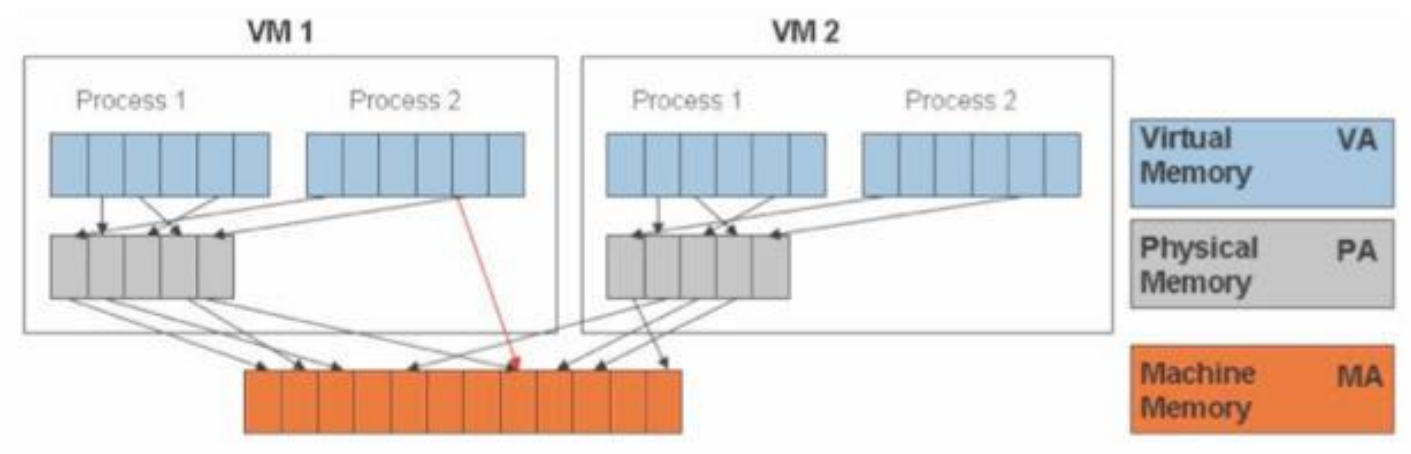
内存虚拟化

KVM通过内存虚拟化共享物理内存，动态分配给虚拟机；

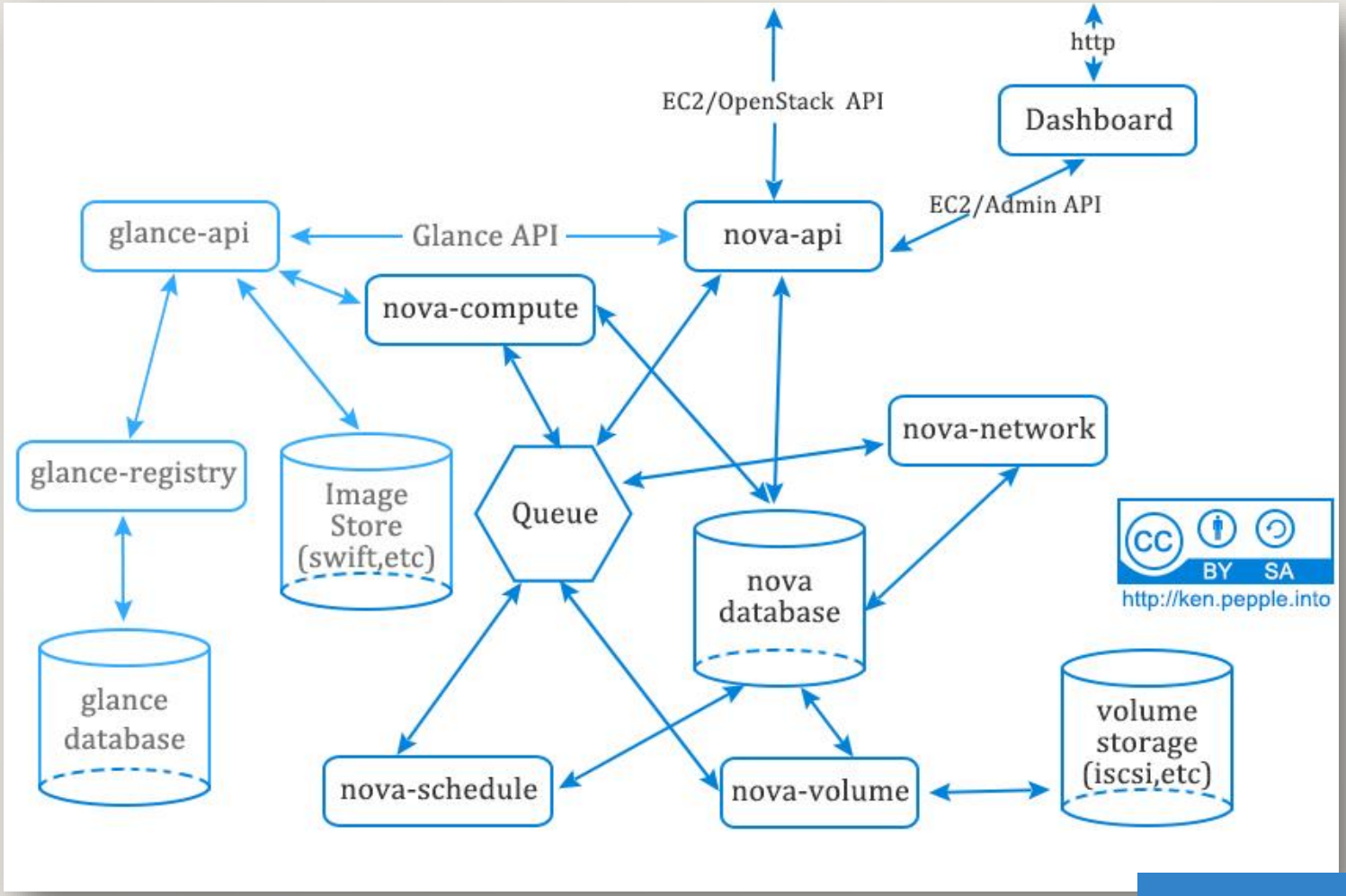
地址转换顺序：VA（虚拟机内存）--> PA(物理内存)-->MA(机器内存)

内存超配：虚拟机内存之和 > 宿主机物理内存

内存是否允许超配需要根据实际负载而定



2 学习计算服务
架构介绍



Nova架构

Nova含服务（7.1）：

1. nova-API
2. nova-compute
3. nova-scheduler
4. nova-conductor

...

Nova 框架图

2 计算服务

Nova服务

Nova服务：

1. nova-API

nova-api是nova组件的接口服务进程；nova-api向外暴露REST API接口（Endpoints）以提供服务；

nova-api兼容Amazon EC2 API， 基于nova-api开发的工具可以管理OpenStack和Amazon EC2.

2. nova-compute

nova-compute在计算节点上管理虚拟机实例：创建、删除等的管理

3. nova-scheduler

创建Instance时，用户需要CPU、内存等资源，nova-scheduler自动实现对CPU等资源的调度；

Nova 框架图

2 计算服务

Nova服务

4. nova-conductor

nova-conductor实现数据库访问，为nova-compute提供instances信息。

基于安全和可扩展性的考虑，nova-compute不直接访问数据库，而是分离出nova-conductor来访问数据库。

5. nova-consoleauth

对访问虚拟机控台的请求提供Token认证

6. Database

使用MySQL，位于控制节点，存放Nova运行时信息

Nova 框架图

2 Nova 创建实例流程

1. 向 nova-api 发送请求

客户（可以是 OpenStack 最终用户，也可以是其他程序）向 API（nova-api）发送请求：“帮我启动这个 Instance”

2. nova-api 发送消息

nova-api 向 Messaging（RabbitMQ）发送了一条消息：“让Scheduler启动这个 Instance”

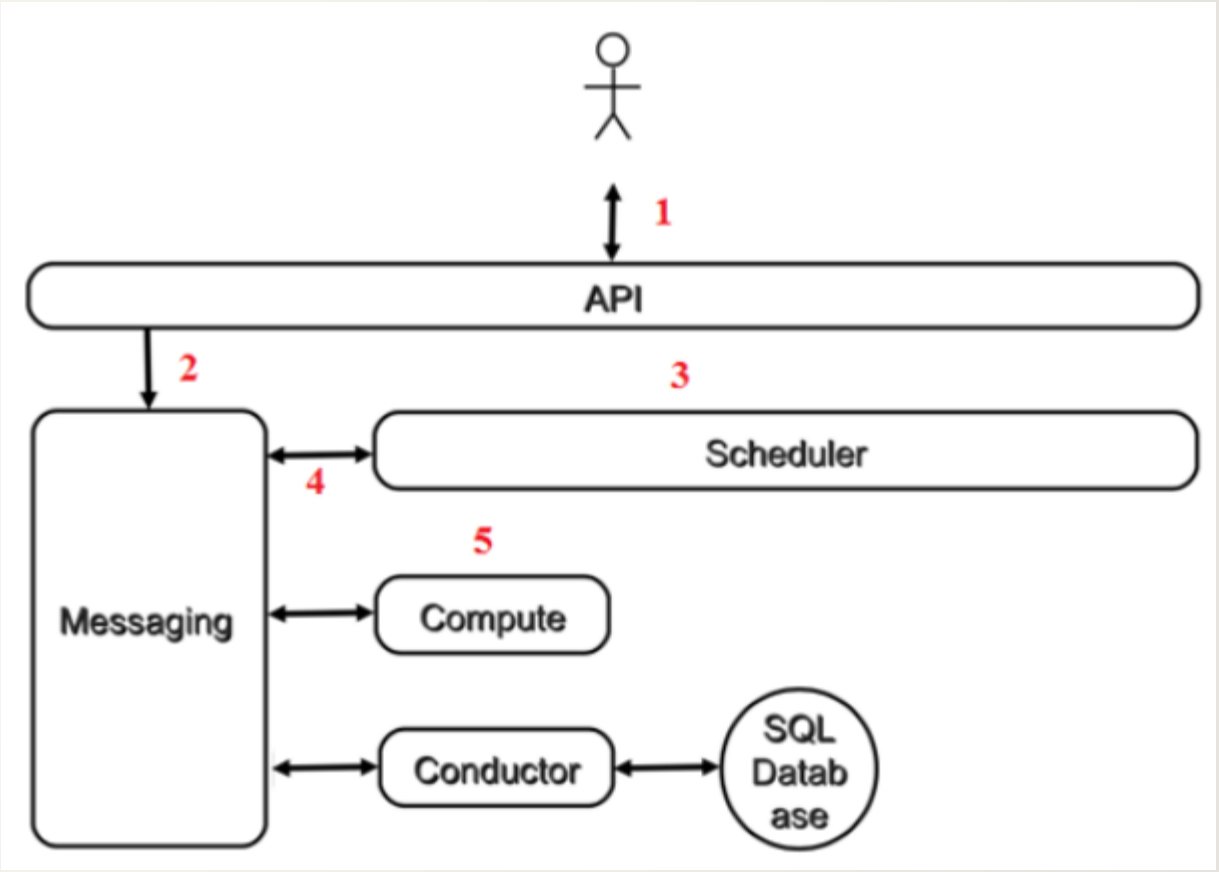
3. nova-scheduler 调度

nova-scheduler获得消息后，执行调度算法，从计算节点选择A；并发送消息”在计算节点A创建Instance“

4. nova-compute 执行操作

计算节点A的nova-compute从Meesageing中获得消息，在本节点Hypervisor上启动虚拟机。

5. 向Conductor发送消息更新数据库



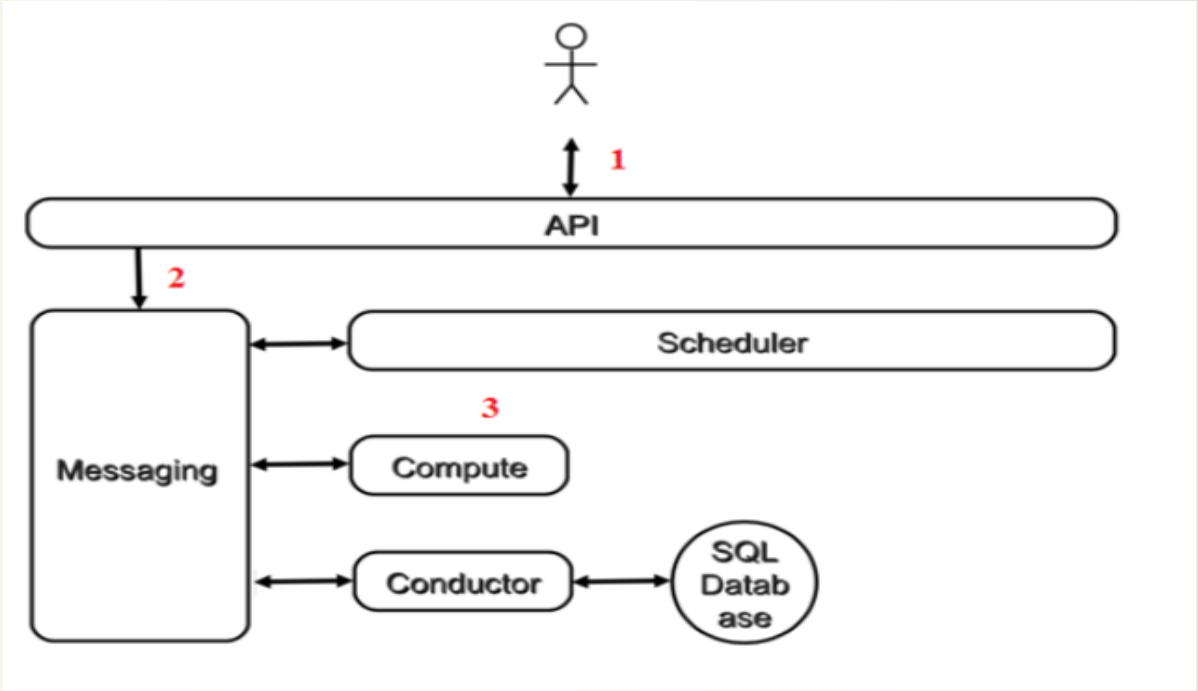
启动实例流程

2 Nova 关闭实例

1. 向 nova-api 发送请求
用户向API发送请求：” 请关闭这个Instance “

2. nova-api 发送消息
nova-api接受消息后向RabbitMQ发送消息” 关闭这个Instace “

3. nova-compute 执行操作
nova-compute接受消息后执行该操作



关闭实例流程

3 Nova控制节点配置

配置Nova应用环境

验证服务

1. 创建Nova数据库并授权（同Keystone）；
2. 在KeyStone创建用户、Service实体和API端点；
3. 控制节点安装nova管理软件；
4. 控制节点配置并启动相关Service；

5. 计算节点初始环境配置（关闭SElinux、Firewall等）；
6. 安装Nova-compute计算软件；
7. 配置并启动Nova 计算服务。

3 组件配置通用模式

安装XXX服务需要以下步骤：

1. 创建XXX数据库；
2. 在keystone 上面注册XXX；
3. 创建XXX服务的API 端点；
4. 安装XXX相关软件;
5. 配置XXX.conf；
6. 启动XXX服务。

XXX服务主配置文件存放在/etc/XXX，名为/XXX-*.conf，在配置文件中需要配置相应参数。

Nova控制节点配置

配置Nova应用环境

1. 创建Nova数据库并授权（同Keystone）

具体命令如下：

```
mysql -u root -p123456
---------------------------------
CREATE DATABASE nova_api;
CREATE DATABASE nova;
CREATE DATABASE nova_cell0;
CREATE DATABASE placement;

GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' IDENTIFIED BY 'nova';
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' IDENTIFIED BY 'nova';
..... (4个数据库)

flush privileges;
show databases;
select user,host from mysql.user;
exit
```

2. Keystone添加nova用户、添加Admin角色、配置端点

具体命令如下：

```
source admin-openrc
openstack user create --domain default --password=nova nova
openstack user list
...
```

3. 修改Nova配置 （注意以下方式和Vi区别）

```
--设定mysql连接地址；RabbitMQ消息地址等
openstack-config --set  /etc/nova/nova.conf DEFAULT transport_url  rabbit://openstack:openstack@controller
openstack-config --set  /etc/nova/nova.conf api_database connection  mysql+pymysql://nova:nova@controller/nova_
openstack-config --set  /etc/nova/nova.conf database connection  mysql+pymysql://nova:nova@controller/nova
...
```

4. 启动Nova服务

3 Nova控制节点配置

配置vnc

controller节点采用novnc虚拟桌面。

vi /etc/nova/nova.conf

```
my_ip = 172.24.2.10
vncserver_listen = 172.24.2.10
vncserver_proxyclient_address = 172.24.2.10
```

```
#server_proxyclient_address=127.0.0.1
#server_proxyclient_address=192.168.0.32
#server_proxyclient_address=114.116.47.43
server_proxyclient_address=0.0.0.0
```

重启Nova相关服务

- service openstack-nova-api restart
- service openstack-nova-cert restart
- service openstack-nova-consoleauth restart
- service openstack-nova-scheduler restart
- service openstack-nova-conductor restart
- service openstack-nova-novncproxy restart

3 Nova计算节点

配置vnc

在compute节点。

vi /etc/nova/nova.conf

```
my_ip = compute
vnc_enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = compute
novncproxy_base_url = http://172.24.2.10:6080/vnc_auto.html
```

```
# * novncproxy_host
# * novncproxy_port
#   (uri value)
#novncproxy_base_url=http://127.0.0.1:6080/vnc_auto.html
#novncproxy_base_url=http://192.168.0.32:6080/vnc_auto.html
novncproxy_base_url=http://114.116.47.43:6080/vnc_auto.html
```

重启Nova相关服务

systemctl restart openstack-nova-compute.service

3 Nova计算节点配置

配置Nova应用环境

1. 如果计算节点为新主机

   参考 OpenStack环境准备

   a) 关闭SElinx、Firewall； b) 时间同步chrony； c) 配置yum源; d) openstack客户端;

2. 安装nova计算节点相关软件包

yum install openstack-nova-compute python-openstackclient openstack-utils -y

3. 修改Nova配置

4. 启动Nova服务

OpenStack通用设计思路

通过Glance、Nova、Neutron的配置

1. API前端

组件由若干子服务构成，任意组件必含有一个API服务端以接受客户服务。

如nova-api是暴露自身API的唯一服务，任何客户请求（CLI、Dashboard、其他组件）通过nova-api发送REST API请求完成。

2. Scheduler调度

OpenStack中的某些操作可由多个实体资源去完成，由scheduler调度算法从中筛选出最合适的去完成。

3. Worker工作服务

调度算法完成调度后，真正执行任务是Worker负责执行

底层 Driver、Messaging、Database

Nova计算服务

例题

例题1：什么服务通常在控制节点上运行?

- 认证服务（KeyStone)
- 镜像服务（Glance)
- Nova 服务，如 Nova API、Nova Scheduler 和 Nova DB
- 块存储和对象存储服务
- Ceilometer 服务
- MariaDB / MySQL 和 RabbitMQ 服务运行在控制节点上
- 网络(Neutron)和网络代理的管理服务
- 编排服务（Heat)

例题2：什么服务通常在计算节点上运行?

- Nova 计算
- 网络服务

例题

例题3：计算节点上虚拟机的默认地址是什么?

虚拟机存储在计算节点的 /var/lib/nova/instances。

例题4：以下那个模块在OpenStack Nova中负责核心计算功能?

A nova-api

B nova-scheduler

C nova-compute

D nova-conductor

ABC

THANK YOU!