一、KeyStone相关概念

Project

User创建VM时从Project获取相应资源。User属于一个或者多个Project。

Service

OpenStack 的 每个Service对应若干模块。每个 Service 都会提供若干个 Endpoint，User 通过 Endpoint 访问资源和执行操作。
Endpoint 是一个网络上可访问的地址，通常是一个 URL。 Service 通过 Endpoint 暴露自己的 API。 Keystone 负责管理和维护每个 Service 的 Endpoint。

```
###查看模块对应的Endpoint
source devstack/openrc admin admin
openstack catalog list
```

Role

Role规定一组权限，属于这个Role的User拥有这些权限。

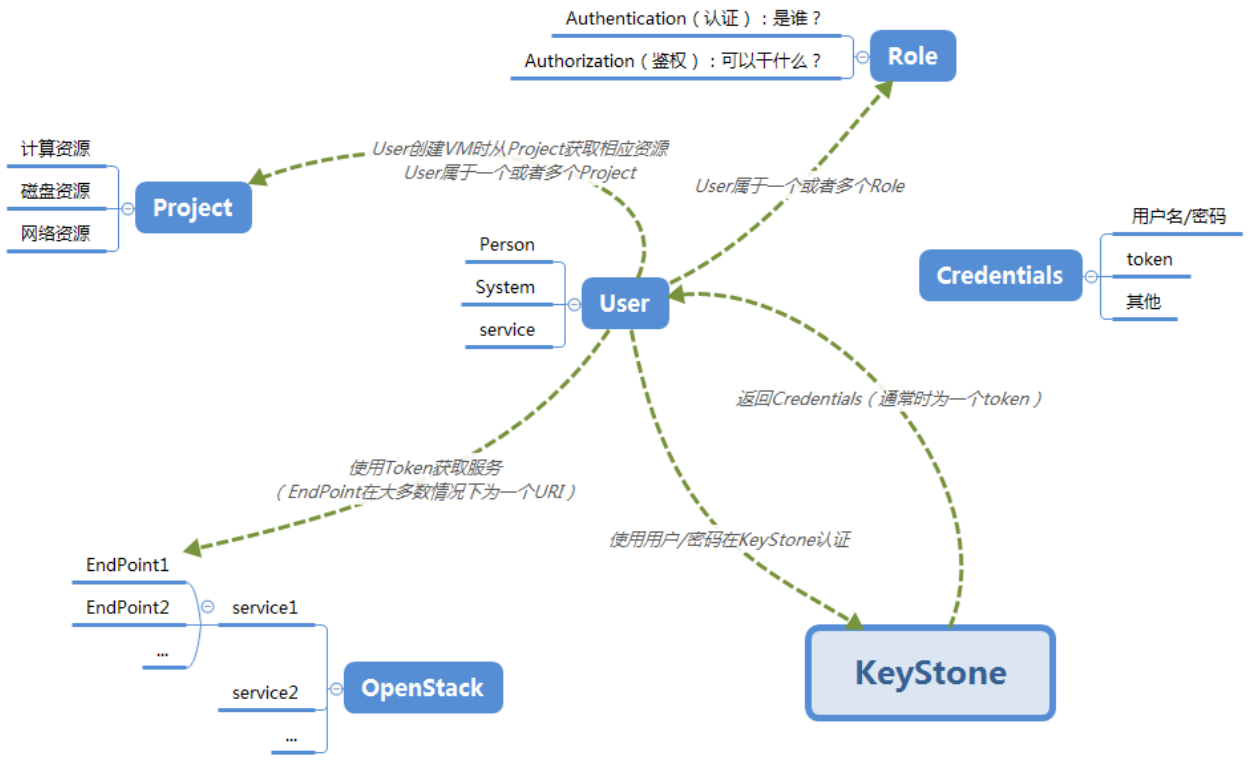
```
#查看所有Role/User/Project
openstack role/user/project list
```

Service 通过各自的 policy.json 文件对 Role 进行访问控制。OpenStack 默认配置只区分 admin 和非 admin Role。 如果需要对特定的 Role 进行授权，可以修改 policy.json

```
###policy.json文件位置
/etc/<模块名>/policy.json
###glance模块示例：  操作动作：允许的Role（空时表示所有角色都允许）
    "get_images": "",
    "modify_image": "",
```

```
    "publicize_image": "role:admin",
    "communitize_image": "",
    "copy_from": "",
```

用户发起一次动作的流程如下



KeyStone的日志

Keystone 主要有两个日志： keystone.log 和 keystone_access.log
如果需要得到最详细的日志信息，可以在 /etc/keystone/keystone.conf 中打开 debug 选项

二、Glance相关概念

glance-api

glance-api 是系统后台运行的服务进程。对外提供 REST API，响应 image 查询、获取和存储的调用。
glance-api 不会真正处理请求。只进行转发。如果是与 image metadata（元数据）相关的操作，glance-api 会把请求转发给 glance-registry；如果是与 image 自身存取相关的操作，glance-api 会把请求转发给该 image 的 store backend。

glance-registry

glance-registry 是系统后台运行的服务进程，负责处理和存取 image 的 metadata。

Store backend

Glance 自己并不存储 image，真正的 image 是存放在 backend 中的。backend可以理解为实际存储镜像的物理位置。

```
# /etc/glance/glance-api.conf中glance_store的配置
[glance_store] filesystem_store_datadir = /opt/stack/data/glance/images/
```

image 存放在控制节点本地目录 /opt/stack/data/glance/images/ ，并且以ID来命名

```
[stack@localhost devstack]$ cd /opt/stack/data/glance/images/
[stack@localhost images]$ ll
total 33096
-rw-r----- 1 stack stack  3740163 Apr 20 17:01 1b22ef03-edad-4298-af94-7c70887b7d9b
-rw-r----- 1 stack stack  4979632 Apr 20 17:01 22117259-fe19-4a15-b14d-97f43842321e
-rw-r----- 1 stack stack 25165824 Apr 20 17:01 a89d2915-45e1-4b94-8efe-bbfcc4e00616
```

Database

一般情况下每个模块都有一些信息需要保存在数据库中，模块直接可以共用一个数据库也可以单独使用一个。默认情况下数据库是MySQL。
Image 的 metadata 会保持到 database 中，默认是 MySQL。

```
##在stack用户下登录mysql
mysql
show database;
```

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| cinder             |
| glance             |
| information_schema |
| keystone           |
| mysql              |
| neutron            |
| nova               |
| nova_api           |
| nova_cell0         |
| performance_schema |
| test               |
+--------------------+
11 rows in set (0.00 sec)
```

```
##切换到glance数据库有如下表
use glance;
show tables;
```

```
MariaDB [glance]> show tables;
+----------------------------------+
| Tables_in_glance                 |
+----------------------------------+
| alembic_version                  |
| artifact_blob_locations          |
| artifact_blobs                   |
| artifact_dependencies            |
| artifact_properties              |
| artifact_tags                    |
| artifacts                        |
| image_locations                  |
| image_members                    |
| image_properties                 |
| image_tags                       |
| images                           |
| metadef_namespace_resource_types |
| metadef_namespaces               |
| metadef_objects                  |
| metadef_properties               |
| metadef_resource_types           |
| metadef_tags                     |
| migrate_version                  |
| task_info                        |
| tasks                            |
+----------------------------------+
21 rows in set (0.01 sec)
```

OpenStack的表结构说明文档？

Glance的日志

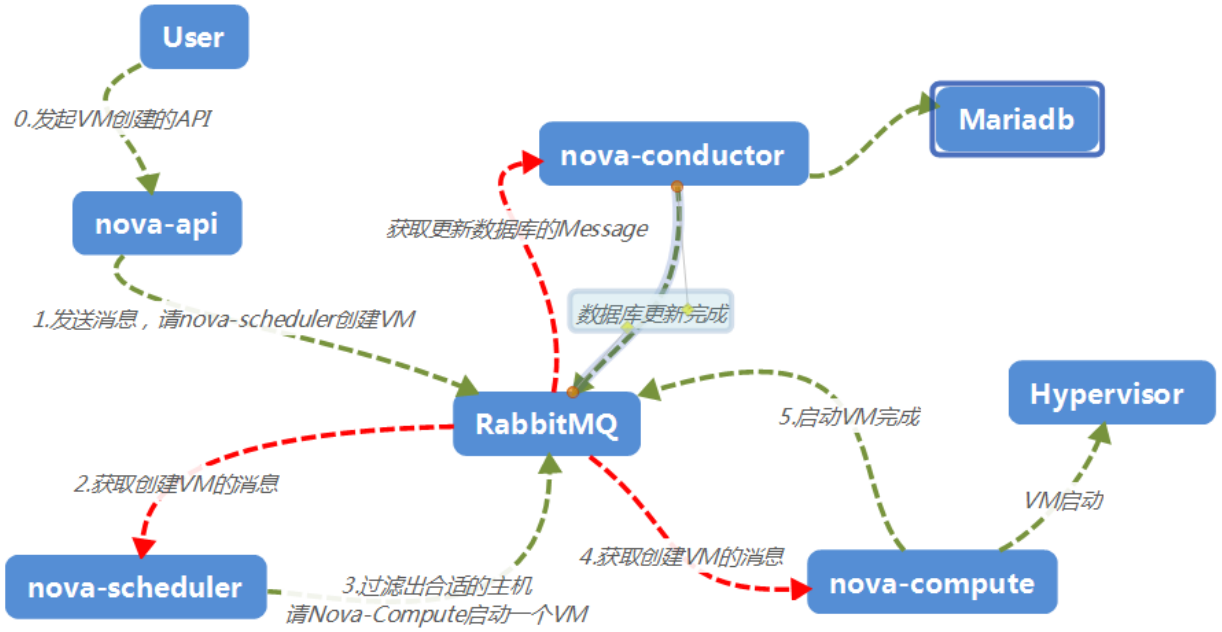
g-api 窗口显示 glance-api 日志，记录 REST API 调用情况 g-reg 窗口显示 glance-registry 日志，记录 Glance 服务处理请求的过程以及数据库操作

三、Nova的相关概念

Nova由很多子服务组成，不同子服务可以部署到不同节点上。 OpenStack的节点一般分为两类：计算节点和控制节点。 计算节点上安装了 Hypervisor，上面运行虚拟机。控制节点主要负责消息管理、元数据管理等任务。对Nova来说只有nova-compute可以部署在计算节点，其他服务全部部署在控制节点。

```
#查看nova的节点部署情况
nova service-list
```

nova各模块相互协调创建VM的过程



Nova的子服务：

1.nova-api

接收和响应客户的API调用。
除了提供 OpenStack自己的API，nova-api还支持Amazon EC2 API。
nova-api向外界暴露HTTP REST API接口。 当API被调用时，nova-api会执行下面的

操作。

A.检查客户端传人的参数是否合法有效。

B.调用 Nova 其他子服务的处理客户端 HTTP 请求。

C.格式化 Nova 其他子服务返回的结果并返回给客户端。

2.nova-scheduler

虚机调度服务，负责决定在哪个计算节点上运行虚机。当启动虚机失败时一般查看这个日志n-sch.log。
创建 Instance 时，用户会提出资源需求，OpenStack 将这些需求定义在 flavor 中，用户指定用 flavor。
flavor定义了VCPU，RAM，DISK 和 Metadata四类信息，其中Metadata定义了一些而外的信息，如CPU架构等。

```
#查看所有flavor
nova flavor-list
```

```
[stack@localhost devstack]$ nova flavor-list
/usr/lib/python2.7/site-packages/novaclient/client.py:278: UserWarning: The 'tenant_id' argument is deprecated in Ocata and its use may result in errors in future releases. As 'p
roject_id' is provided, the 'tenant_id' argument will be ignored.
  warnings.warn(msg)
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| 1  | m1.tiny   | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 42 | m1.nano   | 64        | 0    | 0         |      | 1     | 1.0         | True      |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
| 84 | m1.micro  | 128       | 0    | 0         |      | 1     | 1.0         | True      |
| c1 | cirros256 | 256       | 0    | 0         |      | 1     | 1.0         | True      |
| d1 | ds512M    | 512       | 5    | 0         |      | 1     | 1.0         | True      |
| d2 | ds1G      | 1024      | 10   | 0         |      | 1     | 1.0         | True      |
| d3 | ds2G      | 2048      | 10   | 0         |      | 2     | 1.0         | True      |
| d4 | ds4G      | 4096      | 20   | 0         |      | 4     | 1.0         | True      |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
```

创建时nova通过filter来选择宿主机，相关配置在/etc/nova/nova.conf中scheduler_driver，scheduler_default_filters 这三个参数来配置 nova-scheduler。

```
###默认的filter配置
scheduler_default_filters = RetryFilter,AvailabilityZoneFilter,RamFilter,D
                                                ComputeCapabilitiesFilter,Imag
                                                ServerGroupAffinityFilter,Same
scheduler_driver = filter_scheduler
```

获得过滤的节点后， nova-scheduler 根据计算节点空闲的内存量计算权重值，空闲内存越多，权重越大，instance 将被部署到当前空闲内存最多的计算节点上。

3.nova-compute

管理虚机的核心服务，通过调用 Hypervisor API 实现虚机生命周期管理。实际上该服务不会和Hypervisor直接交互，而是通过LibVirt和不同的Hypervisor交互。 这里LibVirt是Hyervisor的一种驱动（通用驱动？支持KVM、XEN）。

```
#配置文件/etc/nova/nova.conf 中的这个配置项决定了，该节点使用的驱动类型。
compute_driver = libvirt.LibvirtDriver
```

实际上对于某个特定的计算节点OpenStack只允许加载一种driver，
在/opt/stack/nova/nova/virt/ 目录下可以看到OpenStack安装后自带的Driver。
nove-compute的主要工作包括：

A.OpenStack发送性能统计报告

nove-compute通过Hypervisor 来获取VM的这些信息，在n-cpu.log日志中该可以看到这些报告信息。

```
2017-04-20 17:05:45.222 2771 INFO nova.compute.resource_tracker [req-6f498281-c077-4a61-8293-d434a2435795 - -] Final resource view:
name=localhost.localdomain phys_ram=8023MB used_ram=512MB phys_disk=35GB used_disk=0GB total_vcpus=4 used_vcpus=0 pci_stats=[]
2017-04-20 17:05:45.224 2771 DEBUG nova.compute.resource_tracker [req-6f498281-c077-4a61-8293-d434a2435795 - -] Compute_service record updated for
localhost.localdomain:localhost.localdomain _update_available_resource /opt/stack/nova/nova/compute/resource_tracker.py:626
```

B.对VM的生命周期进行管理

nove-compute可以对VM进行launch、shutdown、reboot、suspend、resume、terminate、resize、migration、snapshot等操作。
nova-compute 创建 instance 的过程可以分为 4 步，n-cpu.log清晰的记录了整个过程：
启动日志一个关键字Starting instance...开始

```
[instance: ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d] Starting instance... _do_build_and_run_instance /opt/stack/nova/nova/compute/manager.py:1751
```

以关键字VM Started结束

```
[instance: ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d] VM Started
```

1.为 instance 准备资源
分配memory、disk、vcpu

```
[instance: ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d] Attempting claim: memory 512 MB, disk 1 GB, vcpus 1 CPU
[instance: ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d] Total memory: 8023 MB, used: 1024.00 MB
[instance: ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d] memory limit: 12034.50 MB, free: 11010.50 MB
[instance: ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d] Total disk: 35 GB, used: 1.00 GB
[instance: ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d] disk limit: 35.00 GB, free: 34.00 GB
[instance: ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d] Total vcpu: 4 VCPU, used: 1.00 VCPU
[instance: ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d] vcpu limit not specified, defaulting to unlimited
Require both a host and instance NUMA topology to fit instance on host. numa_fit_instance_to_host /opt/stack/nova/nova/virt/hardware.py:1328
[instance: ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d] Claim successful
```

分配网络资源

```
[instance: ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d] Allocating IP information in the background.  allocate network async /opt/stack/nova/nova/compute/manager.py:1386
```

2.创建 instance 的镜像文件

首先，nove-compute从Glance下载 image到本地，然后将其作为 backing file 创建 instance 的镜像文件。
nove-compute不会重复下载Glance，下载的image被存放在 /opt/stack/data/nova/instances/_base

```
#/etc/nova/nova.conf决定了文件的存放位置
instances_path = /opt/stack/data/nova/instances
```

下载的 image 文件被命名为 60bba5916c6c90ed2ef7d3263de8f653111dd35f，这是 image id 的 SHA1 哈希值。
/opt/stack/data/nova/instances目录下根据instance的id存放有不同的文件夹，其中存放了VM的console.log日志，disk.info文件中似乎定义了很重要的东西！！

```
[stack@localhost 73cf295a-640b-4977-91ae-3c33e8debc87]$ ll
total 8564
-rw-rw-r-- 1 stack qemu       40862 Apr 21 17:01 console.log
-rw-r--r-- 1 stack libvirtd     172 Apr 20 17:21 disk.info
-rw-rw-r-- 1 qemu  qemu     4979632 Apr 20 17:21 kernel
-rw-rw-r-- 1 qemu  qemu     3740163 Apr 20 17:21 ramdisk
```

下载完成image后nova-compute调用qemu-img命令将image装换装换为合适back_file
3.创建 instance 的 XML 定义文件
ocata版本没有找到这个配置文件，但是在日志中打印出来了？？

```
<domain type="kvm">
  <uuid>ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d</uuid>
  <name>instance-00000003</name>
  <memory>524288</memory>
  <vcpu>1</vcpu>
  <metadata>
    <nova:instance xmlns:nova="http://openstack.org/xmlns/libvirt/nova/1.0
      <nova:package version="15.0.4"/>
```

```
      <nova:name>1111</nova:name>
      <nova:creationTime>2017-04-21 08:23:50</nova:creationTime>
      <nova:flavor name="m1.tiny">
        <nova:memory>512</nova:memory>
        <nova:disk>1</nova:disk>
        <nova:swap>0</nova:swap>
        <nova:ephemeral>0</nova:ephemeral>
        <nova:vcpus>1</nova:vcpus>
      </nova:flavor>
      <nova:owner>
        <nova:user uuid="2ee93c4ca63c4257b0545230482947ff">admin</nova:use
        <nova:project uuid="33c8ed8898ee44539376b6a6a2d5c361">admin</nova:
      </nova:owner>
    </nova:instance>
  </metadata>
  <sysinfo type="smbios">
    <system>
      <entry name="manufacturer">OpenStack Foundation</entry>
      <entry name="product">OpenStack Nova</entry>
      <entry name="version">15.0.4</entry>
      <entry name="serial">11a381de-55ca-4238-b43b-4b526afc9053</entry>
      <entry name="uuid">ba28c5ab-5e8c-4b45-8ddf-e8f5a7d5378d</entry>
      <entry name="family">Virtual Machine</entry>
    </system>
  </sysinfo>
  <os>
    <type>hvm</type>
    <boot dev="hd"/>
    <smbios mode="sysinfo"/>
  </os>
  <features>
    <acpi/>
    <apic/>
  </features>
  <cputune>
    <shares>1024</shares>
  </cputune>
  <clock offset="utc">
    <timer name="pit" tickpolicy="delay"/>
    <timer name="rtc" tickpolicy="catchup"/>
```

```
    <timer name="hpet" present="no"/>
  </clock>
  <cpu match="exact">
    <topology sockets="1" cores="1" threads="1"/>
  </cpu>
  <devices>
    <disk type="block" device="disk">
      <driver name="qemu" type="raw" cache="none" io="native"/>
      <source dev="/dev/disk/by-path/ip-172.24.2.216:3260-iscsi-iqn.2010-1
      <target bus="virtio" dev="vda"/>
      <serial>cccf09f0-b133-4d9b-8723-fdd8c8483bef</serial>
    </disk>
    <interface type="bridge">
      <mac address="fa:16:3e:67:62:95"/>
      <model type="virtio"/>
      <source bridge="qbrfd362a0a-73"/>
      <target dev="tapfd362a0a-73"/>
    </interface>
    <serial type="file">
      <source path="/opt/stack/data/nova/instances/ba28c5ab-5e8c-4b45-8ddf
    </serial>
    <serial type="pty"/>
    <graphics type="vnc" autoport="yes" keymap="en-us" listen="127.0.0.1"/
    <video>
      <model type="cirrus"/>
    </video>
    <memballoon model="virtio">
      <stats period="10"/>
    </memballoon>
  </devices>
</domain>
```

4.创建虚拟网络并启动虚拟机

4.nova-conductor 负责更新数据库

5.Console Interface

用户和nova直接交互的模块，包括nova-console（nova-novncproxy、nova-

spicehtml5proxy、nova-xvpnvncproxy ）、nova-consoleauth、nova-cert。
nova-novncproxy，基于 Web 浏览器的 VNC 访问
nova-spicehtml5proxy，基于 HTML5 浏览器的 SPICE 访问
nova-xvpnvncproxy，基于 Java 客户端的 VNC 访问
nova-consoleauth，负责对访问虚机控制台请亲提供 Token 认证
nova-cert，提供 x509 证书支持

6.Message Queue

Nova的子模块不会相互通信，所有通信通过Message Queue。OpenStack 默认是RabbitMQ。