

- 云计算

- OpenStack

- Keystone

- Keystone 概述
    - keystone 的名词概念
    - Keystone 认证管理

- token

- Domain

# 云计算

---

## OpenStack

### Keystone

#### Keystone 概述

- 管理用户及其权限
- 维护 OpenStack Services 的 Endpoint
- Authentication（认证）和 Authorization（授权）

#### 验证用户

验证用户的最简单的方法是请求凭据（登录+密码， 登录+密钥等）， 并通过某些数据库进行检查。单独的服务用户还能忍，百度那么多的服务一个一个的认证登陆，那不扯蛋一样的嘛.

#### 认证包括的内容和关系

- 第一个基础是用户。他们代表某人或某事， 可以通过Keystone访问。用户具有可以检查的凭据， 如密码或API密钥。
- 第二个是租户。它代表了所谓的Nova项目， 这意味着聚合每个服务中的资源数量。例如， 租户可以在Nova中有一些机器，在Swift / Glance中有一些图像，在Neutron中有几对网络。默认情况下， 用户始终绑定到某些租户。
- 第三种与授权相关的对象类型是角色。它们表示假定具有对资源的一些访问的一组用户， 例如， 一些虚拟机在Nova和一些镜像在Glance。用户可以添加到全局或租户中的任何角色。在第一种情况下， 用户获得由角色对所有租户中的资源暗示的访问;在第二种情况下， 用户的访问被限制到相应租户的资源。例如， 用户可以是所有租户的操作者和他自己的项目的管理员。

#### keystone 的名词概念



- **User** -- 指代任何使用 OpenStack 的实体，可以是真正的用户，其他系统或者服务。当 User 请求访问 OpenStack 时，Keystone 会对其进行验证。除了 admin 和 demo，OpenStack 也为 nova、cinder、glance、neutron 服务创建了相应的 User。  
admin 也可以管理这些 User。
- **Credentials** -- User 用来证明自己身份的信息，可以是：
  1. 用户名/密码
  2. Token
  3. API Key
  4. 其他高级方式
- **Authentication** -- 是 Keystone 验证 User 身份的过程。User 访问 OpenStack 时向 Keystone 提交用户名和密码形式的 Credentials，Keystone 验证通过后会给 User 签发一个 Token 作为后续访问的 Credential。
- **Token** -- 是由数字和字母组成的字符串，User 成功 Authentication 后由 Keystone 分配给 User。
  - Token 用做访问 Service 的 Credential
  - Service 会通过 Keystone 验证 Token 的有效性
  - Token 的有效期默认是 24 小时
- **Project** -- 用于将 OpenStack 的资源（计算、存储和网络）进行分组和隔离。根据 OpenStack 服务的对象不同，Project 可以是一个客户（公有云，也叫租户）、部门或者项目组（私有云）。
  - 资源的所有权是属于 Project 的，而不是 User。
  - 在 OpenStack 的界面和文档中，Tenant / Project / Account 这几个术语是通用的，但长期看会倾向使用 Project
  - 每个 User（包括 admin）必须挂在 Project 里才能访问该 Project 的资源。

- 一个User可以属于多个 Project。
- admin 相当于 root 用户，具有最高权限
- Service -- 包括 Compute (Nova)、Block Storage (Cinder)、Object Storage (Swift)、Image Service (Glance)、Networking Service (Neutron) 等。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 告知自己的 API。Keystone 负责管理和维护每个 Service 的 Endpoint。
- Role -- 角色(VIP)。
  - Authentication (认证) 解决的是“你是谁？”的问题
  - Authorization (鉴权) 解决的是“你能干什么？”的问题

Keystone 是借助 Role 来实现 Authorization 的，Keystone 定义 Role，可以为 User 分配一个或多个 Role。Service 通过各自的/etc/\*\*/policy.json 对 Role 进行访问控制，决定每个 Role 能做什么事情

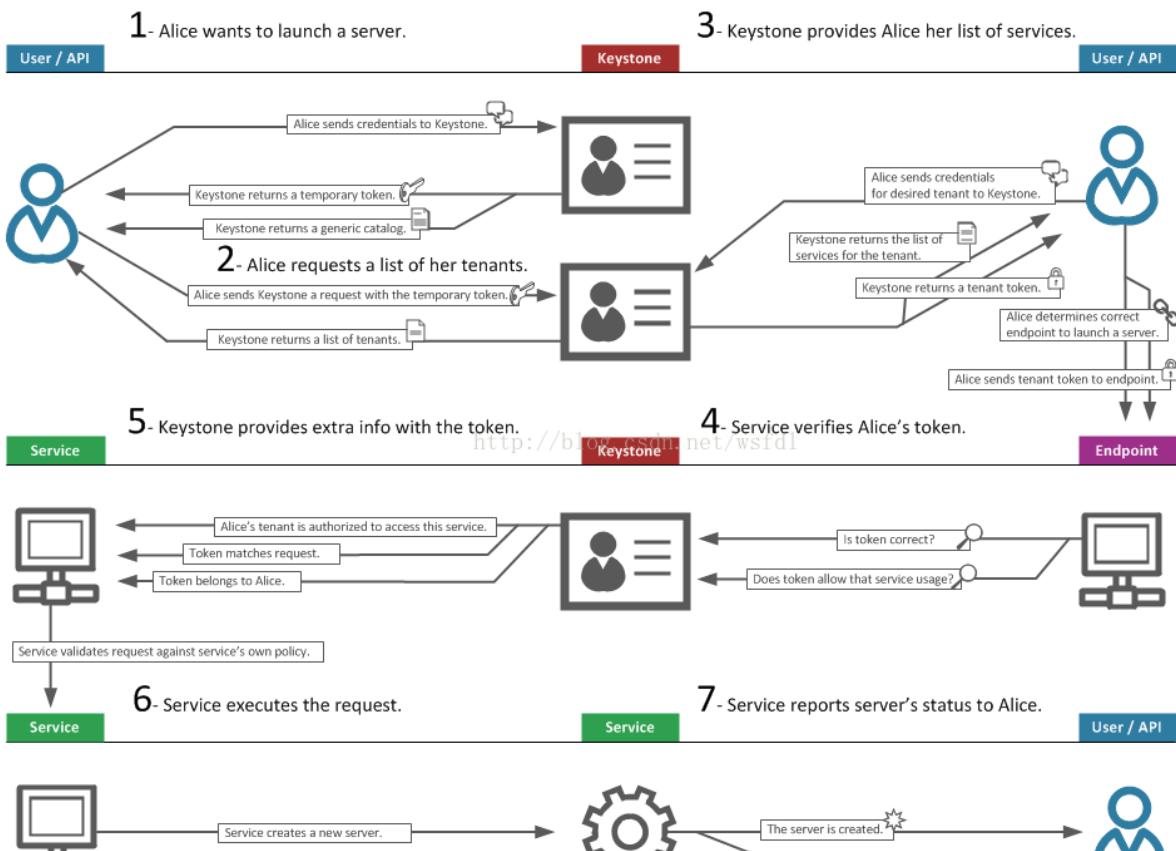
```
{  
    "context_is_admin": "role:admin",  
    "admin_or_owner": "is_admin:True or project_id:%(project_id)s",  
    "default": "rule:admin_or_owner",  
  
    "cells_scheduler_filter:TargetCellFilter": "is_admin:True",  
  
    "compute:create": "",  
    "compute:create:attach_network": "",  
    "compute:create:attach_volume": "",  
    "compute:create:forced_host": "is_admin:True",  
    "compute:get_all": "",  
}
```

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。

**openstack** 适用于全局，可管理和查看各类信息。

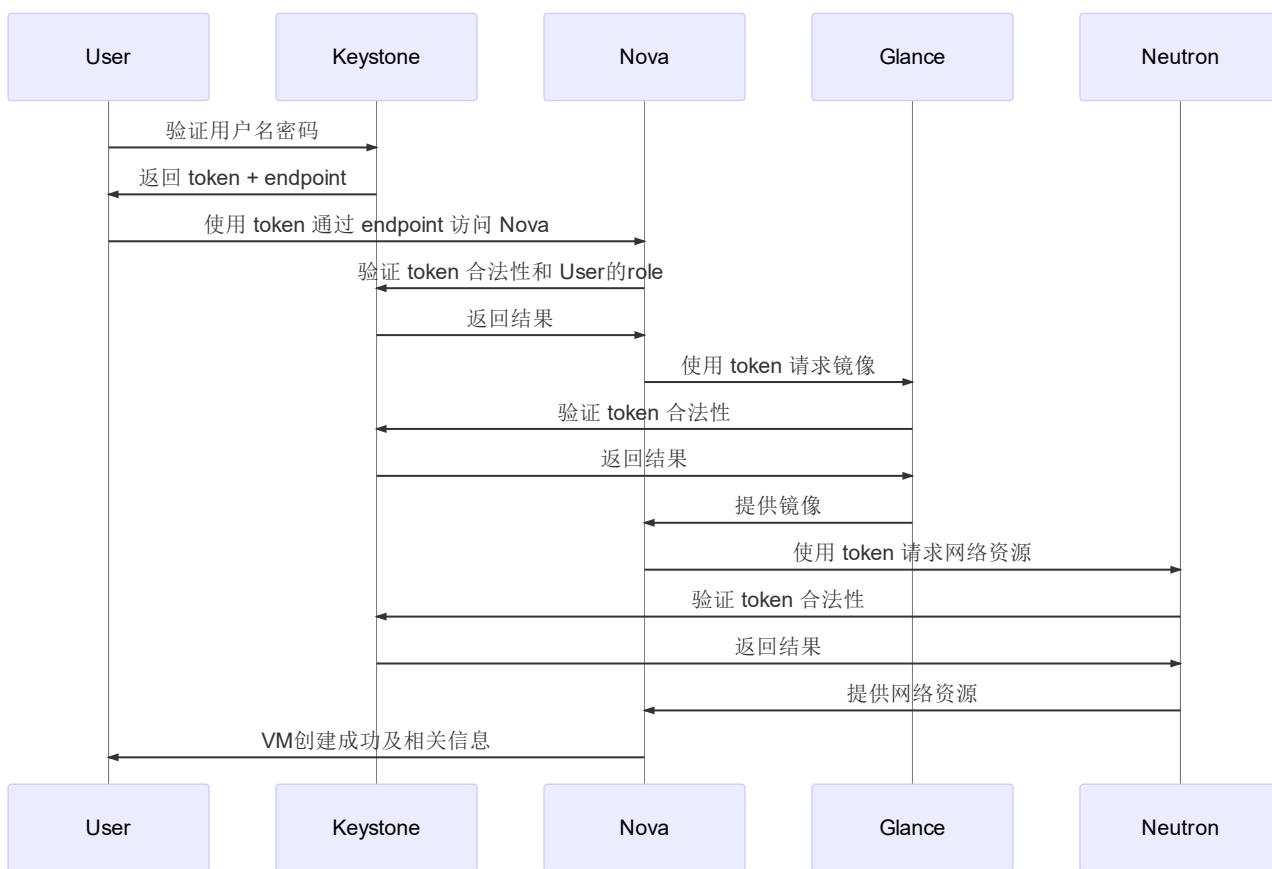
**keystone-manage** 用于启动 keystone 模块数据、初始化数据库、生成 SSL 相关的证书和私钥。

## Keystone 认证管理



1. 用户/API 想创建一个实例，首先会将自己的**credentials**发给keystone。认证成功后，keystone 会颁给用户/API一个临时的令牌(**Token**) 和一个访问服务的**Endpoint**。 PS:**Token**没有永久的
2. 用户/API 把临时**Token**提交给keystone， keystone并返回一个**Tenant(Project)**
3. 用户/API 向keystone发送带有特定租户的凭证，告诉keystone用户/API在哪个项目中， keystone收到请求后，会发送一个项目的**token** 到用户/API PS: 第一个**Token**是来验证用户/API 是否有权限与keystone通信，第二个**Token**是来验证用户/API是否有权限访问我 keystone的其它服务。用户/API 拿着**token**和**Endpoint**找到可访问服务
4. 服务向keystone进行认证， **Token**是否合法，它允许访问使用该服务(判断用户/API中**role**权限)?
5. keystone向服务提供额外的信息。用户/API是允许方法服务，这个**Token**匹配请求，这个**Token** 是用户/API的
6. 服务执行用户/API发起的请求， 创建实例
7. 服务会将状态报告给用户/API。最后返回结果，实例已经创建

创建一个实例



## token

Provider方式	生成方式/长度/加密方式	优点	缺点
UUID	uuid.uuid4().hex, 32字符，没有加密方式。	生成的Token，长度较短，使用方便。 url使用方便，回收的方式，就是从后端删除即可，实现逻辑较为清晰。	需要持久化后端存储，每次访问需要keystone相关服务进行认证。
PKI	cms_sign_data ()，使用base64 encoded进行编码（替换不安全的字符），粗略统计过长度4602字符，使用 Cryptographic Message Syntax (CMS) 进行加密（默认的方式sha256）。	使用证书及私钥生成，可以线下验证（不需要走keystone认证服务）	长度负载重，不推荐用于生产部署： keystone-manage pki_setup，需要使用由一个受信任的CA颁发的证书
PKIZ	同PKI，使用base64url encoding进行编码，在此基础上，使用压缩机制，长度上减小了一半，并且Token使用PKIZ_开头。	较PKI长度上缩小了很多	长度负载较重，同上PKI

Fernet	<p>MessagePacked 负载数据，并使用 <code>crypto.encrypt(payload)</code> 加密，粗略统计长度 183 字符。</p> <p>使用对称加密，为了安全度提升，同时支持定期轮换。</p>	<p>设计的逻辑，引入序列化，负载格式加以控制，基于此进行加密，长度比 PKI (Z) 要短。</p>	对称加密算法，安全性低
--------	--	---	-------------

## Domain

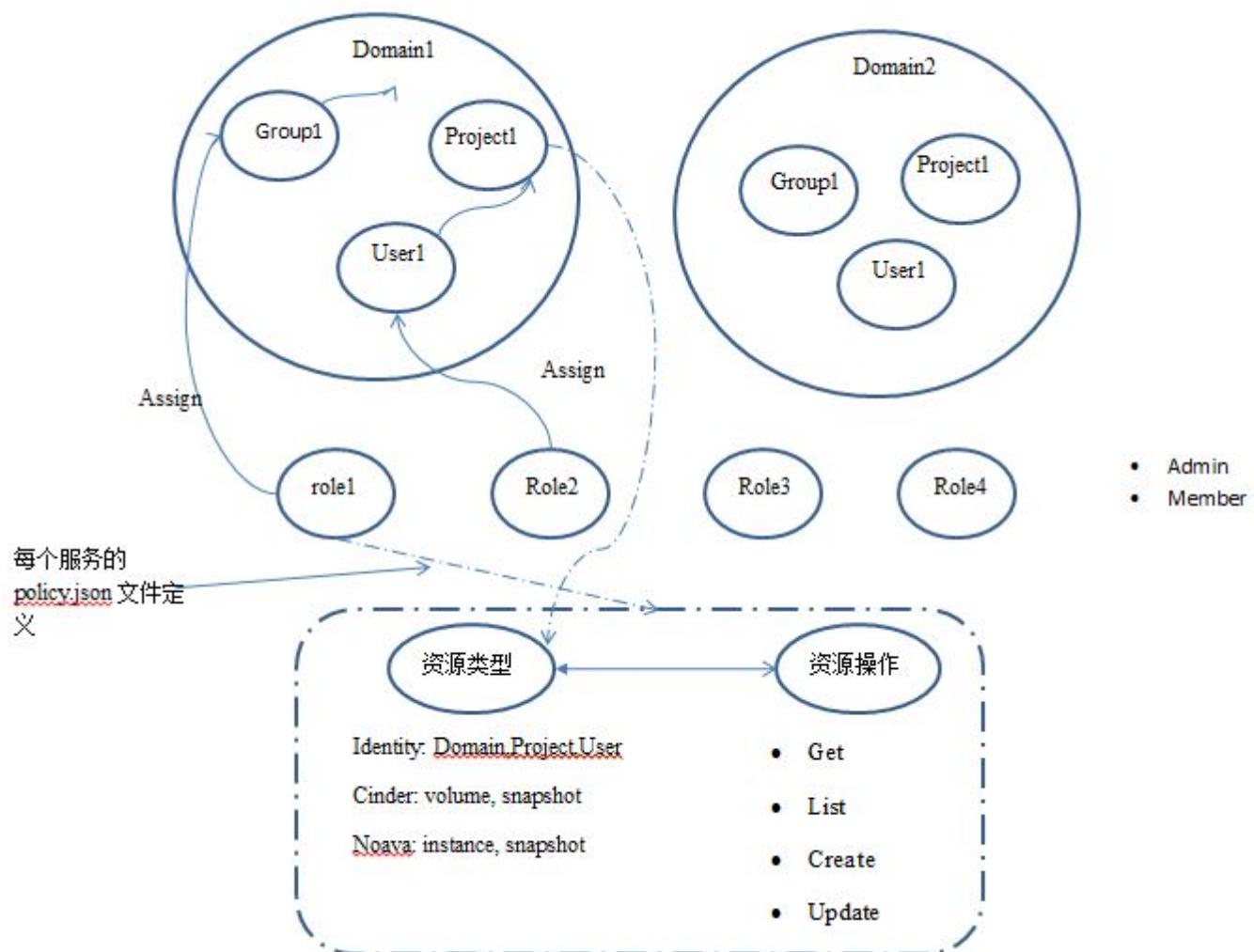
Domain -- 表示 project 和 user 的集合，在公有云或者私有云中常常表示一个客户

Group -- 一个 domain 中的部分用户的集合

Project -- IT 基础设施资源的集合，比如虚机，卷，镜像等

Role -- 角色，表示一个 user 对一个 project resource 的权限

Token -- 一个 user 对于某个目标 (project 或者 domain) 的一个有限时间段内的身份令牌



1. Domain 可以认为是 project, user, group 的 namespace。一个 domain 内，这些元素的名称不可以重复，但是在两个不同的 domain 内，它们的名称可以重复。因此，在确定这些元素时，需要同时使用它们的名称和它们的 domain 的 id 或者 name。
2. Group 是一个 domain 部分 user 的集合，其目的是为了方便分配 role。给一个 group 分配 role，结果会给 group 内的所有 users 分配这个 role。

3. Role 是全局（global）的，因此在一个 keystone 管辖范围内其名称必须唯一。role 的名称没有意义，其意义在于 policy.json 文件根据 role 的名称所指定的允许进行的操作。
4. 简单地，role 可以只有 admin 和 member 两个，前者表示管理员，后者表示普通用户。但是，结合 domain 和 project 的限定，admin 可以分为 cloud admin，domain admin 和 project admin。
5. policy.json 文件中定义了 role 对某种类型的资源所能进行的操作，比如允许 cloud admin 创建 domain，允许所有用户创建卷等
6. project 是资源的集合，其中有一类特殊的project 是 admin project。通过指定 admin\_project\_domain\_name 和 admin\_project\_name 来确定一个 admin project，然后该 project 中的 admin 用户即是 cloud admin。
7. Token 具有 scope(范围) 的概念，分为 unscoped token, domain-scoped token 和 project-scoped token。下文有说明。

Token 是针对不同 scope 认证状态，这里的 scope 是指 project 和 domain，因此一共有三种 scoped token：

1. project-scoped token: 针对一个 project 的 token，它包含 service catalog, a set of roles, 和那个 project 的详细信息
  2. domain-scoped token: 针对一个 domain 的 token，它具有有限的使用场景，只用于 domain 层面的操作。与 project-scoped 相比，它只具有优先的 sevice catalog
  3. unscoped token: 当既不指定 project 也不指定 domain 为 scope，同时 user 也没有 default project 时获得的 token，这是一种特殊的token。
- 
-