1. Openstack中认证的范围（Authorization Scope）有三种。认证以后会根据当时认证的范围给予token，根据这个token来给予对Openstack中资源执行的权限。

第一种是无指定范围的Token(Unscoped token)。无指定范围的token只单纯的用于证明你的身份，用于以后生成指定范围的token，而不需要重复的输入身份信息。使用无范围指定的token用两个条件，第一是申请的时候没有指定认证的范围（即是无指定project,也没有指定domain）

第二种是project层次的Token(Project-scoped token). 这种Token使用户在指定的项目中执行操作。

第三种是Domain层次的Token(Domain-scoped token).这种Token使用户在指定的Domain以及所包含的projects中执行操作。

2. 当我们登陆进OpenStack的dashboard的时候，肯定是属于某一个身份用户。此时点击右上角的project我们可以看到的project, 代表我们是这个project的member.

3.OpenStack的整体架构划分以及作用

OpenStack是一个由美国宇航局NASA与Rackspace公司共同开发的云计算平台项目，且通过Apache许可证授权开放源码。它可以帮助服务商和企业实现类似于Amazon EC2和S3的云基础架构服务。下面是OpenStack官方给出的定义：  
  
OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.

OpenStack是一个可以管理整个数据中心里大量资源池的云操作系统，包括计算、存储及网络资源。管理员可以通过管理台管理整个系统，并可以通过web接口为用户划定资源。  
  
由以上可以知道OpenStack的主要目标是管理数据中心的资源，简化资源分派。它管理三部分资源，分别是：  
***计算资源***：OpenStack可以规划并管理大量虚机，从而允许企业或服务提供商按需提供计算资源；开发者可以通过API访问计算资源从而创建云应用，管理员与用户则可以通过web访问这些资源；

***存储资源***：OpenStack可以为云服务或云应用提供所需的对象及块存储资源；因对性能及价格有需求，很多组织已经不能满足于传统的企业级存储技术，因此OpenStack可以根据用户需要提供可配置的对象存储或块存储功能；

***网络资源***：如今的数据中心存在大量的设置，如服务器、网络设备、存储设备、安全设备，而它们还将被划分成更多的虚拟设备或虚拟网络；这会导致IP地址的数量、路由配置、安全规则将爆炸式增长；传统的网络管理技术无法真正的可高扩展、高自动化地管理下一代网络；因而OpenStack提供了插件式、可扩展、API驱动型的网络及IP管理；

4.

Openstack的版本演变，OpenStack的每个主版本系列以字母表顺序（A~Z）命名，以年份及当年内的排序做版本号。例如第一个版本为Austin，然后发展为Bexar, Cactus, Diablo…等等一直到现在的Rocky以及在开发中的Stein.

5.

OpenStack的架构及组件

|  |  |  |
| --- | --- | --- |
| **服务** | **组件名** | **描述** |
| 控制台 | Horizon | 用户通过该服务与OpenStack的各服务进行交互，如启动虚机实例、分配IP地址、设置访问控制等； |
| 计算 | Nova | 按需分派并管理虚机； |
| 网络 | Neutron | 通常是计算服务通过该服务管理网络设置之间的连接，也可以允许终端用户创建并添加网络接口；通过一个插件式架构支持大量网络广商设备及网络技术； |
| **存储类** | | |
| 对象存储 | Swift | 存取文件，但并不提供传统挂载式的文件服务； |
| 块存储 | Cinder | 向虚机提供可用于持久存储的块存储服务； |
| **共用服务** | | |
| 身份服务 | Keystone | 为OpenStack提供认证及授权服务。 |
| 镜像服务 | Glance | 提供虚机镜像的注册服务；同时计算服务也使用该服务分派实例； |
| 计量/监控服务 | Ceilometer | 用于计费、基准测试及数据统计等功能 |
| **更高层服务** | | |
| 编排组织服务 | Heat | 使用自带的HOT模板或AWS的CloudFormation模板，通过OpenStack中各服务的REST API，将各组件的资源组织形成云应用； |

随着版本的升级，Openstack所包含的模块越来越多，表中所列只是其中几个基础的模块。每个模块都有自己的API.

6. Tenant ID就是Project ID.

8.

Openstack中有一个Openstack库，这个库是用于在CLI上管理Openstack使用的。安装了以后才可以使用，例如执行命令：

openstack project create --domain default --description "Service Project" service

9.

OpenStack 生成一个Instance流程图：

A close up of text on a white background

Description automatically generated

Step1: Horizon用户界面或者CLI获取用户认证口令，然后向Keystone发送验证信息。

Step2: Keystone控件验证口令，然后向用户返回token. 用户将使用该token发送后续的请求。

Step3: 用户使用该token向Nova控件中的api发送要启动一个instance.

Step4: nova-api使用该token向keystone验证用户发送的token是否有效，用户是否有权限区启动这个instance

Step5: keystone将验证信息返回给nova-api, 告诉他用户是否有效。

Step6-7: 如果有效，nova-api在Nova DB中创建instance条目生成instance id存在nova DB中。

Step8: nova-api向Queue发送创建instance消息

Step9: nova-scheduler从queue中取出消息

Step10: nova-scheduler和Nova DB进行沟通，从DB中选择适合的host. [选择合适的DB关乎选择算法，其中关系到很多因素。例如有许多备用的host, 根据用户自定义策略，可以优先选择某一些host, 可以根据负荷等因素最终选出最合适的host]

Step11: nova DB返回合适的host id.

Step12: nova-scheduler向Queue发出启动instance, launching instance.

Step13: nova-compute 取出launching instance request.

Step14,15,16,17,18: nova-computer 需要instance的id, instance会将在哪个host生成的信息才能启动instance. 但是nova-computer不会直接和nova DB进行交流。他会向nova-conductor询问，从nova-conductor作为proxy去查并且告诉nova-computer这些信息。

Step19-21: nova-computer向Glance组件找寻instance的image.

Step22-24: nova-computer向Neutron获取IP网络资源。

Step25-27: nova-computer向cinder获取volumn硬盘资源。

Step28: 最终创建。

创建过程中Step以及对应的状态：

|  |  |  |  |
| --- | --- | --- | --- |
| Status | Task | Power State | Steps |
| Build | Scheduling | None | 3-12 |
| Build | Networking | None | 22-24 |
| Build | Block\_device\_mapping | None | 25-27 |
| Build | Spawing | None | 28 |
| Active | None | Running |  |

10.

Openstack CLI command （统一的Openstack命令，可以查看image, 项目等）:

--help： 当使用CLI command, --help可以打印出下一步所有可用选项以及解释

列举stack的相关资源：openstack stack resource list <stack\_id> --fit-width

列举用户：openstack user list --domain Users

列举角色：openstack role list

列举项目：openstack project list

列举域：openstack domain list

列举image列表清单: openstack image list

列举VM: openstack server list

列举所有compute节点：

列举某个image的详细信息:openstack image show <image-id|image-name> --fit-width

下载image: glance image-download --file [image名] [image id]

上传image: glance image-create --name [上传以后image名字] --visibility public --container-format bare -- --owner [owner id] --disk-format [本地image的格式qcows2/raw] --file [本地image所在路径]

列举用户所被赋予的role: openstack role assignment list --user USER\_NAME --project PROJECT\_ID –names

列举在某个compute host上的全部VM：openstack server list --all-projects --host computeName

将某个用户加入到某个项目当中作为某种角色：

openstack role add --user <user\_id> --project <project\_id> <role\_id>

另外还有模块层面的CLI， 一般推荐统一使用Openstack CLI:

例如image 命令行管理口令：

glance image-list #展示所有image

glance image-show <image-id> #展示特定image 的详细信息

qemu-img info <filename> #展示image的格式 (qcow2, raw) 信息

qemu-img convert -f 原格式· -O 目标格式 源文件 生成文件

11. OpenStack权限概念

User：用户。每一个Openstack使用者都有一个用户，用户必须指定在一个Domain域当中。一个用户可以被安排成一个或者多个角色(role), 一个或者多个项目（projects）

Role: 角色。角色代表一套权限，当这个角色赋予给一个用户的时候，这个用户便拥有这个角色所规定的操作。例如当一个用户被赋予了admin role，那么他就是管理员。

Project: 项目。项目包含一些可供访问的资源以及一些用户。项目内的资源只有属于这个项目的用户们才可以访问。

Domain: 域。域包含了一些项目以及用户，角色。域为项目，用户，角色等设置了一个最大管理边界。

🡪一个Domain中的用户名， 项目名不能重复，不同的Domain中用户名， 项目名可以重复。相当于不同的Domain有自己的一台user, project, role.

🡪一个Domain的用户可以加入另一个Domain的项目当中。