OpenStack基础原理

1. 什么是云计算
2. 可度量服务

---云提供商提供控制和监控资源

1. 资源池

---集中化的设备

---按需分配/重新分配资源

1. 按需自助服务

---无需人工干预

---无处不在的网络接入

---从任何UE接入

1. 快速弹性

---动态的业务性能弹性

1. 云计算的服务模型

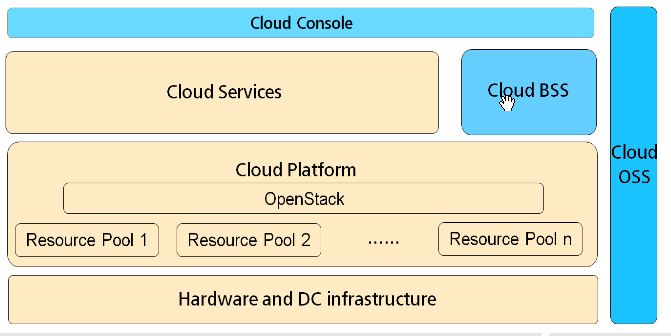
OpenStack的定位

1. 尽管新组件更多的是面向业务的，OpenStack还是可以提供构建网络的基础设施和运行通用虚拟机
2. OpenStack支持包括公有云，私有云，混合云的部署方式；
3. OpenStack不是虚拟化，云计算指（IT能力服务化；按需使用，按量计费；多租户隔离） 虚拟化指（环境隔离，资源复用；降低隔离损耗，提升运行效率；提供高级虚拟化特性）

虚拟化是实现云计算的技术支撑手段之一，但并非云计算的核心关注点

1. OpenStack不是云

为了构建一个云，我们还需要很多东西



1. OpenStack是**目前最主流的开源云操作系统内核**

负责---**资源抽象；资源分配与负载调度；应用生命周期管理；系统运维；人机交互**

内核与完整操作系统的关系：Linux kernel和Ubuntu的关系

1. 三大组件（计算；网络；存储）

共享服务（AAA;数据库；测量；业务发放）

主要用户接口（Dashboard,Horizon）

1. OpenStack的特点
2. 开放

---开源，并尽最大可能重用已有的开源项目

---不要“重复发明轮子”，而要“站在巨人肩膀上”

1. 灵活

---不使用任何不可替代的私有/商业组件

---大量使用插件化方式进行架构设计与实现

1. 可扩展

---由多个相互独立的项目组成

---每个项目包含多个独立服务组件

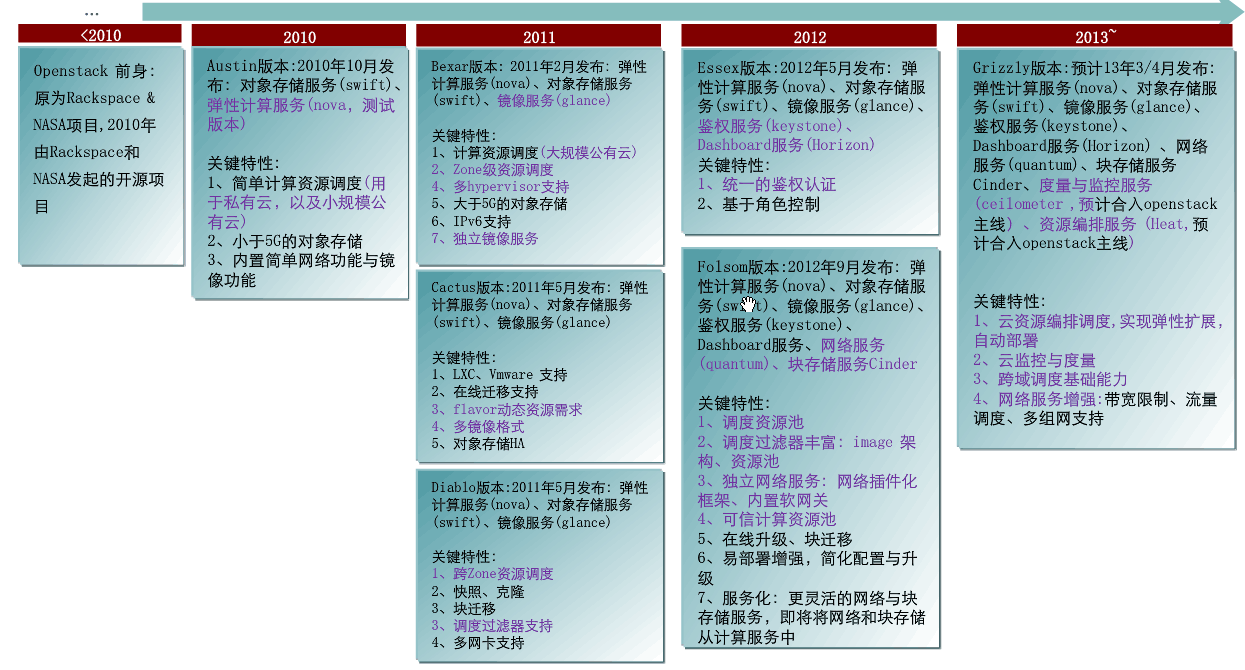
---无中心架构

---无状态架构

1. 约70%的代码使用Python开发；Apache 2.0 License,商业友好

Apache Licence是著名的非盈利开源组织Apache采用的协议。该协议和BSD类似，同样鼓励代码共享和尊重原作者的著作权，同样允许代码修改，再发布（作为开源或商业软件）。

OpenStack基本上每半年发布一个版本 4月和10月



1. OpenStack组织

OpenStack基金会----最高层级的组织，定义“游戏规则”

技术委员会（Technical Committee）----制定OpenStackde 愿景和基本准则（开放性、透明运作、质量）

技术决策的最后决策者

决定正式的项目及团队

执行委员会(Board of Directors)---监控战略和财务

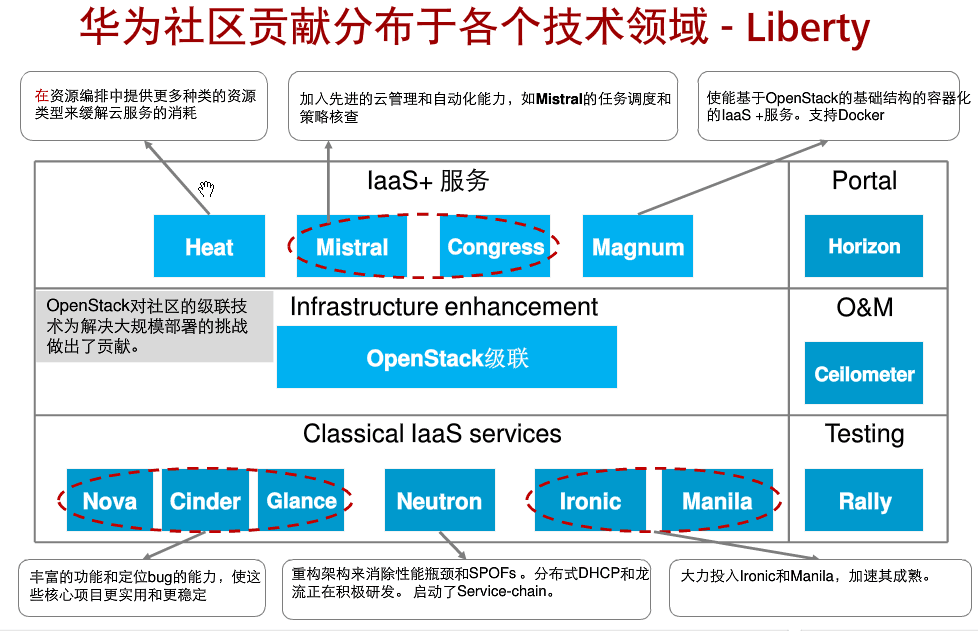
由白金赞助商和黄金赞助商来选定

用户委员会（User Committee）----落户用户的需求

提供指导

跟踪OpenStack的部署

与世界各地的用户群组合作协同



OpenStack起源 存储代码Swift 计算代码 Nova

OpenStack项目发展概况：Nova—计算服务；Swift—对象存储服务；

Glance组件—镜像服务；Neutron—网络服务；

Keystone—身份认证服务；Celimeter—计量服务；

Cinder—块存储服务；Heat编排服务；

OpenStack 架构（综合的云计算管理平台）

Keystone—身份认证服务组件主要负责：身份服务；管理用户，租户，角色，服务和服务端点；可以支持SQL,PAM,LDAP作为后端认证；-------各个组件间认证的核心

Nova—计算服务组件：虚拟机实例的分配以及实例的创建迁移重启，管理云中实例的生命周期，整个OpenStack中的组织控制器。运行的虚拟实例hypervisor支持的有KVM（默认）

Glance组件—镜像服务：镜像格式-raw,qcow,vhd,vmdk,iso

后端存储：Swift,Filesystem,AmazonS3

Swift—对象存储服务:提供存取数据的易用服务，通常用于保存非结构化的数据

Neutron—网络服务：提供支持/实现SDN框架，基于插件的模型

Cinder—块存储服务：为虚拟机实例提供卷的存储化服务，持久化磁盘，基于插件的架构便于扩展；

Horizon组件—外部统一化管理：仪表板；自服务界面；基于云管理功能

Celimeter—监控和计量服务：实现对各个组件信息的统一化处理，平台组件的监控

Nova简介：OpenStack云中的计算组织控制器

管理OpenStack云中实例的生命周期

管理计算资源、网络、认证所需的可扩展性平台

常用术语：KVM---内核虚拟化，OpenStack默认的Hypersvisor层,基于内核的虚拟化技术

Qemu---KVM的替补角色，没有KVM执行效率高，不支持全虚拟化

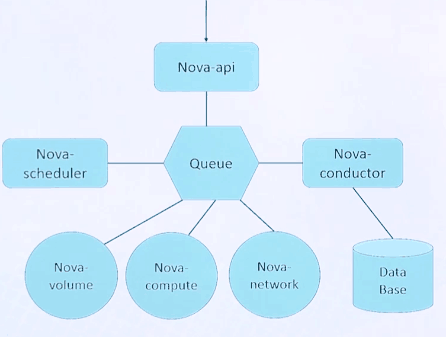
Flavor---新建虚拟机的配置列表，虚拟机模板

Keypair---ssh连接访问实例的密钥对ssh是Secure Shell 的缩写，主要提供远程登录协议

安全组---用来控制实例访问策略的容器

安全组规则---用来控制实例访问的具体策略

Nova框架



当用户通过CLI命令行或者Horizon向Nova组件提出创建实例的请求时，Nova-api作为Nova的入口接收用户请求，然后以消息队列的方式发送到Nova-scheduler, Nova-scheduler去数据库中查询当前节点的负载和使用情况，也要借助于消息队列的形式通过Nova-conductor进而与数据库交互；Nova-scheduler实现实例的调度分配工作

Nova-compute实例的创建

Nova功能特性

1. 实例的生命周期管理
2. 管理平台的计算资源
3. 统一风格的RestAPI
4. 支持透明的hypervisor
5. 各个模块通过消息队列实现交互

Swift—对象存储服务

提供高可用分布式对象存储，为Nova组件提供虚拟机镜像存储的服务

牺牲一定的数据一致性来达到高可用和可伸缩性。适用于互联网应用场景下非结构化的数据存储

常用术语：Account:用户定义的管理存储区域

Container:存储隔间，类似于文件夹或目录

Object:包含了基本的存储实体和他自身的元数据

Ring：记录了磁盘上存储的实体名称和物理位置的映射关系

Swift组件功能：Swift在物理结构上往往会存储对象的多个副本，通常按照物理位置的特点，将对象拷贝到不同的物理位置上，来保证数据的可靠性。

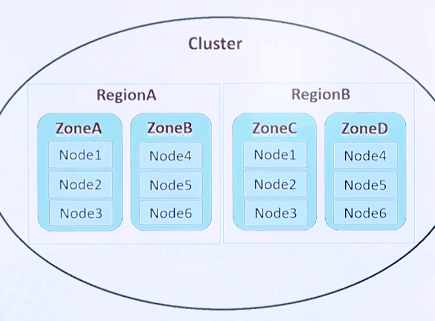
常用术语：Region:地域，从地理位置上划分一个概念

Zone:可用区，按照独立的供网、供电基础设施划分

Node：节点，存储服务器；

Disk:磁盘，物理服务器上的存储设备

Cluster:群集，为冗余考虑的部署架构。



Swift架构：首先用户提出一个对象处理申请，由Swift的API接收处理，收到以后先去找Keystone认证节点对用户的身份进行认证，认证通过后将请求提交给Swift Proxy确定究竟应该将存储对象放在哪一个满足存储要求的存储节点上。最终将对象存储到指定的存储节点上即可。最终将返回结果返回给用户。

Keystone（1）提供身份验证、服务规则和服务令牌功能

* 1. 任何服务之间相互调用，都需要经过Keystone的身份验证

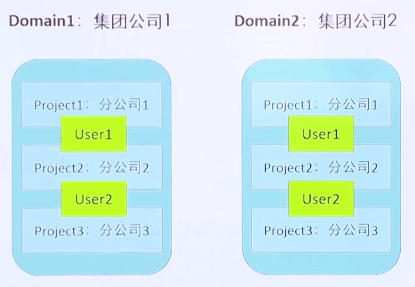
常用语：User---OpenStack最基本的用户

Project----指分配给使用者的资源的集合

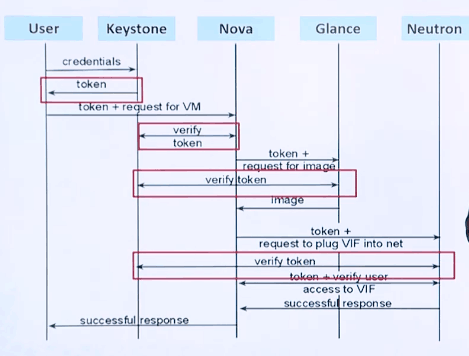
Role---代表一组用户可以访问资源的权限

Domain---定义管理边界，可以包含多个Project/tenant\user\role等

Endpoint---服务的URL路径，暴露出来的访问点



Keystone认证原理



Neutron:负责提供网络服务的核心组件

基于软件定义网络的思想

支持第三方插件

常用术语：Bridge-int---实现内部网络功能的网桥

Br-ex---跟外部网络通信的网桥

Neutron-sever---提供API接口

Neutron-L2-agent---实现二层网络通信的代理，管理VLAN的插件，接收Neutron-sever的指令来创建VLAN

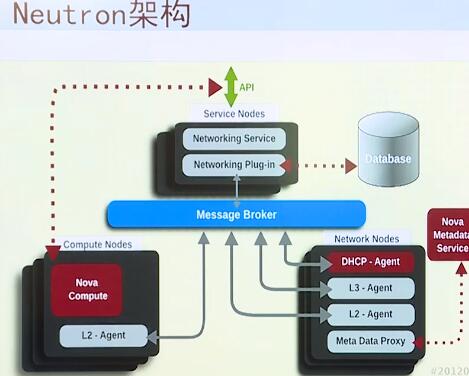
Neutron组件：

Neutron-DHCP-agent---为子网自动分发IP地址

Neutron-l3-agent---租户网络和floating IP间地址转换 通过Linux iPtable中的NAT功能实现

Neutron-metadata-agent---响应NOVA的metadata的请求

Lbaas agent---为多台实例和open vswitch agent提供负载均衡服务



DHCP-Agent实现子网的创建和IP地址的自动分发；

L2-Agent实现相同VLAn下网络的通信；

L3-Agent实现同一个租户网络下，不同子网间的通信

Glance

1. 为Nova提供镜像服务
2. 通常不负责镜像的本地存储
3. 实现对镜像的管理

Glance的作用：为nova提供镜像服务以便启动镜像的组件但不负责镜像的本地存储，可以对镜像做快照、备份等管理  
Glance镜像支持的格式：Raw、vhd、vdi、iso、qcow2、aki ami

Glance 组件：Glance-api---负责提供镜像服务的rest api服务

Glance-registry---主要负责与Glance使用的数据库交互

Glance的架构：当有来自Horizon、Glance CLI、Nova computer发送过来的镜像请求时，由Glance-api进行处理，传递给Glance-registry组件，然后到数据库中查询镜像存储的位置信息，将查询到的结果返回给Glance-api。Glance-api接下来将会调用Storage adapter组件进行查询，用来查询后端的存储，比如SWIFT、Ceph、Amazon S3等，最终获取镜像返回给用户

Cinder

作用：为虚拟机实例提供volume卷的块存储服务；

一个volume卷可以同时挂载到多个实例上；

共享的卷同时只能被一个实例进行写操作；

支持的文件系统类型：

LVM/ISCSI-- iSCSI技术是一种新储存技术，该技术是将现有[SCSI接口](http://baike.baidu.com/view/26111.htm)与以太网络(Ethernet)技术结合，使[服务器](http://baike.baidu.com/view/899.htm)可与使用IP网络的储存装置互相交换资料。

NFS-- [网络文件系统](http://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E6%96%87%E4%BB%B6%E7%B3%BB%E7%BB%9F)，是FreeBSD支持的文件系统中的一种，它允许网络中的计算机之间通过TCP/IP网络共享资源

NetAPP NFS

Gluster: 分布式文件系统

DELL Equal Logic

常用术语：Volume备份—volume卷的备份

Volume快照—卷在某个时刻的状态

Cinder API—为Cinder请求提供统一风格的Rest API服务

Cinder Scheduler—主要负责调度为新建卷指定块存储设备

Cinder Volume—负责与存储的块设备交互，实现卷的创建、删除、修改等操作

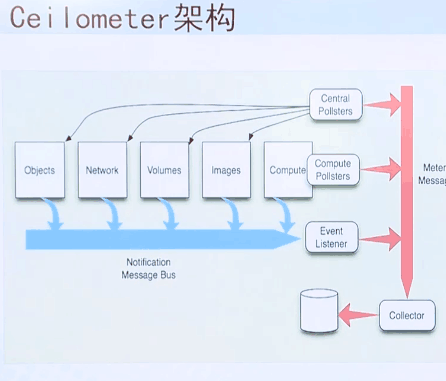
Cinder Backup—备份服务 负责通过驱动和后端的备份设备打交道

Cinder架构：

当有用户或者Nova compute 提出创建卷的请求时首先由Cinder API接收请求，然后以消息队列的方式发送给Cinder schedual来调用，Cinder schedual 监听到消息队列后，到数据库中查询当前存储节点的预定信息，并根据预定策略选择卷的最佳Volume service 节点，然后给Volume service,再去查找Volume Providers，在特点的存储节点上创建相关的卷

Ceilometer

1.Ceilometer的简介：  
Ceilometer是open stack 中的一个子项目，为计费、监控等其他的服务提供数据支撑。  
2. Ceilometer存在的理由：  
更多的公司利用open stack做自己的公有云平台。而作为共有云，计量和监控，这两个基础的服务往往是必不可少的，计量是为了获取平台中用户对自己的使用情况，监控是为了确保资源处于一个健康的状态，因此，Ceilometer在项目提出之处，是为了计量、计费而生。  
3. Ceilometer的核心概念  
Ceilometer-agent-compute:运行在计算节点上，是收集计算节点上信息的代理  
Ceilometer-agent-central：运行在控制节点上，轮询服务的非持续化数据  
Ceilometer-collector:运行在一个或者多个控制节点上，监听Message Bus【消息总线】，将收到的信息写入到数据库中  
Storage:数据存储，支持mongo DB，mysql等等。用于存储收集到的样本数据  
API server:运行在控制节点上，提供对数据库的数据的访问  
Message Bus:计量数据的消息总线，收集数据给Ceilometer-collector  
4. Ceilometer架构  
Ceilometer采用了两种数据采集的方式，其中一种是消费了open stack内各个服务自动发出的notification消息，【图中的蓝色箭头】，另外一种是调用各个服务的API，去主动轮询获取数据。【图中的黑色箭头】  
   
为什么采用两种数据采集的方式？【也是工作架构】  
因为在open stack 中，大部分事件都会发出notification消息，比如创建删除instance实例的时候，这些计量计费的信息时，都会发出notification消息。而作为Ceilometer组件，就是notification消息的最大的消费者。因此，第一种方式，是Ceilometer的首要的数据来源。  
但是，也有一些计量的消息，是notification获取不到的，比如一些instance的CPU的运行时间，或者是CPU的使用率等等。因此，Ceilometer增加了第二种方式，即为周期性的调用相关的API，去轮询这些消息。



Heat简介

1. OpenStack核心项目之一，提供基于模板的编排服务。支持自由的hot模板以及亚马逊的clody fommition模板，支持丰富的模板类型，不仅覆盖了常用的基础架构（包括网络 计算存储等），还覆盖了celometer的警报
2. 常用术语

Stack—-Heat要用到的所有设施和资源的集合

Heat template---是以.yaml结尾的文件，用于创建stack

Heat-api---提供rest api服务，将api请求发送给heat engine去执行

Heat-api-cfn---支持亚马逊格式的Rest api

Heat-engine---Heat的核心模板，接收API请求在openstack中创建资源

1. Heat 组件

Heat-cfntools\Heat-init:

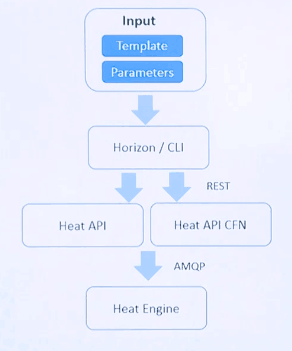
在镜像中安装完成虚拟实例操作任务的工具

Heat-api-cloudwatch:监控编排服务

Resource:底层各种服务抽象的集合

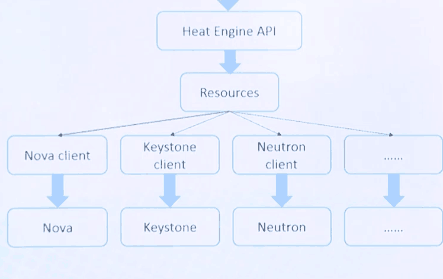
Heat-client:调用访问其他各个组件的client工具

1. Heat架构



当用户在Horizon/命令行中输入模板和参数的创建参数实例的请求时，Heat服务接收请求调用HeatAPI/Heat API CFN，首先验证模板的正确性，验证通过后通过消息队列交给Heat Engine处理。

Heat Engine获取请求后，会把请求解析为各种资源类型，每种资源都有相应的client对应，每个client都会发送相应的请求给其他的服务，获取相关的资源，最终完成请求的处理



整个过程中HeatEngine的处理分为三层:第一层---处理Heat层面的请求，根据模板和输入的参数创建Stack；

第二层---解析Stack里的各种依赖关系，以及Stack的嵌套关系；

第三层---根据解析出来的关系依次调用各种服务的client创建各种资源。