OpenStack

学习手册

OpenStack 是一种免费的开源平台，帮助服务提供商实现类似于亚马逊 EC2 和 S3 的基础设施服务。OpenStack 当前有三个核心项目：计算(Nova)，对象存储(Swift)，镜像管理(Glance)。每个项目可以独立安装运行，该文档将帮助您快速学习 OpenStack。

皮丽华

2012/5/3

目录

OpenStack 背景现状

OpenStack 是由 Rackspace Cloud 和 NASA（美国航天局）于 2010 年 7 月开始共同开发支持，整合了 Rackspace 的 Cloud Files platform 和 NASA 的 Nebula platform 技术，目的是能为任何一个组织创建和提供云计算服务。

目前，超过 150 家公司参与了这个项目，包括 Crtrix Systems, Dell, AMD, Intel, Cisco, HP 等。OpenStack 最近发布了 Austin 产品，它是第一个开源的云计算平台，它是基于 Rackspace 的云服务器加上云服务，以及 NASA 的 Nebula 技术发布的。 似乎是作为对此的响应，Amazon 为新用户提供一年的 AWS 免费使用方式。在 OpenStack 发布 Austin 之后，微软也宣称 Windows Server 2008 R2 Hyper-V 可以与 OpenStack 整合。 微软会为 Cloud.com 提供架构和技术上的指引，它会编写必要的代码，从而 OpenStack 能够在微软的虚拟平台上运行。 这些代码会在 OpenStack.org 上提供。

OpenStack 是什么？

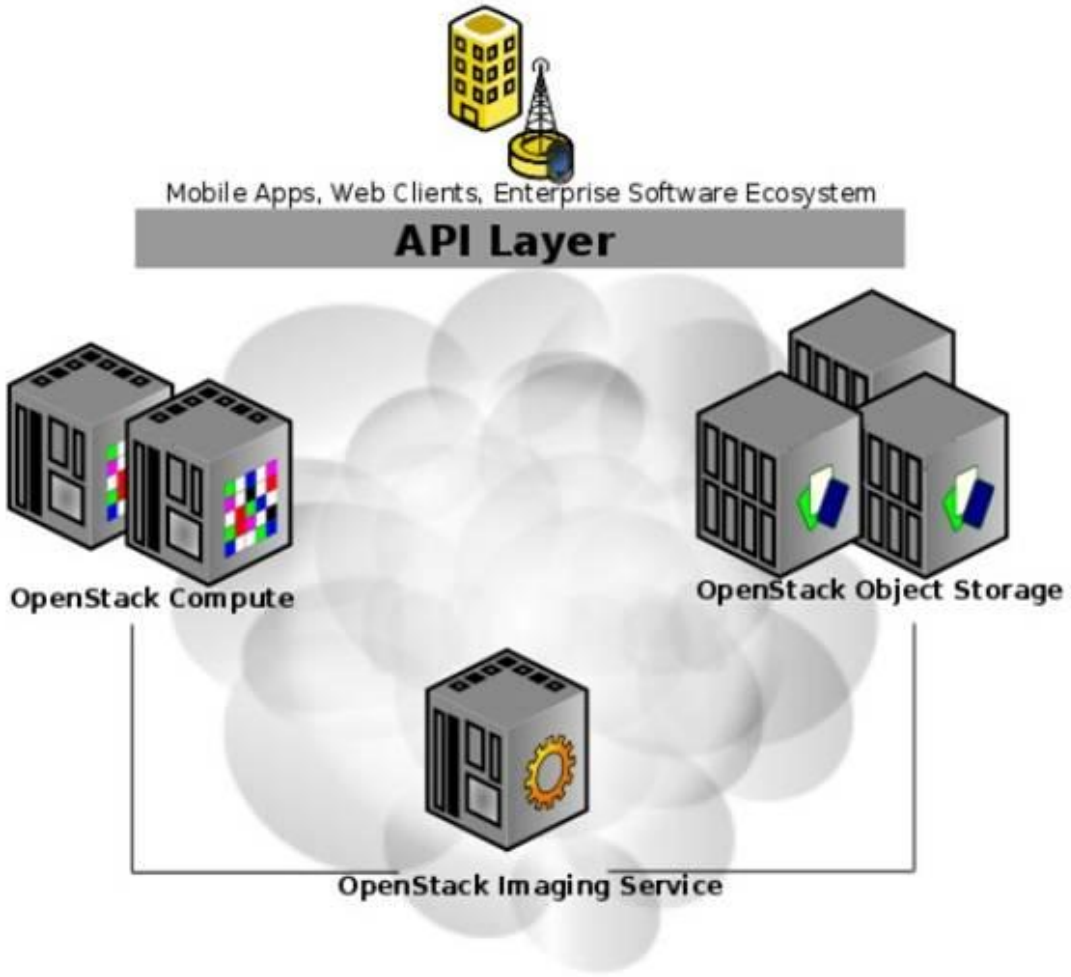
OpenStack 核心项目

OpenStack 是一种免费的开源平台，帮助服务提供商实现类似于亚马逊 EC2 和 S3 的基础设施服务。OpenStack 当前有三个核心项目：计算(Nova)，对象存储(Swift)，镜像管理(Glance)。每个项目可以独立安装运行。另外还有两个新增项目：身份验证(Keystone)和仪表盘(Horizon)。

OpenStack 计算是一个云控制器，用来启动一个用户或一个组的虚拟实例，它也用于配置每个实例或项目中包含多个实例为某个特定项目的联网。

OpenStack 对象存储是一个在具有内置冗余和容错的大容量系统中存储对象的系统。对象存储有各种应用，如备份或存档数据，存储图形或视频（流媒体数据传输到用户的浏览器），储存二级或三级静态数据，发展与数据存储集成新的应用程序，当预测存储容量困难时存储数据，创造弹性和灵活的云存储 Web 应用程序。

OpenStack 镜像服务是一个查找和虚拟机图像检索系统。它可以配置三种方式：使用 OpenStack 对象存储来存储图像;使用亚马逊 S3 直接存储，或使用 S3 对象存储作为 S3 访问中间存储。



OpenStack 版本信息

目前为止共有四个版本：

1. Austin
2. Bexar
3. Cactus
4. Diablo

OpenStack 功能

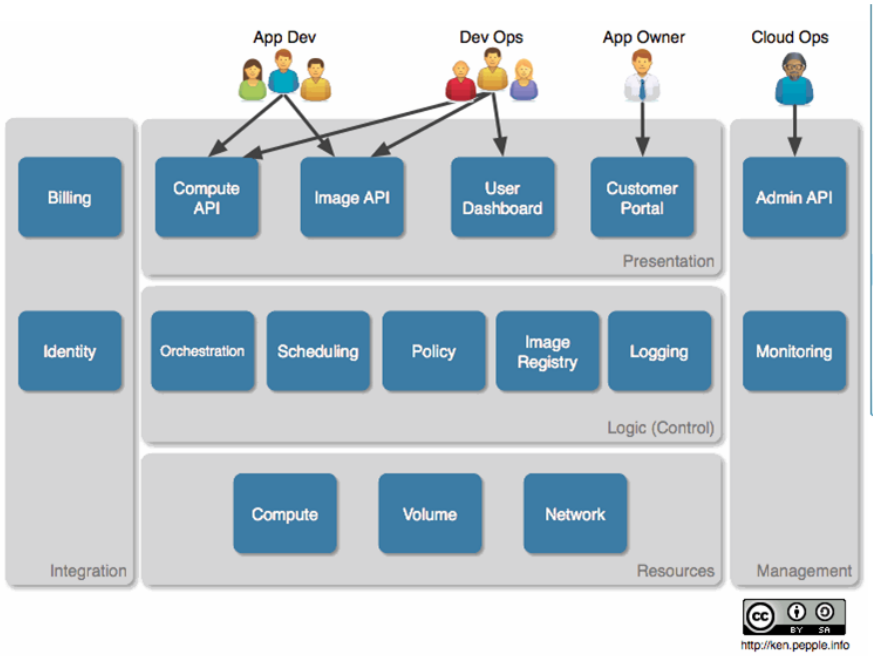
OpenStack 能帮我们建立自己的 IaaS，提供类似 Amazon Web Service 的服务给用户：

1、普通用户可以通过它注册云服务，查看运行和计费情况

2、开发和运维人员可以创建和存储他们应用的自定义镜像，并通过这些镜像启动、监控和

终止实例

3、平台的管理人员能够配置和操作网络，存储等基础架构

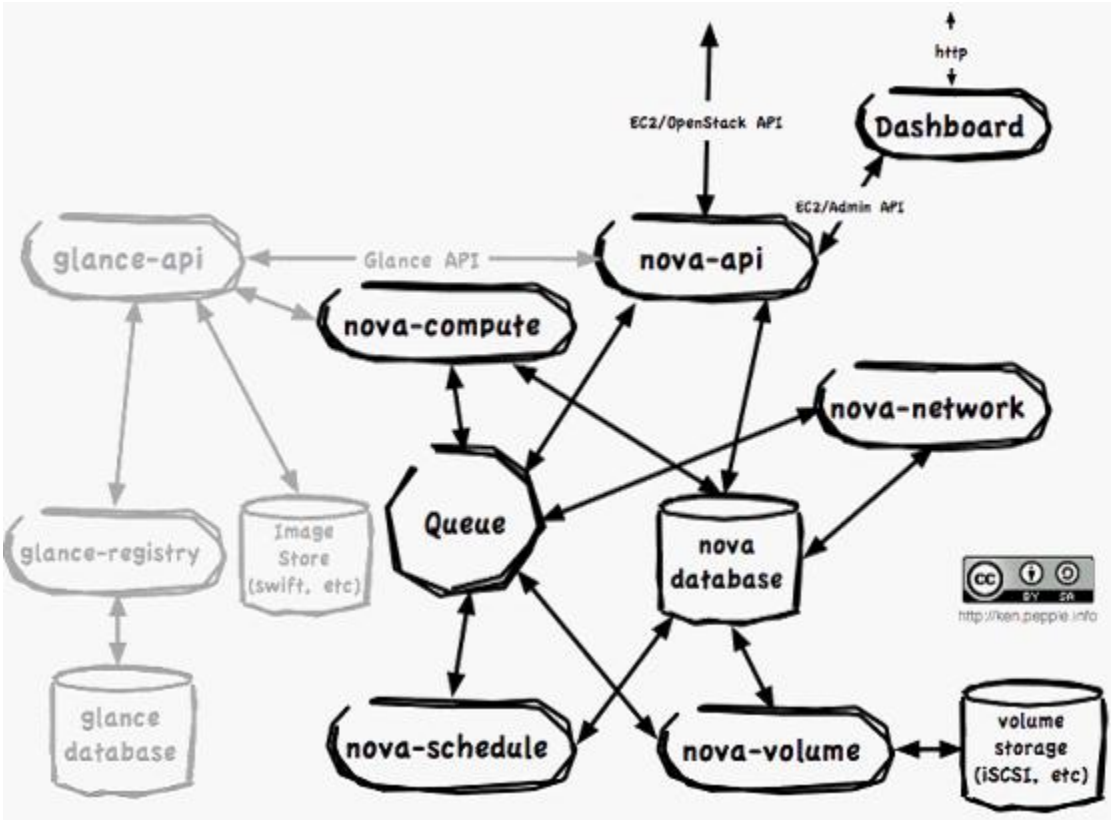


OpenStack 的优势是平台分模块化，由每个独立的组件组成，每个 nova 组件都可以单独安装在独立的服务器上，各个组件之间不共享状态，各个组件之间通过消息队列(MQ)来进行异步通讯。也可以通过选用合适组件来定制个性化服务，便于应用改进。使用 apache 协议可以支持企业使用。

OpenStack 架构

OpenStack 项目架构一: Compute(Nova)的软件架构

下图是 Nova 的软件架构，每个 nova-xxx 组件是由 python 代码编写的守护进程，每个进程之间通过队列（Queue）和数据库（nova database）来交换信息，执行各种请求。而用户通过 nova-api 暴露的 web service 来同其他组件进行交互。Glance 是相对独立的基础架构，nova 通过 glance-api 来和它交互。



Nova 组件的作用

nova-api 是 Nova 的中心。它为所有外部调用提供服务，除了提供 OpenStack 本身的 API 规范外，他还提供了兼容 EC2 的部分 API，所以也可以用 EC2 的管理工具对 nova 进行日常管理。

nova-compute 负责对虚拟机实例进行创建、终止、迁移、Resize 的操作。工作原理可以简单描述为：从队列中接收请求，通过相关的系统命令执行他们，再更新数据库的状态。

nova-volume 管理映射到虚拟机实例的卷的创建、附加和取消。

nova-network 从队列中接收网络任务，然后执行任务控制虚拟机的网络，比如创建桥接网络或改变 iptables 的规则。

nova-scheduler 提供调度，来决定在哪台资源空闲的机器上启动新的虚拟机实例

Queue 为守护进程传递消息。只要支持 AMQP 协议的任何 Message Queue Sever 都可以，当前官方推荐用 RabbitMQ。

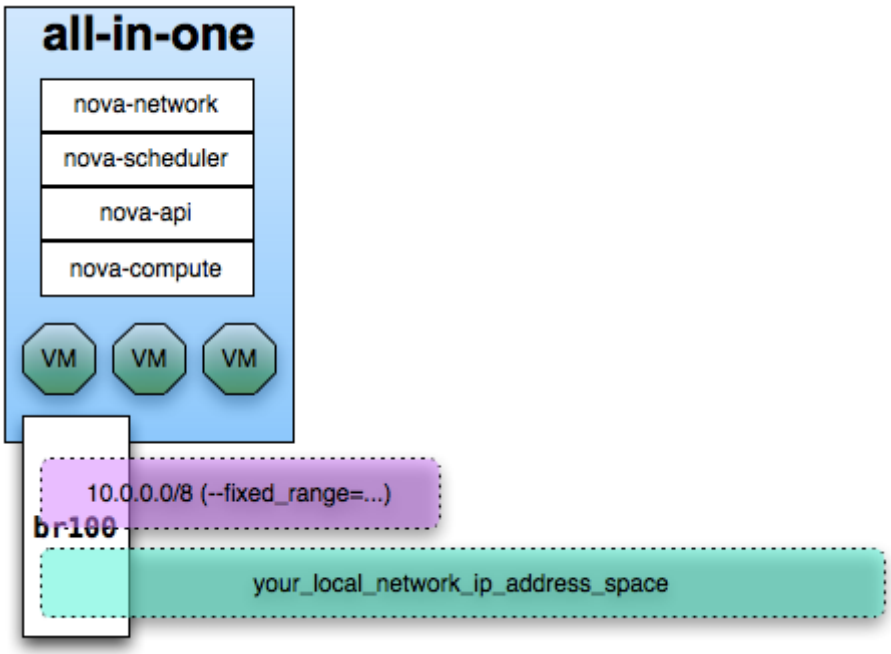
SQL database 存储云基础架构中的各种数据。包括了虚拟机实例数据，网络数据等。

user dashboard 是一个可选的项目。它提供了一个 web 界面来给普通用户或者管理者来管理、配置他们的计算资源。

Nova 的硬件架构

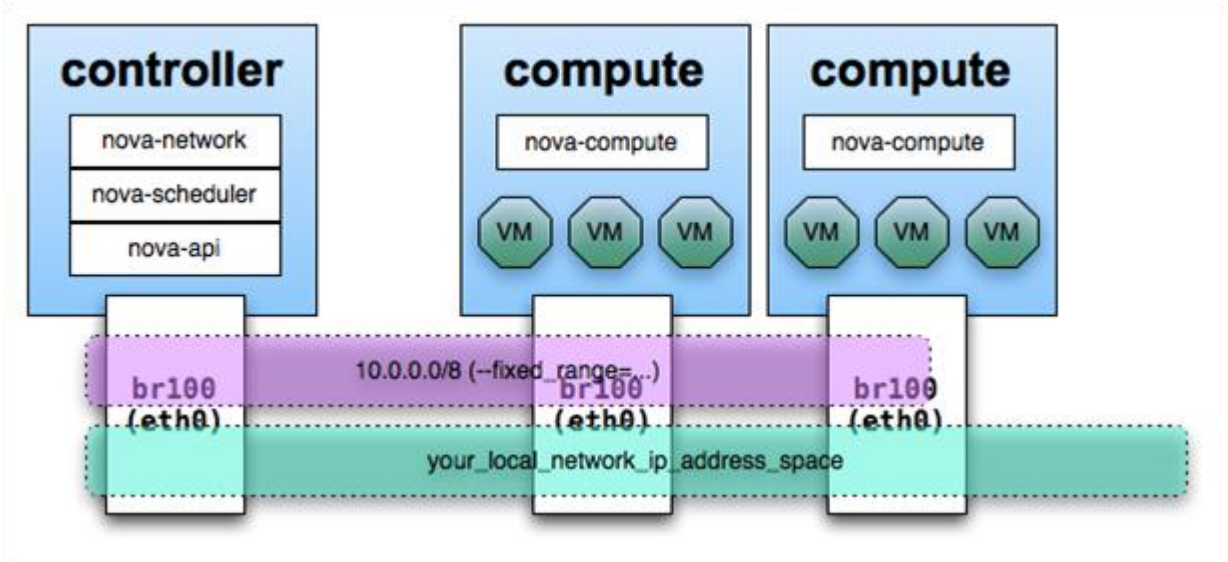
Nova 采用无共享、基于消息的架构，我们能安装每个 nova-xxx 组件在单独的服务器上，这样可以根据不同目的进行不同的配置安装

单结点：一台服务器运行所有的 nova-xxx 组件，同时也驱动虚拟实例。这种配置只为尝试 Nova，或者为了开发目的进行安装。
单结点：一台服务器运行所有的 nova-xxx 组件，同时也驱动虚拟实例。这种配置只为尝试 Nova，或者为了开发目的进行安装。

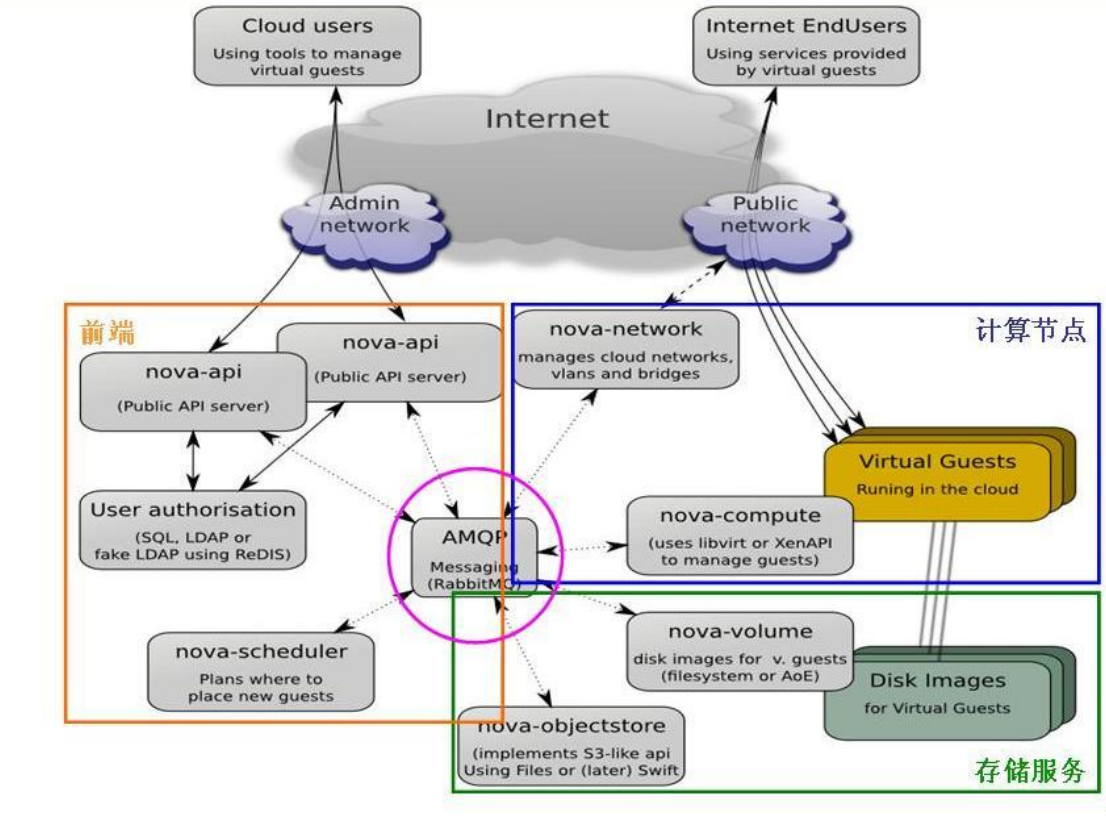


1 控制节点+N 个计算节点：一个控制结点运行除 nova-compute 外的所有 nova-services，然

后其他 compute 结点运行 nova-compute。所有的计算节点需要和控制节点进行镜像交互，网络交互，控制节点是整个架构的瓶颈，这种配置主要用于概念证明或实验环境。



多节点：增加节点单独运行 nova-volume，同时在计算节点上运行 nova-network，并且根据不同的网络硬件架构选择 DHCP 或者 VLan 模式，让控制网络和公共网络的流量分离。



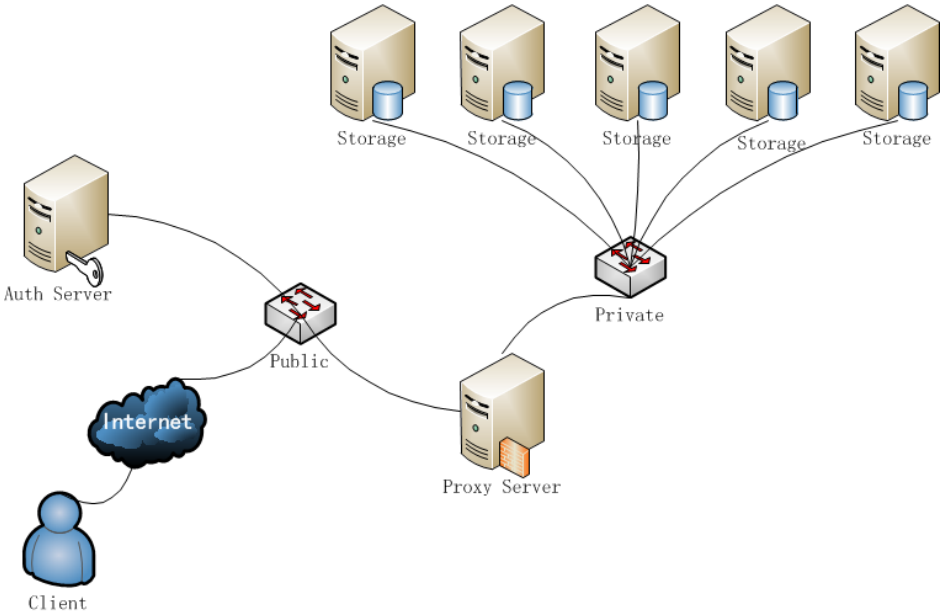
Nova 功能介绍

用户通过访问 horizon(dashboard)请求资源，horizon 会调用 nova-api。OpenStack 首先对用户进行身份认证，这个功能通过 keystone 模块来完成。然后通过任务调度器(nova-scheduler)确定在哪一个计算节点上创建新的虚拟机。所有的任务都会通过 MQ 来进行异步通讯。

云管理员用户也可以通过 Euca2ools 来管理和创建虚拟机，因为 OpenStack 支持 EC2 和 S3 接口。

OpenStack 项目架构二: Swift 架构

OpenStack Object Storage（Swift）是 OpenStack 开源云计算项目的子项目之一。前身是 Rackspace Cloud Files 项目。OpenStack 对象存储是一个在具有内置冗余和容错的大容量系统中存储对象的系统。对象存储有各种应用，如备份或存档数据，存储图形或视频，储存二级或三级静态数据，发展与数据存储集成新的应用程序，当预测存储容量困难时存储数据，创造弹性和灵活的云存储 Web 应用程序。



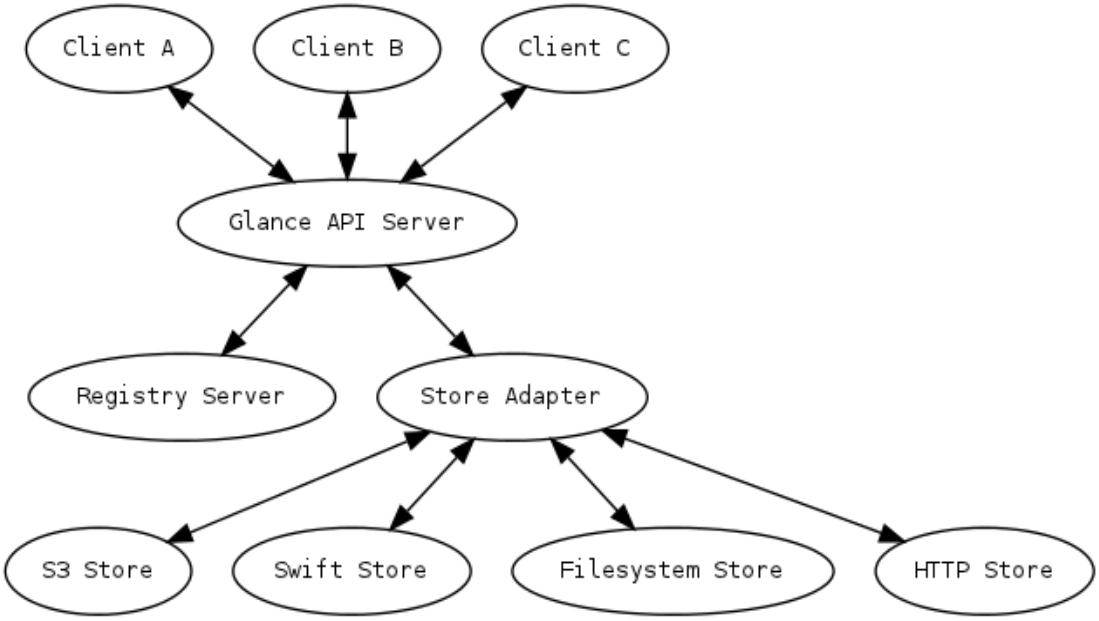
Swift 功能

Swift 使用普通的服务器来构建冗余的、可扩展的分布式对象存储集群，存储容量可达 PB 级。Swift 提供的服务与 AWS S3 相同，可以用以下用途：

1. 作为 IaaS 的存储服务
2. 与 OpenStack Compute 对接，为其存储镜像
3. 文档存储
4. 存储需要长期保存的数据，例如 log
5. 存储网站的图片，缩略图等

OpenStack 项目架构三 – Glance 架构

OpenStack 镜像服务提供 OpenStack Nova 虚拟机镜像的发现，注册，取得服务。通过 Glance，虚拟机镜像可以被存储到多种存储上，比如简单的文件存储或者对象存储（比如 OpenStack 中 swift 项目）。



Glace 组件架构

- Glance 目前提供的参考实现中 Registry Server 仅是使用 Sql 数据库存储 metadata。
- 前端通过 API Server 向多个 Client 提供服务。
- 可以使用多种后端存储。Glance 目前支持 S3，Swift，简单的文件存储及只读的 HTTPS 存储。
- 后续也可能支持其他后端，如分布式存储系统（SheepDog 或 Ceph）

Glace 组件架构特性

1.基于组件的架构 ：便于快速增加新特性

2.高可用性：支持大负荷
3.容错性：独立的进程避免串行错误
4.开放标准： 对社区驱动的 API 提供参考实现

OpenStack 功能

1、 Dashboard 提供资源池管理功能，通过资源池的方式对物理资源进行重新组织。
2、 提供基于命令行的虚拟机在线迁移功能，拟机生命周期管理，例如创建、启动、休眠、唤醒、关闭、迁移、销毁虚拟机。
3、 将常用的运行环境保存为虚拟机模板，可以方便地创建一系列相同或者是相似的运行环境，只能手动创建所需用户模板，类似 Eucalyptus。
4、 在计算资源允许的情况下提供高可用性、动态负载均衡、备份与恢复
5、 对所有的物理机和虚拟机进行监控，生成报表并在必要的情况下发出预警，监控和报表功能据说可以采用外围组件实现。

Openstack 创建 instance 的流程

1. 用户向 nova-api 发送请求

用户发送请求到 nova-api,这里有两种：

a.通过 openstack api

从 server.py's controller.create():

Python 代码

```
1. self.helper.create_instance(req, body, self.compute_api.create)
```

create_instance_helper.CreateInstanceHelper() 查表获取基本信息

b.通过 ec2 api

从 cloud.py.run_instances()

统一调 computer.api.create() 将新的数据插回去

Python 代码

```
1. self._ask_scheduler_to_create_instance(context, base_options,
2.                                        instance_type, zone_blob,
```

```
3.                          availability_zone, injected_files,
4.                          admin_password, image,
5.                          instance_id=instance_id,
6.                          requested_networks=requested_networks)
```

2. API 将处理好的数据通过 MQ 转发给 scheduler .(code from Computer.api)

Python 代码

```
1.  rpc.cast(context,
2.           FLAGS.scheduler_topic,
3.           {"method": "run_instance",
4.            "args": {"topic": FLAGS.compute_topic,
5.                     "instance_id": instance_id,
6.                     "request_spec": request_spec,
7.                     "availability_zone": availability_zone,
8.                     "admin_password": admin_password,
9.                     "injected_files": injected_files,
10.                    "requested_networks": requested_networks}})
```

3. Scheduler 获取信息并作出决定 哪一个 host 可以来 run instance.

Python 代码

```
1.  def __getattr__(self, key):
2.      return functools.partial(self._schedule, key)
```

Python 代码

```
1.  def _schedule(self, method, context, topic, *args, **kwargs):
2.      .......
3.      rpc.cast(context,
4.               db.queue_get_for(context, topic, host),
5.               {"method": method,
6.                "args": kwargs})
7.      LOG.debug(_("Casted to %(topic)s %(host)s for %(method)s") % locals())
```

4. Computer 从池中获取信息 并让 Networker 去准备一个 ip ,让 volume 准备卷，然后初始化相应的信息,例如创建 image,映射 device,创建 domain, 并将 domain 放入 running pool 中，

然后就进入等待直到 instance 的状态变为 running.

a. networker 分配 ip

Python 代码

```
network_info = self.network_api.allocate_for_instance(context,
                    instance, vpn=is_vpn,
                    requested_networks=requested_networks)
```

Python 代码

```
def allocate_floating_ip(self, context):
  return rpc.call(context,
                FLAGS.network_topic,
                {'method': 'allocate_floating_ip',
                 'args': {'project_id': context.project_id}})
```

b 让 volume 准备卷

Python 代码

```
bd_mapping = self._setup_block_device_mapping(context, instance_id)
def create(self, context, size, snapshot_id, name, description,
          volume_type=None, metadata=None, availability_zone=None):
        rpc.cast(context,
                  FLAGS.scheduler_topic,
                       {"method": "create_volume",
                          "args": {"topic": FLAGS.volume_topic,
                          "volume_id": volume['id'],
                           "snapshot_id": snapshot_id}})
```

c call nova.virt.libvirt.firewall.IptablesFirewallDriver 建立网络规则

d call libvirt 创建 domian 并 launch

Python 代码

```
domain = self._create_new_domain(xml)
def _create_new_domain(self, xml, persistent=True, launch_flags=0):
  if persistent:
          # To create a persistent domain, first define it, then launch it.
          domain = self._conn.defineXML(xml)
```

```
            domain.createWithFlags(launch_flags)
    else:
            # createXML call creates a transient domain
            domain = self._conn.createXML(xml, launch_flags)
    return domain
```

e call virt.libvirt.connetion.spwan 等待

Python 代码

```
def spawn(self, context, instance, network_info,
    block_device_info=None):
     ..........
     def _wait_for_boot():
          instance_name = instance['name']
          try:
               state = self.get_info(instance_name)['state']
          except exception.NotFound:
               msg = _("During reboot, %s disappeared.") % instance_name
               LOG.error(msg)
          raise utils.LoopingCallDone

          if state == power_state.RUNNING:
             msg = _("Instance %s spawned successfully.") % instance_name
             LOG.info(msg)
             raise utils.LoopingCallDone

timer = utils.LoopingCall(_wait_for_boot)
return timer.start(interval=0.5, now=True)
```

OpenStack 在企业中的应用

更多的企业不只是谈论 OpenStack，而是在实际生产环境中部署它，包括 Rackspace 基于 Puppet 的公有云。OpenStack 自研发伊始，一直被视作云计算领域的 Linux，其推动开放源代码服务的努力得到了众多公司的支持。目前就有超过 100 个机构参与了代码库的建设，或在其它方面参与该项目。新浪云计算将与 OpenStack 一起合力打造一套可以管理和配置各种虚拟化技术的 IaaS 平台，在开源代码库的建设方面将有着不小的贡献。

新浪云计算加入开源云计算项目 OpenStack

新浪云计算宣布正式加入全球开源云计算项目 OpenStack，选择 OpenStack 作为 IaaS 平台解决方案。作为 OpenStack 中国的积极推动者，这在很大程度上将推动 OpenStack 云开源代码项目的发展。

新浪云计算之所以选择 OpenStack 有这样几个理由：

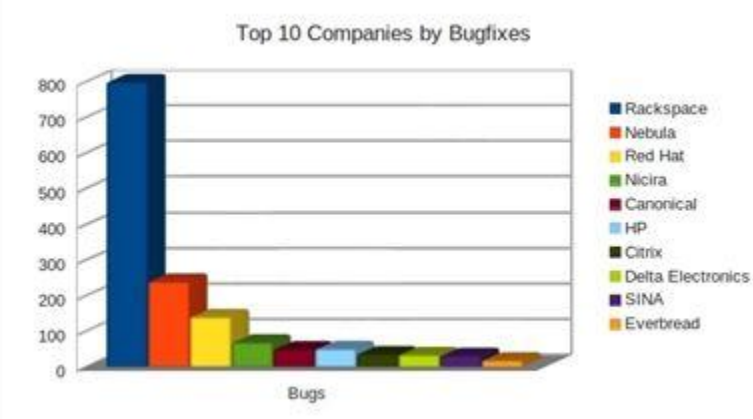
首先，OpenStack 是完全用 Python 编写的唯一开源的 IaaS 项目。与 C/C++或 Java 为基础的项目比较，Python 项目意味着更容易安装，修改，封装和调试。

第二，因为它是开源的，在部署或升级过程中有错误发生时，通过阅读源代码，我们就可以迅速找到原因，并修复它。

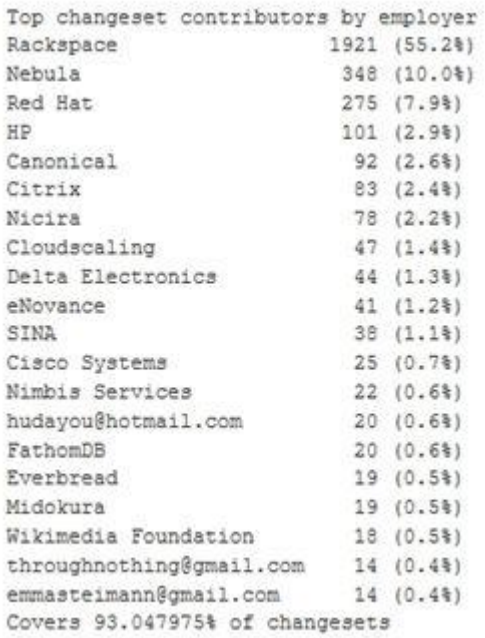
第三，由于有 Rackspace 的参与，OpenStack 实际上是由主机托管/服务提供商行业设计的，因此它非常适用于公有云的大服务器、多租户的工作负载。另外，OpenStack 成熟的功能性、hypervisor 无关性设计及其可扩展特性等，都让我们非常欣赏。

新浪云计算正式加入 OpenStack，成为参与这一全球最大的开源云计算服务研发项目的企业与机构之中领先的 IaaS 平台研发与业务运营的积极实践者，其贡献与意义将不言而喻。

在最近 OpenStack Essex 的贡献者统计数据中，新浪在 bugfix 方面的贡献全球排名第九，在中国公司对 bugfix 贡献的排名中也是首屈一指。



而在 OpenStack 社区的活跃性方面，前三甲分别是其老东家 Rackspace、Nebula，以及后来者 RedHat。从数据中看出，新浪云计算在带动国内广大云技术研发和使用者沟通交流方面，也正积极推动 OpenStack 社区发展。



```
Top changeset contributors by employer
Rackspace                  1921 (55.2%)
Nebula                      348 (10.0%)
Red Hat                     275 (7.9%)
HP                          101 (2.9%)
Canonical                    92 (2.6%)
Citrix                       83 (2.4%)
Nicira                       78 (2.2%)
Cloudscaling                 47 (1.4%)
Delta Electronics            44 (1.3%)
eNovance                     41 (1.2%)
SINA                         38 (1.1%)
Cisco Systems                25 (0.7%)
Nimbis Services              22 (0.6%)
hudayou@hotmail.com          20 (0.6%)
FathomDB                     20 (0.6%)
Everbread                    19 (0.5%)
Midokura                     19 (0.5%)
Wikimedia Foundation         18 (0.5%)
throughnothing@gmail.com     14 (0.4%)
emmasteimann@gmail.com       14 (0.4%)
Covers 93.047975% of changesets
```

OpenStack 社区的活跃性排名

同时，加入 OpenStack，也将为新浪云计算提供更多与全球企业或研发机构交流、合作的机会与渠道，向全球分享中国云计算技术的步伐将更加稳健。这也将极大的鼓舞国内广大云技术研发和使用者，未来以 OpenStack 为代表的开源云端软件在中国将更加普及，国内更多企业将从中受益。

展望未来：OpenStack 发力

IBM 和 Red Hat 可能很快就会加入到支持 OpenStack 的阵营当中，这将很大程度上推动这个云开源代码项目的发展。IBM 对 OpenStack 表明态度，将要像多年支持 Linux 那样的级别去支持 OpenStack。

IBM 一直对开源技术有很强的支持力度，之前 IBM 已经加入 OpenStack Foundation，成为其“白金会员”。IBM 负责软件标准和云的副总裁 Angel Diaz 表示，IBM 对 OpenStack 的支持是不遗余力的。目前来看，OpenStack 也很像 Apache，未来可能取得同样的成功。

Citrix 本周宣布，其 CloudStack 开源软件将加入 Apache 软件基金会。同时，OpenStack 在周四也发布了其软件的新版本 Essex。

OpenStack 与 Puppet Labs 整合引关注, 早期使用者认为 Puppet 的整合会提升 OpenStack 的吸引力。“我无论如何都要部署 OpenStack，因为我知道我可以用它的 API 写一个 Puppet 模块” Joe Julian——一位 Ed Wyse Beauty Supply 公司的资深系统管理员在邮件中这样述说。“这个整合使它更加吸引人因为它会节省我的部署上时间和金钱。”

总的来说，OpenStack 起步比较晚，但是社区活跃度和公司参与度很高，这也使得其技术更新很快，有很大的上升空间。