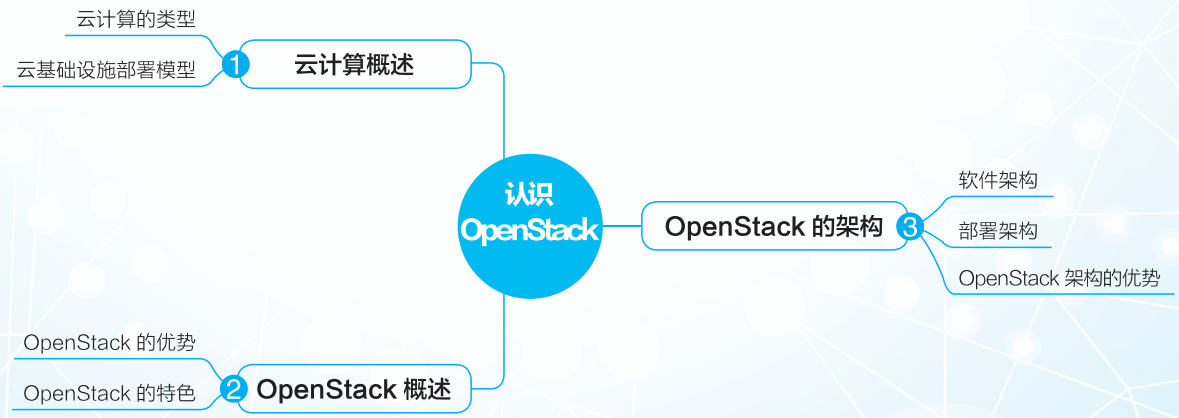
目录

第 1 章 认识 OpenStack

学习目标

1. 了解云计算的类型。
2. 了解云计算的实施部署。
3. 理解云计算与应用开发相关架构。

知识导图



笔记

本章导读

OpenStack 从“作为服务”考虑，将彻底改变软件和应用的部署架构。通过将常见和常规的操作移交给云基础设施，用户可以将时间和想法集中在最重要的事情上——应用的功能。例如，允许上传大文件的传统应用需要为这些文件指定临时和永久的存储路径，并且需要管理存储资源以确保磁盘不会填满。系统管理员或部署人员需要制定一个数据备份策略，或将数据复制到其他的数据中心。但使用正确的云平台后，用户可以简单地将该功能委托给基础设施，并且无须投入特别的努力即可获得所有的优势。

1.1 云计算概述

1.1.1 云计算的类型

关于云计算，美国国家标准与技术研究院（National Institute of Standards and Technology，NIST）提出了以下 5 个关键要素：按需自助服务、广泛的网络访问、池化资源、可伸缩性和可度量的服务。一般情况下，这些要素可以在几种不同的模型中提供。事实上，这些要素可看作栈中的层，每一层在前一层的基础上构建，如图 1-1 所示。

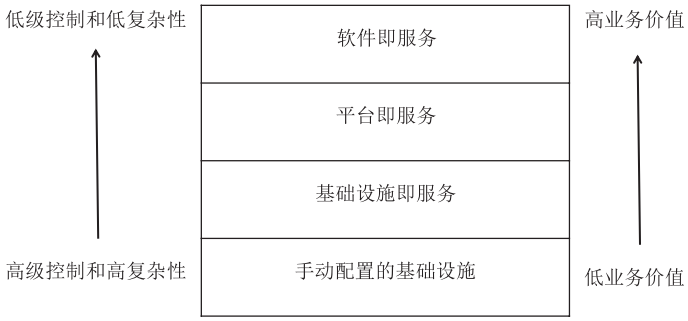


图 1-1　云计算构建

在图 1-1 中，手动配置的基础设施（Manually Provisioned Infrastructure）代表建立信息系统基础设施的传统方法，这种基础设施不是云计算。在这种环境下，物理机器逐个进行上架、连接和配置。这样就提供了完全的控制权，但在建立或需要改变时要求花费大量的时间和精力。当然，所有的云在某个时候都需要运行在物理基础设施上，所以这为其他方面提供了基础。然而，使云计算成功的关键之一是将复杂性从栈中的当前层级移到较高的层级上。

基础设施即服务（Infrastructure as a Service，IaaS）是云计算栈中最基本的层级。这一层是 OpenStack 主要的关注点，也是亚马孙 Web 服务（Amazon Web Services，AWS）的主要关注点。它使得计算、网络和存储能够自动化或自助提供。通常情况下，这些资源作为虚拟机（Virtual Machine，VM）提供，但是也可以用来启动裸机服务器（即物理主机）。这称为“裸机即服务（Mental as a Service）”，并且 OpenStack 提供了管理该服务的项目。或者可以启动容器，而不是虚拟机或裸机服务器。关键是该层能够自动化地提供（可选）连接网络和存储的计算实例。

平台即服务（Platform as a Service，PaaS）建立在 IaaS 之上，能够提供应用而不是可能用来运行应用的基础设施。因此，PaaS 提供了应用所需的核心通用服务，以及配置和

部署应用以使用这些服务的机器。PaaS 通常会提供一个完整的应用栈（Web 服务器、应用服务器、数据库服务器等），在其中可以轻松地部署应用。Heroku（https://www.heroku.com）是一个使用各种标准框架（如 Ruby-on-Rails）构建应用的流行 PaaS 例子。使用 Heroku 可以通过简单的 git push 命令将应用部署到 Internet 上。作为应用的开发者和部署人员，不必担心配置和部署不同的层，甚至不用担心如何扩展它们。如果按照 Heroku 的约定，一切会由 PaaS 处理。

笔记

软件即服务（Software as a Service，SaaS）是离底层物理基础设施最远的层。它可能建立在 IaaS 或 PaaS 之上，但这不是必需的——关键是用户永远不会真正知道这一点。这是从用户的角度来看待云计算最简单的形式，因为他们没有深入了解服务背后实际的机制和体系。它仅仅是用户使用的一个服务。通常，该层以一个网站的形式提供，如 Salesforce.com。但是，用户也可以获取较低级别的服务，如数据库即服务（Database as a Service），其中可以简单地通过含有某些参数的应用程序接口（Application Programming Interface，API）（或网站）获取数据库服务，并且它会提供一个 IP 地址和端口来连接服务。作为服务的用户，不必担心如何扩展服务——但随着所使用服务的增加，用户将支付更多费用。

简单而言，IaaS 提供工具来从底层“构建”系统。PaaS 让用户可以“部署”应用，而无须担心底层的基础架构。SaaS 让用户可以“购买”应用——甚至不需要部署或管理它们。这是减少控制和复杂性的稳步演变，同时提升了直接的业务价值。

这些是云计算的一般模型，但事实上它们之间的区别并不总是一清二楚的。SaaS 和 PaaS 之间的关系尤其复杂。具体的、复杂的 SaaS 可能会使用 PaaS 甚至其他更加精细的 SaaS。PaaS 甚至可能组合更低层的部分作为软件服务的集合。例如，大多数服务需要一个身份管理（认证、授权和计费）服务。这个身份识别服务是 PaaS 提供给应用的关键功能之一。然而，这个服务没有理由不可以由一些外部的 SaaS 提供！在这种情况下，PaaS 的一个关键功能由低层的 SaaS 提供。

1.1.2 云基础设施部署模型

云基础设施部署模型主要有公有云、私有云和混合云 3 种。

公有云是大多数开发者所熟悉的类型。这些云服务对一般公众收费提供，一般是在使用量的基础上收取费用，使组织可以利用其运营预算而不是资本预算。客户没有必要维护或操作硬件 / 云基础设施，而是完全把这个职责交给云计算运营商。

AWS 是目前最大的公有云，并且主导着这个行业。微软和 VMware 也经营公有云，一些服务提供商同样如此。特别是 Rackspace，其提供了一个基于 OpenStack 的公有云，并且是 OpenStack 项目的主要贡献者之一。

私有云位于组织内部，它们代表了传统企业数据中心的演变。只有企业内部客户或者亲密合作伙伴才能使用私有云。企业 IT 部门或承包商将购买、安装和维护私有云的硬件和软件。云基础设施可以通过对业务单元收费来分摊成本，但是云自身仍然专用于单个企业。

组织机构经营私有云的原因有很多。一个运行良好的私有云的成本可能少于公有云。此外，许多行业由于安全或监管的原因，很多工作负载不允许使用公有云。这些组织需要在私有云中运行某些工作负载。

笔记

混合云结合了私有云和公有云，其目标是通过在私有云上运行大部分工作负载，并在 需要时将溢出部分运行在公有云上，来保持低水平的总体运营成本。溢出可能由于容量的原因而产生——也许是在节日期间私有云没有足够的容量或者进行灾难恢复。这种模式避开了私有云的容量限制，同时仍然保持了成本控制。

公有云、私有云和混合云的结构如图 1-2 所示。

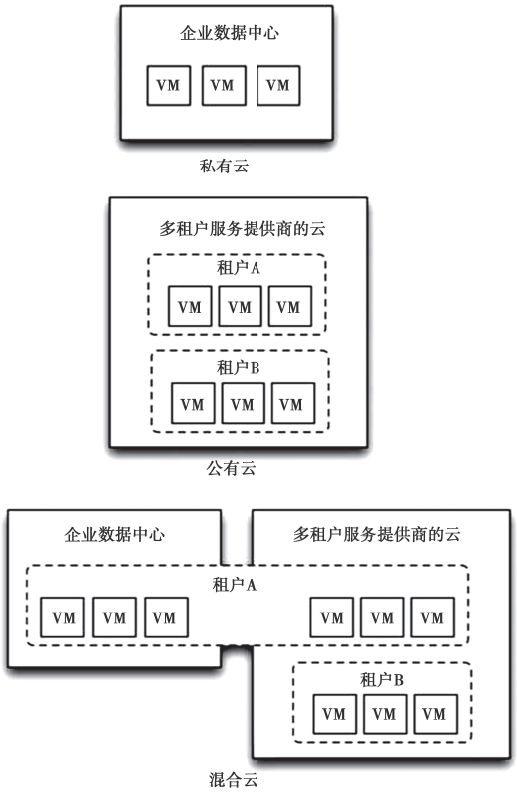


图 1-2　公有云、私有云和混合云的结构

1.2 OpenStack 概述

1.2.1 OpenStack 的优势

OpenStack 是由 Rackspace 公司（世界上最大的主机托管服务商之一）启动的一个开源项目，它旨在实现“云操作系统”，即一个具有部署和管理公有云、私有云以及混合云基础架构能力的平台。它提供了将物理计算、存储和网络资源抽象为池的能力。这些资源可以在用户中以安全的方式分配。用户只需要对其正在使用的资源付费即可，而不需要为其应用提供峰值负载。互联网厂商和云计算提供商是 OpenStack 的潜在用户，这也同样为准备部署云计算基础架构的企业提供了一种选择。

OpenStack 具备以下优势：

（1）模块松耦合。与其他开源软件相比，OpenStack 模块分明，添加独立功能的组件非常简单。有时候，不需要通读整个 OpenStack 的代码，只需要了解其接口规范及 API 的使用规划，即可轻松地添加一个新的模块。

（2）组件配置较为灵活。OpenStack 也需要不同的组件，但是 OpenStack 的组件安装

异常灵活，可以全部安装在一台物理机上，也可以分散至多台物理机中，甚至可以把所有的节点都安装在虚拟机中。

笔记

（3）二次开发容易。OpenStack 发布的 OpenStack API 是 RESTful API。其他组件也采用了这种统一的规范。因此，基于 OpenStack 进行二次开发会较为简单。而其他开源软件则由于耦合性太强而导致添加功能较为困难。

1.2.2 OpenStack 的特色

OpenStack 每 6 个月发布一个正式版本。为了更简单地跟踪这些版本，它们以字母顺序命名。OpenStack 的历史版本如表 1-1 所示。

表 1-1　OpenStack 的历史版本

版本	状态	发布时间
Victoria	发展	2020-10-14
Ussuri	维持	2020-05-13
Train	维持	2019-10-16
Stein	维持	2019-04-10
Rocky	延期维持	2018-08-30
Queens	延期维持	2018-02-28
Pike	延期维持	2017-08-30
Ocata	延期维持	2017-02-22
Newton	终止	2016-10-06
Mitaka	终止	2016-04-07
Liberty	终止	2015-10-15
Kilo	终止	2015-04-30
Juno	终止	2014-10-16
Icehouse	终止	2014-04-17
Havana	终止	2013-10-17
Grizzly	终止	2013-04-04
Folsom	终止	2012-09-27
Essex	终止	2012-04-05
Diablo	终止	2011-09-22
Cactus	终止	2011-04-15
Bexar	终止	2011-02-03
Austin	终止	2010-10-21

除了版本的名称之外，每个版本都通过年份以及该年份中发布的版本标识——年 . 版本 . 补丁。例如，Kilo 作为 2015 年发布的第一个版本也被称为 2015.1，Kilo 的补丁版本是 2015.1.1 和 2015.1.2 等；2015 年的第二个重大发布版本是 Liberty，也称为 2015.2。

表 1-2 显示了 OpenStack 提供的主要服务，包括它们的名称。OpenStack 社区成员经常会以名称提及每个服务，因此，在一个地方查看所有的服务并了解其用途是很有帮助的。

笔记

表 1-2　OpenStack 提供的主要服务

名 称	服 务	描 述
Horizon	仪表板	用于管理云的图形化用户界面
Keystone	身份识别	认证、授权和 OpenStack 服务信息
Nova	计算	建立、管理和终止虚拟机
Cinder	块存储	磁盘卷（比实例更持久）和实例快照
Swift	对象存储	共享的、复制的和冗余的存储，用于存储图片、文件和其他可通过超文本传输协议（Hypertext Transfer Protocol，HTTP）访问的媒体文件
Neutron	网络	提供安全的租户网络
Glance	镜像	提供虚拟机镜像和快照的存储与访问
Heat	编排	通过模板编排计算机、网络和其他资源组
Designate	DNS	创建域并在 DNS 基础设施中记录
Ceilometer	计量	监控整个云的资源使用情况
Trove	数据库	提供私有租户数据库的访问
Ironic	裸机	在物理硬件上启动实例
Magnum	容器	在实例中管理容器
Murano	应用	在多个实例上部署打包应用
Sahara	数据处理集群	将 Hadoop 或 Spark 集群作为服务提供

这些服务都可以通过 RESTful API 访问，也可以通过命令行接口和称为 Horizon 的 Web 用户界面访问。Horizon 可便于临时进行一些设置，但是不提供 API 的完整功能。当然，API 和 CLI 工具可以轻松地实现脚本化，如图 1-3 所示。

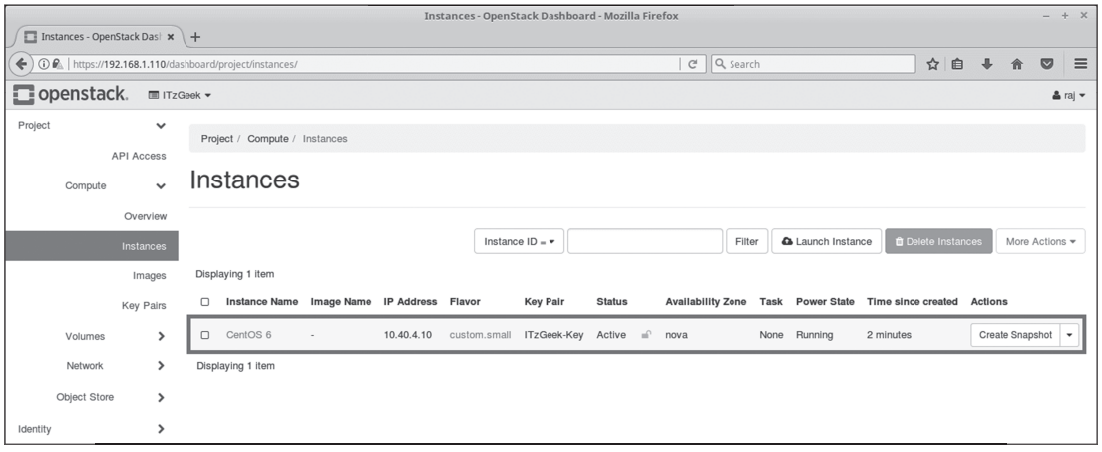


图 1-3　脚本化

OpenStack 的默认安装将包含每个服务的“参考”版本。例如，默认情况下，OpenStack 云会使用基于内核的虚拟机（Kernel-based Virtual Machine，KVM）管理程序来管理虚拟机。然而，OpenStack 架构最重要的一个方面是驱动程序或每个服务基于插件的性质。有了这个设计，除了参考方式以外，都可以使用另一种实现。在云中，可以用 ESXi、Xen 或其他管理程序替换 KVM。不管底层是什么管理程序，用于启动和管理虚拟机的 API 都保持不变。同样的概念贯穿于所有的 OpenStack 服务，不同的服务实现拥有相同的 API。

在后台提供这种程度的灵活性，同时提供一致性的 API，是 OpenStack 成功的关键因素之一。用户可以在 OpenStack 之上建立其应用并实现自动化，而不必担心把自身限定于计算机、网络和存储的单个后端供应商。即便用户换掉后端，API 也不会改变。

OpenStack 在企业私有云中经常使用，但也有一些基于它的公有云服务。也有一些公司将在其数据中心中创建和运营一个私有云。在这种情况下，硬件不与其他的客户共享，因此拥有私有云的可预测性和安全性，但是不需要寻找和雇佣专家来维护它。

即使在私有云环境中，OpenStack 也是一个多租户云平台。这意味着多个用户或用户组（租户）可以利用云的物理资源，同时保持其所有虚拟资源的私有化。对于租户而言，OpenStack 环境在很大程度上看起来好像是自己的，且仅属于自己。但是对于运营商，底层的物理资源和软件系统是共享的。在 OpenStack 中，租户有时候也称为项目。

在一个多租户的 OpenStack 云中，对于各类可以使用的资源，每个租户都分配有额度。该额度为租户提供了特定资源的最大限制。租户会有 CPU、内存、存储、网络、子网、浮动 IP 地址及其他资源的额度。这将防止任何单一租户消耗掉所有的资源。

1.3 OpenStack 的架构

OpenStack 建立在松散耦合的架构上。每个组件独立构建并运行自身的服务。这些服务可能分布在一些不同的计算机上，承担着不同的角色。通过增加特定角色的计算机数量，就能够扩充该功能的服务规模。它同样支持冗余，一种高可用的部署方式将针对每种类型包含多个计算机节点。

1.3.1 软件架构

独立组件之间通过定义良好的 API 进行交互，通常基于表述性状态转移（REpresentational State Transfer，REST）协议，某些情况下使用远程过程调用（Remote Procedure Calls，RPC）或消息总线通知。一般情况下，这些服务会使用关系型数据库保存数据——常见的是 MySQL 和 PostgreSQL。消息总线和数据库可以跨服务共享，但是这些服务之间的交互仍然有清晰的划定。这使得不同的服务只要在 API 中提供向后兼容性，就可以独立地升级和更新。

每一个主要的服务计算（Nova）、网络（Neutron）、块存储（Cinder）等都包含几个内部过程和组件。一般来说，它们都拥有一个 API 服务来提供基于 HTTP 的 RESTful API。该 API 服务会通过消息总线与其他组件进行通信。

Horizon 服务是一个基于 Web 的 UI，它和多个服务进行交互，如图 1-4 所示。类似的，命令行工具也可以与每个服务进行交互。这些工具是可选的；如果愿意，可以直接在服务的 API 上建立自己的接口。Horizon 和官方的 CLI 客户端没有任何特殊的访问方式，它们都使用相同的 API。每一个客户端只需要被告知 Keystone（身份认证服务）的位置即可。Keystone 服务包含了 OpenStack 平台上所有可用的服务和 API 端点的目录。

笔记

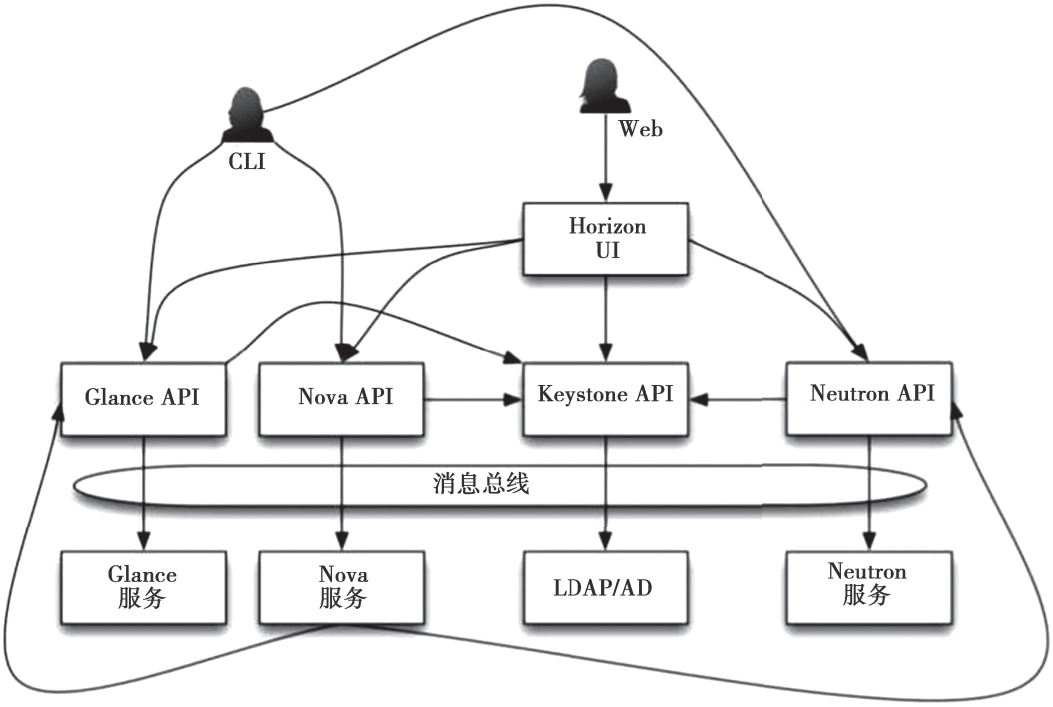


图 1-4　Horizon 服务

在图 1-4 中，可看到服务之间交互的简要描述。每个服务都有一个 API 组件，它通过 HTTPS 与 Keystone API 进行通信来提供认证和授权信息。每个 API 组件利用消息总线与该服务中的其他几个流程（在图 1-4 中称为“服务”）进行通信。在需要时，这些下游的服务流程将调用其他服务的 API。例如，Nova 会调用 Neutron API 来获取特定网络中的端口。

1.3.2 部署架构

OpenStack 的不同功能模块是如何部署到硬件上的呢？这实际上相当灵活。对于开发或者实验，甚至可以在一台计算机上运行所有的模块。然而，一种更典型的部署将由几个控制器节点（出于高可用性的目的）组成。另外，其中可以包含网络节点、计算节点和存储节点。

每个高级服务（计算、网络、存储和其他）都由多个守护进程（后台进程）组成。这些守护进程分散在各种不同类型的节点上。也就是说，不在单个节点上运行单个服务，而是将每个服务分散在不同类型的节点上。

例如，所有的服务都共享数据库和消息传递组件（通常是 MySQL 和 RabbitMQ）。可以在单独的集群上运行这些模块，每个集群分散在不同的故障域上。此外，在一个物理负载均衡器的后方，可以有几个提供 API 端点的物理节点。Nova 和 Neutron 的不同守护进程将分布在网络和计算节点中。图 1-5 展示了其部署架构的简化图形。

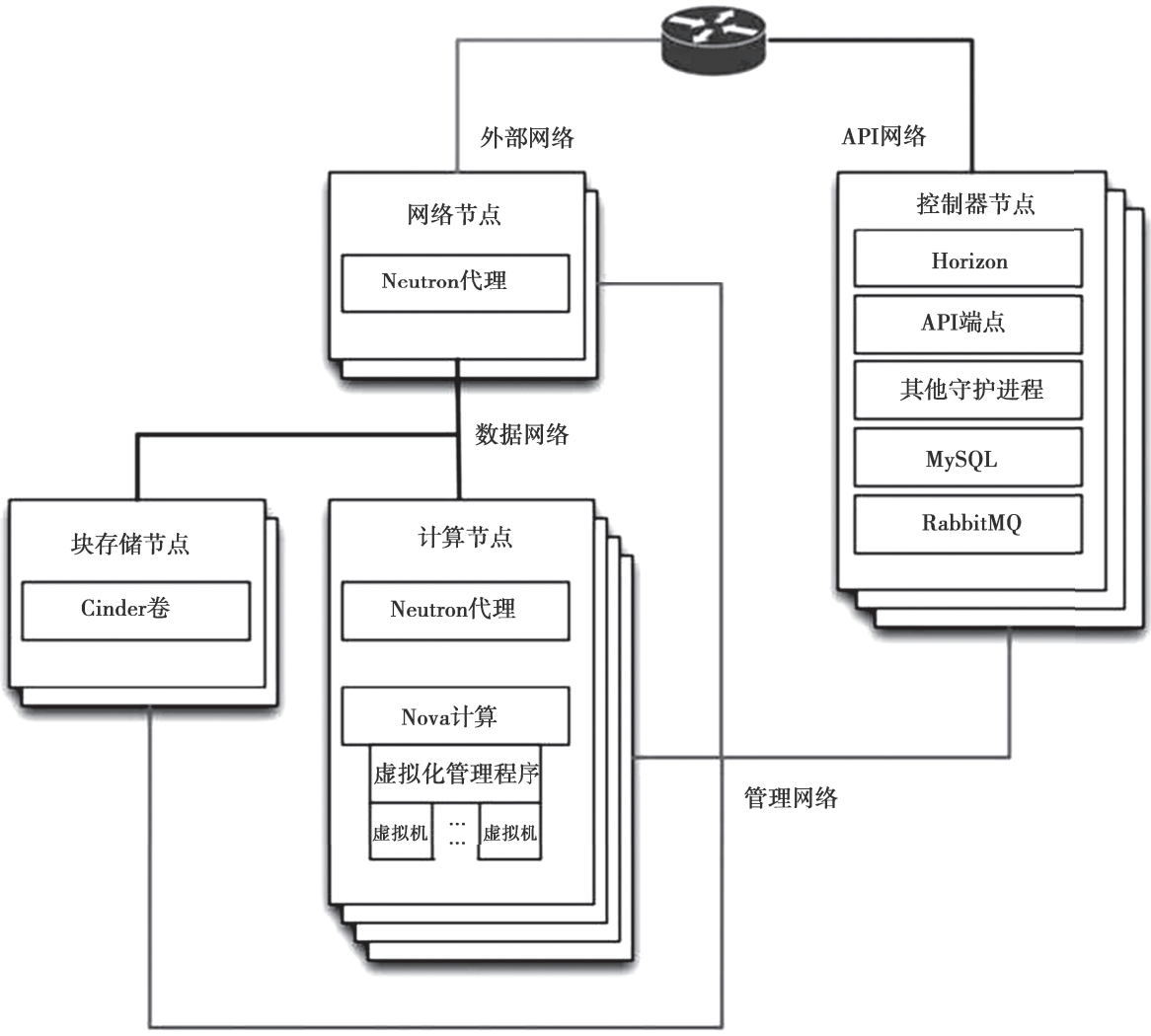


图 1-5　OpenStack 部署架构的简化图形

注意图 1-5 中不同类型的节点。计算节点运行虚拟化管理程序和实际的虚拟机实例，并为这些实例提供临时性存储。它也运行 Neutron 网络代理来管理虚拟机之间的连通性（称为“东西流量”）。

网络节点通常提供虚拟机和云外部之间的连通性（称为“南北流量”），以及高级的网络服务，如负载均衡和 VPN 访问。根据管理员和用户的选择，可能由网络节点上的代理提供网络路由服务，也可能直接由计算节点上的代理提供，或者由这两者提供。

块存储节点为虚拟机实例提供数据卷服务——也就是说，它们提供可与实例挂接和分离的磁盘卷的持久性存储访问。提供对象存储的云也有单独的集群用于此方面。对象存储为图像、文件和其他媒体文件提供共享的、复制的和冗余的存储，并且可以通过 HTTP 访问。

所有节点通过各种隔离的网络连接。每个节点都可以通过管理网络访问，管理网络用于 OpenStack 不同部分之间的通信。所有的消息总线、数据库和跨项目 API 之间的流量都通过管理网络。数据网络连接所有的计算节点、网络节点和块存储节点。内部云租户的流量都通过数据网络，而外部网络提供了对云外部的访问。由于计算节点只与云内其他节点通信而不与外界通信，因此它不需要连接外部网络，而只需要访问数据网络。只有网络节

笔记

点需要连接外部网络。最后，有些安装方式会使用 API 网络，它与租户使用的外部网络相分离，提供了外部世界和 OpenStack 端点（API 和 Horizon）之间的访问。

1.3.3 OpenStack 架构的优势

OpenStack 的架构提供了很高的灵活性。通过云运营商部署额外的节点来扩展基础设施以实现可扩展性。它也能够创建高可用的服务，因为可以拆分每个服务并跨故障域冗余部署。然而，该架构是非常复杂的，并且很难建立和维护。

作为云的用户，云对用户来说是透明的。但是一个正常运行的云只有建立足够的冗余度，OpenStack 基础设施才能拥有高可用性。

这种架构的另一个实质性好处是避免了厂商锁定。每个服务提供一个插件或基于驱动程序的架构。这使得每个服务可以与任意数量的供应商平台合作来提供实际服务。对于计算，可以使用默认的 KVM 管理程序、ESXi、Xen 或其他管理程序。网络服务默认使用 Open vSwitch 来提供第 2 层（数据链路层或 MAC 地址层）连接，并提供 Linux 网络协议栈（IP 表、路由和命名空间）来实现第 3 层（IP 层）功能。然而，有超过 20 个不同的厂商插件可替换全部或部分默认实现。事实上，这些厂商的实现可以在同一个云上的同一时间使用。

通过避免厂商锁定，OpenStack 使厂商之间的竞争变得更加激烈，这推动了市场价格的下跌。同时，使用多个厂商的能力使得从一个厂商过渡到另一个厂商更加切实可行，并且允许选择厂商来解决特定的应用案例。