本三《云计算》

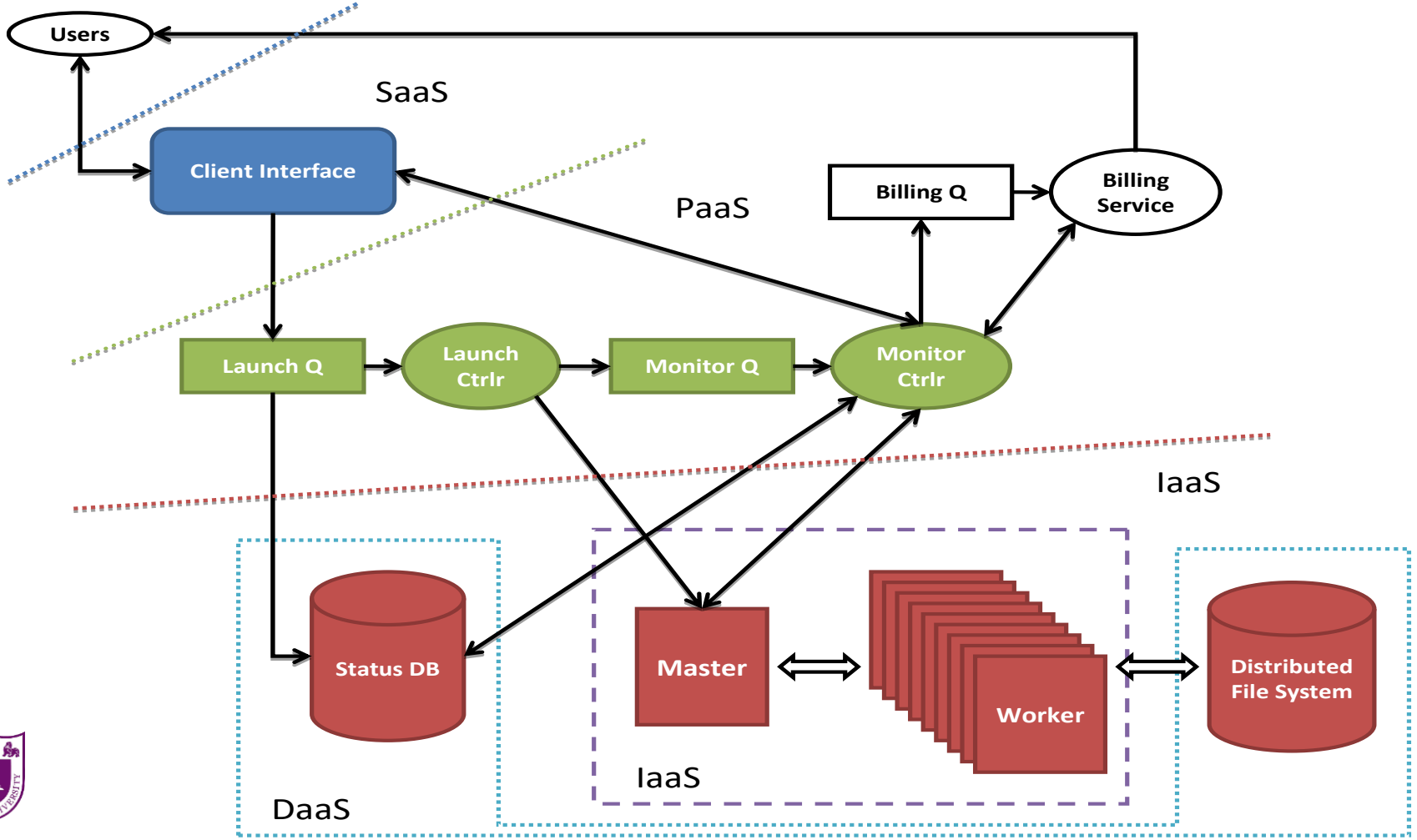
5 OpenStack开源云计算平台

李传艺
2022

云数据中心架构回顾

- IaaS，DaaS，PaaS，SaaS

OpenStack Docs: Wallaby

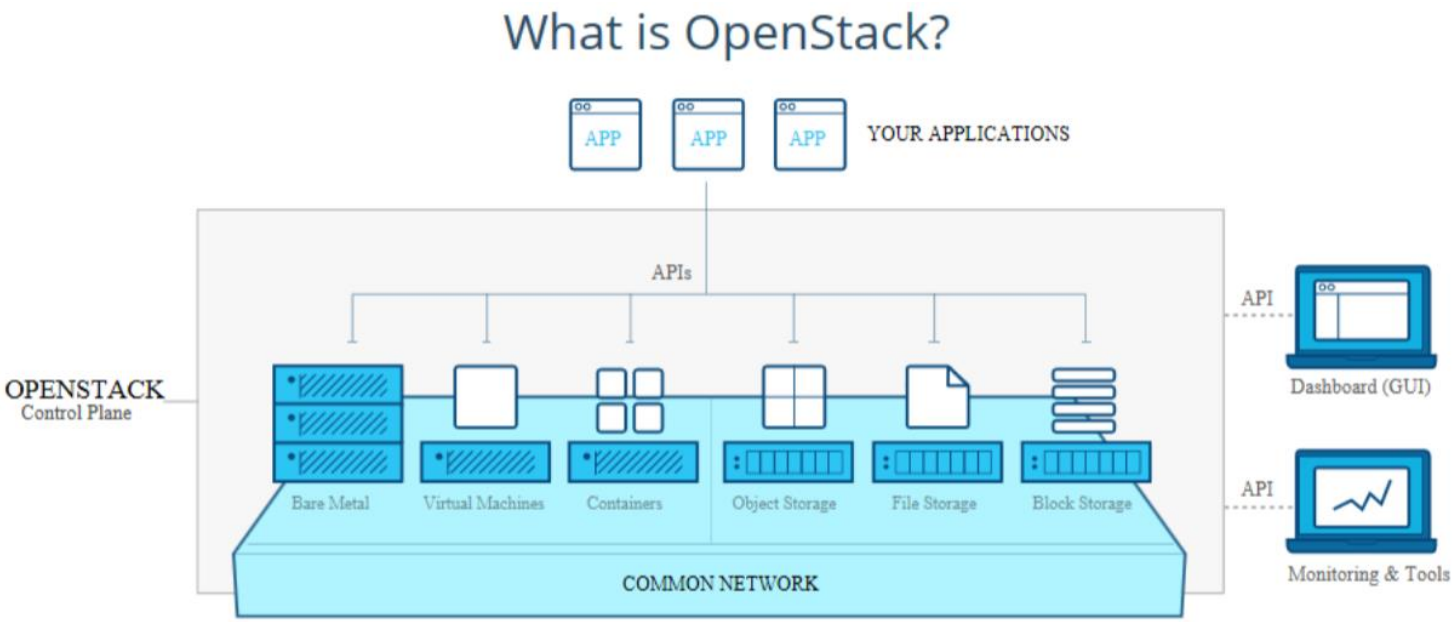


Openstack是什么?

- OpenStack是开源云计算平台，可控制整个数据中心的大型计算，存储和网络资源池。
- 用户能够通过Web界面、命令行或API接口配置资源。

- 自身不提供虚拟化技术
- 调用多种技术实现多资源池管理
- 对外提供统一管理接口

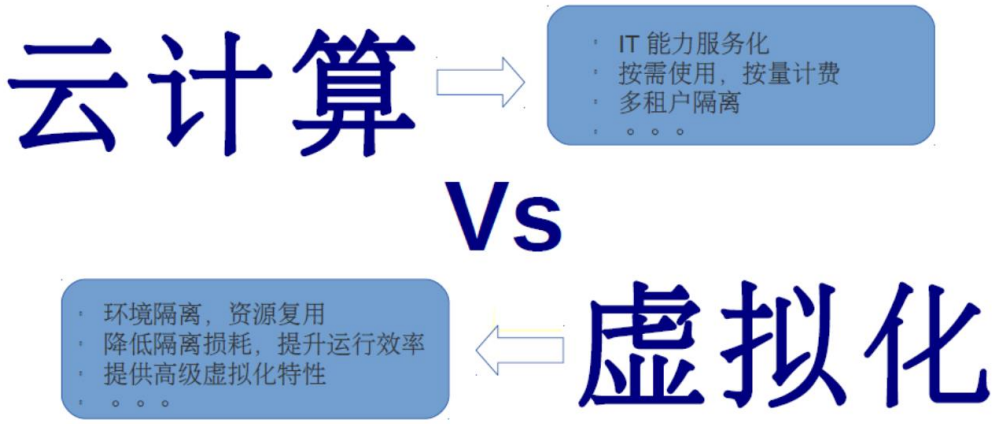
- 环境隔离，资源复用
- 降低隔离损耗，提升运行效率
- 提供高级虚拟化特性



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——云计算操作系统

- 云计算—虚拟化



- 操作系统的功能
  - 资源抽象：计算、存储、网络、I/O等
  - 资源分配与负载调度：各种CPU调度算法、内存调度算法
  - 应用生命周期管理：启动后的不同状态、关闭后的管理
  - 系统运维
  - 人机交互

Openstack的定位：
云计算系统的控制面

执行面

云计算操作系统（1）

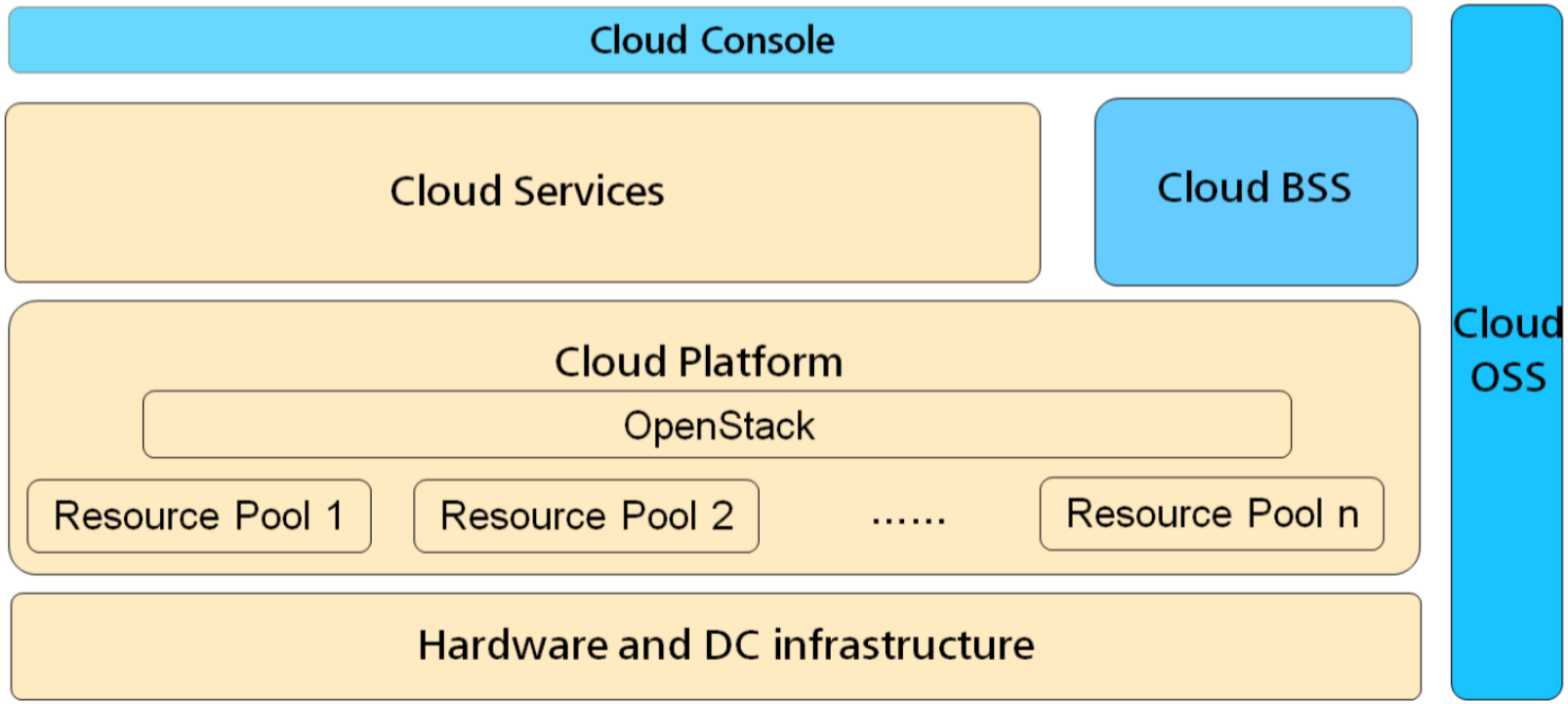
- 概念：云操作系统指构架于服务器、存储、网络等基础硬件资源和单机操作系统、中间件、数据库等基础软件之上，管理海量的基础硬件、软件资源的云平台综合管理系统。
  - 管理和驱动海量服务器、存储等基础硬件，将数据中心的硬件资源逻辑上整合程一台服务器
  - 为云应用软件提供统一的、标准的接口
  - 管理海量的计算任务以及资源调配和迁移

	管理的资源	运行的例程	接口功能	包含的分布式库
传统OS	一台或多台电脑	调度器、虚拟内存分配、文件系统和中断处理程序	提供管理底层硬件的库函数	标准分布式库和软件包
云OS	云资源	提供更多附加功能，虚拟机的分配和释放、任务的分配与融合	提供基于网络的接口管理资源	为分布式应用提供自主扩展和灵活调度的软件支持

- 特点
  - 网络化：
  - 安全
  - 计算的可扩充性

Openstack——云计算操作系统

- 构建云计算还需要很多其他东西



目录

简介——概述

- 既是一个社区，也是一个项目和一个开源软件，提供了一个部署云的操作平台或工具集。用OpenStack易于构建虚拟计算或存储服务的云，既可以为公有云、私有云，也可以为大云、小云提供可扩展、灵活的云计算。
- OpenStack是一个管理计算、存储和网络资源的数据中心云计算开放平台，通过一个仪表板，为管理员提供了所有的管理控制，同时通过Web界面为其用户提供资源。
- Rackspace公司的“云文件”平台Swift+美国宇航局NASA“星云”平台Nova

计算资源管理

OpenStack可以规划并管理大量虚拟机，从而允许企业或服务提供商按需提供计算资源

存储资源管理

OpenStack可以为云服务或云应用提供所需的对象及块存储资源

网络资源管理

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高扩展、高自动化地管理下一代网络

文档： https://docs.openstack.org/install-guide/index.html

简介——设计理念

- 开放：任何个人、企业、组织、政府机构，只要遵循相关开源协议，都可以使用、拿到源代码
- 灵活：带来可以定制的好处，不会因为某一个组件的原因受到限制
- 可扩展：因为是相互独立的项目，可以自己解决可扩展的问题

开放

Open

- 开源，并尽最大可能重用已有开源项目
- 不要“重复发明轮子”，而要“站在巨人肩膀上”

灵活

Flexible

- 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

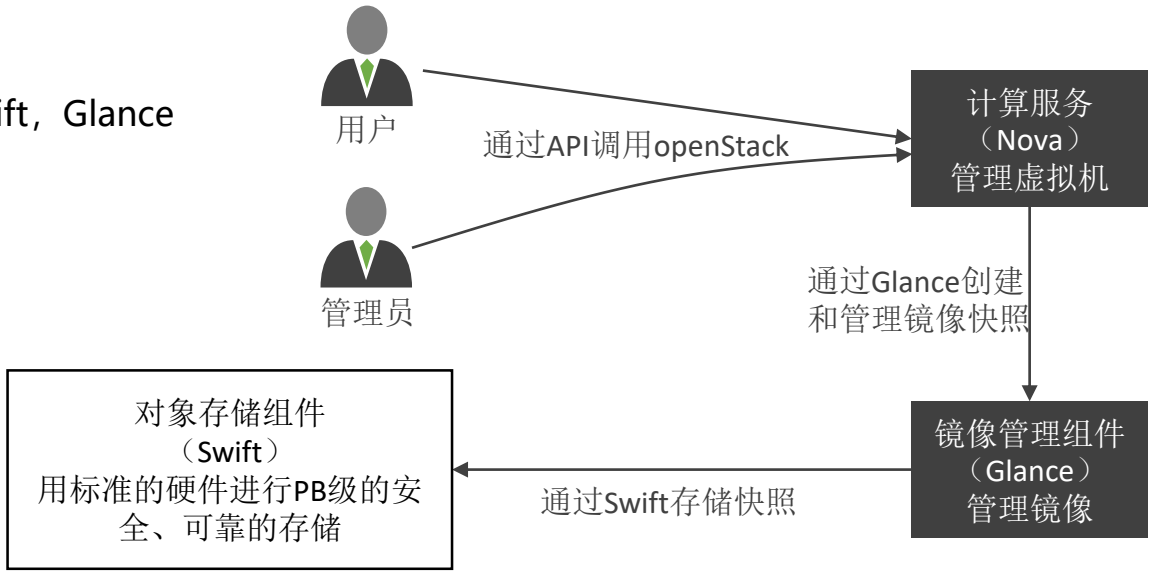
可扩展

Scalable

- 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

简介——参考架构

- 最开始的时候：核心部件
  - Nova（Nebula），Swift，Glance

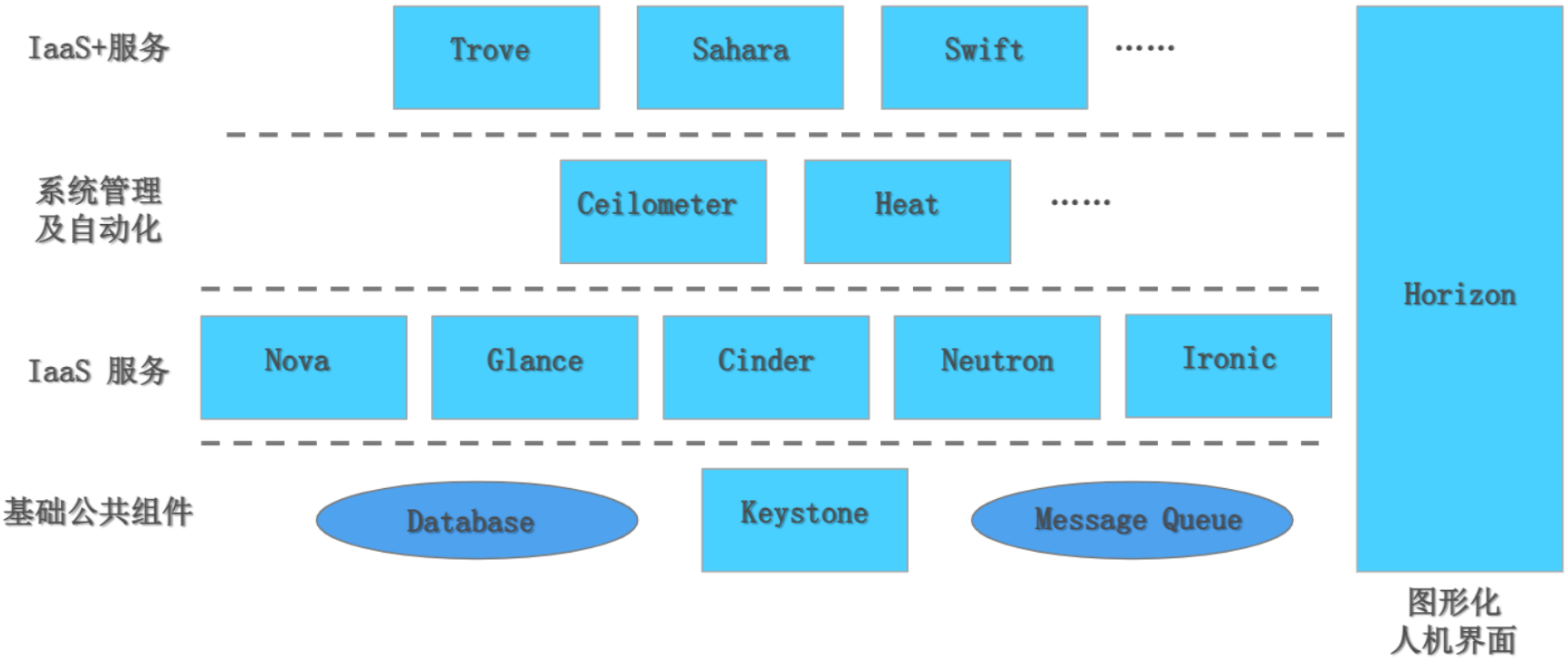


- 逐步发展：其他部件
  - 认证管理服务Keystone：用户或服务的验证或授权
  - 块存储服务Cinder
  - 文件共享存储服务Manila
  - 网络服务Neutron：提供网络即服务的功能
  - 计量服务Telemetry：计量租户资源使用率
  - 编排服务Heat：openstack开发者
  - 仪表盘服务Horizon：无状态无数据的web应用
  - 消息队列：中心化的消息交换器
  - 数据库：构建时和运行时状态信息

文件存储、对象存储、块存储的异同？

简介——参考架构

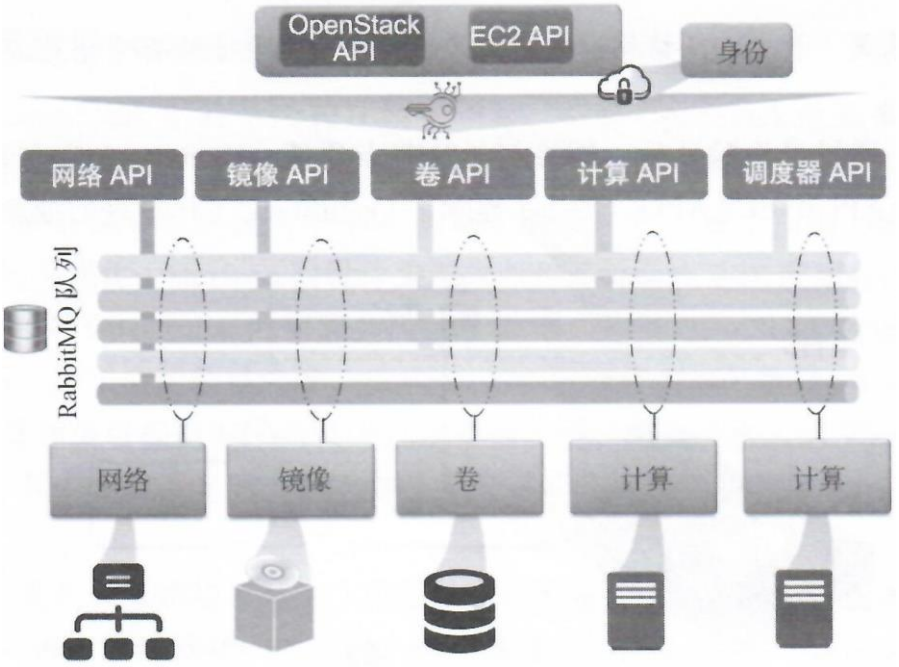
- 分层后的组件



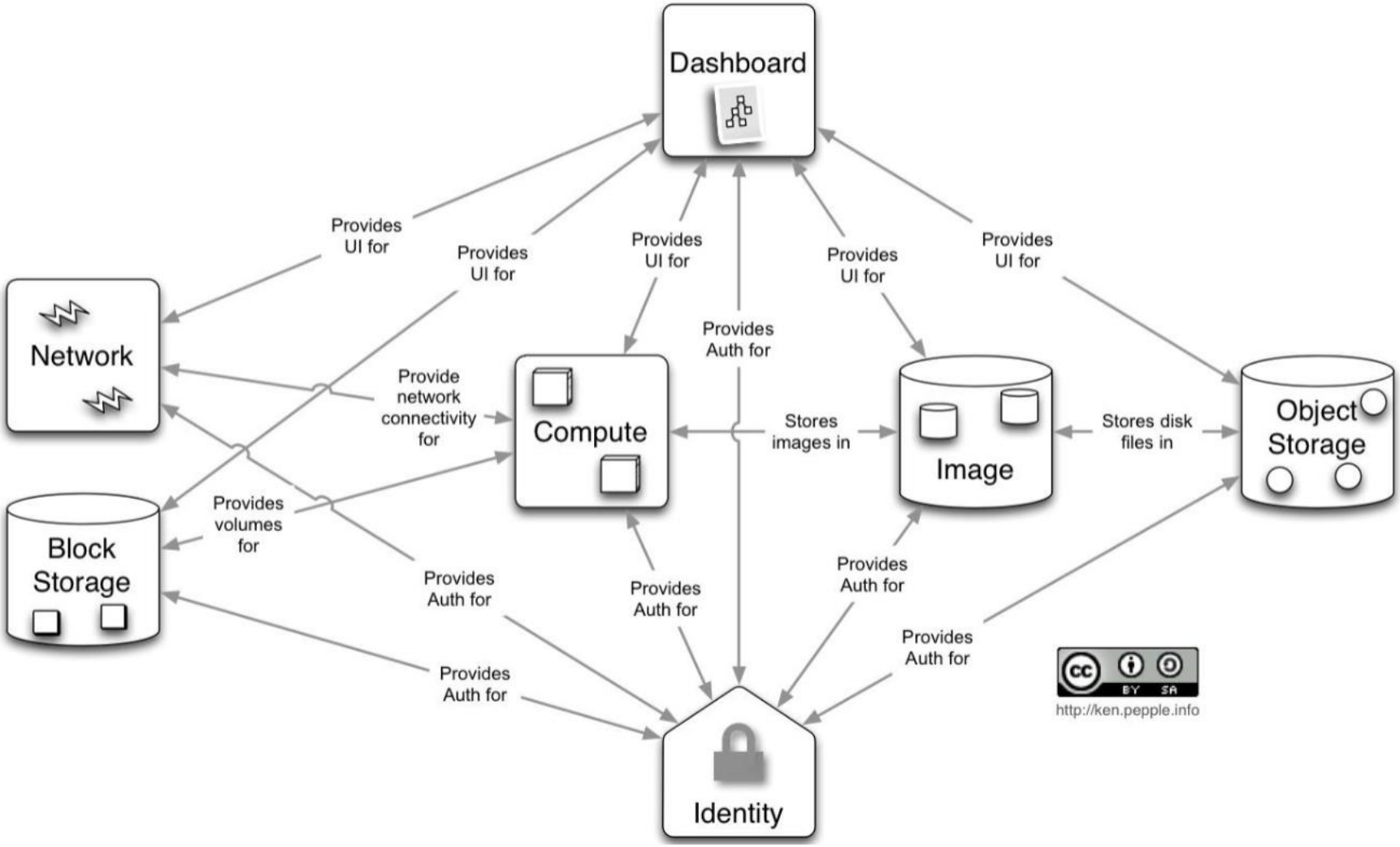
简介——虚拟机创建过程

- 资源准备
  - 通过Keystone进行用户身份认证，通过认证后，用户即可与Openstack API节点通信，触发创建请求
  - 通过调度器获得启动虚拟机的最佳位置：通过工作守护进程获得物理节点上的资源状态；Nova-scheduler

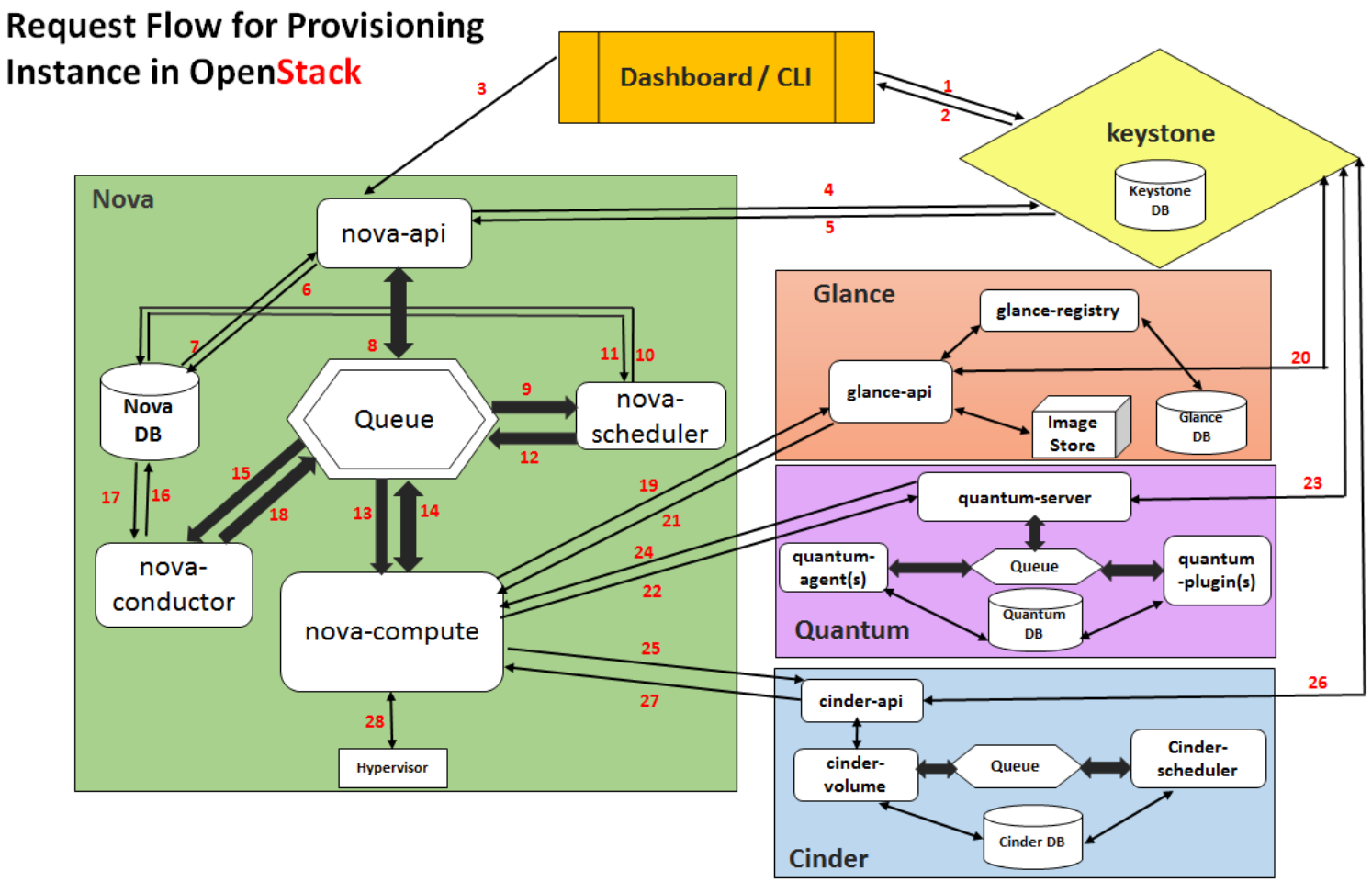
- 创建流程
  - 调用身份认证进行身份验证
  - 生成用于后续调用的令牌
  - 访问镜像服务以获取镜像列表，并获取目标基础镜像
  - 处理计算服务API请求
  - 处理计算服务对安全组和密钥调用的请求
  - 调用网络服务API确定可用网络
  - 通过计算节点调度程序选择Hypervisor节点
  - 调用块存储服务API为实例分配卷
  - 通过计算服务API调用再Hypervisor节点启动实例
  - 调用网络服务API为实例分配网络资源



简介——计算和存储、网络、镜像服务交互



简介——计算和存储、网络、镜像服务交互



Keystone交互多，令牌生成与验证带来性能问题：加密身份验证方法

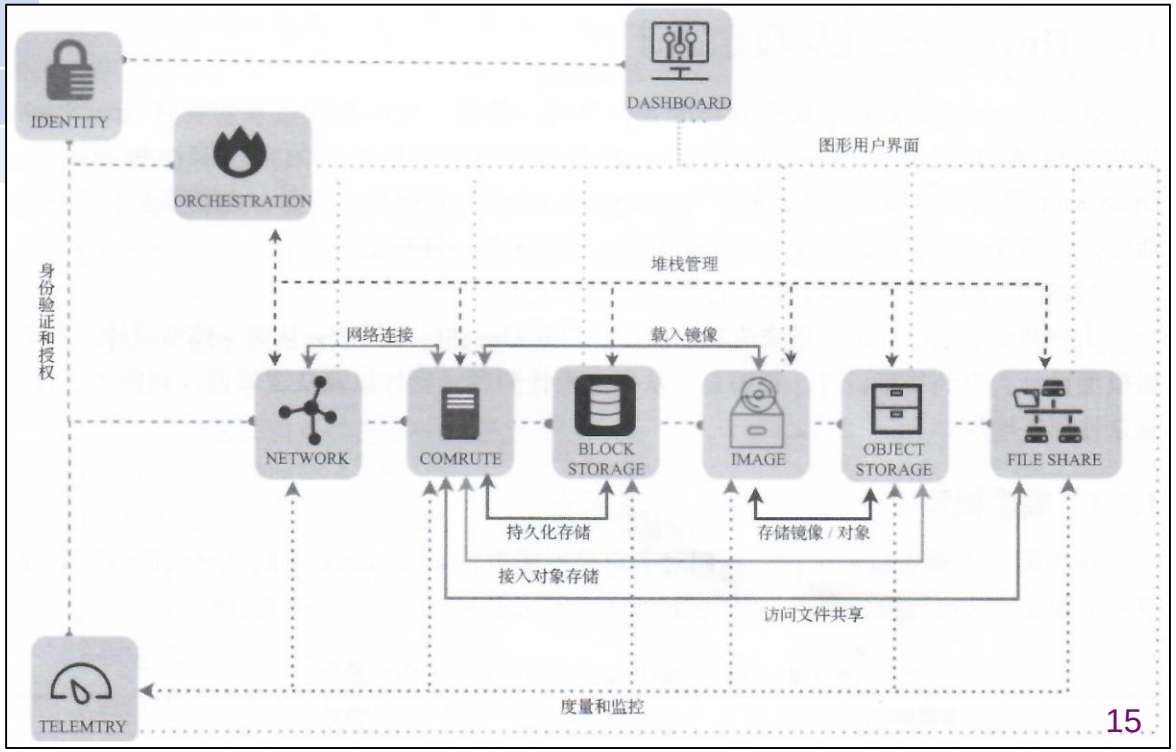
简介——架构设计

功能类别	角色	
计算	存储虚拟机镜像	提供用户接口
镜像	存储磁盘文件	
对象存储	存储对象	
块存储	提供卷	
网络	提供网络连接	
计量	提供度量、指标和报警	
文件共享	提供横向扩展的文件共享系统	
编排	为堆栈创建提供编排引擎	
身份认证	提供认证	
仪表盘	提供图形化用户界面	

概念模型设计

↓

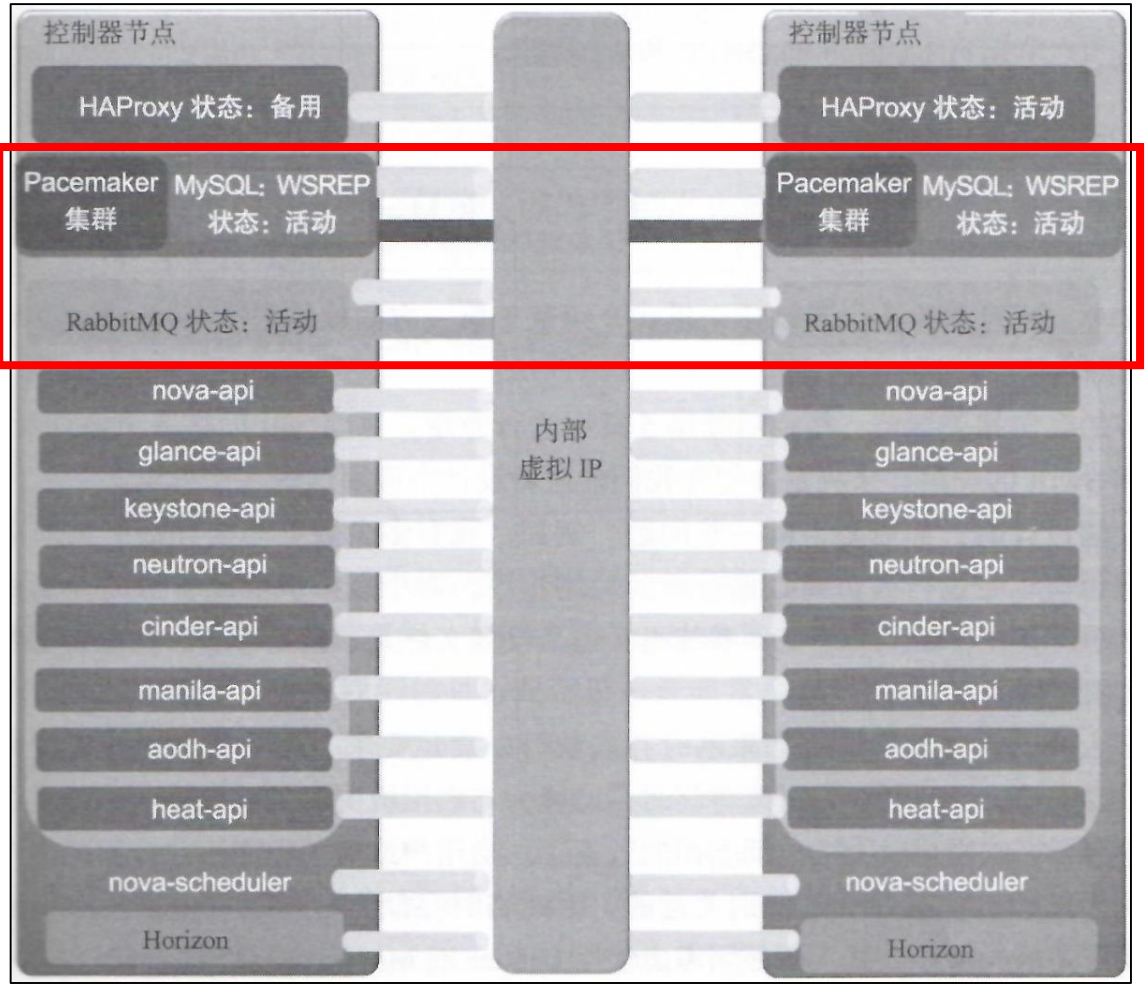
各个功能模块的功能及其之间的交互



简介——架构设计

逻辑模型设计

控制节点中封装了大量的Openstack的服务

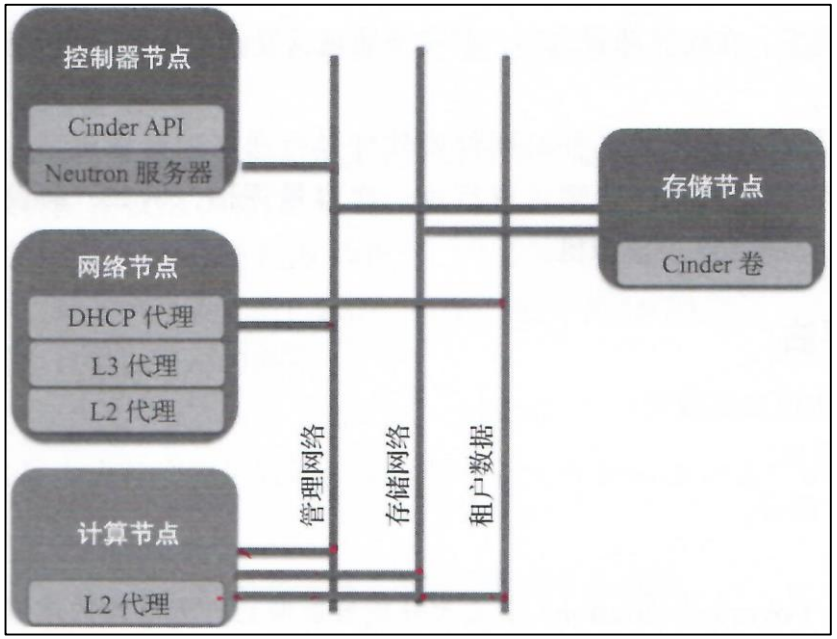


- 核心服务之间的依赖关系
- 运行Openstack服务的节点
  - 云控制器：控制所有服务的节点
  - 网络节点：运行网络服务的节点
  - 计算节点：运行虚拟机的节点（也可以运行网络服务）

简介——架构设计

- 存储设计
  - 有哪些数据需要存储？虚拟机快照备份的存储要求是什么？需要文件共享系统吗？是否需要虚拟机之间实现存储共享？

    系统运行时数据
    系统持久存储
    虚拟机快照
    系统镜像
    用户信息



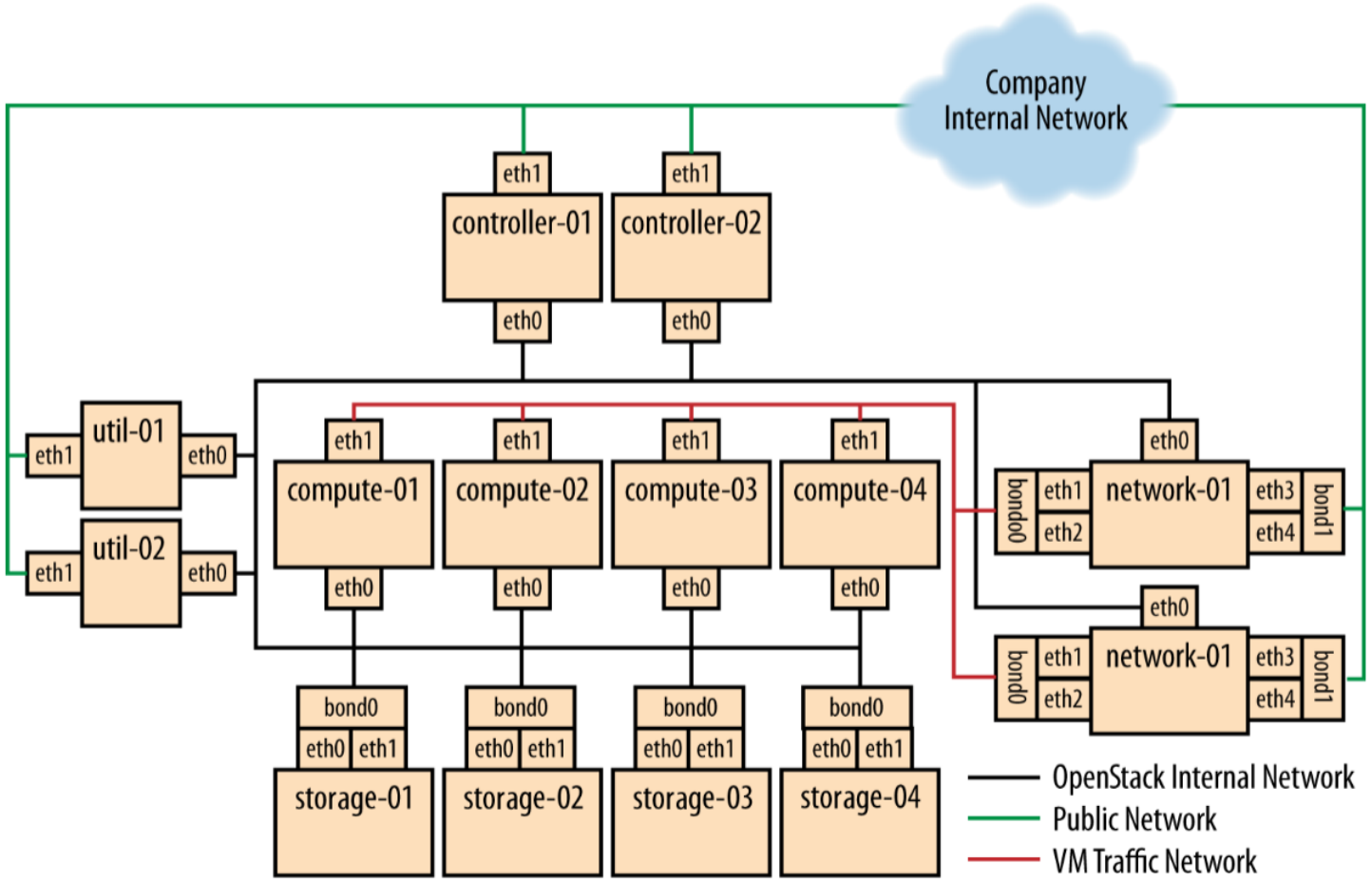
- 网络设计
  - 网络服务是Openstack服务中最复杂的
  - 使用隔离的物理网络处理不同类型的网络流量
    - 租户数据网络：为租户创建的虚拟网络提供物理路径
    - 管理和API网络：Openstack服务之间的通信，可细分
    - 存储网络：虚拟机和存储节点之间的物理连接

    租户网络是连接互联网的，是要计费的

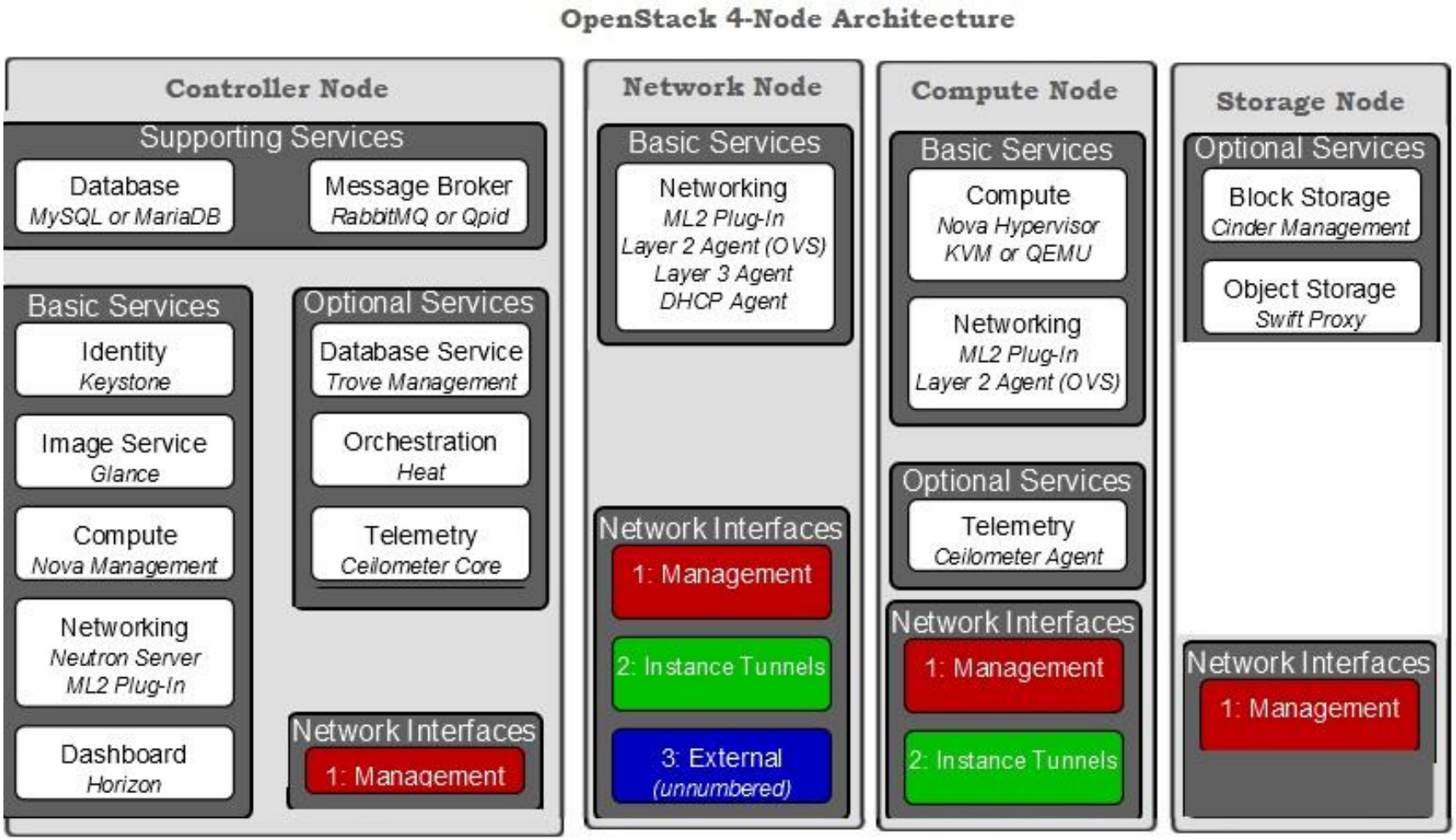
  - 虚拟网络类型
    - 外部网络：全局互联并使用可路由的IP寻址；SNAT虚拟机访问外网；DNAT外网访问虚拟机
    - 租户网络：虚拟机之间专用网络；私有IP空间；租户流量隔离

简介——物理部署示例

- 五种节点：部署各个系统的不同组件
- 三层网络：不同的使用目的；OpenStack内部网络、外部网络、虚机网络



简介——四种节点


OpenStack 4-Node Architecture
Controller Node
Supporting Services
Database
MySQL or MariaDB
Message Broker
RabbitMQ or Qpid
Basic Services
Identity
Keystone
Image Service
Glance
Compute
Nova Management
Networking
Neutron Server
ML2 Plug-In
Dashboard
Horizon
Optional Services
Database Service
Trove Management
Orchestration
Heat
Telemetry
Ceilometer Core
Network Interfaces
1: Management
Network Node
Basic Services
Networking
ML2 Plug-In
Layer 2 Agent (OVS)
Layer 3 Agent
DHCP Agent
Network Interfaces
1: Management
2: Instance Tunnels
3: External
(unnumbered)
Compute Node
Basic Services
Compute
Nova Hypervisor
KVM or QEMU
Networking
ML2 Plug-In
Layer 2 Agent (OVS)
Optional Services
Telemetry
Ceilometer Agent
Network Interfaces
1: Management
2: Instance Tunnels
Storage Node
Optional Services
Block Storage
Cinder Management
Object Storage
Swift Proxy
Network Interfaces
1: Management

简介——物理部署模型-估算

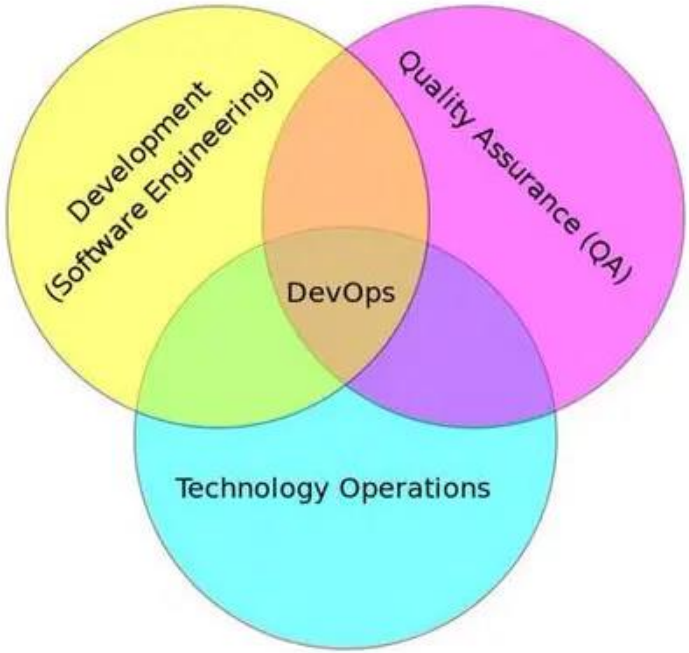
- 物理模型设计——估算硬件容量
  - 例如：在Openstack环境中运行200个虚拟机
    - CPU评估
      - 逻辑CPU数：物理CPU*核数*超线程数；一个逻辑CPU对应一个虚拟CPU；一个虚拟CPU支持多个虚拟机【影响性能；可以使用频数计算】；一个虚拟机可以使用多个虚拟CPU，但不能多于逻辑CPU数；
      - 可以超额分配，但是不能超额执行；超分：单台物理机中虚拟机个数 * 频数>物理频数
    - 内存评估
      - 单台物理机虚拟机个数*虚拟机最大动态分配内存+主机内存用量
    - 网络评估
      - 网卡带宽>单个虚拟机带宽*虚拟机个数；公共IP地址个数；浮动IP地址个数
    - 存储评估
      - 虚拟机的临时存储+永久存储；存储节点的对象存储、块存储和文件共享存储
  - 最佳实践
    - 分析需求、考虑需求增长情况、持续跟踪每一个服务单元、丢弃和替换服务单元；按需调整

目录

部署——DevOps

- 矛盾
  - 对于运维来说，稳定压倒一切，新 Feature 越少越好。而对于研发来说，却希望能开发更多的功能。
- DevOps 的理念
  - 让研发（Development）和运维（Operations）一体化，让团队从业务需求出发，向着同一个目标前进。

- DevOps 是一种软件研发管理的思想，方法论，追求的是一种没有隔阂的理想的研发协作的状态，可能涉及到的角色有开发、测试、产品、项目管理、运维等等。

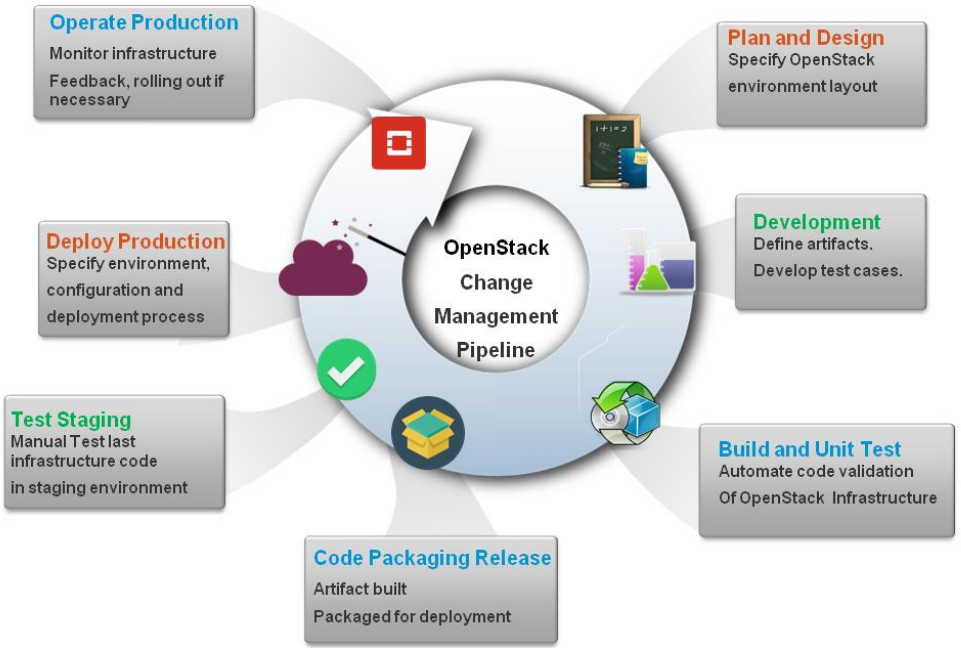


DevOps（Development和Operations的组合）是一组过程、方法与系统的统称，用于促进开发（应用程序/软件工程）、技术运营和质量保障（QA）部门之间的沟通、协作与整合。

部署——DevOps和Openstack

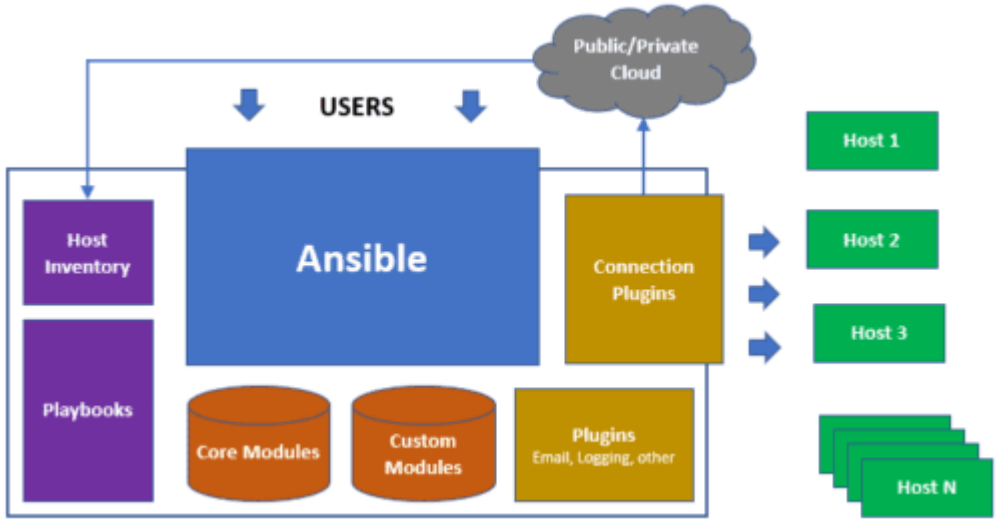
- 要求开发人员、网络工程师和运维人员必须相互协作，来部署、运维和维护OpenStack云基础架构
- OpenStack和DevOps结合
  - 要求
    - OpenStack软件和部署云基础架构的持续增长的复杂性必须被简化
  - 方式
    - 基础架构中的所有内容都必须自动化
    - 将OpenStack分解为多个部分
  - 要点
    - 简化和模块化OpenStack服务；像开发构建模块一样开发OpenStack服务
    - 在不影响整个系统的情况下，促进服务的定制和改进
    - 使用正确的工具来构建服务，确保服务在相同输入情况下输出相同结果
    - 将服务愿景从如何做切换到要做什么

基础架构代码生命周期管理



部署——Ansible

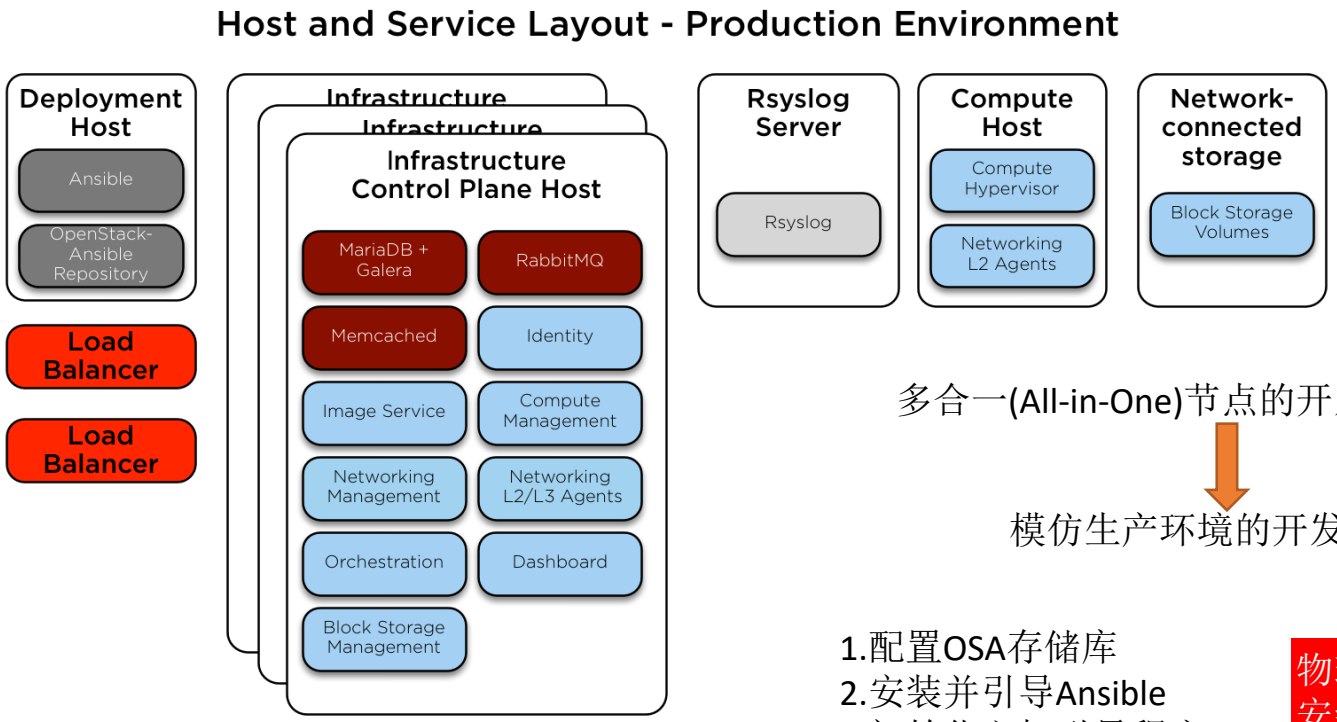
- 基础架构自动化引擎：安装Ansible自身部署系统和管理的目标系统
- 模块
  - 封装系统资源或操作的组件，对资源及其属性进行建模；通过执行模块改变目标系统状态；执行后清除
- 变量
  - 代表更新中的动态部分，可用于修改模块行为；可以用户自定义，作为模块一部分



- 清单
  - 由Ansible管理的主机列表；支持将主机分类到多个组中；同一个主机可以出现在多个分段中
- 角色
  - 表示在一组主机上配置服务时必须执行的任务集合；封装了在主机上部署服务所需的任务、变量、处理程序和其他相关功能；在Web服务器集群中为主机分配角色：Web服务器，数据库服务器，负载平衡器等
- Playbook
  - Ansible的主要配置文件；描述了完整的系统部署计划，由一系列任务组成；用YAML编写
  - Yet Another Markup Language

部署——OpenStack Ansible

- 自动化部署OpenStack
- 聚焦于提供多个角色和playbook，用于部署可扩展的、可理解投入生产的OpenStack环境



多合一(All-in-One)节点的开发者模式安装

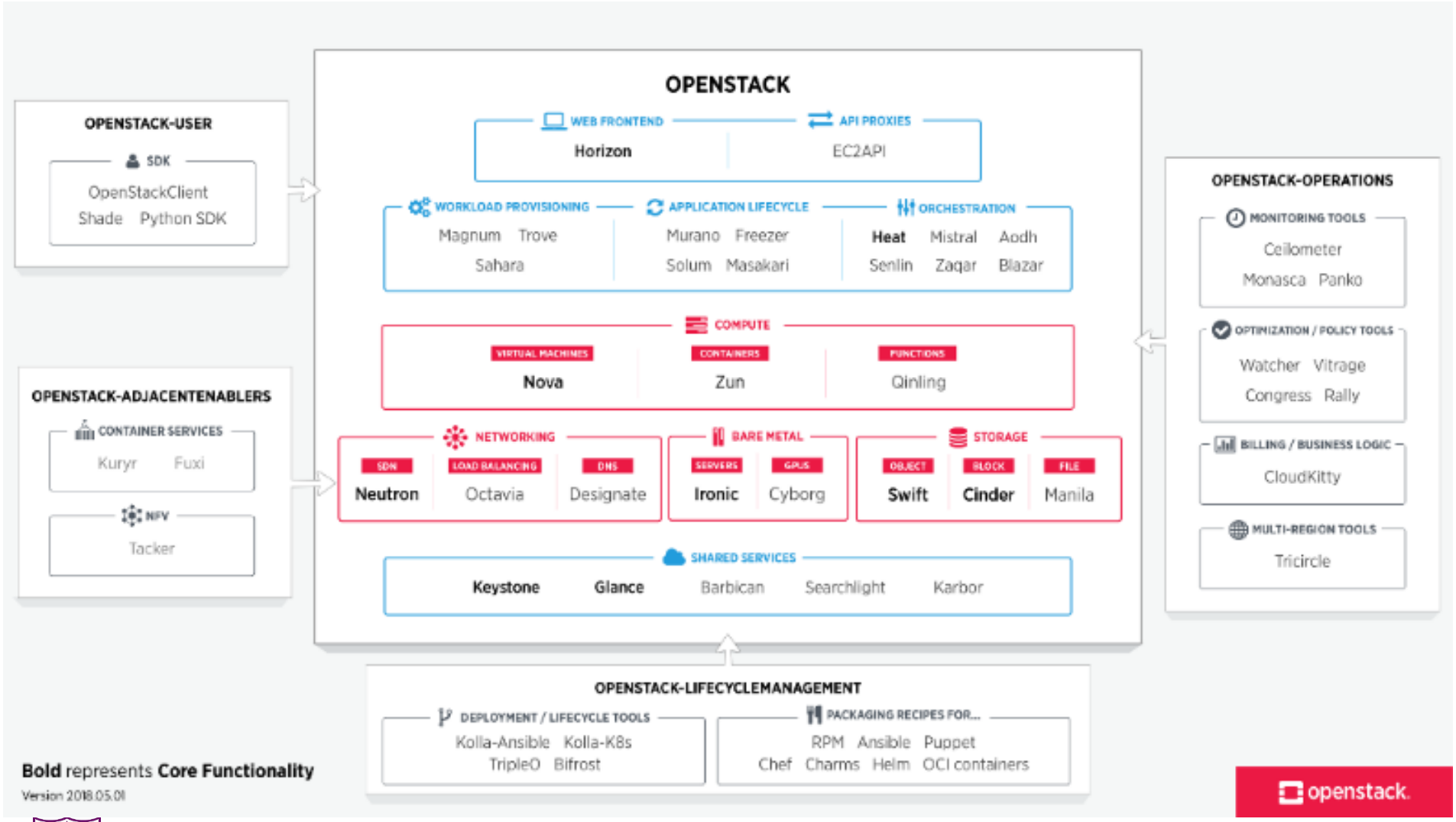
↓

模仿生产环境的开发测试环境

1.配置OSA存储库
2.安装并引导Ansible
3.初始化主机引导程序
4.运行Playbook

物理环境构建、开发机初始化、安装Git、执行引导程序仍需要手动完成

Map of OpenStack Projects



目录

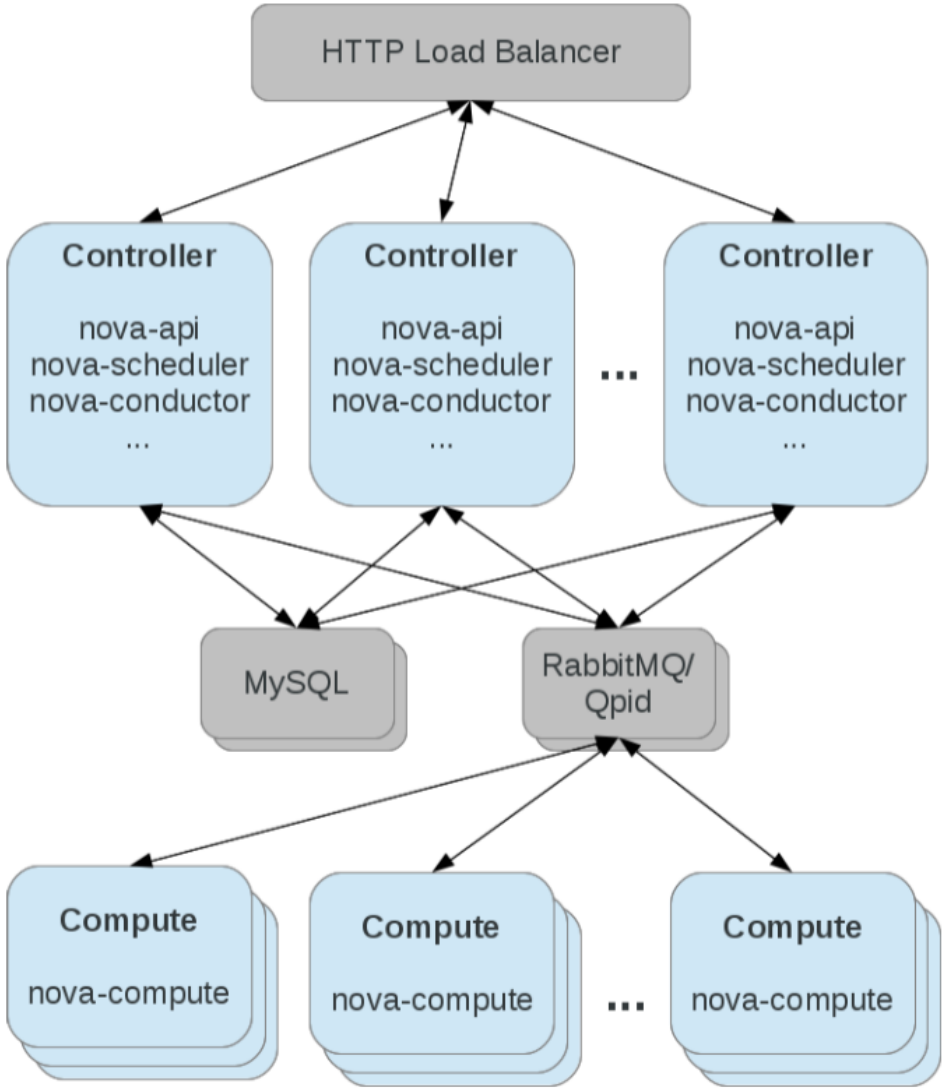
云控制器

- 集群：提供高可用性
  - 两台或多台服务器的能力聚合就是服务器集群，通过堆积机器即可实现这种聚合
    - 向上扩展（垂直扩展）：向服务器添加更多的CPU和内存
    - 向下扩展（水平扩展）：增加更多的标准商用服务器
  - 非对称集群
    - 备服务器只有在主服务器发生故障时才接管系统，处于高可用目的，包含故障切换配置
  - 对称集群
    - 所有节点都是活动的，共享工作负载，可视为负载均衡集群
- OpenStack的功能被广泛分布到多个服务中，包含了基础架构服务和OpenStack服务
  - 基础架构服务
    - 不属于OpenStack对外提供的公共服务，但被多个Openstack组件使用
    - 消息队列：必须是集群式的；RabbitMQ，ZeroMQ，Qpid
    - 整合数据库：OpenStack的环境数据，使用数据等；MySQL和MongoDB
  - OpenStack服务
    - 身份、镜像、计算、网络、存储等

云控制器——物理部署

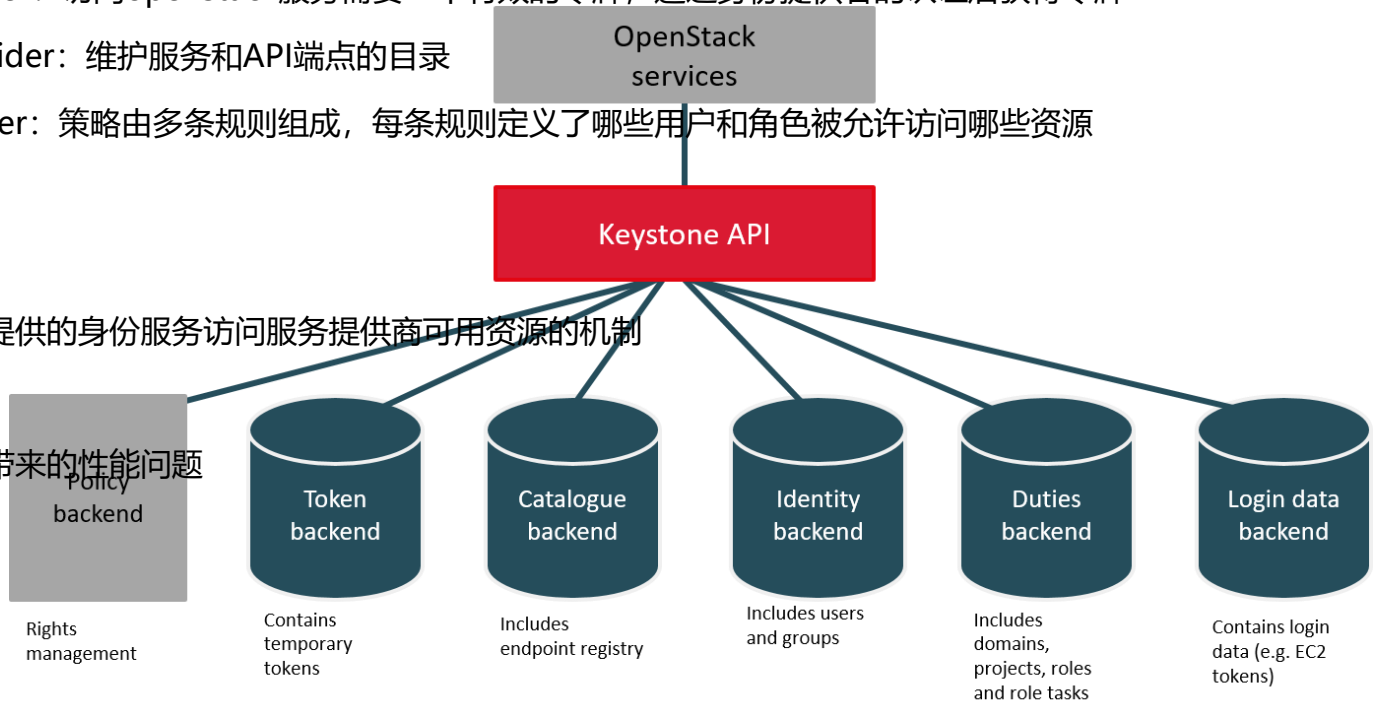
- 物理部署模型

- 无中心结构
- 节点无本地状态
- 多控制器：请求聚合+负载均衡
- 水平可扩展：控制节点、计算节点



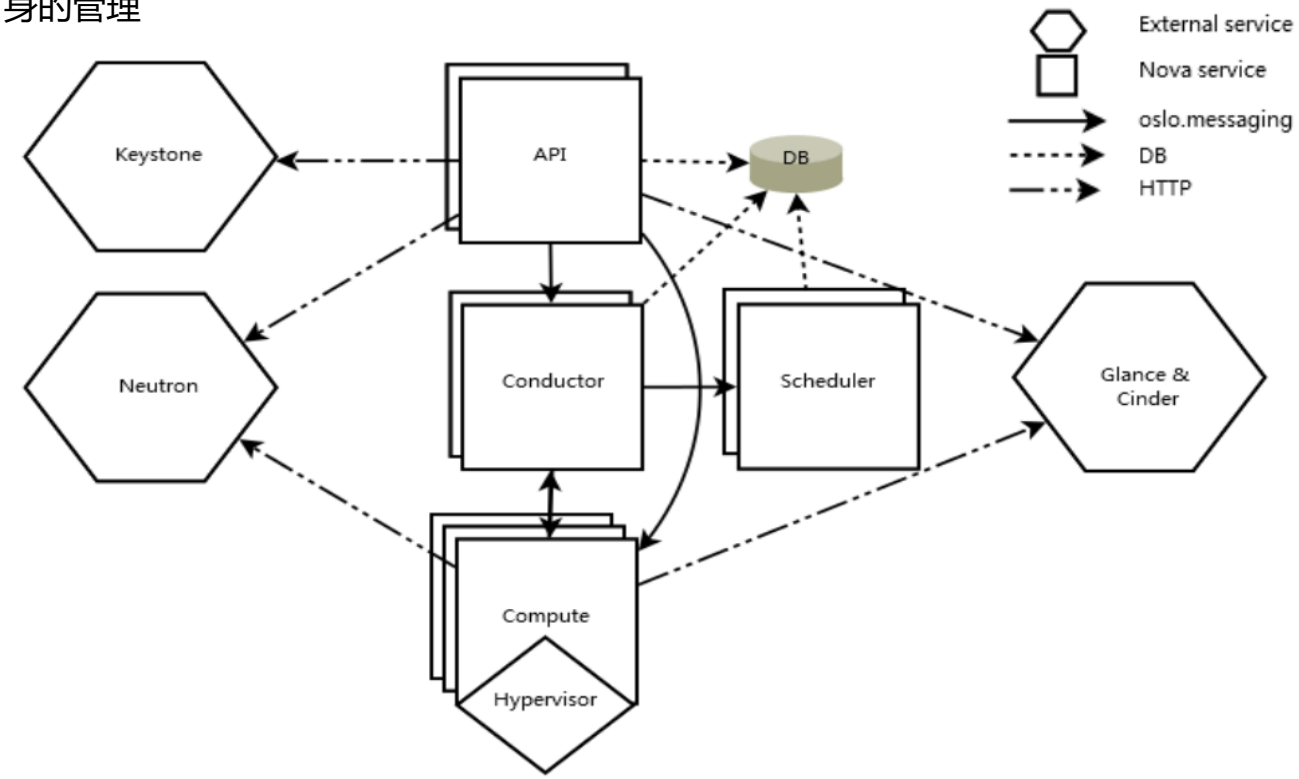
云控制器——Keystone

- 身份认证和服务目录（所有其他服务需向Keystone注册API端点）
- 由多个Provider组成
  - 身份提供者Identity Provider：服务用户、管理员用户、终端用户
  - 资源提供者resource Provider：project、domain
  - 认证提供者Authorization Provider：用户和用户组和他们角色之间的关系
  - 令牌提供者Token Provider：访问openstack服务需要一个有效的令牌，通过身份提供者的认证后获得令牌
  - 目录提供者Catalog Provider：维护服务和API端点的目录
  - 策略提供者Policy Provider：策略由多条规则组成，每条规则定义了哪些用户和角色被允许访问哪些资源
- 高级特性
  - 联邦keystone
    - 使用外部身份提供者提供的身份服务访问服务提供商可用资源的机制
  - Fernet令牌
    - 解决原生令牌提供者带来的性能问题



云控制器——Nova管理组件及API

- Nova是Openstack中提供计算资源服务的项目，是OpenStack最核心的项目
- 负责：
  - 虚拟机生命周期管理
  - 其他计算资源的生命周期管理
- 不负责：
  - 承载虚拟机的物理机自身的管理
  - 全面的系统状态监控



云控制器——Nova管理组件及API

- 运行在云控制器上的组件
  - Nova-api
    - 云控制器中的编排引擎；通过API将消息写入数据库和消息队列向其他守护进程传递消息
  - Nova-conductor
    - 代表计算节点上nova-compute执行数据库操作，提供数据库访问隔离；将来自计算节点请求并行化
  - Nova-scheduler
    - 专门的调度算法；确定虚拟机创建的最佳放置位置；支持过滤器检查计算节点上资源可用性，通过加权机制过滤出计算节点列表，再确定启动虚拟机的最佳位置；支持自定义度量标准和策略考量配置
    - 在生产环境下，配置单独的存储主机，其调度器运行在存储主机上
- Nova-compute则运行在计算节点上
- Nova-network则可以被Neutron替代

- API服务
  - 类似Nova-api，也将身份、镜像、网络和存储API都放在云控制器上运行

云控制器——其他服务

- 云控制器托管用于镜像管理的Glance服务
  - Glance API提供外部REST接口，用于查询虚拟机镜像及相关元数据
  - Glance registry将镜像元数据存储在数据库中，并利用存储后端实际存储镜像
    - 例如使用Swift对象存储作为镜像存储后端
- 网络服务采用类似Nova的部署概念
  - 网络API驻留在云控制器中，独立配置网络服务器节点
- 仪表盘服务
  - 运行在Apache Web服务器后端，可以将其运行在可以访问API端点的单独节点上，降低云控制器负载
- 计量服务
  - 跟踪资源使用情况，计算资源利用率
  - 多种使用目的：计费、容量规划、按需求和吞吐量的虚拟基础架构自动扩展等

目录

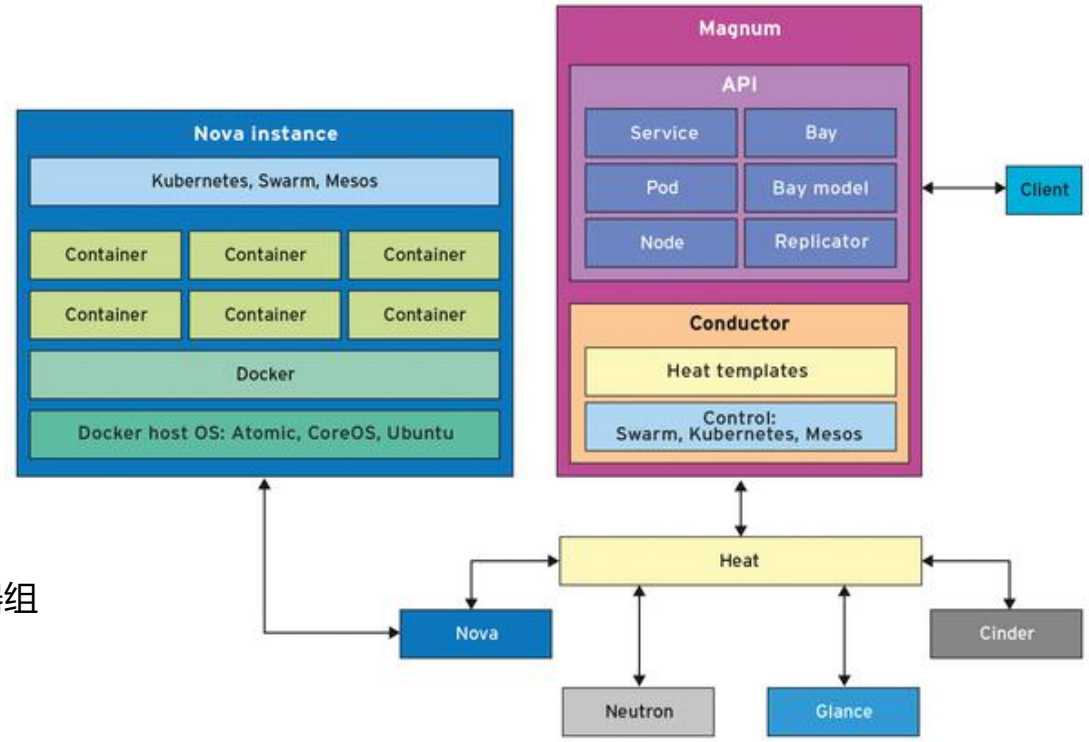
计算节点——Hypervisor

- 计算节点运行nova-compute服务，负责启动和终止虚拟机，通过消息总线监听虚拟机相关请求

- Hypervisor是计算节点的核心，又称VMM
  - Openstack中虚拟机管理程序能够支持各种VMM以及Docker

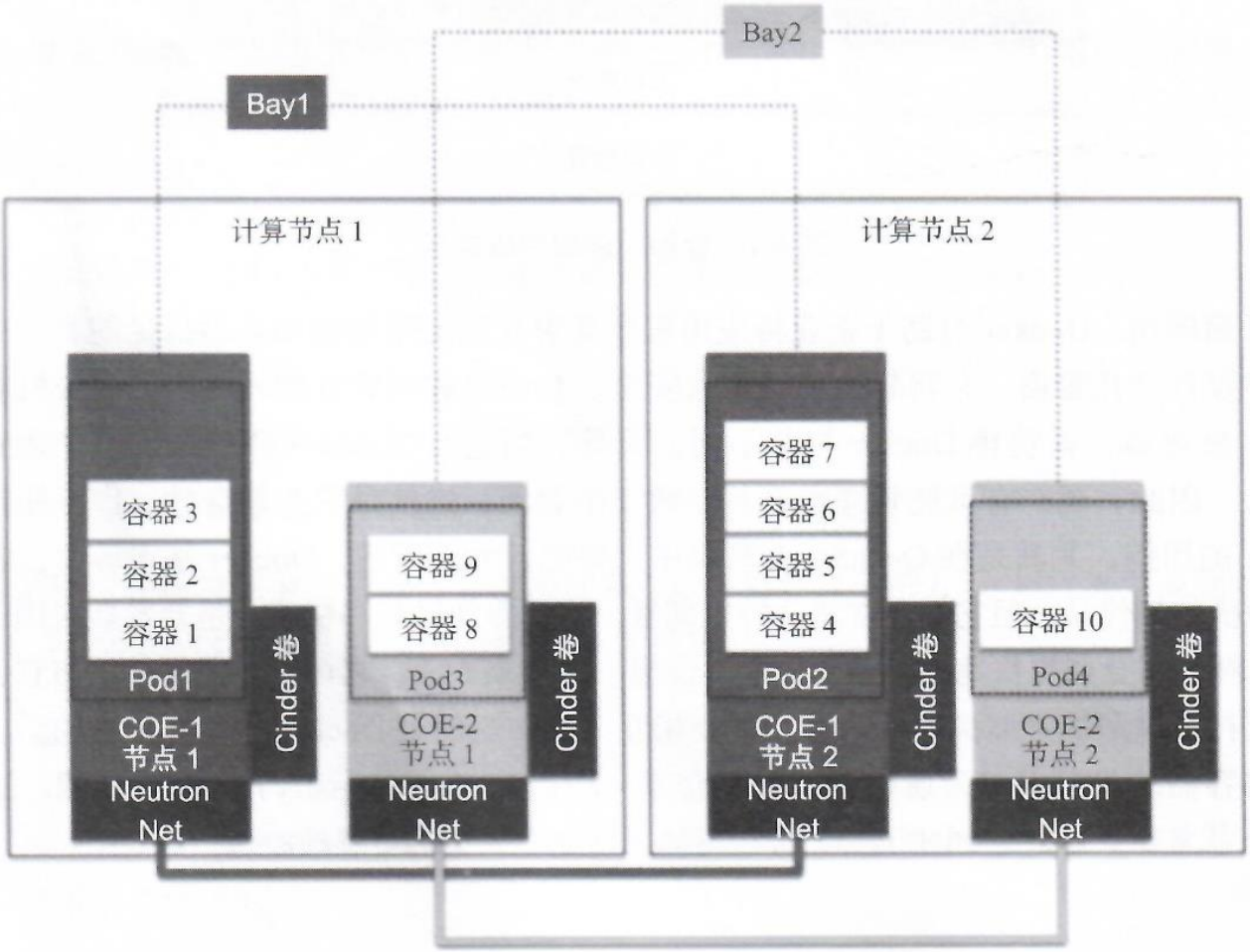
- 虚拟化技术
- Docker容器

- Nova Docker
  - 类似虚拟机实例启动和管理容器生命周期
  - 然而一个应用或服务往往需要容器组构建
- Magnum
  - 使用容器编排引擎(COE)管理连接着的容器组
  - COE节点部署为Nova实例
  - Heat用于编排COE虚拟基础架构



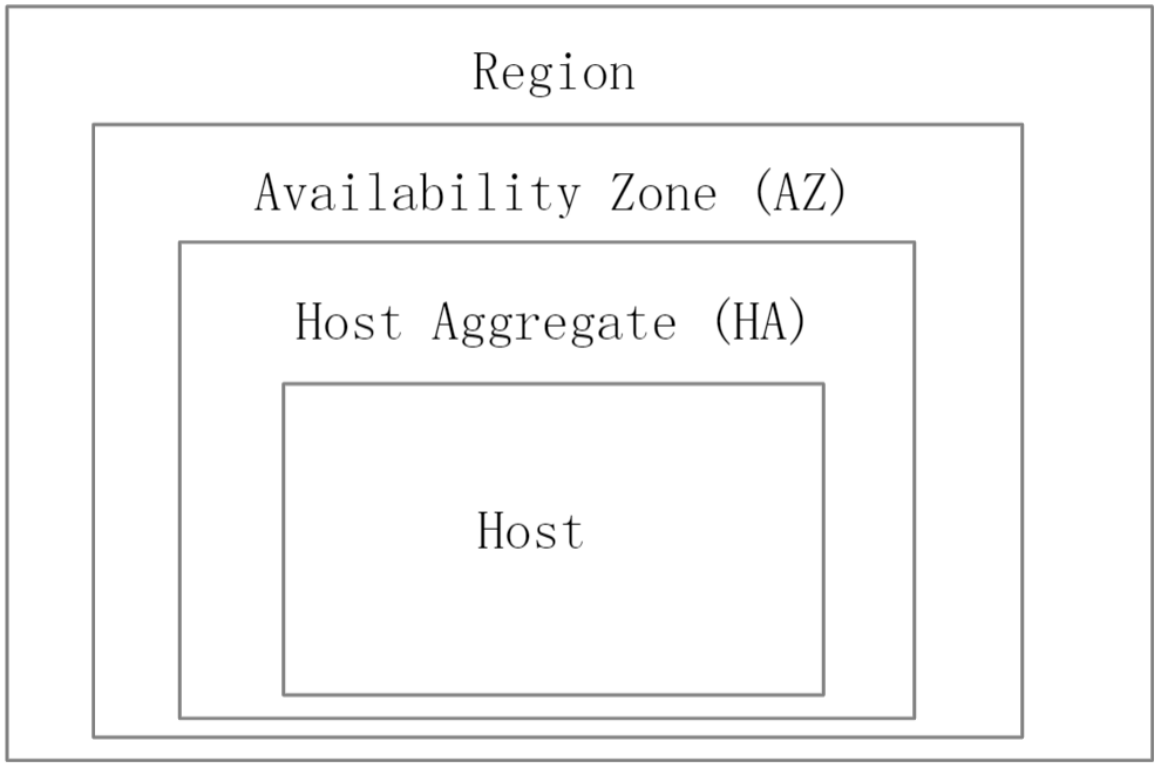
计算节点——Magnum

- Bay是一组运行COE软件的节点，表示一个容器集群
- Pod是COE最基本的部署调度单元，逻辑上对应一个服务实例，运行一组容器，必须有一个作为网络路由



计算节点——Nova相关概念（1）

- Nova中对主机有一个层级划分——高于一台物理主机



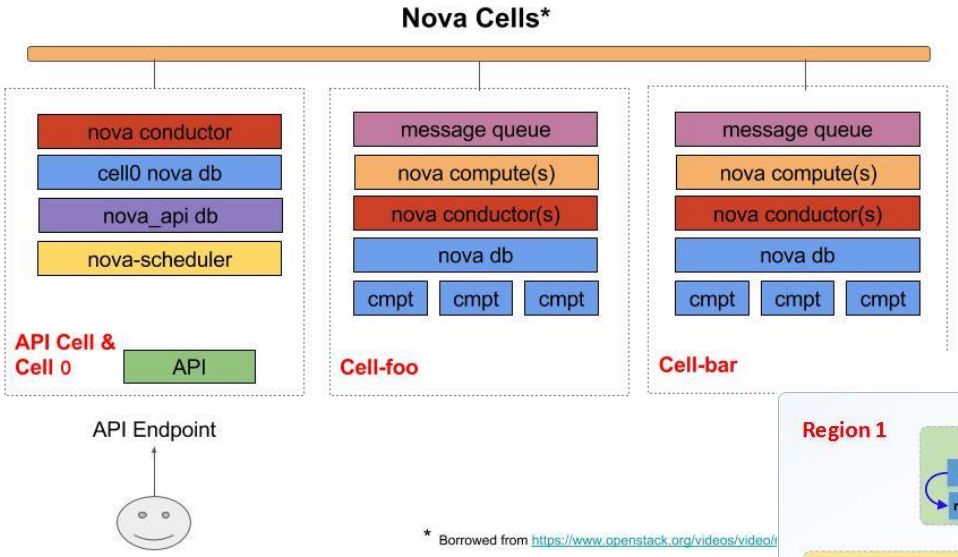
- Region：地理区域
- Availability Zone：对某个地理区域的用户而言的可用区域
- Host 类型将物理主机放到一起Aggregate：按照组成区域
- Host：具体的物理主机

计算节点——Nova相关概念（2）

名称	简介	说明
Server/Instance	虚拟机	Nova管理提供的云服务资源。Nova中最重要的数据对象。
Server metadata	虚拟机元数据	通常用于为虚拟机附加必要描述信息。key/value。
Flavor	虚拟机规格模板	用于定义一种虚拟机类型，如一种具有2个VCPU、4GB内存、40GB本地存储空间的虚拟机。Flavor由系统管理员创建，供普通用户在创建虚拟机时使用。
Quota	资源配额	用于指定租户最多能够使用的逻辑资源上限。
Hypervisor / node	节点	对于KVM、Xen等虚拟化技术，一个node即对应于一个物理主机。对于vCenter，一个node对应于一个cluster。
Host	主机	对于KVM、Xen等虚拟化技术，一个host即对应于一个物理主机，同时对应于一个node。对于vCenter，一个host对应于一套vCenter部署。
Host Aggregate	主机聚合	一个HA内包含若干host。一个HA内的物理主机通常具有相同的CPU型号等物理资源特性。
Server Group	虚拟机亲和性/反亲和组	同一个亲和性组的虚拟机，在创建时会被调度到相同的物理主机上。同一个反亲和性组的虚拟机，在创建时会被调度到不同的物理主机上。
Service	Nova各个服务	管理nova相关服务的状态，包括nova-compute，nova-conductor，nova-scheduler，nova-novncproxy，nova-consoleauth,nova-console

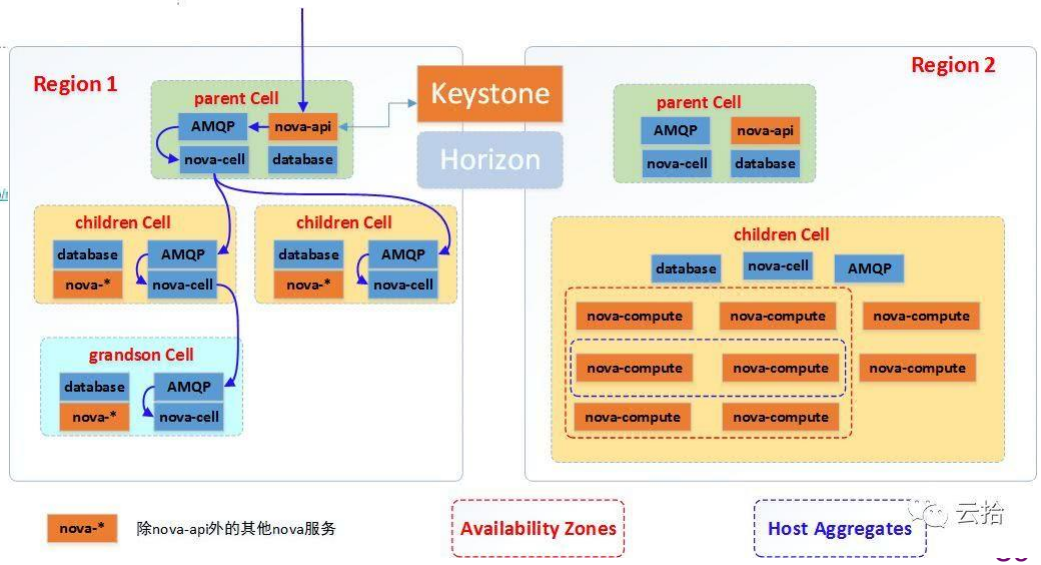
计算节点——Nova单元

- 按照计算节点为最小单位的管理视角——Nova单元
- 所有计算节点都需要与消息总线和数据库服务器通信，随着计算节点增多，使得消息队列和数据库过载，因此产生Nova计算单元

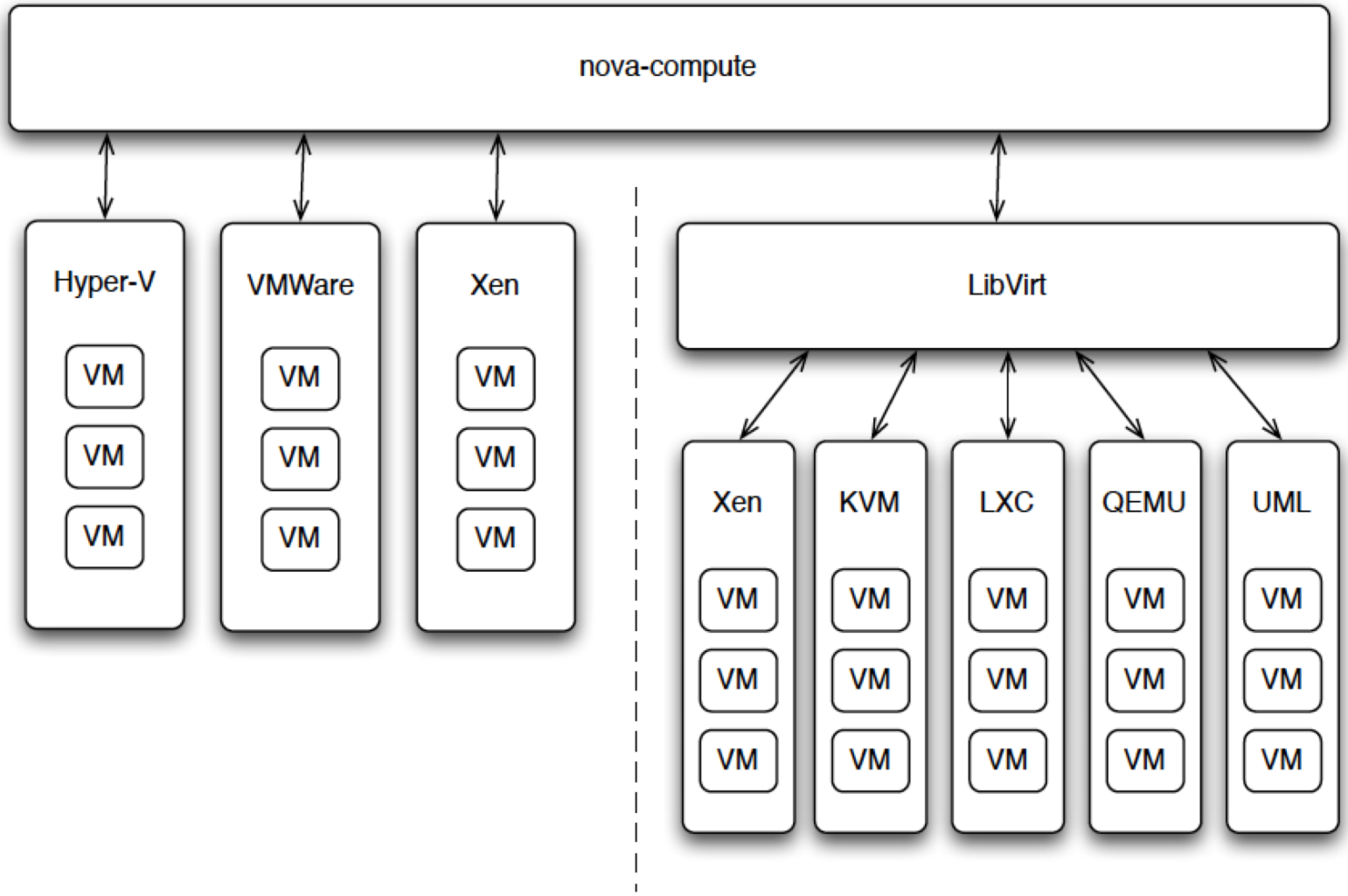


* Borrowed from https://www.openstack.org/videos/video/

- 由多个计算节点组成一个单元
- API单元：作为树根，使用基于消息队列的RPC调用和计算节点交互
- 计算单元：每个单元有自己的消息队列和数据库



计算节点——使用多种Hypervisor



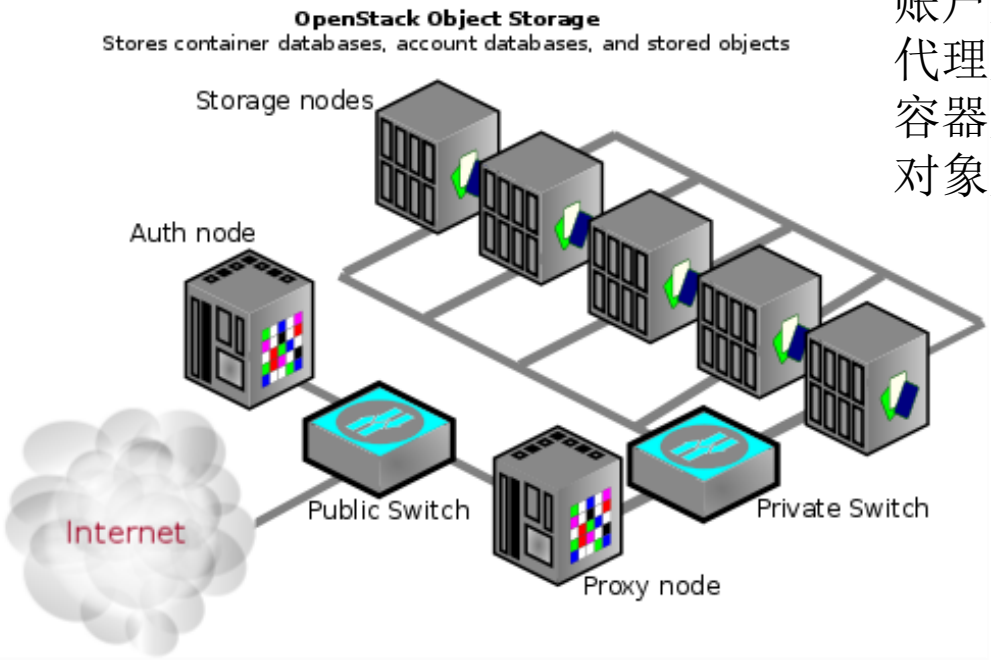
计算节点——Nova操作

分组	说明
虚拟机生命周期管理	虚拟机创建、删除、启动、关机、重启、重建、规格更改、暂停、解除暂停、挂起、继续、迁移、在线迁移、锁定、解锁、疏散，拯救，解拯救，搁置，删除搁置， 恢复搁置，备份，虚拟机导出镜像、列表、详细信息、信息查询更改，密码修改
卷和快照管理操作	本质上是对Cinder API的封装。卷创建、删除、列表、详细信息查询。快照创建、删除、列表、详细信息查询
虚拟机卷操作	虚拟机挂卷、虚拟机卸卷、虚拟机挂卷列表、虚拟机挂卷详细信息查询
虚拟网络操作	本质上是对Neutron API的封装。虚拟网络创建、删除、列表、详细信息查询
虚拟机虚拟网卡操作	虚拟机挂载网卡、虚拟机卸载网卡、虚拟机网卡列表。
虚拟机镜像的操作	本质上是对Glance API的封装，支持镜像的创建、删除、列表、详细信息查询。
虚拟机HA	自研虚拟机的可靠性操作。可手动触发(FS5.1)。
其他资源其他操作	Flavor，主机组，keypairs，quota等

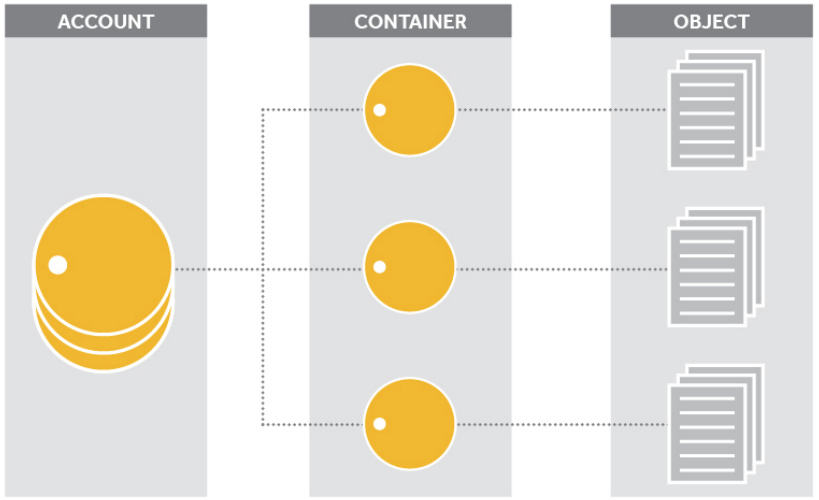
目录

存储节点——Swift

- 数据与其副本作为二进制大对象存储在对象存储服务器上
- 对象存储没有文件存储的层次结构，对象存储在扁平的命名空间——一种特殊的文件系统
- 使用REST或SOAP等API访问对象存储，而不是文件协议
- 不适用于高性能要求以及经常更改的机构化数据，例如数据库



账户服务器：租户、身份认证
代理服务器：运行API
容器服务器：“桶”，一人多桶，一桶多对象
对象服务器：一个对象只能在一个桶中

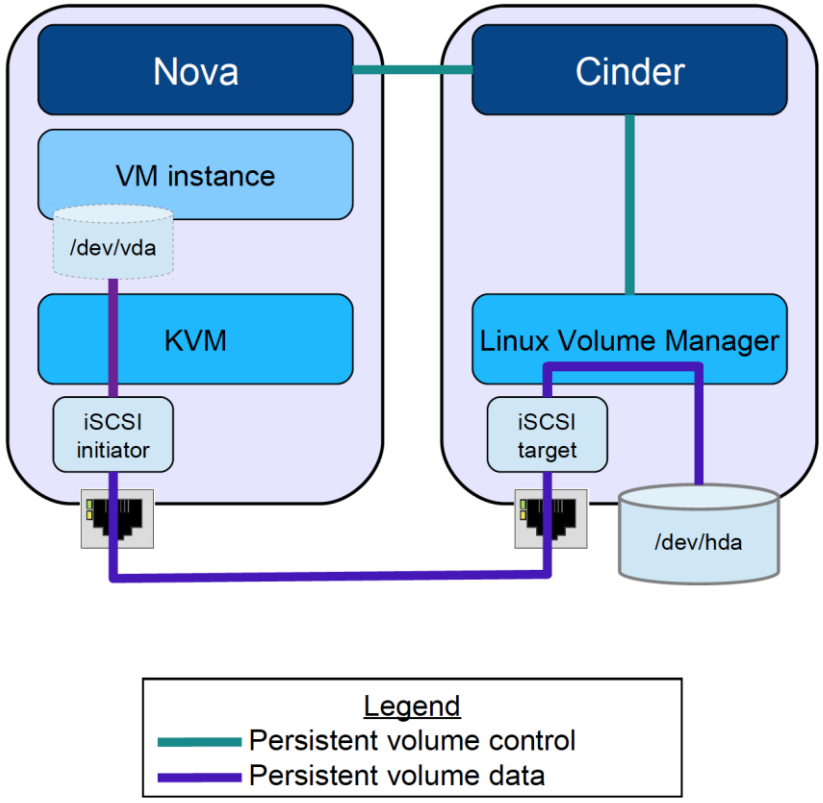


Swift使用环（ring）将账户、容器和对象映射到集群上的物理位置。账户环、容器环、对象环。

存储节点——Cinder

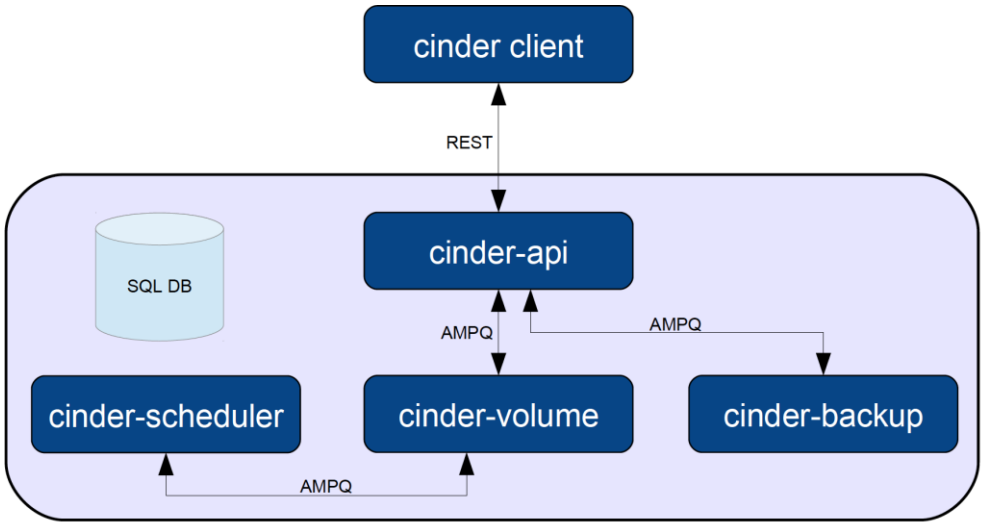
- 为云平台提供统一接口，按需分配的，持久化的块存储服务
- 核心功能是对卷的管理，允许对卷、卷的类型、卷的快照、卷备份进行操作
- 为后端不同的存储设备提供了统一的接口，不同的块设备服务厂商在Cinder 中实现其驱动支持以与OpenStack进行整合

计算节点和Cinder的网络连接

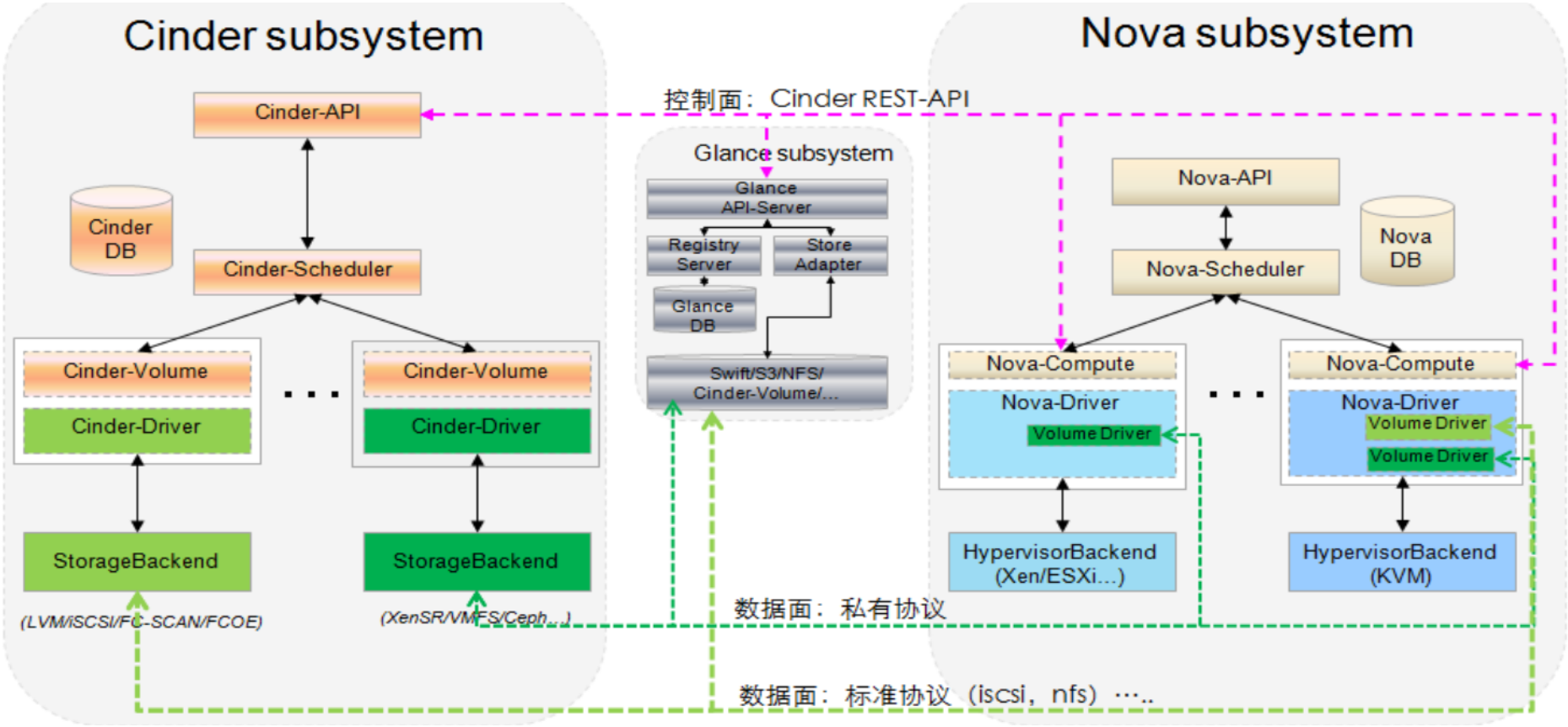


块存储作为虚拟机硬盘使用时，必须先分区、创建文件系统，再挂载到虚拟机文件系统层次结构中。

Cinder内部架构



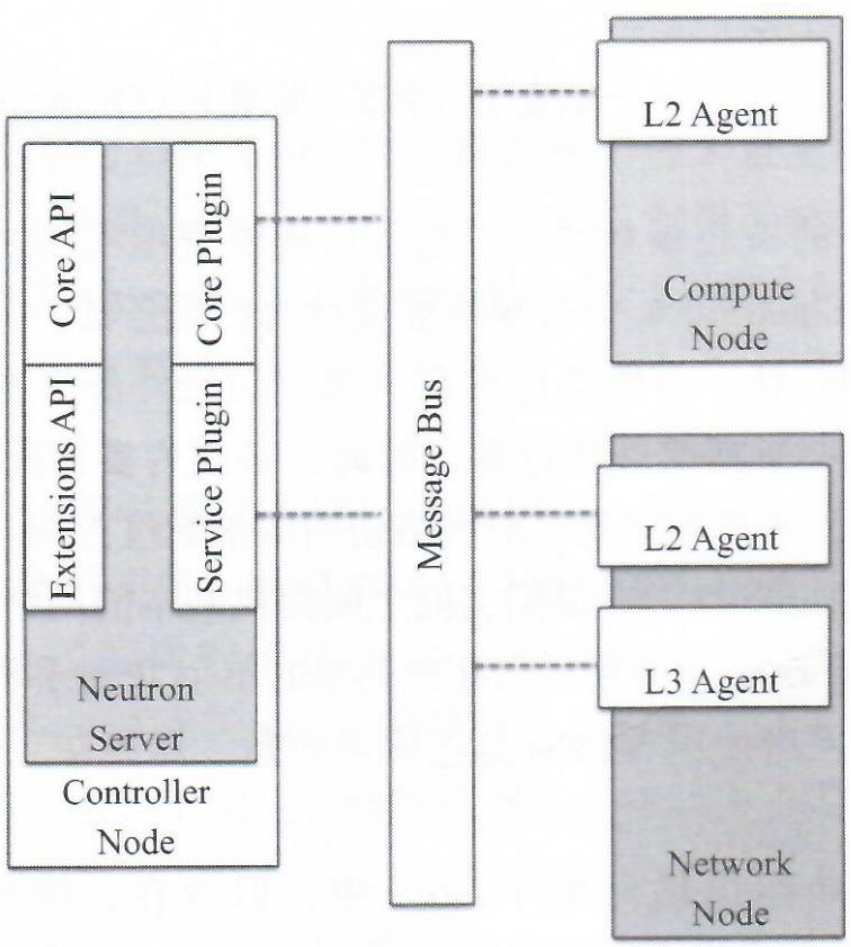
存储节点——Cinder


Cinder subsystem
Nova subsystem
控制面：Cinder REST-API
Cinder-API
Cinder DB
Cinder-Scheduler
Cinder-Volume
Cinder-Driver
Cinder-Volume
Cinder-Driver
StorageBackend
StorageBackend
(LVM/iSCSI/FC-SCAN/FCOE)
(XenSR/VMFS/Ceph...)
Glance subsystem
Glance API-Server
Registry Server
Store Adapter
Glance DB
Swift/S3/NFS/ Cinder-Volume/...
Nova-API
Nova DB
Nova-Scheduler
Nova-Compute
Nova-Driver
Volume Driver
Nova-Compute
Nova-Driver
Volume Driver
Volume Driver
HypervisorBackend (Xen/ESXi...)
HypervisorBackend (KVM)
数据面：私有协议
数据面：标准协议（iscsi，nfs）…..

目录

网络节点——Neutron

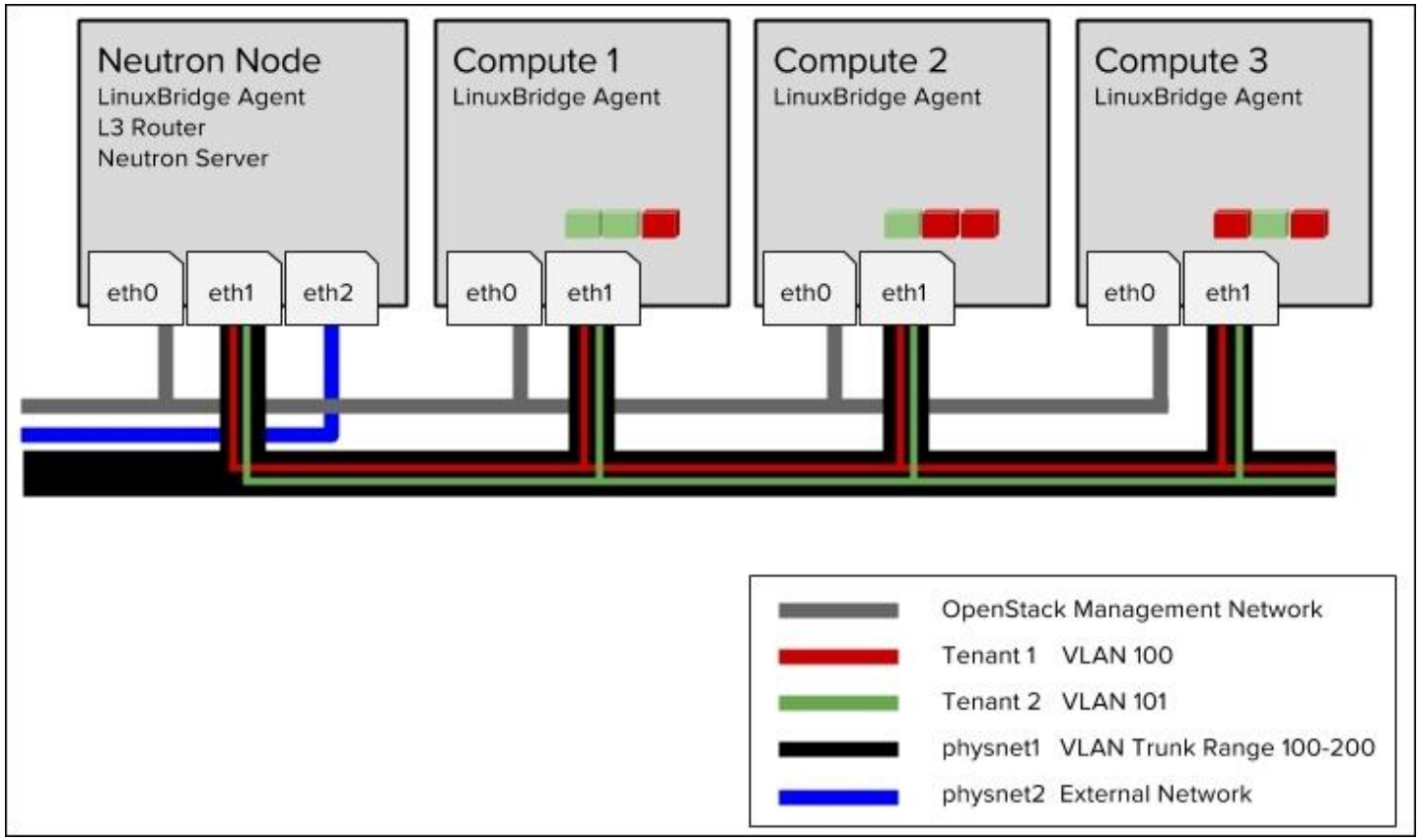
- 使得运营商能够构建和管理具有所有必要元素的完整网络拓扑
  - 网络、子网、路由器、负载均衡器、防火墙和端口
- API服务器：运行在云控制器上
  - 基于插件的架构
  - API服务将请求转发到特定插件，插件通过代理和设备交互或者控制资源
  - 提供了基于开源技术的插件和代理参考实现
  - 网络节点提供资源实现网络服务：路由、防火墙、负载均衡器和VPN
- 核心插件：创建和管理网络、端口和子网
- 服务插件：实现高阶网络服务，路由、防火墙等
- 代理：部署在计算和网络节点上，通过消息总线上的RPC和服务器交互
  - L2代理：运行在计算和网络基点上，将虚拟机和网络设备连接到二层网络上
  - DHCP代理、L3代理和VPN代理



网络节点——虚拟网络VLAN实现

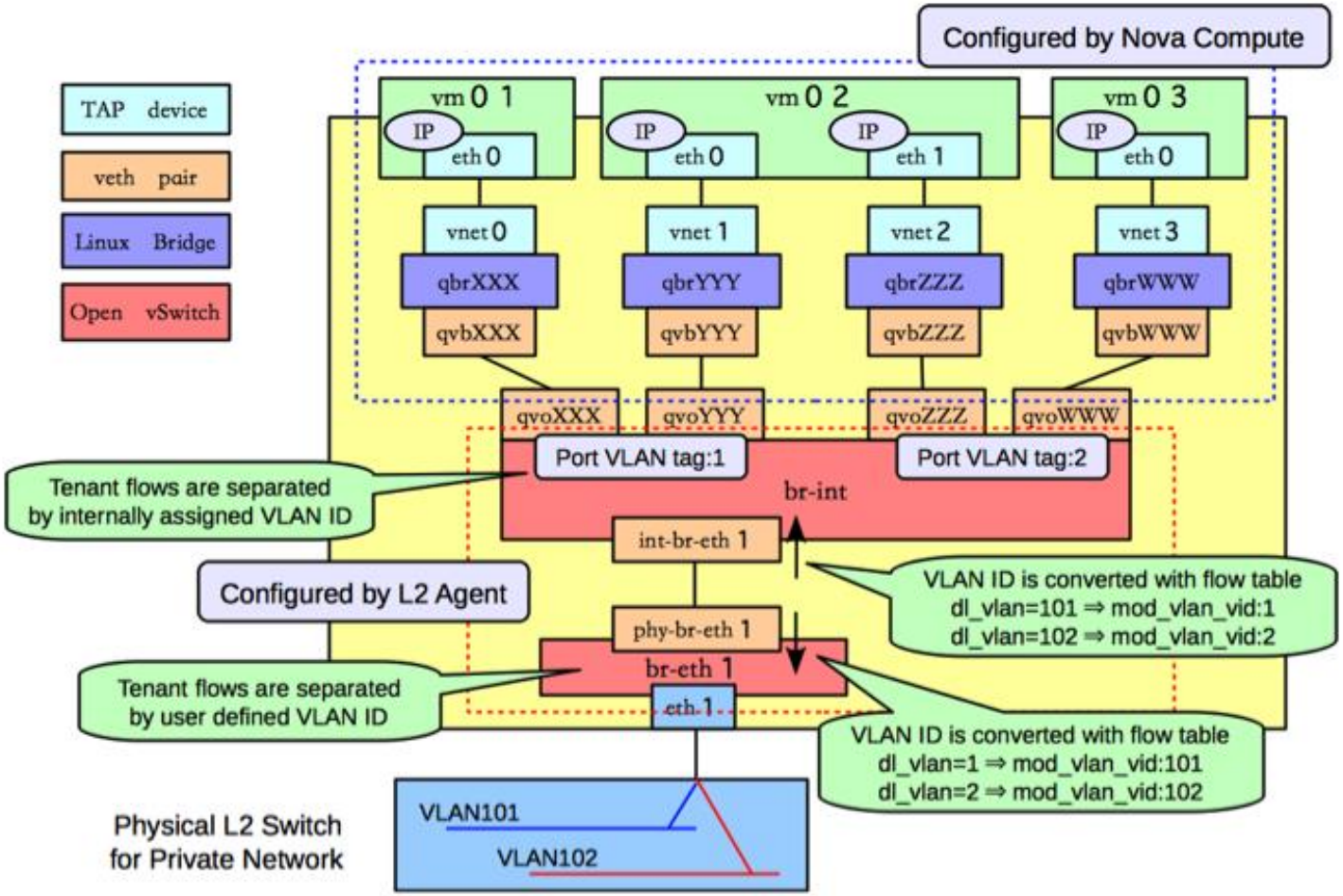
- 核心插件处理虚拟网络和端口的创建
- 虚拟网络的创建方式
  - VLAN方式：编排连接计算节点和网络节点的物理交换机路径
  - 隧道网络：将二层数据包封装在IP数据包中；IP网络是已经封装的网络流量传输者

物理网络连接
和
虚拟网络连接
的关系



网络节点——虚拟网络VLAN实现

- 计算节点上的L2代理



总结

- OpenStack——云平台构建、管理和监控系统
- 体系架构
  - 云控制器
  - 计算节点
  - 存储节点
  - 网络节点
- 如何部署——DevOps理念
- 推荐自行了解的其他关键问题
  - 对象存储、文件共享如何架构？如何与云控制节点、计算节点、网络节点交互？
  - 虚拟机如何访问外网？
  - 外网如何访问虚拟机？

其他内容：
OpenStack的高可用和容错机制
OpenStack集群监控和故障排查
OpenStack ELK日志处理系统
OpenStack基准测试和性能调优

实验：部署OpenStack

谢 谢

费彝民楼917

南京大學
NANJING UNIVERSITY