OpenStack存储管理

前言

- OpenStack提供多种类型的存储服务，用户可以根据业务需求，自由选择存储服务。
- 本章节重点介绍OpenStack中的块存储服务Cinder，简单介绍对象存储服务Swift。
- 本章节分为两个部分：理论和实验
  - 理论部分主要讲解Cinder作用、架构、工作原理和流程，Swift作用和架构。
  - 实验部分重点锻炼学员Cinder日常运维操作，帮助学员理论联系实际，真正掌握Cinder 。

目标

- 学完本课程后，您将能够：
  - 描述OpenStack不同存储类型
  - 描述Cinder作用
  - 描述Cinder架构
  - 描述Cinder工作流程
  - 描述Swift作用
  - 描述Swift架构
  - 具备Cinder日常运维能力

目录

1. **OpenStack存储概述**
2. 块存储Cinder
3. 对象存储Swift

OpenStack有哪些存储类型?

- OpenStack中的存储可以分为两类：

Ephemeral Storage，临时存储	Persistent Storage，持久性存储
• 如果只部署了Nova服务，则默认分配给虚拟机的磁盘是临时的，当虚拟机终止后，存储空间也会被释放。 • 默认情况下，临时存储以文件形式放置在计算节点的本地磁盘上。	• 持久化存储设备的生命周期独立于任何其他系统设备或资源，存储的数据一直可用，无论虚拟机是否运行。 • 当虚拟机终止后，持久性存储上的数据仍然可用。

- 目前OpenStack支持三种类型的持久性存储：块存储、对象存储和文件系统存储。

OpenStack持久化存储简介

块存储

操作对象是磁盘，直接挂载到主机，一般用于主机的直接存储空间和数据库应用，DAS和SAN都可以提供块存储。

Cinder

对象存储

操作对象是对象(object)，一个对象名称就是一个域名地址，可以直接通过REST API的方式访问对象。

Swift

文件存储

操作对象是文件和文件夹，在存储系统上增加了文件系统，再通过NFS或CIFS协议进行访问。

Manila

因Manila目前使用较少，本章节只重点介绍Cinder和Swift。

OpenStack存储类型对比

	用途	访问方式	访问客户端	管理服务	数据生命周期	存储设备容量	典型使用案例
临时存储	运行操作系统和提供启动空间	通过文件系统访问	虚拟机	Nova	虚拟机终止	管理员配置的Flavor指定容量	虚拟机中第一块磁盘10GB，第二块磁盘20GB
块存储	为虚拟机添加额外的持久化存储	块设备被分区、格式化后挂载访问（例如 /dev/vdc）	虚拟机	Cinder	被用户删除	用户创建时指定	1 TB磁盘
对象存储	存储海量数据，包括虚拟机映像	REST API	任何客户端	Swift	被用户删除	可用物理存储空间和数据副本数量	10s TB级数据集存储
共享文件系统存储	为虚拟机添加额外的持久化存储	共享文件系统存储被分区、格式化后挂载访问（例如 /dev/vdc）	虚拟机	Manila	被用户删除	•用户创建时指定 •扩容时指定 •用户配额指定 •管理员指定容量	NFS

讨论：如何选择不同OpenStack存储?

- 请讨论或思考如下场景中，如何选择合适的OpenStack存储?

场景一

测试OpenStack虚拟机发放功能，测试完即删除虚拟机

场景二

OpenStack生产环境中的虚拟机，需保证虚拟机数据长期保存

场景三

存放OpenStack中的Glance镜像文件

目录

1. OpenStack存储概述
2. **块存储Cinder**
   - Cinder简介
   - Cinder架构
   - Cinder组件详细讲解
   - Cinder典型工作流程
   - OpenStack动手实验： Cinder操作
3. 对象存储Swift

OpenStack块存储服务是什么?

CINDER

块存储服务
首次出现在OpenStack的“Folsom” 版本中。

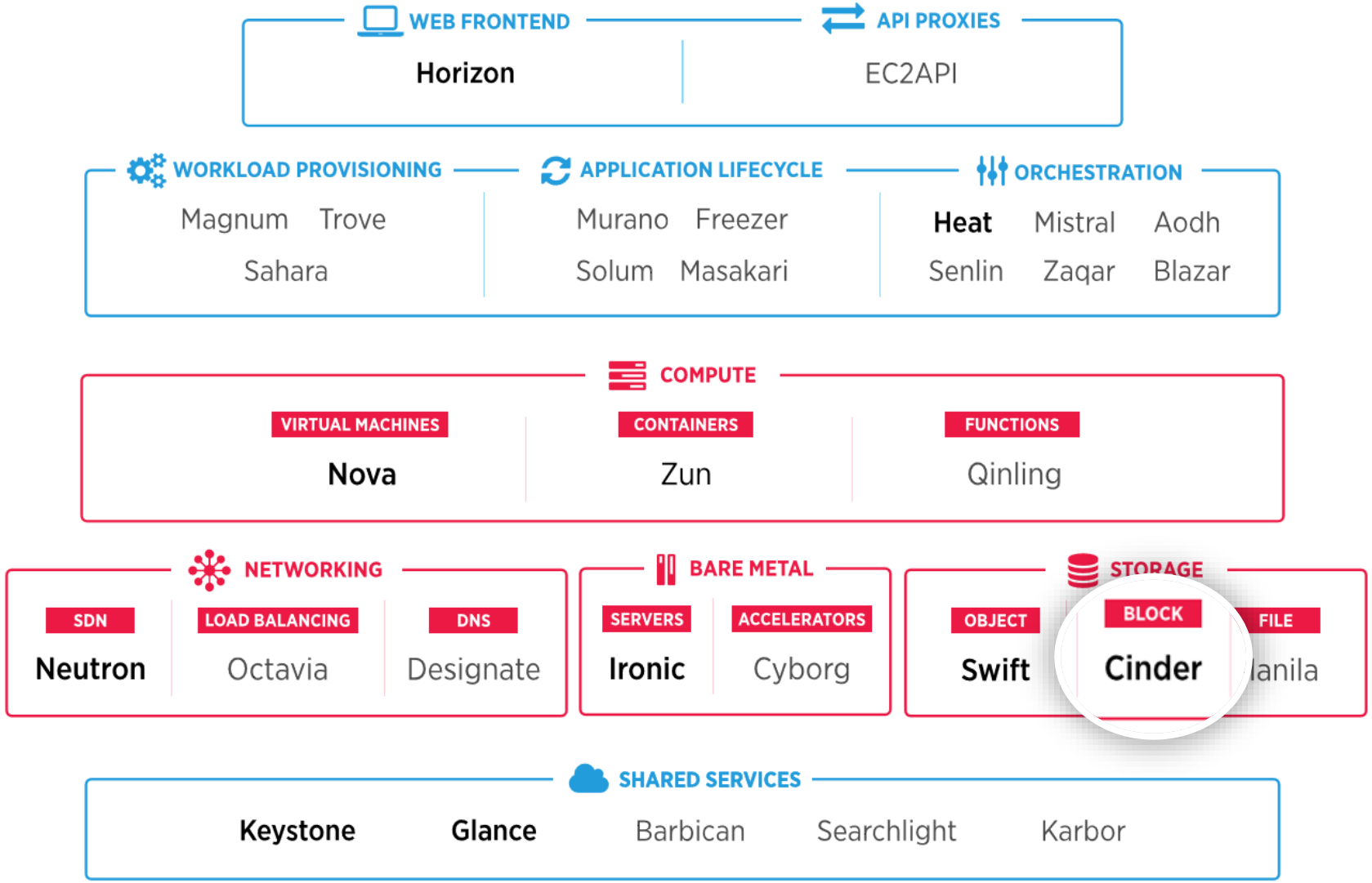
简介

Cinder提供块存储服务，为虚拟机实例提供持久化存储。
Cinder调用不同存储接口驱动，将存储设备转化成块存储池，用户无需了解存储实际部署的位置或设备类型。

依赖的OpenStack服务

Keystone

Cinder在OpenStack中的位置和作用

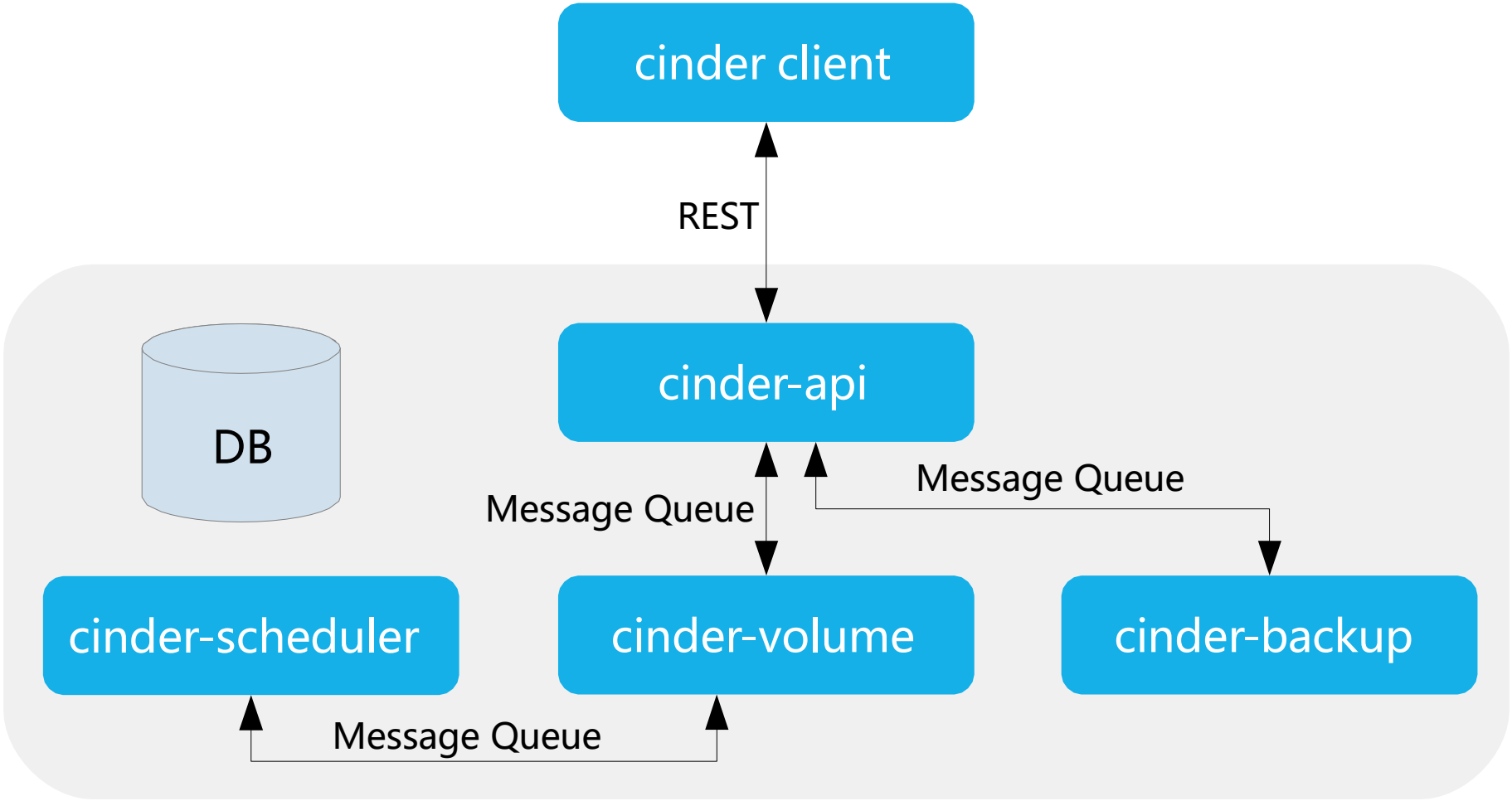


source: openstack.org

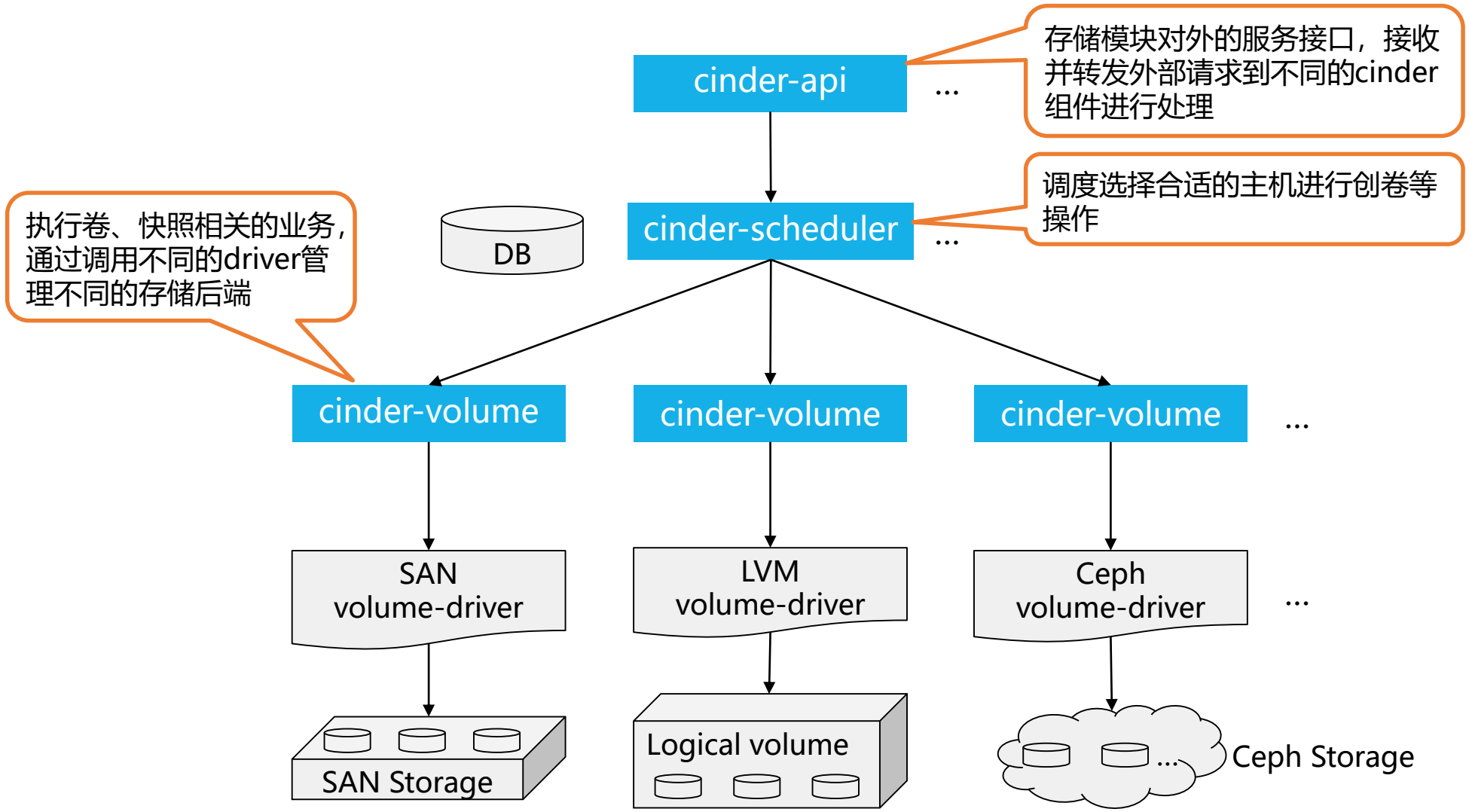
目录

1. OpenStack存储概述
2. **块存储Cinder**
   - Cinder简介
   - Cinder架构
   - Cinder组件详细讲解
   - Cinder典型工作流程
   - OpenStack动手实验： Cinder操作
3. 对象存储Swift

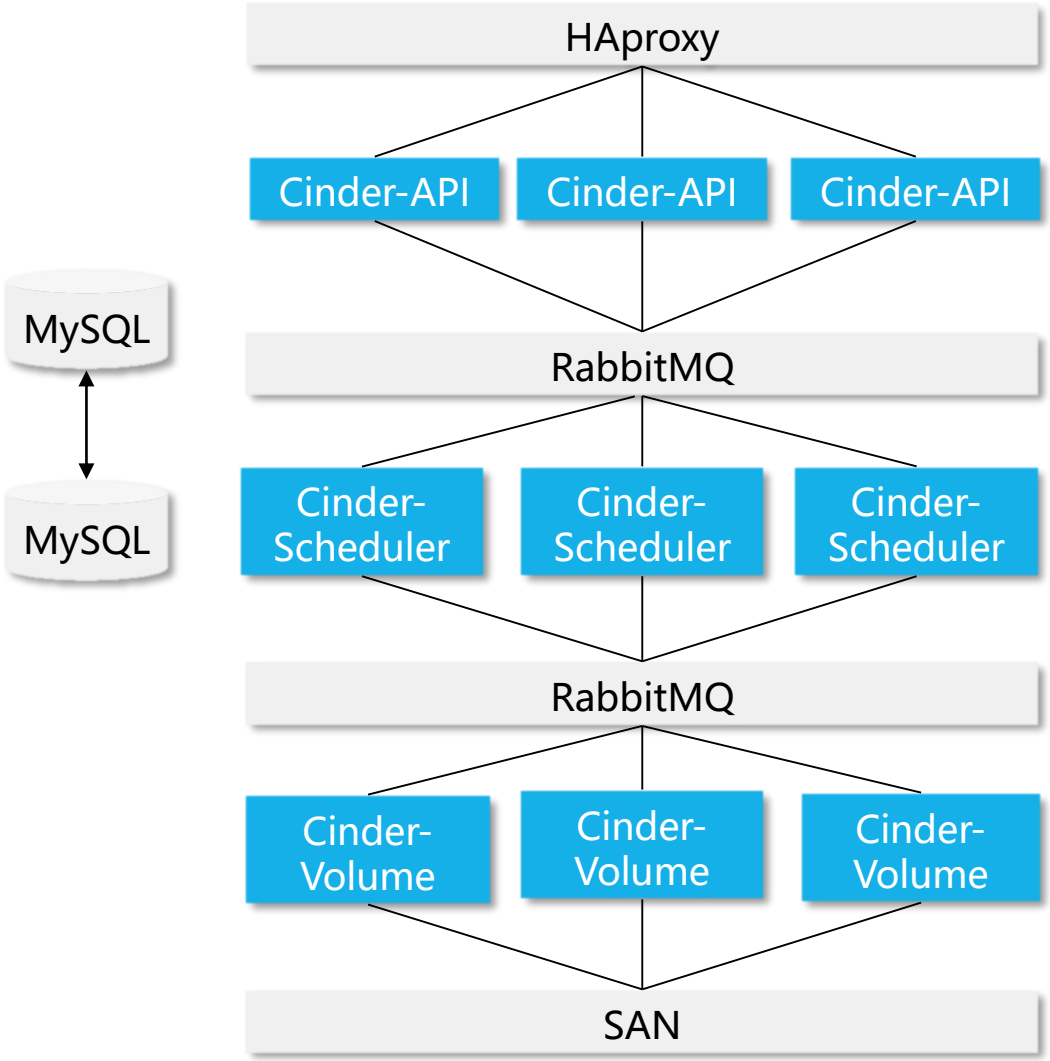
Cinder架构



Cinder架构说明



Cinder架构部署：以SAN存储为例



- Cinder-api，Cinder-Scheduler，Cinder-Volume可以选择部署到一个节点上，也可以分别部署。
- API采用AA模式，Haproxy作为LB，分发请求到多个Cinder API。
- Scheduer也采用AA模式，有rabbitmq以负载均衡模式向3个节点分发任务，并同时从rabbitqmq收取Cinder volume上报的能力信息，调度时，scheduler通过在DB中预留资源从而保证数据一致性。
- Cinder Volume也采用AA模式，同时上报同一个backend容量和能力信息，并同时接受请求进行处理。
- RabbitMQ，支持主备或集群。
- MySQL，支持主备或集群。

目录

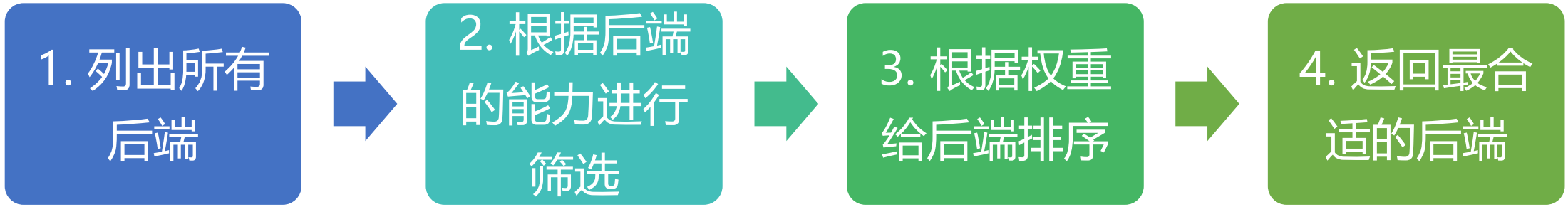
1. OpenStack存储概述
2. **块存储Cinder**
   - Cinder简介
   - Cinder架构
   - Cinder组件详细讲解
   - Cinder典型工作流程
   - OpenStack动手实验： Cinder操作
3. 对象存储Swift

Cinder组件 - API

- Cinder API对外提供REST API，对操作需求进行解析，并调用处理方法：
  - 卷create/delete/list/show
  - 快照create/delete/list/show
  - 卷attach/detach (Nova调用)
  - 其他:
    - Volume types
    - Quotas
    - Backups

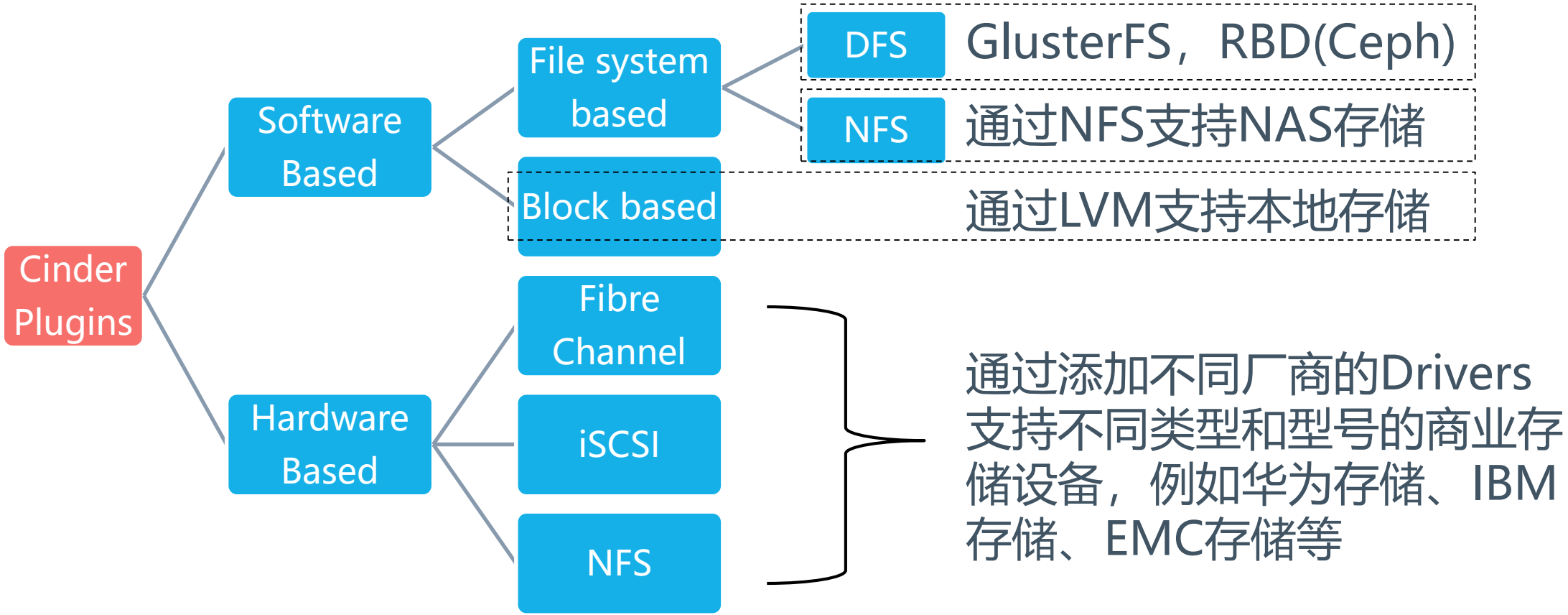
Cinder组件 - Scheduler

- Cinder scheduler负责收集后端上报的容量、能力信息，根据设定的算法完成卷到指定cinder-volume的调度。
- Cinder scheduler通过过滤和称权，筛选出合适的后端：



Cinder组件 - Volume

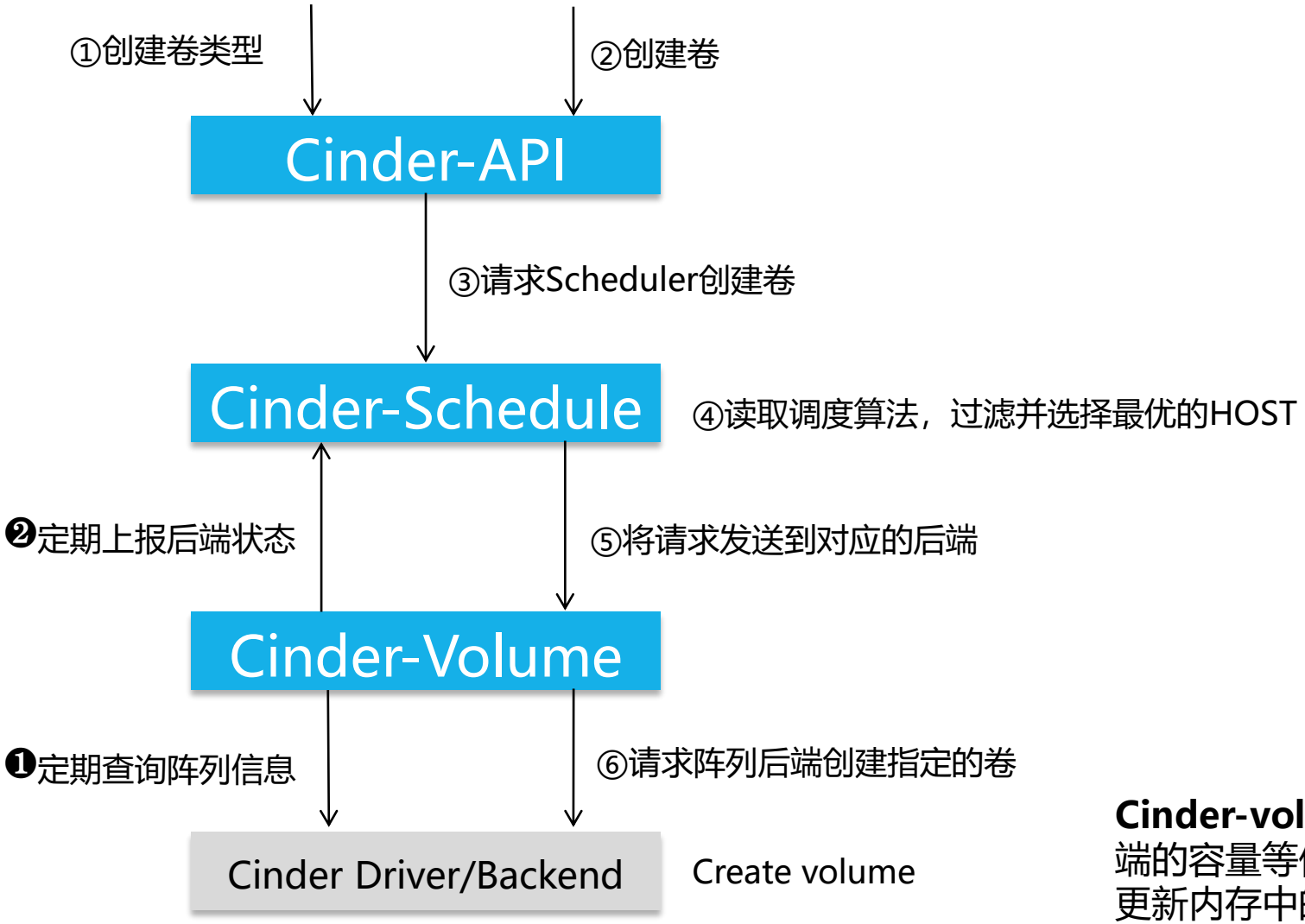
- Cinder volume多节点部署，使用不同的配置文件、接入不同的后端设备，由各存储厂商插入Driver代码与设备交互，完成设备容量和能力信息收集、卷操作等。



目录

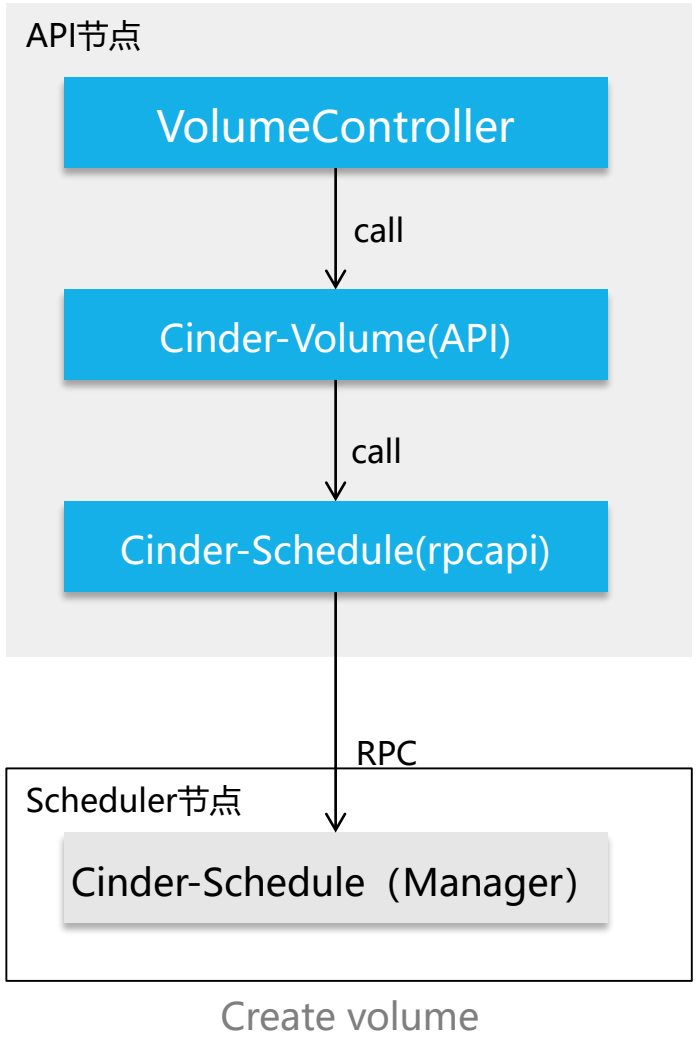
1. OpenStack存储概述
2. **块存储Cinder**
   - Cinder简介
   - Cinder架构
   - Cinder组件详细讲解
   - Cinder典型工作流程
   - OpenStack动手实验： Cinder操作
3. 对象存储Swift

Cinder创建卷流程



Cinder-volume: 会定期收集底层后端的容量等信息，并通知Scheduler更新内存中的Backend信息。

Cinder创建卷流程 - Cinder API

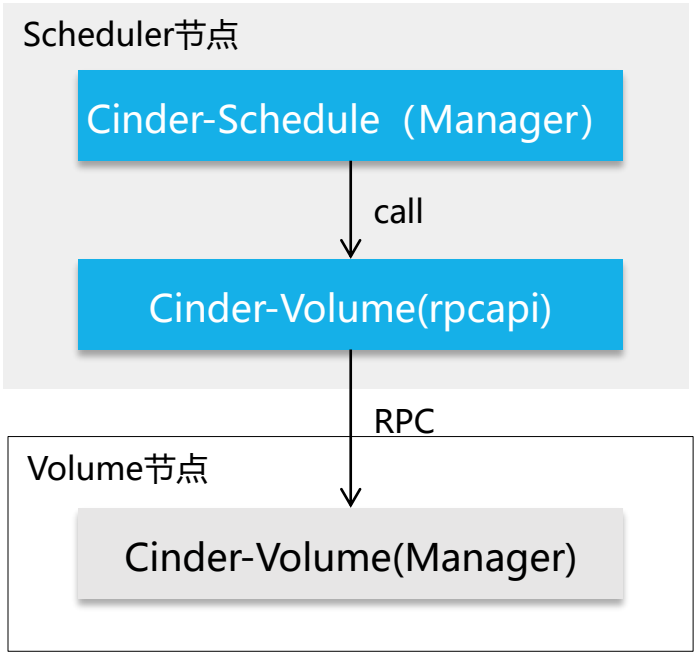


Create volume

Cinder API

- 检查参数合法性(用户输入，权限，资源是否存在等)。
- 准备创建的参数字典，预留和提交配额。
- 在数据库中创建对应的数据记录。
- 通过消息队列将请求和参数发送到Scheduler

Cinder创建卷流程 - Cinder Scheduler

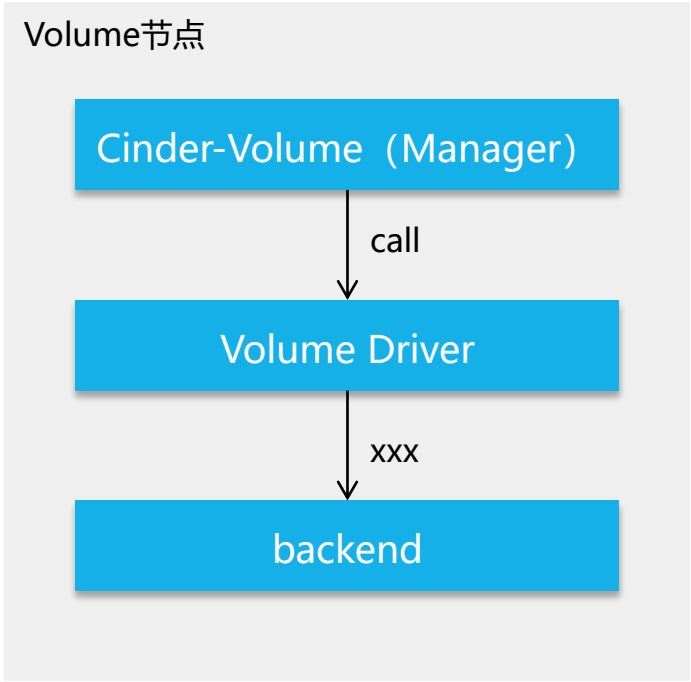


Create volume

Cinder Scheduler服务

- 提取接收到的请求参数
- 通过配置的filter和输入参数对后端进行过滤
  - Availability_zone_filter
  - Capacity_filter
  - Capabilities_filter
  - Affinity_filter(SameBackendFilter/DifferentBackendFilter)
  - ……
- Weigher计算后端进行权重
  - CapacityWeigher/AllocatedCapacityWeigher
  - ChanceWeigher
  - GoodnessWeigher
  - …….
- 选取最优的Backend并通过消息队列将请求发送到指定的后端

Cinder创建卷流程 - Cinder Volume

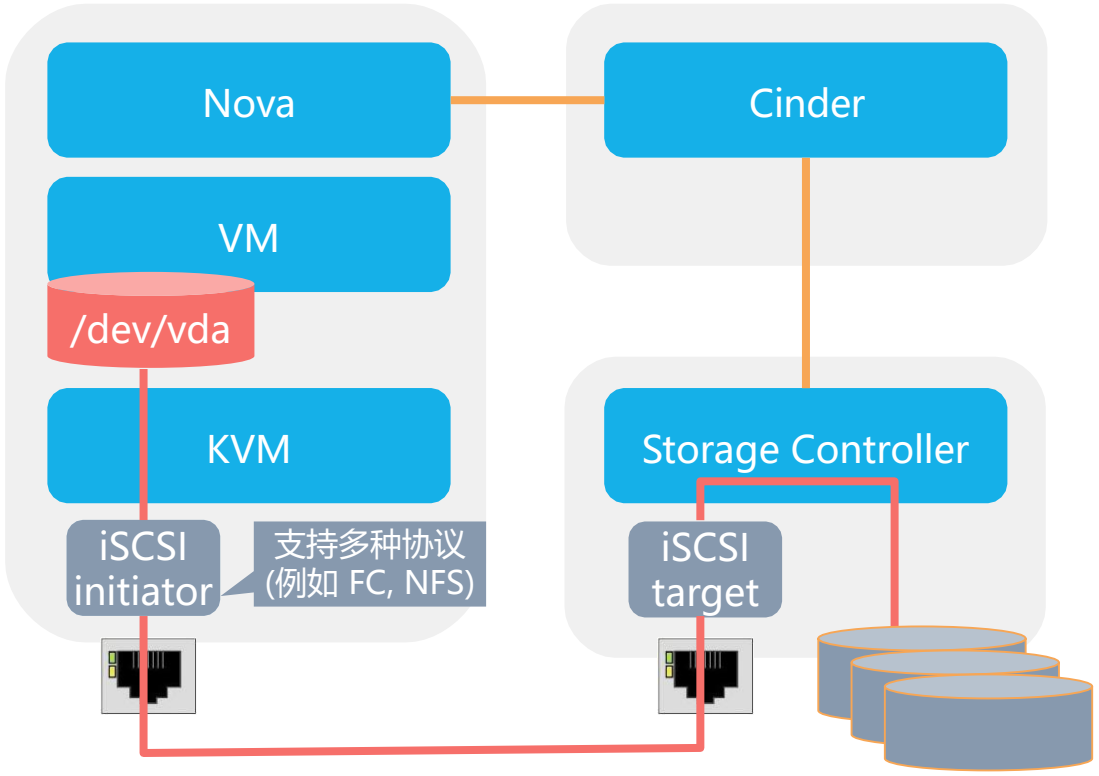


Create volume

Cinder Volume服务

- 提取接收到的请求参数
- 调用对应的Driver在后端创建实际的卷
- 使用Driver返回的模型更新数据库中的记录

Cinder挂载卷流程



挂卷流程： 挂卷是通过Nova和Cinder的配合最终将远端的卷连接到虚拟机所在的Host节点上，并最终通过虚拟机管理程序映射到内部的虚拟机中。

目录

1. OpenStack存储概述
2. **块存储Cinder**
   - Cinder简介
   - Cinder架构
   - Cinder组件详细讲解
   - Cinder典型工作流程
   - OpenStack动手实验： Cinder操作
3. 对象存储Swift

Cinder主要操作

功能分类	功能	功能分类	功能
卷操作	create	快照操作	snapshot-create
	delete		snapshot-delete
	show		snapshot-list
	rename		snapshot-rename
	upload-to-image		snapshot-reset-state
	extend		snapshot-show
	force-delete		snapshot-metadata
	list		snapshot-metadata-show
	migrate		snapshot-metadata-update-all
	reset-state	备份操作	backup-create
	rate-limits		backup-delete
	retype		backup-list
	set-bootable		backup-restore
	manage		backup-show
	unmanage		backup-export
	metadata		backup-export

Cinder主要操作主要三个资源:

Volume:
块设备卷，提供创建，删除，扩容，挂载/卸载等功能。

Snapshot:
针对于块设备卷的快照创建，删除, 回滚等功能。

Backup:
提供对块设备卷的备份，恢复能力。

动手实验：Cinder操作

- 命令help
- 卷类型管理
- 卷QoS管理
- 卷管理

目录

1. OpenStack存储概述
2. 块存储Cinder
3. **对象存储Swift**
   - Swift简介
   - Swift架构

对象存储服务是什么?

SWIFT

对象存储服务
首次出现在OpenStack的“Austin” 版本中。

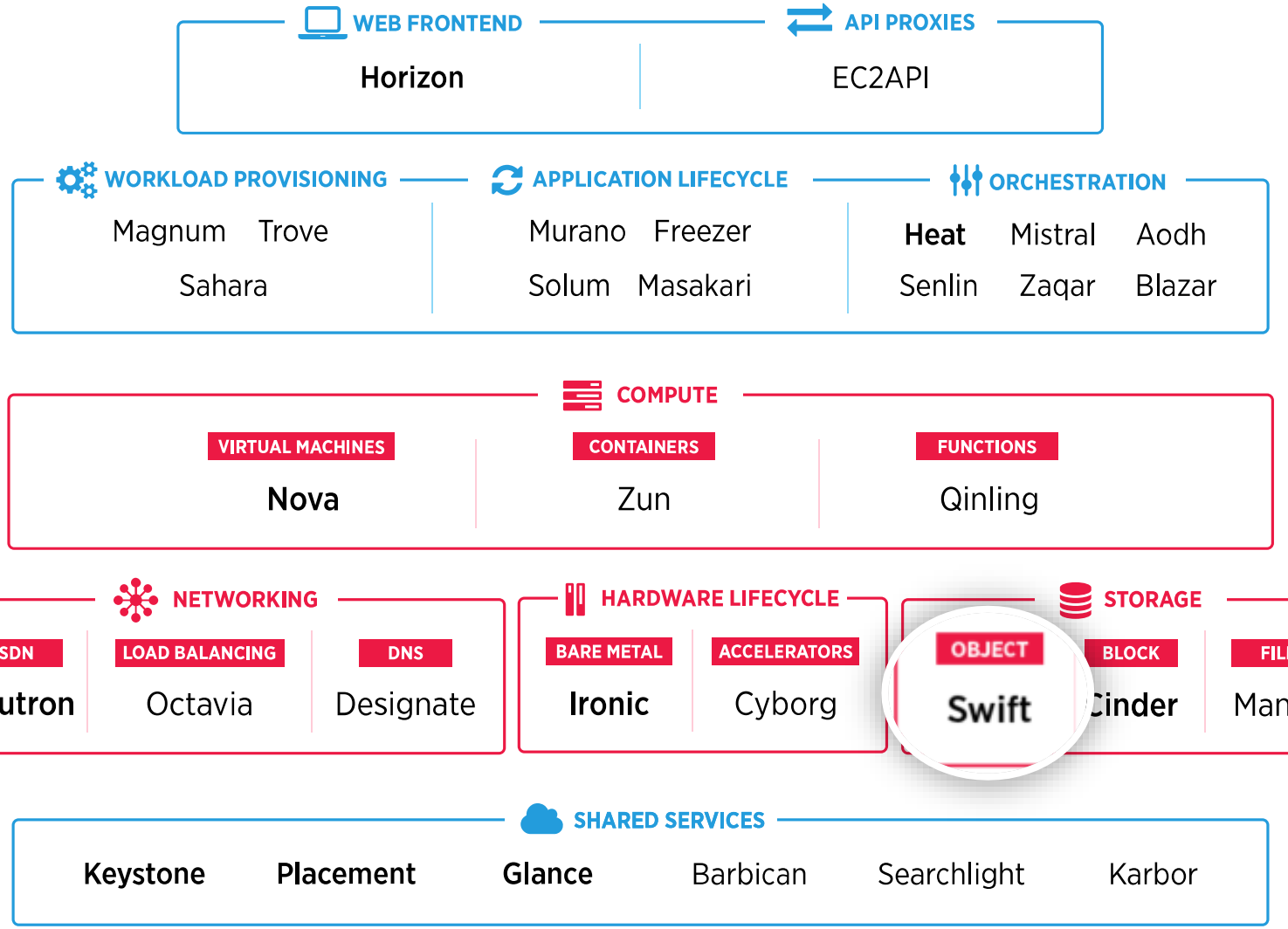
简介

Swift提供高度可用、分布式、最终一致的对象存储服务。
Swift可以高效、安全且廉价地存储大量数据。
Swift非常适合存储需要弹性扩展的非结构化数据。

依赖的OpenStack服务

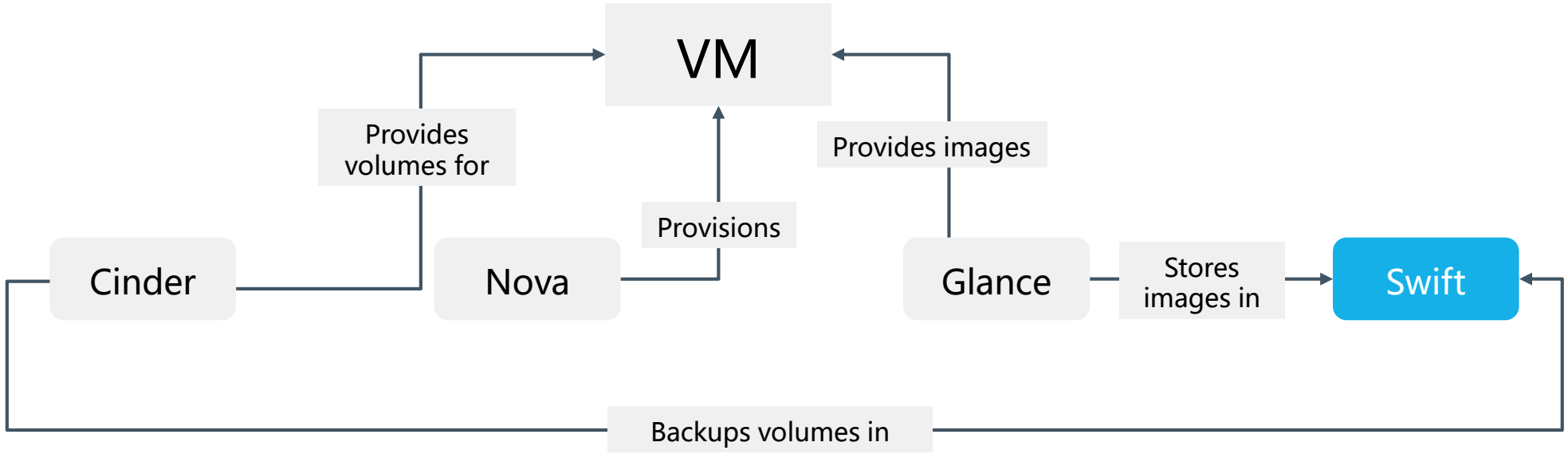
为其他OpenStack服务提供对象存储服务。

Swift在OpenStack中的位置



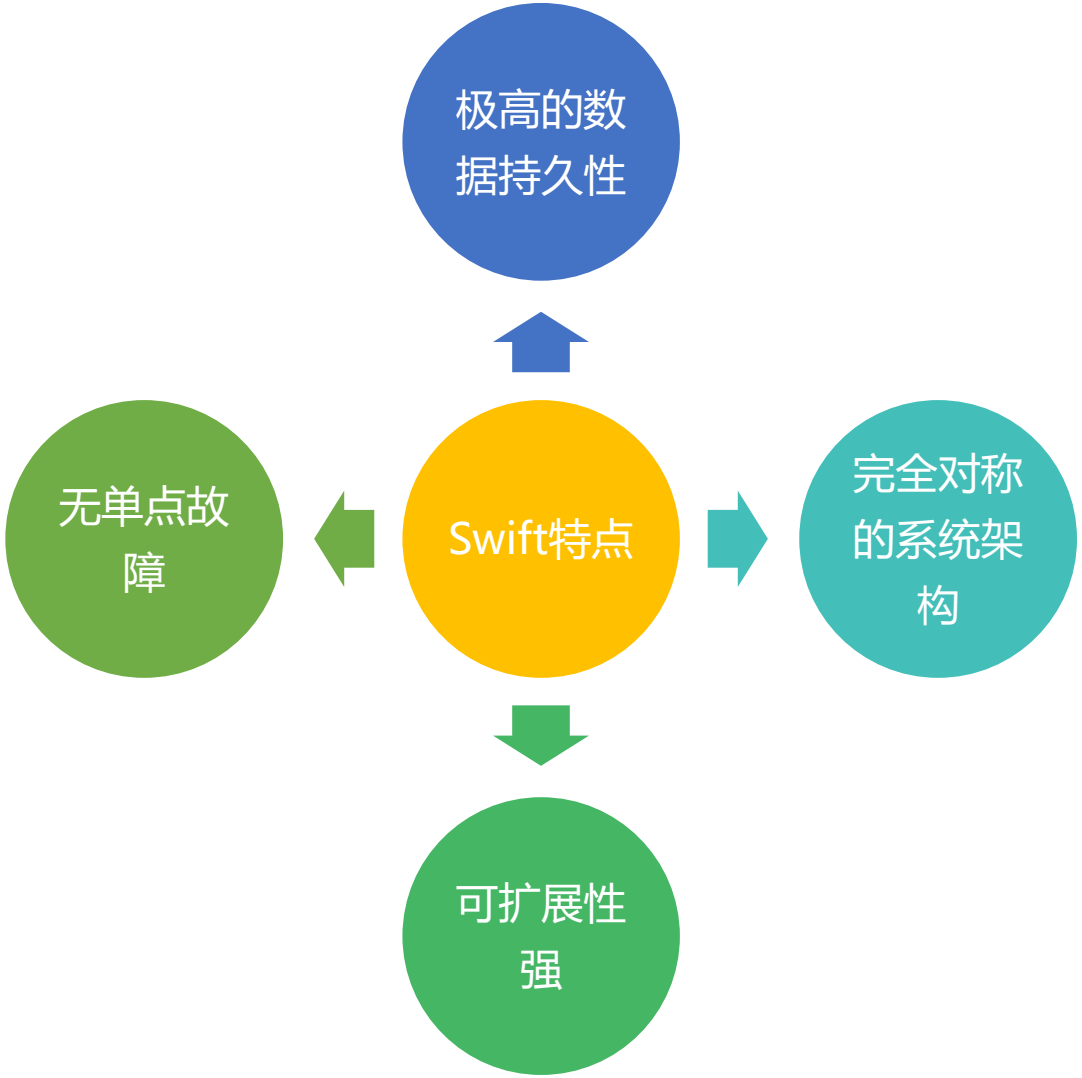
source: openstack.org

Swift在OpenStack中的作用



Swift并不是文件系统或者实时的数据存储系统，它称为对象存储，用于永久类型的静态数据的长期存储，这些数据可以检索、调整，必要时进行更新。
最适合存储的数据类型的例子是虚拟机镜像、图片存储、邮件存储和存档备份。
因为没有中心单元或主控结点，Swift提供了更强的扩展性、冗余和持久性。

Swift特点



Swift应用场景

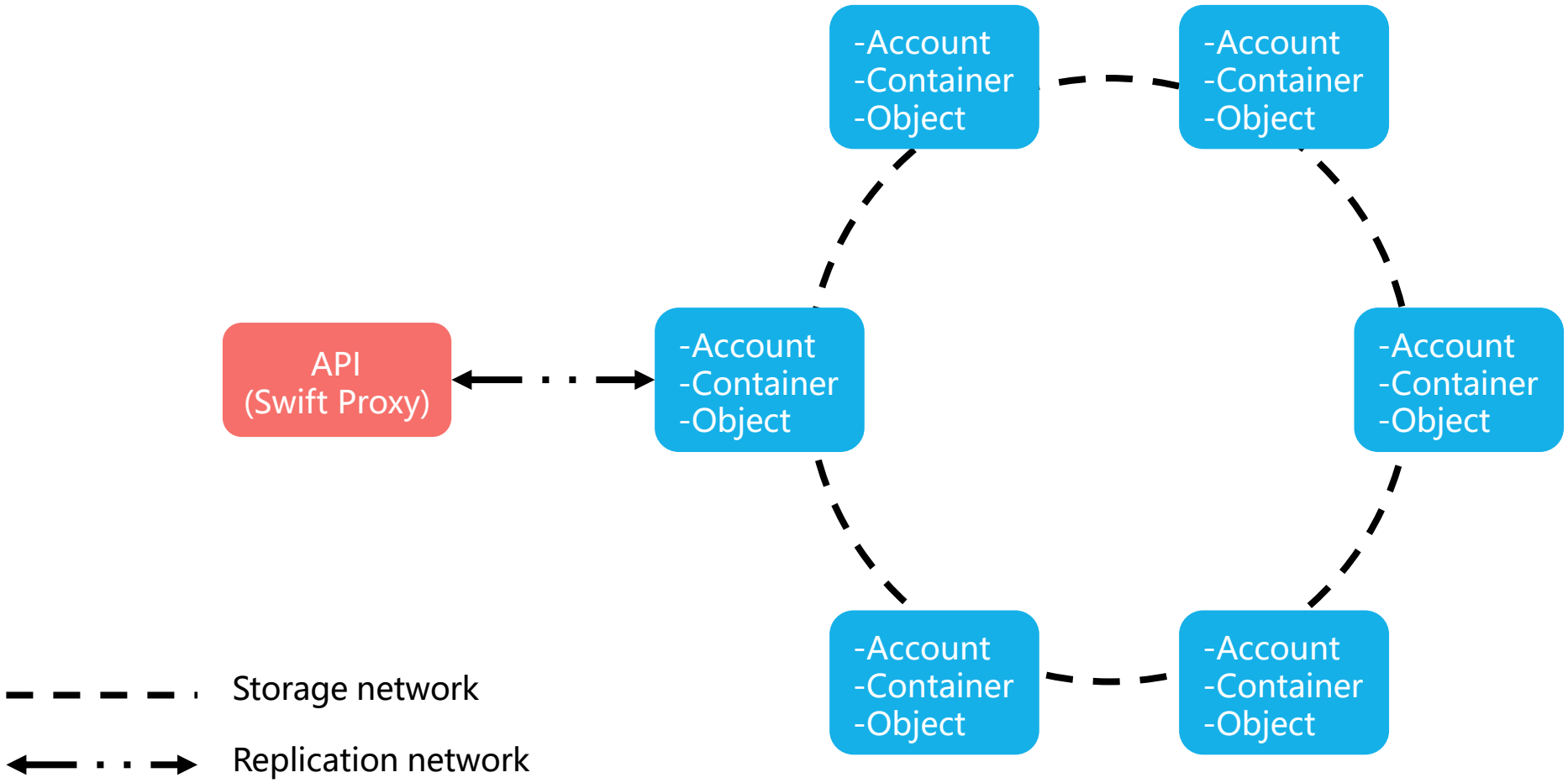
- 镜像存储后端
  - 在OpenStack中与镜像服务Glance结合，为其存储镜像文件。
- 静态数据存储
  - 由于Swift的扩展能力，适合存储日志文件和数据备份仓库。

目录

1. OpenStack存储概述
2. 块存储Cinder
3. **对象存储Swift**
   - Swift简介
   - Swift架构

对象存储服务的架构

- 完全对称、面向资源的分布式系统架构设计



Swift组件

Proxy Server

- 对外提供对象服务API，由于采用无状态的 REST 请求协议，可以进行横向扩展来均衡负载。

Account Server

- 提供账户元数据和统计信息，并维护所含容器列表的服务，每个账户的信息被存储在一个SQLite 数据库中。

Container Server

- 提供容器元数据和统计信息，并维护所含对象列表的服务，每个容器的信息也存储在一个SQLite 数据库中。

Swift组件

Object Server

- 提供对象元数据和内容服务，每个对象的内容会以文件的形式存储在文件系统中，元数据会作为文件属性来存储，建议采用支持扩展属性的 XFS 文件系统。

Replicator

- 检测本地分区副本和远程副本是否一致，发现不一致时会采用推式（Push）更新远程副本，并且确保被标记删除的对象从文件系统中移除。

Updater

- 当对象由于高负载的原因而无法立即更新时，任务将会被序列化到在本地文件系统中进行排队，以便服务恢复后进行异步更新。

Swift组件

Auditor

- 检查对象，容器和账户的完整性，如果发现比特级的错误，文件将被隔离，并复制其他的副本以覆盖本地损坏的副本；其他类型的错误会被记录到日志中。

Account Reaper

- 移除被标记为删除的账户，删除其所包含的所有容器和对象。

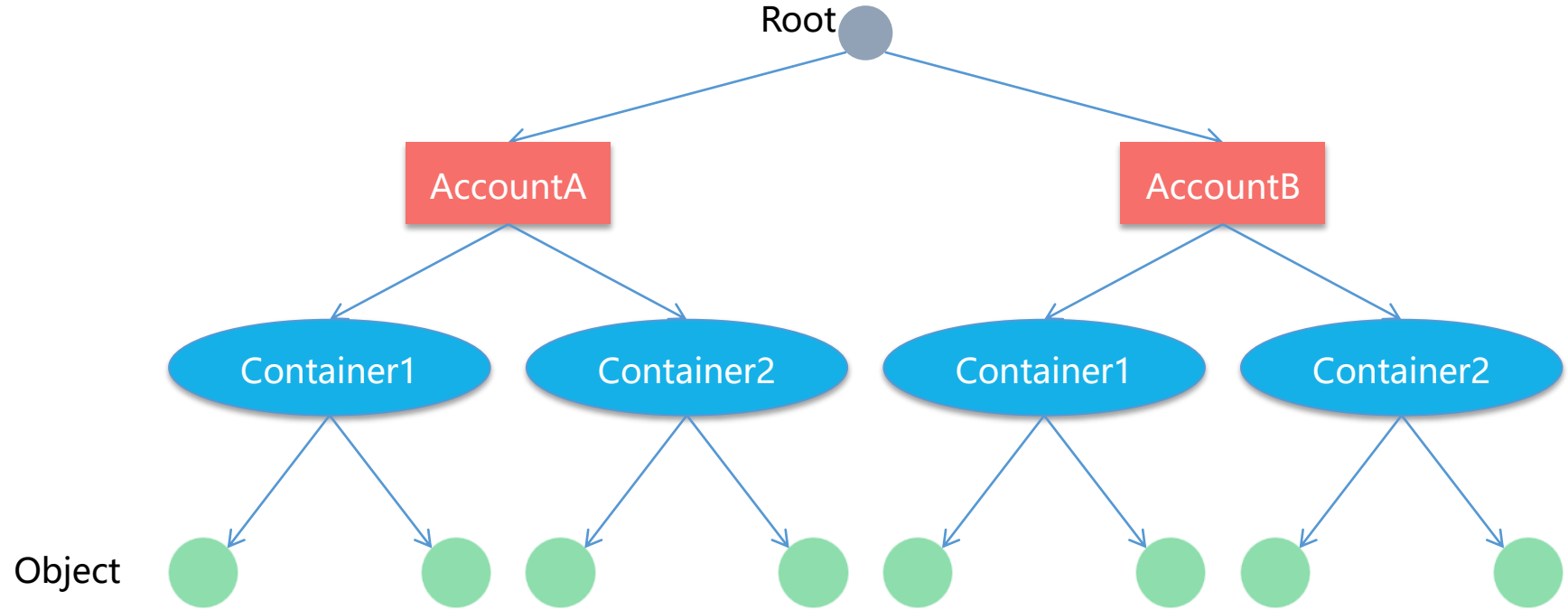
Swift API

- Swift 通过 Proxy Server 向外提供基于 HTTP 的 REST 服务接口，对账户、容器和对象进行 CRUD 等操作。
- Swift RESTful API 总结

资源类型	URL	GET	PUT	POST	DELETE	HEAD
账户	/account/	获取容器列表	-	-	-	获取账户元数据
容器	/account/container	获取对象列表	创建容器	更新容器元数据	删除容器	获取容器元数据
对象	/account/container/object	获取对象内容和元数据	创建、更新或复制对象	更新对象元数据	删除对象	获取对象元数据

Swift数据模型

- Swift共设三层逻辑结构：Account/Container/Object（即账户/容器/对象)。
- 每层节点数均没有限制，可以任意扩展。



思考题

1. OpenStack中有哪几种类型的存储?
2. 块存储和对象存储分别适用于哪些场景?
3. 块存储服务主要包含哪些组件?
4. 块存储服务是怎么创建卷的?
5. 对象存储服务主要包含哪些组件?

本章总结

- OpenStack存储概述
- 块存储Cinder
- 对象存储Swift

学习推荐

- OpenStack社区
  - https://www.openstack.org/


谢 谢
www.huawei.com