SDC
STORAGE DEVELOPER CONFERENCE
SNIA ■ SANTA CLARA, 2014

OpenStack Cloud Storage

Sam Fineberg
HP Storage

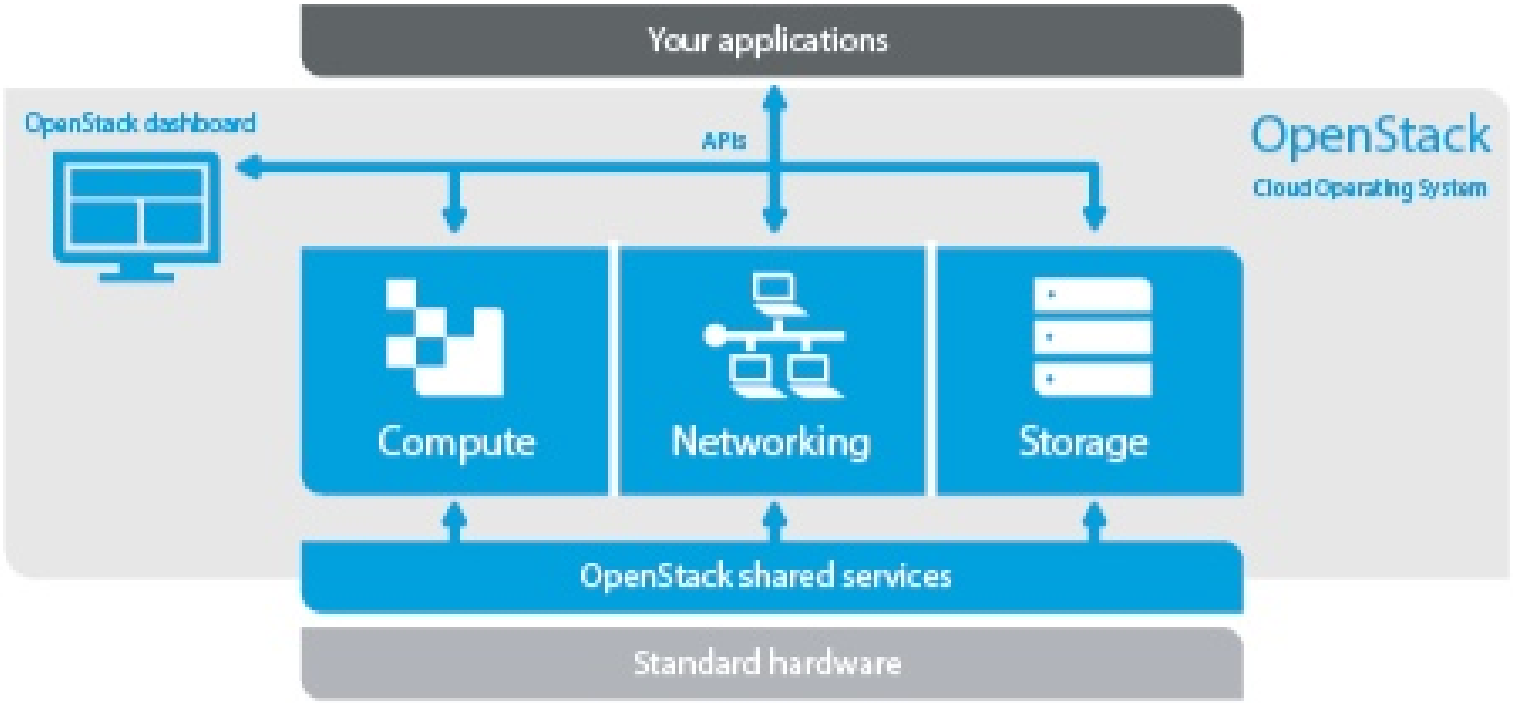
What is OpenStack®

Free open source (Apache license) software governed by a non-profit foundation (corporation) with a mission **to produce the ubiquitous Open Source Cloud Computing platform that will meet the needs of public and private clouds regardless of size, by being simple to implement and massively scalable.**

- **Massively scalable** cloud operating system that controls large pools of **compute, storage**, and **networking** resources
- **Community open source** with contributions from **1000+ developers** and **180+** participating **organizations**
- **Open** web-based API ***Programmatic*** **Infrastructure** as a Service
- **Plug-in architecture**; allows different hypervisors, block storage systems, network implementations, hardware agnostic, etc.

What is OpenStack®

A series of interrelated projects that control pools of compute, storage, and networking infrastructure exposed as a consistent and open layer (API) for a heterogeneous Infrastructure as a Service (IaaS) environment.



OpenStack® programs (13 Integrated; 2 supporting)

- 13 packages with 200+ configuration items

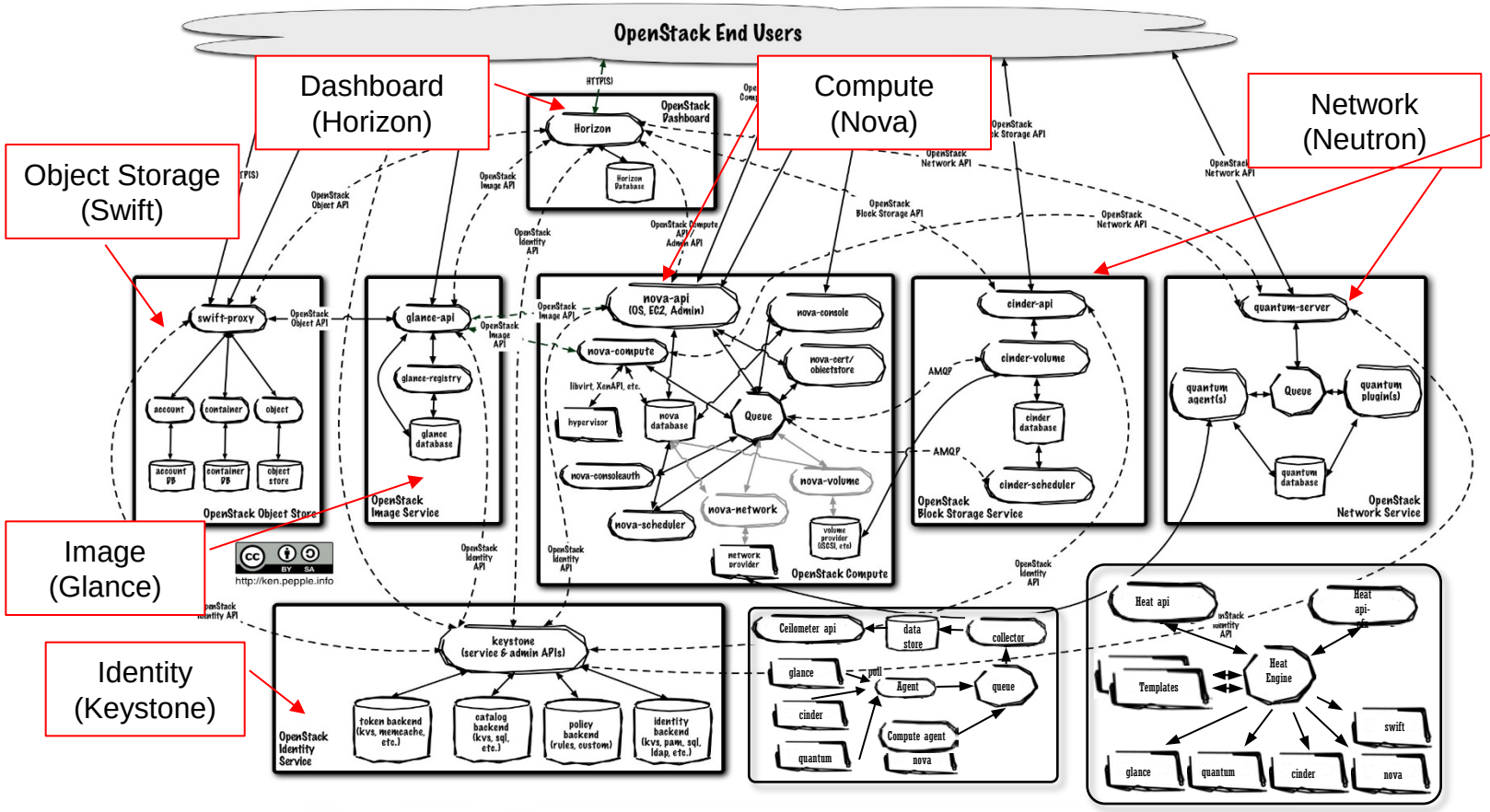
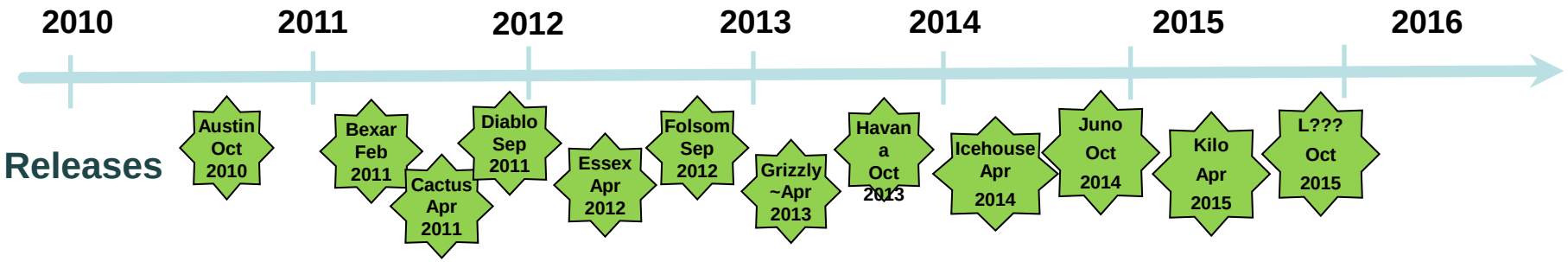


Image source: http://docs.openstack.org

OpenStack® releases



6-Month Release Cycles

- Spring and fall releases for predictable availability
- Release names use alphabetic naming

Planning

- Developers plan the next release at the Design Summit
- Sessions are selected by projects leads and are generally driven by blueprint topics
- Project Technical Leads (PTLs) accept a number of blueprints into the release plan

Development &Testing

- Milestone iterations (commonly 5 weeks) are defined and followed
- Development/documentation occurs per the blueprints and plan
- End of cycle testing with a defined Release Criteria process

Release

- Release Candidate is made available
- Next release is open for development
- After hardening, release occurs

Cloud Storage 101

Block Storage

- Generally SCSI protocol based, organized by volumes
- Boot volumes for VMs
- Ephemeral vs. Persistent
- Not directly consumed by applications, usually used to hold a filesystem
- Low level storage abstraction upon which file and object storage is built

File Storage

- Files organized in directory hierarchy and accessed by pathname
- File-based NAS protocols like NFS and CIFS
- Rich and complex application support: random access, multiple readers, in-place file updates, locking, etc.

Object Storage

- Efficient flat namespace: objects organized by accounts, containers, object keys, and metadata
- HTTP / REST / URL based – easily scriptable, many language choices
- Relatively simple interface compared to file storage
- Scalable to very high object counts
- More easily scaled across multiple geographies
- Ideal for relatively static data

Cloud Storage 101

Cinder

Block Storage

- Generally SCSI protocol based, organized by volumes
- Boot volumes for VMs
- Ephemeral vs. Persistent
- Not directly consumed by applications, usually used to hold a filesystem
- Low level storage abstraction upon which file and object storage is built

Manila

File Storage

- Files organized in directory hierarchy and accessed by pathname
- File-based NAS protocols like NFS and CIFS
- Rich and complex application support: random access, multiple readers, in-place file updates, locking, etc.

Swift

Object Storage

- Efficient flat namespace: objects organized by accounts, containers, object keys, and metadata
- HTTP / REST / URL based – easily scriptable, many language choices
- Relatively simple interface compared to file storage
- Scalable to very high object counts
- More easily scaled across multiple geographies
- Ideal for relatively static data

Cloud Storage 101

Block Storage

- Generally SCSI protocol based, organized by volumes
- Boot volumes for VMs
- Ephemeral vs. Persistent
- Not directly consumed by applications, usually used to hold a filesystem
- Low level storage abstraction upon which file and object storage is built

File Storage

- Files organized in directory hierarchy and accessed by pathname
- File-based NAS protocols like NFS and CIFS
- Rich and complex application support: random access, multiple readers, in-place file updates, locking, etc.

Object Storage

- Efficient flat namespace: objects organized by accounts, containers, object keys, and metadata
- HTTP / REST / URL based – easily scriptable, many language choices
- Relatively simple interface compared to file storage
- Scalable to very high object counts
- More easily scaled across multiple geographies
- Ideal for relatively static data

OpenStack block storage - Cinder

Types of block storage

- Ephemeral storage
  - Usually used as a boot disk
  - Backed by VM Host local disk
  - Disappears when a VM is deleted or if a VM host fails
- Persistent block storage
  - Boot or additional storage volume
  - Not tied to the VM host, doesn't go away if VM disappears
  - Accessed over a network
  - Must be provisioned first, and then attached to a VM

OpenStack Block Storage - Cinder

- Nova Volume
  - Originally OpenStack Compute (Nova) included support for ephemeral volumes
    - Used for boot/runtime storage of VMs
    - Volumes were typically backed by VM server files
  - Nova Volume had limited support persistent volumes on iSCSI
- Beginning with the Folsom release, a separate persistent block storage service, Cinder, was created
  - Cinder is a core part OpenStack project
  - Consists of a plug-in interface for supporting various block storage devices

Cinder Core Functionality

- Volumes
  - Create, Show, Update, Extend, Delete Volume
  - List Volume Summaries/Details
- Snapshots
  - Create, Show, Update, Delete Snapshot
  - List Snapshot Summaries/Details
- Volume types
  - List/Show volume types
  - Volume types defined in cinder.conf file, “extra specs” for advanced features
- Other features
  - Backup to swift
  - Volume migration
  - QoS – extended specs

Cinder Supported Devices

- Drivers are available for
  - Ceph RADOS Block Device
  - Coraid AoE driver configuration
  - Dell EqualLogic volume driver
  - EMC SMI-S iSCSI driver
  - GlusterFS driver
  - HDS iSCSI volume driver
  - HP 3Par StoreServ Fibre Channel and iSCSI drivers
  - HP StoreVirtual (LeftHand)
  - Huawei storage driver
  - IBM XIV/DS8K volume driver
  - IBM GPFS volume driver
  - IBM Storwize family and SVC volume driver
  - NetApp unified driver
  - Nexenta drivers
  - NFS driver
  - SolidFire
  - VMware VMDK driver
  - Windows
  - XenAPINFS
  - XenAPI Storage Manager volume driver
  - Zadara

Cinder volume attachment

- NFS/shared file system
  - Volume file created on file share, attached through libvirt or other VM specific mechanism
  - All VM hosts must already be attached to file share
- iSCSI
  - Attach to iSCSI volume over TCP/IP network
  - Cinder provisions volumes on target, coordinates initiator and target connection
- FC
  - Pre-zoned/flat – Grizzly-Havana
  - Zoning – Icehouse-Juno
- Other options
  - Ceph RBD, ATAoE, …

Key new Cinder features - Icehouse

- Ability to change the type of an existing volume (retype)
- Add volume metadata support to the Cinder Backup Object
- Implement Multiple API workers
- Add ability to delete Quota
- Add ability to import/export backups in to Cinder
- Added Fibre Channel Zone manager for automated FC zoning during volume attach/detach
- Ability to update a volume type encryption
- Ceilometer notifications on attach/detach

Cinder Futures

- Better testing – continuous integration
- Better backup support - incrementals
- HA
  - LVM driver – DRBD
- Bare metal boot
- Volume manage/unmanage
- Consistency groups
- Storage Pools

Cloud Storage 101

Block Storage

- Generally SCSI protocol based, organized by volumes
- Boot volumes for VMs
- Ephemeral vs. Persistent
- Not directly consumed by applications, usually used to hold a filesystem
- Low level storage abstraction upon which file and object storage is built

File Storage

- Files organized in directory hierarchy and accessed by pathname
- File-based NAS protocols like NFS and CIFS
- Rich and complex application support: random access, multiple readers, in-place file updates, locking, etc.

Object Storage

- Efficient flat namespace: objects organized by accounts, containers, object keys, and metadata
- HTTP / REST / URL based – easily scriptable, many language choices
- Relatively simple interface compared to file storage
- Scalable to very high object counts
- More easily scaled across multiple geographies
- Ideal for relatively static data

OpenStack Object Storage - Swift

- Swift was one of the original components of OpenStack
  - Originally developed by RackSpace
  - Distributed scale-out shared-nothing object storage
- Web APIs for data access
  - HTTP Put, Get, Post, Delete, …
- Swift is open source code, not an API standard
  - There is no compatibility suite and the API is subject to change
  - Enhanced direct implementations
    - SwiftStack, HP (StoreAll), etc.
  - Some vendors implement Swift compatible APIs
    - Ceph, Cleversafe, etc.
  - Some vendors have swift plugins – diskfile layer
    - Gluster, Seagate Kinetic, etc.

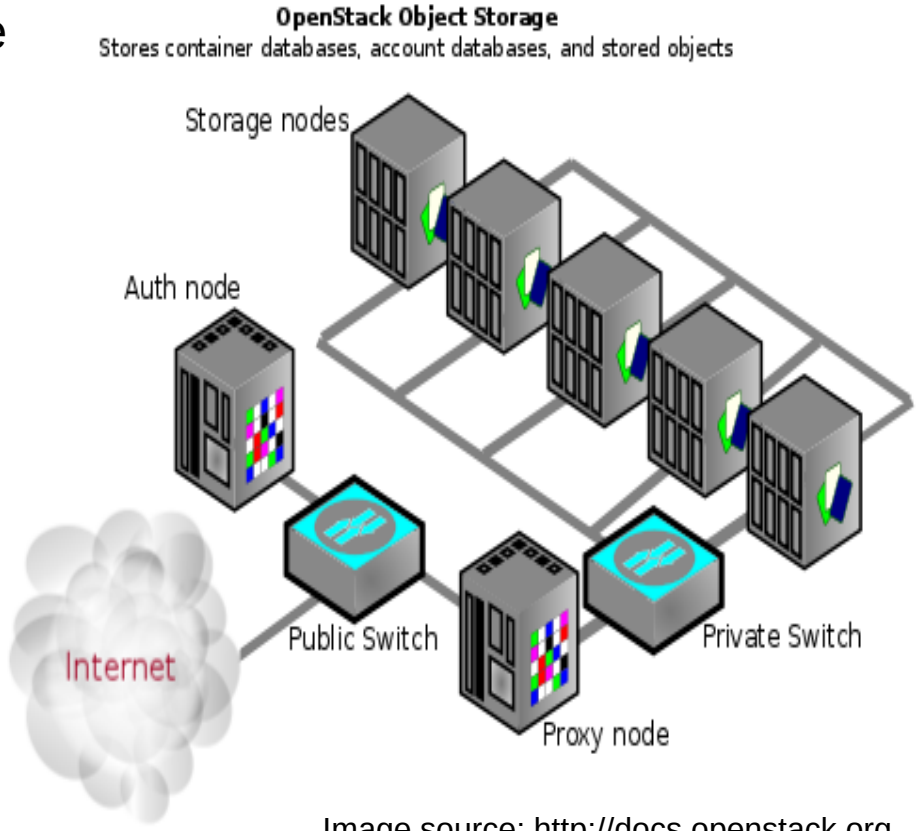


Image source: http://docs.openstack.org

Swift Storage Model

- Users authenticate with Keystone
- Each account owns a set of containers
- Containers hold a set of objects
  - Containers have metadata
  - Access permissions at container
- Objects contain data
  - Size is limited, composite objects supported for larger objects
  - Objects also have metadata
  - User specifies object name
  - Name can contain pseudo-paths

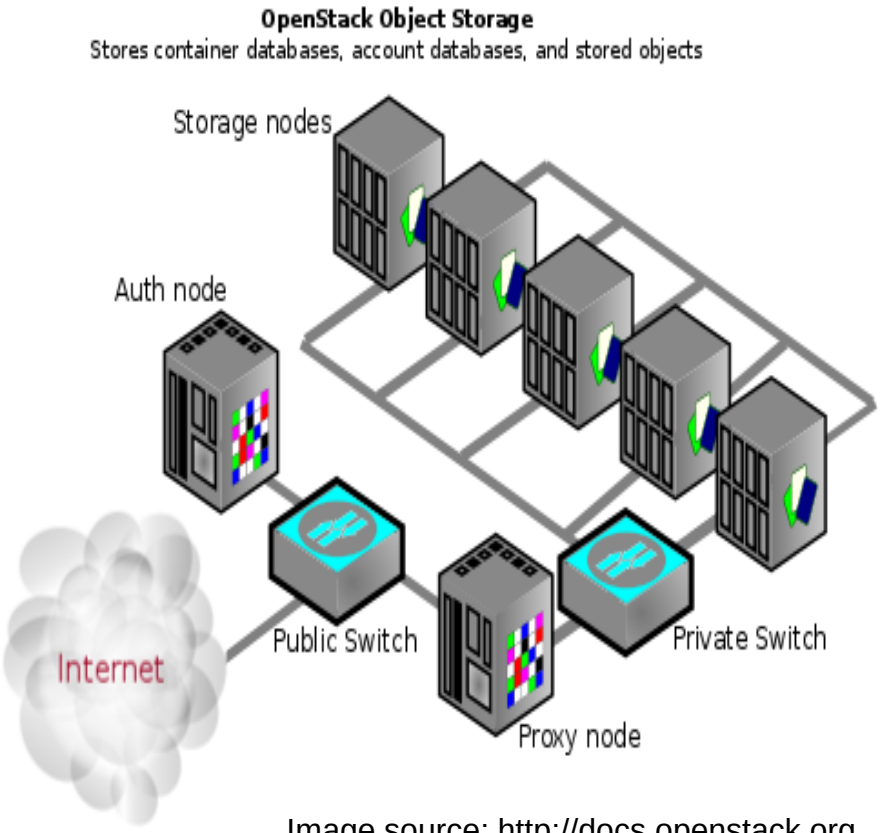


Image source: http://docs.openstack.org

Swift Operation

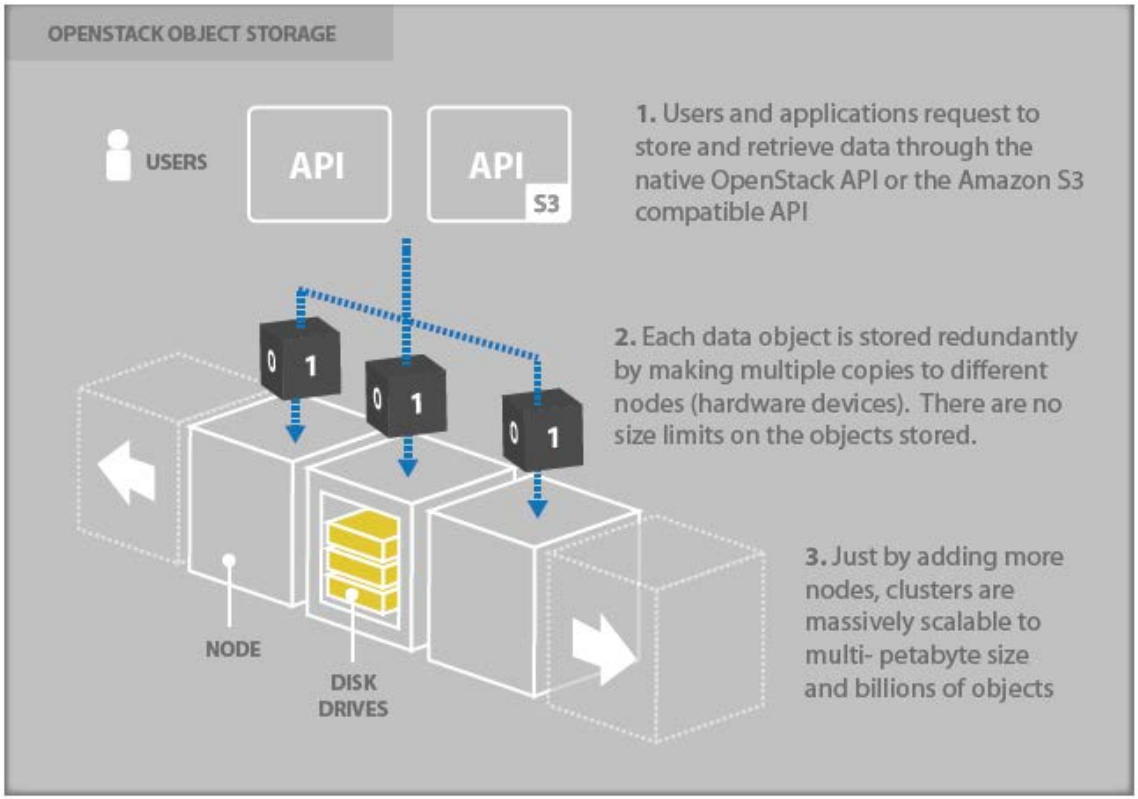


Image source: http://rogerluethy.wordpress.com

Swift Details

- The Ring
  - hash map between names and physical location
  - separate rings for accounts, containers, and objects
  - uses zones, devices, partitions, and replicas.
  - Partitions are replicated, by default, 3 times
- Proxy Server
  - responsible for tying together the Object Storage architecture.
  - for each request, it looks up the location and routes the request.
  - public developer API is exposed through Proxy Server.
- Container Server
  - primary job is to handle listings of objects
  - doesn't know where those object's are
  - listings are stored as SQLite files, replicated across cluster

Swift Details (cont.)

- Account Server
  - similar to the Container Server, responsible for list of containers
  - list are stored as replicated sqlite database files
- Object Server
  - simple server that can store, retrieve and delete objects
  - objects are stored as binary files on the filesystem
  - metadata stored in extended attributes
  - DiskFile API can be used to access external stores

Swift Details (cont.)

- Replication
  - keeps the system in a consistent state through temporary errors
  - compares local data with remote copies to maintain the latest version.
  - replication updates are push based, updating is rsync or ssync
  - pushes records over HTTP or rsync/ssync database files.
- Updaters
  - sometimes container or account can not be immediately updated.
  - if an update fails, it is queued locally, the updater will process it
  - eventual consistency
- Auditors
  - Auditors crawl the local server checking integrity
  - If corruption found, file is quarantined, replication will replace
  - other errors are logged

Other Features

- ACLs
  - Account ACLs supported at container level
- Global clusters
  - Support for geo-dispersed zones – region tier
  - Differing replica counts per region
  - Read from closest replica – based on timing
  - Write affinity
- Bulk requests
  - Auto extract of archive files (e.g., tar)
  - Bulk delete
- Quotas
  - Allows admin to set per account limits (bytes)

Swift Futures - Policies

- Policies (Juno)
  - Allows user to specify how they want data stored
    - Hardware, encoding, # copies
    - Based on multiple ring architecture
  - Precursor for erasure codes or other differentiated storage

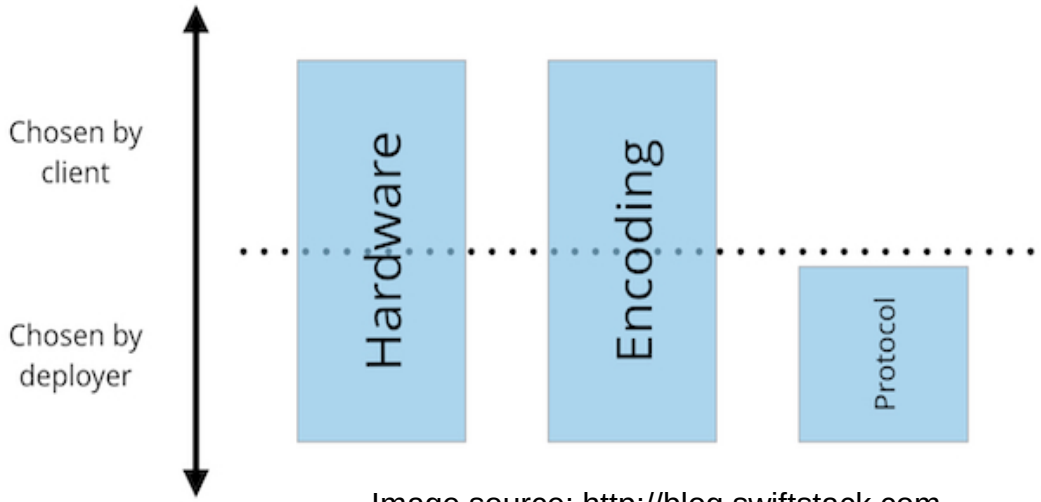


Image source: http://blog.swiftstack.com

Swift Futures – Erasure Codes

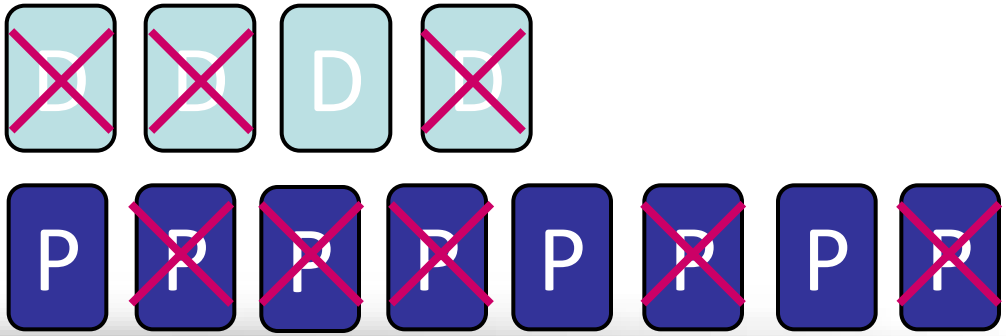
- Erasure coding has been used widely in RAID for years
  - Can yield improved storage efficiency and/or reliability
  - Objects broken into data and parity “shards” by proxy servers,
  - Shards stored like objects, reconstruction at proxy when retrieved

Put: 

Data: 

Parity: 

Fragments, schematically:



Swift Futures – Erasure Codes

- Project status
  - Driven by Intel, SwiftStack, Box and EVault
  - Still in progress, probably won't make Juno

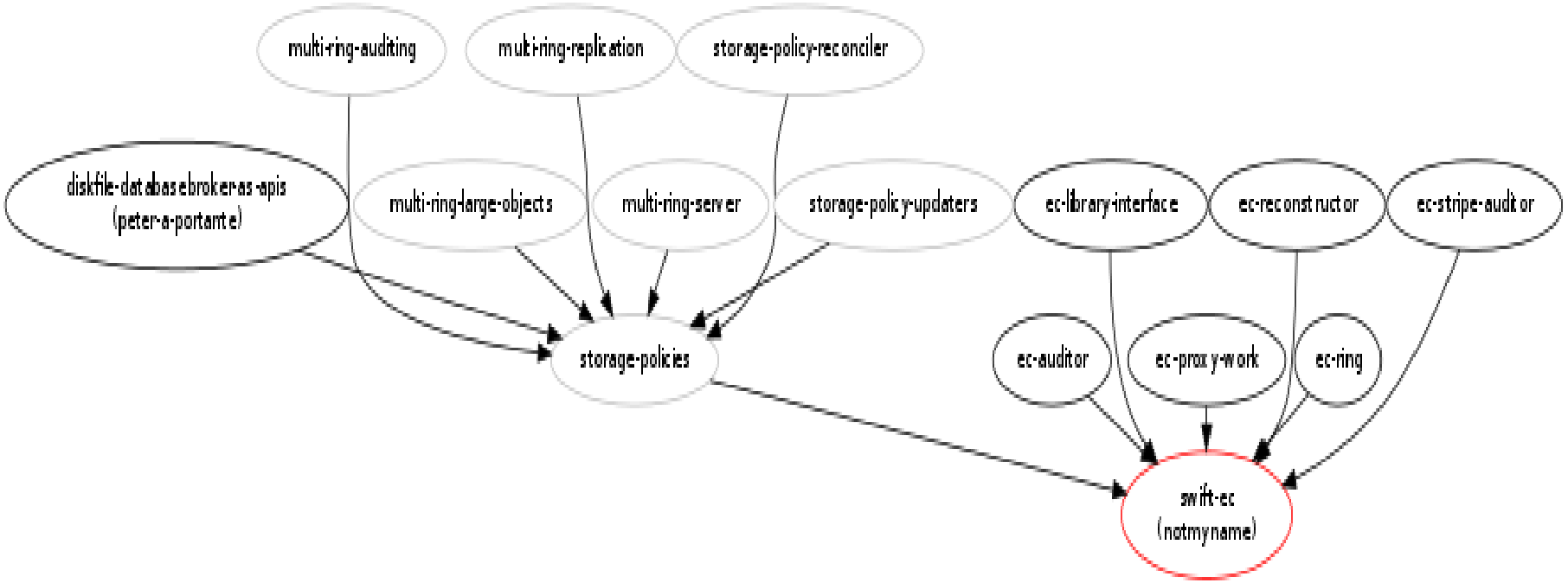


Image source: http://blueprints.Launchpad.net

Cloud Storage 101

Block Storage

- Generally SCSI protocol based, organized by volumes
- Boot volumes for VMs
- Ephemeral vs. Persistent
- Not directly consumed by applications, usually used to hold a filesystem
- Low level storage abstraction upon which file and object storage is built

File Storage

- Files organized in directory hierarchy and accessed by pathname
- File-based NAS protocols like NFS and CIFS
- Rich and complex application support: random access, multiple readers, in-place file updates, locking, etc.

Object Storage

- Efficient flat namespace: objects organized by accounts, containers, object keys, and metadata
- HTTP / REST / URL based – easily scriptable, many language choices
- Relatively simple interface compared to file storage
- Scalable to very high object counts
- More easily scaled across multiple geographies
- Ideal for relatively static data

OpenStack Shared File Storage

- Not originally planned for OpenStack,
  - Clouds like AWS don't have shared file storage
  - Security and networking issues are difficult
- Demand has continued for this type of a service
  - For legacy applications that are not object storage enabled
  - Other options like running NFS on a VM or Gluster across nodes are complex to get right
- Originally proposed as an extension to Cinder
  - Implemented in Grizzly Cinder
  - Split into a new project "Manila" in Havana

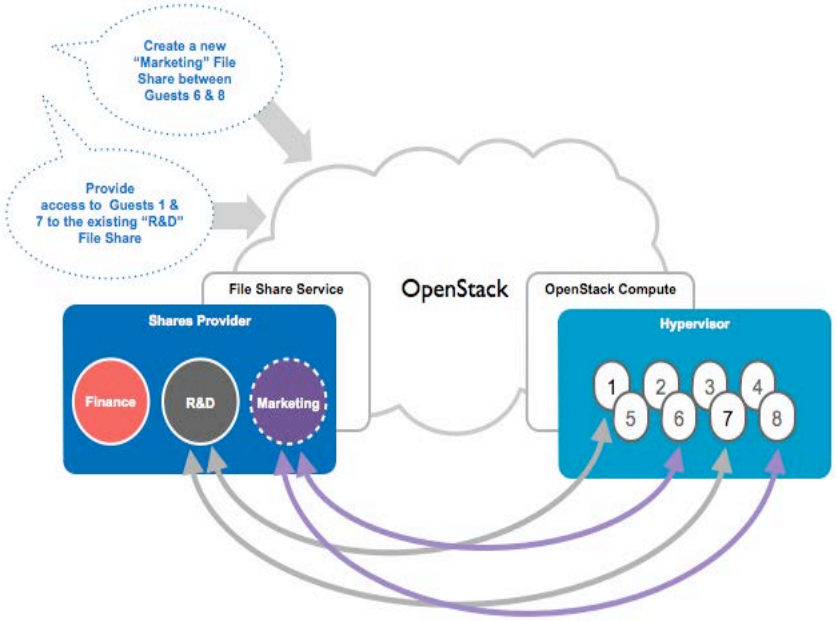


Image source: http://wiki.openstack.org/wiki/Manila

Manila vs. Cinder

- Shared Filesystsms
  - Instances are able to communicate directly with the storage backend over the network
  - Uses NFS or CIFS to talk to shares, multiple readers/writers are expected
  - Storage backend is able to serve multiple tenants while maintaining secure separation between them
- Block storage
  - Usually iSCSI or FC, like a disk
  - Exposed through the hypervisor, actual “mount” is to VM host
  - Typically single attach (or if multiple attach coordination is required to corruption)
- Additional work is needed for the networking portion of Manila
  - Attaching a shared filesystem to one or more instances in a tenant network
  - Setup of security domains and other features that exist in a NAS environment but not a SAN environment

Manila core functionality

- Back-ends
  - Generic – creates a VM running SMB or NFS
  - Vendor drivers
- Functions
  - Create or delete a share
  - Show or list shares
  - Allow/deny access to a share
  - Create/delete share snapshots, list snapshots

Manila Project

- Manila emerged as a separate project over Summer 2013
  - Recently promoted to “Incubator” project for Juno
  - Goal is to make core in Kilo (or later)
- Still in development, and looking for more participation

Manila Roadmap

- Icehouse
  - full multitenancy so that drivers are able to automatically create virtual instances
  - "generic" driver which implements multitenancy by layering on top of Nova and Cinder
  - Drivers for hardware that can't support secure multitenancy still allowed to exist in single tenant mode.
- Juno
  - Broaden driver support
  - Create a mechanism to provide gateway-based secure multitenancy
    - To support storage backends that only support single tenant mode.
- Kilo
  - Progress towards inclusion in core
  - Hardening, broader support

Summary

- OpenStack is free open source software for implementing public and private clouds
  - Many companies and individuals are involved in development
  - Products and services based on OpenStack exist today, and more are appearing as it matures
- OpenStack has support for block, object, and file storage
  - Both open source and commercial storage products supported
  - These provide basic and advanced functionality
  - More features, drivers, implementations are in the works
- OpenStack is maturing quickly
  - Give it a try – multiple free and commercial source distributions
  - There is plenty more work to do
  - The development process is open to anyone who wants to help

Questions?

- More Information
  - http://openstack.org
  - http://docs.openstack.org
  - https://launchpad.net/cinder
  - https://launchpad.net/swift
  - https://launchpad.net/manila
- Contact me:
  - fineberg@hp.com