OpenStack Object Storage

Software to reliably store billions of objects
distributed across standard hardware

Training Goals

- By the end of this course you should:
  - Be able to list the components in Swift
  - Have an understanding of the Swift architecture
  - Be able to interact with an OpenStack Object Storage deployment
  - Know where to go to find more information

openstack™

Object Storage Summary

FULLY DISTRIBUTED

COMMODITY HARDWARE

FEATURES OPTIMIZED FOR SCALE

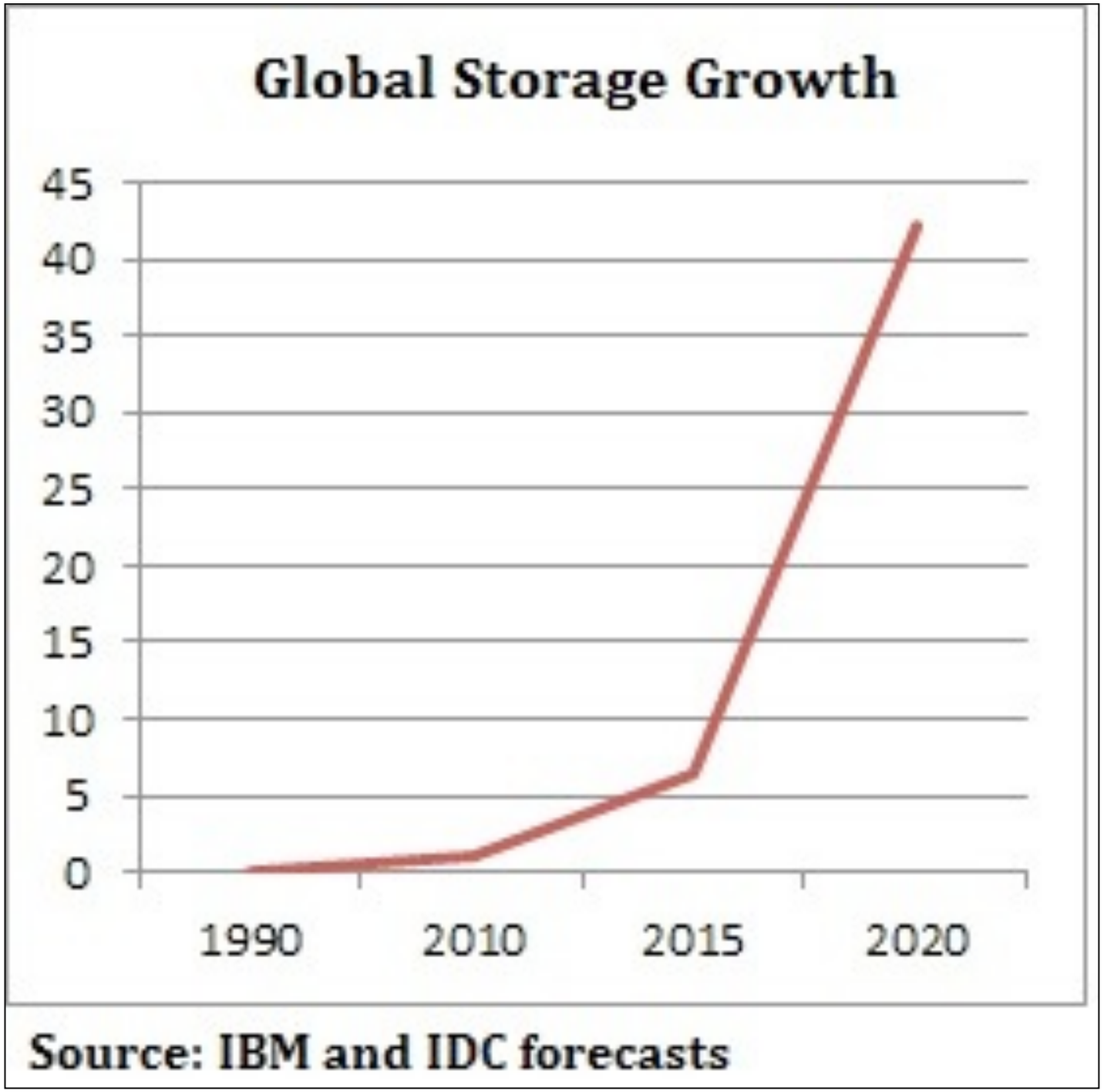
DATA PROTECTION IN SOFTWARE

NOT A FILESYSTEM

AUGMENTS SAN/NAS/DAS, DOESN'T REPLACE

Why

- Explosion in unstructured data
- High operational costs



Source: IBM and IDC forecasts

openstack™

Zettabyte

1,000 Exabytes
1,000,000 Petabytes
All of the data on Earth today
(150GB of data per person)

Zettabyte

2% OF THE DATA ON EARTH IN 2020

Data Must Be Stored Efficiently

If we stored all of the global data as “an average” enterprise..

ITEM	MONTHLY FIGURES
ENTERPRISE AVERAGE STORAGE COST	$1.98 PER GIGABYTE
WORLD GDP	$5.13 TRILLION
COST TO STORE A ZETTABYTE	$1.98 TRILLION

..it would take..
..38.5% of the World GDP!

Example Small Scale Deployment

5 Storage Nodes, $0.13 per GB (monthly)

490TB of disk, 160TB usable

18.5kVA, 35 RU, 5 “half cabinets”

$0.114 per GB OPEX (monthly)

$0.014 per GB CAPEX ($200k, 36 month refresh)

$0.38/GB (with 3 copies), < 20% of “average”

OpenStack Object Storage Key Features

REST-based **API**

Data distributed evenly throughout system

Scalable to multiple petabytes, billions of objects

Account/Container/Object structure (not file system, no nesting) plus **Replication** (N copies of accounts, containers, objects)

No central database

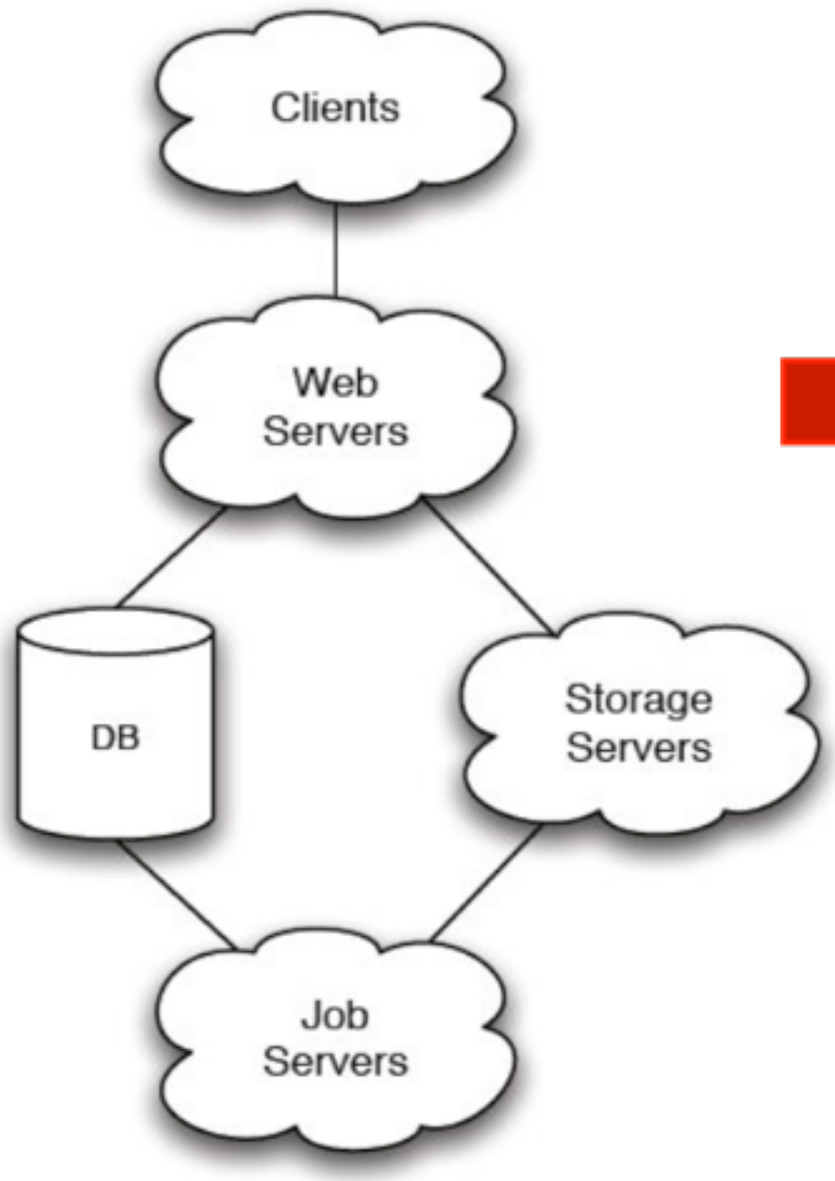
Hardware agnostic: standard hardware, RAID not required

Swift vs. RAID

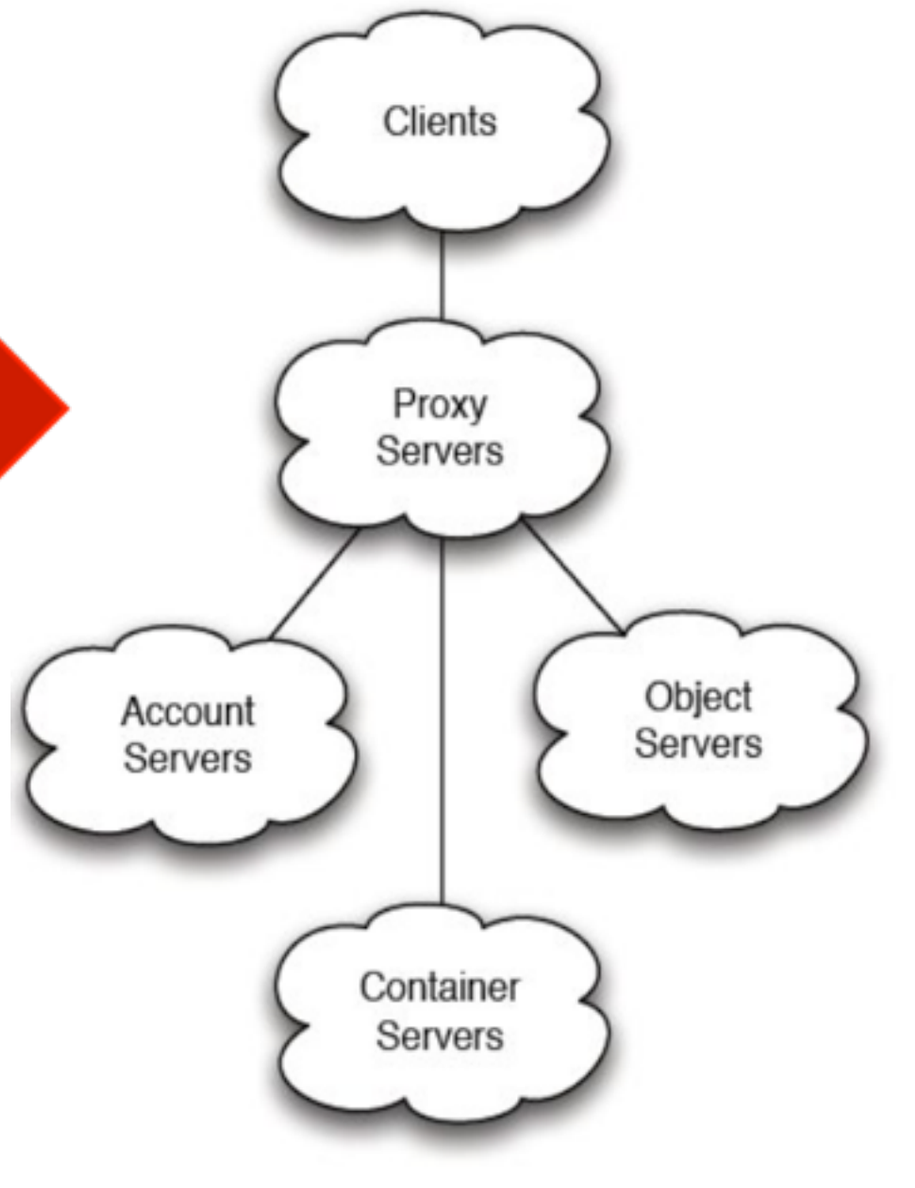
SWIFT	RAID
Massively scalable multiple container storage	Limited to # of disks in a physical form factor
Easily add capacity, only moving rebalanced data	May not be possible to resize
66%+ loss of capacity	0-50% loss of data capacity
3x+ data redundancy	0-2x maximum data redundancy
Designed for remote large/long term file storage	Designed for performance/direct access
Uses commodity hardware	Typically requires high end hardware

Evolution of Object Storage Architecture

Version 1: Central DB
(Rackspace Cloud Files 2008)



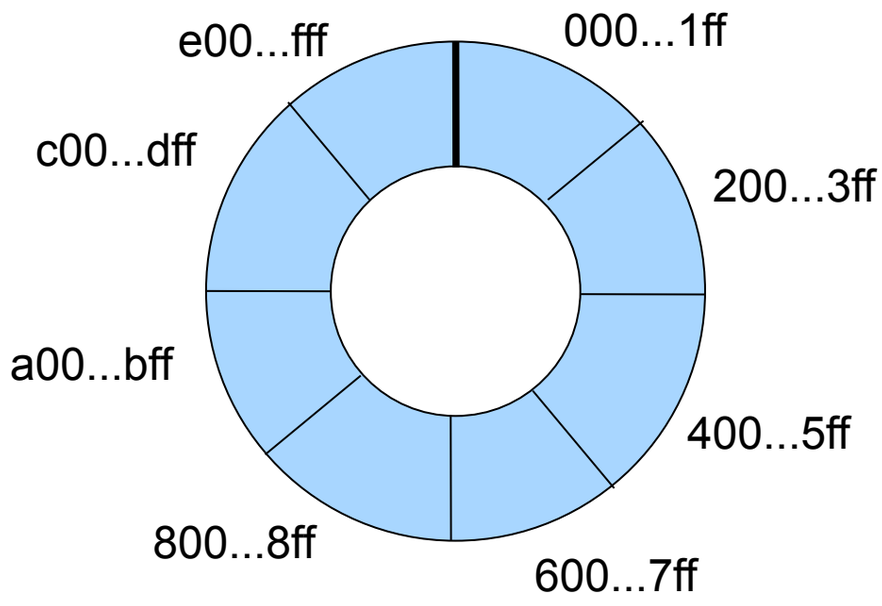
Version 2: Fully Distributed
(OpenStack Object Storage 2010)



openstack™

Swift Components: The Ring

- Objects in swift are identified by MD5 hash
- 128 bit identifier, based on unique identifiers of object
  - (account_id, container, object) = object
  - (account_id, container) = container
  - (account_id,) = account
- MD5 key space broken into “partitions” that map to a specific storage location
  - Storage Server IP
  - Storage Server Port
  - Storage Server Device (/dev/sdb)

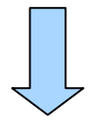


openstack™

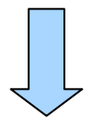
Swift Components (Object Ring)

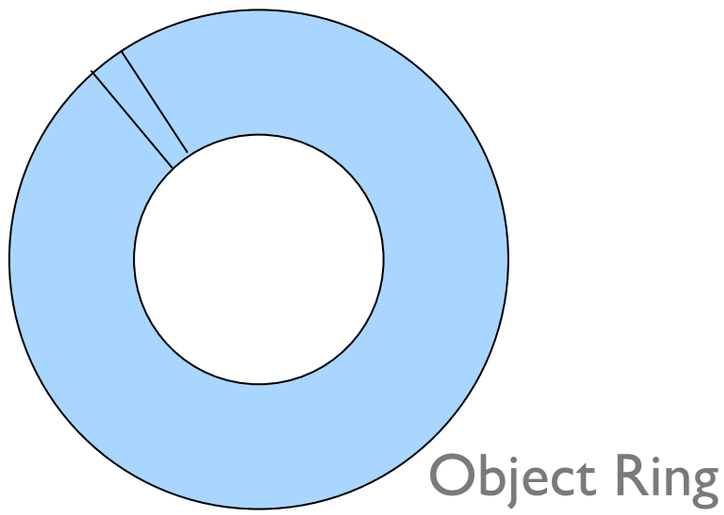
PUT /v1.0/<account_id>/<container>/<object>

(Upload object to container)

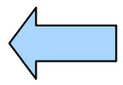


ecb25d1facd7c6760f7663e394dbeddb





Object Ring



Partition #93823

z1-10.1.0.2:6000/sda1

z5-10.1.0.18:6000/sdf1

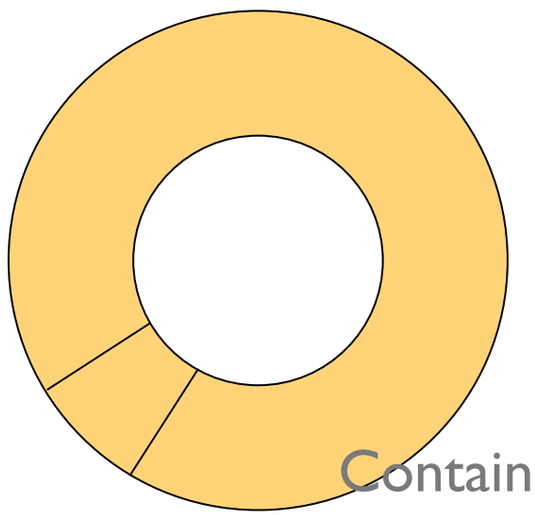
z2-10.1.0.13:6000/sdb1

Swift Components (Container Ring)

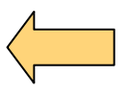
GET /v1.0/<account_id>/<container>

(Get objects in container)

415b952f70ceff5ee85cfcae165ed329



Container Ring



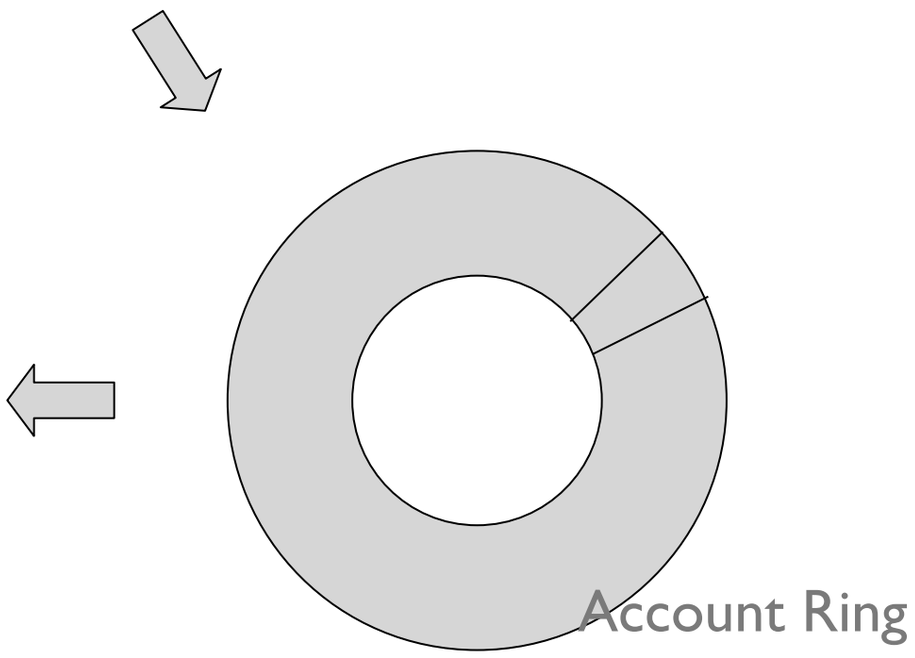
Partition #3764
z2-10.1.0.13:6001/sdg1
z4-10.1.0.6:6001/sda1
z1-10.1.0.12:6001/sdc1

Swift Components (Account Ring)

GET /v1.0/<account_id>

(Get containers in account)

89c5270c0e27c648cd2a27e0034f3b85



Partition #341
z3-10.1.0.26:6002/sdc1
z6-10.1.0.18:6002/sdj1
z5-10.1.0.32:6002/sdm1

System Components

- **The Ring**: Mapping of names to entities (accounts, containers, objects) on disk.
  - Stores data based on zones, devices, partitions, and replicas
  - Weights can be used to balance the distribution of partitions
  - Used by both the proxy server and storage nodes for many background processes
- **Proxy Server**: Request routing, exposes the public API
- **Object Server**: Blob storage server, uses xattrs, uses binary format
  - Recommended to run on XFS
  - Object location based on path from name hash & timestamp

System Components (Cont.)

- **Container Server**: Handles listing of objects, stores as SQLite DB
- **Account Server**: Handles listing of containers, stores as SQLite DB
- **Replication**: Keep the system consistent, handle failures
- **Updaters**: Process failed or queued updates
- **Auditors**: Verify integrity of objects, containers, and accounts

Replication

- Account and Container replication
  - Hash comparison of SQLite databases per node
  - Update only from row X based on tuple of known records
  - If DB is missing, entire DB is pushed
- Object replication
  - Hash comparison of directories and files
  - Rsync worker for changed folders only
  - Push based approach

Exercise: Multi-Node Installation

- Based on the Swift Multi-Node Install documents on wiki.openstack.org
  - http://swift.openstack.org/howto_installmultinode.html
- Simulated hardware environment on Rackspace Cloud Servers
- Training setup and cut/paste format installers on github
  - http://github.com/rpedde/swift-training-kick

Agenda (Day 2)

Morning

User Management
- Swauth versus Keystone
- Service catalog
- Exercise: user managment

Swift CLI
- Operations (stat/put/upload/download)
- Exercise: using the “swift” tool

Understanding ACLs
- Theory: referrer vs. account
- Exercise: Getting and setting ACLs

Afternoon

Operations overview
- swift-drive-audit
- handoff node operations
- Exercise: install swift-drive-audit
- Exercise: observe handoff node actions
- Chassis failure/switch failure

Monitoring overview
- Necessary monitoring points
- Scaling and updates
- Exercise: dispersion reports

Basic user management

- Swift acounts/users can be managed with built in utilities starting with “swauth-”
- Adding an admin/user:
  - swauth-add-user -A http://localhost:8080/auth/ -K superpass –a account username password
- Verify users, accounts, passwords
  - swauth-list -A http://localhost:8080/auth/ -K superpass [account]
- Delete a user:
  - swauth-delete-user -A http://localhost:8080/auth/ -K superpass account username

Exercise: Basic user management

- Create an account called “testaccount” with users:
  - test1 (administrative account)
  - test2 (non-administrative account)

- Verify the users exist using swauth-list

Swift Tool CLI (“swift”)

- swift is part of the swift packages
- Show storage stats for a user:
  - swift -A http://url:8080/auth/v1.0/ -U account:user -K password stat
- Upload a file:
  - swift -A http://url:8080/auth/v1.0/ -U account:user –K password upload container yourfile.txt
- Download your file:
  - swift -A http://url:8080/auth/v1.0/ -U account:user –K password download container yourfile.txt –o outputfile.txt

Exercise: Swift Tool CLI (“swift”)

- Create a test container called “testcontainer” using the test1 administrative account previously created
- Upload a file to the test container
- Verify it can be downloaded with the test1 user
- Can the file be downloaded with the test2 user?

Swift ACLs

- Read and Write ACLs, set with “swift post -r/-w”
- Based on referrer, account, or user
  - Referrer
    - .r:* (all referrers)
    - .r:.somewhere.com (only from *.somewhere.com)
    - .r:-.microsoft.com (not from *.microsoft.com
  - Accounts/Users
    - testaccount (any user in the testaccount account)
    - testaccount:test1 (only the test1 user)

Swift ACLs

- ACLs can be combined
  - .r:*,.r:-specifichost.specificdomain.com
  - testaccount:test1,testaccount:test2

- ACLs evaluated top to bottom, last ACL wins
  - .r:-specifichost.specificdomain.com,.r:*
    - Bad: still allows specifichost
  - .r:*,.r:-specifichost.specificdomain.com
    - Good: allows anyone except specifichost

Exercise: ACLs

- Can the test2 user download the test file?
  - Why not?
- Set read ACLs on the container to allow test2 to read the file
  - Can the test2 user read the test file?
  - Can it write a new file to the container?
- Set write ACLs on the container to allow test2 to upload a file

Operations

- If a single drive fails and is not expected to be replaced quickly unmount the drive and remove it from the ring using swift-ring-builder so Swift can work around the failure.
- Once the drive is replaced add it back to the ring and properly mount it.
- The replication services will automatically repopulate the data on the drive.
- Swift-drive-audit can be used in a cron to audit the kern.log and unmount any drives that appear to be reaching a failure threshold.

Exercise: Observing handoff operation

- Observe storage locations with swift-get-nodes
- Unmount drives and watch data move
- Remove drive from rings and push ring data
- Observe data motion

Exercise: Drive Auditing

- Install swift-drive-audit script
- Set up drive auditor to run out of cron

Operations

- If a storage node fails, determine length of time the node will be out of service
  - Long period of time: Remove the node from the ring using swift-ring-builder so Swift can work around the failure.
  - Short period of time: Swap chassis/replace node and let replication bring the device back into sync

Swift monitoring

- Lots of metrics!
  - Host/Network (Traditional monitoring)
    - Cab uplinks
    - Proxy interfaces
    - LB interfaces
  - Log trawling
    - Bytes in, out, GETs, PUTs, POSTs, etc
    - Proxy response codes
    - Replication times

Swift monitoring (etc)

- Swift-specific monitoring
  - Storage capacity (swift-stats)
  - Async Pending (manual script)
  - Dispersion

Exercise: Dispersion Reports

- Set up dispersion reports

Additional Resources

- Swift administration guide:
  - http://swift.openstack.org/admin_guide.html
- The Ring – explained in 5 parts:
  - http://tlohg.wordpress.com/2011/02/07/building-a-consistent-hashing-ring-part-1/

Appendix

openstack™

Partition Power

Determining your Ring Size

Devices at Max Cluster Size		Number of Ring Partitions Per Device		Target Number of Partitions in the Ring
5	X	100	=	500

Closest Partition Power		Total Number of Partitions in the Ring		Partition Power setting in Ring Builder
2^9	=	512		9

openstack™

Packages for Labs

deb http://ops.rcb.me/packages maverick diablo-d5

Resources

- http://www.openstack.org
- https://launchpad.net/openstack
- https://github.com/openstack
- https://github.com/cloudbuilders
- http://www.referencearchitecture.org/
- http://devstack.org/
- http://programmerthoughts.com/
- http://www.unchainyourbrain.com
- http://www.tlohg.com

Tony Campbell

- tony.campbell@rackspace.com
- @tonytcampbell (twitter)
- +Tony Campbell (google+)
- 1 210-312-4150