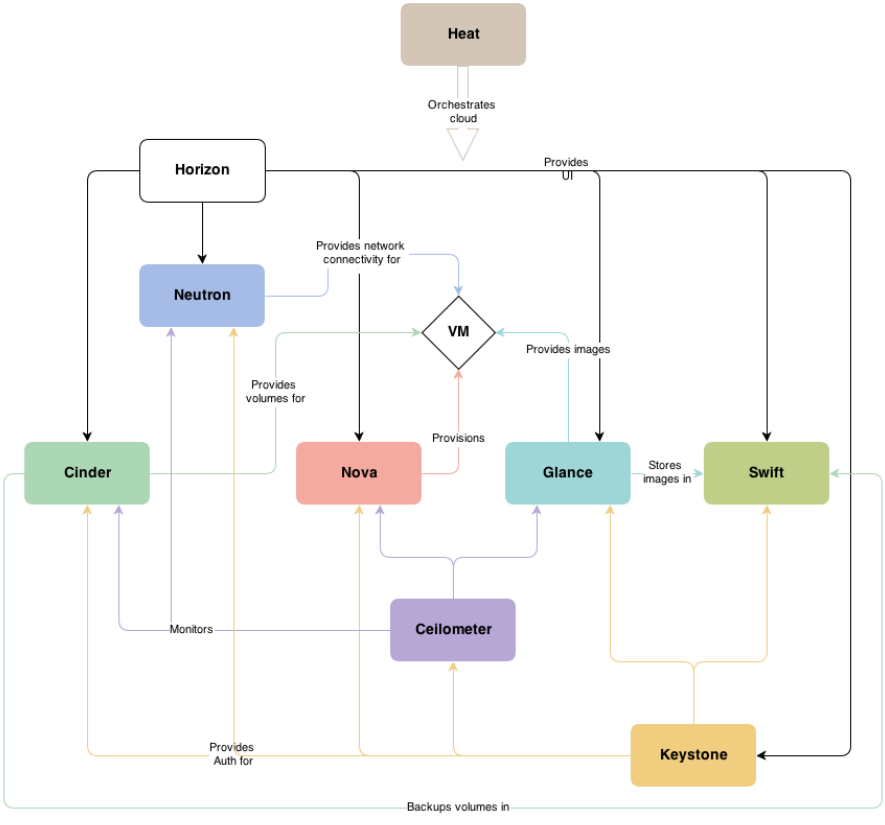
OpenStack Object Storage

Codename: Swift

Swift Overview



OpenStack Object Storage (Swift)

Software to **reliably** store **billions** of objects **distributed** across **standard hardware**

Goals

By the end of this session you should

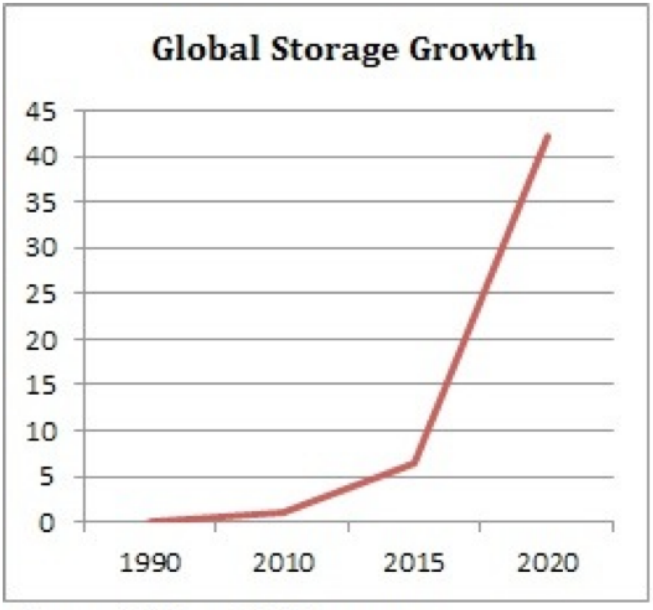
- Be able to list the components in Swift
- Have an understanding of the Swift architecture
- Be able to interact with an OpenStack Object Storage deployment
- Know where to go to find more information

Object Storage Summary

- Fully distributed
- Designed to run on commodity hardware
- Features optimized for scale
- Data protection built into the software
- NOT A FILESYSTEM!
- Augments SAN/NAS/DAS, does not replace

Why?

- Explosion in unstructured data
- High operational costs



Source: IBM and IDC forecasts

What is a Zettabyte?

1,000 Exabytes

1,000,000 Petabytes

All of the data on earth today

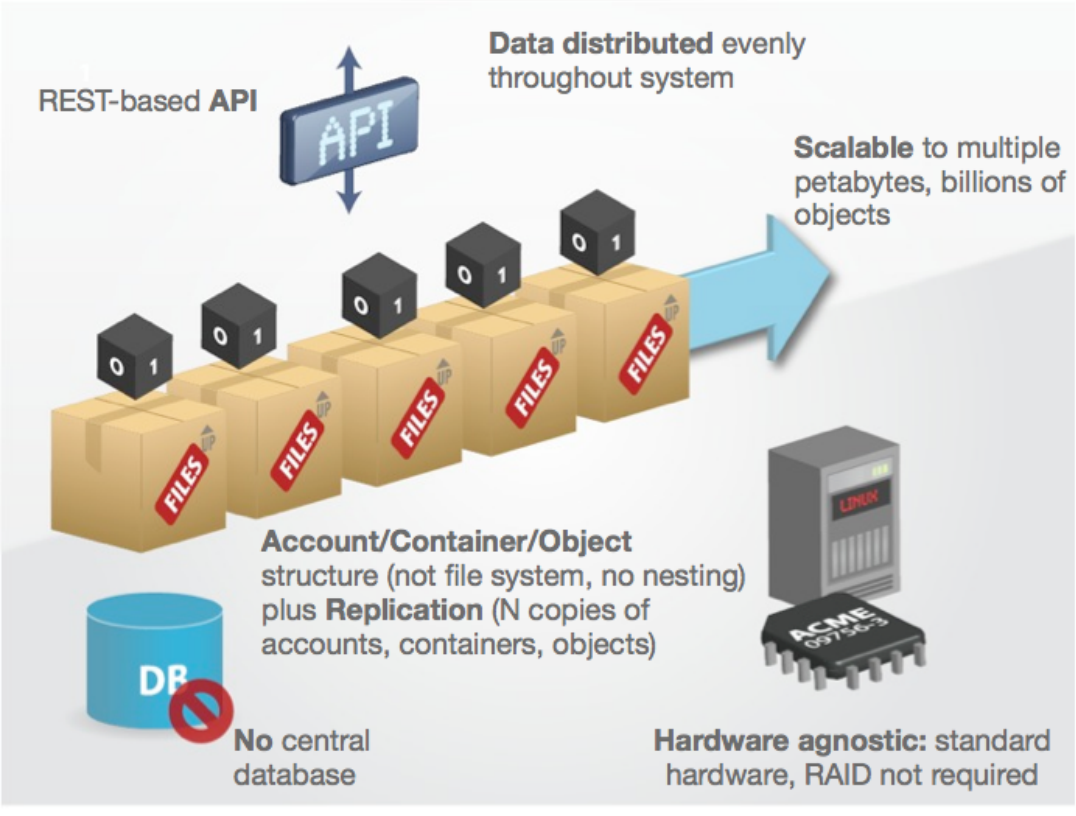
(150GB of data per person)

... and ...

Zettabyte

2% OF THE DATA ON EARTH IN 2020

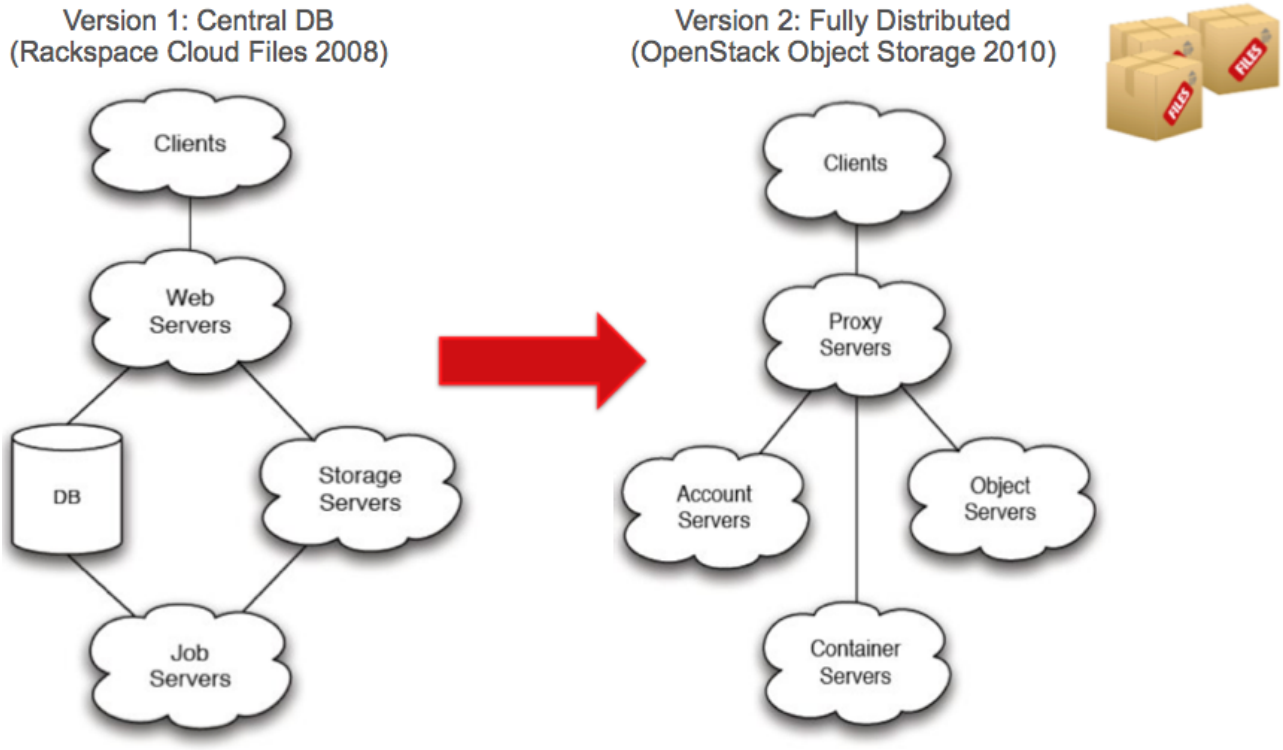
Key Features



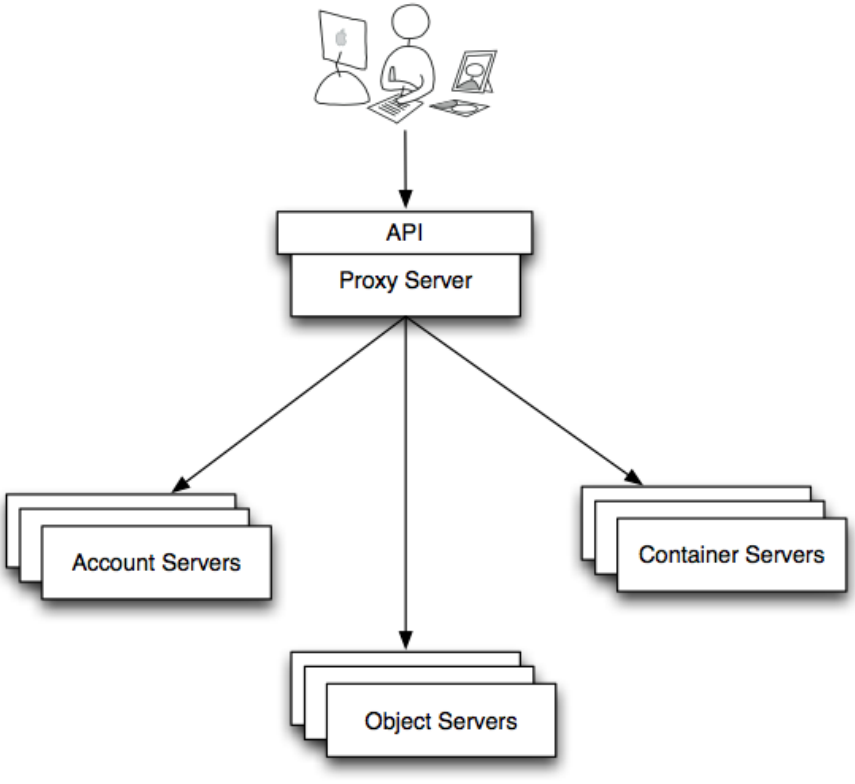
Swift vs RAID

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

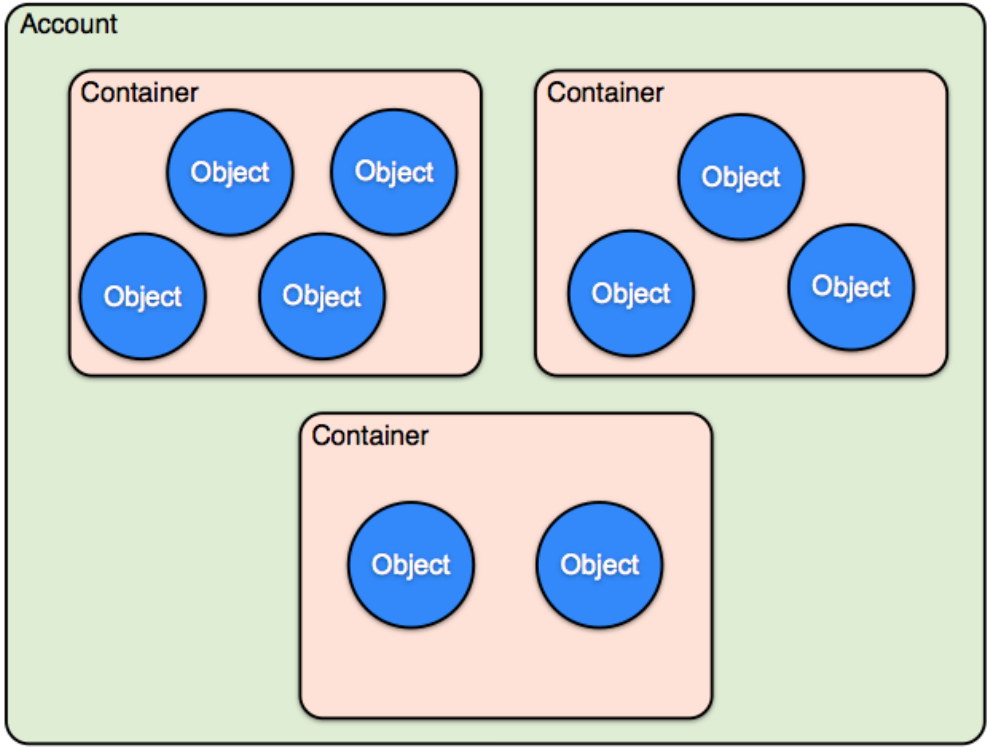
Evolution of the Architecture



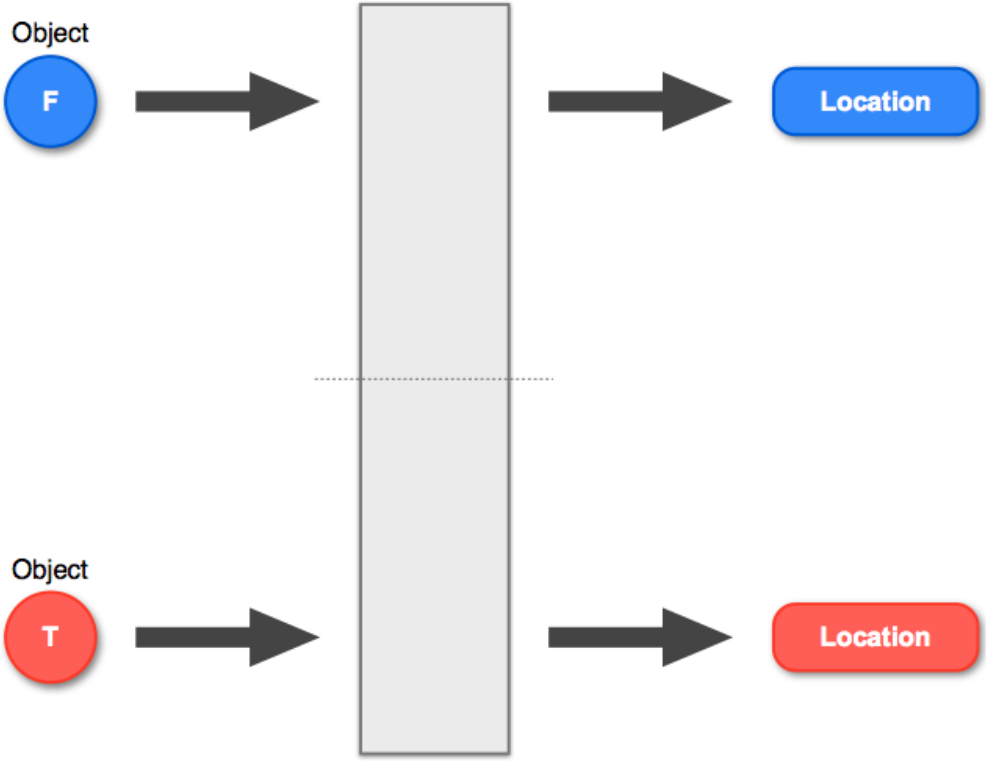
Swift Components



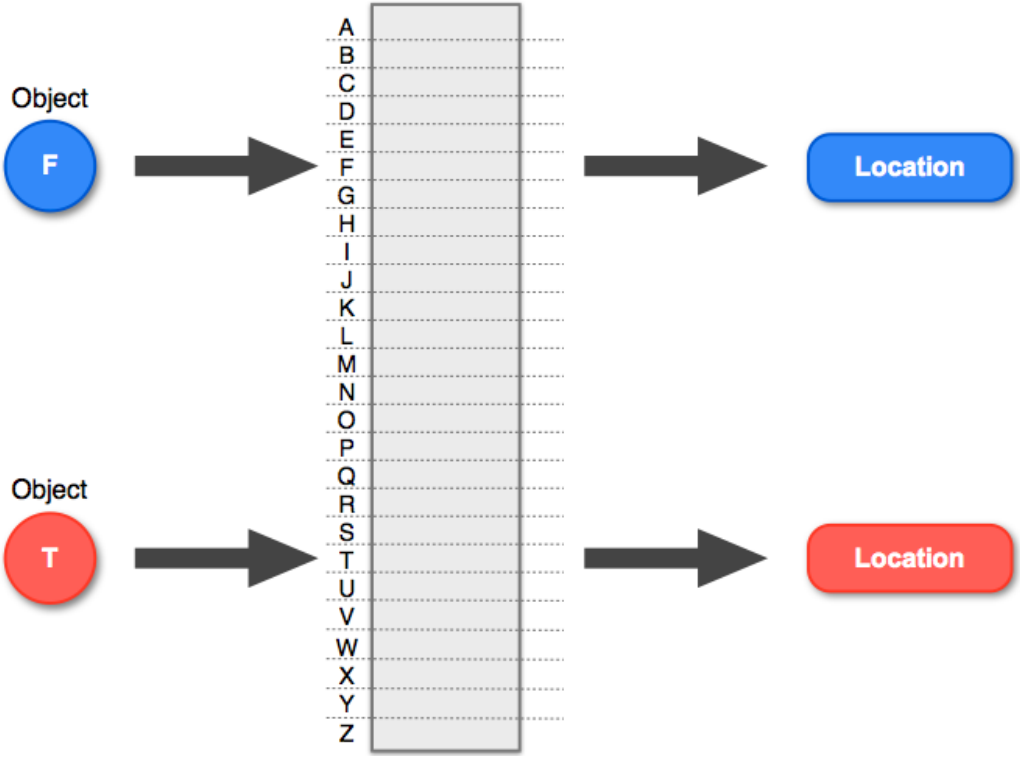
Swift Components



The Ring



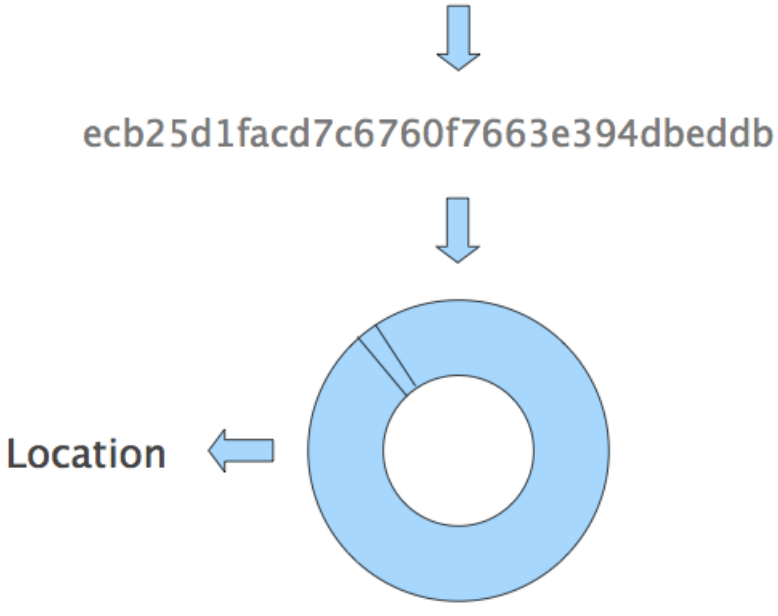
The Ring


Object
F
Object
T
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
Location
Location

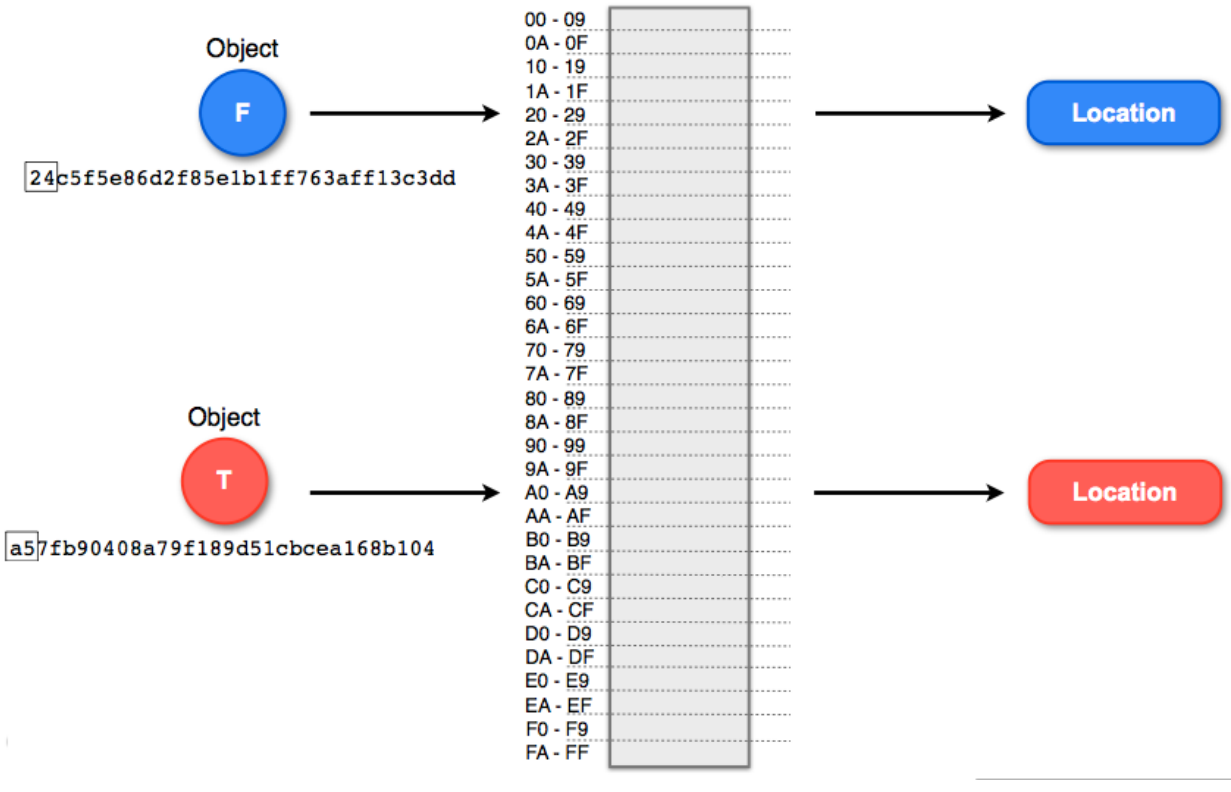
The Ring

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

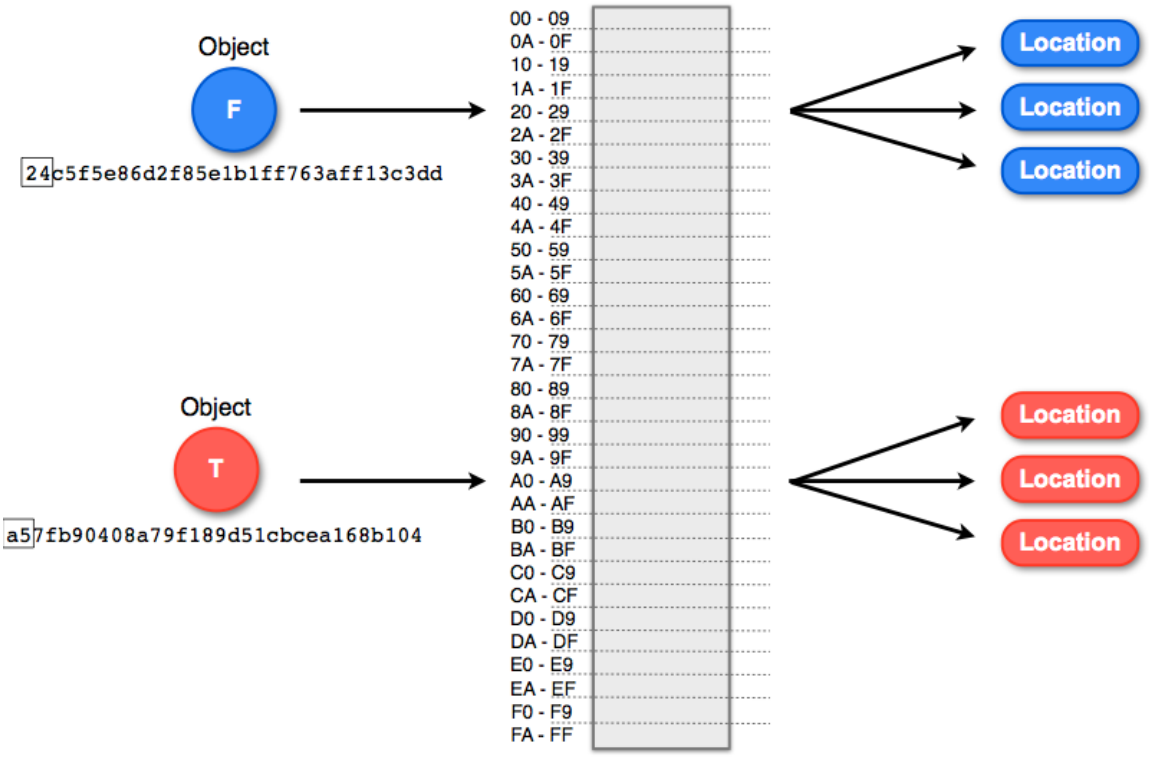
(Upload object to container)



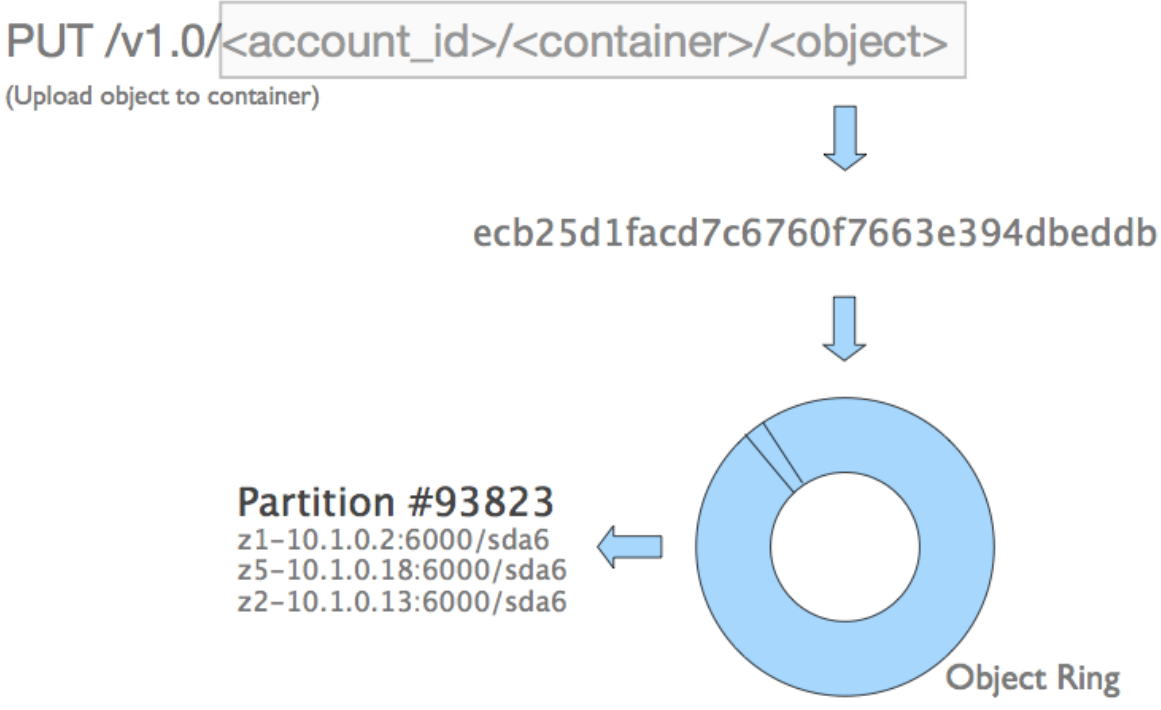
The Ring


Object
F
24c5f5e86d2f85e1b1ff763aff13c3dd
Object
T
a57fb90408a79f189d51cbcea168b104
00 - 09
0A - 0F
10 - 19
1A - 1F
20 - 29
2A - 2F
30 - 39
3A - 3F
40 - 49
4A - 4F
50 - 59
5A - 5F
60 - 69
6A - 6F
70 - 79
7A - 7F
80 - 89
8A - 8F
90 - 99
9A - 9F
A0 - A9
AA - AF
B0 - B9
BA - BF
C0 - C9
CA - CF
D0 - D9
DA - DF
E0 - E9
EA - EF
F0 - F9
FA - FF
Location
Location

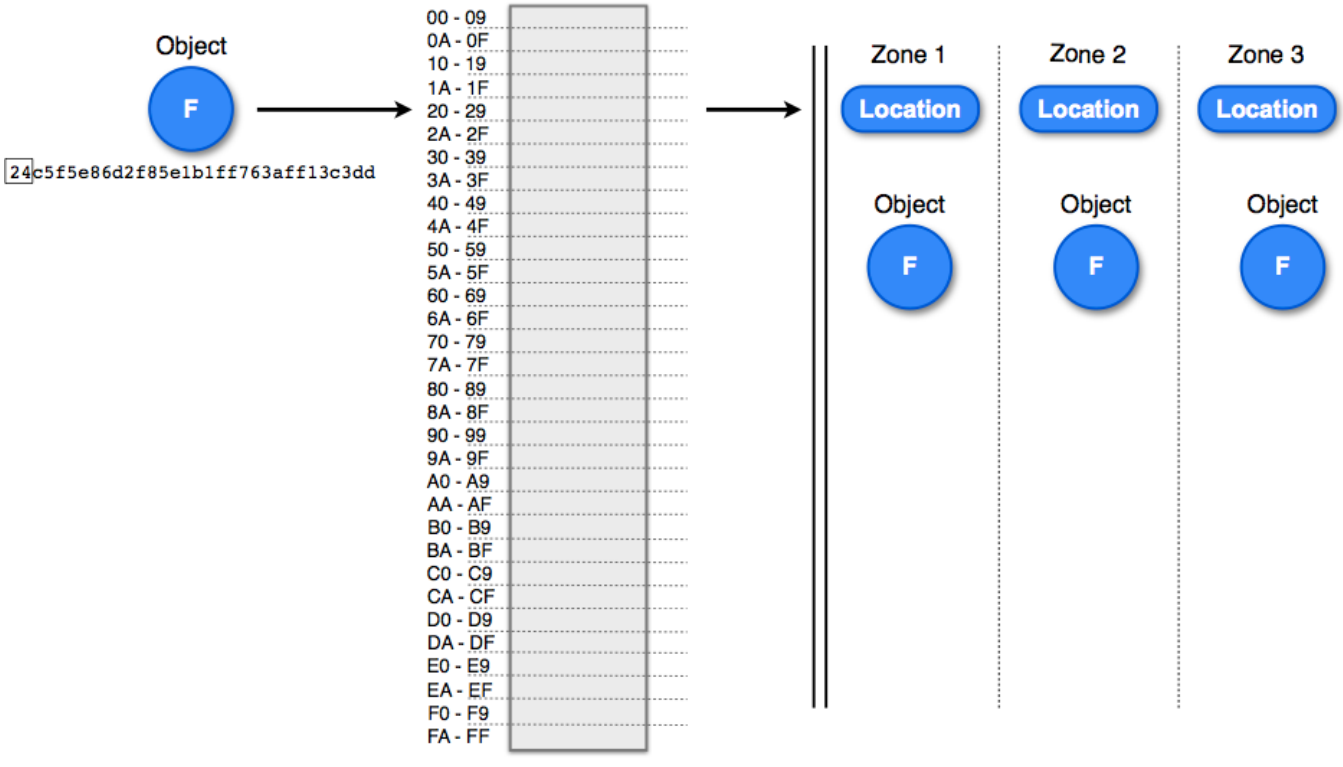
The Ring


Object
F
24c5f5e86d2f85e1b1ff763aff13c3dd
Object
T
a57fb90408a79f189d51cbcea168b104
00 - 09
0A - 0F
10 - 19
1A - 1F
20 - 29
2A - 2F
30 - 39
3A - 3F
40 - 49
4A - 4F
50 - 59
5A - 5F
60 - 69
6A - 6F
70 - 79
7A - 7F
80 - 89
8A - 8F
90 - 99
9A - 9F
A0 - A9
AA - AF
B0 - B9
BA - BF
C0 - C9
CA - CF
D0 - D9
DA - DF
E0 - E9
EA - EF
F0 - F9
FA - FF
Location
Location
Location
Location
Location
Location

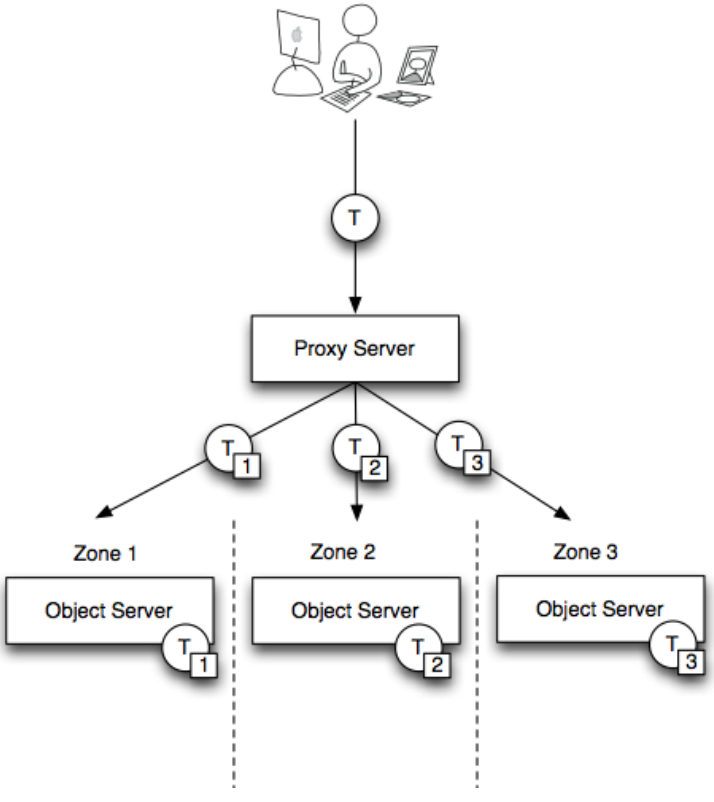
Object Ring


PUT /v1.0/<account_id>/<container>/<object>
(Upload object to container)
ecb25d1facd7c6760f7663e394dbeddb
Partition #93823
z1-10.1.0.2:6000/sda6
z5-10.1.0.18:6000/sda6
z2-10.1.0.13:6000/sda6
Object Ring

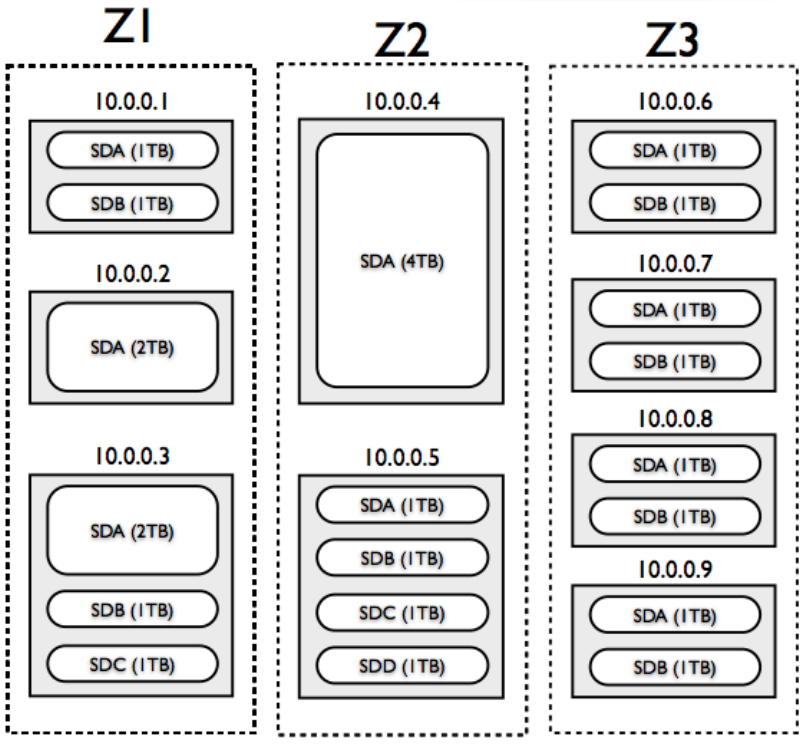
The Ring


Object
F
24c5f5e86d2f85e1b1ff763aff13c3dd
00 - 09
0A - 0F
10 - 19
1A - 1F
20 - 29
2A - 2F
30 - 39
3A - 3F
40 - 49
4A - 4F
50 - 59
5A - 5F
60 - 69
6A - 6F
70 - 79
7A - 7F
80 - 89
8A - 8F
90 - 99
9A - 9F
A0 - A9
AA - AF
B0 - B9
BA - BF
C0 - C9
CA - CF
D0 - D9
DA - DF
E0 - E9
EA - EF
F0 - F9
FA - FF
Zone 1
Location
Object
F
Zone 2
Location
Object
F
Zone 3
Location
Object
F

Swift Object Replicas



Zone Capacity

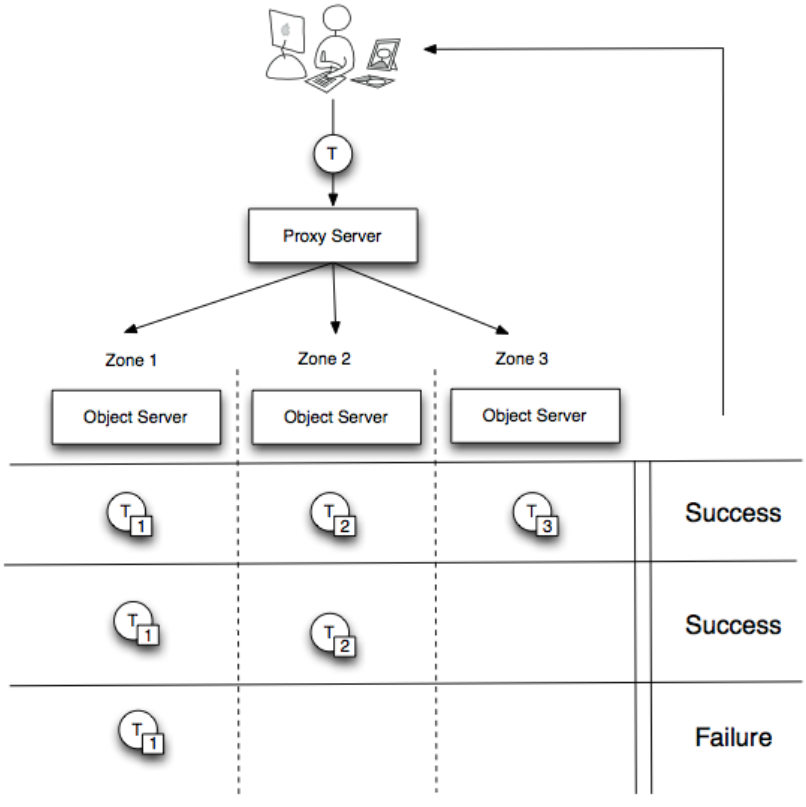


z1-10.0.0.1:6000/sda 1000
z1-10.0.0.1:6000/sdb 1000
z1-10.0.0.2:6000/sda 2000
z1-10.0.0.3:6000/sda 2000
z1-10.0.0.3:6000/sdb 1000
z1-10.0.0.3:6000/sdc 1000

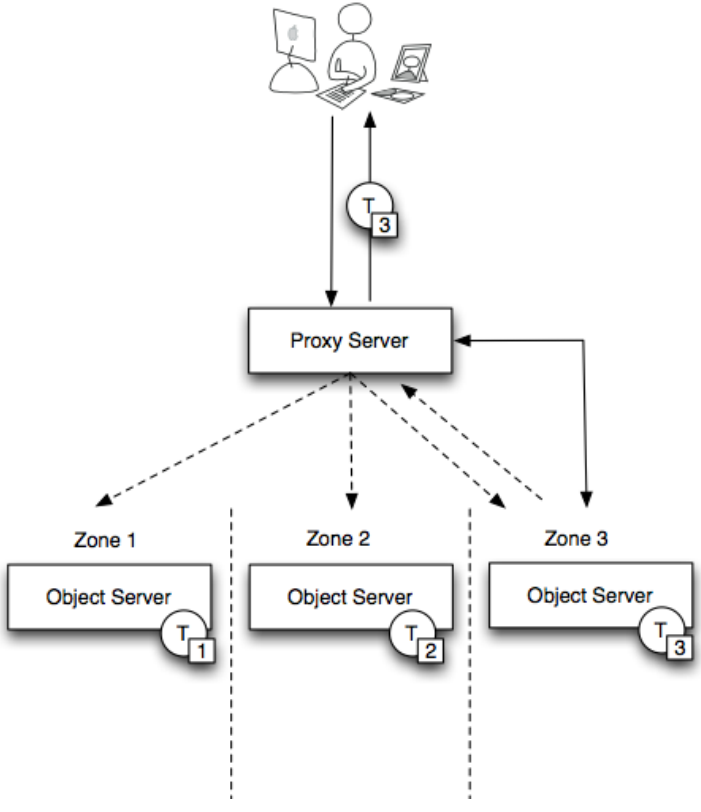
z2-10.0.0.4:6000/sda 4000
z2-10.0.0.5:6000/sda 1000
z2-10.0.0.5:6000/sdb 1000
z2-10.0.0.5:6000/sdc 1000
z2-10.0.0.5:6000/sdd 1000

z3-10.0.0.6:6000/sda 1000
z3-10.0.0.6:6000/sdb 1000
z3-10.0.0.7:6000/sda 1000
z3-10.0.0.7:6000/sdb 1000
z3-10.0.0.8:6000/sda 1000
z3-10.0.0.8:6000/sdb 1000
z3-10.0.0.9:6000/sda 1000
z3-10.0.0.9:6000/sdb 1000

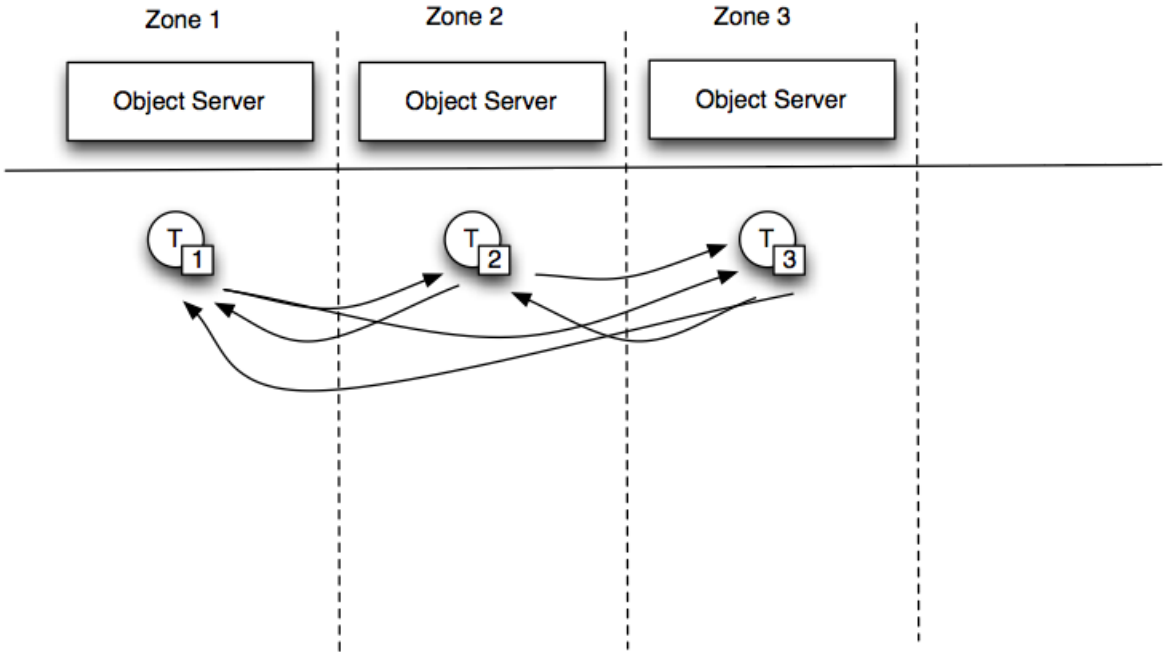
Writing Objects



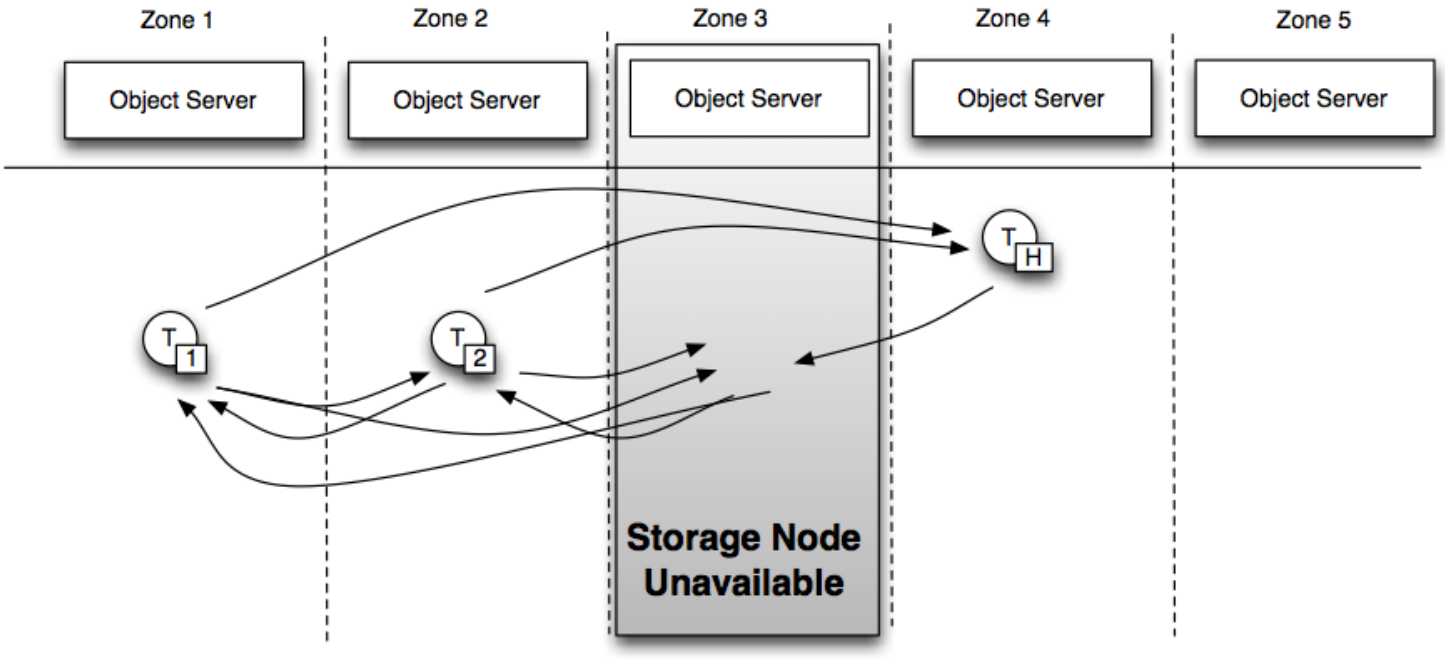
Reading Objects



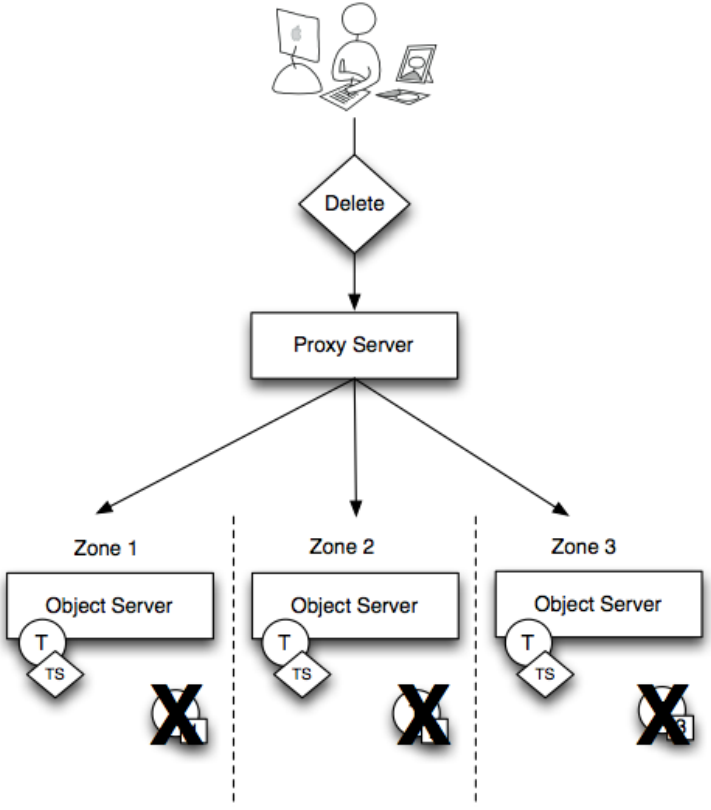
Object Replication



Handoff Nodes

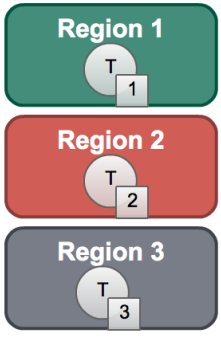


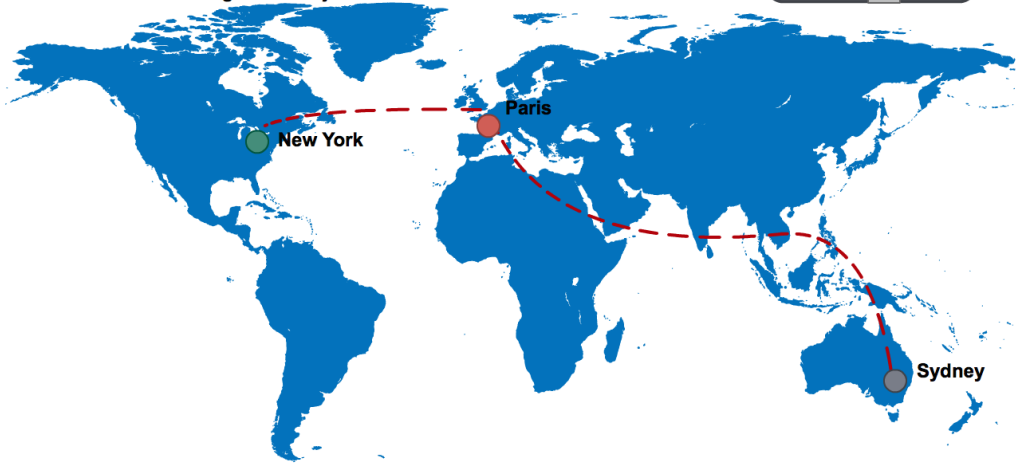
Deleting Objects



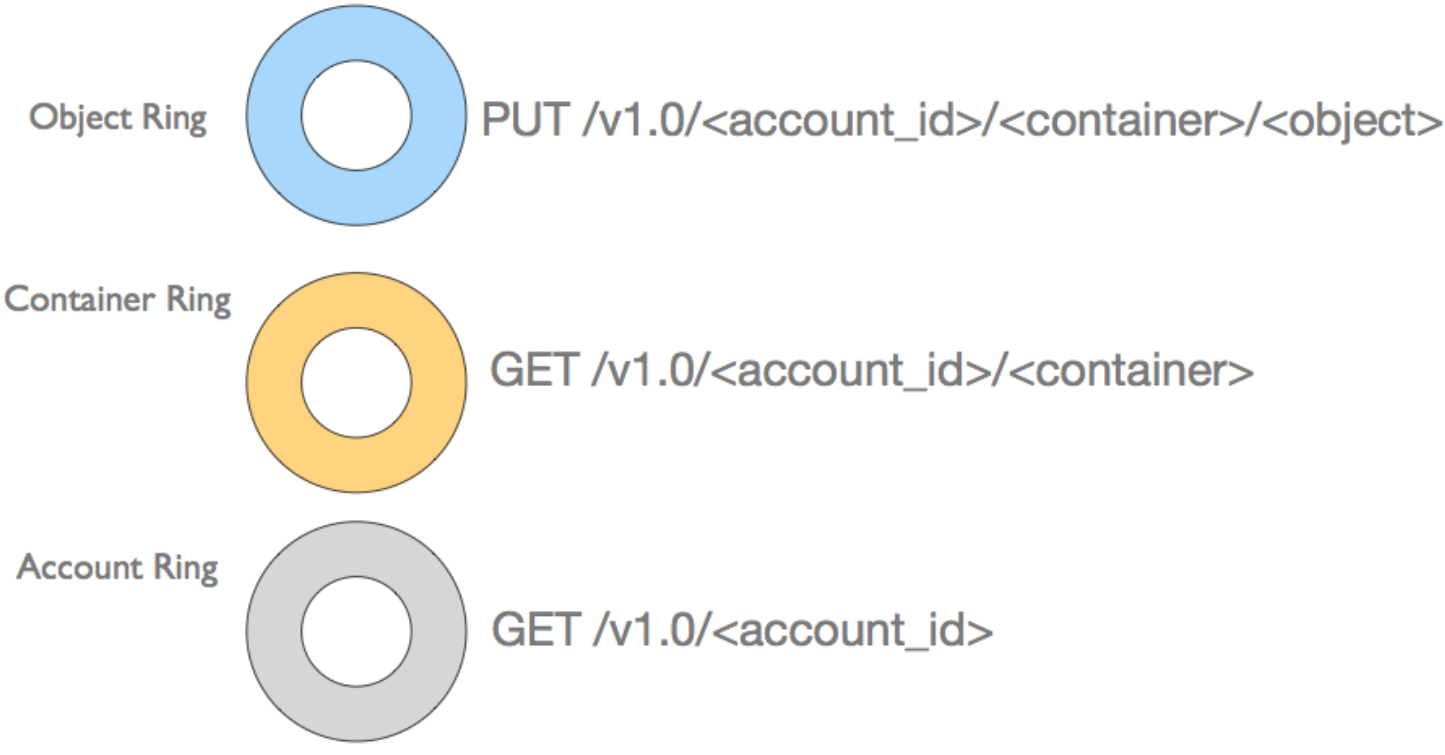
Swift Multi-Region Support

```
sudo swift-ring-builder object.builder add r1z1-192.168.2.1:6000/sda 1000
sudo swift-ring-builder object.builder add r1z2-192.168.2.2:6000/sdb 1000
sudo swift-ring-builder object.builder add r1z3-192.168.2.3:6000/sdc 1000
sudo swift-ring-builder object.builder add r1z4-192.168.2.4:6000/sdd 1000
sudo swift-ring-builder object.builder add r1z5-192.168.2.5:6000/sde 1000
sudo swift-ring-builder object.builder add r2z1-66.150.45.1:6000/sda 1000
sudo swift-ring-builder object.builder add r2z2-66.150.45.2:6000/sdb 1000
sudo swift-ring-builder object.builder add r2z3-66.150.45.3:6000/sdc 1000
sudo swift-ring-builder object.builder add r2z4-66.150.45.4:6000/sdd 1000
sudo swift-ring-builder object.builder add r2z5-66.150.45.5:6000/sde 1000
sudo swift-ring-builder object.builder add r3z1-208.88.32.101:6000/sda 1000
sudo swift-ring-builder object.builder add r3z2-208.88.32.102:6000/sdb 1000
sudo swift-ring-builder object.builder add r3z3-208.88.32.103:6000/sdc 1000
sudo swift-ring-builder object.builder add r3z4-208.88.32.104:6000/sdd 1000
sudo swift-ring-builder object.builder add r3z5-208.88.32.105:6000/sde 1000
```

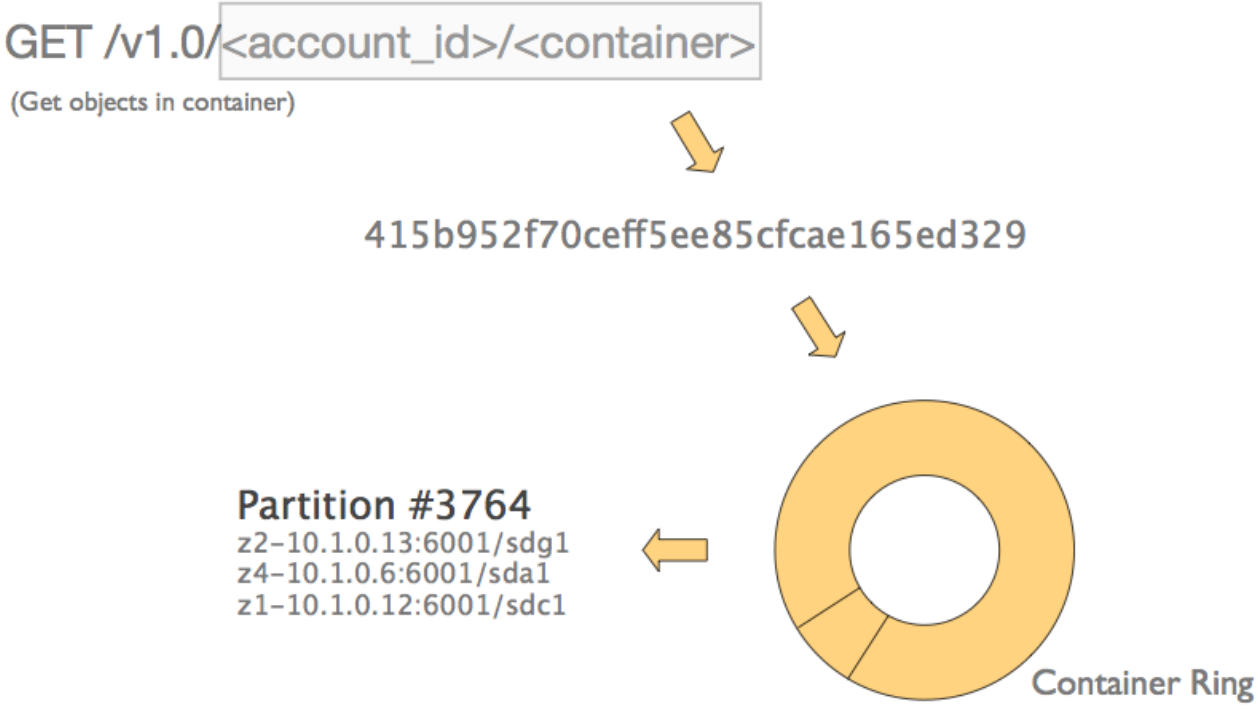




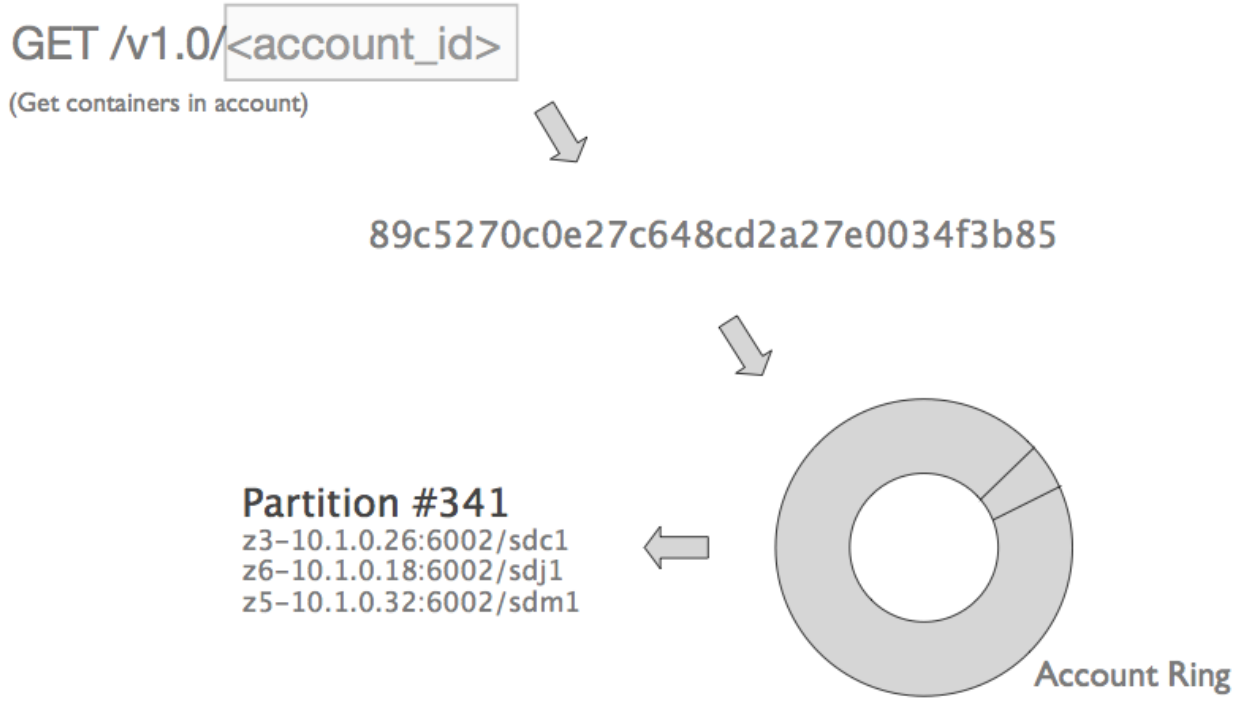
Swift Rings


Object Ring
PUT /v1.0/<account_id>/<container>/<object>
Container Ring
GET /v1.0/<account_id>/<container>
Account Ring
GET /v1.0/<account_id>

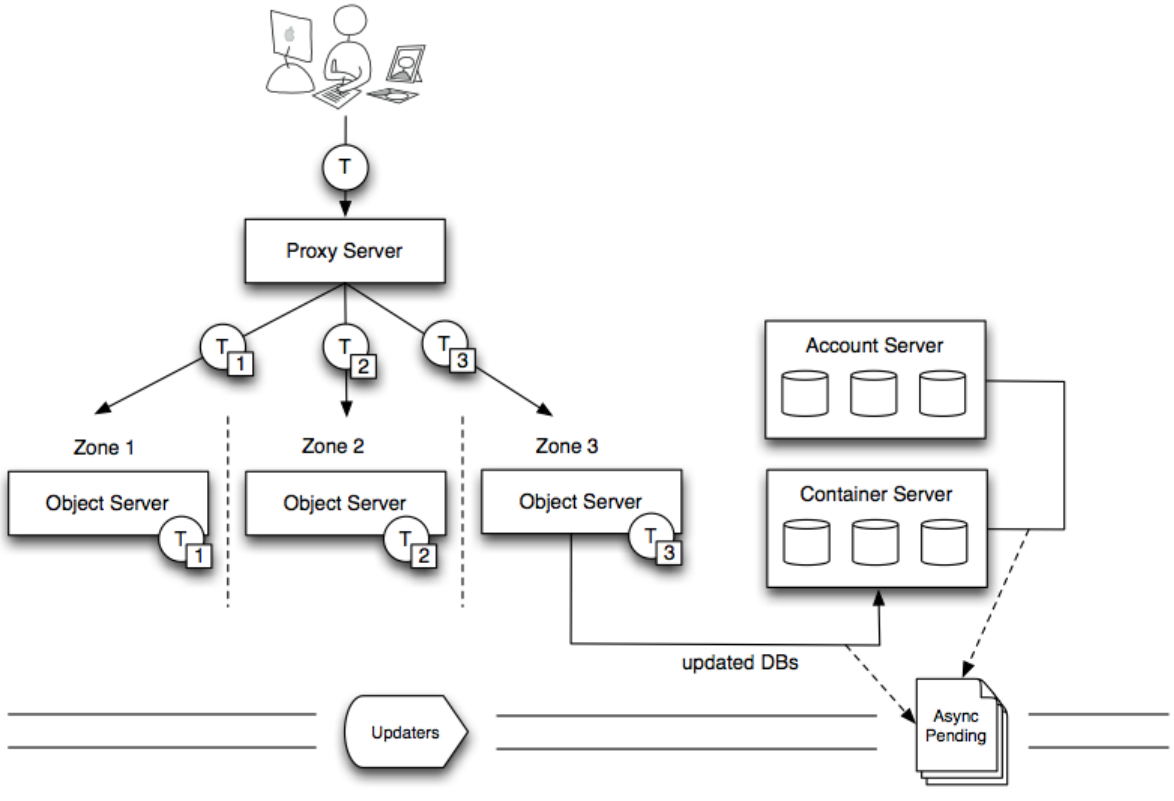
Container Ring



Swift


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
Account Ring

Updating Accounts & Containers



Swift 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

Swift Components

Container Server:
Handles listing of objects, stored as a SQLite DB

Account Server:
Handles listing of containers, stored as a 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

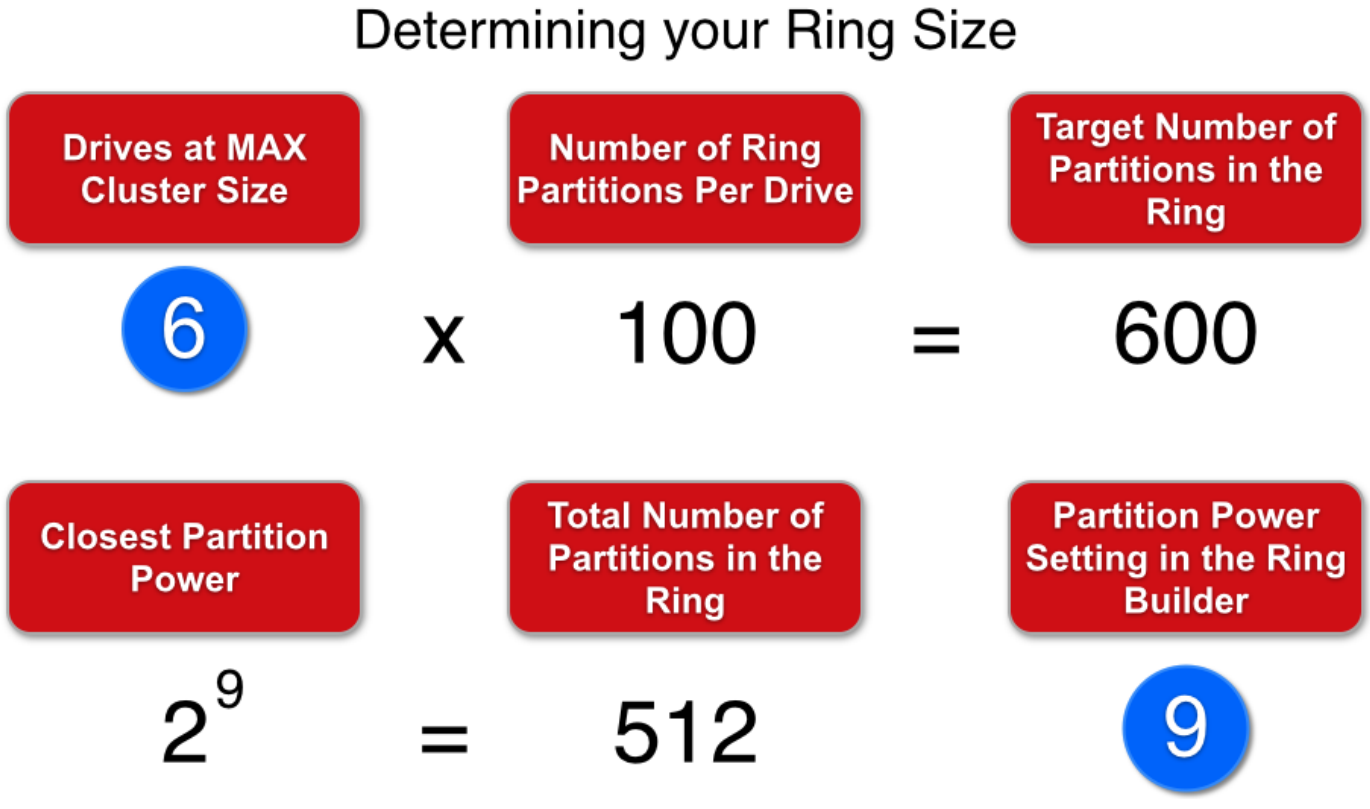
Exercise:

Swift Preliminary Setup

Creating The Ring

1. Determine the approximate number of storage locations at max cluster size
2. Determine the number of replicas (3 is a good choice)
3. Create a builder file based on our determinations above
4. Add storage locations to the builder file
5. Generate the ring from the builder file
6. Push the generated rings to all nodes in the cluster
7. Start uploading objects into the cluster!

Partition Power



Exercise:

Swift Ring Creation

Swift Client

Show storage stats for a user:

```
swift stat
```

Upload a file

```
swift upload yourcontainer yourfile.txt
```

Download a file

```
swift download yourcontainer yourfile.txt -o outputfile.txt

swift download yourcontainer yourfile.txt -o -
```

Exercise:

Swift Client

Swift Authentication

Swift originally had its own authentication system (prior to Keystone) and now allows authentication to be handled by an external system or subsystem (such as Keystone) that follows this basic behavior:

- The authentication system or subsystem follows the Python WSGI (Web Server Gateway Interface) specification.
- A Swift user passes an auth token with each request to Swift
- Swift validates each token with the authentication system and caches the result
- The token need not change from request to request, but does expire after a configurable period of time.

In this course, we assume the use of Keystone as the authentication system.

When using Keystone for authentication, Keystone *projects* (or the older term, *tenants*) are mapped to Swift *accounts*.

Users given roles of *admin* or *swiftoperator* in the authentication system are given the role of *operator* in Swift. Other users have no privileges in swift until they are granted using Swift ACLs.

Access Control Lists (ACLs)

Read and Write ACLs

- `swift post -r <READ RULES>`
- `swift post -w <WRITE RULES>`

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)

Access Control Lists (ACLs)

ACLs can be combined

```
.r:*,.r:-specifichost.specificdomain.com
```

```
testaccount:test1,testaccount:test2
```

ACLs evaluated left to right, last ACL wins

- Bad -- still allows specifichost:

```
.r:-specifichost.specificdomain.com,.r:*
```

- Good: allows anyone except specifichost:

```
.r:*,.r:-specifichost.specificdomain.com
```

Exercise:

Swift Access Control Lists

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:

Swift Quarantine

Replication

- 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:

Swift Replication

Dispersion Report

- Basic utility for measuring overall cluster health
- Verifies a set of deliberately distributed containers and objects are currently in their proper places within the cluster
- A single object's health, especially an older object, usually reflects the health of that entire partition the object is in
- Enough objects on a distinct percentage of the partitions in the cluster, give a valid estimate of overall cluster health
- 1% partition coverage balances well between accuracy and the time to gather results

Exercise:

Swift Dispersion Report

Drive Auditing

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

Exercise:

Swift Drive Audit

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

Monitoring

Lots of metrics!

– Host / Network (traditional monitoring)

- Cabinet uplinks
- Proxy interfaces
- Load Balancer interfaces

– Log trawling

- Bytes in, out, GETs, PUTs, POSTs, etc
- Proxy response codes
- Replication times

– Swift-specific monitoring

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

Exercise:

Swift Recon