The speaker notes are not yet complete, but will be re-updated with final remarks before and after the presentation.
The speaker notes are not yet complete, but will be re-updated with final remarks before and after the presentation.
RGW S3: Feature Progress, Limitations & Testing

Robin H. Johnson, DigitalOcean
Ali Maredia, Red Hat
Contents

- Definitions
- AWS changes, and AWS vs. RGW
- S3-tests: is, is not, what’s missing?
- Deletion, Garbage Collection, Lifecycle
- The Future of Testing RGW & S3
- S3 Global Ecosystem

Red Hat
Background: Robin H. Johnson

- Senior Engineer, Spaces (Ceph/RGW) at DigitalOcean
- Wrote (most) of RGW static website for DreamHost
  - Credit to Yehuda Saleda for early work
- Gentoo Linux core developer (since 2003)
- github.com/robbat2

See GitHub & LinkedIn for other contact methods.
Background: Ali Maredia

- Software Engineer at Red Hat
  - RGW, a maintainer for S3-tests
- Software Engineer at CohortFS
- github.com/alimaredia

Started working on Ceph at a company called CohortFS. In 2015 the CohortFS team joined Red Hat to work on Ceph.
Quick terminology

- S3: the protocol itself
- Specification: Public AWS S3 API document
- AWS-S3: shortened to AWS
- RGW-S3: shortened to RGW
- S3 API calls may include specific features in their requests
- S3 API calls may have only immediate or persistent impact
Specification

- Amazon publishes a single API specification as:
  - Amazon Simple Storage Service, API Reference, API Version 2006-03-01
- The version number has never been bumped
- Document history is a high-level summary only
- No public itemized list of changes known

If anybody has a more detailed changelog for the API specification, it would be invaluable; there is a high-level list of changes
S3: AWS vs RGW - a recap

○ Storage: configured per-object, persistent
○ Access: specific to the upload/download process
○ Services: interact with buckets/objects indirectly

Recap of last year's presentation, since this is built on it

Breakdown of how S3 works in pillars:
- Storage covers how objects are kept, indexed, mutated
- Access: how is it we actually interact with the stored objects
- Services: Meta operations about buckets & operations; it might be immediate OR later on.
AWS S3 Functionality (as of 2018/Feb)

- **Storage:** configured per-object, persistent
  - ACL, Expiration, SSE, Storage Classes**, Tagging, Versioning
- **Access:** specific to the upload/download process
  - Accelerate, Browser POST, CORS, Policy, requestPayment, STS, torrent, website
- **Services:** interact with buckets/objects indirectly
  - Analytics, Inventory, Lifecycle, Logging, Metrics, Notification, Replication

My slide from last year to start with! The state of AWS as of Cephalocon 2018 in Beijing
STS: secure token service, short-lived keys
Analytics: Storage Usage (age of data, what fraction of aged data is accessed, object size statistics)
Metrics: CloudWatch metrics on activity
Replication: user-configured bucket-to-bucket replication
Features: AWS vs RGW (Luminous)

- **Storage:** configured per-object, persistent
  - ACL, Expiration, SSE, Storage Classes**, Tagging, Versioning
- **Access:** specific to the upload/download process
  - Accelerate, Browser POST, CORS, Policy, requestPayment,STS, torrent, website
- **Services:** interact with buckets/objects indirectly
  - Analytics, Inventory, Lifecycle, Logging, Metrics, Notification, Replication

---

Same slide as last year; Formatted slightly differently.

** = exists but is implemented differently or partially

Underlined/Red: New in this version

Strikeout = in AWS, but not RGW
Features: AWS vs RGW (Mimic)

- **Storage**: configured per-object, persistent
  - ACL, Expiration, SSE, **Storage Classes***, Tagging, Versioning
- **Access**: specific to the upload/download process
  - Accelerate, Browser POST, CORS, **Policy**, requestPayment, STS, torrent, website
- **Services**: interact with buckets/objects indirectly
  - Analytics, Inventory, Lifecycle, Logging, Metrics, Notification, Replication

---

A little later than Beijing Cephalocon

** = exists but is implemented differently or partially

Underlined/Red: New in this version

Strikeout = in AWS, but not RGW
AWS S3 New/Changes (as of 2019/May)

- Storage: configured per-object, persistent
  - Object Lock, ACL*
- Access: specific to the upload/download process
  - Region-specific behavior*, Bucket names*, v4-signature*, path-style*, DevPay
- Services: interact with buckets/objects indirectly
  - SELECT, Batch, Intelligent Tiering, Logging*

Object Lock also known as Legal Retention
ACL: subtle change in making emails not valid in ACL in some regions
Region-specific behaviors: AWS always had that, it continues to vary slightly
Bucket Names: us-east-1 is now stricter on bucket names
Signatures: v2-signature depreciation
Path-style: recently in the news, that path-style support was going to be turned off; and then the change was scaled back & postponed
SELECT: SQL-service about bucket structure/index, as well as Athena [Presto]
S3 Batch Operations: *Huge for doing complex stuff offloaded to S3 itself; previously the only batch operation was MultiDelete

RGW should return 405 lost on data that got lost: RtnCodeLostDataRtnCodeLostData
Lifecycle GOT faster: “Amazon S3 removes an expired object delete marker no sooner than 48 hours after the object expiredAmazon S3 removes an expired object delete marker no sooner than 48 hours after the object expired”

DevPay gone entirely!
Replication: cannot be used with some SSE modes!
Notifications: interactions with SSE
Features: AWS vs RGW (Nautilus)

- **Storage:** configured per-object, persistent
  - ACL, Expiration, SSE, **Storage Classes**, Tagging, Versioning
  - Object Lock, ACL*

- **Access:** specific to the upload/download process
  - Accelerate, Browser POST, CORS, Policy, requestPayment, **STS**, torrent, website

- **Services:** interact with buckets/objects indirectly
  - Analytics, Inventory, Lifecycle, Logging, Metrics, **Notification**, Replication**
  - SELECT, Batch, Intelligent Tiering, Logging*

** = exists but is implemented differently or partially
Underlined/Red: New in this version
Strikeout = in AWS, but not RGW
Notification: PubSub
Replication: Archive Sync
**Features: RGW-unique (new in Nautilus)**

- **Storage**: configured per-object, persistent
  - Append Object
- **Access**: specific to the upload/download process
  - BEAST, Authentication (OPA, OAuth2, OpenID-Connect)
- **Services**: interact with objects indirectly
  - Replication: Multisite, Archive Sync

RGW also has some unique features that aren’t in AWS, and Nautilus added to this
What s3-tests IS (part 1)

- (some) Tests for (some) S3 behaviors
  IMPLEMENTED in RGW
- Run regularly and as regression suite for releases
- Implemented using Boto (most in v3 some in v2)

S3-test is for testing for S3 implementations

Boto - it's a Amazon supported client, which in our case we're using to test the Amazon supported client behaviour in our server.
What `s3-tests` IS (part 2)

- Also Go & Java tests (Outreachy interns!)
  - Nanjekye Joannah @Captain_Joannah (2017 May-August)
  - Antoaneta Damyanova (2018 May-August)

Go tests -

Java tests - a few years ago, issues streaming HMAC 256 uploads, users using the Java SDK that caught AWS V4 signatures. Hadoop and Spark SDA use it.

Decoding the expiration header in the java sdk

Guarantee with AWS SDKs major. Java and python have higher and lower level APIs. Some of the tool kits have lower level APIs
What s3-tests IS NOT (part 1)

- Not intended to cover RGW backend specifics
  - RADOS Class operations
  - RGW admin operations (zonegroups etc.)
- RGW Performance
  - Performance of buckets at scale, deletions, lifecycle & more

- s3-tests does NOT cover RGW backend specifics like rgw-admin operations on zones and zonegroups for example.
  - OR on RADOS Class operations (CLS aka Obj class).
- s3-tests does not cover the RGW at scale, or performance
What s3-tests **IS NOT** (part 2)

- **Testing RGW-unique S3 functionality**
  - Bucket Notifications
    - PubSub instead of AWS SNS
  - Metadata Search
  - GetObjectLayout, AppendObject

- s3-tests does not cover RGW unique features with S3 functionality
  - an example of this are Bucket Notifications. In the RGW we have PubSub instead of AWS SNS.
  - others include Metadata Search, GetObjectLayout, AppendObject
What s3-tests **IS NOT** (part 3)

- Cover grey areas in the S3 specification
  - lack of or unclear definition
- Run AGAINST AWS regularly
  - Should PASS on AWS first
- 501-NotImplemented:
  - CopyObject-SSE
  - v4 signatures on some operations
  - STREAMING-AWS4-HMAC-SHA256-PAYLOAD for non-PUT

- S3 tests doesn't cover the grey areas in the S3 specification
- Some examples will be coming in the coming slides
- s3-tests is not run against AWS regularly. Tests written aren't written and run on AWS first then the RGW second.

RGW returns 501 NotImplemented to some very specific requests
Concrete examples of s3-test misses

- **Headers:** CopyObject `x-amz-copy-source`
  - URL encoding but what! Query param vs path

```
PUT /destinationObject HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
x-amz-copy-source: /source_bucket/sourceObject
```

The name of the source bucket and key name of the source object, separated by a slash (`/`).
This string must be URL-encoded.

The text just says URL encoding, without clarity about path vs queryparam
URL encoding spec specifies DIFFERENT rules for path vs query parameter
`/` is NOT encoded in path segment
But IS encoded in query parameter, as `%2F`
Other characters also have differing rules, like whitespace as `%20` OR a literal plus (only in queryparam!)
Concrete examples of s3-test misses

- Body: Complex variations of Lifecycle Policies
  - XML elements changed!

```
<Rule>
  <ID>delete-all-glacierobjects-in-30-days</ID>
  <Prefix>glacierobjects</Prefix>
</Rule>

<LifecycleConfiguration>
  <Rule>
    <Filter>
      <Prefix>key-prefix</Prefix>
    </Filter>
  </Rule>
</LifecycleConfiguration>
```

Lifecycle Policies:
- AWS spec not clear on interactions between some rules
- Testing various editions of Policy, because the XML formats have changed!
- Rule/Prefix, changed to Rule/Filter/Prefix; because tags came later!
Concrete examples of s3-test misses

- Header/Body interactions
  - CreateBucket v4 signature SHA256 bug
  - Browser POST: content-length-range bits

CreateBucket v4 signature
Similar to https://tracker.ceph.com/issues/20463 that was for ListObjects
TODO: file a Tracker issue for this one from mailing list notes & Robin’s work ticket

Browser POST policies:
Allow specification of what an end-user client can upload back to RGW; and encode those requirements into the signed URL… but then it interprets some of the policy wrong
RGW: RADOS Object Layout

- Small RGW objects: (<4MiB approx)
  - Head-only, no tails
- Large RGW objects, without multi-part:
  - Head
  - Striped tails
- Large RGW objects, with multi-part:
  - Head (Manifest in xattrs, no data) [Initiate Multipart Upload]
  - Parts (Optionally with stripes) [Upload Part]

A single RGW file is almost certainly spread data over more multiple RADOS objects. The only case where it’s a single 1:1 RGW objects to RADOS object is small files. Approximately less than 4MiB; the approximately is because there is some accounting for overhead.

The exactly boundary between head & tail parts varies depending on your configurations!
RGW Delete: Performance

- Synchronous part:
  - Update of index
  - Delete of head in the data pool
  - Write entry into GC, with all the tails listed
  - Delete of an empty RADOS object is faster than not-empty!

- Asynchronous:
  - Tails garbage collected!
  - No inline write to the data pool at all
  - OMAP keys/values written to objects in GC pool

So what happens when you actually call DeleteObject?
RGW Delete: how to go faster?

- Heads
  - Go Manifest everywhere, in a dedicated pool (on faster OSD)
  - Bonus: Metadata-only requests get big boost

- Synchronous
  - Update Index & Manifest
  - Write path to Manifest to GC

- Asynchronous
  - Read Manifest name from GC queue
  - Fetch tails from Manifest to actually delete

Both the index & manifest should state that the object is pending deletion, so that if you have to rebuild the index you know that the object is pending deletion. This would not otherwise be known if the pending flag was only in the index. It needs to be in the index so that you don’t need to read the object. Need to figure out some cases around overwrites here still. (Potentially copy the Manifest to the GC?)

radosgw-admin 'object rewrite' already exists to rewrite how RGW objects are stored as RADOS objects
RGW always-Manifest-on-different-pool: other bonuses

- Faster Metadata!
  - Lifecycle scan
  - Conditional HTTP requests
  - Index reads that sometimes hit objects in interim states
  - Glacier Storage class: retain the metadata online but the data offline, RestoreObject to bring it back

Having manifest-only heads, with other metadata (in the xattrs of the head RADOS object) provide other boosts to performance!
RGW GC: problems (part 1)

- Large backlog of customer deletions
  - Pending deletions eat space! (one cluster below)

- Line is the 12 hour average
- Note that this graph is the net result of direct customer deletions as well as Lifecycle deletions
- Includes a 24 hour GC deferral
- Lots of uploads in the peak hours
- Cleanup catches up overnight & weekends
RGW GC: problems (part 2)

- GC runs one thread per RGW instance :-(
- No dynamic controls for GC thread
  - Copied-by-value, interactions to stopping/starting
- Workarounds by running `radosgw-admin gc process`
  - Needs minor patching for granular shard control
  - But watch out...

- GC not without performance issues as well:
  - Hard to dynamically stop/start without restarting the instance. Cannot use the actual GC threads config knob, instead have to be creative with the GC variables to make it a no-op.
  - Very little insight data into the distribution & processing of GC entries.
  - Dumping 'gc list' for stats is expensive.
  - Large backlogs of customer deletions
RGW GC: problems (part 3)

- GC process hits OSDs hard

This is not all the pools, I trimmed some off the image.
RGW GC: problems (part 4)

- No insight for distribution of GC queue
  - How much work? How fast is it going?
- Running `gc list` is very expensive
- Moving GC threads out of RGW daemon
  - New daemon with more control & stats

- Very little insight data into the distribution & processing of GC entries.
- Dumping 'gc list' for stats is expensive.
- Moving GC threads out of RGW and into dedicated daemon/tool
Lifecycle: Background

- Big thank you to Yehuda, Matt & others for Nautilus updates
  - Cleanups, better XML parsing
- Clients declare an Lifecycle policy that:
  - Describes transitions (deletions, classes)
  - Based on age, path prefix, tags
How does lifecycle work right now:
- Creating or modifying a lifecycle policy inserts a new OMAP k/v into the lifecycle shards, with the state of UNINITIAL
- The policy itself is stored in the bucket metadata
- States
  - UNINITIAL
  - PROCESSING
  - COMPLETE
  - FAILED

How does lifecycle work right now:
- Creating or modifying a lifecycle policy inserts a new entry into the LC list, with state UNINITIAL
- for all buckets with a lifecycle policy:
- every N time (daily): reset lifecycle state to UNINITIAL (not started)
- states: UNINITIAL->PROCESSING->COMPLETE/FAILED
How does lifecycle work right now:
- for each bucket that is not yet complete:
  - visit all items, in a single thread, in some order
  - until a max timer is reached, then continue to next bucket

Lifecycle: How does it work?
- Every N time
- For every key in the LC OMAP
- Daily: Reset state to UNINITIAL
- If not COMPLETE, set PROCESSING, and:
  - For each LC Rule prefix
  - List the entire bucket for that prefix
  - Check mtime and/or metadata (reads object xattrs)
  - Maybe do something to the object
  - Break if max time hit
**Lifecycle: RGW gains**

- **Mimic**:  
  - Speedup of traversal by unordered listing  
  - Object tags
- **Nautilus**:  
  - Storage class transitions  
  - Lots of cleanups (thanks again)
- **Octopus**:  
  - Run hook/lambda at transition?
- Metadata accesses: loading metadata means hitting xattrs from potentially slow OSDs (spinning EC)
- Large bucket pain
- For sufficiently large buckets with slow index scans, items returned early in the index order will be correctly evaluated for deletion, but those later might never be processed :-(

Lifecycle: Problems

- Metadata access expensive
- Not enough threads: code from GC, 1 thread/daemon
- Large buckets
  - Sufficiently large buckets (with slow index list)
  - Might NEVER expire objects late in the listing
- No visibility to outstanding work/progress
Lifecycle: Current & Future improvements

- Workarounds include custom coding around 'lc process'
  - Specifically try to run more buckets in parallel, but let each bucket run longer

- Better solutions could include remembering the approximate index marker position to try and resume at (usable for partial ordering)
- Moving LC threads out of RGW and into dedicated daemon/tool
The future of testing RGW

- Rolling upgrades have been troublesome
  - Mismatched OSD and RGW versions
  - Mismatched cluster versions over multisite

- Consistent testing of background behaviours
  - GC, LC, Quotas
The future of testing S3

- Uniform Coverage of S3 SDKs
- Coverage of entire S3 specification
- Beyond SDKs: Coverage of AWS behaviors
  - Those critical to SDK/S3 clients
  - Those covering older SDK choices

- Mention that AWS ships 13 different official SDKs and growing!
- That’s not counting community SDKs like Fog
- AWS behavior continues to support older SDK versions, even after the docs are updated!
- Concrete example: Old versions of Boto work! Even with the signature bug!
Roadmap of S3 testing

- **June 2019:**
  - Python3 port, v2 & v4 signatures, Java & Go testing

- **Summer/Fall 2019:**
  - Support for python in a multi-language testing framework
  - `s3-tests --framework boto3`
    - `--suite BucketLifecycle,MultiDelete`
    - `--options signature=v4-sts,callingformat=path`
    - `--target server.yml`
**Roadmap of S3 testing (part 2)**

- **Ceph Octopus release:**
  - Support for other languages (ex: java & go) in a multi-language testing framework, to provide test matrix!
  - `s3-tests --framework boto3,aws-sdk-go-v2,fog` \
    --suite BucketLifecycle,MultiDelete \
    --options signature=v4-sts,callingformat=path \
    --target server.yml
What is the S3 Global Ecosystem?

- **Closed-source Public Clouds**
  - AWS
  - GCP
  - Oracle
  - Tencent

- **Open-Source Public Clouds**
  - Ceph RGW (DigitalOcean, DreamHost, China Mobile, UMCLOUD, xSky)
  - Minio (maybe Alicloud??)
  - OpenIO (Scaleway)

Know what implementations other large S3 providers are offering?
Esp. interested in open-source offerings NOT listed above
Testing the S3 Global Ecosystem

○ Collaboration between the S3 ecosystem members?
  ○ Common test platforms / inter-operability labs
Questions from the Session (1/n)

Dan van der Ster:
- Would it be possible to add an API call to query quota usage?
- Matt Benjamin & Yehuda answers Yes, we can do that.
- Robin: clarify that this is an RGW API, not an S3 API
Questions from the Session (2/n)

?? (didn’t catch the name)
- Q: Is there any plan to get a formal IETF/RFC S3 specification
- Robin answers: Maybe, as of 2018 Cephalocon Amazon didn’t give the time of day to requests about this. Recently got a new contact, so can revisit
- Suggests something similar to the IETF IMAP Keyword registry, rather than a single large spec, many small specs.
Questions from the Session (3/n)

Last person on the left of the room:
- Q: what about testing some of the RGW specifics?
- Robin: future here is for a separate RGW suite, since many of these don’t touch S3 API at all
- Teuthology has some tiny pieces that DO touch this already; and some of these tests would probably be called from that.
Interested in contributing?

https://github.com/ceph/s3-tests/
Thank you!

Robin H. Johnson
rjohnson@digitalocean.com
github.com/robbat2
IRC: robbat2

Ali Maredia
amaredia@redhat.com
github.com/alimaredia
IRC: amaredia