DigitalOcean

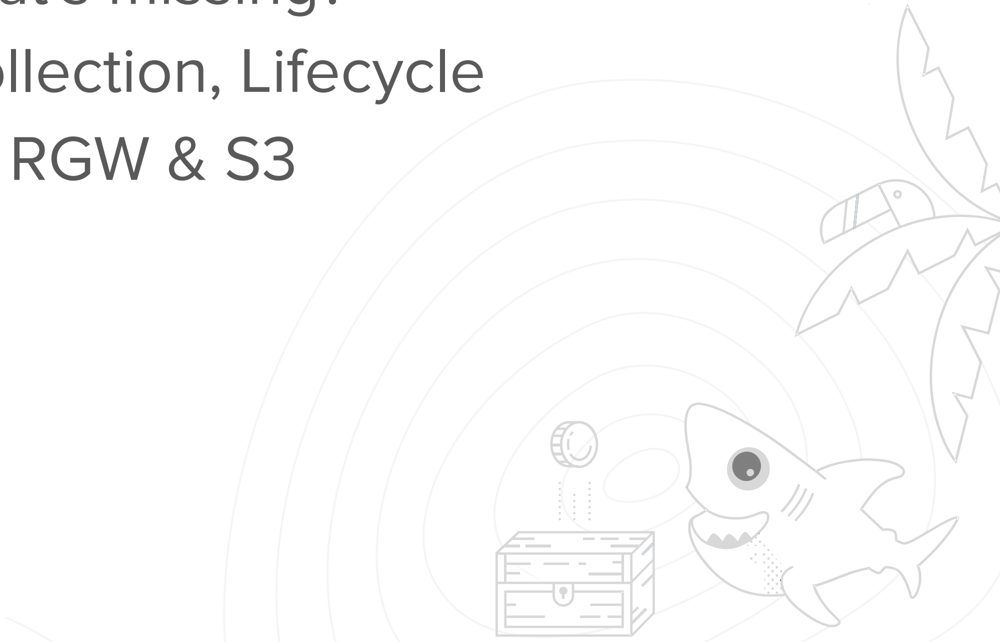
Red Hat

RGW S3: Feature Progress, Limitations & Testing

Robin H. Johnson, DigitalOcean
Ali Maredia, Red Hat

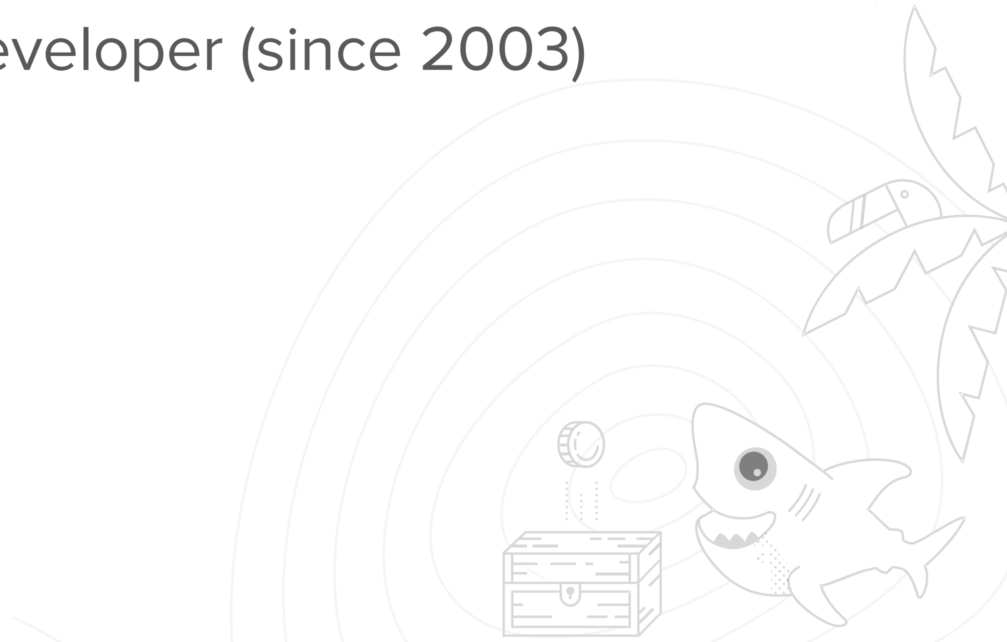
Contents

- 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



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

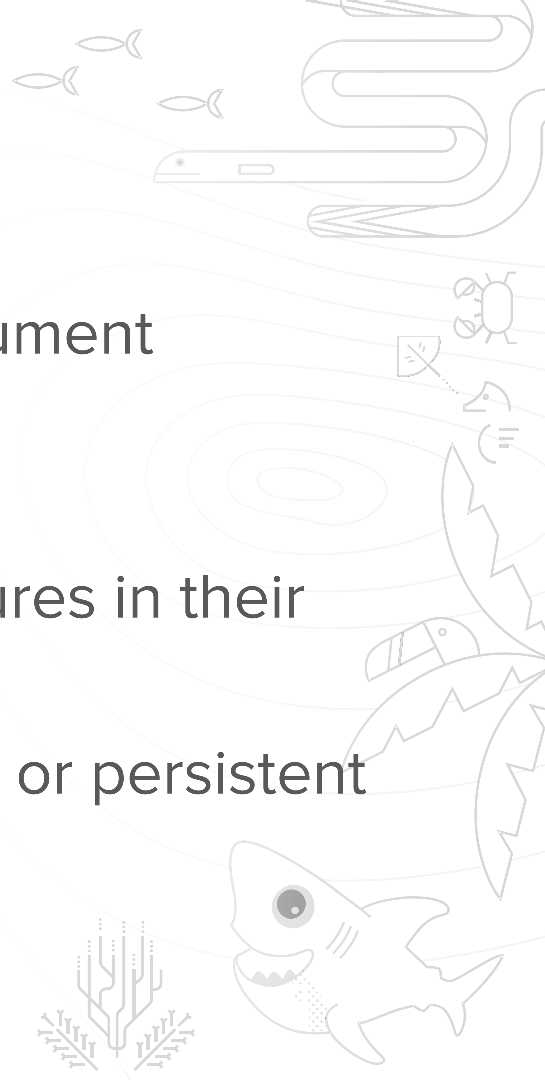


Background: Ali Maredia

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

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

S3: AWS vs RGW - a recap

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

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

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~~

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~~

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*

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*~~

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

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)

What s3-tests IS (part 2)

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

Red Hat

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

What s3-tests IS NOT (part 2)

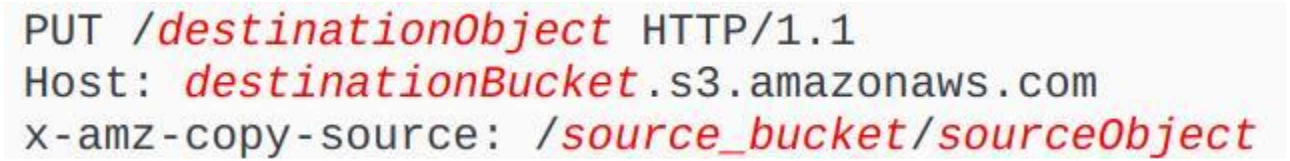
- **Testing RGW-unique S3 functionality**
  - Bucket Notifications
    - PubSub instead of AWS SNS
  - Metadata Search
  - GetObjectLayout, AppendObject

What s3-tests IS NOT (part 3)

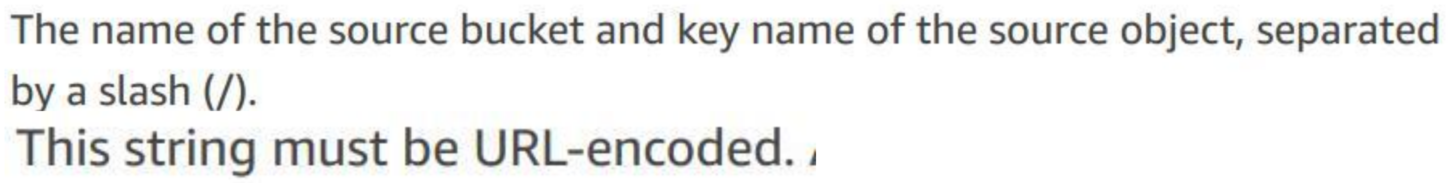
- Cover grey areas in the S3 specification
  - lack of or unclear definition
- Run AGAINST AWS regularly
  - Should PASS on AWS first
- 501-NotImplemented:
  - CopyObject-SSE
  - v4 signatures on some operations
  - STREAMING-AWS4-HMAC-SHA256-PAYLOAD for non-PUT

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
```

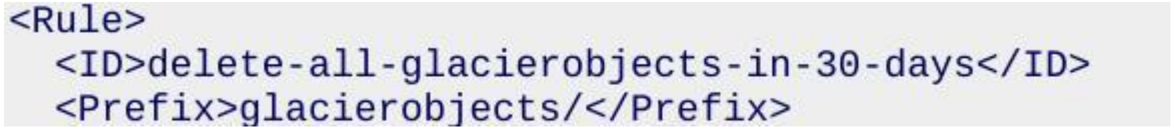


The name of the source bucket and key name of the source object, separated by a slash (/).

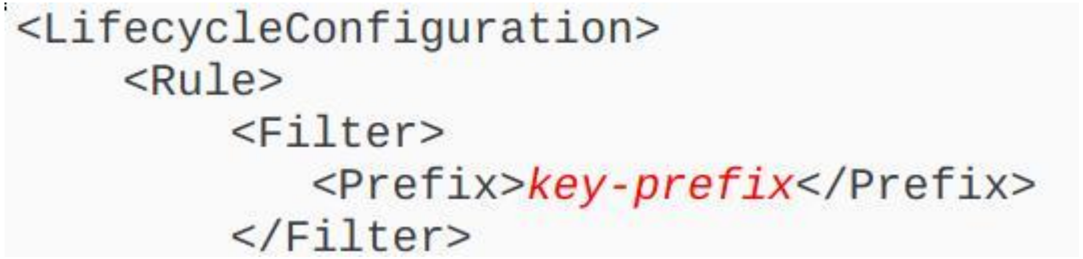
This string must be URL-encoded.

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>
```



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

Concrete examples of s3-test misses

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

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]

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

RGW Delete: how to go faster?

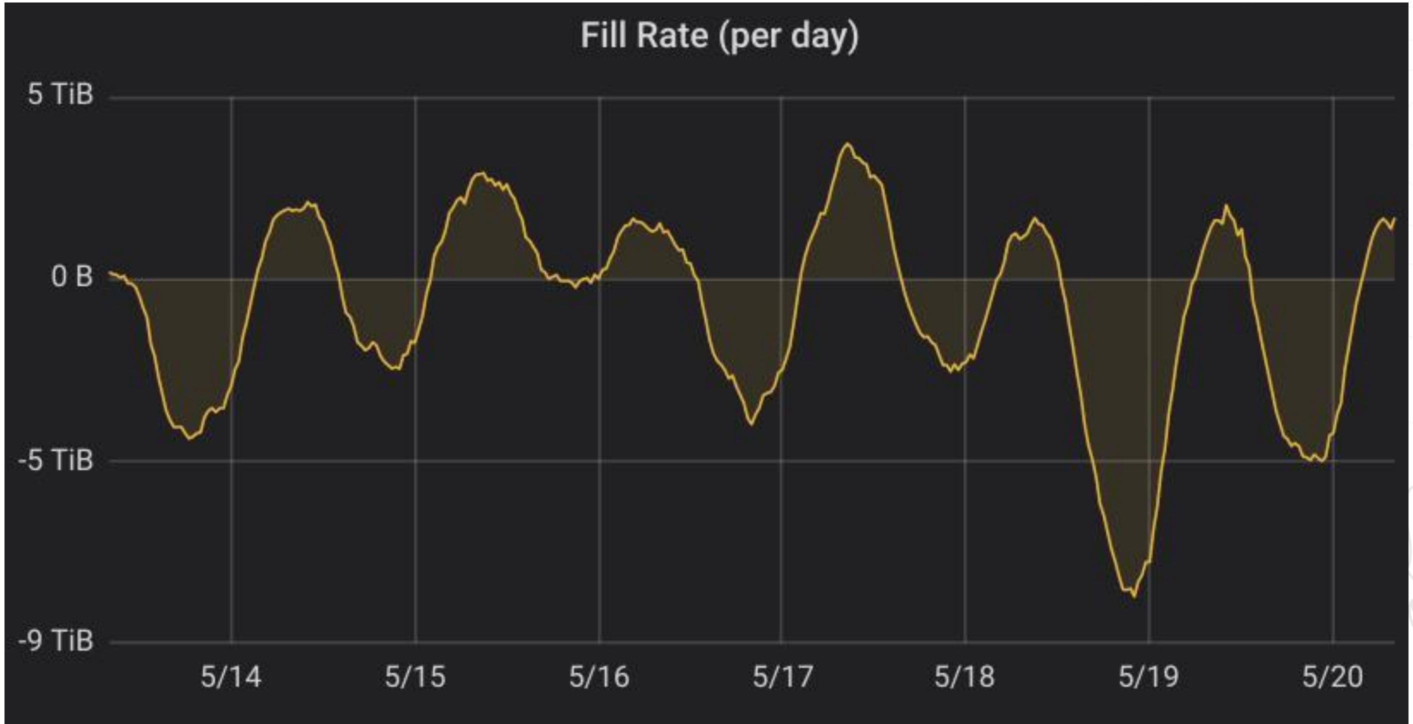
- Heads
  - Go Manifest everywhere, in a dedicated pool (on faster OSD)
  - Bonus: Metadata-only requests get big boost
- Synchronous
  - Update Index & Manifest
  - Write path to Manifest to GC
- Asynchronous
  - Read Manifest name from GC queue
  - Fetch tails from Manifest to actually delete

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

RGW GC: problems (part 1)

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

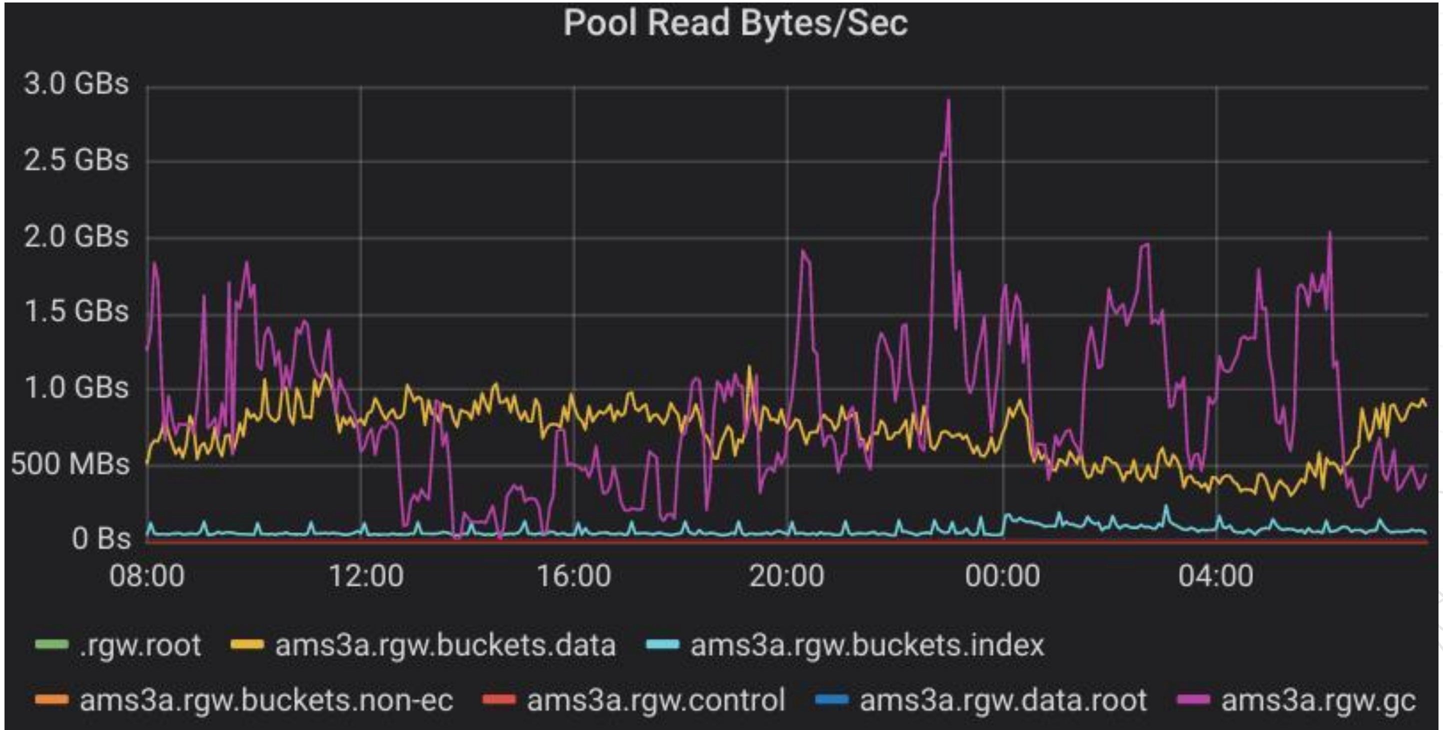


RGW GC: problems (part 2)

- 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...

RGW GC: problems (part 3)

- GC process hits OSDs hard



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

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

Lifecycle: How is it stored?

- Creating or modifying 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

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?

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

- Custom `radosgw-admin lc process` supervisor
  - Trivial but messy patch to run LC on a single bucket
- Index marker for resuming on a given bucket
- Secondary storage PER prefix, ordered by time
  - Trivial queue?
  - Write cost to maintain index vs cost of repeated full-bucket scans
- Dedicated daemon
- Testing! Some s3-tests coverage, but not enough...

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

Red Hat

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
    ```

Red Hat

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)

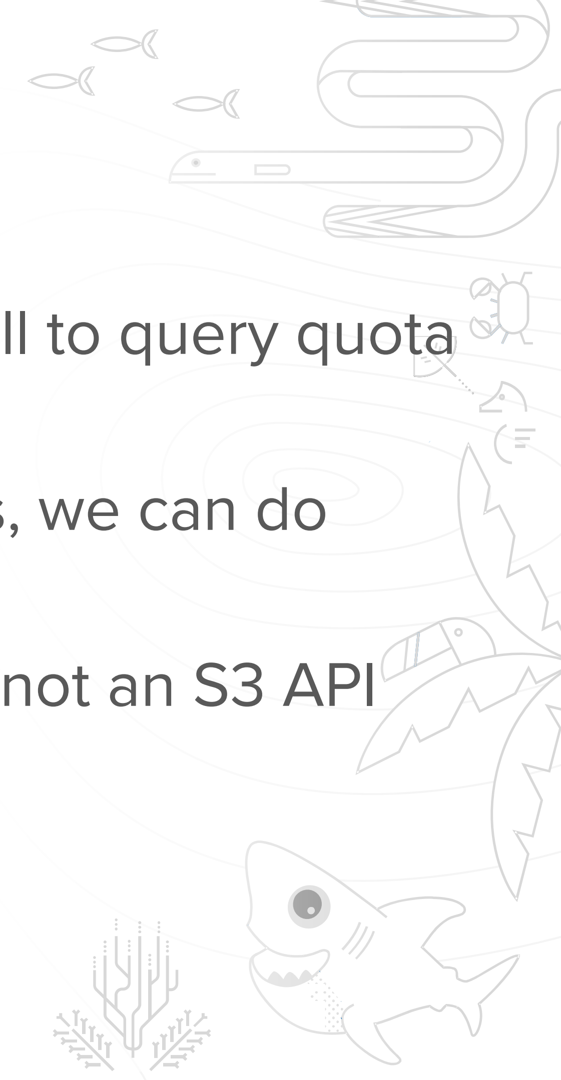
Testing the S3 Global Ecosystem

- 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 seperate 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

Red Hat