Cloud object storage: the right way

Orit Wasserman
Open Source Summit 2018
About me

● 20+ years of development

● 10+ in open source:
  ○ Nested virtualization for KVM
  ○ Maintainer of live migration in Qemu/kvm

● 4 years as Ceph core developer at Red Hat

● Architect at lightbits labs
Cloud object storage: the right way

- Introduction to cloud object storage
- Features:
  - Multipart upload
  - Versioning
  - Life cycle
  - Prefix
  - Static website
- Security
- DR
- Summary
Introduction to cloud object storage

The Evolution of Data Storage
Object storage

- Flat namespace
- Objects are immutable
- Range Read
- Rich Metadata:
  - Ownership (Users and tenants)
  - ACL
  - User metadata
Cloud object storage

- Restful API
- Common clouds:
  - AWS S3
  - Swift (openstack)
  - Google cloud storage
  - Azure blob storage
  - Ceph
  - Digital Ocean
Example: Media
**Example: Documents**

**Manage versions**

Drive keeps older versions of 'Fosdem-RGW.pdf' for 30 days. [Learn more]

<table>
<thead>
<tr>
<th>Current version</th>
<th>Fosdem-RGW.pdf</th>
<th>Jan 31, 2016, 12:21 AM</th>
<th>Ort Wasserman</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1</td>
<td>Fosdem-RGW.pdf</td>
<td>Jan 28, 2016, 5:46 PM</td>
<td>Ort Wasserman</td>
</tr>
</tbody>
</table>
When to use cloud object storage

- Cloud or large scale environment
- Lots of large objects that are rarely updated.
- Small objects that are updated infrequently and are not performance sensitive.
- Hard drives
When not to use cloud object storage

● If the application does lots of inplace writes inside big files.
  ○ Change workload to larger writes
  ○ Divide big file into smaller ones

● Legacy application
  ○ File on object (NFS on RGW, s3fs …)
Cloud object storage features
Multipart upload

- Upload a single object as a set of parts
- Transaction:
  - Initiate
  - Upload parts
  - Complete
Multipart upload

- Improved throughput
- Quick recovery from any network issues
- Pause and resume object uploads
- Begin an upload before you know the final object size
- Instead of FS rename
Multipart upload pitfalls

- Due to the performance impact not recommend for small objects
- Regular upload is up to 5 GB
- Check your framework/SDK defaults!
- Orphans ...
Versioning

- Keeps the previous copy of the object in case of overwrite or deletion
- Problem: space usage
Life cycle

- Configure automatic object transition:
  - Expiration: used to clean old objects, older versions and failed multipart uploads
  - Tiering: move object to colder storage
virtual hierarchy

- Add a prefix to an object
- Listing a sub folder by listing objs with a specific prefix
Static website

Host a static website directly from the cloud object storage
Security
Signature: AWS4

- More secure:
  - Key is not part of the request
  - All requests are signed
  - Streaming support
- Not all SDK use it by default or even support it

1. StringToSign
   A string based on select request elements

2. Signing Key
   - DateKey = HMAC-SHA256("AWS4" + "<SecretAccessKey>"", "<yyyyMMdd>")
   - DateRegionKey = HMAC-SHA256(DateKey, "<aws-region>")
   - DateRegionServiceKey = HMAC-SHA256(DateRegionKey, "<aws-service>")
   - SigningKey = HMAC-SHA256(DateRegionServiceKey, "aws4_request")

3. Signature
   signature = Hex(HMAC-SHA256(SigningKey, StringToSign))
Protocol and transport

- Encrypt the traffic
- High performance penalty
- Options:
  - Tunneling
  - Terminate at the load balancers like HAProxy and use http for your internal network
Encryption

● Server side encryption is not enough
● Use client side encryption:
  ○ SSE-C: Customer provided keys
  ○ SSE-KMS: Key management service
Bucket and Object ACL

- Owner
- System/Admin user
- Other users: Read/Write/Read ACP/Write ACP/Full control
# Canned ACL

<table>
<thead>
<tr>
<th>Canned ACL</th>
<th>Applies to</th>
<th>Permissions added to ACL</th>
</tr>
</thead>
<tbody>
<tr>
<td>private</td>
<td>Bucket and object</td>
<td>Owner gets FULL_CONTROL. No one else has access rights (default).</td>
</tr>
<tr>
<td>public-read</td>
<td>Bucket and object</td>
<td>Owner gets FULL_CONTROL. The AllUser's group (see Who Is a Grantee?) gets READ access.</td>
</tr>
<tr>
<td>public-read-write</td>
<td>Bucket and object</td>
<td>Owner gets FULL_CONTROL. The AllUser's group gets READ and WRITE access. Granting this on a bucket is generally not recommended.</td>
</tr>
<tr>
<td>aws-exec-read</td>
<td>Bucket and object</td>
<td>Owner gets FULL_CONTROL. Amazon EC2 gets READ access to GET an Amazon Machine Image (AMI) bundle from Amazon S3.</td>
</tr>
<tr>
<td>authenticated-read</td>
<td>Bucket and object</td>
<td>Owner gets FULL_CONTROL. The AuthenticatedUsers group gets READ access.</td>
</tr>
<tr>
<td>bucket-owner-read</td>
<td>Object</td>
<td>Object owner gets FULL_CONTROL. Bucket owner gets READ access. If you specify this canned ACL when creating a bucket, Amazon S3 ignores it.</td>
</tr>
<tr>
<td>bucket-owner-full-control</td>
<td>Object</td>
<td>Both the object owner and the bucket owner get FULL_CONTROL over the object. If you specify this canned ACL when creating a bucket, Amazon S3 ignores it.</td>
</tr>
<tr>
<td>log-delivery-write</td>
<td>Bucket</td>
<td>The LogDelivery group gets WRITE and READ_ACP permissions on the bucket. For more information about logs, see (Amazon S3 Server Access Logging).</td>
</tr>
</tbody>
</table>
Be careful of public buckets

198 million Americans hit by 'largest ever' voter records leak
Personal data on 198 million voters, including analytics data that suggests who a person is likely to vote for and why, was stored on an unsecured Amazon server.

GoDaddy Leaks 'Map of the Internet' via Amazon S3 Cloud Bucket Misconfig

Data Centre » Cloud
When it absolutely, positively needs to be leaked overnight: 120k FedEx customer files spill from AWS S3 silo
Passport scans, drivers licenses, etc, exposed online

Verizon partner data breach exposes millions of customer records
Accessed through an unprotected Amazon S3 storage server
Bucket and Users policy

- Access policies for users and buckets:
  - Grant access from multiple accounts
  - Cross account permission
  - Read only for anonymous users
  - Restricting access to a IP specific
Grant access from multiple accounts

```json
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AddCannedAcl",
      "Effect":"Allow",
      "Principal": {"AWS": ["arn:aws:iam::111122223333:root","arn:aws:iam::444455556666:root"]},
      "Action": ["s3:PutObject","s3:PutObjectAcl"],
      "Resource": ["arn:aws:s3:::examplebucket/*"],
      "Condition": {"StringEquals": {"s3:x-amz-acl": ["public-read"]}}
    }
  ]
}
```
Secure Token Service

- Provides a temporary token to access the cloud storage
- Assume rule
- Used by storage class and glacier
Disaster Recovery

OUR DISASTER RECOVERY PLAN GOES SOMETHING LIKE THIS...

HELP! HELP!

SOMEDAY WE HOPE TO HAVE A BUDGET.
Test your DR plan!

Yes Madam, software as a service does mean you won’t need to install software on your computer - but no, it won’t make your laptop any lighter.
Solution: geo replication

- Global object storage clusters with a single namespace
- Enables deployment of clusters across multiple geographic locations
- Clusters synchronize, allowing users to read from or write to the closest one
- Disaster recovery in case of a zone failure
RGW Multisite definitions

- Realm - namespace
- Zone - represent a geographical location, cannot cross clusters
- ZoneGroup - group of replicating zones
- Period - current realm configuration. Updates are local and are only applied when committed.
How does the replication works

**Metadata ops**
- User and bucket updates
- Small amount of data
- Rare updates
- Wide effect
- Synchronous
- Meta master (master zone in the master zonegroup)

**Data ops**
- Objects update
- Large amount of data
- Frequent operations
- Only affects a single object
- Asynchronous
- All zones
RGW default setup

Created automatically first time radosgw runs without any multisite configuration
RGW local configuration

Used to set zonegroup parameters like:

- Default bucket index shards
- Placement target

Realm: myrealm
Zonegroup: local-zg
Zone: local-zone

Radosgw
RGW Simple DR configuration

Realm: myrealm
Zonegroup: us
Zone: us-west

Realm: myrealm
Zonegroup: us
Zone: us-east

Radosgw

Radosgw
RGW Local and replicated data configuration

Realm: myrealm
Zonegroup: local-zg
Zone: local-zone

Realm: myrealm
Zonegroup: us
Zone: us-west

Realm: myrealm
Zonegroup: us
Zone: us-east

Radosgw

Radosgw

Radosgw

Radosgw
Cloud sync

Replicate your data to public cloud for DR
One cloud is not enough

Disaster recovery to a different public cloud

Replicate your private cloud data to public cloud
Metadata search

- API to query based on object metadata
- Integration with ElasticSearch
Summary

- Object storage was designed for large scale and for the cloud
- Use object storage api to get all it advance features.
- Make sure your data is safe!
- Test your DR plan!
- Use Ceph for private cloud object storage!
github.com/oritwas
@oritwas