Optimize Ceph Object Storage for Production in Multisite Clouds

Michael Hackett
Vikhyat Umrao
Principal Software Maintenance Engineers - Ceph
Red Hat, Inc.

Objectives

- High level understanding of Ceph Object Storage
- How to plan for a Ceph Object Storage cluster at scale
- How to address important success criteria--Do’s and Don’ts
- Understand hardware price/performance tradeoffs
- How to get the results you want for the time and resources you invest
- Understanding Ceph Object Storage Multisite Cloud configuration
- How to configure Ceph Object Storage Multisite between two production clusters
- How to manage Multisite clouds using Rados Gateway(RGW)
- How to troubleshoot RGW Multisite issues

About Ceph

Enterprise-class cloud storage

Ceph delivers object, block and file storage on one platform, delivering:

- Scalability from petabytes to exabytes
- High Availability--hardware failure is an expectation, not just an exception
  - Data durability: replication or erasure coding
  - Data distribution: Data is evenly and pseudo-randomly distributed
  - Fault tolerance: Ability to confine failures. No single point of failure.
  - Resilience: Automatic recovery from a degraded state.

Ceph is a very appealing cloud storage solution for OpenStack.

Ceph Object Storage

What is Ceph Object Storage?

Ceph Object Gateway (RGW) provides an object storage service with:

- Well-known RESTful S3 and Swift APIs
- User Management, Tenants, Users, Usage and Quotas
- Multi-site and Failover Capabilities
- Ability to integrate with OpenStack, LDAP/AD
- Multiple Performance Profiles Simultaneously
    - Throughput-optimized
    - Capacity-optimized
    - I/O-optimized

Ceph Object Storage

Why Ceph Object Storage?

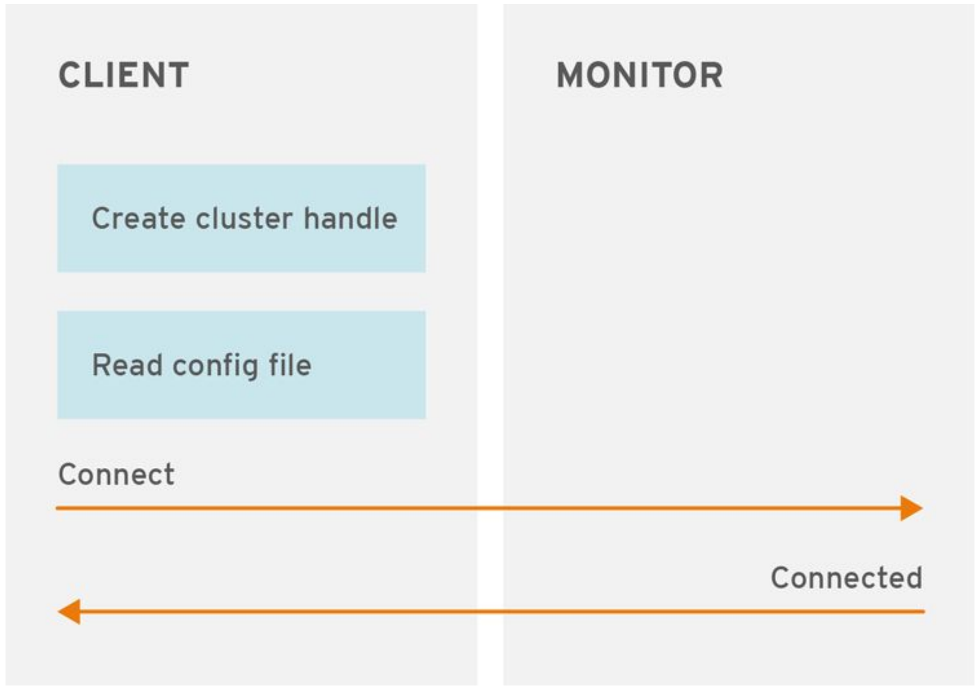
It’s an increasingly important cloud storage solution, because:

- It is ideally suited for unstructured data
  - Streaming Audio and Video, 4K
  - Graphics
  - Imaging
  - Analytics
- Emerging as a foundation for data lakes and analytics solutions
- Disaster Recovery Solution
- The same skills required for other Ceph clients

Understanding the Object Gateway and Storage Pools

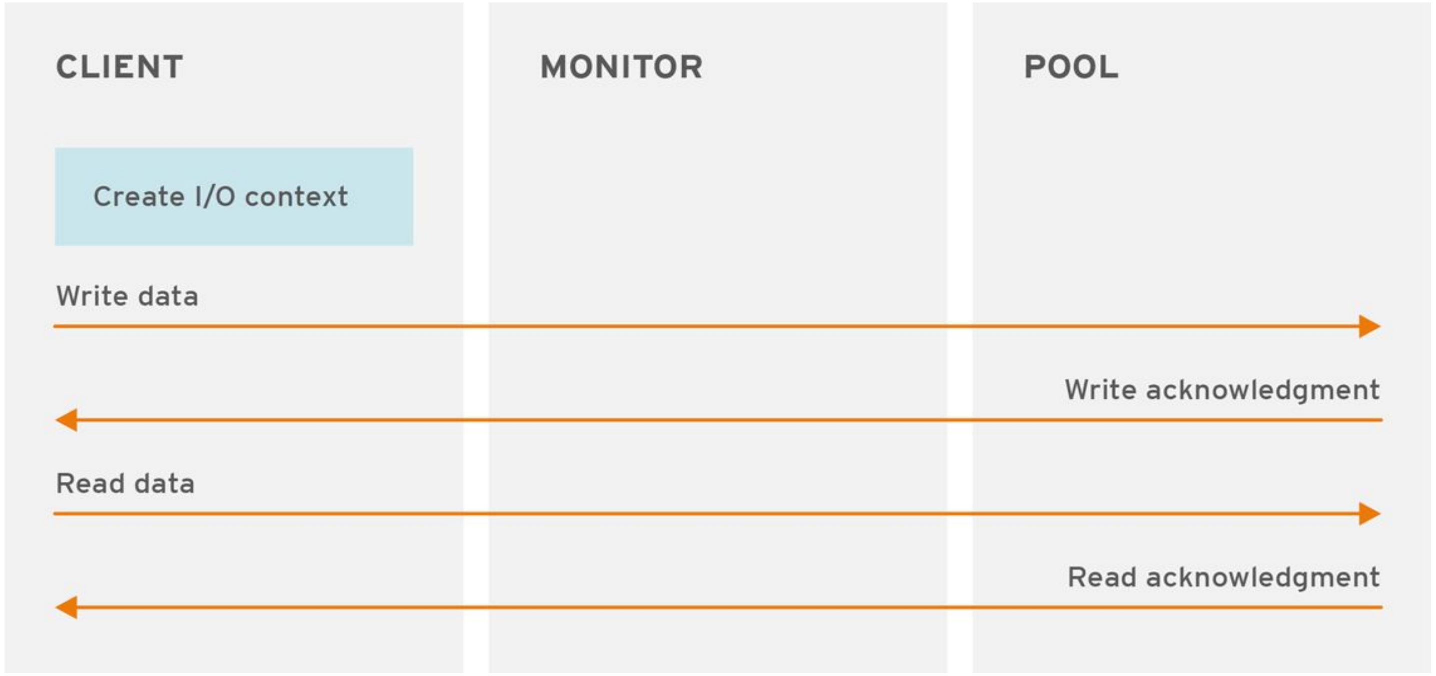
All Ceph clients retrieve the cluster map

Ceph Object Gateway is a client, and so it retrieves a cluster map first



Clients bind to a pool

Ceph Object Gateway binds to pools and reads and writes data



Service Pools

There are 10 Pools just for the Service Layer

Service pools are completely separate from the pools that store client data.

- Root: Realm, zone group and zone configuration.
- Control
- Garbage Collection
- Logs: Multisite replication Logging and Intent Logging
- Intent Log pool
- User Management: S3 UIDs, Swift UIDs, access keys, user emails, quotas/usage.
- Usage

Service pools can use the same CRUSH hierarchy and must use replication for data durability.
May use the same CRUSH hierarchy just for the service pools.

Data Placement Pools

There are 3 Pools Per Placement Policy

Placement policies allow you to have different hardware configurations for different performance characteristics. There are three pools per placement policy:

- *Bucket Index:* Stores an index of objects per bucket. **Use SSD/NVMe and Sharding!**
- *Bucket:* Stores the data objects of a bucket. **Use erasure coding!**
- *Extras:* Stores other data, such as multipart uploads. **Use SSD/NVMe!**
- *Multi-site Replication pools, .log pool:* Stores replication logs. **Use SSD/NVMe!**

Hardware Planning

Hardware Defines Performance

...even though it is software-defined storage

Sometimes the term “software-defined storage” leads to misconceptions.

- “Commodity hardware” doesn’t mean old or low performance hardware
- Monitors are lightweight, but they are mission critical.
- The gateway supports multiple performance profiles--with different hardware
- Ceph (like all distributed systems) relies on networking.
- Saving money on hardware may increase expenses elsewhere

Hardware plays a significant role in achieving your objectives. The Ceph Homepage can be reviewed for minimal HW recommendations for your cluster. Additional research is recommended into reference architectures for your expected workload. Reference architectures are an invaluable source of information!
SuperMicro, Cisco and Micron have some Reference architectures worth reviewing.

Ceph Hardware Recomendations
RHCS HardwareGuide
SuperMicro Ceph References
RHCS Architecture Guide

Hardware Planning

Identifying your general use case(s)

- Throughput-optimized (HDD/SSD)
- Capacity-optimized (HDD)
- IOPS-optimized (All Flash)

Estimating scale and growth requirements

- Data pool uses erasure coding. 1.5x
- Other service pools use replication. 3x
- Factor for multi-site requirements
- Consider compression too! CPU usage overhead should be evaluated!
- How fast is the cluster expected to grow? Capacity planning is important!

Identifying workload(s)

- Write (and delete), read intensive?
- What are the typical object sizes?
- What is a typical client load?

Estimating storage density

- How many OSDs per host?
- NVMe/SSD disks per host? Block.db, wal and Index
- Bandwidth requirements
- Gateway to OSD ratio? 1 RGW:per:50-100 OSDs

Networking

Ceph High Availability depends on Network High Availability

Performance problems often begin with the network, not Ceph.

- High availability
  - Rack/leaf switches can be a single point of failure
  - A switch failure in a single rack cluster can bring down the cluster.
  - Nodes should always have at least two NICs
  - Nodes should use at least 10GB
  - Inter-rack bandwidth should be at least 40GB (or LCAP Mode 4 bonded 10GB)
- Always provision a cluster network to use more bandwidth than the public network
- Consider separate 1G networks for IPMI and management

Networking

Additional considerations

Performance problems often begin with the network, not Ceph.

- Benchmark test your throughput. Bent cat-6 cables can reduce throughput.
- Ceph loves jumbo frames (MTU 9000)
- 10Gbps is the bare minimum for a production cluster.
- Flash (SSD, NVMe, etc) easily saturates a 10Gbps network. Bottleneck!
- Consider serving public and cluster networks with the same rack switch (and make them redundant), so that a switch failure doesn’t create a more significant problem for either the public or cluster network.
- Bandwidth between multiple sites must be sufficient to achieve timely consistency

Storage Drives

Carefully Evaluate Storage Media

Storage drives play an important performance role, especially under heavy workloads

- HDD capacity is increasing faster than sustained transfer rates.
  - Generally faster RPMs provide higher sustained transfer
  - For higher capacity drives, require higher sustained transfer (300Mbps)
- HBA vs on-board controller. UPS required if enabling write back cache!
- Disable drive cache
- Identify the bottlenecks: Drive? Controller? Network?

Storage Drives

Carefully Evaluate Storage Media

SSD/NVMe drives play an important performance role

- For flash drives, always review the specifications and use enterprise-grade drives.
- Use SSDs for monitors to address synchronous write latency. Ensure enough capacity.
- Use SSDs for Bluestore block.db. Consider db to SSD/NVMe ratio. 4% of data.block is recommended for block.db for RGW workloads.
- Block.wal is only required if three tiers of media is available. No need for wal if block.db and wal will reside on same media.
- Use SSDs for bucket indexes
- Use SSDs for extras (multipart uploads)
- Use SSDs for Replication pool (Multi-site)
- Use SSDs for service pools.

Storage Density

Don’t be too dense!

Don’t assume everyone understands your high availability requirements.

- All drives should be hot-swappable.
- Dense storage may require you to pull operating drives to fix broken ones.
- Excessive density can lead to other impacts
  - Dense storage servers require much more bandwidth. OSD:NIC ratio.,
  - Dense storage servers are a bigger point of failure.
  - Backfilling and recovery requires substantial bandwidth in the cluster network.
  - Erasure-coding requires more nodes, not less. K=8; m=4 == 12

Summary

Monitor nodes

- 3 monitor nodes (minimum)
- Use SSDs
- Use uninterrupted power supply (UPS)

OSD Nodes

- ~12 OSDs per node
- Always use flash (SSD) for block.db.
- Always use flash (SSD) for bucket indexes, service pools.
- Identical hardware in a CRUSH hierarchy

OSD Nodes (continued)

- 10 Gbps+ NIC for public network
- 25 Gbps+ NIC for cluster network
- 12 nodes minimum for best erasure code durability (K=8; m=4 requires 12 nodes).
- Replication should use 3 copies for HDD. 2 copies per SSD/NVMe if disk MTBF is reviewed.

RGW Nodes

- At least two per zone.
- 1 Gateway to 50-100 OSDs
- HAProxy/keepalived load balancing

Cluster Tuning

Adjusting the Cluster Map Size

- Jewe and prior large clusters can benefit from adjusting the cluster map size.
  - Edit the Ceph configuration for the following:

```
[global]
osd_map_message_max=10
[osd]
osd_map_cache_size=20
osd_map_max_advance=10
osd_map_share_max_epochs=10
osd_pg_epoch_persisted_max_stale=10
```

- This isn’t necessary in Luminous and above, because the ceph-manager daemon offloads pg operations.

Cluster Tuning

Adjust Default Scrubbing Settings

- Scrubbing during high loads can significantly impact I/O leading some to turn off scrubbing. Consider scrubbing during low-load time windows or low load thresholds. For example:

```
[osd]
#night time scrubbing
osd_scrub_begin_hour = 23
osd_scrub_end_hour = 6
osd_scrub_load_threshold = 0.25 #low load scrubbing
osd_scrub_during_recovery = false #scrub during recovery
```

Cluster Tuning

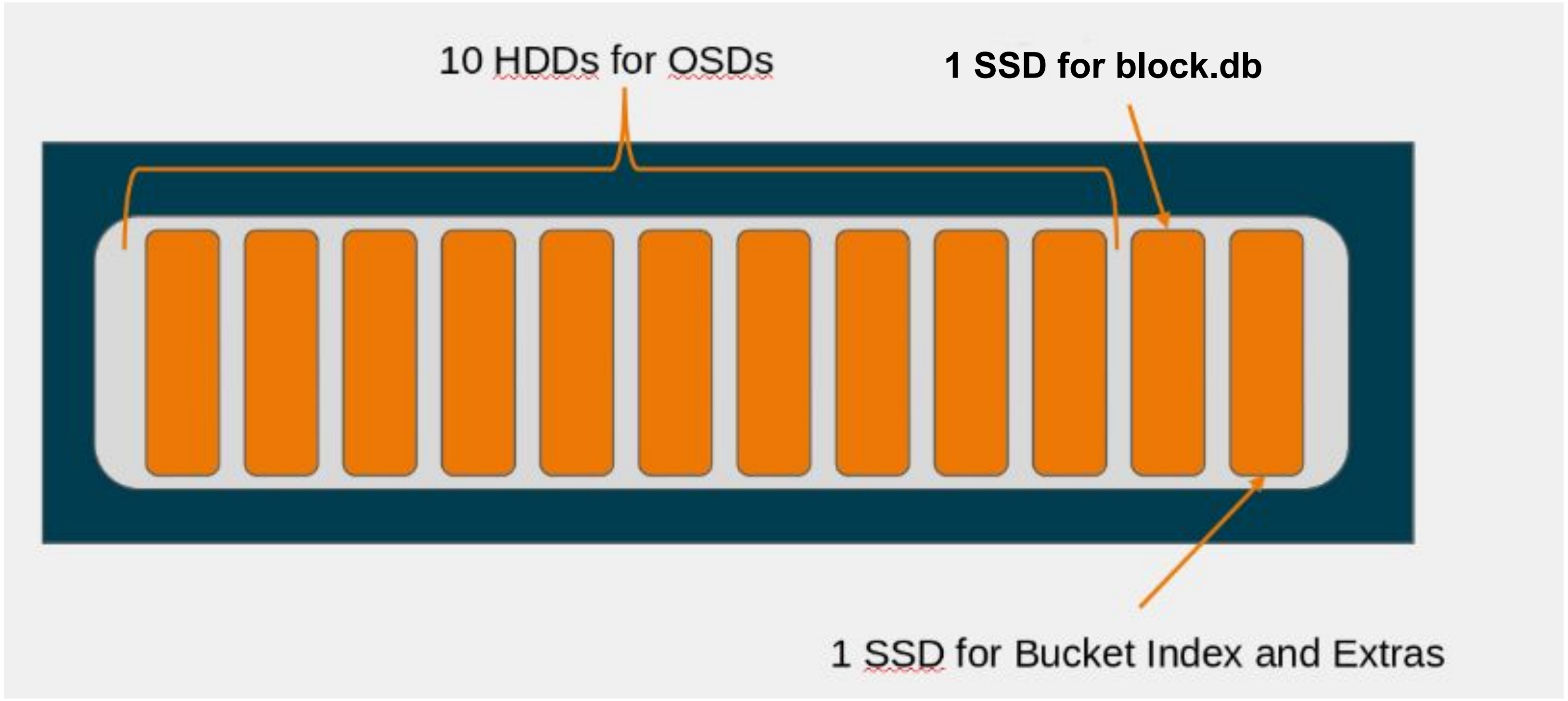
Adjust Default Backfill Settings

- Adjust backfill settings so that backfill doesn’t impact I/O significantly. These can be temporarily increased during low loads if backfill is ongoing.

```
[osd]
osd_max_backfills = 1
osd_recovery_max_active = 1
osd_recovery_op_priority = 1
```

Developing Storage Strategies

Example Server



Create CRUSH Hierarchies and Rules

Underlying hardware and rules get mapped to pools

For each use case, create CRUSH hierarchies and for:

- The data pool. The CRUSH rule will use erasure-coding.
- The service pools. **May** use the same hierarchy as data, but will use replication.
- Index, log and extras pools. **Separate** SSD hierarchy using replication.

Create CRUSH Hierarchies and Rules

CRUSH Hierarchy

```
# ceph osd crush add-bucket t-put root
# ceph osd crush add-bucket rack1 rack
# ceph osd crush add-bucket node-a host
# ceph osd crush add-bucket node-b host
# ceph osd crush add-bucket rack2 rack
# ceph osd crush add-bucket node-c host
# ceph osd crush add-bucket node-d host
# ceph osd crush move rack1 root=t-put
# ceph osd crush move node-a rack=rack1
# ceph osd crush move node-b rack=rack1
# ceph osd crush move rack2 root=t-put
# ceph osd crush move node-c rack=rack2
# ceph osd crush move node-d rack=rack2
```

CRUSH Rules

```
# ceph osd erasure-code-profile set
data-profile \
   k=8 \
   m=4 \
   crush-failure-domain=host
   crush-root=throughput
   crush-device-class=hdd

# ceph osd crush rule create-replicated
service t-put host hdd
# ceph osd crush rule create-replicated
bucket-index t-put host ssd
# ceph osd crush rule create-erasure data
data-profile
```

Create Pools

Pools are the Object Gateway's interface to RADOS

For each data placement policy, create pools for:

- The data bucket. This pool will use erasure coding.
- The services. **May** use the same hierarchy as data, but should use replication.
- Index, log and extras pools. Will use a separate SSD hierarchy with replication.

Create Service Pools

Service pools may use the same CRUSH hierarchy and rule

- Use fewer PGs per pool, because many pools may use the same CRUSH hierarchy.

```
# ceph osd pool create .<zone>.rgw.root <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.control <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.gc <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.log <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.intent-log <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.usage <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.users.keys <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.users.email <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.users.uid <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.users.swift <pg> <pgp> replicated service
```

Create Data Placement Pools

Service pools may use the same CRUSH hierarchy and rule

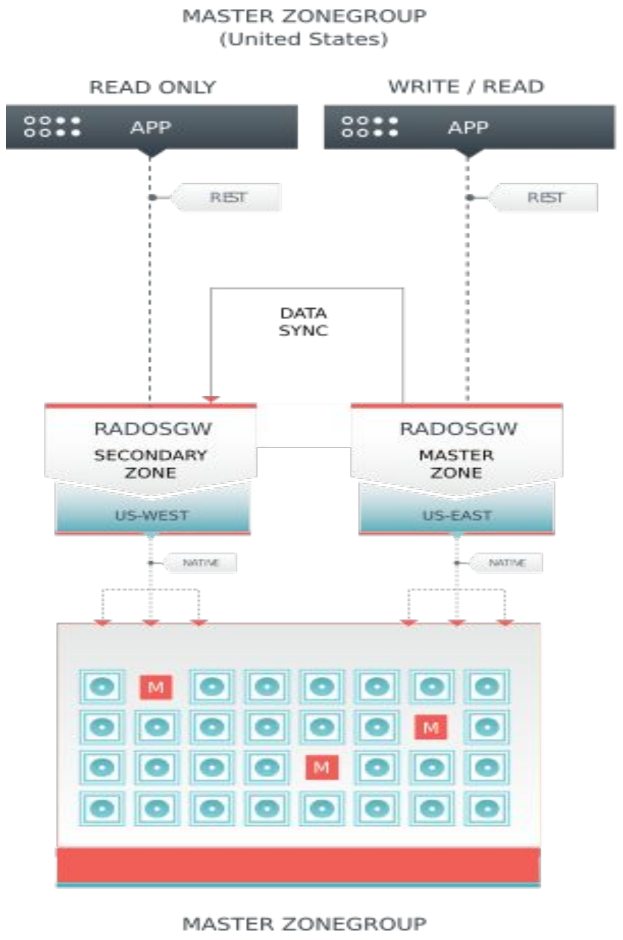
- Use fewer PGs per pool, because many pools may use the same CRUSH hierarchy.

- The data pool uses the erasure code profile:
  # ceph osd pool create .<zone>.buckets <pg> <pgp> erasure data
- The bucket index profile uses SSDs
  # ceph osd pool create .<zone>.buckets.index <pg> <pgp> replicated bucket-index
- The extras uses SSDs
  # ceph osd pool create .<zone>.buckets.extra <pg> <pgp> replicated bucket-index

Understanding Ceph Object Storage Multisite Cloud configuration

Understanding Rados Gateway Multisite basic building blocks

- **Zone:** In a zone, one or more Ceph Object Gateways are logically grouped.
- **Zone group:** A zone group is a container of multiple zones. In a multi-site configuration there should be a master zone group. All the changes to configurations are handled by the master zone group.
- **Realm:** A realm can have multiple zone groups. It allows separation of the zone groups themselves between clusters. There can be multiple realms for having different configurations in the same cluster.
- **Period:** Every realm has a corresponding current period. Each period is a container of an epoch and an unique id. A period holds the current state of configuration of the zone groups and object storage strategies. Each period's commit operation, as well as any configuration change for a non-master zone, will increment the period’s epoc.


MASTER ZONEGROUP
(United States)
READ ONLY
WRITE / READ
APP
APP
REST
REST
DATA SYNC
RADOSGW
SECONDARY ZONE
US-WEST
RADOSGW
MASTER ZONE
US-EAST
NATIVE
NATIVE
M
M
M
MASTER ZONEGROUP

How to configure Ceph Object Storage Multisite between two production clusters

Configure RGW Multisite

Document to follow - Ceph Object Storage Multisite

- **Create the realm**

  radosgw-admin realm create --rgw-realm={realm-name} [--default]

  For example:

  radosgw-admin realm create --rgw-realm=productioncloud --default

Configure RGW Multisite

- **Create the master zonegroup**

  radosgw-admin zonegroup create --rgw-zonegroup={name} --endpoints={url} [--rgw-realm={realm-name}|--realm-id={realm-id}] --master --default

  For example:

  radosgw-admin zonegroup create --rgw-zonegroup=support --endpoints=http://rgw1.node.com:80 --rgw-realm=productioncloud --master --default

Configure RGW Multisite

- **Create the master zone**

  ```
  radosgw-admin zone create --rgw-zonegroup={zone-group-name} \
                              --rgw-zone={zone-name} \
                              --master --default \
                              --endpoints={http://fqdn}[,{http://fqdn}]
  ```
  For example:

  ```
  radosgw-admin zone create --rgw-zonegroup=support --rgw-zone=us-east \
                              --master --default \
                              --endpoints=http://rgw1.us.east.node.com:80
  ```

- After this you can delete default zone, zonegroup and their respective pools if you are not using your default zone and zonegroup.

Configure RGW Multisite

- **Create the system user - replication user**

  radosgw-admin user create --uid="{user-name}" --display-name="{Display Name}" --system

  For example:

  radosgw-admin user create --uid="synchronization-user" --display-name="Synchronization User" --system

- Make a note of the access_key and secret_key, as the secondary zones will require them to authenticate with the master zone.

Configure RGW Multisite

- **Modify the system user - to add secret and access keys**

  radosgw-admin zone modify --rgw-zone=us-east --access-key={access-key} --secret={secret}

- **Update the period**

  radosgw-admin period update --commit

- As we discussed before updating the period changes the epoch, and ensures that other zones will receive the updated configuration.

Configure RGW Multisite

- **Update the ceph configuration file**

```
[client.rgw.{instance-name}]
...
rgw_zone={zone-name}
```

  For example:

```
[client.rgw.rgw1]
host = rgw1
rgw frontends = "civetweb port=80"
rgw_zone=us-east
```

- **Restart the RGW daemon**

```
systemctl restart ceph-radosgw@rgw.`hostname -s`
```

Configure RGW Multisite

- **Pull the realm on the secondary site**

```
# radosgw-admin realm pull --url=http://rgw1.us.east.node.com:80
--access-key={access-key} --secret={secret}
```

Important - Here url is nothing but master zone gateway url and access and secret key of the system(replication) user which was created in master zone and we made a note of it.

Configure RGW Multisite

- **Pull the current period on the secondary site**

  radosgw-admin period pull --url={url-to-master-zone-gateway} --access-key={access-key} --secret={secret}

  Pulling the period retrieves the latest version of the zone group and zone configurations for the realm from master site.

Configure RGW Multisite

- **Create the secondary zone**

```
radosgw-admin zone create --rgw-zonegroup={zone-group-name} \
                          --rgw-zone={zone-name} \
                          --master --default \
                          --endpoints={http://fqdn}[,{http://fqdn}]
```

  For example:

```
radosgw-admin zone create --rgw-zonegroup=support --rgw-zone=us-west \
                          --master --default \
                          --endpoints=http://rgw1.us.west.node.com:80
```

- After this you can delete default zone, zonegroup and their respective pools if you are not using your default zone and zonegroup.

Configure RGW Multisite

- **Update the period on the secondary site**

  radosgw-admin period update --commit

- As we discussed before updating the period changes the epoch, and ensures that other zones will receive the updated configuration. Now master zone us-east will have information about us-west secondary zone.

Configure RGW Multisite

- **Update the ceph configuration file on the secondary site**

```
[client.rgw.{instance-name}]
...
rgw_zone={zone-name}
```

  For example:

```
[client.rgw.rgw1]
host = rgw1
rgw frontends = "civetweb port=80"
rgw_zone=us-west
```

- **Restart the RGW daemon**

```
systemctl restart ceph-radosgw@rgw.`hostname -s`
```

Configure RGW Multisite

- **Ceph Ansible and RGW Multisite**
  - Latest ceph ansible can be used to configure RGW multi-site these PR's have bring the support for it.
    - https://github.com/ceph/ceph-ansible/pull/1944
    - https://github.com/ceph/ceph-ansible/pull/3560

Advance RGW Multisite Configuration

- **One master zone and multiple secondary zones**
  - You can have this type of configuration where you have one master site and multiple DR sites.
  - You can achieve this configuration with the help of zone flags sync-from-all and sync-from.
  - In this scenario you need to disable sync-from-all in all zones and use sync-from option to specify zone name from where you to sync the data.
    - *radosgw-admin zone modify --rgw-zone=<zone name> --sync-from-all=false --sync-from=<source zone name>*
    - *radosgw-admin period update --commit*

How to manage Multisite clouds using radosgw-admin command

RGW Multisite sync status

- **Check synchronization status on secondary site**
  Once the secondary zone is up and running, If you will check the synchronization status on the secondary site you will see both *metadata* and *data* syncing from master.

```
radosgw-admin sync status
          realm f3d16b29-54ba-437e-9732-c9ea647e8e27 (productioncloud)
      zonegroup 15124921-b94b-4f0a-9807-1c54b16051cd (support)
           zone fb62f3a4-d1c5-4074-8aee-013853302feb (us-west)
  metadata sync syncing
                full sync: 0/64 shards
                incremental sync: 64/64 shards
                metadata is caught up with master
      data sync source: a9595924-6d2c-41ec-a1aa-872363988ebc (us-east)
                        syncing
                        full sync: 0/128 shards
                        incremental sync: 128/128 shards
                        data is caught up with source
```

RGW Multisite sync status

- **Check synchronization status on master site**
  Once the secondary zone is up and running, If you will check the synchronization status on the master site you will see only *data* syncing from secondary as all the metadata operations are allowed in master site and by default configuration is not read-only on secondary.

```
  radosgw-admin sync status
        realm f3d16b29-54ba-437e-9732-c9ea647e8e27 (productioncloud)
    zonegroup 15124921-b94b-4f0a-9807-1c54b16051cd (support)
         zone a9595924-6d2c-41ec-a1aa-872363988ebc (us-east)
metadata sync no sync (zone is master)
    data sync source: fb62f3a4-d1c5-4074-8aee-013853302feb (us-west)
                      syncing
                      full sync: 0/128 shards
                      incremental sync: 128/128 shards
                      data is caught up with source
```

RGW data sync perf counter

- **Recently merged in master and getting backported to Nautilus**
  - http://tracker.ceph.com/issues/38549
  - https://github.com/ceph/ceph/pull/26722
  - Nautilus - http://tracker.ceph.com/issues/38918
  - Nautilus - https://github.com/ceph/ceph/pull/27921

- This feature would be also available for Mimic and Luminous releases.
- This feature adds the following performance counters to multi-site configuration of the Ceph Object Gateway to measure data sync:
- **poll_latency:** measures the latency of requests for remote replication logs.
- **fetch_bytes:** measures the number of objects and bytes fetched by data sync.

RGW data sync perf counter

- **Example in Master site us-east zone RGW admin socket**

```
# ceph --admin-daemon /var/run/ceph/ceph-client.rgw.`hostname -s`.asok perf dump data-sync-from-us-west
{
    "data-sync-from-us-west": {
        "fetch_bytes": {
            "avgcount": 1,
            "sum": 8273
        },
        "fetch_not_modified": 0,
        "fetch_errors": 0,
        "poll_latency": {
            "avgcount": 3,
            "sum": 0.005299022,
            "avgtime": 0.001766340
        },
        "poll_errors": 0
    }
}
```

RGW data sync perf counter

- **Example in Secondary site us-west zone RGW admin socket**

```
# ceph --admin-daemon /var/run/ceph/ceph-client.rgw.dell-per630-2.asok perf dump data-sync-from-us-east
{
    "data-sync-from-us-east": {
        "fetch_bytes": {
            "avgcount": 1,
            "sum": 9854
        },
            "fetch_not_modified": 1,
        "fetch_errors": 0,
        "poll_latency": {
            "avgcount": 4,
            "sum": 0.125598514,
            "avgtime": 0.031399628
        },
            "poll_errors": 0
    }
}
```

RGW Multisite other important commands

- **Bucket level sync enable/disable**

```
radosgw-admin bucket sync disable
radosgw-admin  bucket sync enable
```

- **Bucket level sync status**

```
radosgw-admin bucket sync status --bucket=<bucket name>
```

- **Period get**

```
radosgw-admin period get
```

RGW Multisite Issues and Solutions

Bucket resharding in RGW Multisite

- RGW dynamic bucket resharding is not supported in a Multisite environment hence it is highly recommended to plan number of objects per bucket keeping future growth in mind and configure this option rgw_override_bucket_index_max_shards value properly so that you won't have to reshard the buckets in future.
- If buckets goes above millions objects without a proper shard count then you may start seeing issue(flapping OSDs) which I and Mike had discussed this in last Cephalocon in China https://www.slideshare.net/VikhyatUmrao/cephalocon-apac-china you can check this presentation and it has resolution.

Checking replication for a particular replication shard

- This command lists which log shards, if any, which are behind their source zone. If the results of the sync status you have run above reports log shards are behind, run the following command substituting the shard-id for X.

radosgw-admin data sync status --shard-id=X
Replace… X with the ID number of the shard.

Checking replication for a particular replication shard

Example

```
[root@rgw ~]# radosgw-admin data sync status --shard-id=27
{
  "shard_id": 27,
  "marker": {
        "status": "incremental-sync",
        "marker": "1_1534494893.816775_131867195.1",
        "next_step_marker": "",
        "total_entries": 1,
        "pos": 0,
        "timestamp": "0.000000"
  },
   "pending_buckets": [],
   "recovering_buckets": [
        "test-bucket:4ed07bb2-a80b-4c69-aa15-fdc17ae6f5f2.314303.1:26"
   ]
}
```

Checking replication for a particular replication shard

- The output lists which buckets are next to sync and which buckets, if any, are going to be retried due to previous errors.
- Inspect the status of individual buckets with the following command, substituting the bucket id for X.

  radosgw-admin bucket sync status --bucket=X.
  Replace…
  X with the name of the bucket.
- The result shows which bucket index log shards are behind their source zone. Read errors written to the sync error log, which can be read with the following command:

  radosgw-admin sync error list

The syncing process will try again until it is successful. Errors can still occur that can require intervention.

OMAP growth because of RGW Multisite data_log and bilog trimming

- We had some issues with RGW multisite data_log and bilog trimming and this was causing lot of omap growth in replication log pool pgs backed OSD's and causing them to flap if not tuned properly.
- This has been fixed with the help of following trackers:
  - https://tracker.ceph.com/issues/38373
  - https://tracker.ceph.com/issues/38075
  - https://tracker.ceph.com/issues/39487
  - https://tracker.ceph.com/issues/39283

Thank you!

Questions?

Michael Hackett
Vikhyat Umrao
Principal Software Maintenance Engineers - Ceph
mhackett@redhat.com
vumrao@redhat.com

redhat.com | ceph.com