

Ceph S3 - Dynamic Placement and Optimized Retention

Storage Classes, LUA and LifeCycle Policies

Frédéric Nass | Senior Ceph Engineer @CLYSO

Content

01. **Storage Classes**
Placement targets & pools

02. **Dynamic Placement**
LUA scripting

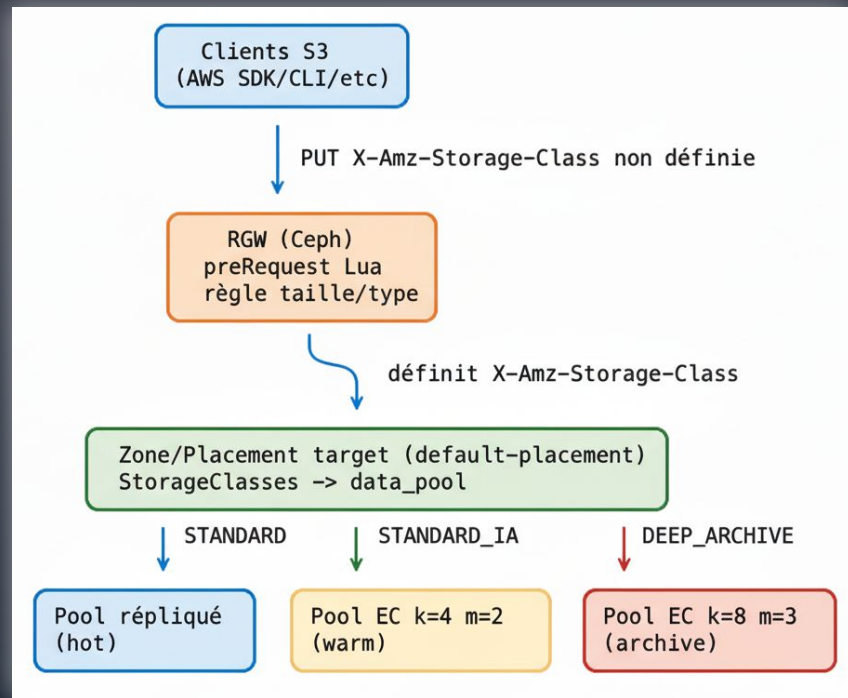
03. **Optimized Retention**
LifeCycle Policies

04. **Demo**
Writing S3 Objects

Storage Classes

- One or more in S3 storage
- Associated with a data placement model: replication, erasure coding, compression
- Optimize S3 object placement based on different criteria: performance, access frequency, durability, costs
- The S3 client can indicate the storage class it wishes to use for its object

But clients don't always chose the right Storage Class. Sometimes, they don't specify one and sometime they force it when they shouldn't.

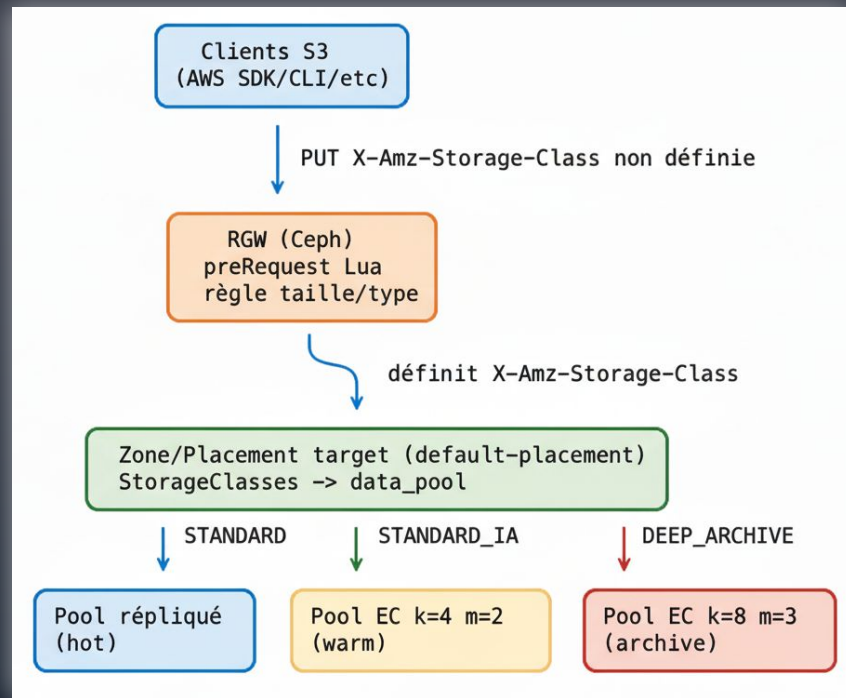


Dynamic Placement

...what if we chose for them?

- When the client does not specify it or incongruously forces it.
- Automatically assign a storage class to an object based on:
 - the object type (.mp4, .data, .pdf)
 - the object size (< 64kB, > 1MB)
 - the tenant (labo1, labo2)
 - the bucket (large or small objects)
 - the upload method (MPU or not)

L'efficacité de stockage et la performance d'accès seraient assurées dès l'écriture des données



Dynamic Placement - LUA Scripting

- S3 gateways can execute LUA scripts on the fly based on the context (prerequest, postrequest, background, getdata, putdata)
- These LUA scripts can read and modify an object's metadata before writing it

We can therefore dynamically define or modify an object's storage class upon writing, according to certain criteria

docs.ceph.com/en/latest/radosgw/lua-scripting

```
-- Rule line: STORAGECLASS;PATTERN;OP;BYTES;BUCKET;TENANT;OVERRIDE
-- OP in {<, <=, =, >=, >, *}
local function parse_rule(line, lineno)
    local parts = {}
    for field in string.gmatch(line, "([^\;]+)") do parts[#parts+1] =
trim(field) end
    if #parts < 1 then return nil end
    local bytes_num, bytes_pretty = parse_size(parts[4] or "0")
    local r = {
        storage_class = parts[1] or "",
        pattern       = (parts[2] ~= "" and parts[2]) or "*",
        op            = (parts[3] ~= "" and parts[3]) or "*",
        bytes         = bytes_num,
        bytes_str     = bytes_pretty,
        bucket        = (parts[5] ~= "" and parts[5]) or "*",
        tenant        = (parts[6] ~= "" and parts[6]) or "*",
        override      = to_bool(parts[7], false),
        lineno        = lineno
    }
    if r.storage_class == "" then return nil end
    return r
end

local function size_matches(op, threshold, content_len)
    if op == "*" then return true end
    if not content_len then return false end
    if op == "<" then return content_len < threshold end
    if op == "<=" then return content_len <= threshold end
    if op == ">" then return content_len > threshold end
    if op == ">=" then return content_len >= threshold end
    if op == "=" then return content_len == threshold end
    return false
end
```

Optimized Retention - Lifecycle Policies

- Lifecycle Policies ensure the transition of S3 objects between storage classes after a certain time
- The oldest data can be rewritten to occupy less space in the cluster

The rule show here :

- cleans up MPU parts after 10 days
- moves objects to DEEP_ARCHIVE mapped to the EC 8+3 pool after 30 days
- deletes objects after 365 days

```
{
  "Rules": [
    {
      "Filter": {
        "Prefix": ""
      },
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "DEEP_ARCHIVE"
        }
      ],
      "AbortIncompleteMultipartUpload": {
        "DaysAfterInitiation": 10
      },
      "Expiration": {
        "Days": 365
      },
      "ID": "double transition and expiration"
    }
  ]
}
```

docs.ceph.com/en/latest/radosgw/lua-scripting

DEMO

Dynamic Placement and Optimized Retention

RGW (S3) Service Configuration

Creating Ceph pools

```
ceph osd erasure-code-profile set ec42 k=4 m=2
ceph osd erasure-code-profile set ec83 k=8 m=3

ceph osd pool create s3.hot.data 256 256 replicated
ceph osd pool create s3.warm.data 128 128 erasure ec42
ceph osd pool create s3.archive.data 64 64 erasure ec83

ceph osd pool application enable s3.hot.data rgw
ceph osd pool application enable s3.warm.data rgw
ceph osd pool application enable s3.archive.data rgw
```

Adding Storage Classes and Default Storage Class

```
radosgw-admin zonegroup placement add --rgw-zonegroup france
--placement-id default-placement --storage-class STANDARD_IA

radosgw-admin zonegroup placement add --rgw-zonegroup france
--placement-id default-placement --storage-class DEEP_ARCHIVE

radosgw-admin zonegroup placement default --rgw-zonegroup france
--placement-id default-placement --storage-class STANDARD_IA

radosgw-admin zonegroup get | grep default_placement
  "default_placement": "default-placement/STANDARD_IA"

radosgw-admin period update --commit
```

Verification

```
radosgw-admin zone placement list
[
  {
    "key": "default-placement",
    "val": {
      "index_pool": "s3.buckets.index",
      "storage_classes": {
        "DEEP_ARCHIVE": {
          "data_pool": "s3.archive.data",
          "compression_type": "zstd"
        },
        "STANDARD": {
          "data_pool": "s3.hot.data"
        },
        "STANDARD_IA": {
          "data_pool": "s3.warm.data",
          "compression_type": "lz4"
        }
      },
      "data_extra_pool": "s3.buckets.non-ec",
      "index_type": 0,
      "inline_data": true
    }
  }
]
```


RGW (S3) Service Configuration

Configuring the data, index and non-ec pools in the zone

```
# STANDARD est déjà présente; s'assurer qu'elle pointe vers le pool répliqué "hot"
radosgw-admin zone placement add --rgw-zone nancy \
  --placement-id default-placement \
  --storage-class STANDARD \
  --data-pool s3.hot.data \
  --index-pool s3.buckets.index \
  --data-extra-pool s3.buckets.non-ec

# STANDARD_IA -> EC 4+2
radosgw-admin zone placement add --rgw-zone nancy \
  --placement-id default-placement \
  --storage-class STANDARD_IA \
  --data-pool s3.warm.data \
  --compression lz4

# DEEP_ARCHIVE -> EC 8+3
radosgw-admin zone placement add --rgw-zone nancy \
  --placement-id default-placement \
  --storage-class DEEP_ARCHIVE \
  --data-pool s3.archive.data \
  --compression zstd
```

Applying the configuration

```
radosgw-admin period update --commit
```

RGW (S3) Service Configuration

Modifying RGW service configuration

```
$ ceph orch ls --export --service_type=rgw --format yaml > rgw.yaml

$ vim rgw.yaml
service_type: rgw
service_id: monde-france-nancy
service_name: rgw.monde-france-nancy
placement:
  count: 1
  label: rgws_nancy
spec:
  rgw_frontend_port: 8080
  rgw_realm: monde
  rgw_zone: nancy
  rgw_zonegroup: france
custom_configs:
  - mount_path: /etc/ceph/rgw_storageclass_rules.conf
    content: |
      # Paramètres globaux MPU
      mpu_default_class=DEEP_ARCHIVE
      mpu_force=true

      # Paramètres globaux défaut non-MPU
      default_class=STANDARD_IA
      default_force=false

      # Règles PUT objet (non-MPU)
      # STORAGECLASS;PATTERN;OP;BYTES;BUCKET;TENANT;OVERRIDE
      #STANDARD_IA;%.pdf;*;0;*;true
      #INTELLIGENT_TIERING;*;<32768;bucket-logs;*;true
      #ONEZONE_IA;%.eml;*;0;*;tenant-a;false
      #GLACIER;%.iso;<;1073741824;media-bucket;*;true

      STANDARD;*;<=2MiB;*;true
      DEEP_ARCHIVE;%.data;*;0B;*;true
```

Applying configuration and redeploy RGWs

```
$ ceph orch apply -i rgw.yaml
Scheduled rgw.monde-france-metz update...
Scheduled rgw.monde-france-nancy update...

$ ceph orch redeploy rgw.monde-france-nancy
Scheduled to redeploy rgw.monde-france-nancy.r03h01.vyyyjq on host 'r03h01'
```

Verification

```
$ container_id=$(ssh -n r03h01 podman ps | grep rgw | awk '{print $1}')

$ ssh -n r03h01 podman exec -it $container_id cat
/etc/ceph/rgw_storageclass_rules.conf
# MPU PUTs settings
mpu_default_class=DEEP_ARCHIVE
mpu_force=true

# non-MPU PUTs settings
default_class=STANDARD_IA
default_force=false

# non-MPU PUTs rules
# STORAGECLASS;PATTERN;OP;BYTES;BUCKET;TENANT;OVERRIDE
#STANDARD_IA;%.pdf;*;0;*;true
#INTELLIGENT_TIERING;*;<32768;bucket-logs;*;true
#ONEZONE_IA;%.eml;*;0;*;tenant-a;false
#GLACIER;%.iso;<;1073741824;media-bucket;*;true

STANDARD;*;<=2MiB;*;true
DEEP_ARCHIVE;%.data;*;0B;*;true
```

RGW (S3) Service Configuration

Downloading LUA script

github.com/frednass/s3-dynamic-placement-and-archiving

```
$ curl -k -s https://raw.githubusercontent.com/frednass/s3-dynamic-placement-and-archiving/refs/heads/main/rgw_storageclass_rules.lua -o rgw_storageclass_rules.lua  
$ radosgw-admin script put --infile=./rgw_storageclass_rules.lua --context=preRequest
```

Adding LUA script to Rados Gateway (RGW)

```
$ curl -k -s https://raw.githubusercontent.com/frednass/s3-dynamic-placement-and-archiving/refs/heads/main/rgw_storageclass_rules.lua -o rgw_storageclass_rules.lua  
$ radosgw-admin script put --infile=./rgw_storageclass_rules.lua --context=preRequest
```

Applying S3 Lifecycle Policy

Creating the bucket

```
$ rclone mkdir s3:/newbucket2
```

Adding Lifecycle Policy to the test bucket

```
$ aws s3api put-bucket-lifecycle-configuration \
--endpoint http://10.38.1.59:8080 \
--bucket newbucket2 \
--lifecycle-configuration file://lifecycle.json
```

Creating the Lifecycle Policy

```
$ vim lifecycle.json
{
  "Rules": [
    {
      "Filter": {
        "Prefix": ""
      },
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "DEEP_ARCHIVE"
        }
      ],
      "AbortIncompleteMultipartUpload": {
        "DaysAfterInitiation": 10
      },
      "Expiration": {
        "Days": 365
      },
      "ID": "double transition and expiration"
    }
  ]
}
```

Check that it applies

```
$ aws s3api --endpoint http://10.38.1.59:8080 get-bucket-lifecycle-configuration
--bucket newbucket2
{
  "Rules": [
    {
      "Expiration": {
        "Days": 365
      },
      "ID": "double transition and expiration",
      "Prefix": "",
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "DEEP_ARCHIVE"
        }
      ],
      "AbortIncompleteMultipartUpload": {
        "DaysAfterInitiation": 10
      }
    }
  ]
}
```

Applying S3 Lifecycle Policy

Check Lifecycle status

```
$ radosgw-admin lc list
[
  {
    "bucket":
":newbucket2:f50809af-d67e-426c-bccd-78f8ff6af983.1276537.1",
    "shard": "lc.6",
    "started": "Thu, 01 Jan 1970 00:00:00 GMT",
    "status": "UNINITIAL"
  }
]
```

Manually trigger its execution

```
$ radosgw-admin lc process
$ radosgw-admin lc list
[
  {
    "bucket":
":newbucket2:f50809af-d67e-426c-bccd-78f8ff6af983.1276537.1",
    "shard": "lc.6",
    "started": "Sun, 28 Sep 2025 09:10:43 GMT",
    "status": "COMPLETE"
  }
]
```

Dynamic Placement Verification

Increasing debug level

```
$ ceph config set global debug_rgw 20
```

Monitoring pools activity

```
Every 2.0s: rados df | grep -i -E 's3.*data|objects' ; echo ---- ; ceph df | grep -i -E 'objects|s3.*data'           r01h01: Mon Sep 29 09:02:20 2025
```

POOL_NAME	USED	OBJECTS	CLONES	COPIES	MISSING_ON_PRIMARY	UNFOUND	DEGRADED	RD_OPS	RD	WR_OPS	WR	USED	COMPR	UNDER	COMPR
s3.archive.data	0 B	0	0	0	0	0	0	5435	0 B	17030	12 GiB	0 B	0 B	0 B	0 B
s3.hot.data	0 B	0	0	0	0	0	0	136913	1.5 GiB	288241	4.0 GiB	0 B	0 B	0 B	0 B
s3.warm.data	0 B	0	0	0	0	0	0	36005	8.3 GiB	85110	16 GiB	0 B	0 B	0 B	0 B

```
----
```

POOL	ID	PGS	STORED	OBJECTS	USED	%USED	MAX AVAIL
s3.hot.data	24	32	0 B	0	0 B	0	37 GiB
s3.warm.data	25	32	0 B	0	0 B	0	74 GiB
s3.archive.data	26	32	0 B	0	0 B	0	80 GiB

Creating objects of different sizes from 16KiB to 4.7 MiB

```
for i in {1..300} ; do dd if=/dev/urandom of=file_${i}.dd bs=16k count=${i} ; done
```

Push objects to S3 storage

```
rclone sync . s3:/newbucket2 --s3-upload-cutoff 200M --s3-upload-concurrency 8
```

Check S3 Storage Classes associated with objects

```
radosgw-admin bucket list --bucket newbucket2
```

Dynamic Placement Verification

Rados Gateway (RGW) Logs

- Object size > 2 MiB (4.7 MiB) is assigned the default SC STANDARD_IA and the EC 4+2 pool

```
2025-09-29T06:45:21.783+0000 7fbc51c28640 20 Lua INFO: Rule #1 (line 16) NO MATCH: obj='file_288.dd' size=4718592 bucket='newbucket2'
tenant='' reason=size op mismatch threshold=2mib
2025-09-29T06:45:21.783+0000 7fbc51c28640 20 Lua INFO: Rule #2 (line 17) NO MATCH: obj='file_288.dd' size=4718592 bucket='newbucket2'
tenant='' reason=name pattern mismatch threshold=0b
2025-09-29T06:45:21.783+0000 7fbc51c28640 20 Lua INFO: No rule applied: apply default StorageClass='STANDARD_IA' (default_force=false)
```

- Object size < 2 MiB (475 KiB) is assigned the SC STANDARD and the x3 replication pool

```
2025-09-29T06:45:21.806+0000 7fbb9babc640 20 Lua INFO: Rule #1 (line 16) MATCH -> apply StorageClass='STANDARD' (override=true)
obj='file_29.dd' size=475136 bucket='newbucket2' tenant='' threshold=2mib
2025-09-29T06:45:21.806+0000 7fbb9babc640 20 Lua INFO: Rule #2 (line 17) NO MATCH: obj='file_29.dd' size=475136 bucket='newbucket2'
tenant='' reason=name pattern mismatch threshold=0b
2025-09-29T06:45:21.806+0000 7fbb9babc640 20 Lua INFO: Applied StorageClass='STANDARD' to object 'file_29.dd'
```

- Object sent as multipart upload (MPU) placed directly in the archives

```
2025-09-29T07:46:38.837+0000 7fbbac2dd640 20 Lua INFO: MPU initiate: apply default StorageClass='DEEP_ARCHIVE' (force=true)
```

Optimized Retention Verification

Monitoring how the different pools are filling up

```
Every 2.0s: rados df | grep -i -E 's3.*data|objects' ; echo ---- ; ceph df | grep -i -E 'objects|s3.*data' r01h01: Mon Sep 29 09:05:01 2025
```

POOL_NAME	USED	OBJECTS	CLONES	COPIES	MISSING_ON_PRIMARY	UNFOUND	DEGRADED	RD_OPS	RD	WR_OPS	WR	USED	COMPR	UNDER	COMPR
s3.archive.data	0 B	0	0	0	0	0	0	5781	0 B	17722	13 GiB	0 B	0 B	0 B	0 B
s3.hot.data	387 MiB	128	0	84	1257 MiB	0	0	137169	1.7 GiB	288625	4.2 GiB	0 B	0 B	0 B	0 B
s3.warm.data	870 MiB	346	0	76	0	0	0	40217	8.9 GiB	98364	17 GiB	0 B	0 B	0 B	0 B

POOL	ID	PGS	STORED	OBJECTS	USED	%USED	MAX AVAIL
s3.hot.data	24	32	129 MiB	128	387 MiB	0.35	36 GiB
s3.warm.data	25	32	580 MiB	346	870 MiB	0.77	73 GiB
s3.archive.data	26	32	0 B	0	0 B	0	79 GiB

Reducing time between each execution

```
$ ceph config set global rgw_lc_debug_interval 1
```

Executing the Lifecycle Policy

```
$ radosgw-admin lc process
```

Checking at 30s (after radosgw-admin gc process --include-all)

```
Every 2.0s: rados df | grep -i -E 's3.*data|objects' ; echo ---- ; ceph df | grep -i -E 'objects|s3.*data' r01h01: Mon Sep 29 09:06:27 2025
```

POOL_NAME	USED	OBJECTS	CLONES	COPIES	MISSING_ON_PRIMARY	UNFOUND	DEGRADED	RD_OPS	RD	WR_OPS	WR	USED	COMPR	UNDER	COMPR
s3.archive.data	978 MiB	346	0	3806	0	0	0	5905	113 MiB	18192	13 GiB	0 B	0 B	0 B	0 B
s3.hot.data	0 B	0	0	0	978 MiB	0	0	137425	1.8 GiB	288753	4.2 GiB	0 B	0 B	0 B	0 B
s3.warm.data	0 B	300	0	1800	0	0	0	42275	9.4 GiB	104041	17 GiB	0 B	0 B	0 B	0 B

POOL	ID	PGS	STORED	OBJECTS	USED	%USED	MAX AVAIL
s3.hot.data	24	32	0 B	0	0 B	0	36 GiB
s3.warm.data	25	32	0 B	300	0 B	0	73 GiB
s3.archive.data	26	32	711 MiB	346	978 MiB	0.87	80 GiB

Checking at 365s (after radosgw-admin gc process --include-all) → pools are empty

Optimized Retention Verification

Rados Gateway (RGW) Logs

Transition at 15 and Expiration at 30 during the tests

- Object written

```
2025-09-29T08:12:56.866+0000 7fbc34bee640 1 beast: 0x7fbb4f01e6f0: 10.38.1.55 - fred [29/Sep/2025:08:12:56.842 +0000] "PUT /newbucket2/file_98.dd?x-id=PutObject HTTP/1.1" 200 1605632 - "rclone/v1.71.1" - latency=0.023999780s
```

```
2025-09-29T08:12:56.869+0000 7fbbf7373640 1 beast: 0x7fbb4f01e6f0: 10.38.1.55 - fred [29/Sep/2025:08:12:56.867 +0000] "HEAD /newbucket2/file_98.dd HTTP/1.1" 200 0 - "rclone/v1.71.1" - latency=0.000999991s
```

- Object transitioned to Storage Class DEEP_ARCHIVE

```
2025-09-29T08:13:11.052+0000 7fbc5bc5d640 20 lifecycle: operator(): key=file_98.ddwp_thrd: 2, 0
2025-09-29T08:13:11.052+0000 7fbc5bc5d640 20 lifecycle: obj_has_expired(): mtime=2025-09-29T08:12:56.861730+0000 days=30 base_time=2025-09-29T08:13:11.053797+0000
timediff=15.0538 cmp=30 is_expired=0
2025-09-29T08:13:11.052+0000 7fbc5bc5d640 20 lifecycle: check(): key=file_98.dd: is_expired=0 wp_thrd: 2, 0
2025-09-29T08:13:11.052+0000 7fbc5bc5d640 20 lifecycle: obj_has_expired(): mtime=2025-09-29T08:12:56.861730+0000 days=15 base_time=2025-09-29T08:13:11.053827+0000
timediff=15.0538 cmp=15 is_expired=1
2025-09-29T08:13:11.052+0000 7fbc5bc5d640 20 lifecycle: check(): key=file_98.dd: is_expired=1 wp_thrd: 2, 02025-09-29T08:09:24.627+0000 7fbc5ac5b640 2 lifecycle:
TRANSITIONED::newbucket2[f50809af-d67e-426c-bccd-78f8ff6af983.1788835.1]):file_98.dd -> DEEP_ARCHIVE wp_thrd: 2, 2
```

- Object deleted

```
2025-09-29T08:13:26.005+0000 7fbc5d460640 20 lifecycle: operator(): key=file_98.ddwp_thrd: 1, 1
2025-09-29T08:13:26.005+0000 7fbc5d460640 20 lifecycle: obj_has_expired(): mtime=2025-09-29T08:12:56.861730+0000 days=30 base_time=2025-09-29T08:13:26.005972+0000
timediff=30.006 cmp=30 is_expired=1
2025-09-29T08:13:26.005+0000 7fbc5d460640 20 lifecycle: check(): key=file_98.dd: is_expired=1 wp_thrd: 1, 1
2025-09-29T08:13:26.005+0000 7fbc5d460640 20 lifecycle: obj_has_expired(): mtime=2025-09-29T08:12:56.861730+0000 days=15 base_time=2025-09-29T08:13:26.005992+0000
timediff=30.006 cmp=15 is_expired=1
2025-09-29T08:13:26.005+0000 7fbc5d460640 20 lifecycle: check(): key=file_98.dd: is_expired=1 wp_thrd: 1, 1
2025-09-29T08:09:39.524+0000 7fbc5b45c640 2 lifecycle: DELETED::newbucket2[f50809af-d67e-426c-bccd-78f8ff6af983.1788835.1]):file_98.dd wp_thrd: 2, 1
```

Acknowledgments

- Yuval Lifshitz for adding Lua scripting into the Rados Object Gateway
- Steven Umbehocker (OSNEXUS) for [his work](#) on RGW autotiering that inspired this presentation
- Anthony D'Atri and Curt Bruns [talk](#) on RGW Lua scripting

Questions ?

Frédéric Nass
frederic.nass@clyso.com