CEPH Object Storage and Rados GateWay

«OpenStack Administration 101» , 30 Nov. – 3 Dec. 2021

Alessandro Costantini – INFN CNAF

01/12/2021

Overview

- **Object storage**
  - What is and why do we need

- **CEPH**
  - Highlights
  - Architecture
  - Interaction with Openstack

- **CEPH Object Storage (RadosGW)**
  - Support to S3 and Swift API
  - Encryption and integration

What is object storage?

Object storage (also known as object-based storage) is a computer data storage architecture that **manages data as objects**, as opposed to other storage architectures like file systems which manages data as a file hierarchy (Posix), and block storage (RBD) which manages data as blocks within sectors and tracks.

Each object typically includes the data itself, a variable amount of metadata, and a globally unique identifier.

What is object storage?

Object storage can be implemented at multiple levels, including the device level (object-storage device), the system level, and the interface level.

Object storage enables capabilities not addressed by other storage architectures

- interfaces that are directly programmable by the application (**API**s)
- a namespace that can span multiple instances of physical hardware
- data-management functions like data (geo)replication and data distribution at object-level granularity
- authenticated, remote access

What is Object Storage used for?

- **Storage for Unstructured Data** such as music, media files, or text documents. Any type of data that doesn't have a distinct structure to it, has metadata (ex. a song's artist, album title, etc.), and likely won't be manipulaed often is a great fit for Object Storage. Some popular products that use object storage in this category that you might recognize are Netflix and Spotify who use is to store their media files.
- **Backup and Recovery** of critical business applications and workloads. With the rise of digital products, mobile devices and the internet, consumers and enterprises expect applications to always be on and functioning. Due to the highly resilient nature and low cost of Object Storage, many businesses use it to backup their data and workloads to ensure business continuity and to prevent data loss in the event of a disaster.
- **Archived data for long term retention**. Sometimes customers specifically in the financial services industry and healthcare industry have requirements to keep data under retention or records for a certain time period (x number of years). Since this data will persist and not be manipulated frequently, object storage is a perfect cost-effective solution for this use case.
- **Cloud Native Applications** for a persistent data store. As businesses look to modernize their approach to application development in an effort to minimize the time to bring their solutions to market, they need a data store that will scale and not cause costs to sky rocket. Object Storage is a great solution for this as applications can connect directly to the object store and will allow for data to scale simply effectively as the business grows with its number of users and locations.
- **Data Lake for Analytics**. With the acceleration of the number of devices generating data (Smartphones, smart devices, IOT sensors etc.), there will be lots of data circulating around that can be processed for intelligent insights. Current storage solutions such as NAS and others are just not effective enough to support this vast growth of data being produced through these various sources. Object Storage can be a great solution for storing all types of data (structured, semi-structured, and unstructured data) that will give businesses a place to dump data before processing and analyzing in order to enable critical insights.

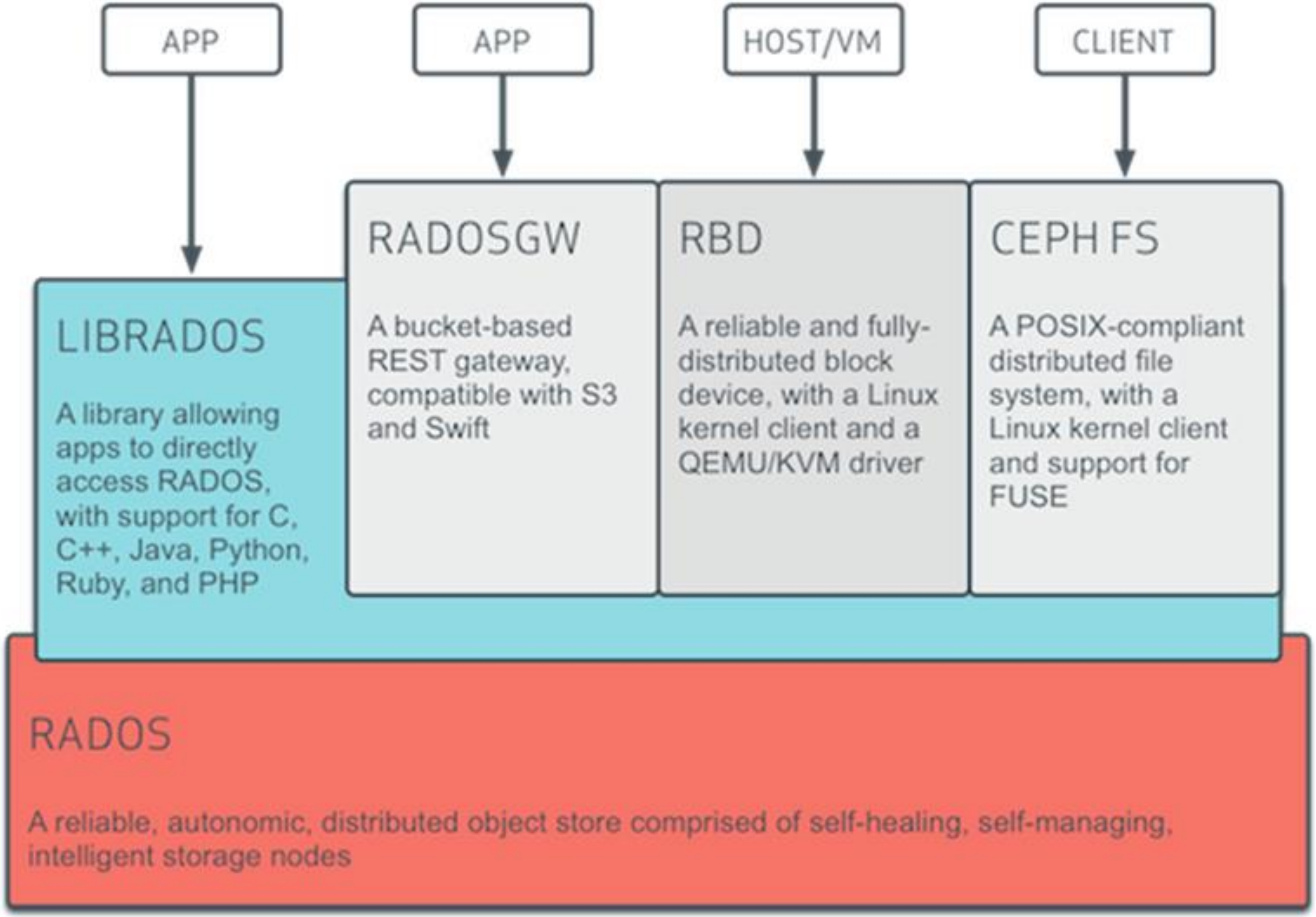
CEPH highlight

- Project started in 2007
- An object based parallel file-system
- Open source project (LGPL licensed)
- Written in C++ and C
- kernel level support
- Posix compliant
- Both data and metadata could be replicated dynamically
- Configuration is config file based
- Adopts flexible replica stragies
  - Could be defined "per Pool"

CEPH Features

- In CEPH everything is an object
- No database for object position on the cluster
- There is a “rule” to place where store data on the cluster:
  - Each node of the cluster can calculate the object position

CEPH Architecture



APP

APP

HOST/VM

CLIENT

LIBRADOS

A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP

RADOSGW

A bucket-based REST gateway, compatible with S3 and Swift

RBD

A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver

CEPH FS

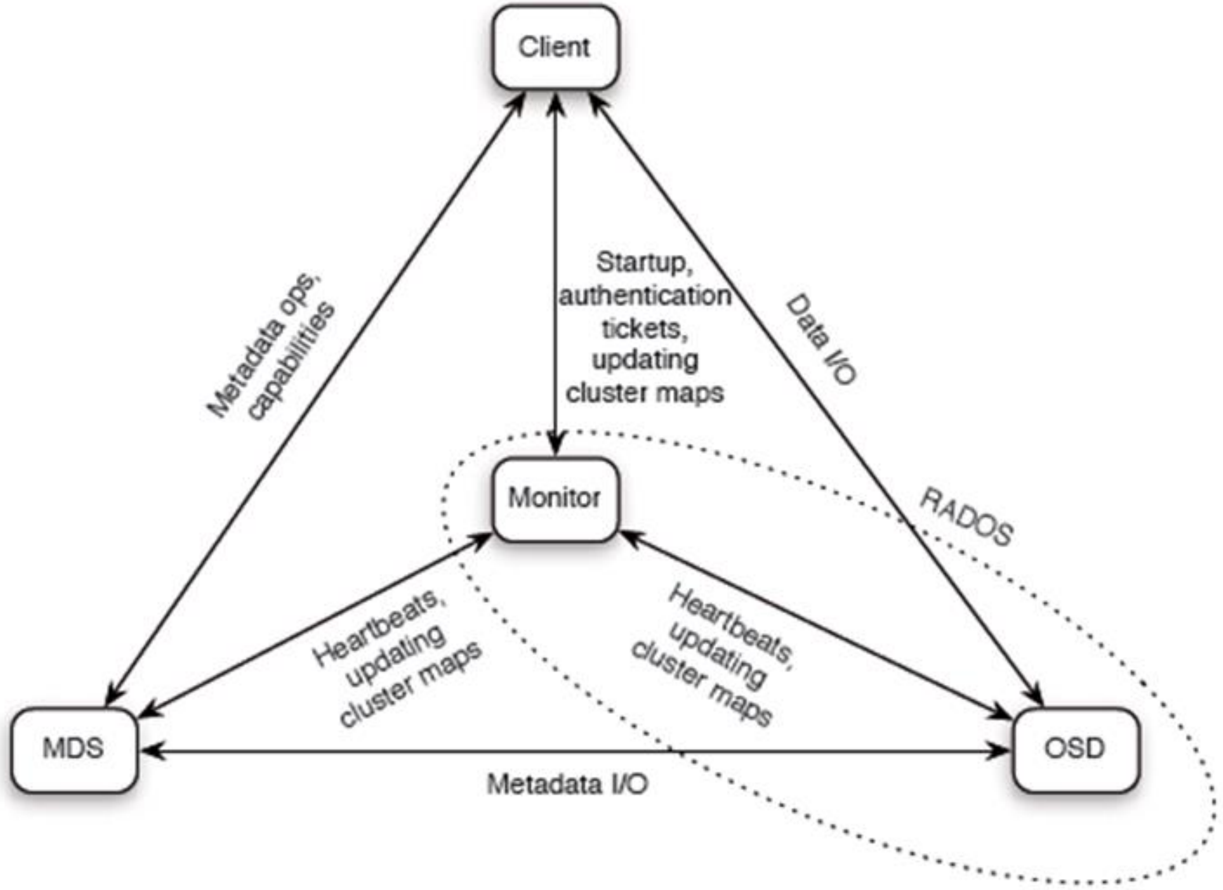
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE

RADOS

A reliable, autonomic, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

CEPH Architecture

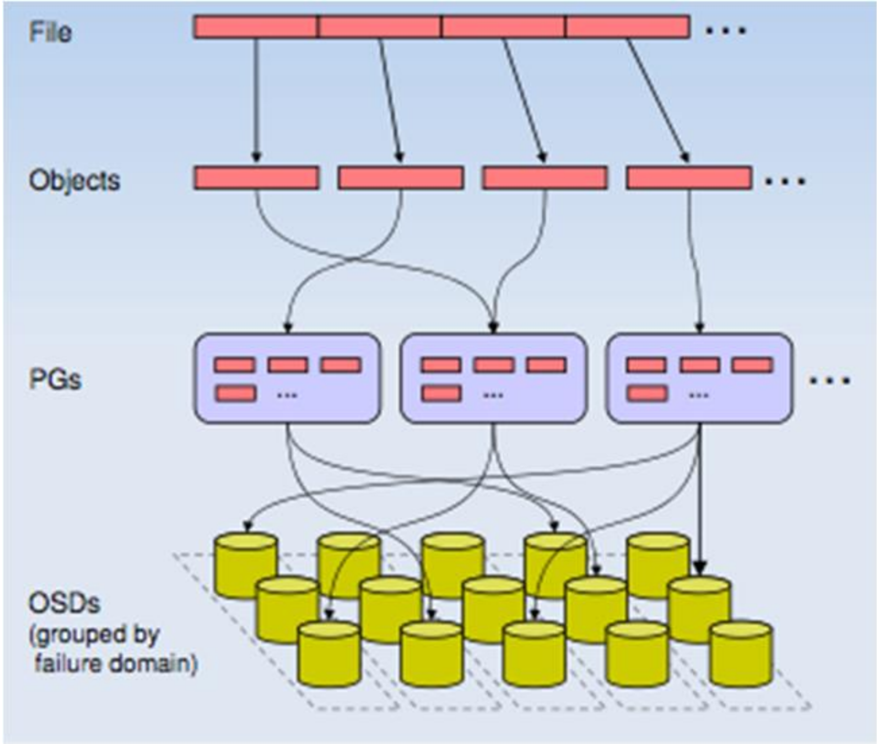
- Object storage daemons (RADOS) and services
- CRUSH tells us where data should go
  - small “osd map” records cluster state at point in time
  - ceph-osd node status (up/down, weight)
  - CRUSH function specifying desired data distribution


Client
Metadata ops, capabilities
Startup, authentication tickets, updating cluster maps
Data I/O
Monitor
RADOS
Heartbeats, updating cluster maps
Heartbeats, updating cluster maps
MDS
Metadata I/O
OSD

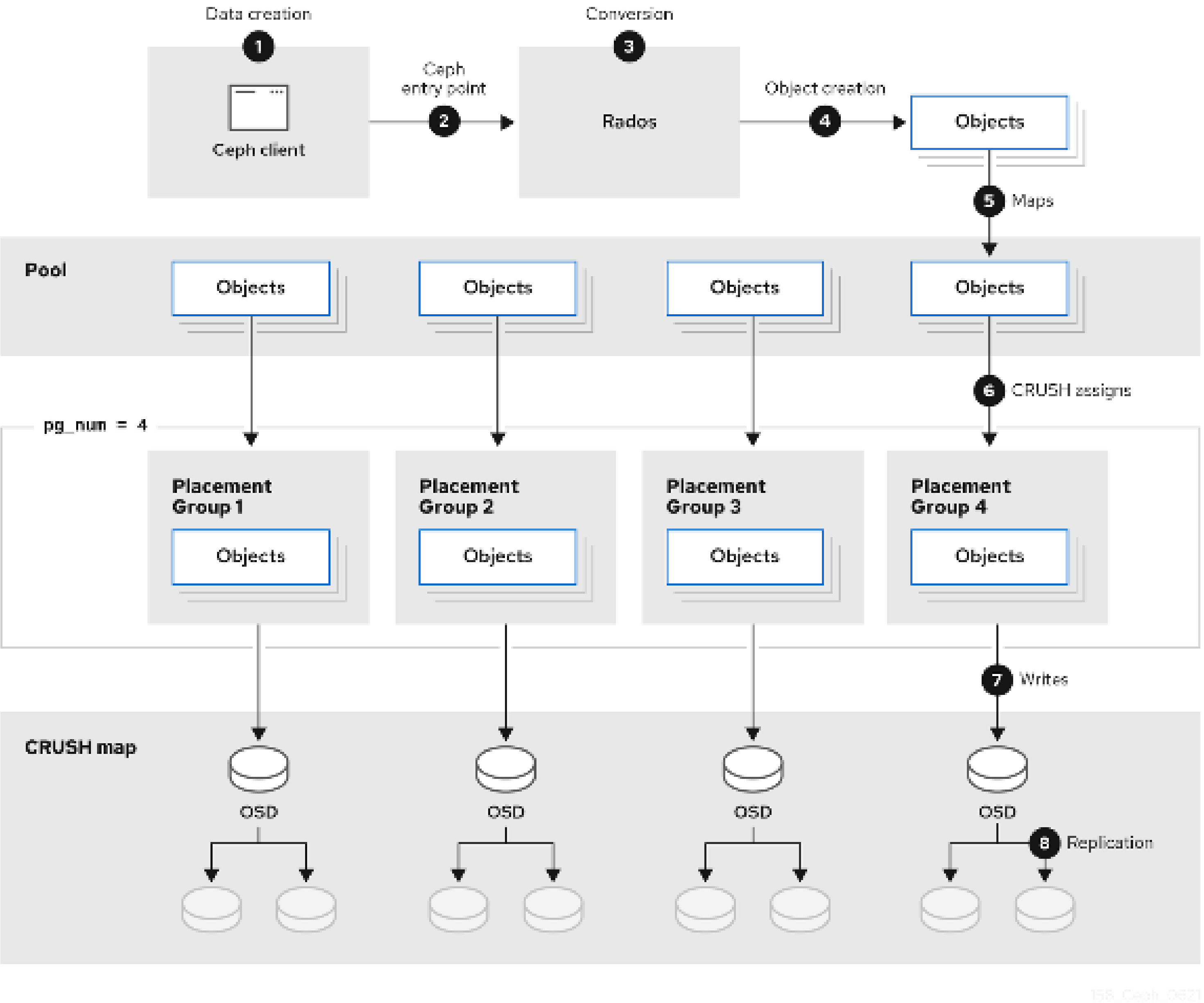
CEPH Architecture

- **Pools**
  - Ceph stores data within pools, which are logical groups for storing objects. Pools manage the number of placement groups, the number of replicas, and the ruleset for the pool

- **Placement Groups**
  - Placement groups (PGs) are shards or fragments of a logical object pool that place objects as a group into OSDs

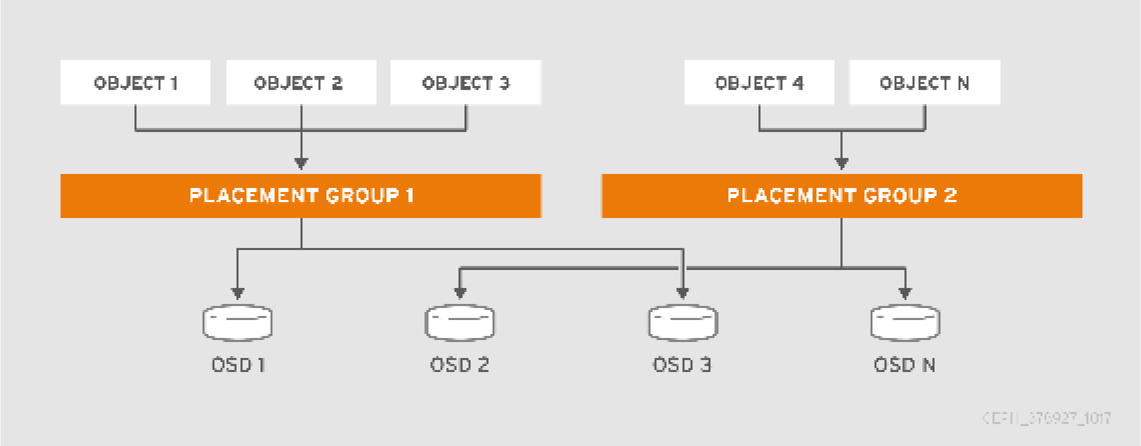


CEPH logical data flow



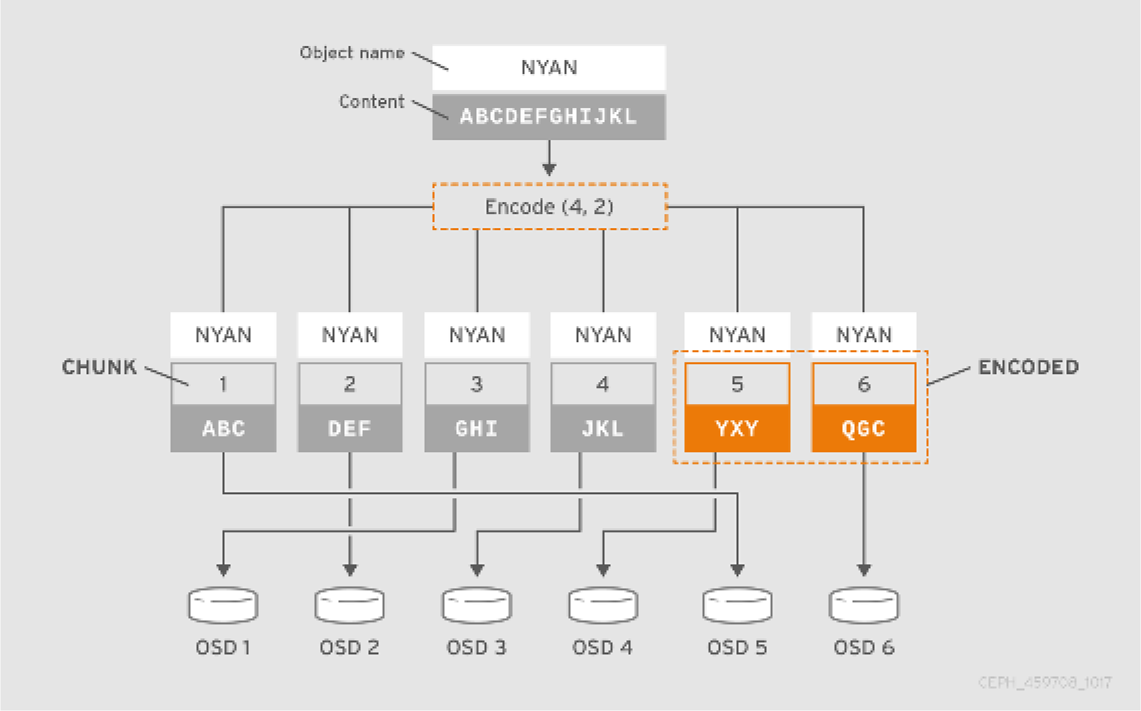
CEPH Replication strategy

Replicated



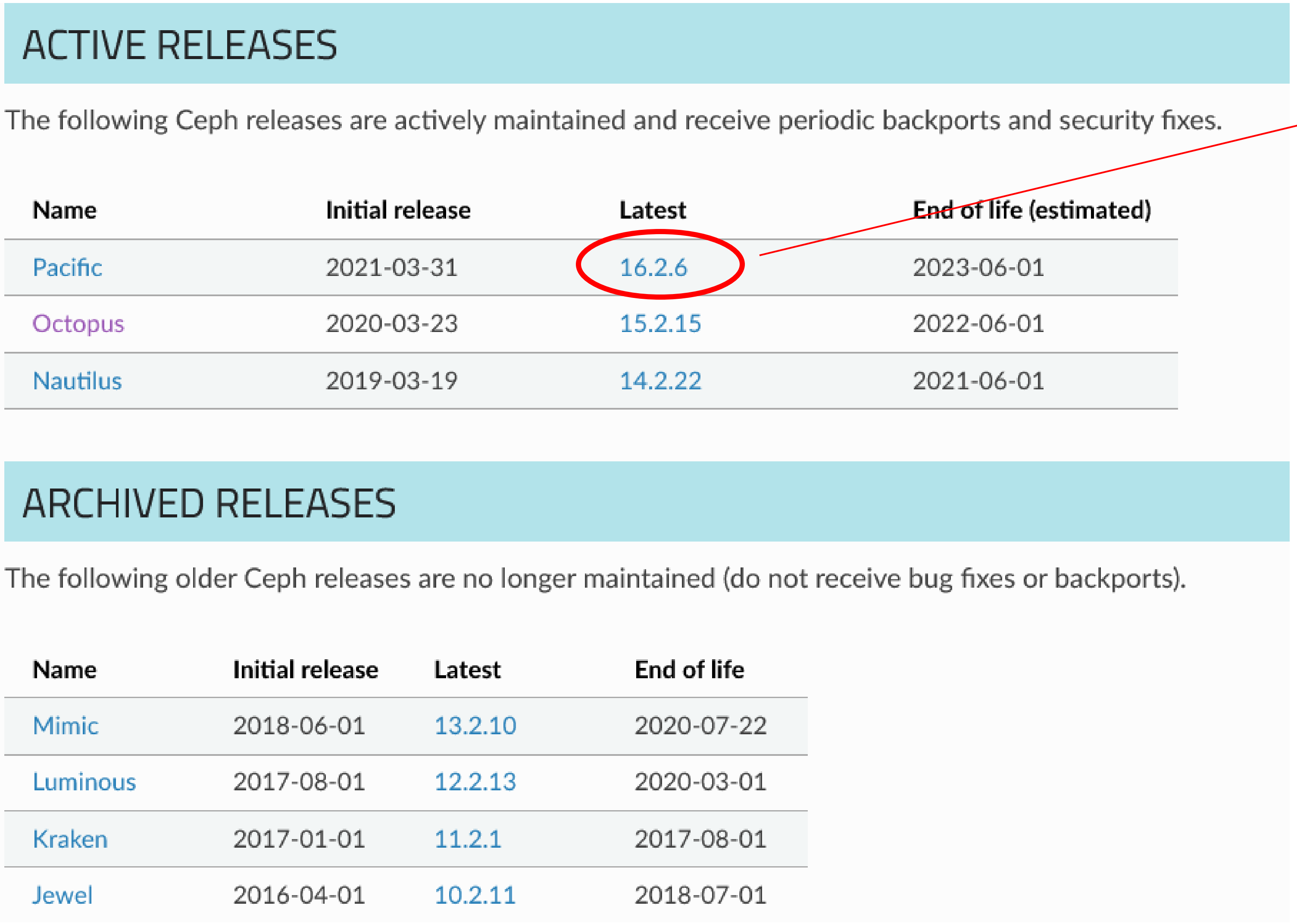
CEPH_376927_1017

Erasure coding



CEPH_459708_1017

CEPH maturity



ACTIVE RELEASES

The following Ceph releases are actively maintained and receive periodic backports and security fixes.

Name	Initial release	Latest	End of life (estimated)
Pacific	2021-03-31	16.2.6	2023-06-01
Octopus	2020-03-23	15.2.15	2022-06-01
Nautilus	2019-03-19	14.2.22	2021-06-01

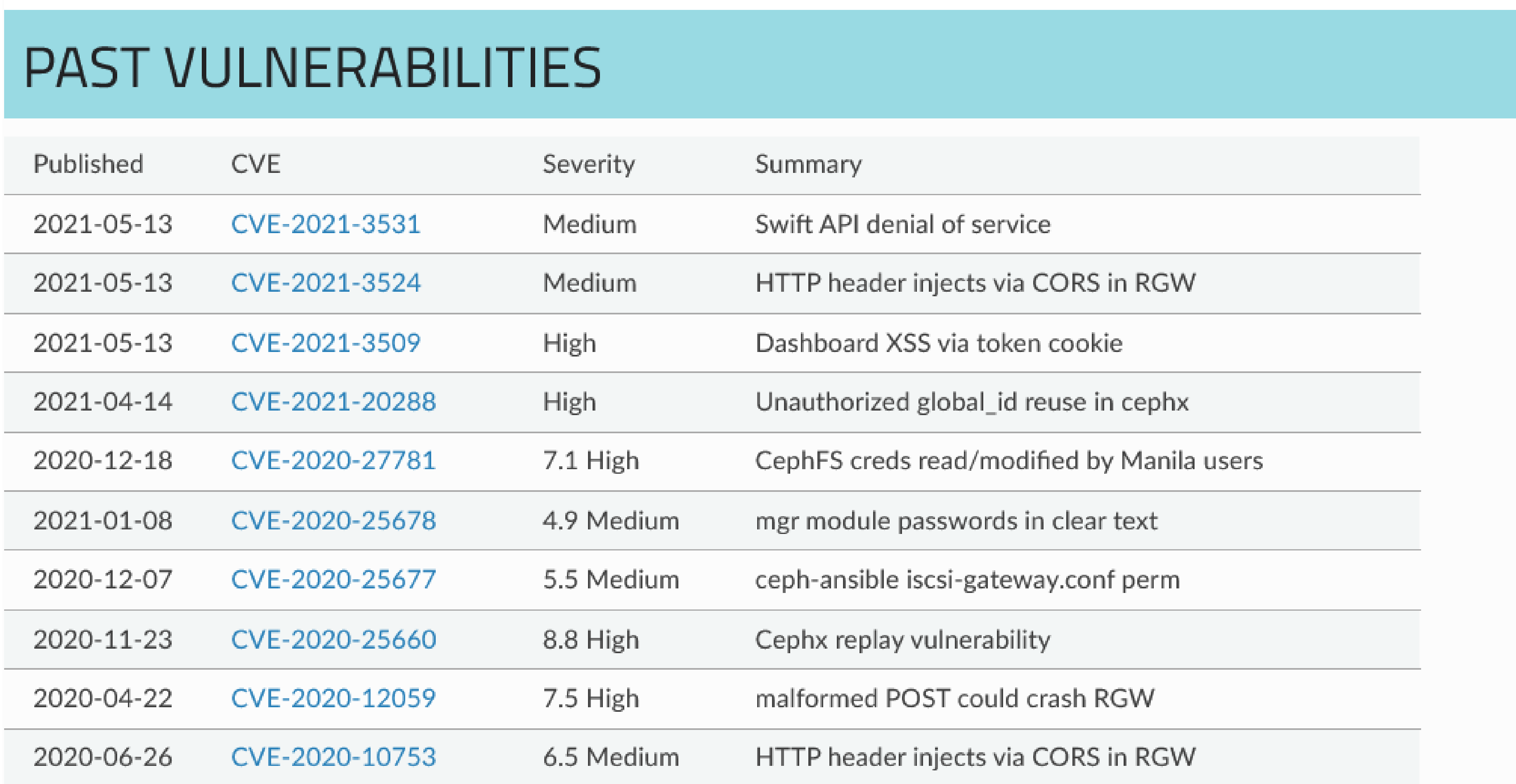
ARCHIVED RELEASES

The following older Ceph releases are no longer maintained (do not receive bug fixes or backports).

Name	Initial release	Latest	End of life
Mimic	2018-06-01	13.2.10	2020-07-22
Luminous	2017-08-01	12.2.13	2020-03-01
Kraken	2017-01-01	11.2.1	2017-08-01
Jewel	2016-04-01	10.2.11	2018-07-01

September 2021

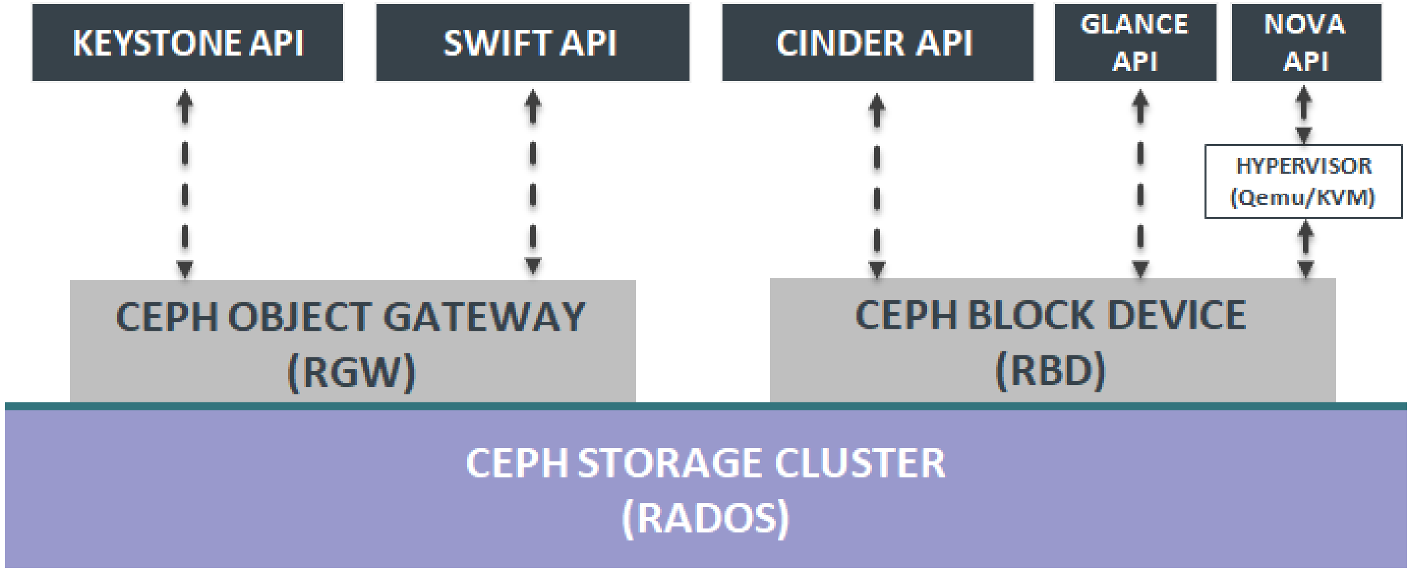
CEPH Security



PAST VULNERABILITIES

Published	CVE	Severity	Summary
2021-05-13	CVE-2021-3531	Medium	Swift API denial of service
2021-05-13	CVE-2021-3524	Medium	HTTP header injects via CORS in RGW
2021-05-13	CVE-2021-3509	High	Dashboard XSS via token cookie
2021-04-14	CVE-2021-20288	High	Unauthorized global_id reuse in cephx
2020-12-18	CVE-2020-27781	7.1 High	CephFS creds read/modified by Manila users
2021-01-08	CVE-2020-25678	4.9 Medium	mgr module passwords in clear text
2020-12-07	CVE-2020-25677	5.5 Medium	ceph-ansible iscsi-gateway.conf perm
2020-11-23	CVE-2020-25660	8.8 High	Cephx replay vulnerability
2020-04-22	CVE-2020-12059	7.5 High	malformed POST could crash RGW
2020-06-26	CVE-2020-10753	6.5 Medium	HTTP header injects via CORS in RGW

CEPH & OpenStack

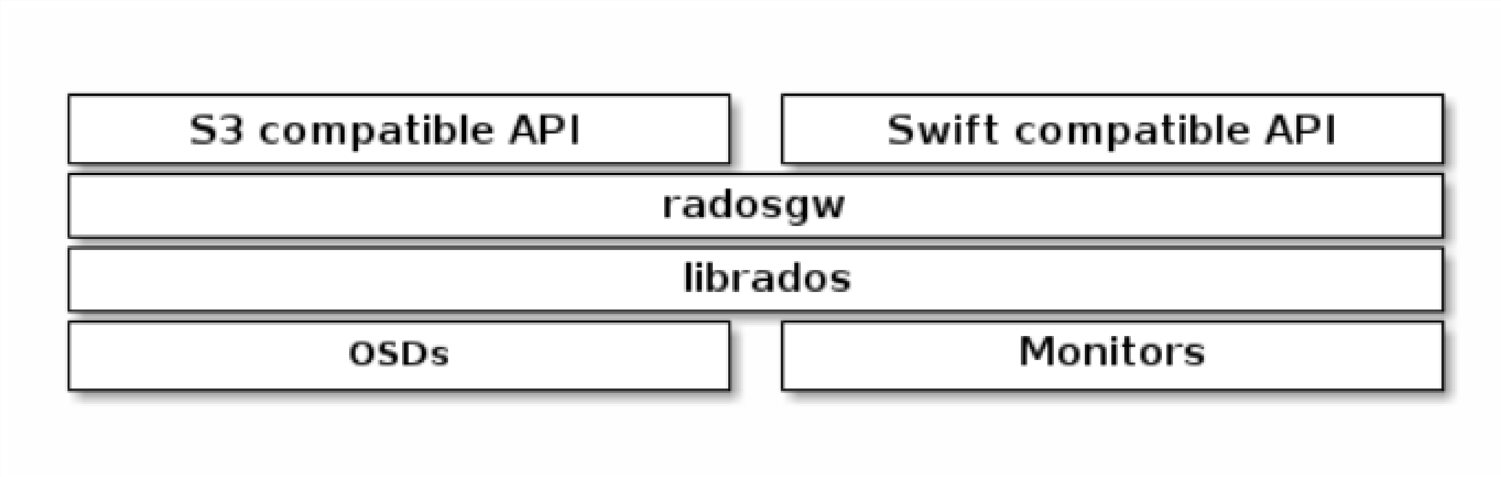


CEPH Object Gateway

Ceph Object Gateway is an object storage interface built on top of librados to provide applications with a RESTful gateway to Ceph Storage Clusters. Ceph Object Storage supports two interfaces:

1. **S3-compatible:** Provides object storage functionality with an interface that is compatible with a large subset of the Amazon S3 RESTful API.

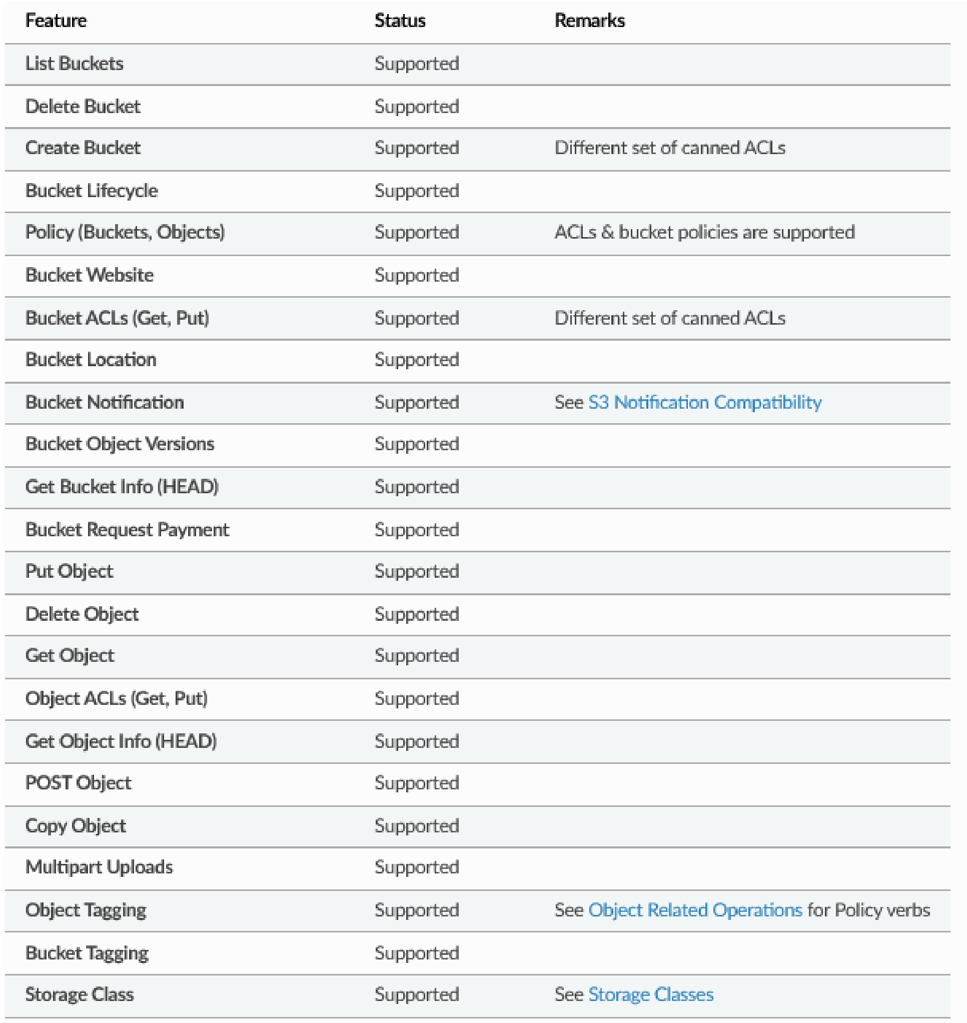
2. **Swift-compatible:** Provides object storage functionality with an interface that is compatible with a large subset of the OpenStack Swift API.



API support

Ceph Object Gateway S3 API

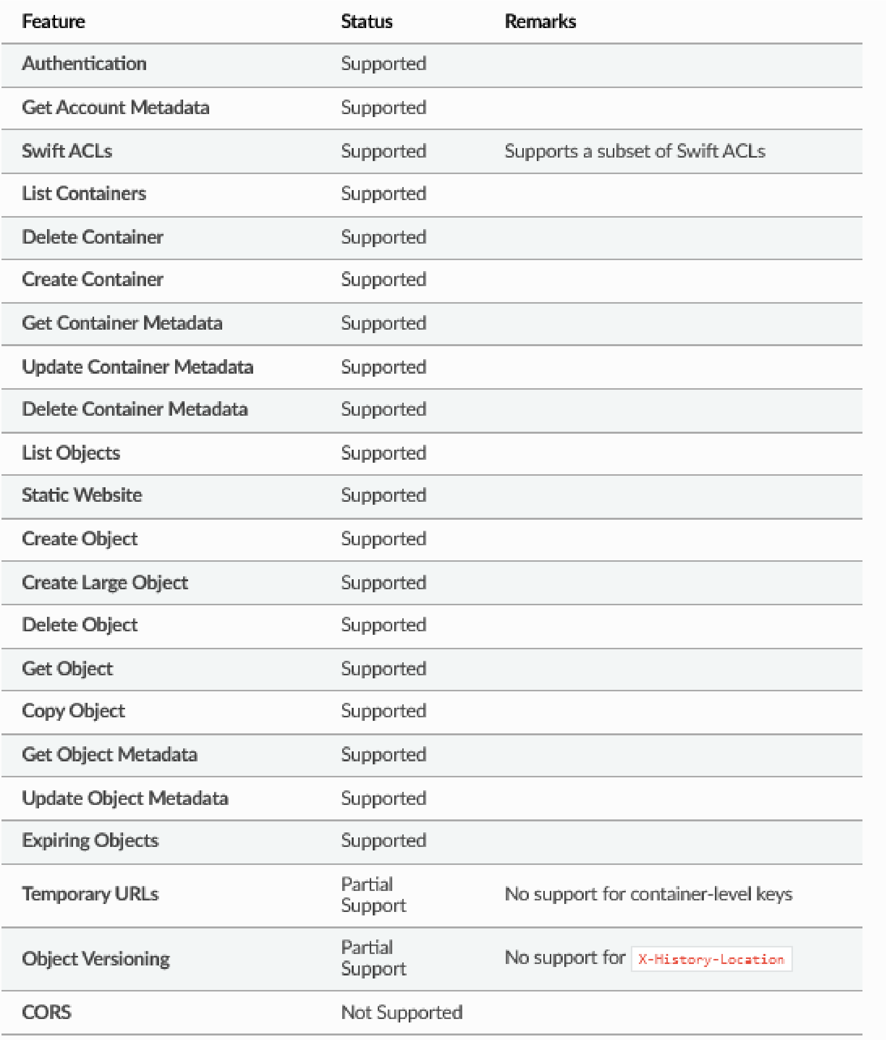
Ceph supports a RESTful API that is compatible with the basic data access model of the Amazon S3 API.



Feature	Status	Remarks
List Buckets	Supported	
Delete Bucket	Supported	
Create Bucket	Supported	Different set of canned ACLs
Bucket Lifecycle	Supported	
Policy (Buckets, Objects)	Supported	ACLs & bucket policies are supported
Bucket Website	Supported	
Bucket ACLs (Get, Put)	Supported	Different set of canned ACLs
Bucket Location	Supported	
Bucket Notification	Supported	See S3 Notification Compatibility
Bucket Object Versions	Supported	
Get Bucket Info (HEAD)	Supported	
Bucket Request Payment	Supported	
Put Object	Supported	
Delete Object	Supported	
Get Object	Supported	
Object ACLs (Get, Put)	Supported	
Get Object Info (HEAD)	Supported	
POST Object	Supported	
Copy Object	Supported	
Multipart Uploads	Supported	
Object Tagging	Supported	See Object Related Operations for Policy verbs
Bucket Tagging	Supported	
Storage Class	Supported	See Storage Classes

Ceph Object Gateway Sfift API

Ceph supports a RESTful API that is compatible with the basic data access model of the Swift API.



Feature	Status	Remarks
Authentication	Supported	
Get Account Metadata	Supported	
Swift ACLs	Supported	Supports a subset of Swift ACLs
List Containers	Supported	
Delete Container	Supported	
Create Container	Supported	
Get Container Metadata	Supported	
Update Container Metadata	Supported	
Delete Container Metadata	Supported	
List Objects	Supported	
Static Website	Supported	
Create Object	Supported	
Create Large Object	Supported	
Delete Object	Supported	
Get Object	Supported	
Copy Object	Supported	
Get Object Metadata	Supported	
Update Object Metadata	Supported	
Expiring Objects	Supported	
Temporary URLs	Partial Support	No support for container-level keys
Object Versioning	Partial Support	No support for `X-History-Location`
CORS	Not Supported	

CEPH Object Gateway: install&Configure

There are several different ways to install Ceph. Choose the method that best suits your needs.

- Cephadm installs and manages a Ceph cluster using containers and systemd
- ceph-ansible deploys and manages Ceph clusters using Ansible
- ceph-deploy* is a tool for quickly deploying clusters
- ceph-salt installs Ceph using Salt and cephadm
- Ceph can also be installed manually

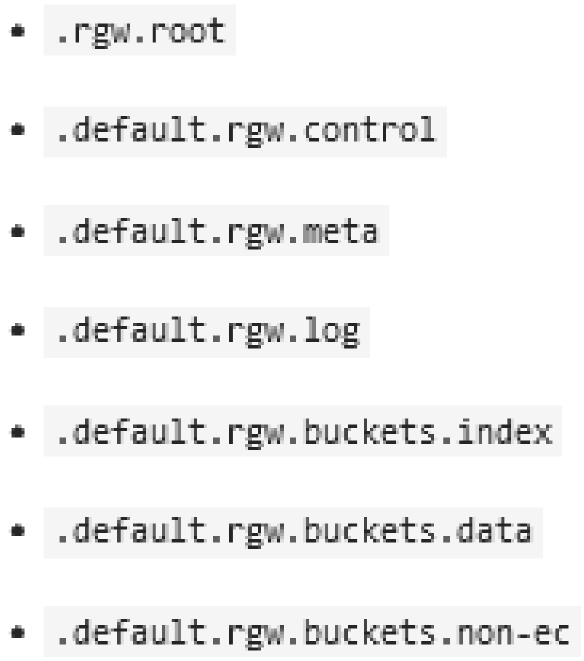
By default, Ceph Object Gateway is running on Civetweb as web server

*ceph-deploy is no longer actively maintained.
It is not tested on versions of Ceph newer than Nautilus.
It does not support RHEL8, CentOS 8.

CEPH Object Gateway: install&Configure

A Ceph Object Gateway stores administrative data in a series of pools defined in an instance's zone configuration. For example, the buckets, users, user quotas and usage statistics.

By default, Ceph Object Gateway will create the pools and map them to the default zone

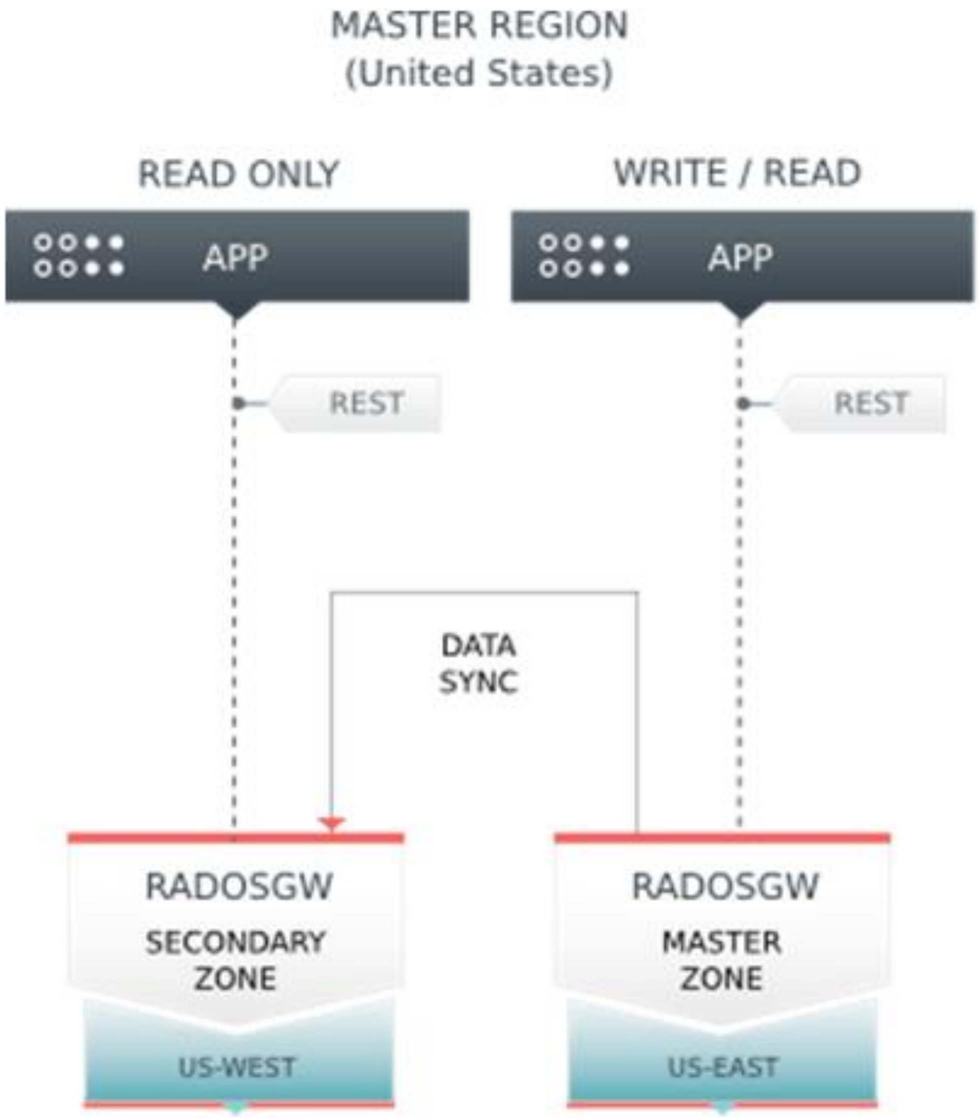


- `.rgw.root`
- `.default.rgw.control`
- `.default.rgw.meta`
- `.default.rgw.log`
- `.default.rgw.buckets.index`
- `.default.rgw.buckets.data`
- `.default.rgw.buckets.non-ec`

Federated RadosGW installations

Region: A region represents a *logical* geographic area and contains one or more zones. A cluster with multiple regions must specify a master region.

Zone: A zone is a *logical* grouping of one or more Ceph Object Gateway instance(s). A region has a master zone that processes client requests.



CEPH Object Gateway: install&Configure

A Ceph Object Gateway and Openstack integration

It is possible to integrate the Ceph Object Gateway with Keystone, the OpenStack identity service. This sets up the gateway to accept Keystone as the users authority. A user that Keystone authorizes to access the gateway will also be automatically created on the Ceph Object Gateway

CEPH side

```
[client.radosgw.gateway]
rgw keystone api version
rgw keystone url
rgw keystone admin token
rgw keystone admin token path
rgw keystone accepted roles
rgw keystone token cache size
rgw implicit tenants
or
rgw keystone admin user
rgw keystone admin password
rgw keystone admin domain
```

Keystone side

```
openstack service create --name=swift \
                         --description="Swift Service" \
                         object-store

openstack endpoint create --region RegionOne \
    --publicurl   "http://radosgw.example.com:8080/swift/v1" \
    --adminurl    "http://radosgw.example.com:8080/swift/v1" \
    --internalurl "http://radosgw.example.com:8080/swift/v1" \swift
```

Openstack User must have the role `'SwiftOperator'` to use the swift interface and commands

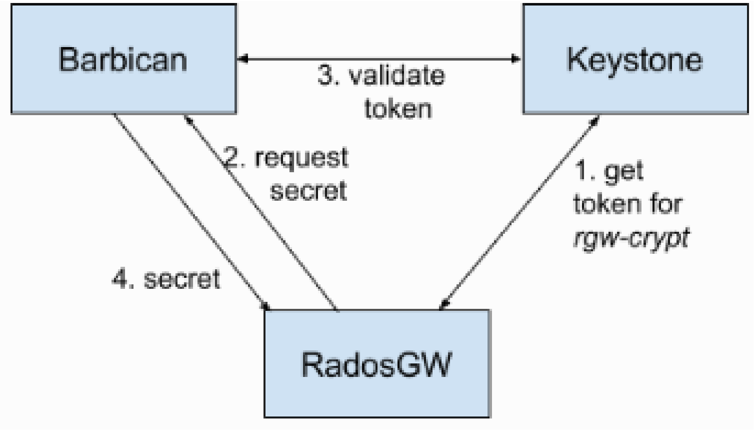
CEPH Object Gateway Integration

Integrating with OpenStack Keystone

It is possible to integrate the Ceph Object Gateway with Keystone, the OpenStack identity service. This sets up the gateway to accept Keystone as the users authority. A user that Keystone authorizes to access the gateway will also be automatically created on the Ceph Object Gateway (if didn't exist beforehand). A token that Keystone validates will be considered as valid by the gateway.

OpenStack Barbican Integration

OpenStack Barbican can be used as a secure key management service for Server-Side Encryption (SSE-KMS).

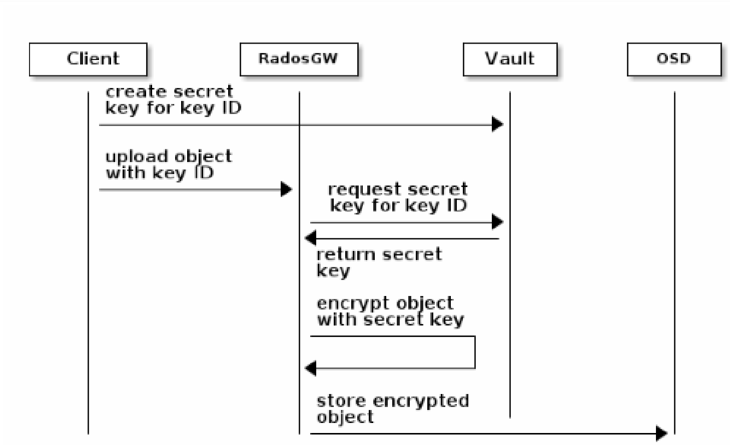


HashiCorp Vault Integration

HashiCorp Vault can be used as a secure key management service for Server-Side Encryption (SSE-KMS).

Keycloak integration

Keycloak can be setup as an OpenID Connect Identity Provider, which can be used by mobile/ web apps to authenticate their users.



Client
RadosGW
Vault
OSD
create secret key for key ID
upload object with key ID
request secret key for key ID
return secret key
encrypt object with secret key
store encrypted object

Support for Multifactor Authentication

The S3 multifactor authentication (MFA) feature allows users to require the use of one-time password when removing objects on certain buckets.

LDAP Authentication

Ceph Object Gateway can delegate authentication to an LDAP server.

Server-side encryption

Encryption

The Ceph Object Gateway supports server-side encryption (SSE) of uploaded objects: the data is sent over HTTP in its unencrypted form, and the Ceph Object Gateway stores that data in the Ceph Storage Cluster in encrypted form.

- **Customer-Provided Keys**
  - In this mode, the client passes an encryption key along with each request to read or write encrypted data. It is the client's responsibility to manage those keys and remember which key was used to encrypt each object.
  - This is implemented in S3 according to the Amazon SSE-C (customer-provided encryption keys) specification.
- **Key Management Service**
  - This mode allows keys to be stored in a secure key management service and retrieved on demand by the Ceph Object Gateway to serve requests to encrypt or decrypt data.
  - This is implemented in S3 according to the Amazon SSE-KMS (AWS Key Management Service) specification.
- **Bucket Encryption APIs**

Bucket Encryption APIs to support server-side encryption with Amazon S3-managed keys (SSE-S3)

References

- https://docs.ceph.com/en/latest/radosgw/
- https://docs.ceph.com/en/latest/radosgw/s3/
- https://docs.ceph.com/en/latest/radosgw/swift/
- https://docs.ceph.com/en/latest/radosgw/encryption/
- https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/3/html-single/storage_strategies_guide/index
- https://agenda.infn.it/event/8785/contributions/75489/attachments/54996/64853/ceph_multi-site_CCR.pdf
- https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/4/pdf/architecture_guide/red_hat_ceph_storage-4-architecture_guide-en-us.pdf