Red Hat Ceph Storage 1.3 Glossary Guide

List of Red Hat Ceph Storage terms

Red Hat Ceph Storage Documentation Team

Red Hat Ceph Storage 1.3 Glossary Guide

List of Red Hat Ceph Storage terms

Legal Notice

Abstract

This document is a glossary for Red Hat Ceph Storage.

Table of Contents

CHAPTER 1. OVERVIEW

1.1. LIST OF TERMS

Cache-tiering

A cache tier contains recently written or read data. The cache tier is a physically separate pool from the backing storage pool; however, it is transparent to the client. A cache tier typically stores data in very fast storage media like solid-state drives.

Ceph Block Device, RBD

The block storage component of Ceph.

Ceph Block Storage

The block storage "product," service or capabilities when used in conjunction with **`librbd`**, a hypervisor such as QEMU or Xen, and a hypervisor abstraction layer such as **`libvirt`**.

Ceph Client Libraries

The collection of libraries that can be used to interact with components of the Ceph System.

Ceph Clients, Ceph Client

The collection of Ceph components which can access a Ceph Storage Cluster. These include the Ceph Object Gateway, the Ceph Block Device, and their corresponding libraries, and FUSEs.

Ceph Cluster Map, cluster map

The set of maps comprising the monitor map, OSD map, PG map, and CRUSH map.

Ceph Interim Release

Versions of Ceph that have not yet been put through quality assurance testing, but may contain new features.

Ceph Monitor, MON

The Ceph monitor software.

Ceph Node, Node, Host

Any single machine or server in a Ceph System.

Ceph Object Storage

The object storage "product", service or capabilities, which consists essentially of a Ceph Storage Cluster and a Ceph Object Gateway.

Ceph Object Gateway, RADOS Gateway, RGW

The S3/Swift gateway component of Ceph.

Ceph OSD Daemon, Ceph OSD

The Ceph OSD software, which interacts with a logical disk (OSD). Sometimes, Ceph users use the term "OSD" to refer to "Ceph OSD Daemon", though the proper term is "Ceph OSD".

Ceph Point Release

Any ad-hoc release that includes only bug or security fixes.

Ceph Project

The aggregate term for the people, software, mission and infrastructure of Ceph.

Ceph Release

Any distinct numbered version of Ceph.

Ceph Release Candidate

A major version of Ceph that has undergone initial quality assurance testing and is ready for beta testers.

Ceph Stable Release

A major version of Ceph where all features from the preceding interim releases have been put through quality assurance testing successfully.

Ceph Storage Cluster, RADOS, Reliable Autonomic Distributed Object Store

The core set of storage software which stores the user's data (MON+OSD).

Ceph System, Ceph Stack

A collection of two or more components of Ceph.

Ceph Test Framework, Teuthology

The collection of software that performs scripted tests on Ceph.

cephx

The Ceph authentication protocol. Cephx operates like Kerberos, but it has no single point of failure.

Cloud Platforms, Cloud Stacks

Third party cloud provisioning platforms such as OpenStack, CloudStack, OpenNebula, ProxMox, etc.

CRUSH

Controlled Replication Under Scalable Hashing. It is the algorithm Ceph uses to compute object storage locations.

Erasure coding

Erasure coding is a method of efficiently storing data without the higher overhead of replicating it completely. Ceph supports erasure-coded pools.

Failure domain

A failure domain is any failure that prevents access to one or more OSDs. That could be a stopped daemon on a host; a hard disk failure, an OS crash, a malfunctioning NIC, a failed power supply, a network outage, a power outage, and so forth. When planning out your hardware needs, you must balance the temptation to reduce costs by placing too many responsibilities into too few failure domains, and the added costs of isolating every potential failure domain.

Object Storage Device, OSD

A physical or logical storage unit, such as a LUN. Sometimes, Ceph users use the term "OSD" to refer to Ceph OSD Daemon, though the proper term is "Ceph OSD".

Pool, Pools

Pools are logical partitions for storing objects. You may assign users with access permissions to pools.

Ruleset

A set of CRUSH data placement rules that applies to a particular pool(s).

1.2. PLACEMENT GROUPS

Acting Set

The ordered list of OSDs who are (or were as of some epoch) responsible for a particular placement group.

Authoritative History

A complete, and fully ordered set of operations that, if performed, would bring an OSD's copy of a placement group up to date.

Epoch

A (monotonically increasing) OSD map version number.

Interval

A sequence of OSD map epochs during which the **Acting Set** and **Up Set** for a particular placement group do not change.

Last Epoch Clean

The last **Epoch** at which all nodes in the **Acting set** for a particular placement group were completely up to date (both placement group logs and object contents). At this point, **recovery** is deemed to have been completed.

Last Epoch Start

The last epoch at which all nodes in the **Acting Set** for a particular placement group agreed on an **Authoritative History**. At this point, **Peering** is deemed to have been successful.

Missing Set

Each OSD notes update log entries and if they imply updates to the contents of an object, adds that object to a list of needed updates. This list is called the **Missing Set** for that **`<OSD,PG>`**.

Peering

The process of bringing all of the OSDs that store a Placement Group (PG) into agreement about the state of all of the objects (and their metadata) in that PG. Note that agreeing on the state does not mean that they all have the latest contents.

Placement Group

Placement groups are logical containers for objects. Placement groups get assigned to OSDs. By addressing hundreds or thousands of placement groups to distribute data across the cluster, Ceph doesn't have to maintain a list of objects or specifically requests millions of objects when rebalancing a cluster.

Placement Group Information

Basic metadata about the placement group's creation epoch, the version for the most recent write to the placement group, **last epoch started**, **last epoch clean**, and the beginning of the **current interval**. Any inter-OSD communication about placement groups includes the **PG Info**, such that any OSD that knows a placement group exists (or once existed) also has a lower bound on **last epoch clean** or **last epoch started**.

Placement Group Log

A list of recent updates made to objects in a placement group. Note that these logs can be truncated after all OSDs in the **Acting Set** have acknowledged up to a certain point.

Primary

The member (and by convention first) of the **Acting Set**, that is responsible for coordination peering, and is the only OSD that will accept client-initiated writes to objects in a placement group.

up_thru

Before a **Primary** can successfully complete the **Peering** process, it must inform a monitor that is alive through the current OSD map **Epoch** by having the monitor set its **up_thru** in the osd map. This helps **Peering** ignore previous **Acting Sets** for which **Peering** never completed after certain sequences of failures, such as the second interval below:

- **acting set** = [A,B]
- **acting set** = [A]
- **acting set** = [] very shortly after (e.g., simultaneous failure, but staggered detection)
- **acting set** = [B] (B restarts, A does not)

Up Set

The ordered list of OSDs responsible for a particular placement group for a particular epoch according to CRUSH. Normally this is the same as the **Acting Set**, except when the **Acting Set** has been explicitly overridden via **`pg_temp`** in the OSD Map.

Stray

An OSD that is not a member of the current **Acting Set**, but has not yet been told that it can delete its copies of a particular placement group.

Recovery

Ensuring that copies of all of the objects in a placement group are on all of the OSDs in the **Acting Set**. Once **Peering** has been performed, the **Primary** can start accepting write operations, and **Recovery** can proceed in the background.

Replica

A non-primary OSD in the **Acting Set** for a placement group (and who has been recognized as such and **activated** by the primary).

1.3. ERASURE CODING

chunk

When the encoding function is called, it returns chunks of the same size. Data chunks which can be concatenated to reconstruct the original object and coding chunks which can be used to rebuild a lost chunk.

K

The number of data *chunks*, i.e. the number of *chunks* in which the original object is divided. For instance if *K* = 2 a 10KB object will be divided into *K* objects of 5KB each.

M

The number of coding *chunks*, i.e. the number of additional *chunks* computed by the encoding functions. If there are 2 coding *chunks*, it means 2 OSDs can be out without losing data.