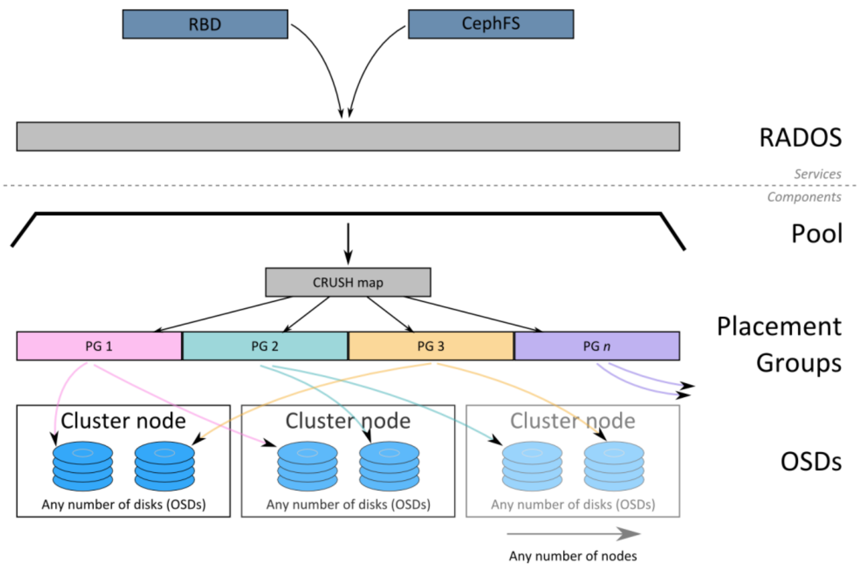
Chapter 1:

Introduction to CEPH

- What is CEPH?
- CEPH Components.
- Data Placements.
- Network Configuration.
- CEPH Architecture.
- CEPH Storage Cluster.
- Storing Data.
- Scalability and High Availability.
- OSD Interaction.
- CEPH Block Devices.
- Reference.

<u>What is CEPH?</u>—

Ceph is a free software storage platform that **stores data on a single distributed computer cluster**, and provides interfaces for object-, block- and file-level storage. It's a fault tolerant distributed clustered file system.



<u>CEPH COMPONENTS</u> ---

A Ceph Storage Cluster requires at least **one Ceph Monitor** and at least **two Ceph OSD** Daemons. The **Ceph Metadata** Server is essential when running Ceph Filesystem clients.

1) OSD—

OSD stands for **Object Storage Device and corresponds to physical disk**. OSD **stores data, handles data replication, recovery, backfilling, rebalancing,** and provides some monitoring information to Ceph Monitors by checking other Ceph OSD Daemons for a heartbeat. A Ceph Storage Cluster requires at least two Ceph OSD Daemons to achieve an active + clean state when the cluster makes two copies of your data.

2) Monitors-

A *Ceph Monitor* **maintains maps** of the cluster state, including the monitor map, the OSD map, the Placement Group (PG) map, and the CRUSH map. Ceph **maintains a history** (called an "epoch") of each state change **in the Ceph Monitors**, Ceph OSD Daemons, and PGs.

3) MDS—

A *Ceph Metadata Server* (MDS) stores metadata on behalf of the *Ceph Filesystem* (i.e., Ceph Block Devices and Ceph Object Storage do not use MDS). Ceph Metadata Servers make it feasible for POSIX file system users to execute basic commands like ls, find, etc. without placing an enormous burden on the Ceph Storage Cluster.

Ceph stores a client's data as objects within storage pools. Using the CRUSH algorithm, Ceph calculates which placement group should contain the object, and further calculates which Ceph OSD Daemon should store the placement group. The CRUSH algorithm enables the Ceph Storage Cluster to scale, rebalance, and recover dynamically.

<u>DATA PLACEMENT</u>—

Ceph stores, replicates and rebalances data objects across a RADOS cluster dynamically. With many different users storing objects in different pools for different purposes on countless OSDs, Ceph operations require some data placement planning. The main data placement planning concepts in Ceph include:

1) 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.** To store data in a pool, you must have an authenticated user with permissions for the pool.
- A pool is the layer at which most user-interaction takes place. This is the important stuff like GET, PUT and DELETE actions for objects in a pool.
- Pools contain a number of PGs, not shared with other pools (if you have multiple pools). The number of PGs in a pool is defined when the pool is first created, and can't be changed later.

2) Placement Group (PG) —

- Ceph maps objects to placement groups (PGs). Placement groups (PGs) are **fragments of a logical objects pool that place objects as a group into OSDs**. Placement groups reduce the amount of per-object metadata when Ceph stores the data in OSDs. **A larger number of placement groups (e.g., 100 per OSD) leads to better balancing.**
- Placement groups help ensure performance and scalability, as tracking metadata for each individual object would be too costly.
- A PG collects objects from the next layer up and manages them as a collection. It represents a mostly-static mapping to one or more underlying OSDs. **Replication is done at the PG layer**: the degree of replication (number of copies) is asserted higher, up at the Pool level, and all PGs in a pool will replicate stored objects into multiple OSDs.
- As an example in a system with 3-way replication:
  PG-1 might map to OSDs 1, 37 and 99
  PG-2 might map to OSDs 4, 22 and 41
  PG-3 might map to OSDs 18, 26 and 55
  Etc.

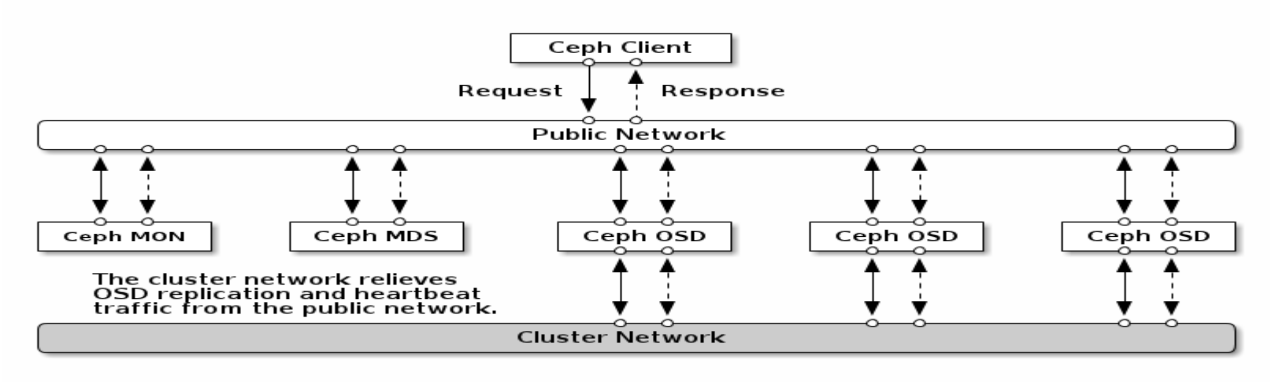
- Any object that happens to be stored on PG-1 will be written to all three OSDs (1,37,99). Any object stored in PG-2 will be written to its three OSDs (4,22,41). And so on.

3) CRUSH maps—

- CRUSH is a big part of what allows Ceph to scale without performance bottlenecks, without limitations to scalability, and without a single point of failure. **CRUSH maps provide the physical topology of the cluster to the CRUSH algorithm to determine where the data for an object and its replicas should be stored,** and how to do so across failure domains for added data safety among other things.
- CRUSH mappings are specified on a per-pool basis, and serve to skew the **distribution of objects into OSDs according to administrator-defined policy**. This is important for ensuring that **replicas don't end up on the same disk/host/rack**, which would break the entire point of having replicate copies. The CRUSH algorithm determines how to store and retrieve data by computing data storage locations. CRUSH empowers **Ceph clients to communicate with OSDs directly rather than through a centralized server or broker.** With an algorithmically determined method of storing and retrieving data, Ceph avoids a single point of failure, a performance bottleneck, and a physical limit to its scalability.
- **CRUSH requires a map of your cluster**, and uses the CRUSH map to pseudo-randomly **store and retrieve data in OSDs with a uniform distribution of data across the cluster**. A CRUSH map is written by hand, then compiled and passed to the cluster. CRUSH maps contain a list of OSDs, a list of 'buckets' for aggregating the devices into physical locations, and **a list of rules that tell CRUSH how it should replicate data in a Ceph cluster's pools.**
- When you create a configuration file and deploy Ceph with mkcephfs, Ceph generates a default CRUSH map for your configuration. The default CRUSH map is fine for your Ceph sandbox environment. However, when you deploy a large-scale data cluster, you should give significant consideration to developing a custom CRUSH map because it will help you manage your Ceph cluster, improve performance and ensure data safety.
- For example, **if an OSD goes down, a CRUSH map can help you can locate the physical data centre, room, row and rack** of the host with the failed OSD in the event you need to use onsite support or replace hardware.
- Similarly, CRUSH may help you identify faults more quickly. For example, if all OSDs in a particular rack go down simultaneously, the fault may lie with a network switch or power to the rack or the network switch rather than the OSDs themselves.

NETWORK CONFIGURATION –

We recommend running a Ceph Storage Cluster with **two networks**: **a public (front-side) network and a cluster (back-side) network**. To support two networks, each *Ceph Node* will need to have more than one NIC



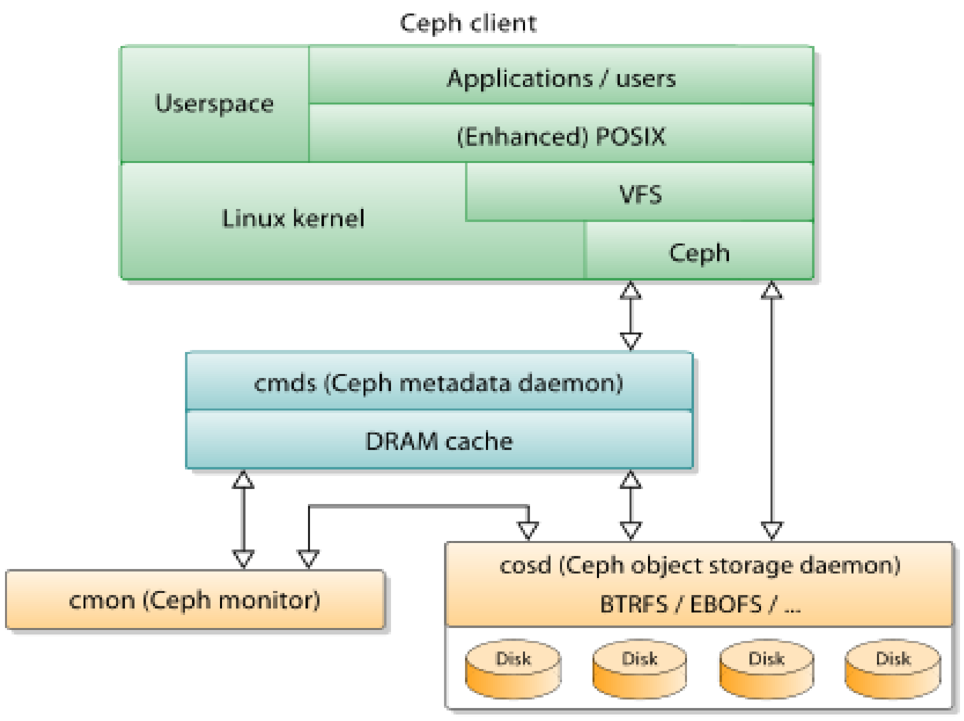
There are several reasons to consider operating two separate networks:

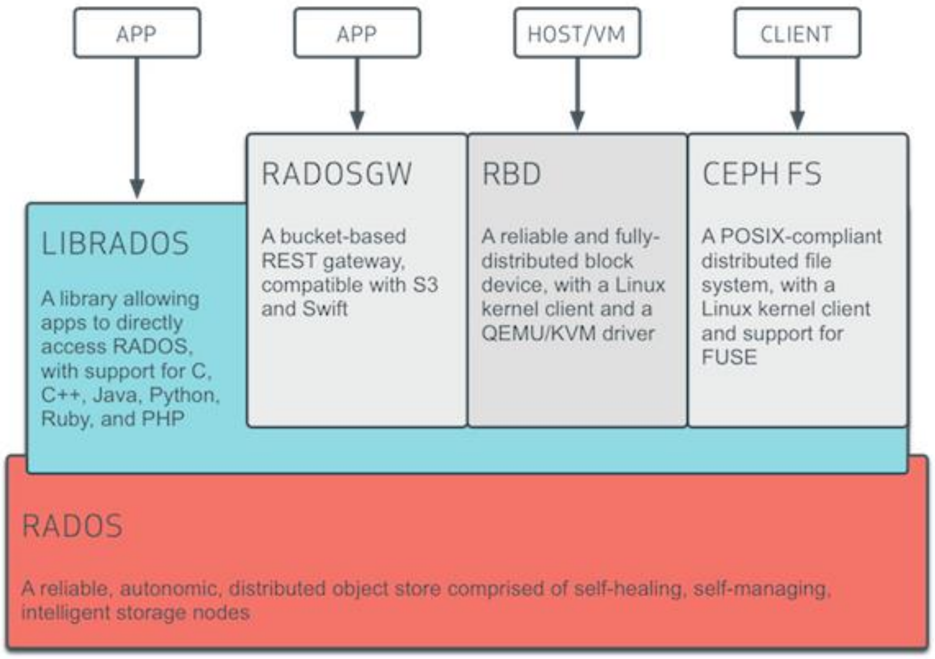
1. **Performance:** Ceph OSD Daemons handle **data replication for the Ceph Clients**. When Ceph OSD Daemons replicate data more than once, **the network load between Ceph OSD Daemons easily affects the network load between Ceph Clients and the Ceph Storage Cluster**. This can introduce latency and create a performance problem. Recovery and rebalancing can also introduce significant latency on the public network.
2. **Security**: While most people are generally civil, a very tiny segment of the population likes to engage in what's known as a **Denial of Service** (DoS) attack. When traffic between Ceph OSD Daemons gets interrupted this may prevent users from reading and writing data. A great way to defeat this type of attack is to maintain a **completely separate cluster network that doesn't connect directly to the internet**. Also, consider using Message Signatures to defeat spoofing attacks.

CEPH ARCHITECTURE

Ceph is highly reliable, easy to manage, and free. The power of Ceph can transform your company's IT infrastructure and your ability to manage vast amounts of data. **Ceph delivers extraordinary scalability–thousands of clients accessing Petabytes to Exabyte of**

data. A *Ceph Storage Cluster* accommodates large numbers of nodes, which communicate with each other to replicate and redistribute data dynamically.





APP
APP
HOST/VM
CLIENT
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
LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP
RADOS
A reliable, autonomic, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

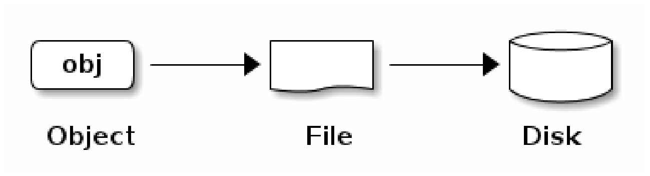
CEPH STORAGE CLUSTER

Storage cluster clients and each *Ceph OSD Daemon* use the **CRUSH algorithm to efficiently compute information about data location, instead of having to depend on a**

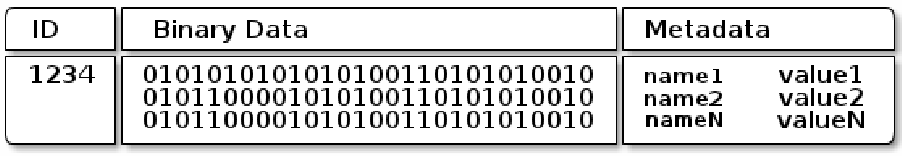
central lookup table. Ceph's high-level features include providing a native interface to the Ceph Storage Cluster via librados, and a number of service interfaces built on top of librados.

STORING DATA

The Ceph Storage Cluster receives data from *Ceph Clients* – whether it comes through a *Ceph Block Device*, *Ceph Object Storage*, the *Ceph Filesystem* or a custom implementation created using librados–and it stores the data as objects.



Each **object corresponds to a file in a filesystem**, which is stored on an *Object Storage Device*. Ceph OSD Daemons handle the read/write operations on the storage disks. **Ceph OSD Daemons store all data with no hierarchy of directories.** An object has an identifier, binary data, and metadata consisting of a set of name/value pairs. **Ceph File System uses metadata to store file attributes such as the file owner, created date, last modified date** and so forth.



ID	Binary Data	Metadata
1234	0101010101010100110101010010 0101100001010100110101010010 0101100001010100110101010010	name1 value1 name2 value2 nameN valueN

SCALABILITY AND HIGH AVAILABILITY

In traditional architectures, **clients talk to a centralized component** (e.g., a gateway, broker, API, facade, etc.), **which acts as a single point of entry to a complex subsystem**. This imposes a limit to both performance and scalability, while introducing a single point of failure (i.e., if the centralized component goes down, the whole system goes down, too).

Ceph **eliminates the centralized gateway** to enable clients to interact with Ceph OSD Daemons directly. Ceph OSD Daemons create **object replicas on other Ceph Nodes to ensure data safety and high availability**. Ceph also uses a cluster of monitors to ensure high availability. To eliminate centralization, Ceph uses an algorithm called CRUSH.

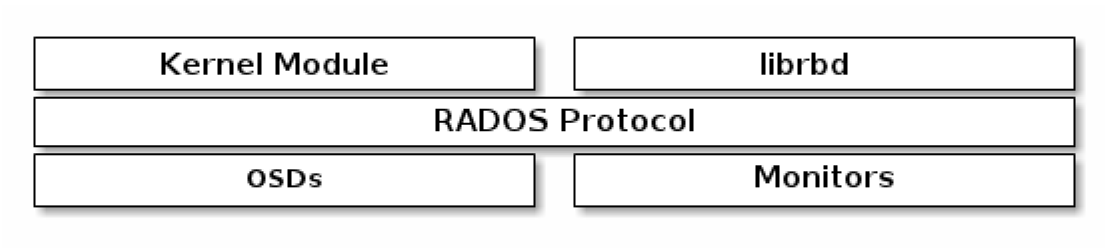
Before Ceph Clients can read or write data, they must contact a Ceph Monitor to obtain the most recent copy of the cluster map. A Ceph Storage Cluster can operate with a single monitor; however, this introduces a single point of failure (i.e., if the monitor goes down, Ceph Clients cannot read or write data).

For added reliability and fault tolerance, Ceph supports a cluster of monitors. In a cluster of monitors, latency and other faults can cause one or more monitors to fall behind the current state of the cluster. For this reason, Ceph must have agreement among various monitor instances regarding the state of the cluster.

<u>OSD INTERACTION</u> –

After you have completed your initial Ceph configuration, you may deploy and run Ceph. When you execute **a command such as ceph health or ceph -s, the *Ceph Monitor* reports on the current state of the *Ceph Storage Cluster***. The Ceph Monitor knows about the Ceph Storage Cluster by requiring reports from each *Ceph OSD Daemon*, and by receiving reports from Ceph OSD Daemons about the status of their neighbouring Ceph OSD Daemons. If the Ceph Monitor doesn't receive reports, or if it receives reports of changes in the Ceph Storage Cluster, the Ceph Monitor updates the status of the *Ceph Cluster Map*.

<u>CEPH BLOCK DEVICES</u> –



Ceph block devices are thin-provisioned, resizable and **store data striped over multiple OSDs** in a Ceph cluster. Ceph block devices leverage RADOS capabilities such as snapshotting, replication and consistency. Ceph's RADOS Block Devices (RBD) interacts with OSDs using kernel modules or the librbd library.

Ceph's block devices **deliver high performance with infinite scalability to kernel modules**, or to KVMs such as **Qemu, and cloud-based computing systems like OpenStack and CloudStack that rely on libvirt and Qemu to integrate with Ceph block devices.** You can use the same cluster to operate the Ceph RADOS Gateway, the Ceph FS filesystem, and Ceph block devices simultaneously.

<u>REFERENCE –</u>

http://docs.ceph.com/docs/v0.80.5/architecture/