# Geo replication and disaster recovery for cloud object storage with Ceph rados gateway

Orit Wasserman
Senior Software engineer
owasserm@redhat.com

Linuxcon EU 2016

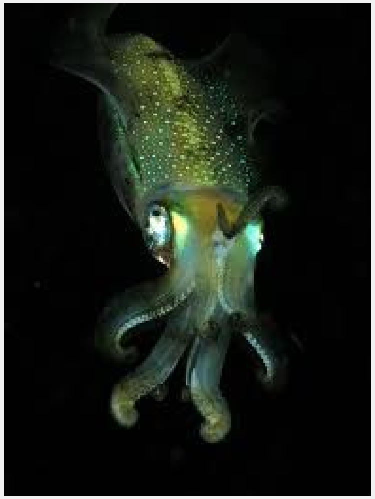
# AGENDA

- What is Ceph?
- Rados Gateway (radosgw) architecture
- Geo replication in radosgw
- Questions

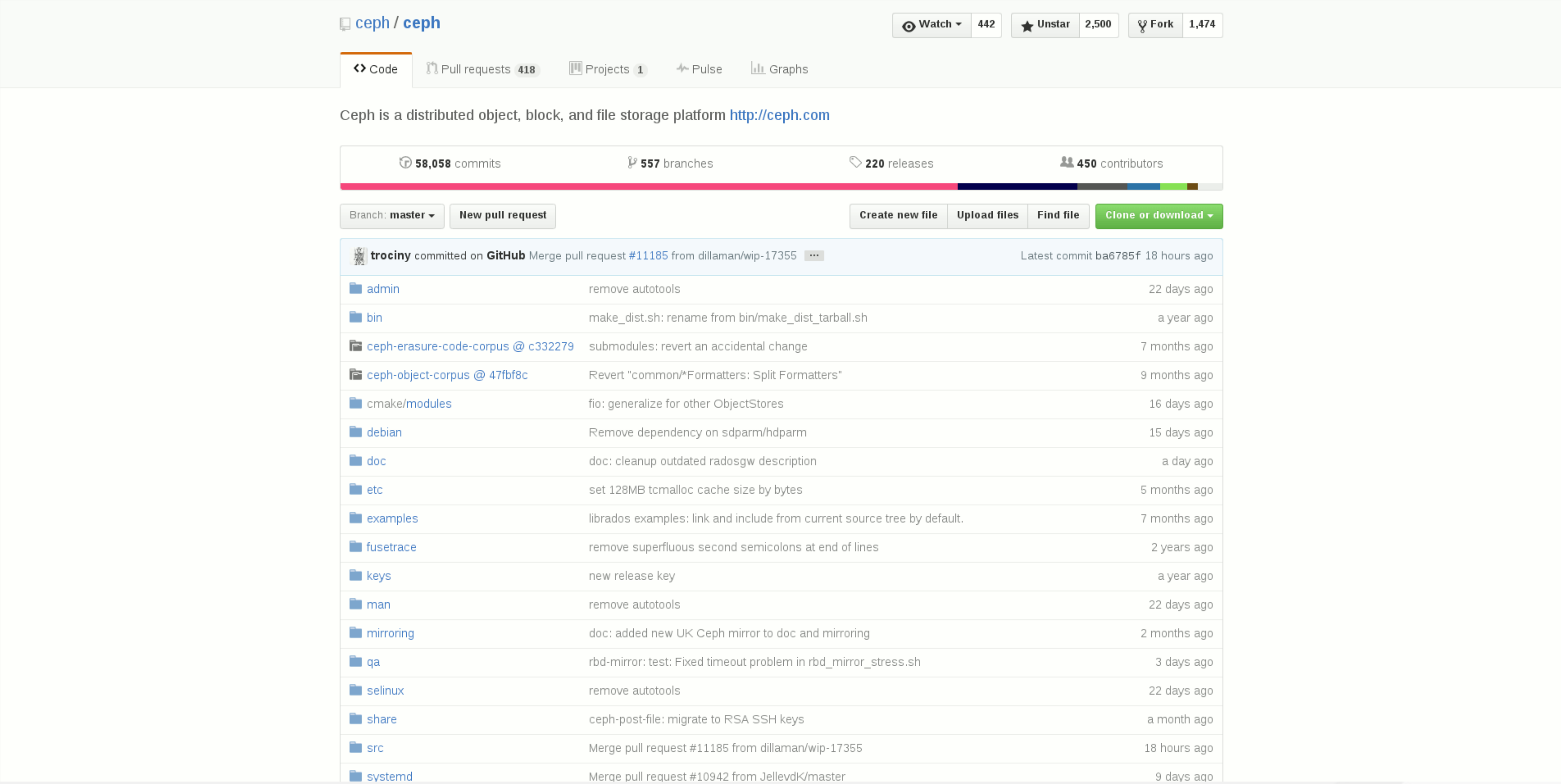
# Ceph architecture

# Cephalopod

A cephalopod is any member of the molluscan class Cephalopoda. These exclusively marine animals are characterized by bilateral body symmetry, a prominent head, and a set of arms or tentacles (muscular hydrostats) modified from the primitive molluscan foot. The study of cephalopods is a branch of malacology known as teuthology.



# Ceph



ceph / ceph

Watch 442 | Unstar 2,500 | Fork 1,474

Ceph is a distributed object, block, and file storage platform http://ceph.com

58,058 commits | 557 branches | 220 releases | 450 contributors

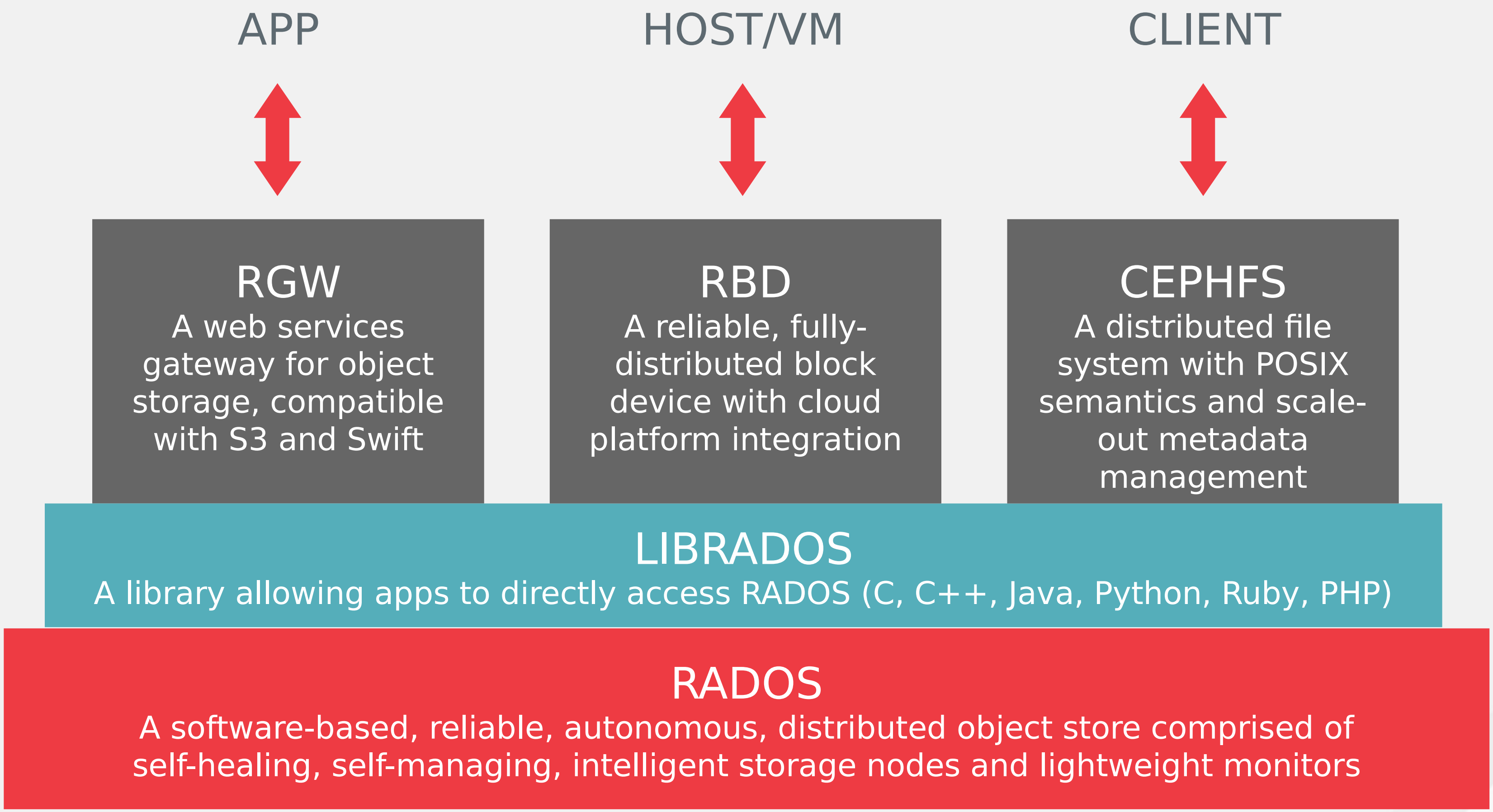
trociny committed on GitHub Merge pull request #11185 from dillaman/wip-17355 … Latest commit ba6785f 18 hours ago

| | | |
|---|---|---|
| admin | remove autotools | 22 days ago |
| bin | make_dist.sh: rename from bin/make_dist_tarball.sh | a year ago |
| ceph-erasure-code-corpus @ c332279 | submodules: revert an accidental change | 7 months ago |
| ceph-object-corpus @ 47fbf8c | Revert "common/*Formatters: Split Formatters" | 9 months ago |
| cmake/modules | fio: generalize for other ObjectStores | 16 days ago |
| debian | Remove dependency on sdparm/hdparm | 15 days ago |
| doc | doc: cleanup outdated radosgw description | a day ago |
| etc | set 128MB tcmalloc cache size by bytes | 5 months ago |
| examples | librados examples: link and include from current source tree by default. | 7 months ago |
| fusetrace | remove superfluous second semicolons at end of lines | 2 years ago |
| keys | new release key | a year ago |
| man | remove autotools | 22 days ago |
| mirroring | doc: added new UK Ceph mirror to doc and mirroring | 2 months ago |
| qa | rbd-mirror: test: Fixed timeout problem in rbd_mirror_stress.sh | 3 days ago |
| selinux | remove autotools | 22 days ago |
| share | ceph-post-file: migrate to RSA SSH keys | a month ago |
| src | Merge pull request #11185 from dillaman/wip-17355 | 18 hours ago |
| systemd | Merge pull request #10942 from JellevdK/master | 9 days ago |

# Ceph

- Open source
- Software defined storage
- Distributed
- No single point of failure
- Massively scalable
- Self healing
- Unified storage: object, block and file
- IRC: OFTC #ceph,#ceph-devel
- Mailing lists:
    - ceph-users@ceph.com
    - ceph-devel@ceph.com

# Ceph architecture



APP

HOST/VM

CLIENT

RGW

A web services gateway for object storage, compatible with S3 and Swift

RBD

A reliable, fully-distributed block device with cloud platform integration

CEPHFS

A distributed file system with POSIX semantics and scale-out metadata management

LIBRADOS

A library allowing apps to directly access RADOS (C, C++, Java, Python, Ruby, PHP)

RADOS

A software-based, reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes and lightweight monitors

# Rados

- Reliable Distributed Object Storage
- Replication
- Erasure coding
- Flat object namespace within each pool
  - Different placement rules
- Strong consistency (CP system)
- Infrastructure aware, dynamic topology
- Hash-based placement (CRUSH)
- Direct client to server data path

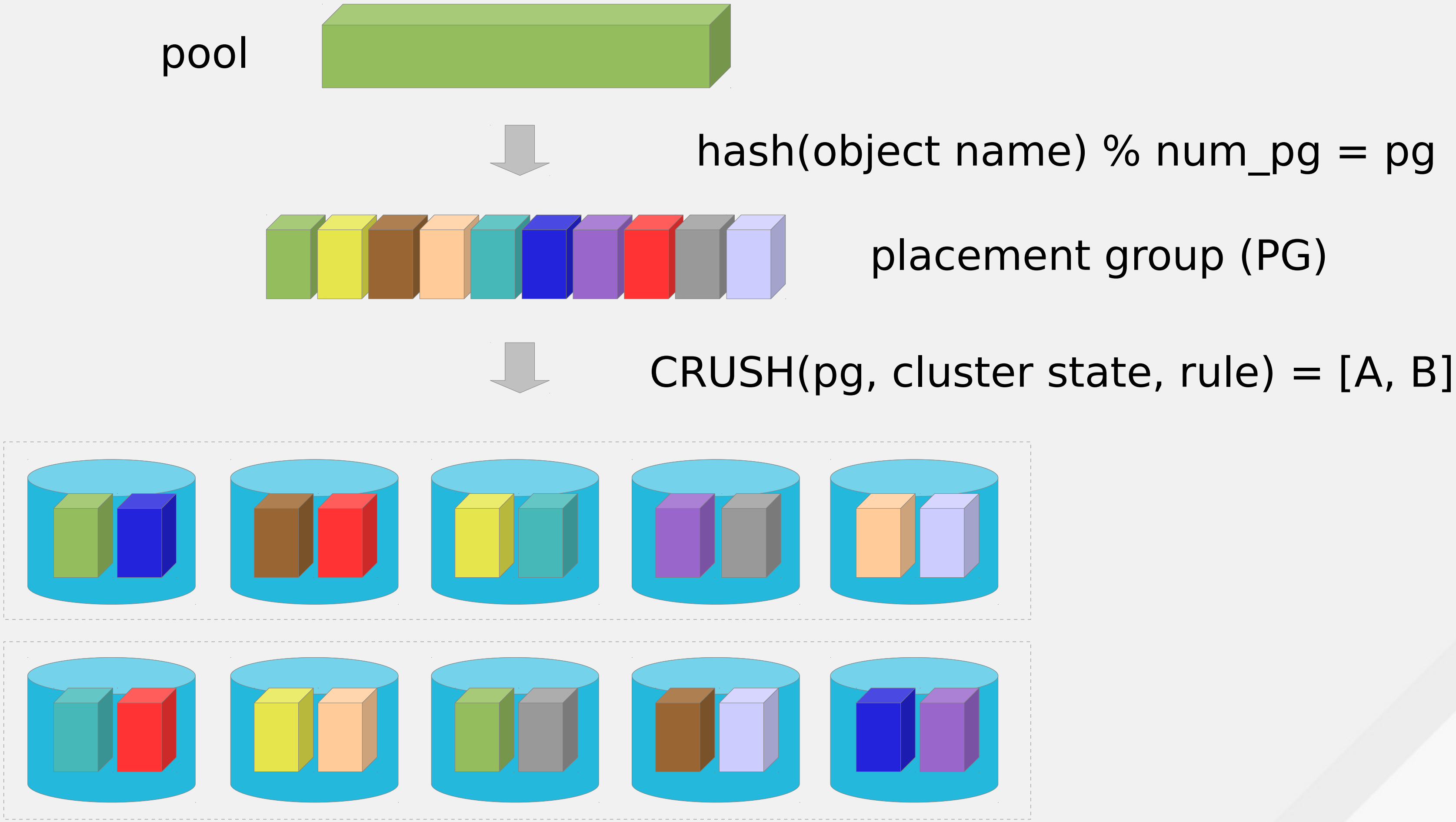
# OSD node

- 10s to 10000s in a cluster
- One per disk (or one per SSD, RAID group…)
- Serve stored objects to clients
- Intelligently peer for replication & recovery

# Monitor node

- Maintain cluster membership and state
- Provide consensus for distributed decision-making
- Small, odd number
- These do not serve stored objects to clients

# object placement



pool

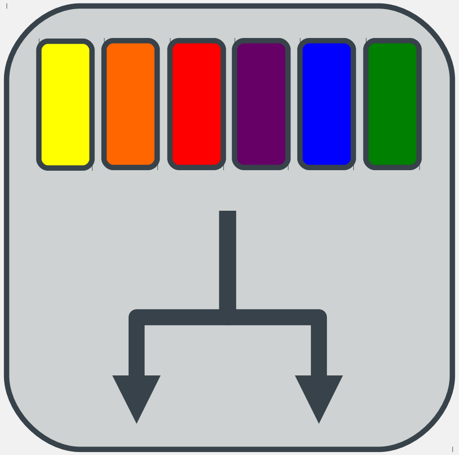
hash(object name) % num_pg = pg

placement group (PG)

CRUSH(pg, cluster state, rule) = [A, B]

# Crush

- pseudo-random placement algorithm
  - fast calculation, **no lookup**
  - repeatable, deterministic
- statistically uniform distribution
- stable mapping
  - limited data migration on change
- rule-based configuration
  - infrastructure topology aware
  - adjustable replication
  - allows weighting

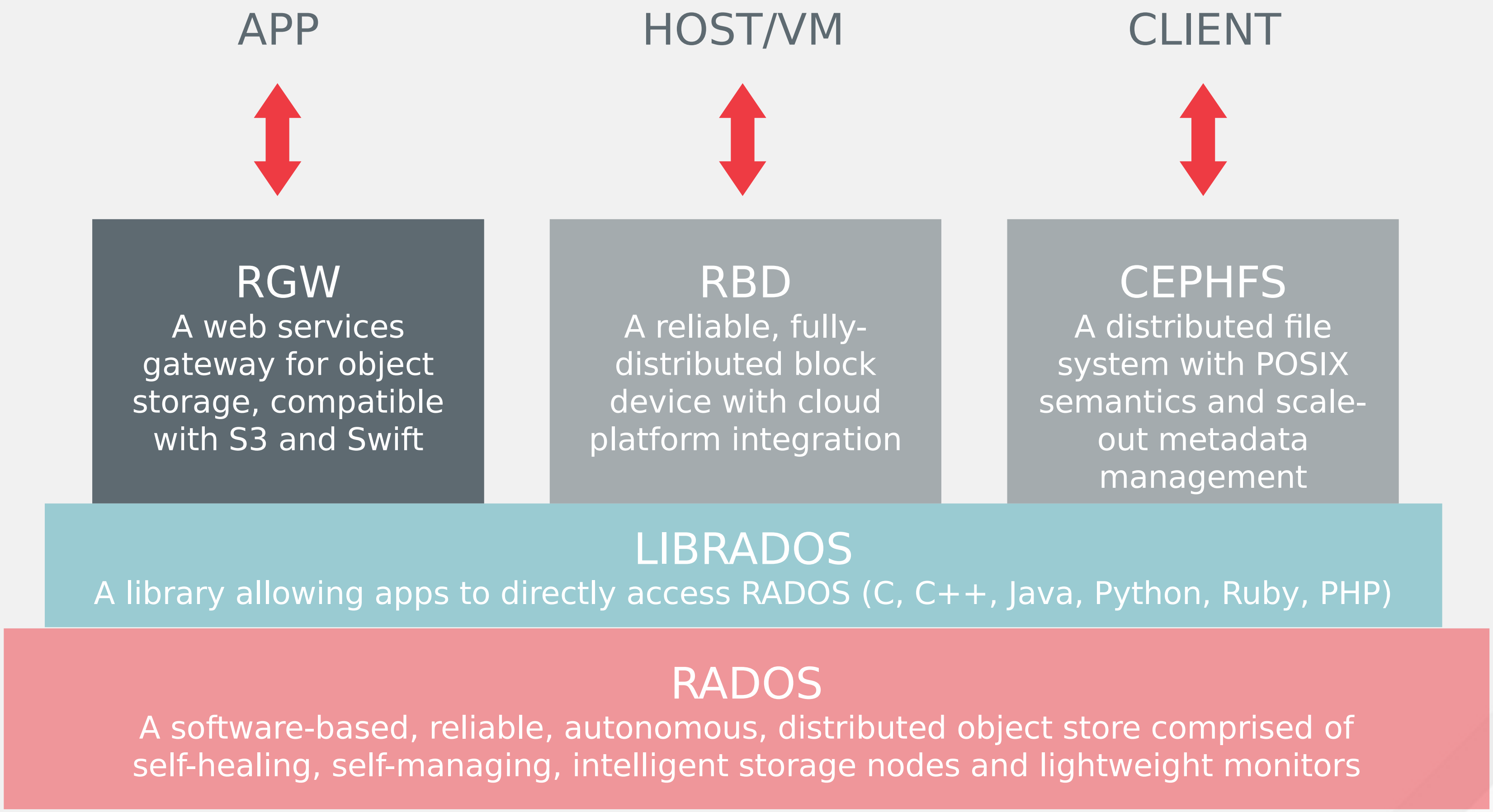


# Librados API

- Efficient key/value storage inside an object
- Atomic single-object transactions
  - update data, attr, keys together
  - atomic compare-and-swap
- Object-granularity snapshot infrastructure
- Partial overwrite of existing data
- Single-object compound atomic operations
- RADOS classes (stored procedures)
- Watch/Notify on an object

# Rados Gateway

# Rados Gateway



APP

HOST/VM

CLIENT

**RGW**
A web services gateway for object storage, compatible with S3 and Swift

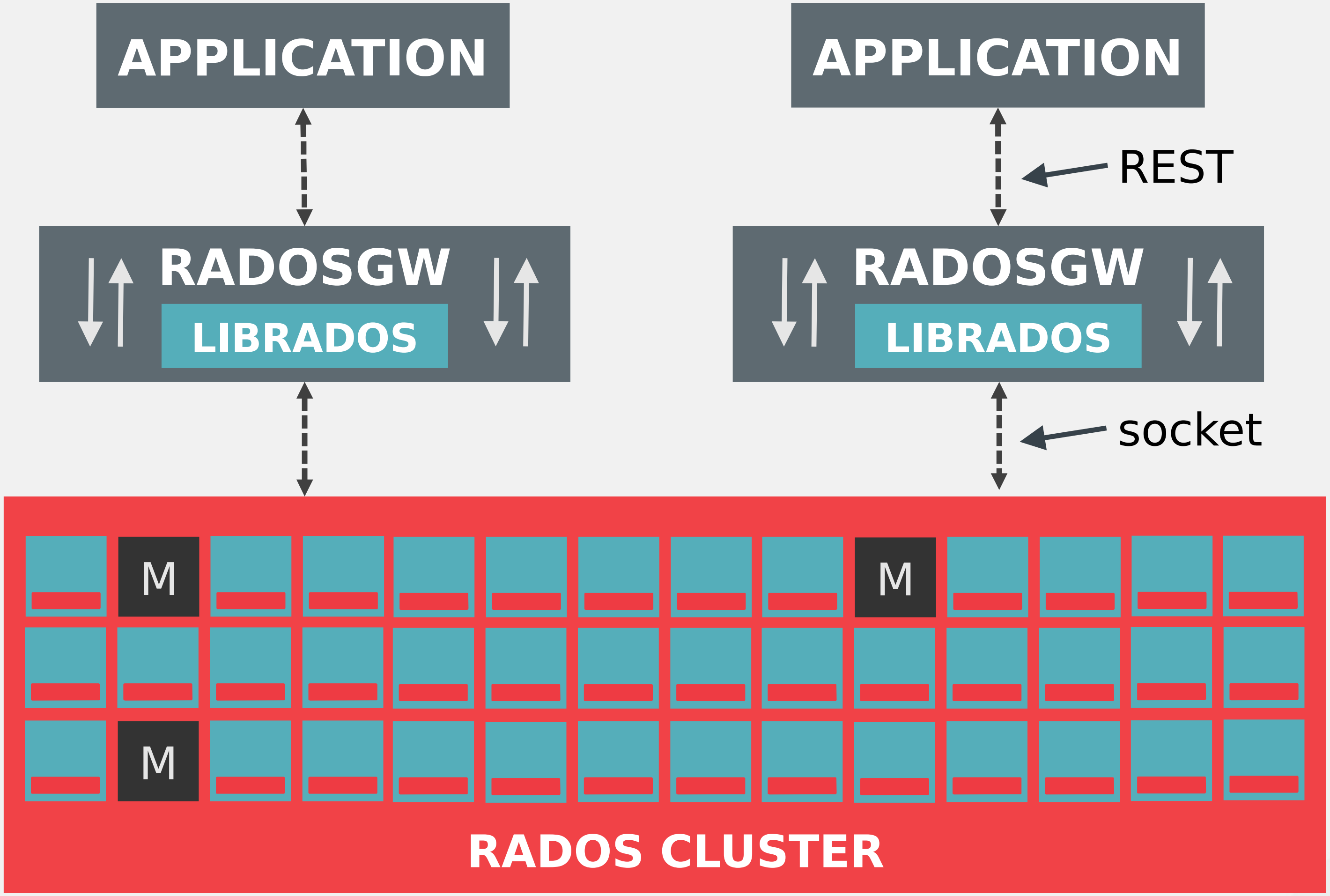
**RBD**
A reliable, fully-distributed block device with cloud platform integration

**CEPHFS**
A distributed file system with POSIX semantics and scale-out metadata management

**LIBRADOS**
A library allowing apps to directly access RADOS (C, C++, Java, Python, Ruby, PHP)

**RADOS**
A software-based, reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes and lightweight monitors

# Rados Gateway



APPLICATION

RADOSGW

LIBRADOS

APPLICATION

REST

RADOSGW

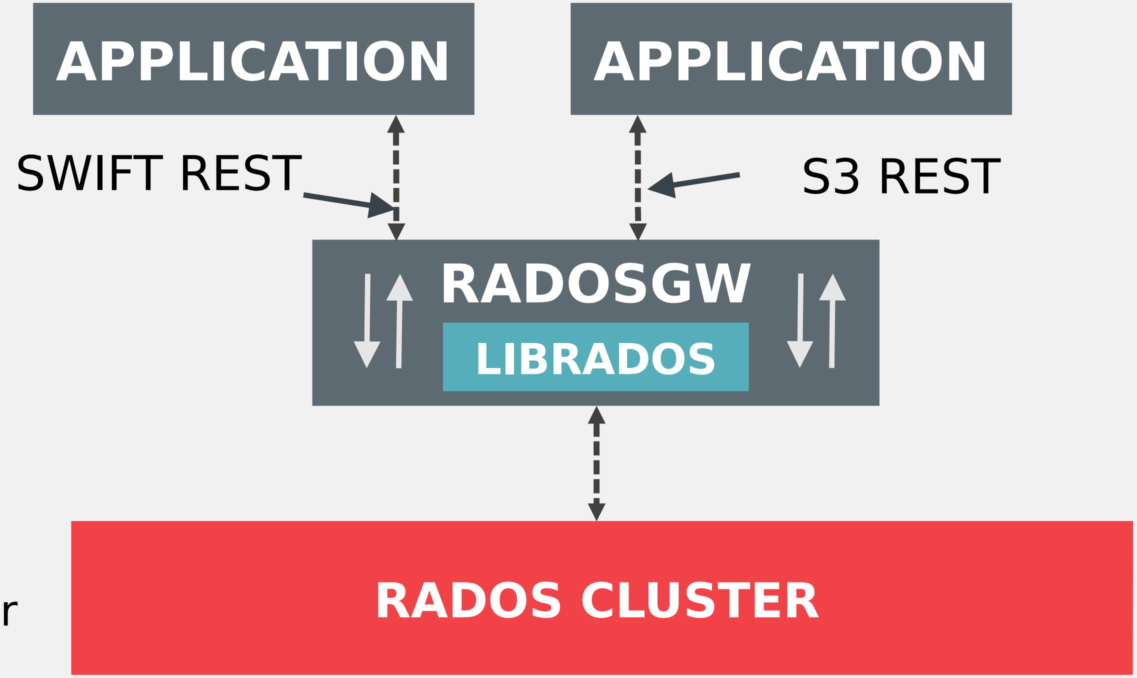
LIBRADOS

socket

RADOS CLUSTER

# RESTful OBJECT STORAGE

- Data
  - Users
  - Buckets
  - Objects
  - ACLs
- Authentication
- APIs
  - S3
  - Swift
  - Librgw (used for NFS)



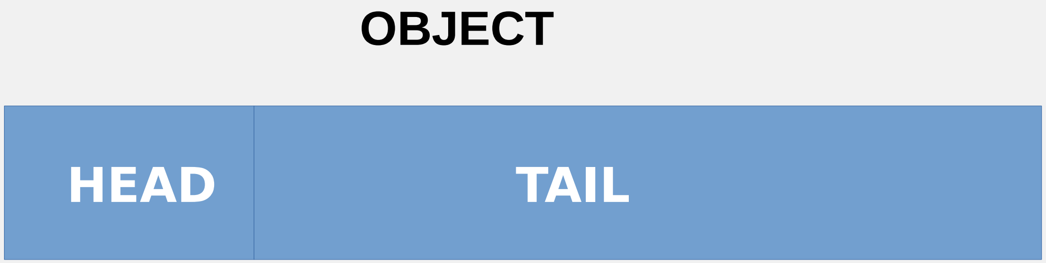
# RGW vs RADOS object

- RADOS
  - Limited object sizes
  - Mutable objects
  - Not indexed
  - No per-object ACLs
- RGW
  - Large objects (Up to a few TB per object)
  - Immutable objects
  - Sorted bucket listing
  - Permissions

# RGW objects requirements

- Large objects
- Fast small object access
- Fast access to object attributes
- Buckets can consist of a very large number of objects

# RGW objects



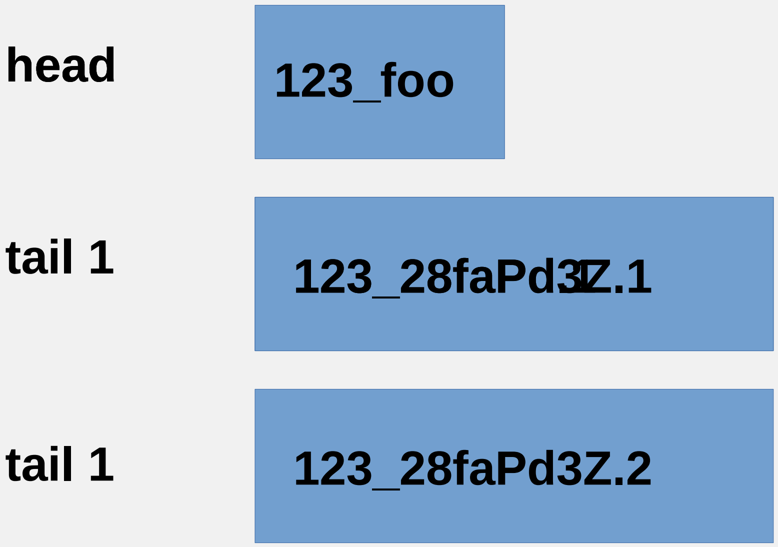
- Head
  - Single rados object
  - Object metadata (acls, user attributes, manifest)
  - Optional start of data
- Tail
  - Striped data
  - 0 or more rados objects

# RGW Objects

**OBJECT: foo**

**BUCKET: boo**

**BUCKET ID: 123**

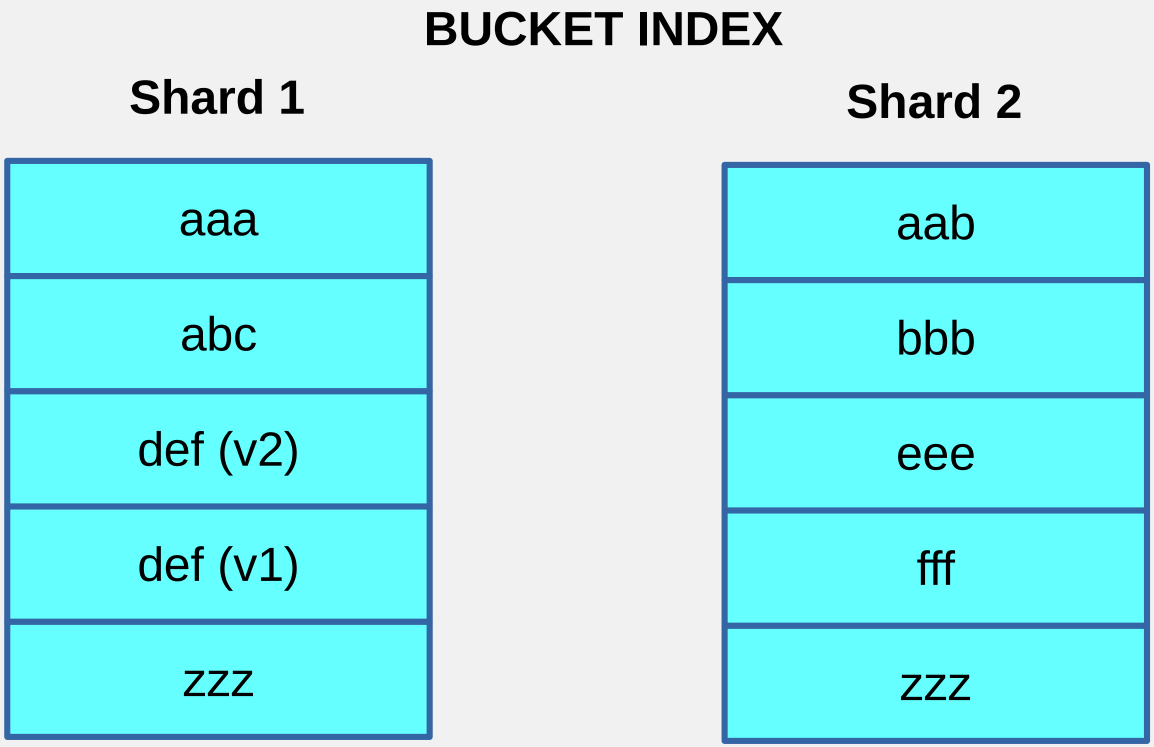


**head** 123_foo

**tail 1** 123_28faPd3Z.1

**tail 1** 123_28faPd3Z.2

# RGW bucket index



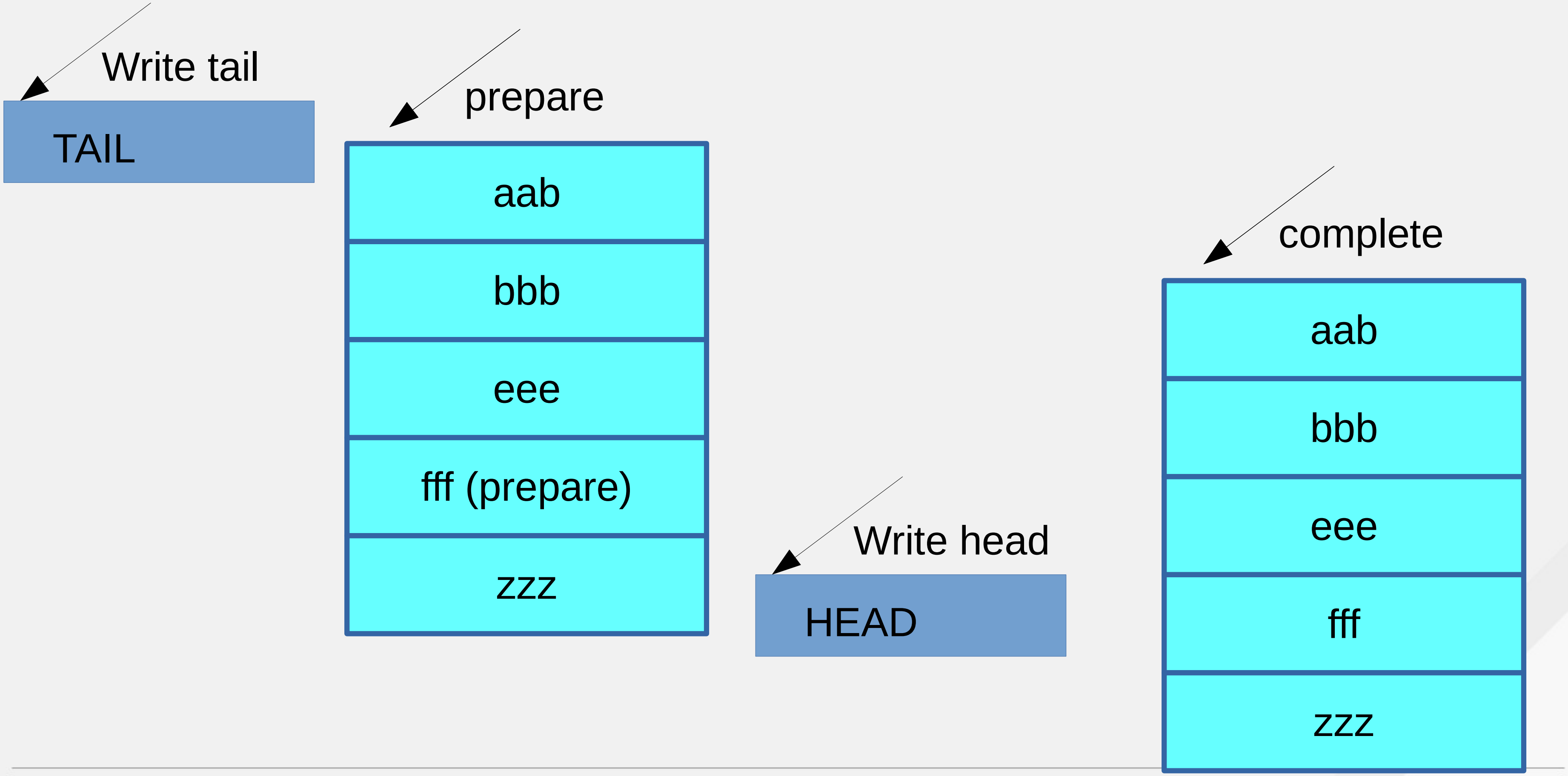
**BUCKET INDEX**

| Shard 1 | Shard 2 |
|---|---|
| aaa | aab |
| abc | bbb |
| def (v2) | eee |
| def (v1) | fff |
| zzz | zzz |

# RGW object creation

- When creating a new object we need to:
  - Update bucket index
  - Create head object
  - Create tail objects
- All those operations need to be consist

# RGW object creation



Write tail

TAIL

prepare

aab

bbb

eee

fff (prepare)

zzz

Write head

HEAD

complete

aab

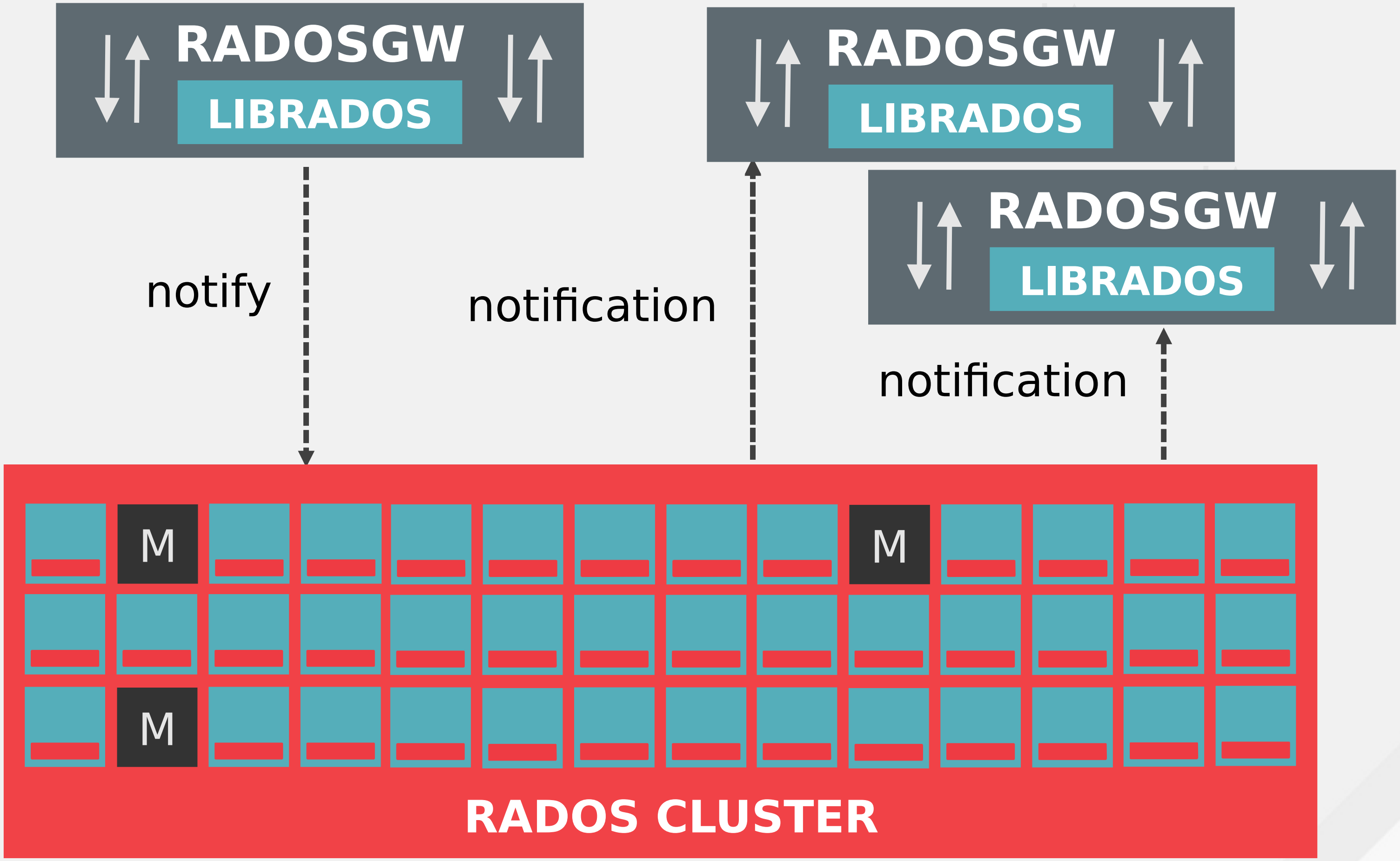
bbb

eee

fff

zzz

# RGW metadata cache

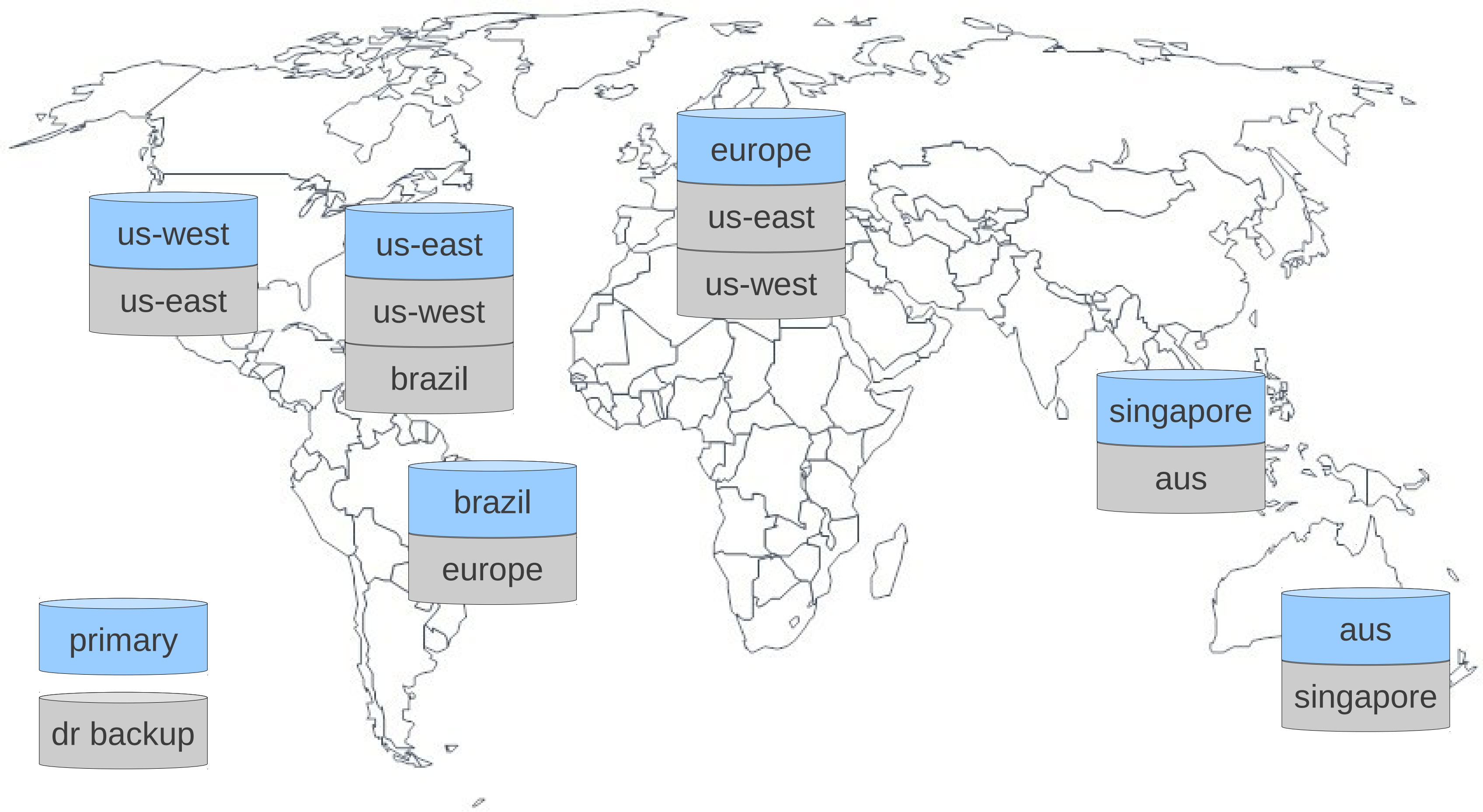


# Geo replication

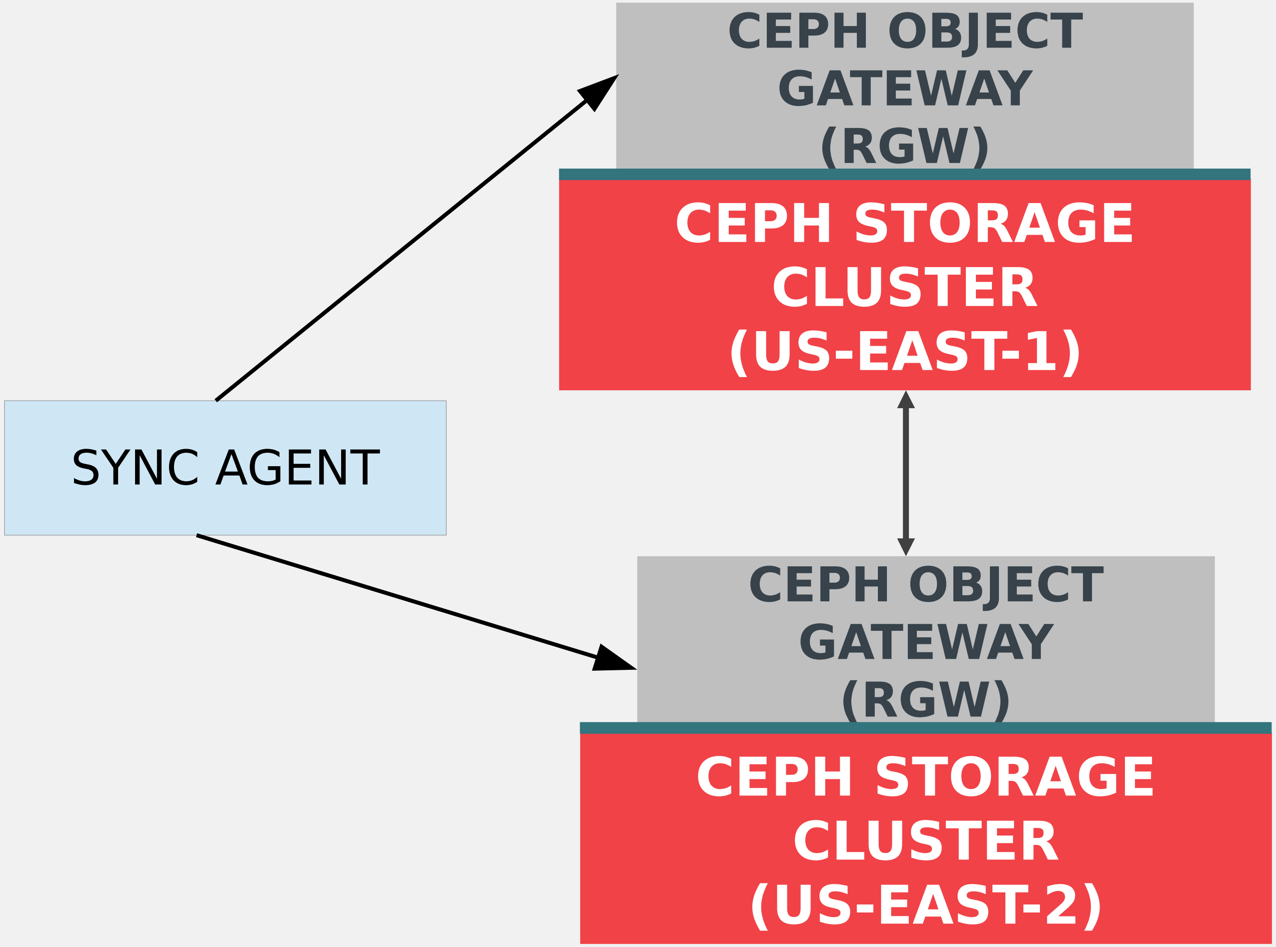
# Geo replication

- Data is replicated on different physical locations
- High and unpredictable latency between those location
- Used for disaster recovery

# Geo replication



# Sync agent (old implementation)


CEPH OBJECT GATEWAY (RGW)
CEPH STORAGE CLUSTER (US-EAST-1)
SYNC AGENT
CEPH OBJECT GATEWAY (RGW)
CEPH STORAGE CLUSTER (US-EAST-2)

# Sync agent (old implementation)

- External python implementation
- No Active/Active support
- Hard to configure
- Complicate failover mechanism
- No clear sync status indication
- A single bucket synchronization could dominate the entire sync process
- Configuration updates require restart of the gateways

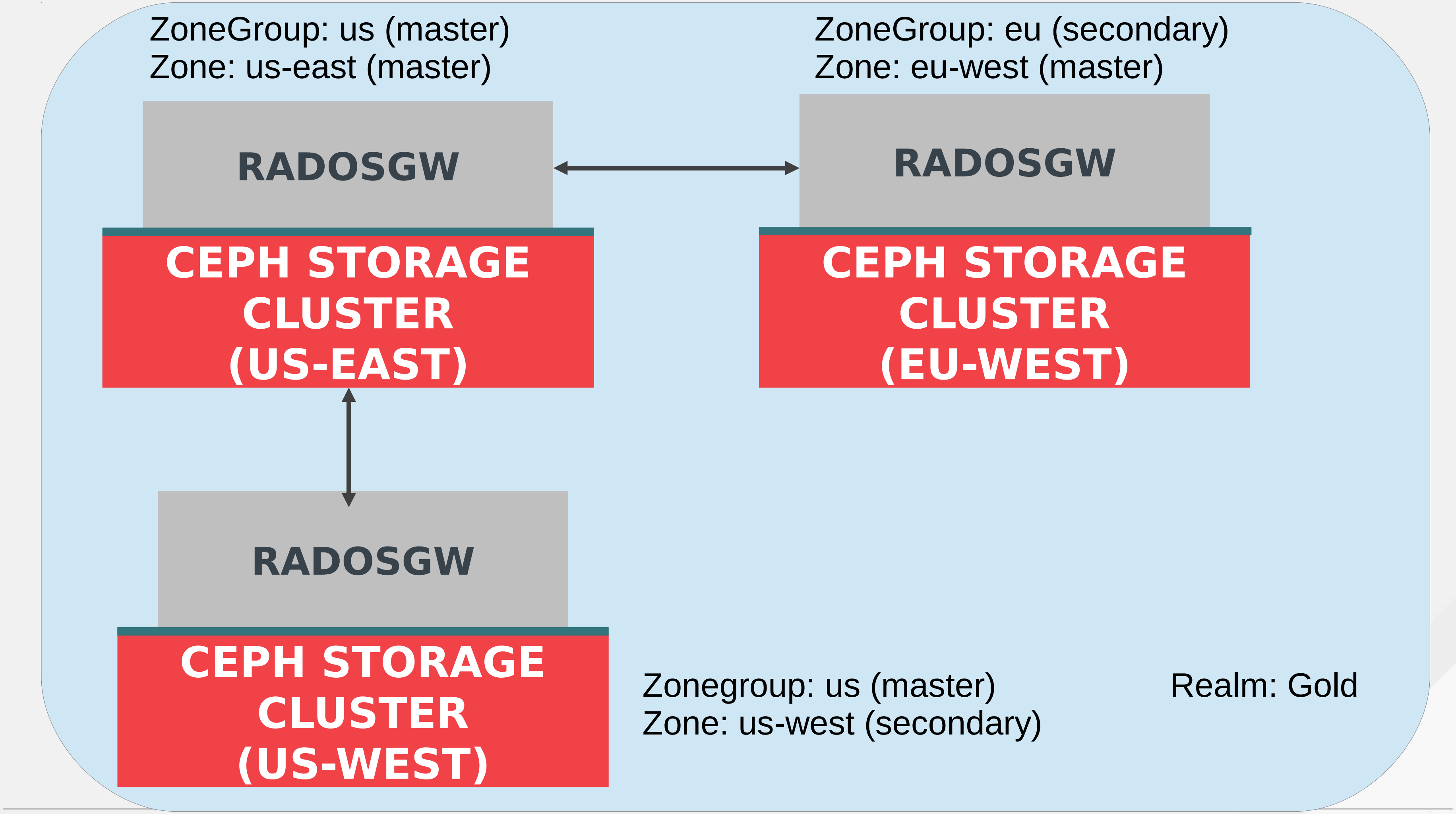
# New implementation

- part of the radosgw (written in c++)
- Active/active support for data replication
- Simpler configuration
- Simplify failover/failback
- Dynamic reconfiguration
- Backward compatibility with the sync agent

# Multisite configuration

- Realm
  - Namespace
  - contains the multisite configuration and status
  - Allows running different configurations in the same cluster
- Zonegroup
  - Group of zones
  - Used to be called region in old multisite
  - Each realm has a single master zonegroup
- Zone
  - One or more Radosgw instances all running on the same Rados cluster
  - Each zonegroup has a single master zone

# Multisite environment example



ZoneGroup: us (master)
Zone: us-east (master)

ZoneGroup: eu (secondary)
Zone: eu-west (master)

RADOSGW
CEPH STORAGE CLUSTER (US-EAST)
RADOSGW
CEPH STORAGE CLUSTER (EU-WEST)
RADOSGW
CEPH STORAGE CLUSTER (US-WEST)

Zonegroup: us (master)
Zone: us-west (secondary)

Realm: Gold

# Configuration change

- Period:
  - Each period has a unique id
  - Contains: realm configuration, an epoch and it's predecessor period id (except for the first period)
- Every realm has an associated current period and a chronological list of periods
- Git like mechanism:
  - User configuration changes are stored locally
  - Configuration updated are stored in a stagging period (using radosgw-admin period update command)
  - Changes are applied only when the period is commited (using radosgw-admin period commit command)
- Each zone can pull the period information (using radosgw-admin period pull command)

# Configuration change – new master zone

- Period commit will results in the following actions:
  - A new period is generated with a new period id and epoch of 1
  - Realm's current period is updated to point to the newly generated period id
  - Realm's epoch is incremented
  - New period is pushed to all other zones by the new master
- We use watch/notify on the realm rados object to detect changes and apply them on the local radosgw

# Configuration change

- Period commit will only increment the period epoch.
- The new period information will be pushed to all other zones
- We use watch/notify on the realm rados object to detect changes on the local radosgw

# Sync process

- Metadata changes:
  - Bucket ops (Create,  Delete and enable/disable versioning)
  - Users ops
- Metadata changes have wide system effect
- Metadata changes are rare
- Data changes: all objects updates
- Data changes are frequent

# Metadata sync

- Metadata changes are replicated synchronously across the realm
- Each realm has a single meta master, the master zone in the master zonegroup
- Only the meta master can executes metadata changes
- Separate log for metadata changes
- Each Ceph cluster has a local copy of the metadata log
- If the meta master is down the user cannot perform metadata updates till a new meta master is assigned

# Metadata sync

- updates to metadata originating from a different zone:
  - forwarded request to the meta master
  - update the metadata log
  - meta master perform the change
  - meta master pushes metadata updates to all the other zones
  - Each zone will pull the updated metadata log and apply changes locally
- All zones check periodically for metadata changes

# Data sync

- Data changes are handled locally and replicated asynchronously (eventual consistency)
- Default is Active/Active sync
- User can configure a zone to be read only for Active/Passive
- We first complete a full sync and than continue doing an incremental sync
- Each bucket instance within each zone has a unique incremented version id that is used to keep track of changes on that specific bucket.

# Data sync

- Data sync run periodically
- Init phase: fetch the list of all the bucket instances
- Sync Phase:
  - for each bucket
    - If bucket does not exist, fetch bucket and bucket instance metadata from meta master zone. Create new bucket
    - Sync bucket
    - Check to see if need to send updates to other zones
- Incremental sync keeps a bucket index position to continue from

# Sync status

- Each zone keeps the metadata sync state against the meta master
- Each zone keeps the data sync state where it is synced with regard to all its peers

# Sync status command

```
radosgw-admin sync status
```

```
realm f94ab897-4c8e-4654-a699-f72dfd4774df (gold)
     zonegroup 9bcecc3c-0334-4163-8fbb-5b8db0371b39 (us)
          zone 153a268f-dd61-4465-819c-e5b04ec4e701 (us-west)
  metadata sync syncing
                full sync: 0/64 shards
                metadata is caught up with master
                incremental sync: 64/64 shards
      data sync source: 018cad1e-ab7d-4553-acc4-de402cfddd19 (us-east)
                        syncing
                        full sync: 0/128 shards
                        incremental sync: 128/128 shards
                        data is caught up with source
```

# A little bit of the Implementation

- We use co-routines for asynchronous execution based on boost::asio::coroutine with our own stack class.
- See code here: https://github.com/ceph/ceph/blob/master/src/rgw/rgw_coroutine.h
- We use leases for locking

# What's next

# WHAT'S NEXT

- Log trimming – clean old logs
- Sync modules – framework that allows forwarding data (and metadata) to external tiers. This will allow external metadata search (via elasticsearch)

# THANK YOU!

Email: owasserm@redhat.com

IRC: owasserm OFTC #ceph,
#ceph-devel