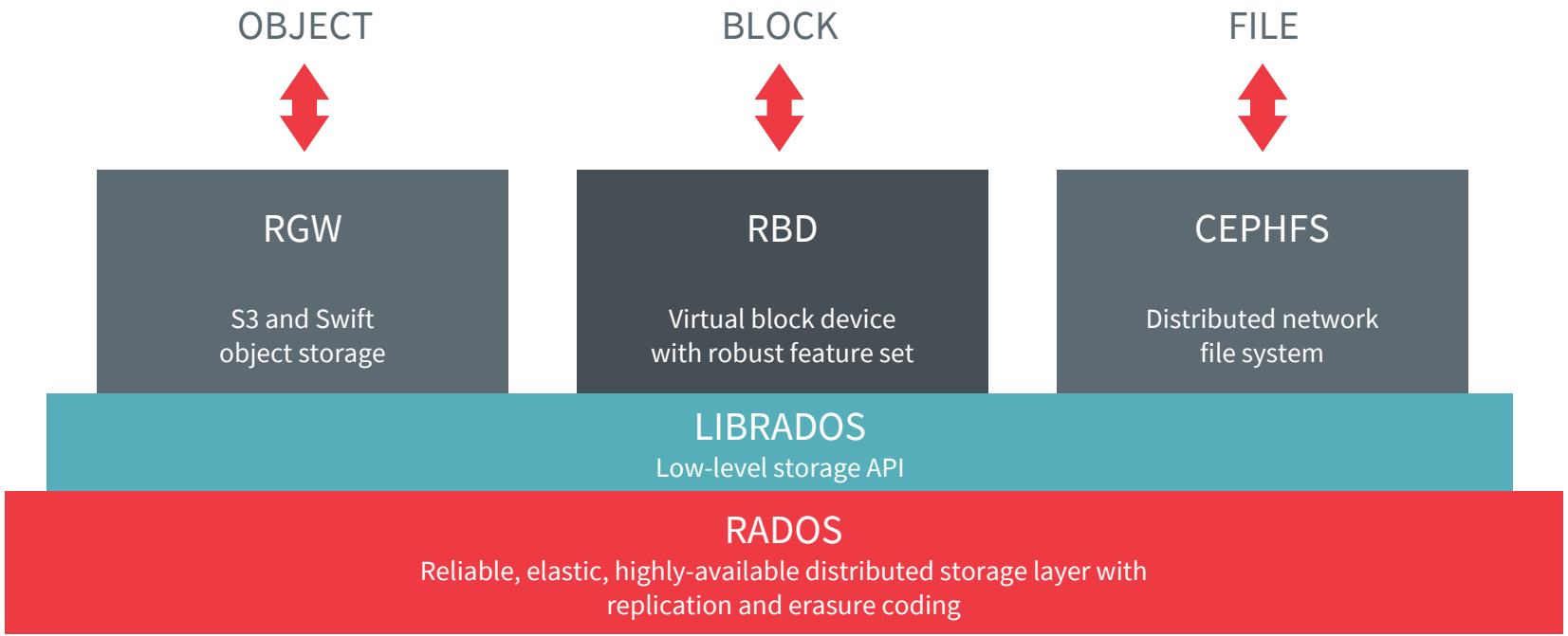
CEPH DATA SERVICES
IN A MULTI- AND HYBRID CLOUD WORLD

Sage Weil - Red Hat
FOSDEM - 2019.02.02

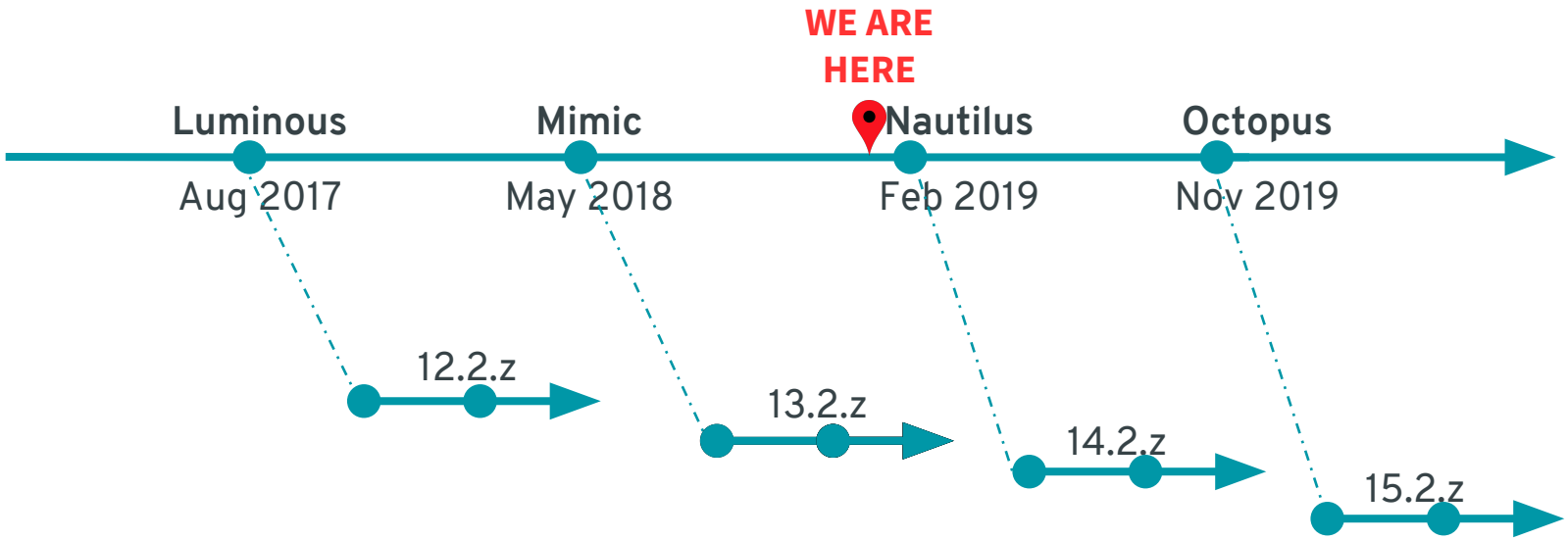
OUTLINE

- Ceph
- Data services
- Block
- File
- Object
- Edge
- Future

UNIFIED STORAGE PLATFORM



RELEASE SCHEDULE



- Stable, named release every 9 months
- Backports for 2 releases
- Upgrade up to 2 releases at a time
  - (e.g., Luminous → Nautilus, Mimic → Octopus)

FOUR CEPH PRIORITIES

Usability and management

Container ecosystem

Performance

Multi- and hybrid cloud

MOTIVATION - DATA SERVICES

A CLOUDY FUTURE

- IT organizations today
  - Multiple private data centers
  - Multiple public cloud services
- It's getting cloudier
  - "On premise" → private cloud
  - Self-service IT resources, provisioned on demand by developers and business units
- Next generation of cloud-native applications will span clouds
- "Stateless microservices" are great, but real applications have state
- Managing moving or replicated state is hard

“DATA SERVICES”

- Data placement and portability
  - Where should I store this data?
  - How can I move this data set to a new tier or new site?
  - Seamlessly, without interrupting applications?
- Introspection
  - What data am I storing? For whom? Where? For how long?
  - Search, metrics, insights
- Policy-driven data management
  - Lifecycle management
  - Compliance: constrain placement, retention, etc. (e.g., HIPAA, GDPR)
  - Optimize placement based on cost or performance
  - Automation

MORE THAN JUST DATA

- Data sets are tied to applications
  - When the data moves, the application often should (or must) move too
- Container platforms are key
  - Automated application (re)provisioning
  - “Operators” to manage *coordinated* migration of state *and* the applications that consume it

kubernetes

DATA USE SCENARIOS

- Multi-tier
  - Different storage for different data
- Mobility
  - Move an application and its data between sites with minimal (or no) availability interruption
  - Maybe an entire site, but usually a small piece of a site (e.g., a single app)
- Disaster recovery
  - Tolerate a complete site failure; reinstantiate data and app in a secondary site quickly
  - Point-in-time consistency with bounded latency (bounded data loss on failover)
- Stretch
  - Tolerate site outage without compromising data availability
  - Synchronous replication (no data loss) or async replication (different consistency model)
- Edge
  - Small satellite (e.g., telco POP) and/or semi-connected sites (e.g., autonomous vehicle)

SYNC VS ASYNC

Synchronous replication

- Applications initiates a write
- Storage writes to all replicas
- Application write completes

- Write latency may be high since we wait for all replicas
- All replicas always reflect applications' completed writes

Asynchronous replication

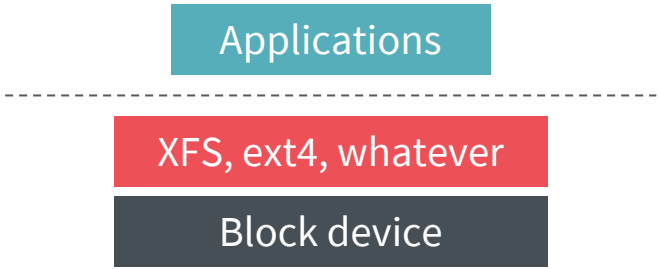
- Application initiates a write
- Storage writes to one (or some) replicas
- Application write completes
- Storage writes to remaining (usually remote) replicas later

- Write latency can be kept low
- If initial replicas are lost, application write may be lost
- Remote replicas may always be somewhat stale

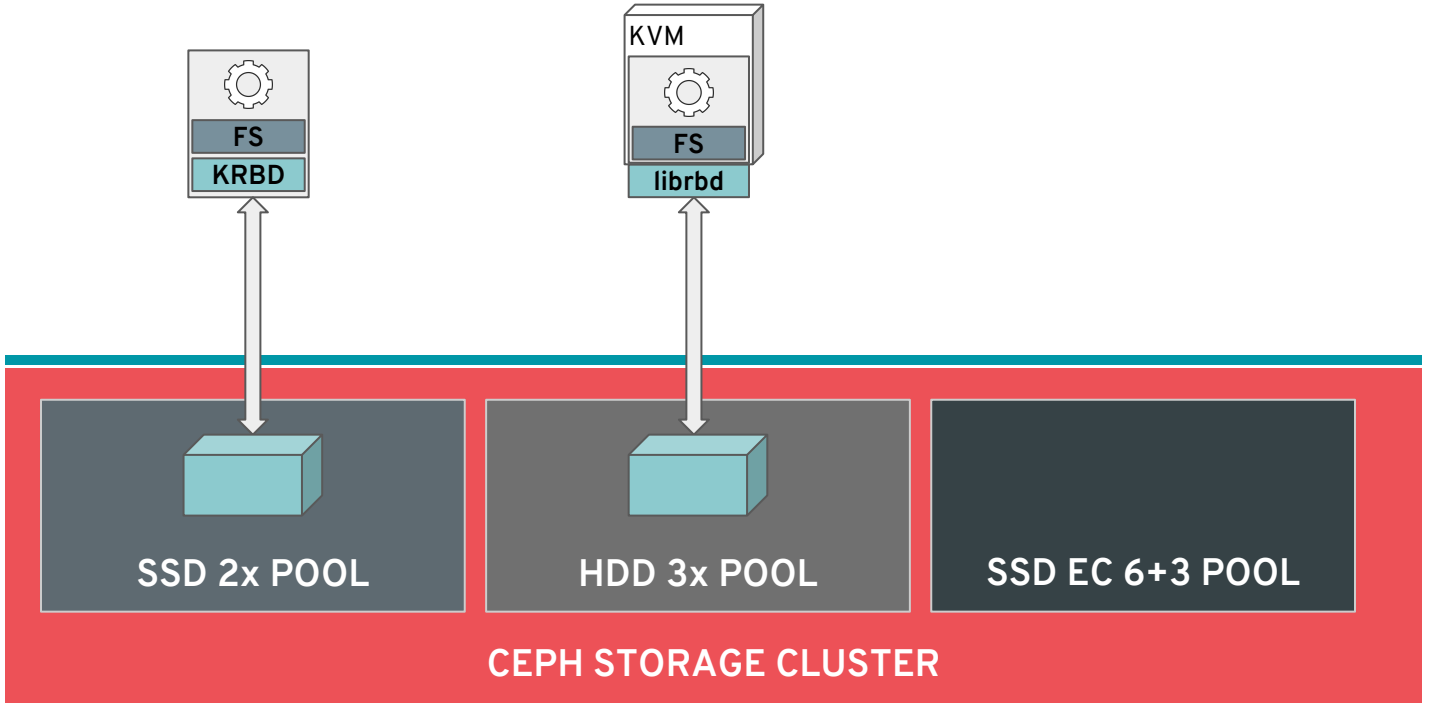
BLOCK STORAGE

HOW WE USE BLOCK

- Virtual disk device
- Exclusive access by nature (with few exceptions)
- Strong consistency required
- Performance sensitive
- Basic feature set
  - Read, write, flush, maybe resize
  - Snapshots (read-only) or clones (read/write)
    - Point-in-time consistent
- Often self-service provisioning
  - via Cinder in OpenStack
  - via Persistent Volume (PV) abstraction in Kubernetes



RBD - TIERING WITH RADOS POOLS

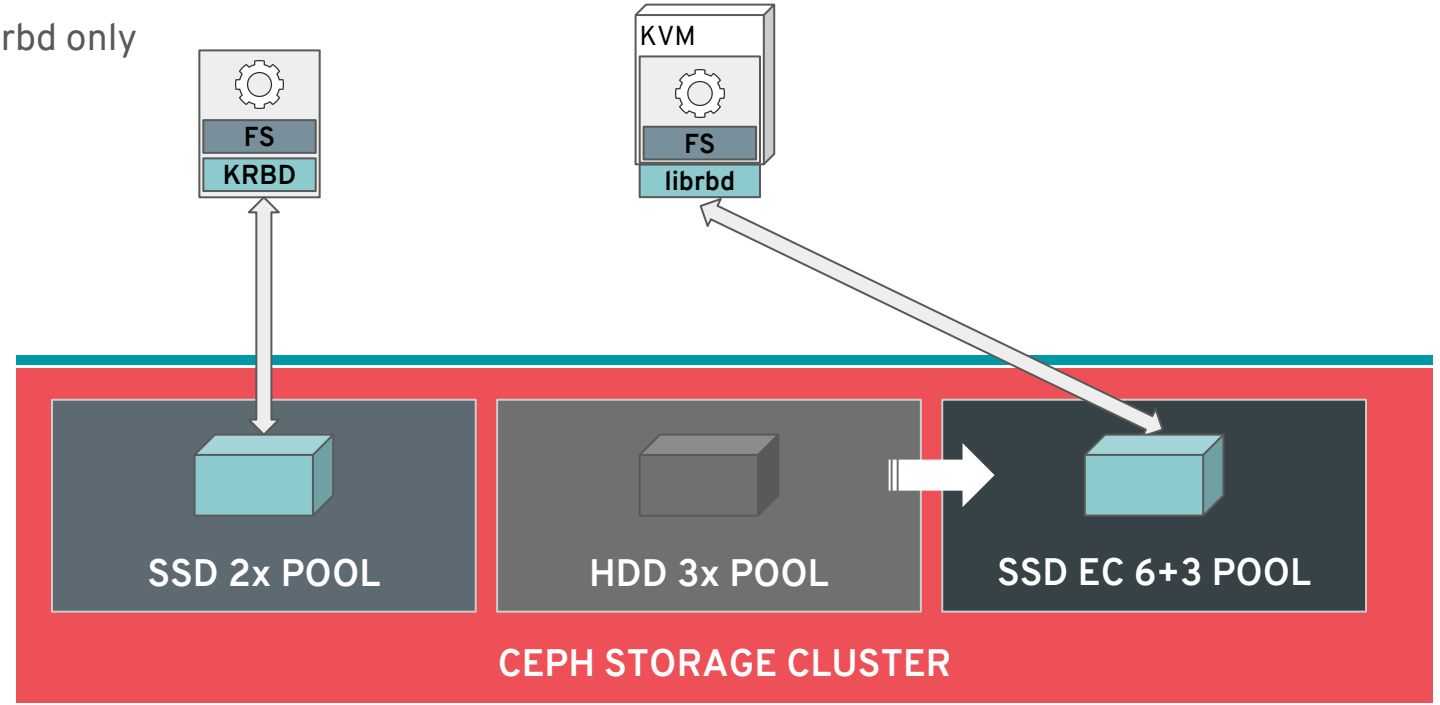


- ✓ Multi-tier
- ❑ Mobility
- ❑ DR
- ❑ Stretch
- ❑ Edge

RBD - LIVE IMAGE MIGRATION

- New in Nautilus
- librbd only

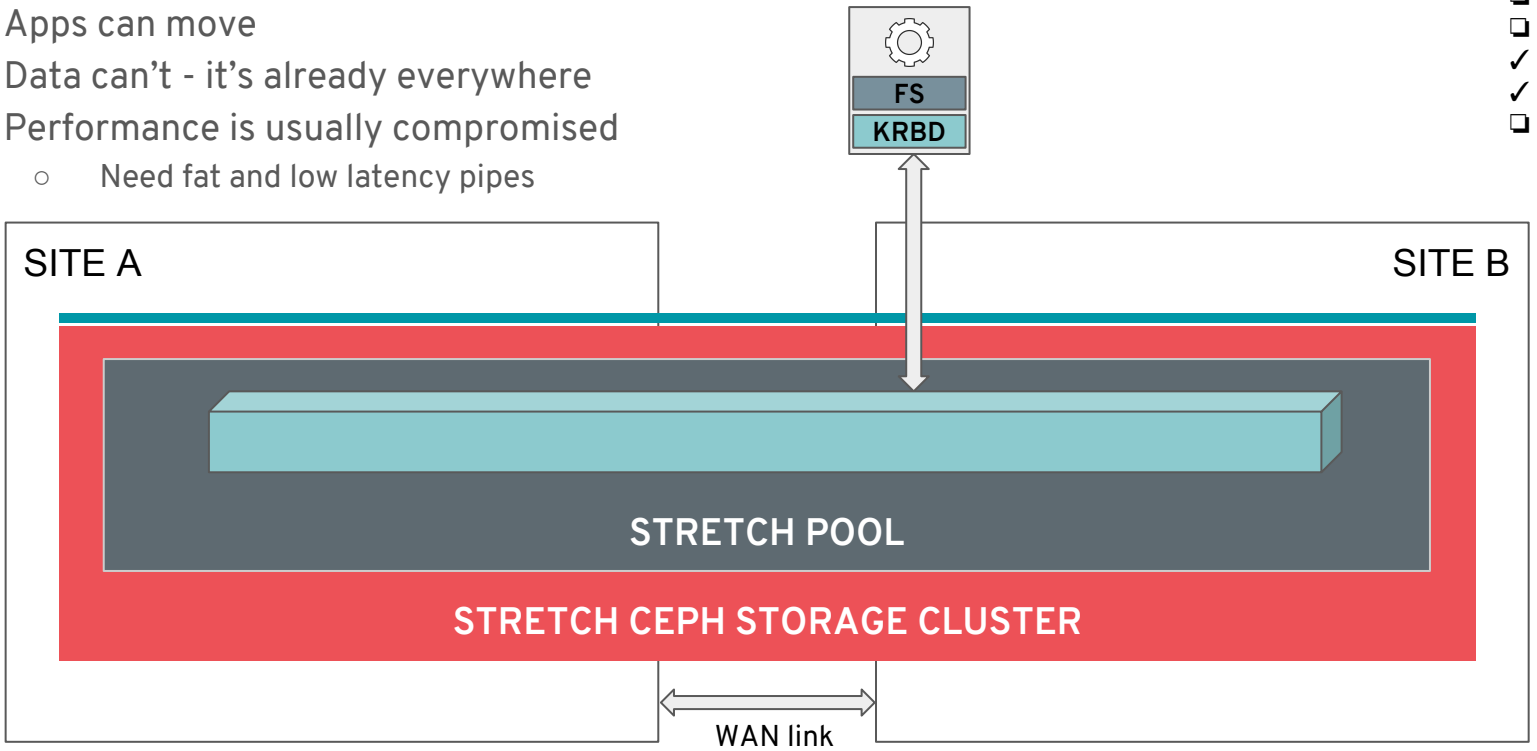
✓ Multi-tier
✓ Mobility
❑ DR
❑ Stretch
❑ Edge



RBD - STRETCH

- Apps can move
- Data can't - it's already everywhere
- Performance is usually compromised
  - Need fat and low latency pipes

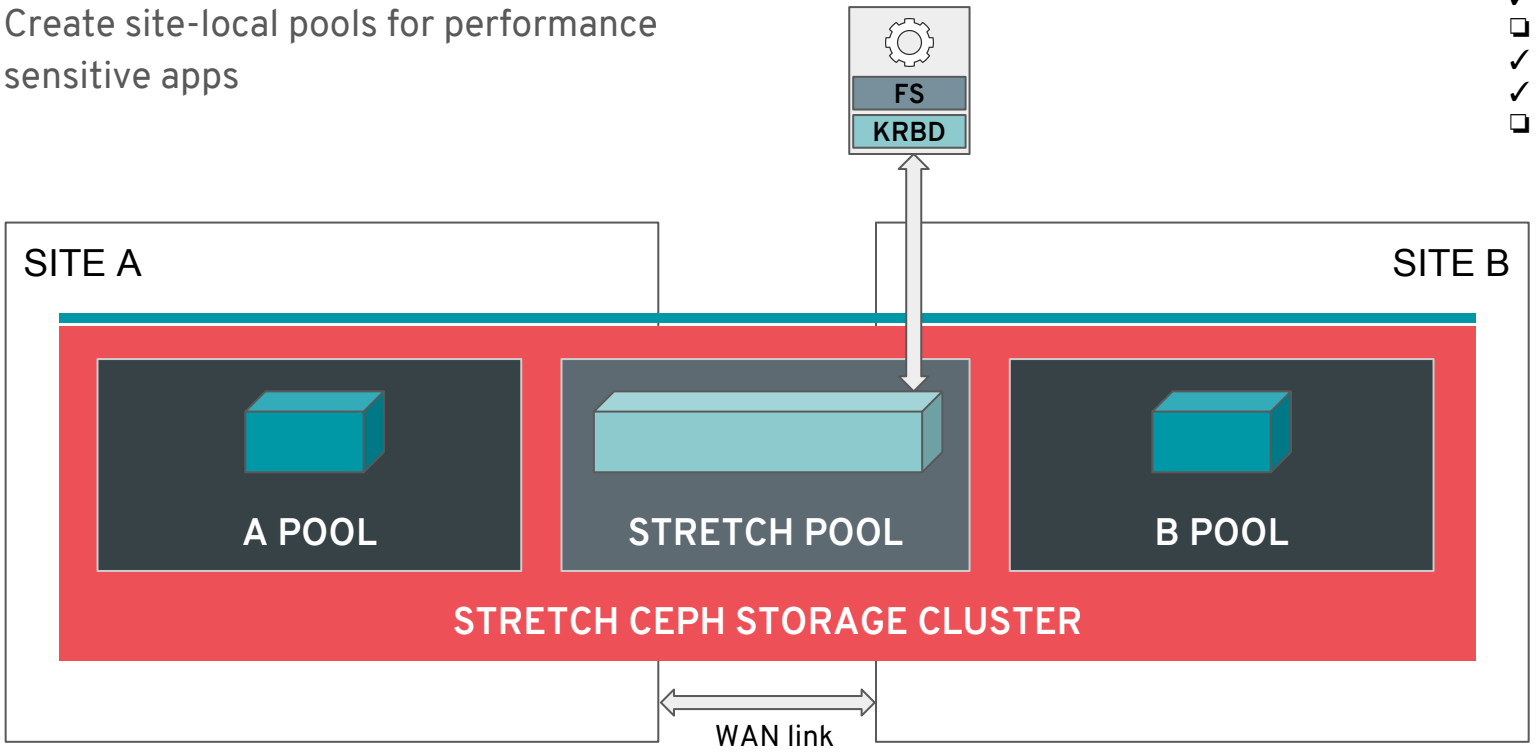
❑ Multi-tier
❑ Mobility
✓ DR
✓ Stretch
❑ Edge



RBD - STRETCH WITH TIERS

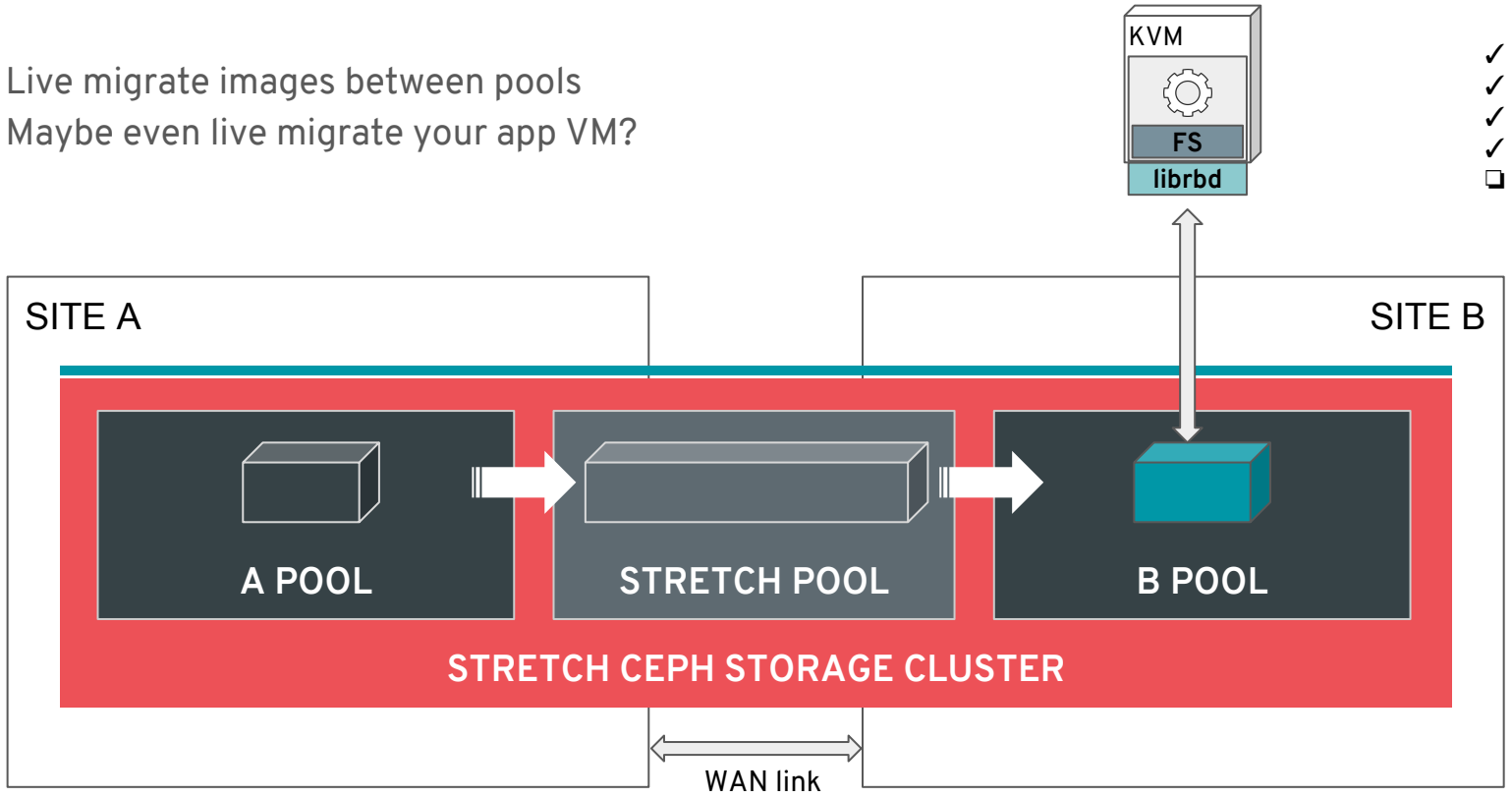
- Create site-local pools for performance sensitive apps

✓ Multi-tier
❑ Mobility
✓ DR
✓ Stretch
❑ Edge



RBD - STRETCH WITH MIGRATION

- Live migrate images between pools
- Maybe even live migrate your app VM?

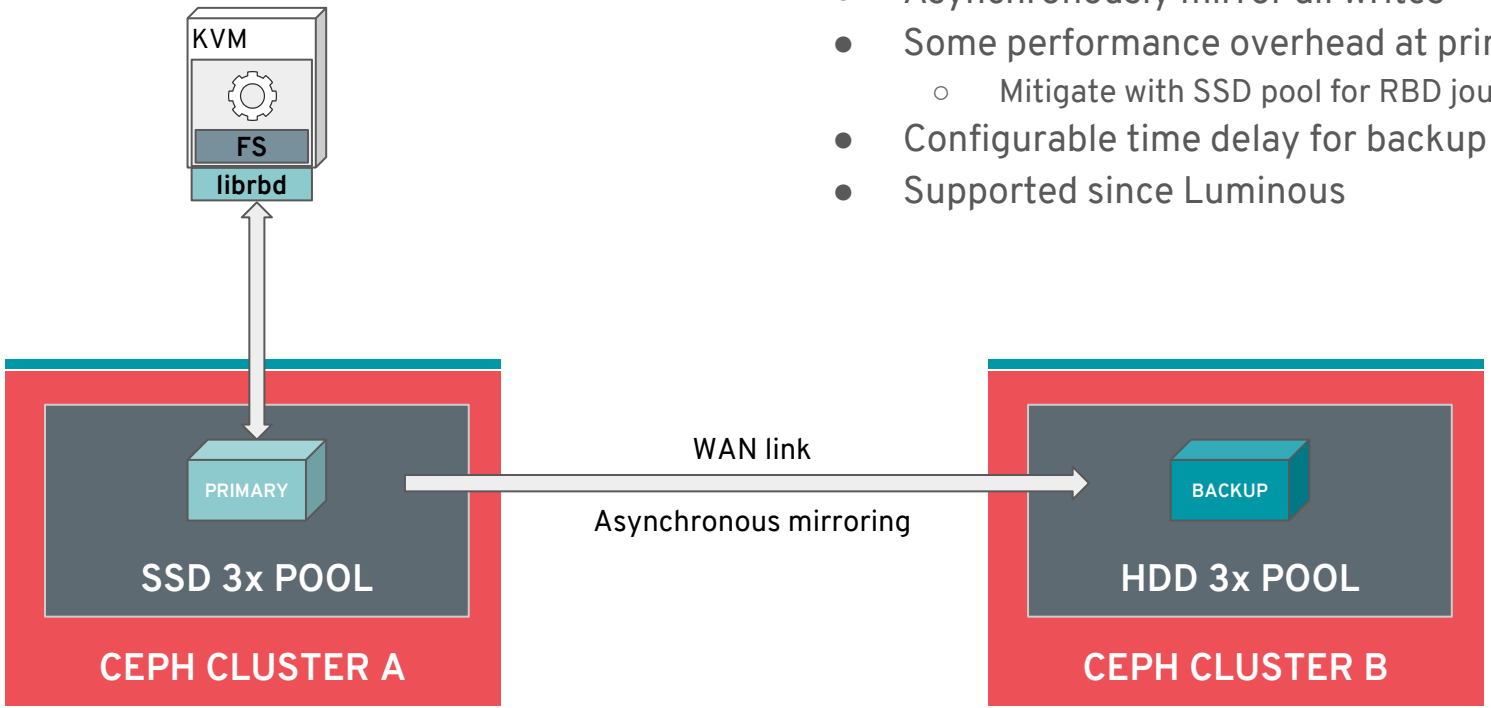


✓ Multi-tier
✓ Mobility
✓ DR
✓ Stretch
❑ Edge

STRETCH IS SKETCH

- Network latency is critical
  - Want low latency for performance
  - Stretch requires nearby sites, limiting usefulness
- Bandwidth too
  - Must be able to sustain rebuild data rates
- Relatively inflexible
  - Single cluster spans all locations; maybe ok for 2 datacenters but not 10?
  - Cannot “join” existing clusters
- High level of coupling
  - Single (software) failure domain for all sites
- Proceed with caution!

RBD ASYNC MIRRORING

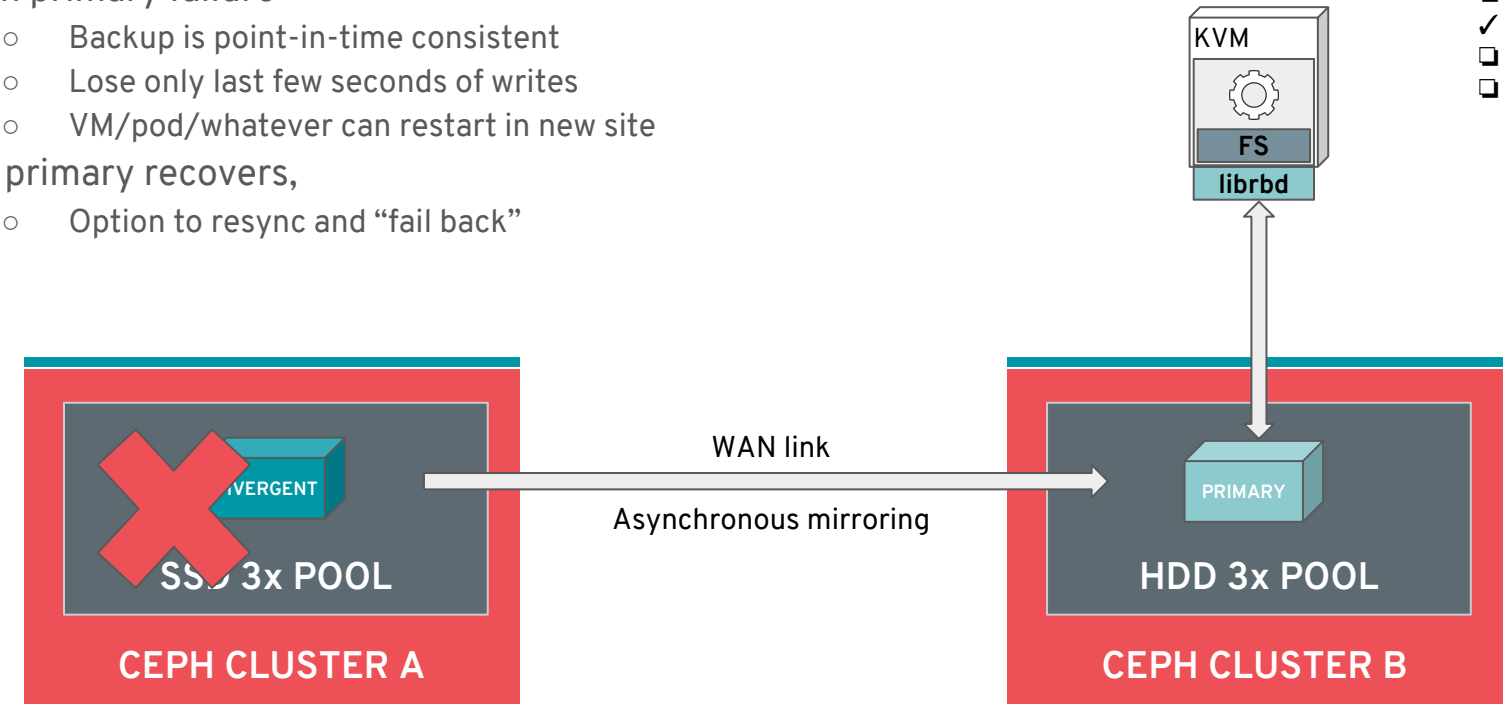


- Asynchronously mirror all writes
- Some performance overhead at primary
  - Mitigate with SSD pool for RBD journal
- Configurable time delay for backup
- Supported since Luminous

RBD ASYNC MIRRORING

- On primary failure
  - Backup is point-in-time consistent
  - Lose only last few seconds of writes
  - VM/pod/whatever can restart in new site
- If primary recovers,
  - Option to resync and “fail back”

- ❑ Multi-tier
- ❑ Mobility
- ✓ DR
- ❑ Stretch
- ❑ Edge



RBD MIRRORING IN OPENSTACK CINDER

- Ocata
  - Cinder RBD replication driver
- Queens
  - ceph-ansible deployment of rbd-mirror via TripleO
- Rocky
  - Failover and fail-back operations

- Gaps
  - Deployment and configuration tooling
  - Cannot replicate multi-attach volumes
  - Nova attachments are lost on failover

MISSING LINK: APPLICATION ORCHESTRATION

- Hard for IaaS layer to reprovision app in new site
- Storage layer can't solve it on its own either
- Need automated, declarative, structured specification for entire app stack...

kubernetes

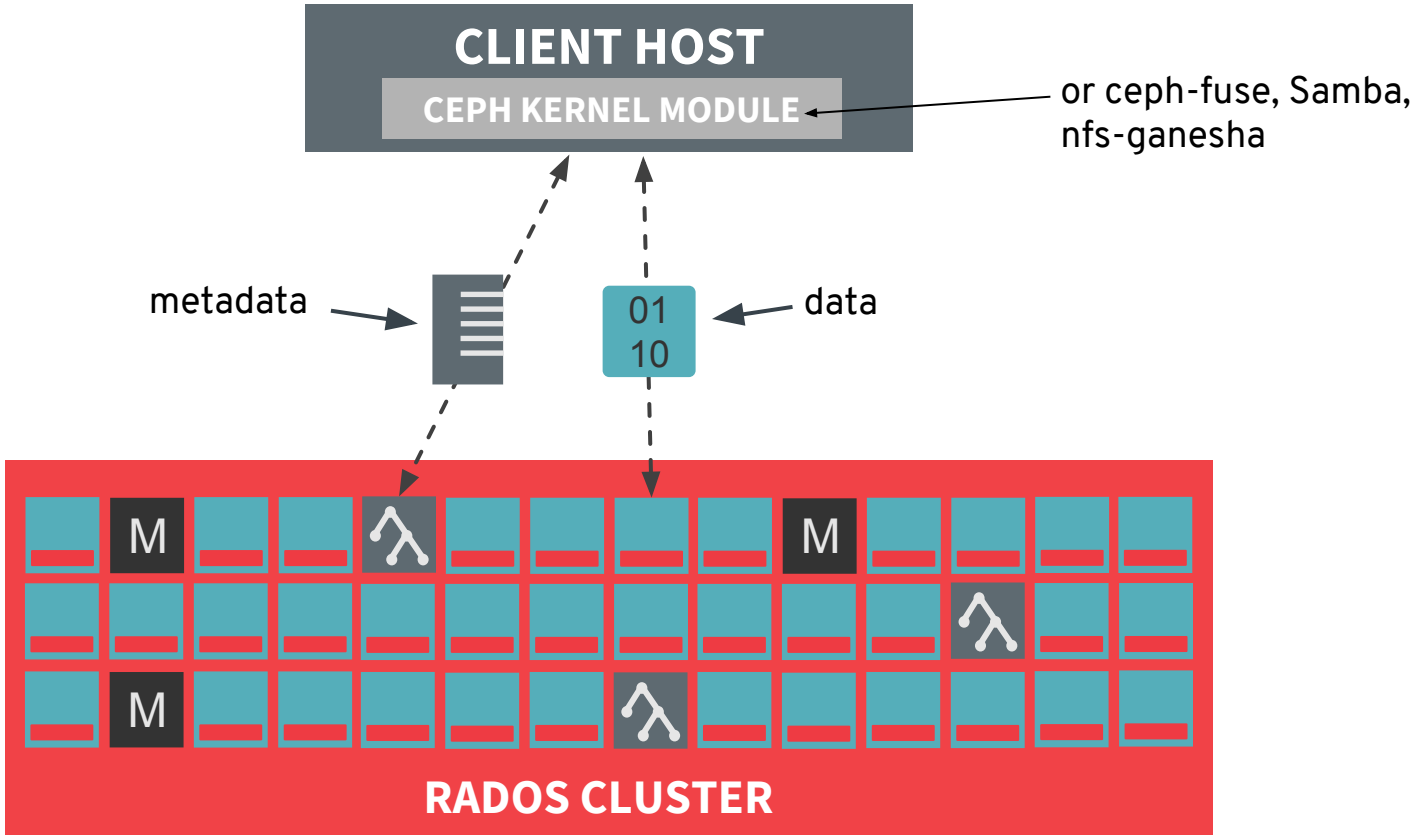
FILE STORAGE

CEPHFS STATUS

- Stable since Kraken
- Multi-MDS stable since Luminous
- Snapshots stable since Mimic
- Support for multiple RADOS data pools
  - Per-directory subtree policies for placement, striping, etc.
- Fast, highly scalable
- Quota, multi-volumes, multi-subvolume
- Provisioning via OpenStack Manila and Kubernetes
- Fully awesome

✓ Multi-tier
❑ Mobility
❑ DR
❑ Stretch
❑ Edge

CEPHFS


CLIENT HOST
CEPH KERNEL MODULE
or ceph-fuse, Samba,
nfs-ganesha
metadata
01
10
data
M
M
M
RADOS CLUSTER

CEPHFS - STRETCH?

- We can stretch CephFS just like RBD pools
- It has the same limitations as RBD
  - Latency → lower performance
  - Limited by geography
  - Big (software) failure domain
- Also,
  - MDS latency is critical for file workloads
  - ceph-mds daemons will run in one site; clients in other sites will see higher latency

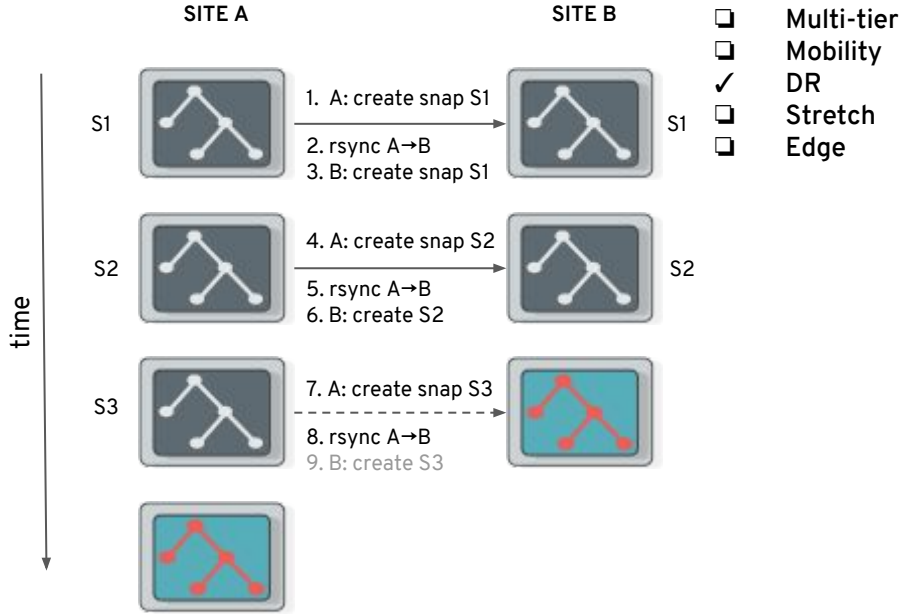
❑ Multi-tier
❑ Mobility
✓ DR
✓ Stretch
❑ Edge

CEPHFS - FUTURE OPTIONS

- What can we do with CephFS across sites and clusters?

CEPHFS - SNAP MIRRORING?

- CephFS snapshots provide
  - point-in-time consistency
  - granularity (any directory in the system)
- CephFS rstats provide
  - rctime = recursive ctime on any directory
  - We can efficiently find changes
- rsync provides
  - efficient file transfer
- Time bounds on order of minutes
- Gaps and TODO
  - “rstat flush” coming in Nautilus
    - Xuehan Xu @ Qihoo 360
  - rsync support for CephFS rctime
  - scripting / tooling
  - easy rollback interface
- Matches enterprise storage feature sets



DO WE NEED POINT-IN-TIME FOR FILE?

- Yes.
- Sometimes.
- Some geo-replication DR features are built on rsync...
  - Consistent view of individual files (maybe?),
  - Lack point-in-time consistency between files
- Some (many? most?) apps are not picky about cross-file consistency...
  - Content stores
  - Casual usage without cross-site modification of the same files

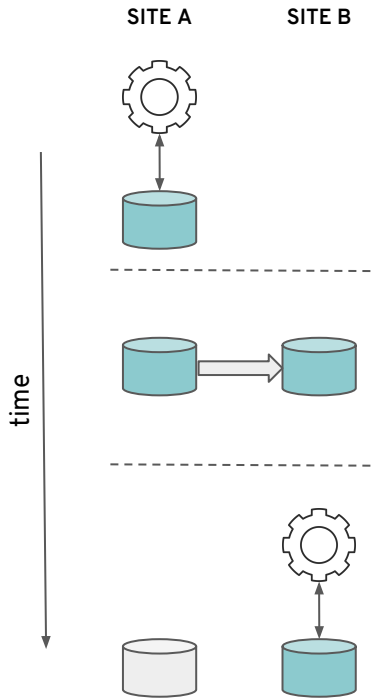
CEPHFS - UPDATE LOG ASYNC SYNC?

- ❑ Multi-tier
- ❑ Mobility
- ✓ DR
- ❑ Stretch
- ❑ Edge

- Idea
  - Each ceph-mds daemon generates an update log
  - Replication worker daemons replicate updates asynchronously
- Benefits
  - Generally timely replication of updates
  - Should scale reasonably well (e.g., if we allow N workers per MDS)
- Limitations
  - No point-in-time consistency
- Challenges
  - Semantics of namespace operations (e.g., directory rename) may be tricky when workers are not in sync

ABOUT MIGRATION...

MIGRATION: STOP, MOVE, START

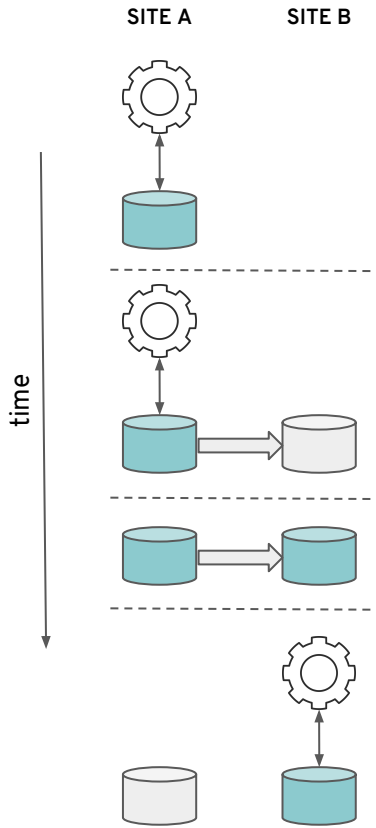


- App runs in site A
- Stop app in site A
- Copy data A→B
- Start app in site B

- App maintains exclusive access
- Long service disruption

❑ Multi-tier
✓ Mobility
❑ DR
❑ Stretch
❑ Edge

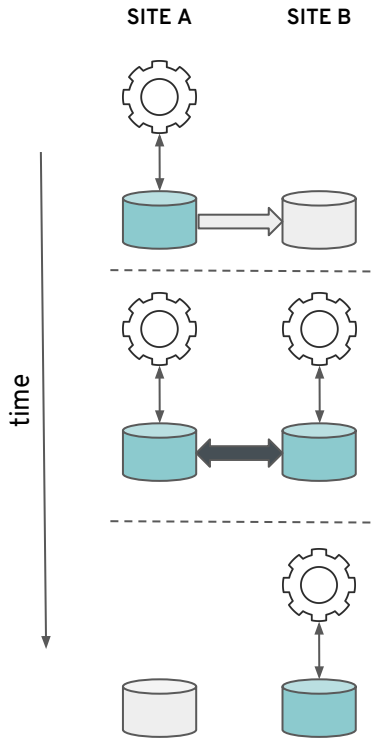
MIGRATION: PRESTAGING



- App runs in site A
- Copy most data from A→B
- Stop app in site A
- Copy last little bit A→B
- Start app in site B

- App maintains exclusive access
- Short availability blip

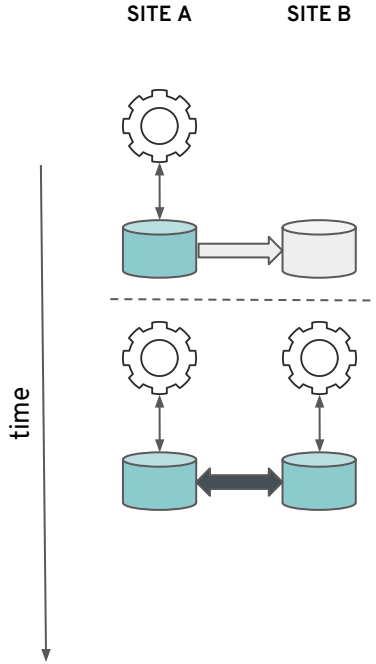
MIGRATION: TEMPORARY ACTIVE/ACTIVE



- App runs in site A
- Copy most data from A→B
- Enable bidirectional replication
- Start app in site B
- Stop app in site A
- Disable replication

- No loss of availability
- Concurrent access to same data
- Performance degradation only during active/active period

ACTIVE/ACTIVE



- App runs in site A
- Copy most data from A→B
- Enable bidirectional replication
- Start app in site B

- Highly available across two sites
- Concurrent access to same data
  - Consistency model?
  - Sync or async?

CEPHFS - BIDIRECTIONAL FILE REPLICATION?

- We don't have general-purpose bidirectional file replication
- It is hard to resolve conflicts for any POSIX operation
  - Sites A and B both modify the same file
  - Site A renames /a → /b/a while Site B: renames /b → /a/b
- But applications can only go active/active if they are cooperative
  - i.e., they carefully avoid such conflicts
  - e.g., mostly-static directory structure + last writer wins
- So we could do it if we simplify the data model...
- But wait, that sounds a bit like object storage...

OBJECT STORAGE

WHY IS OBJECT SO GREAT?

- Based on HTTP
  - Interoperates well with web caches, proxies, CDNs, ...
- Atomic object replacement
  - PUT on a large object atomically replaces prior version
  - Trivial conflict resolution (last writer wins)
  - Lack of overwrites makes erasure coding easy
- Flat namespace
  - No multi-step traversal to find your data
  - Easy to scale horizontally
- No rename
  - Vastly simplified implementation

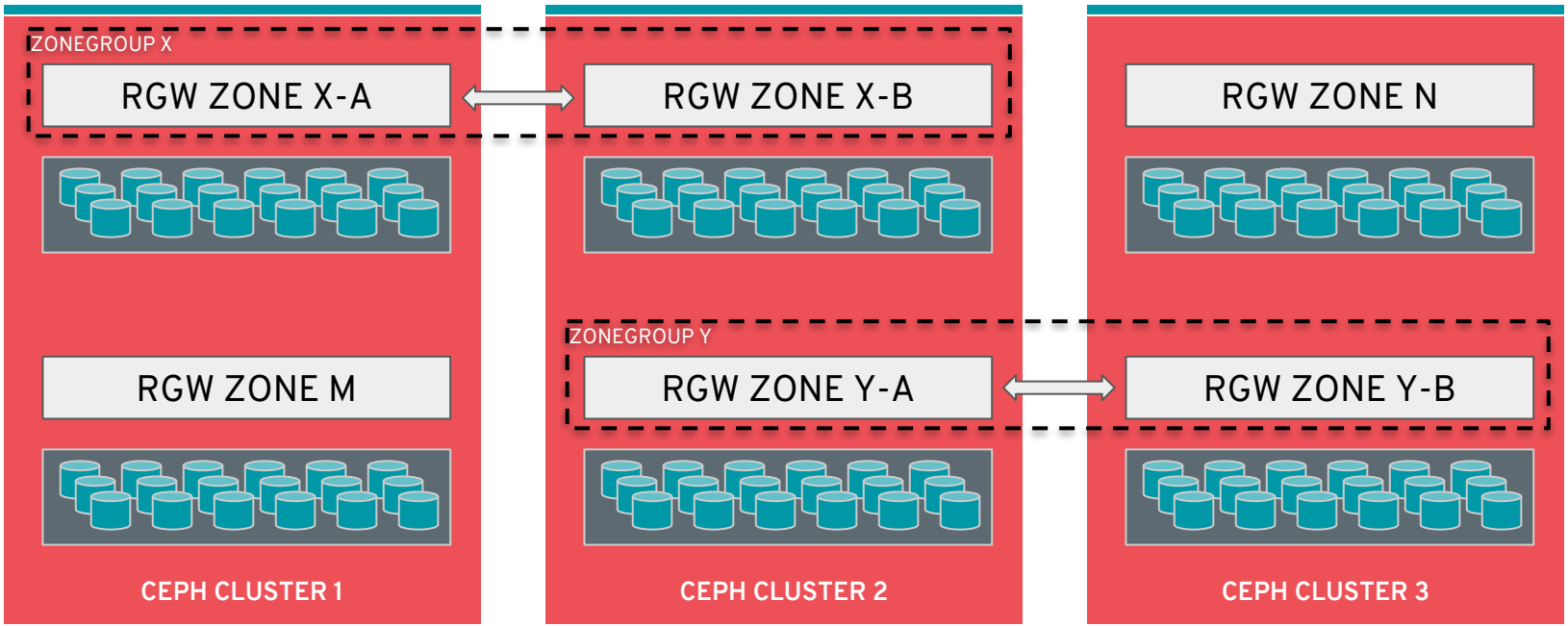
THE FUTURE IS... OBJECTY

- File is not going away, and will be critical
  - Half a century of legacy applications
  - It's genuinely *useful*
- Block is not going away, and is also critical infrastructure
  - Well suited for exclusive-access storage users (boot devices, etc)
  - Performs better than file due to local consistency management, ordering etc.
- **Most new data will land in objects**
  - Cat pictures, surveillance video, vehicle telemetry, medical imaging, genome data...
  - Next generation of cloud native applications will be architected around object

RGW FEDERATION MODEL TODAY

- Zone
  - Collection of RADOS pools in one Ceph cluster
  - Set of RGW daemons serving that content
  - Can have many RGW zones per Ceph cluster
- ZoneGroup
  - Collection of 2+ Zones with a replication relationship
  - Active/Passive or Active/Active
- Namespace
  - Independent naming for users and buckets
  - All Zones replicate user and bucket metadata pool
  - One Zone per Namespace serves as the leader to handle User and Bucket creations/deletions
- Failover is driven externally
  - Human (or other?) operators decide when to write off an unreachable master zone, resynchronize, etc.

RGW FEDERATION TODAY



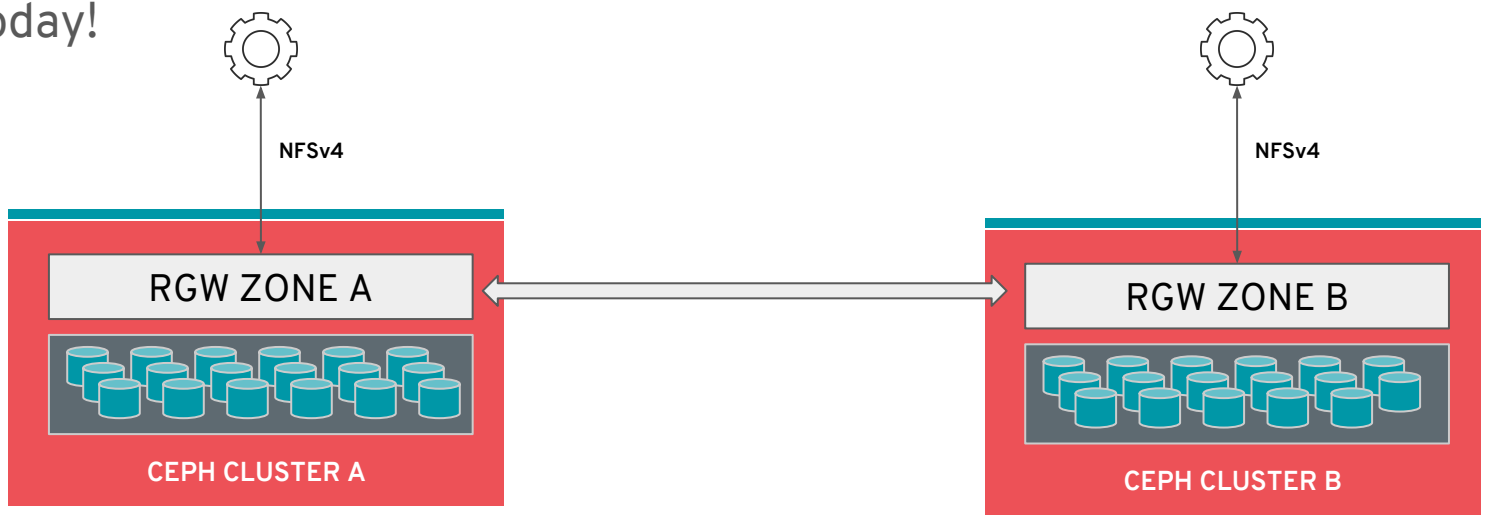
- ❏ Multi-tier
- ❏ Mobility
- ✓ DR
- ✓ Stretch
- ❏ Edge

- Gap: granular, per-bucket management of replication

ACTIVE/ACTIVE FILE ON OBJECT

- ❏ Multi-tier
- ❏ Mobility
- ✓ DR
- ✓ Stretch
- ❏ Edge

- Data in replicated object zones
  - Eventually consistent, last writer wins
- Applications access RGW via NFSv4
- Today!

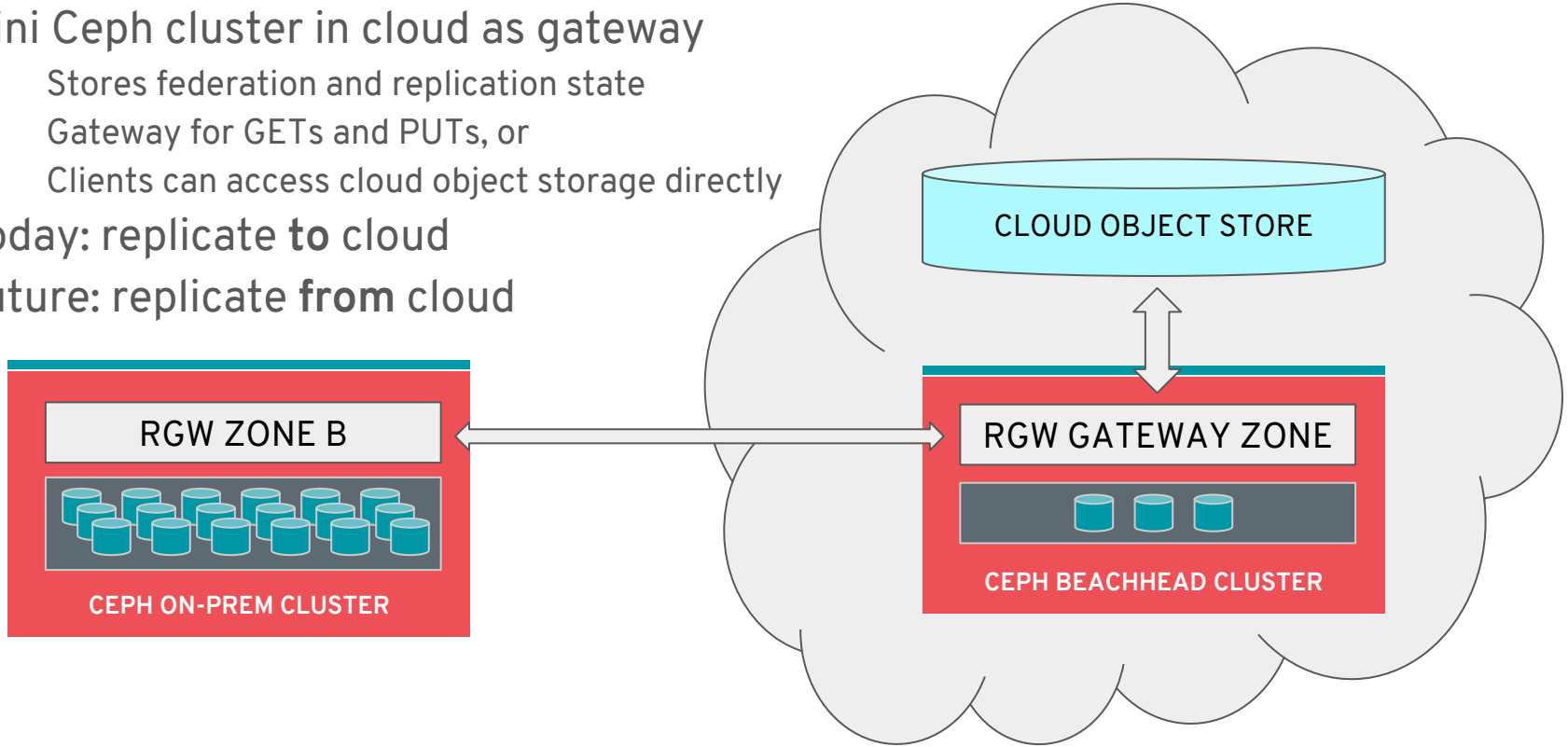


OTHER RGW REPLICATION PLUGINS

- ElasticSearch (Luminous)
  - Index entire zone by object or user metadata
  - Query API
- Cloud sync (Mimic)
  - Replicate entire zone or specific buckets to external object store (e.g., S3)
  - Can remap RGW buckets into individual S3 buckets, or same S3 bucket
  - Remaps ACLs, etc
- Archive (Nautilus)
  - Replicate all writes in one zone to another zone, preserving all versions
- Pub/Sub (Nautilus)
  - Subscribe to event notifications for actions like PUT
  - Integrates with knative serverless! (See Huamin's talk from Kubecon Seattle)

PUBLIC CLOUD STORAGE IN THE MESH

- Mini Ceph cluster in cloud as gateway
  - Stores federation and replication state
  - Gateway for GETs and PUTs, or
  - Clients can access cloud object storage directly
- Today: replicate **to** cloud
- Future: replicate **from** cloud



RGW TIERING

✓ Multi-tier
❑ Mobility
❑ DR
❑ Stretch
❑ Edge

Today: Intra-cluster

- Many RADOS pools for a single RGW zone
- Primary RADOS pool for object “heads”
  - Single (fast) pool to find object metadata and location of the tail of the object
- Each tail can go in a different pool
  - Specify bucket policy with PUT
  - Per-bucket policy as default when not specified
- Policy
  - Retention (auto-expire)

Nautilus

- Lifecycle policy
  - Automated tiering between RADOS pools based on age, ...

Future

- Tier objects to an external store
  - Initially something like S3
  - Later: tape backup, other backends...
- Encrypt data in external tier
- Compression
- (Maybe) cryptographically shard across multiple backend tiers

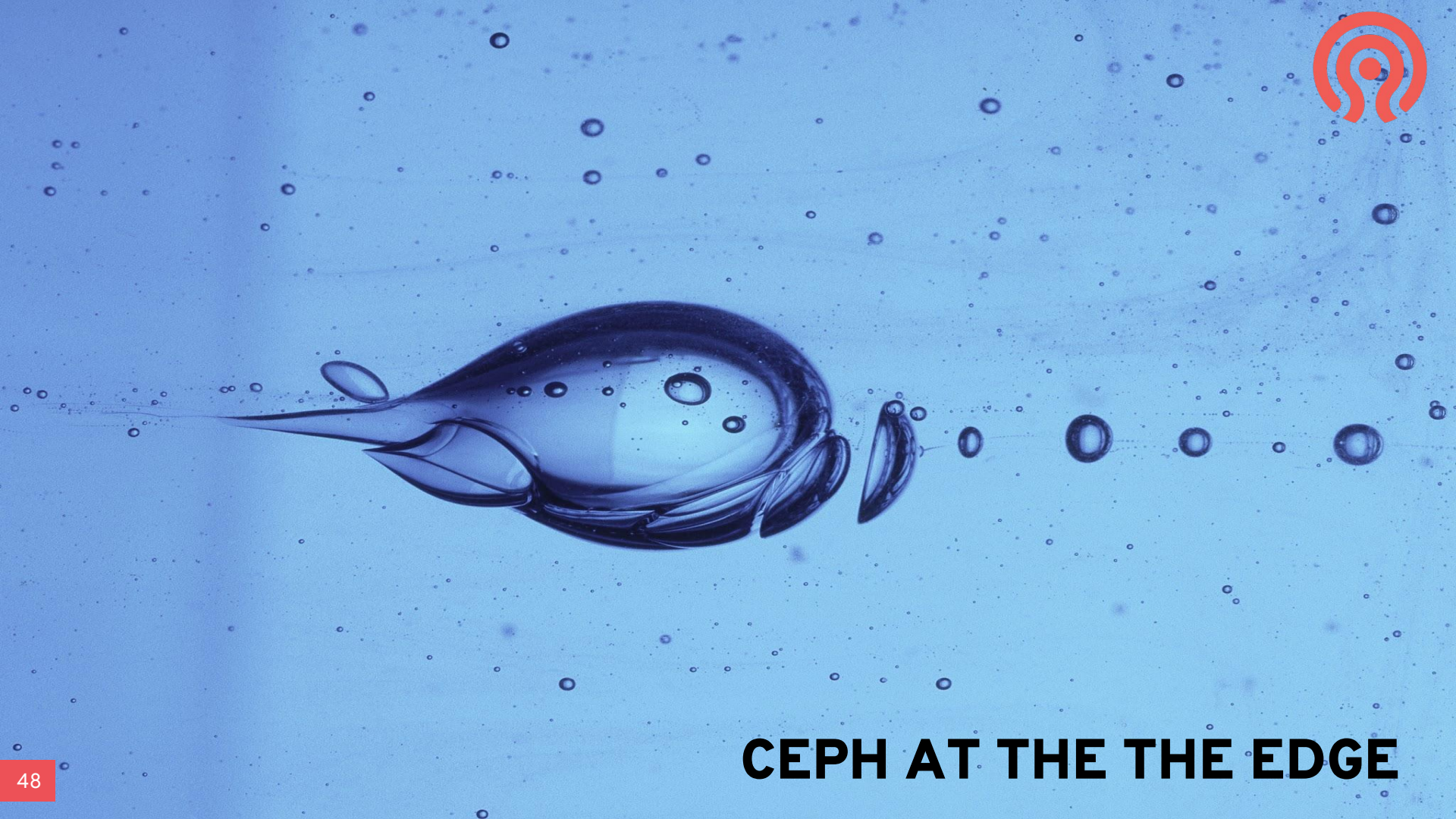
RGW - THE BIG PICTURE

Today

- RGW as gateway to a RADOS cluster
  - With some nifty geo-replication features
- RGW redirects clients to the correct zone
  - via HTTP Location: redirect
  - Dynamic DNS can provide right zone IPs
- RGW replicates at zone granularity
  - Well suited for disaster recovery

Future

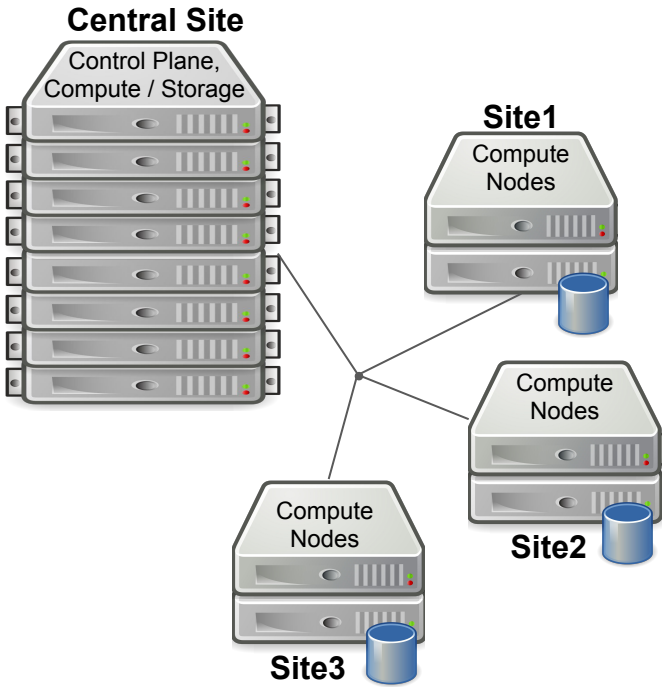
- RGW as a gateway to a mesh of sites
  - With great on-site performance
- RGW may redirect or **proxy** to right zone
  - Single point of access for application
  - Proxying enables coherent local caching
- RGW may replicate at bucket granularity
  - Individual applications set durability needs
  - Enable granular application mobility



CEPH AT THE THE EDGE

CEPH AT THE EDGE

- A few edge examples
  - Telco POPs: ¼ - ½ rack of OpenStack
  - Autonomous vehicles: cars or drones
  - Retail
  - Backpack infrastructure
- Scale down cluster size
  - Hyperconverge storage and compute
  - Nautilus: brings better memory control
- Multi-architecture support
  - aarch64 (ARM) builds upstream
  - POWER builds at OSU / OSL
- Hands-off operation
  - Operator-based provisioning (Rook)
  - Ongoing usability work
- Possibly unreliable WAN links



DATA AT THE EDGE

❑ Multi-tier
❑ Mobility
❑ DR
❑ Stretch
✓ Edge

- Block: async mirror edge volumes to central site
  - For DR purposes
- Data producers
  - Write generated data into objects in local RGW zone
  - Upload to central site when connectivity allows
  - Perhaps with some local pre-processing first
- Data consumers
  - Access to global data set via RGW (as a “mesh gateway”)
  - Local caching of a subset of the data
- We’re most interested in object-based edge scenarios

KUBERNETES

WHY ALL THE KUBERNETES TALK?

kubernetes

- True mobility is a partnership between orchestrator and storage
- Kubernetes is emerging leader in application orchestration
- Persistent Volumes
  - Basic Ceph drivers in Kubernetes, ceph-csi on the way
  - Rook for automating Ceph cluster deployment and operation
- Object
  - Trivial provisioning of Ceph via Rook
  - Coming soon: on-demand, dynamic provisioning of Object Buckets and Users (via Rook)
  - Consistent developer experience across different object backends (RGW, S3, minio, etc.)

BRINGING IT ALL TOGETHER…

SUMMARY

- Data services: mobility, introspection, policy
- Need a partnership between storage layer and application orchestrator
- Ceph already has several key multi-cluster capabilities…
  - Block mirroring
  - Object federation, replication, cloud sync, pub/sub; cloud tiering coming
  - Introspection (elasticsearch) and policy for object
- …and gaps
  - Object multi-site leveraging external clouds, granular management
  - Multi-site file mirroring
  - Orchestration of multi-site capabilities via Kubernetes

Block
Object
File

KEY EFFORTS

- Defining Kubernetes-based multi-cluster use-cases
  - RWO (block) PV DR, migration
  - RWX (file) PV DR, migration, active/active (CephFS or RGW-backed)
  - Dynamic bucket provisioning
  - Bucket policy, placement
- Extending RGW object capabilities
  - Bucket-granularity policy for multisite replication
  - Leveraging external cloud object stores with “thin” RGW zones
- Planning/designing CephFS multi-cluster modes
  - Snapshot-based mirroring (DR)
  - Loosely consistent mirroring (DR)
  - Multi-directional async mirroring (Mobility and Stretch)

BOTTOM LINE

Traditional view

Emerging view

Unified storage system

Multi-cloud data services platform

- Object, block, file
- Software Defined Storage
- Hardware agnostic

- Multi-cluster federation
- Sync and async replication
- Policy driven management

THANK YOU

http://ceph.io/
sage@redhat.com
@liewegas