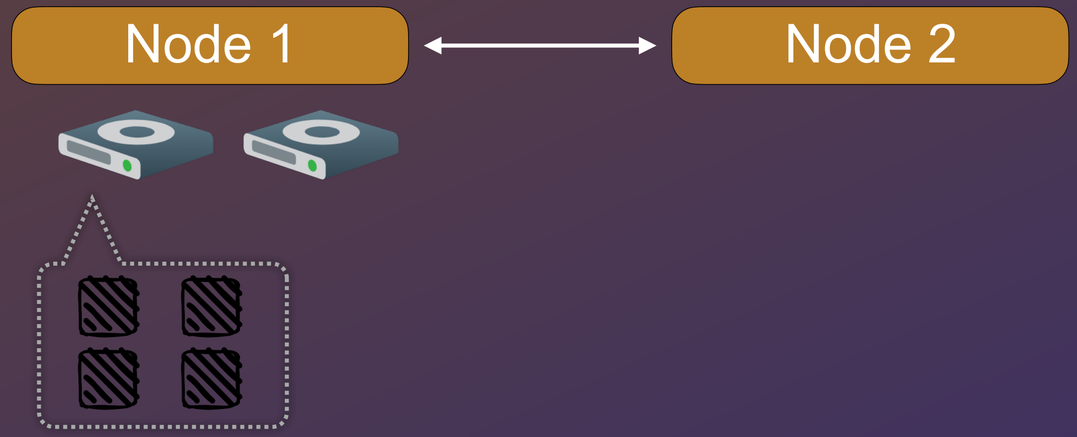
COMMUNITY
THE ASF CONFERENCE
CODE

When Hardware Fails, Ozone Prevails

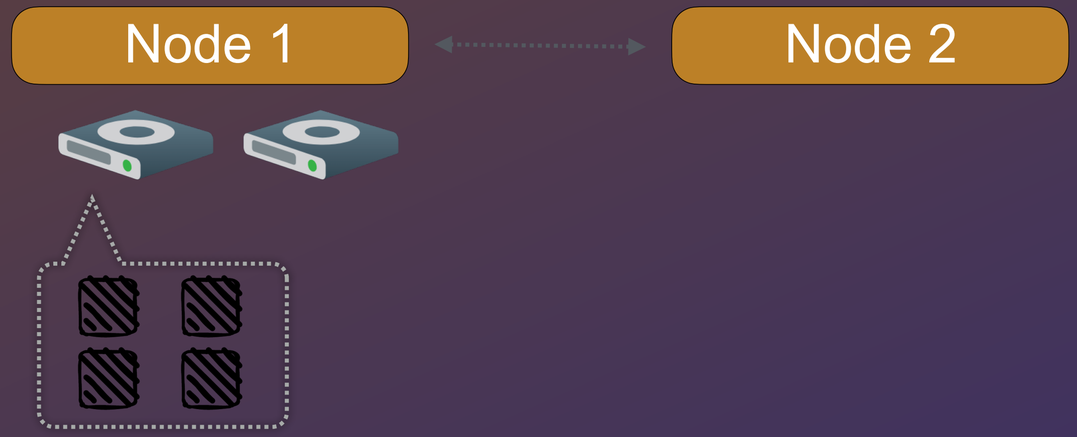
Ethan Rose
Senior Software Engineer, Cloudera Inc
Apache Ozone Committer, PMC

Failure Domains



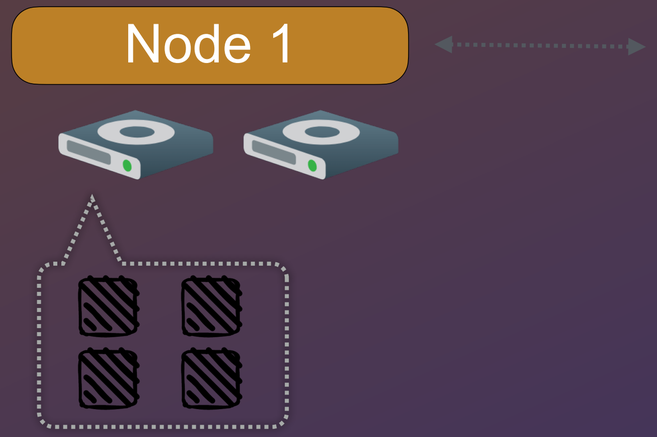
Failure Domains

- Network failure



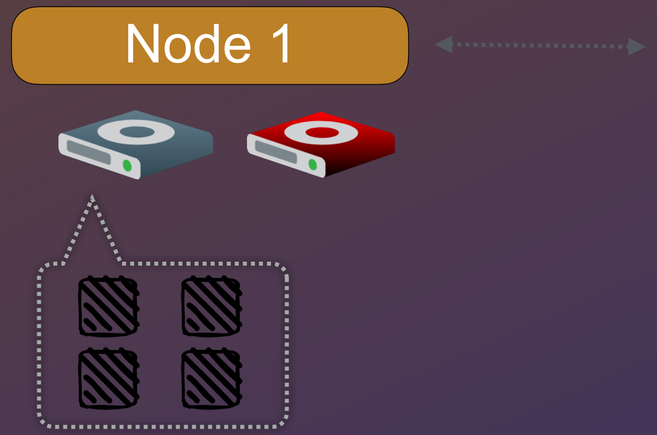
Failure Domains

- Network failure
- Node failure



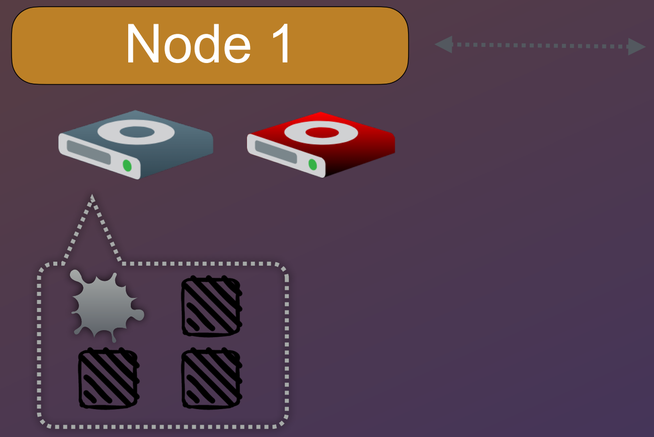
Failure Domains

- Network failure
- Node failure
- Disk failure



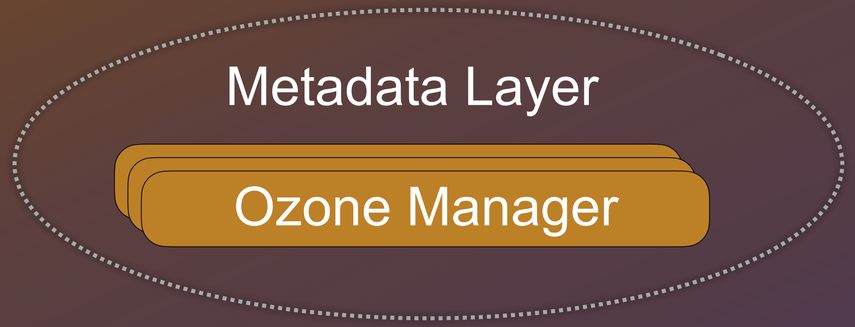
Failure Domains

- Network failure
- Node failure
- Disk failure
- Bit rot/corruption

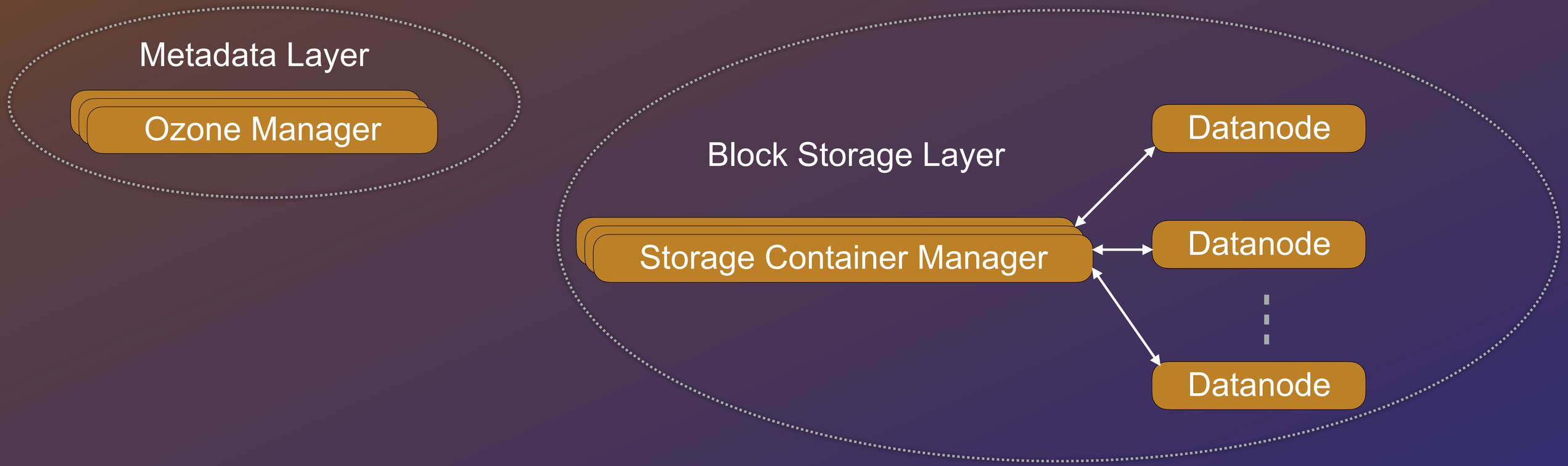


Ozone Architecture

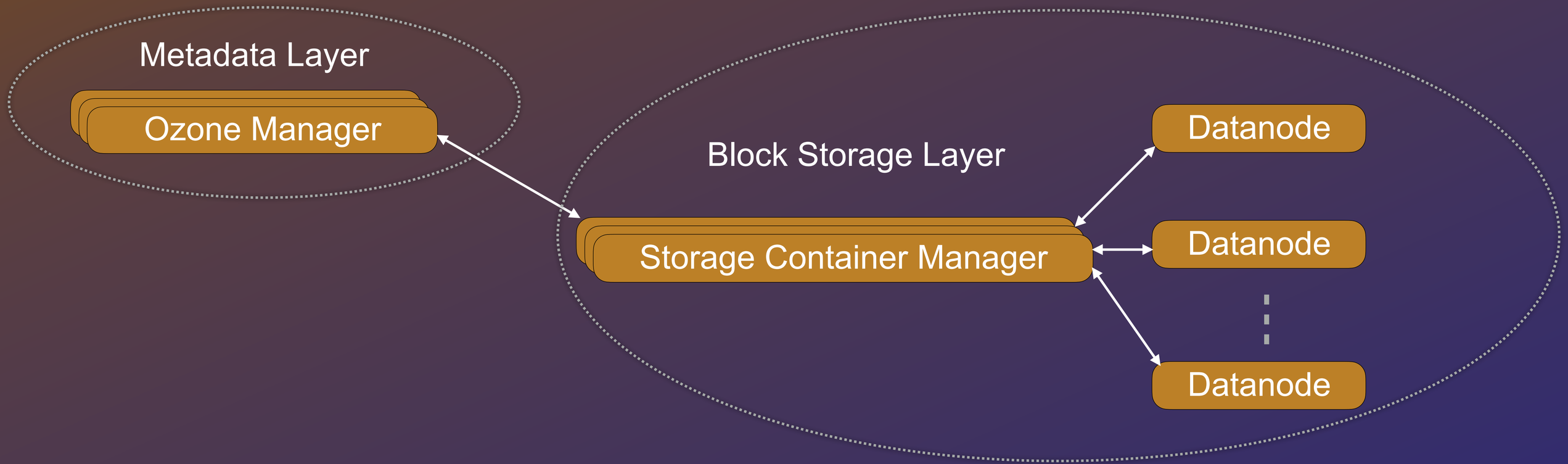
Ozone Architecture



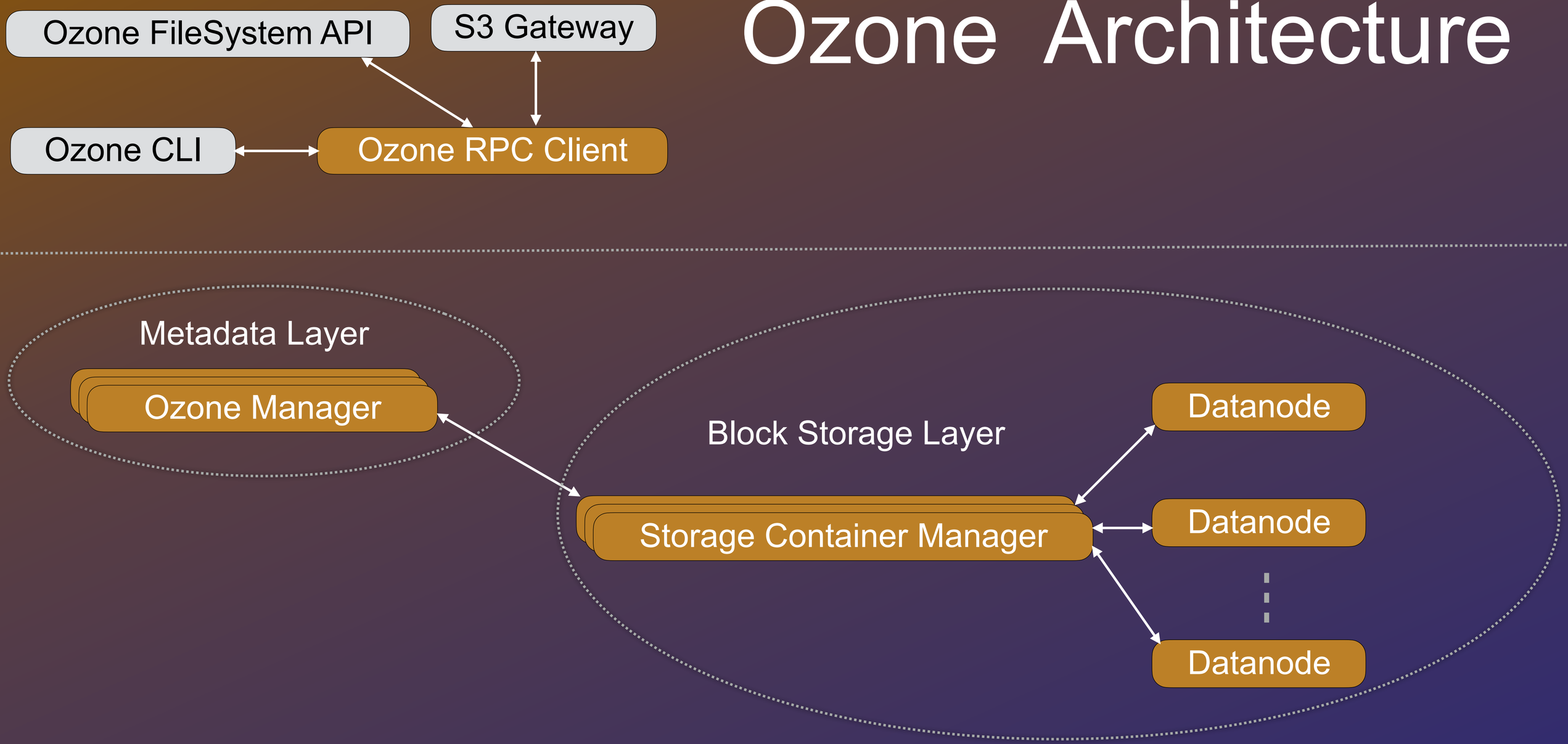
Ozone Architecture



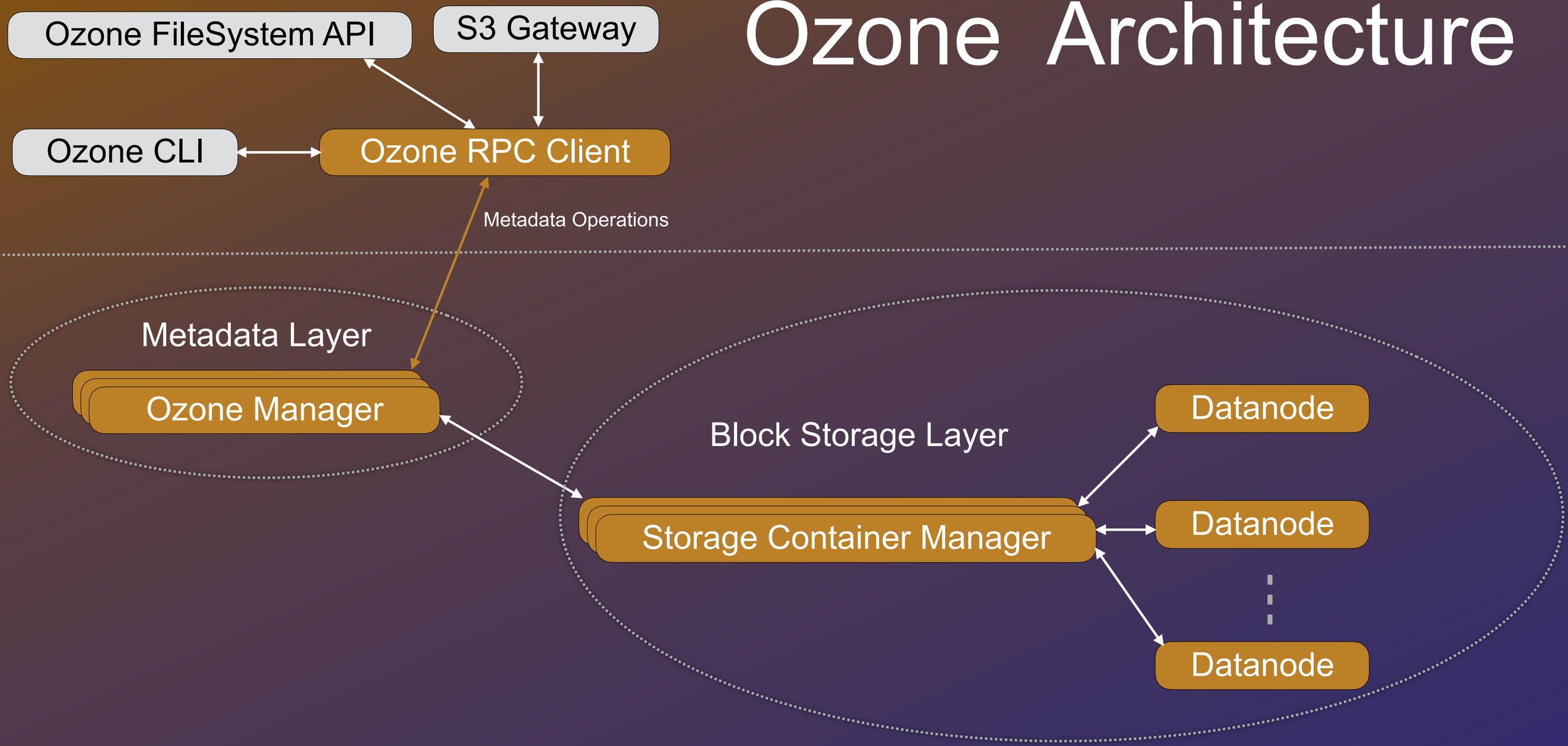
Ozone Architecture



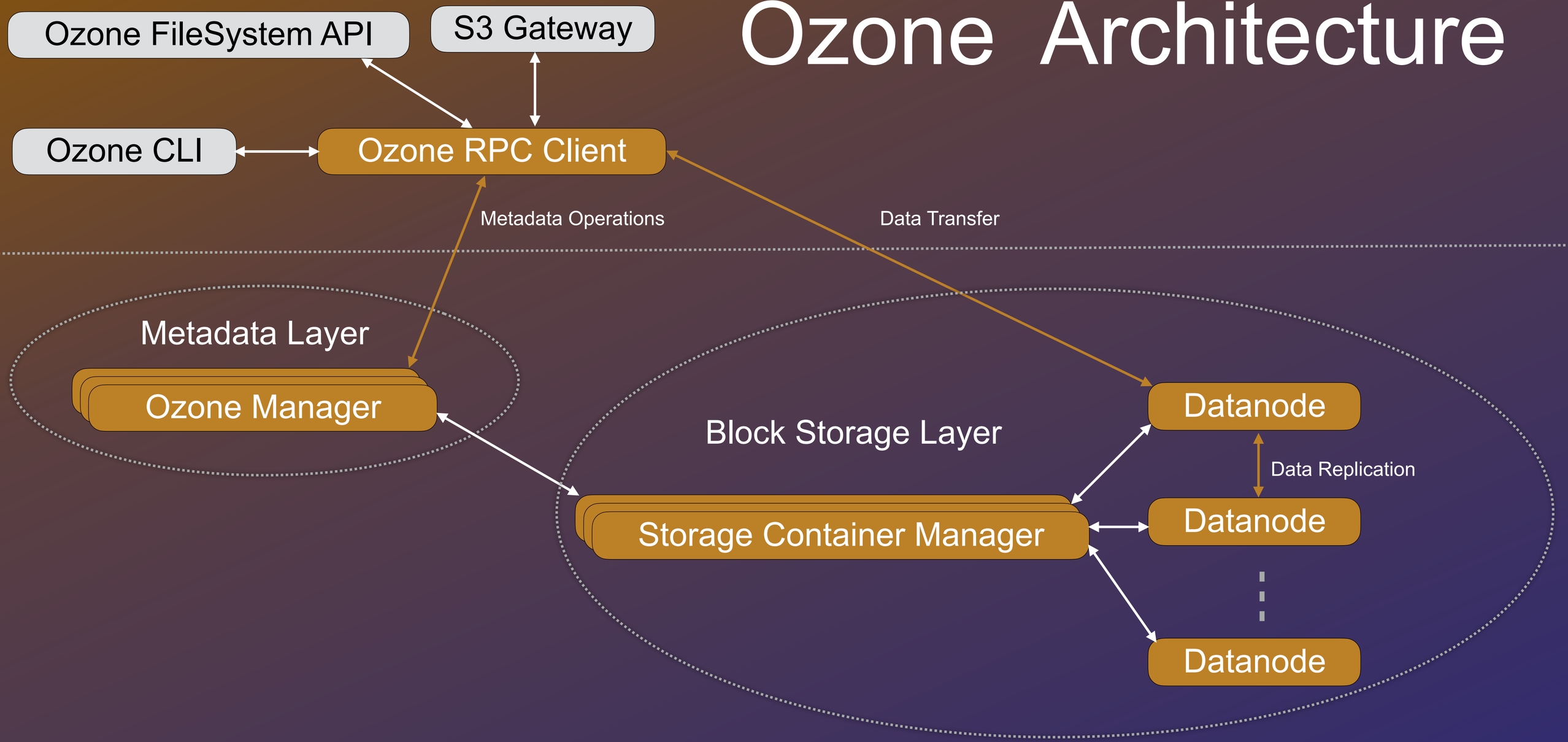
Ozone Architecture



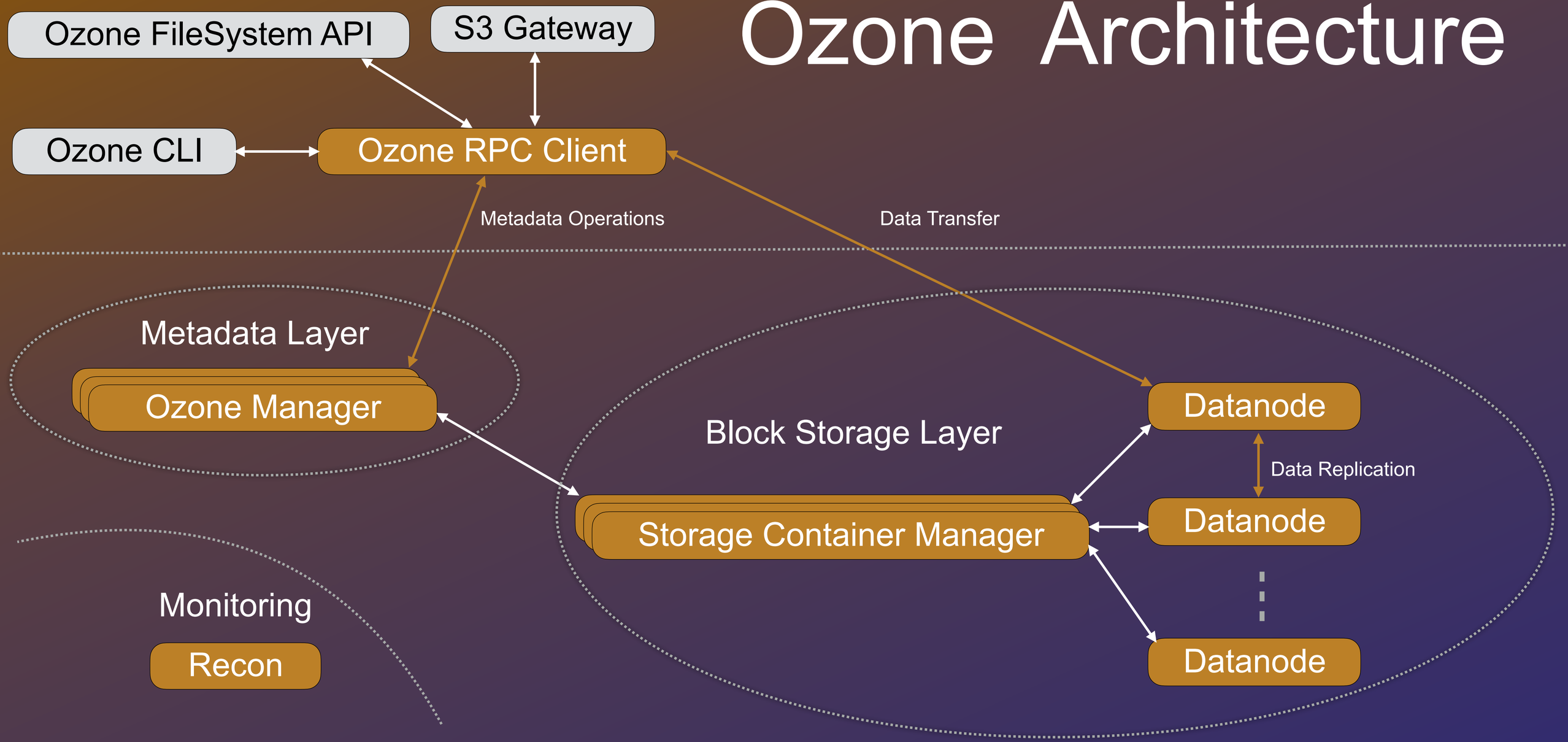
Ozone Architecture



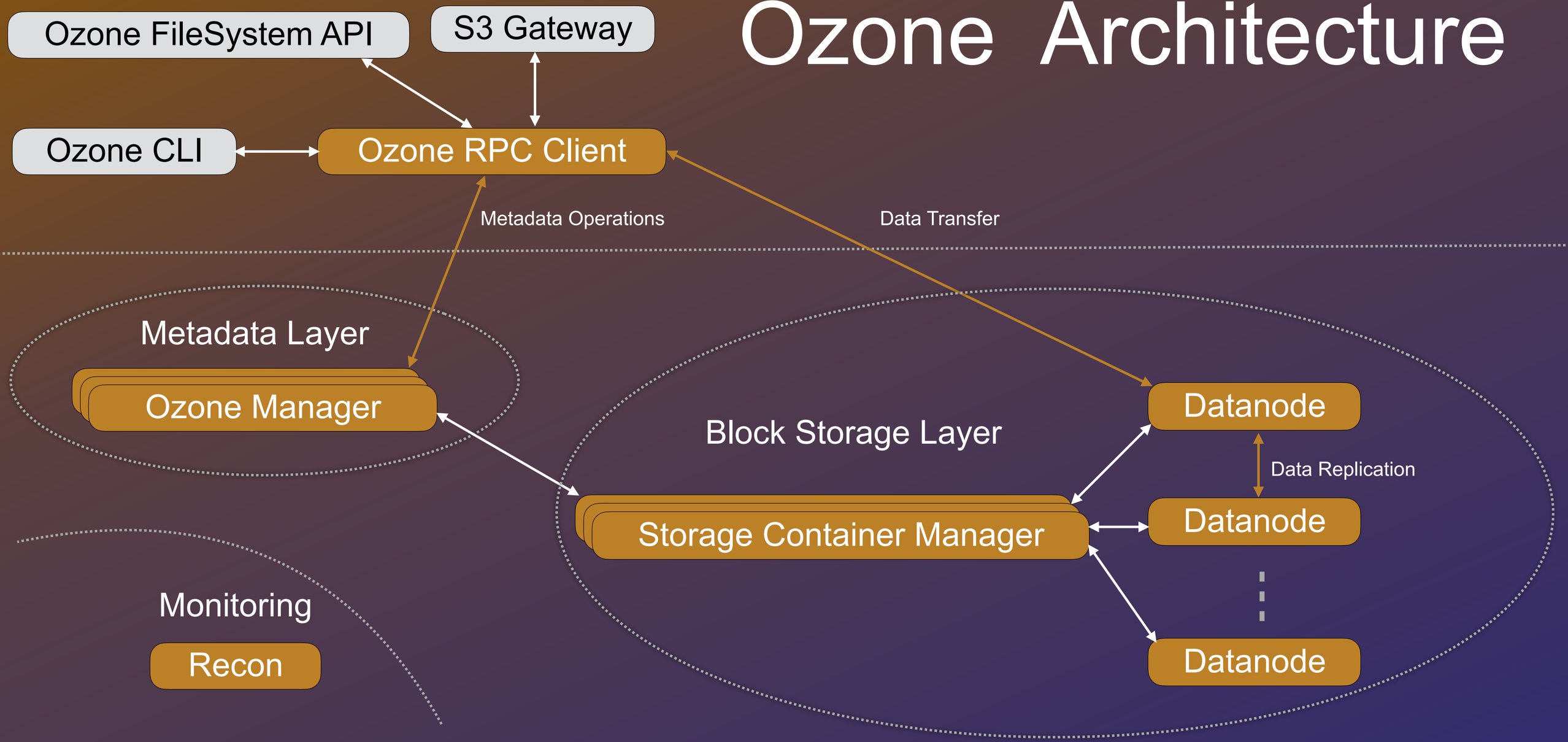
Ozone Architecture



Ozone Architecture

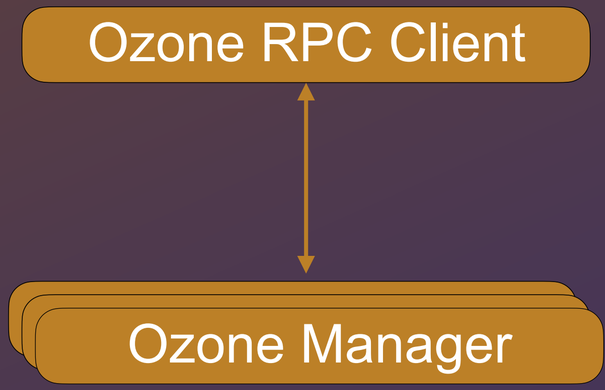


Ozone Architecture



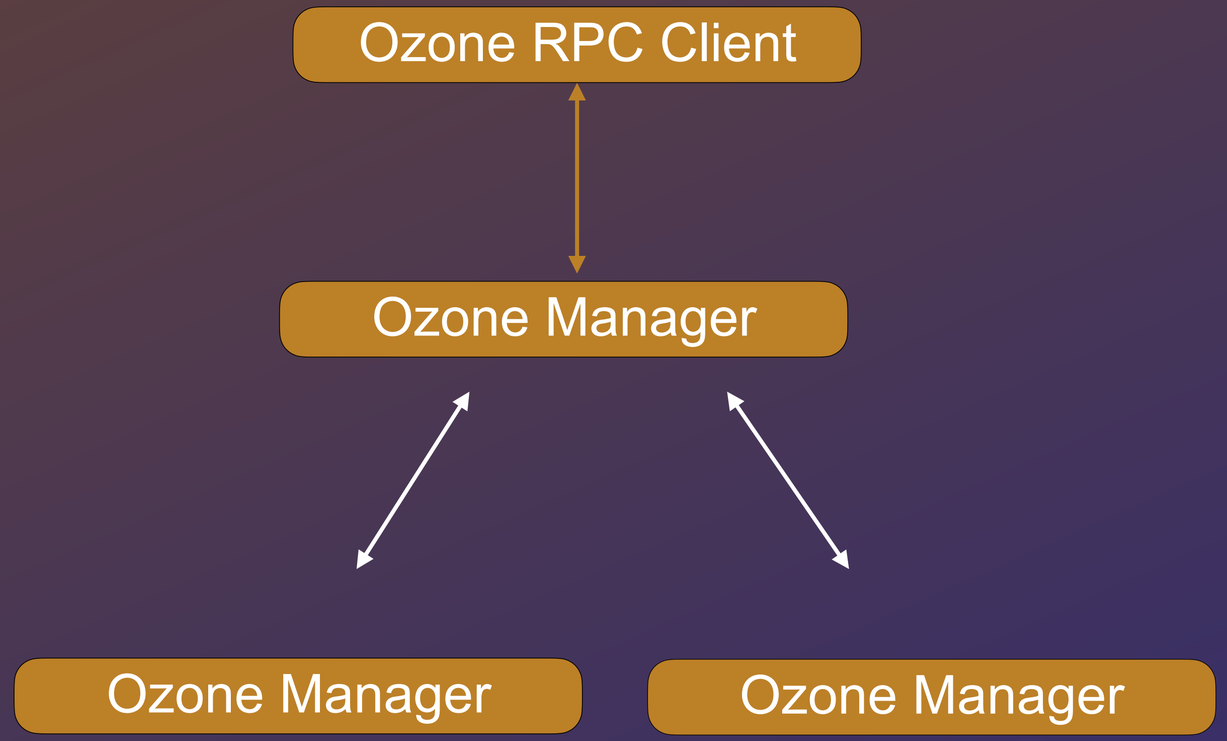
Ozone Manager (OM)

- Store Metadata Only
- Volumes, buckets, keys, ACLs



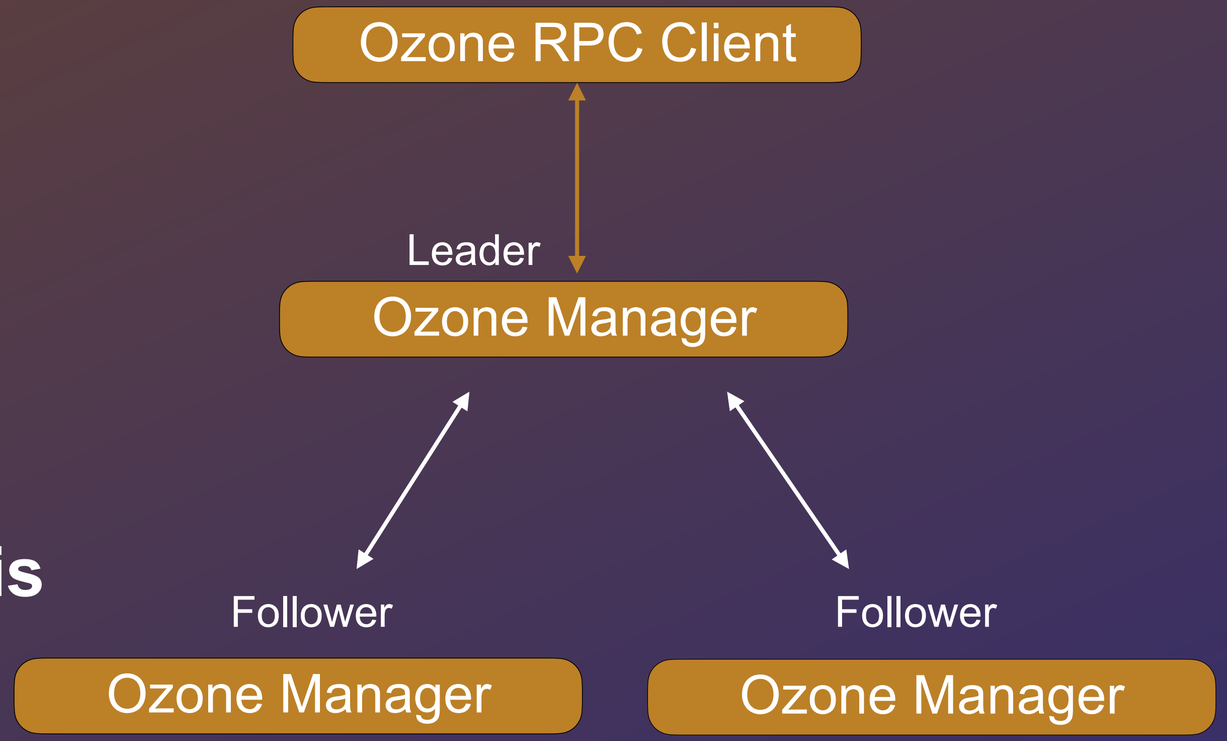
Ozone Manager (OM)

- Store Metadata Only
- Volumes, buckets, keys, ACLs

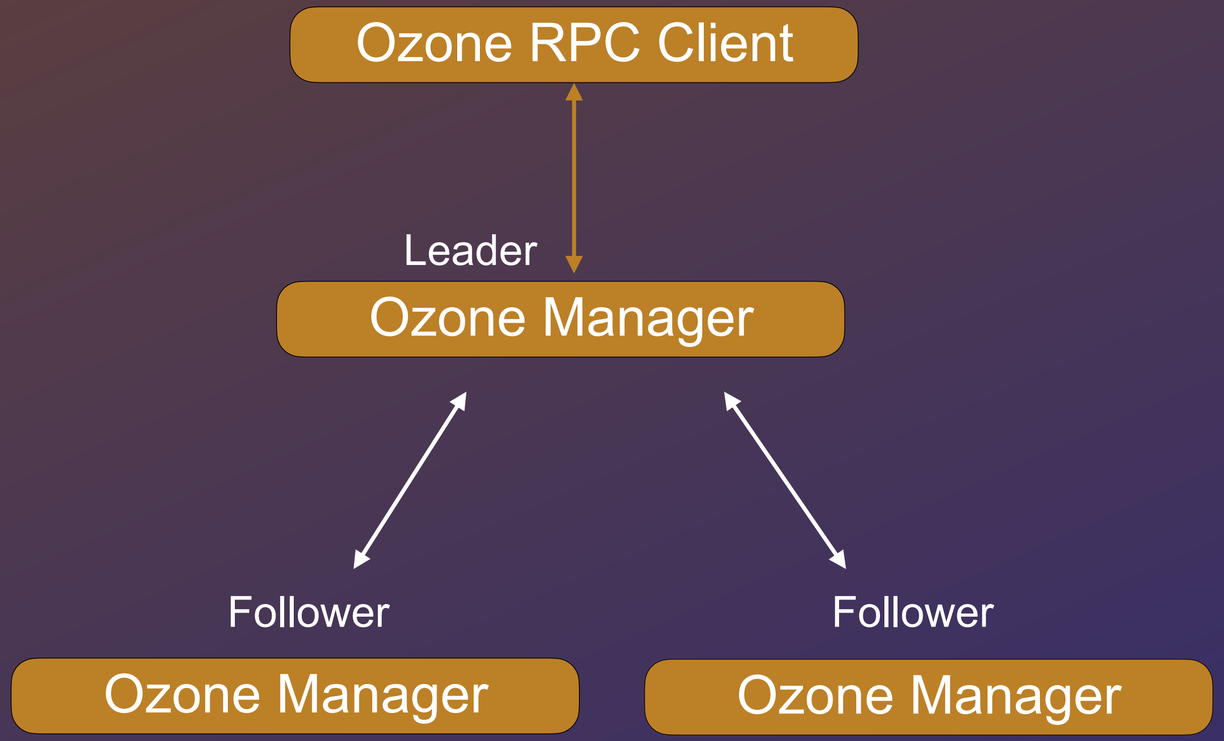


Ozone Manager (OM)

- Store Metadata Only
- Volumes, buckets, keys, ACLs
- Data replicated with **Apache Ratis** (Raft Implementation)



OM + Client: Network Failures

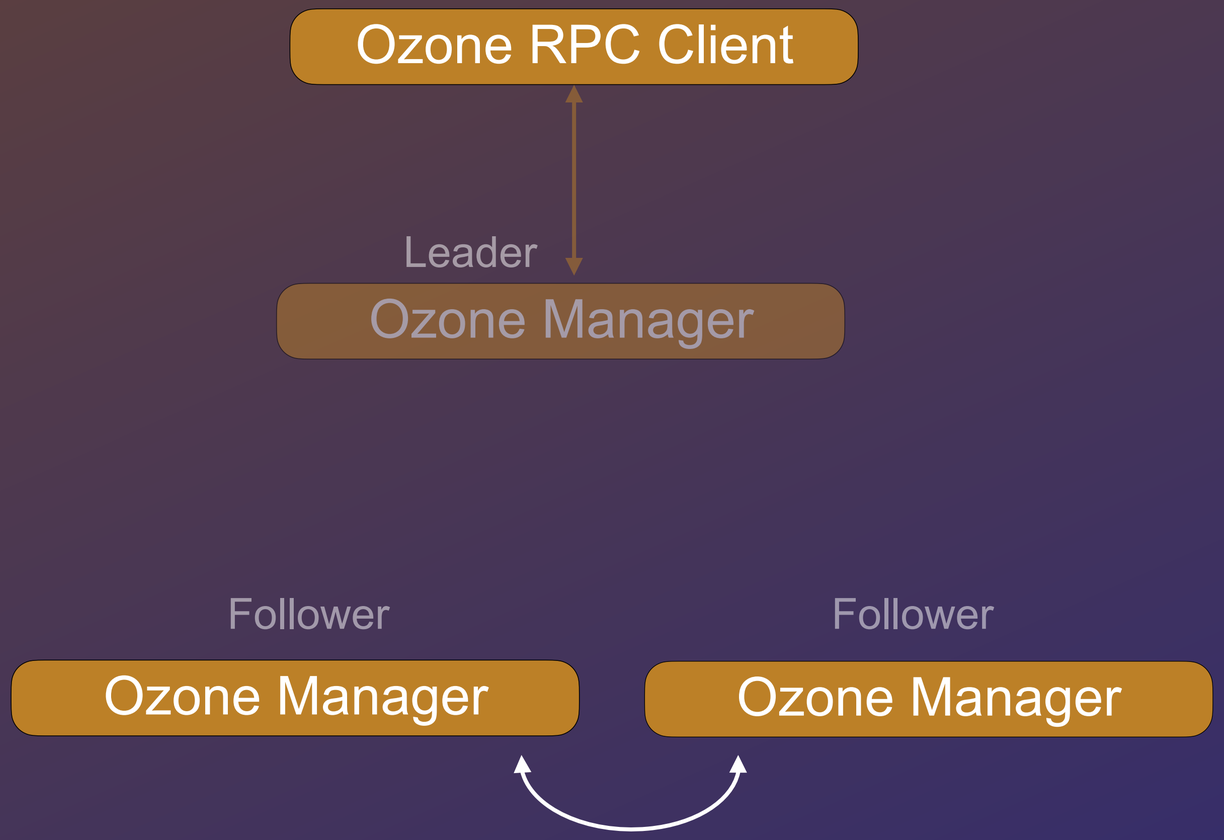


OM + Client: Network Failures

- If current leader is partitioned, a new leader will be elected.

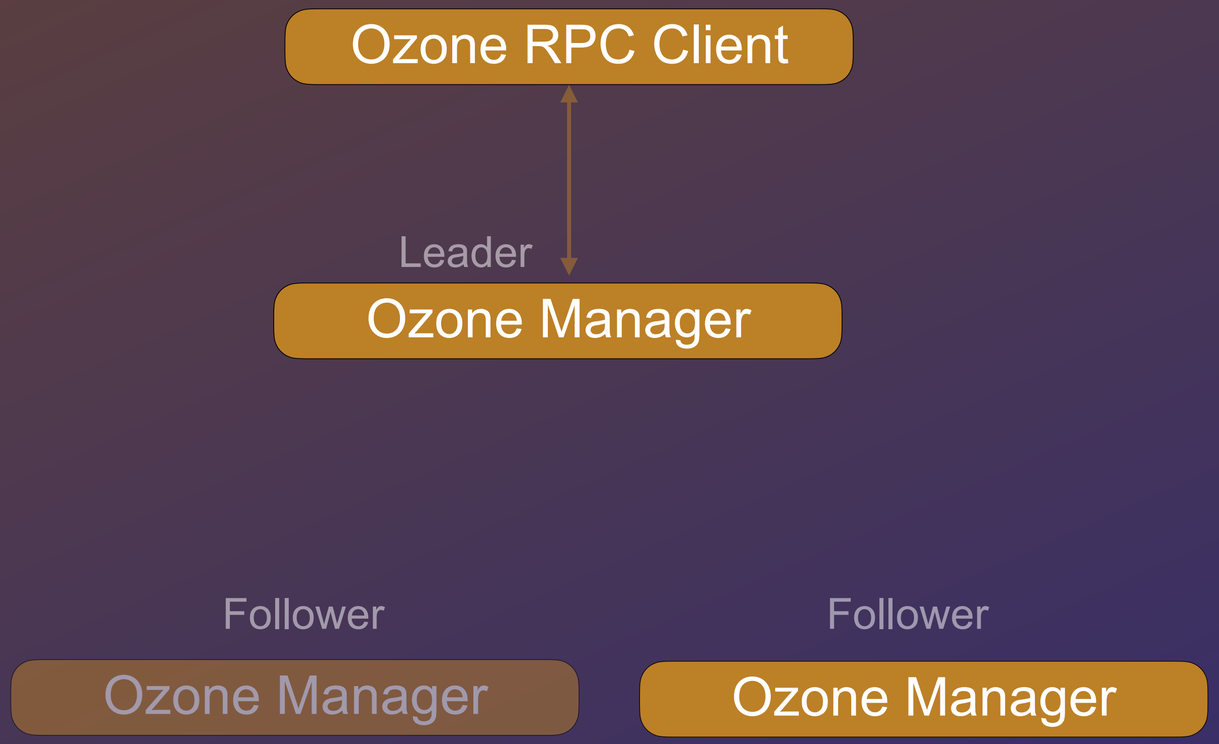
OM + Client: Network Failures

- If current leader is partitioned, a new leader will be elected.



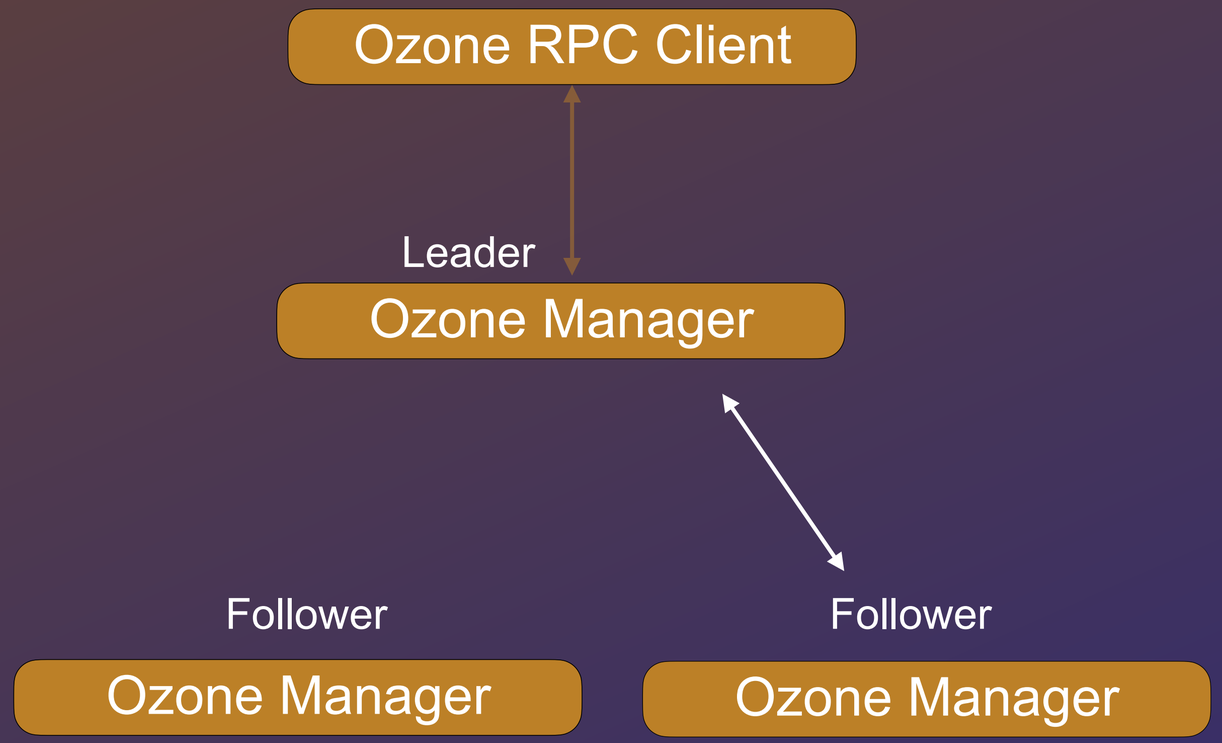
OM + Client: Network Failures

- If current leader is partitioned, a new leader will be elected.



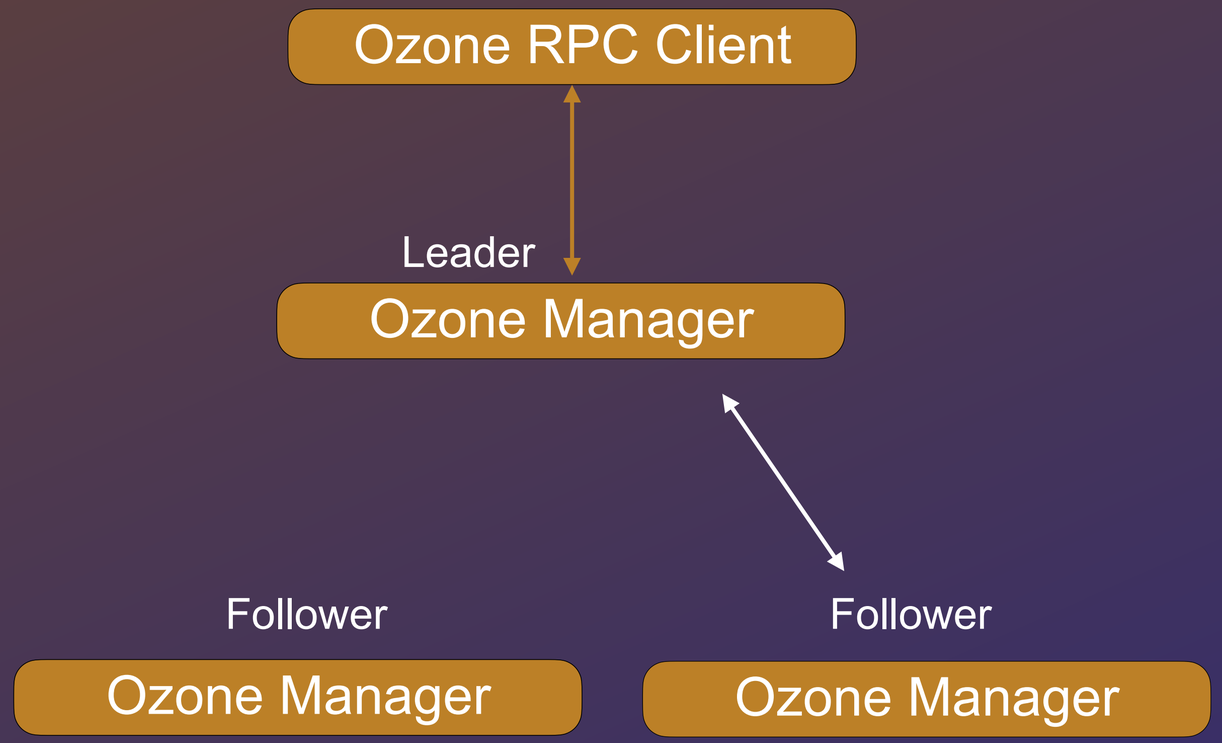
OM + Client: Network Failures

- If current leader is partitioned, a new leader will be elected.



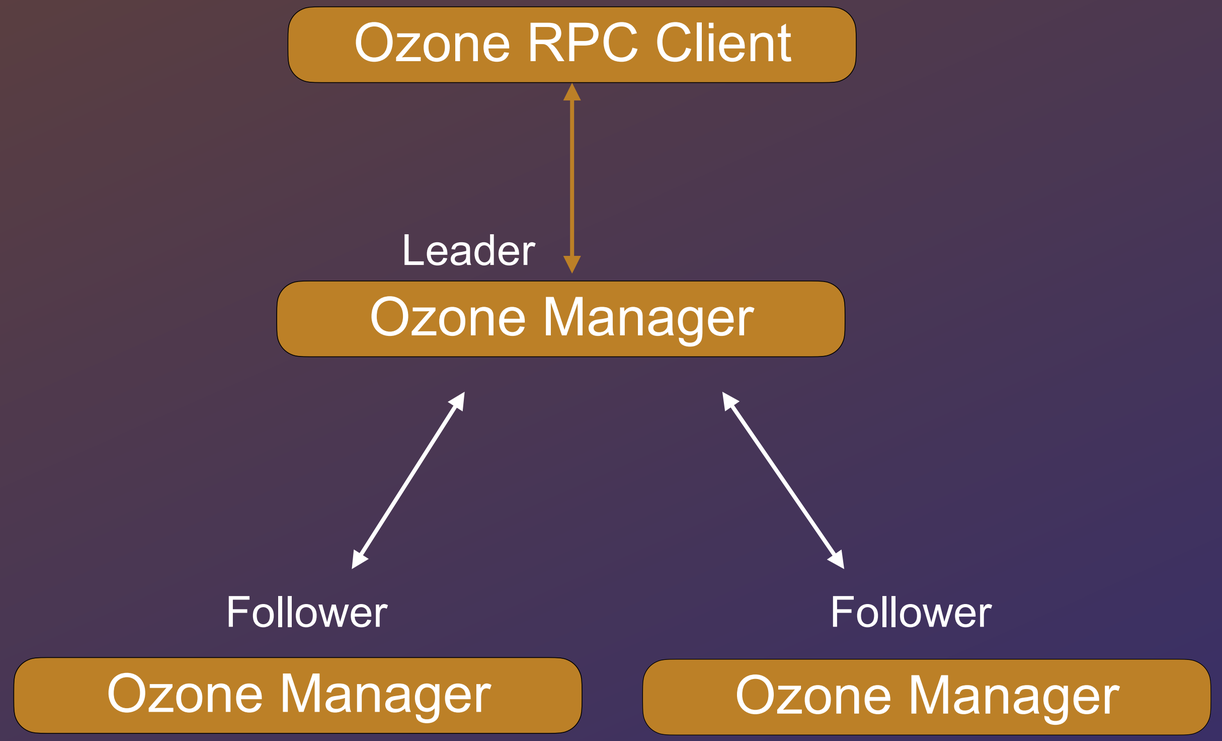
OM + Client: Network Failures

- If current leader is partitioned, a new leader will be elected.
- Client read/write must be retried on the new leader.



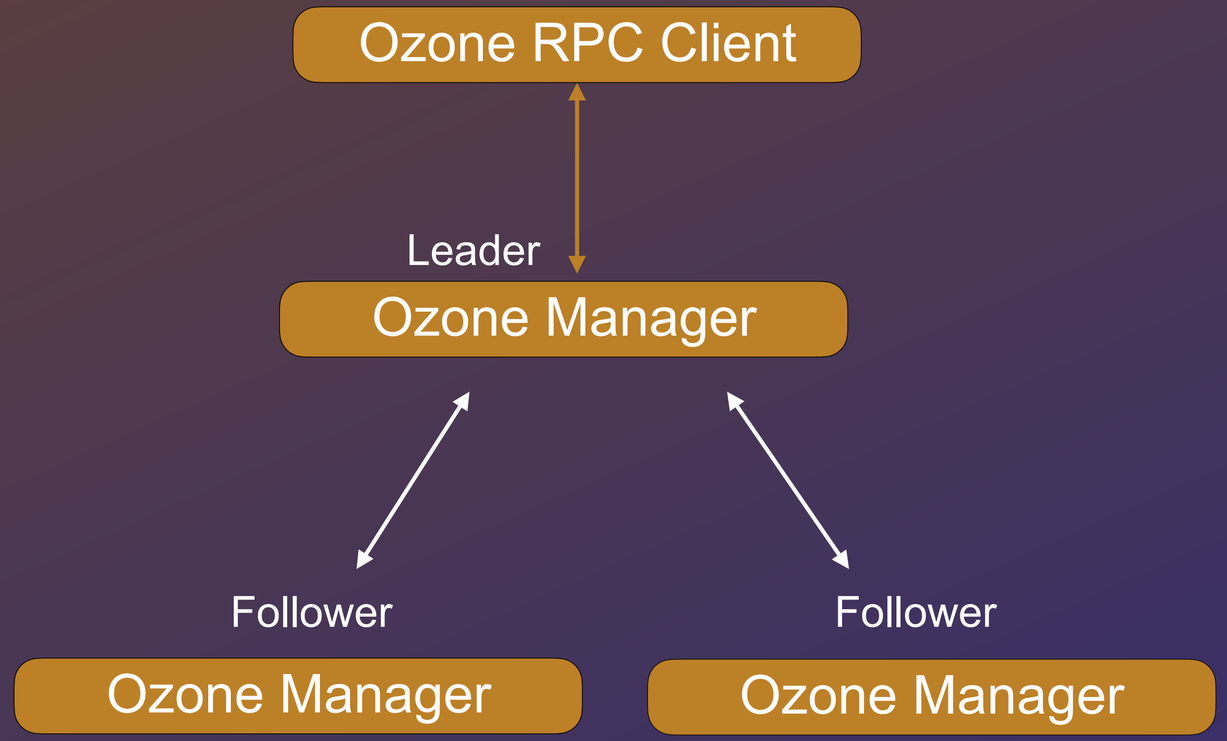
OM + Client: Network Failures

- If current leader is partitioned, a new leader will be elected.
- Client read/write must be retried on the new leader.
- System recovers automatically when the network heals



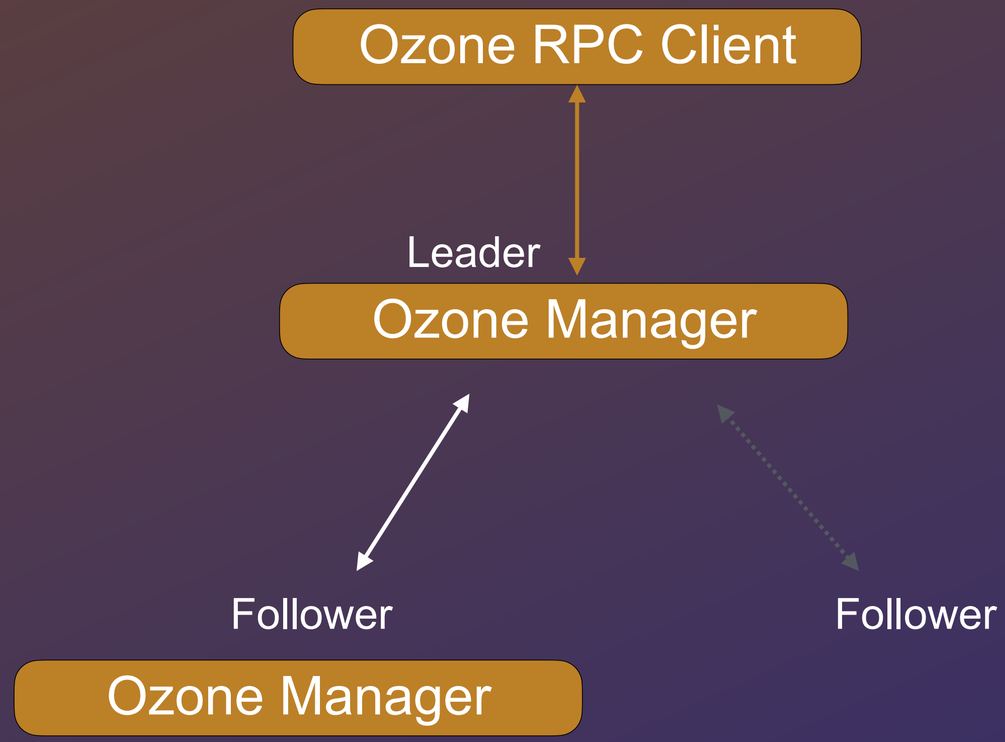
OM + Client: Node Failures

- Needs 2/3 (majority) of OMs available for service.
- Need at least one follower available to commit
- System continues to run while node is replaced



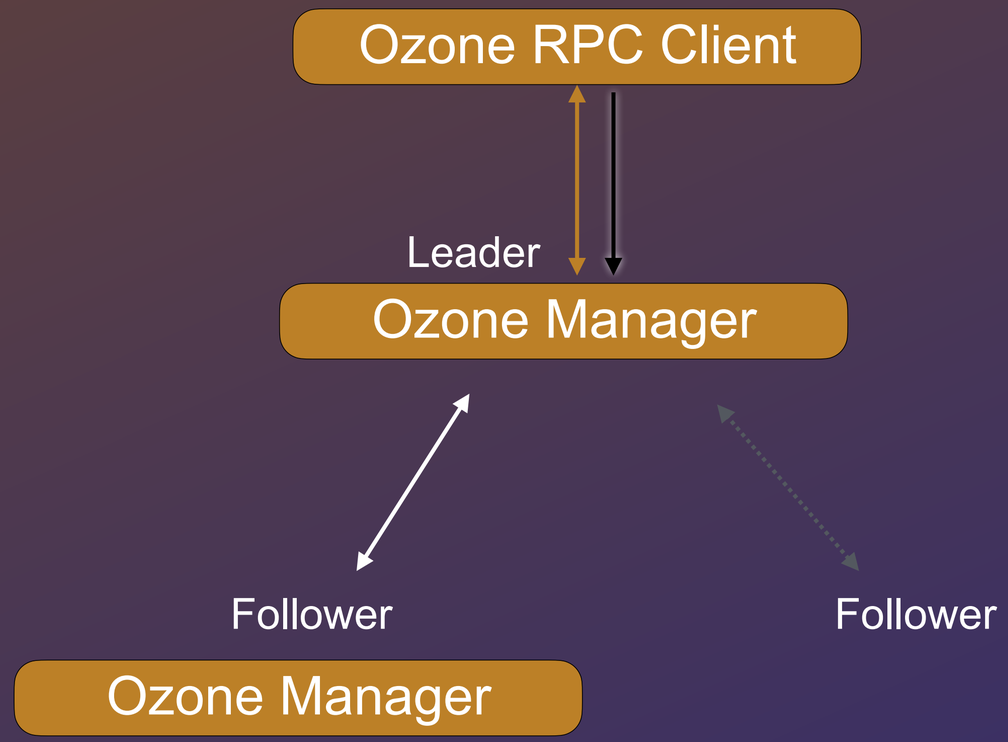
OM + Client: Node Failures

- Needs 2/3 (majority) of OMs available for service.
- Need at least one follower available to commit
- System continues to run while node is replaced



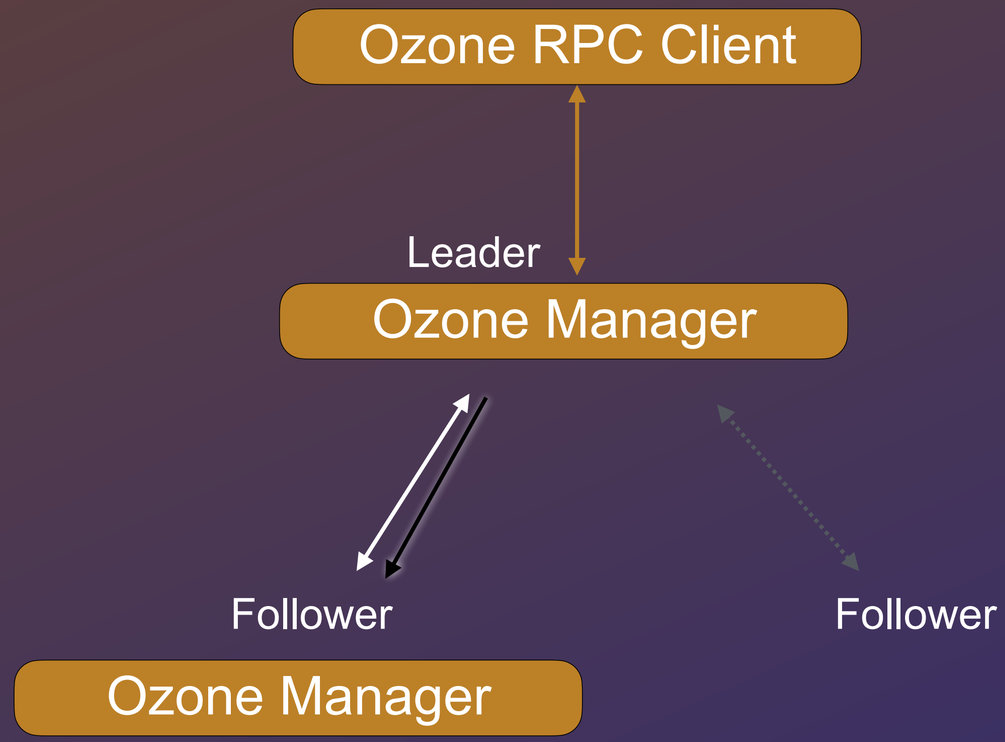
OM + Client: Node Failures

- Needs 2/3 (majority) of OMs available for service.
- Need at least one follower available to commit
- System continues to run while node is replaced



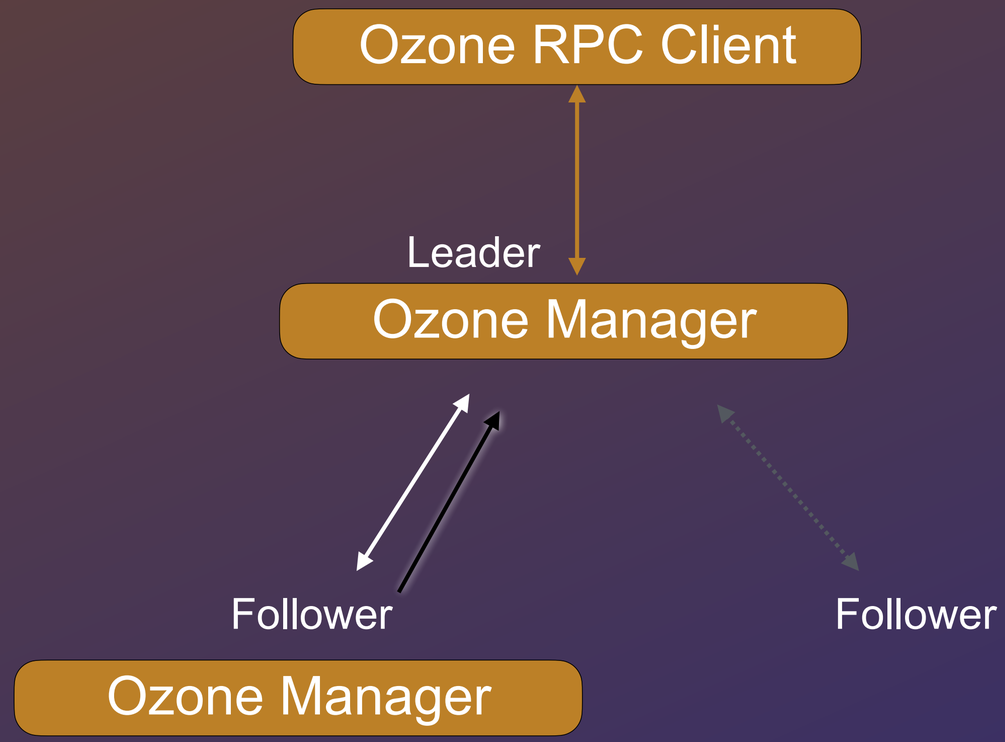
OM + Client: Node Failures

- Needs 2/3 (majority) of OMs available for service.
- Need at least one follower available to commit
- System continues to run while node is replaced



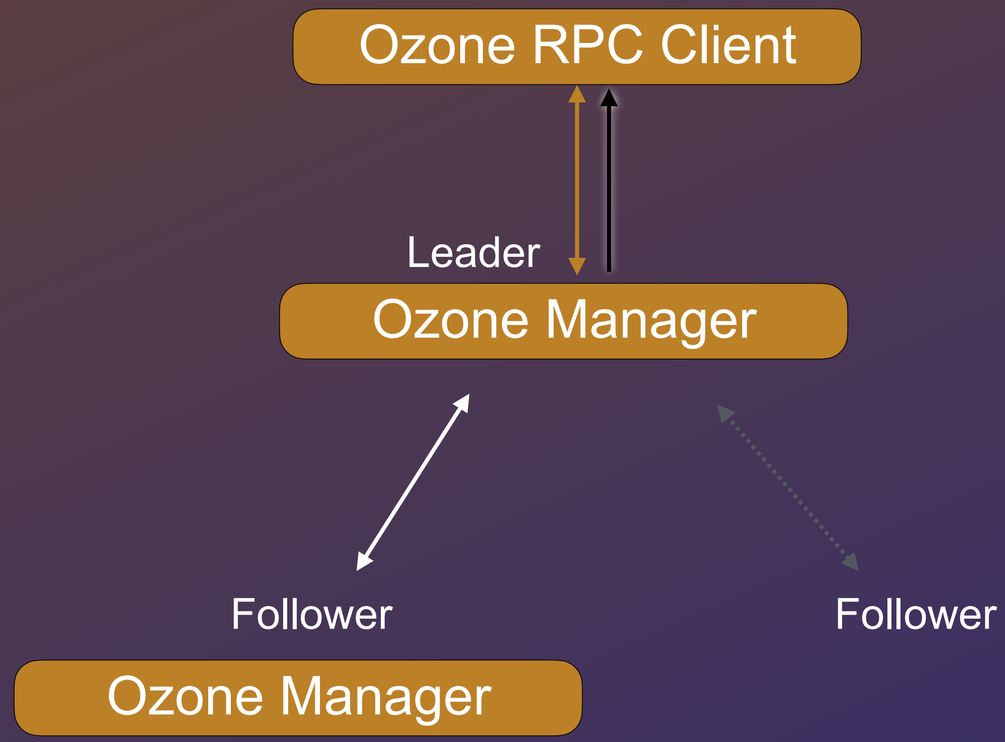
OM + Client: Node Failures

- Needs 2/3 (majority) of OMs available for service.
- Need at least one follower available to commit
- System continues to run while node is replaced



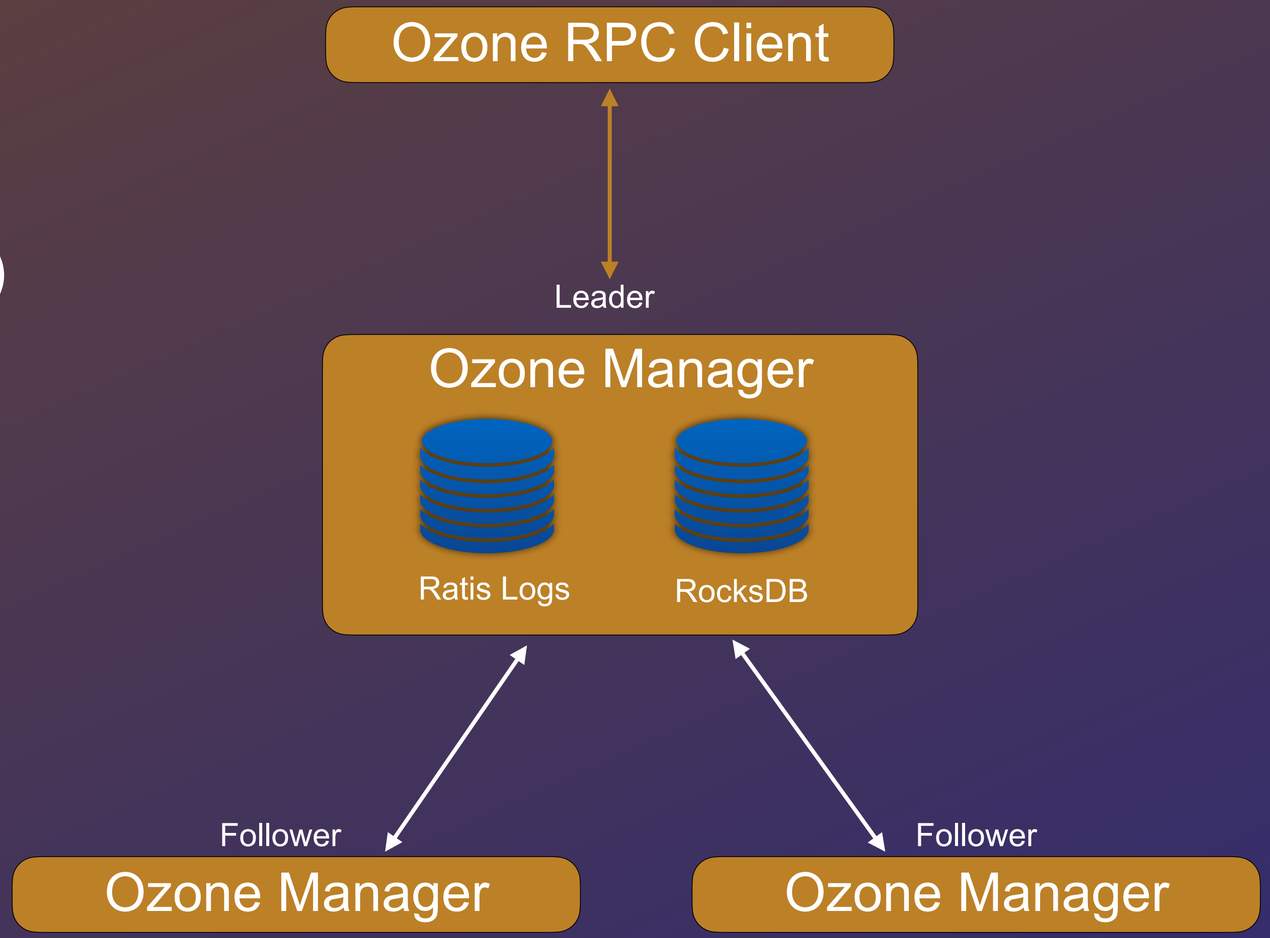
OM + Client: Node Failures

- Needs 2/3 (majority) of OMs available for service.
- Need at least one follower available to commit
- System continues to run while node is replaced



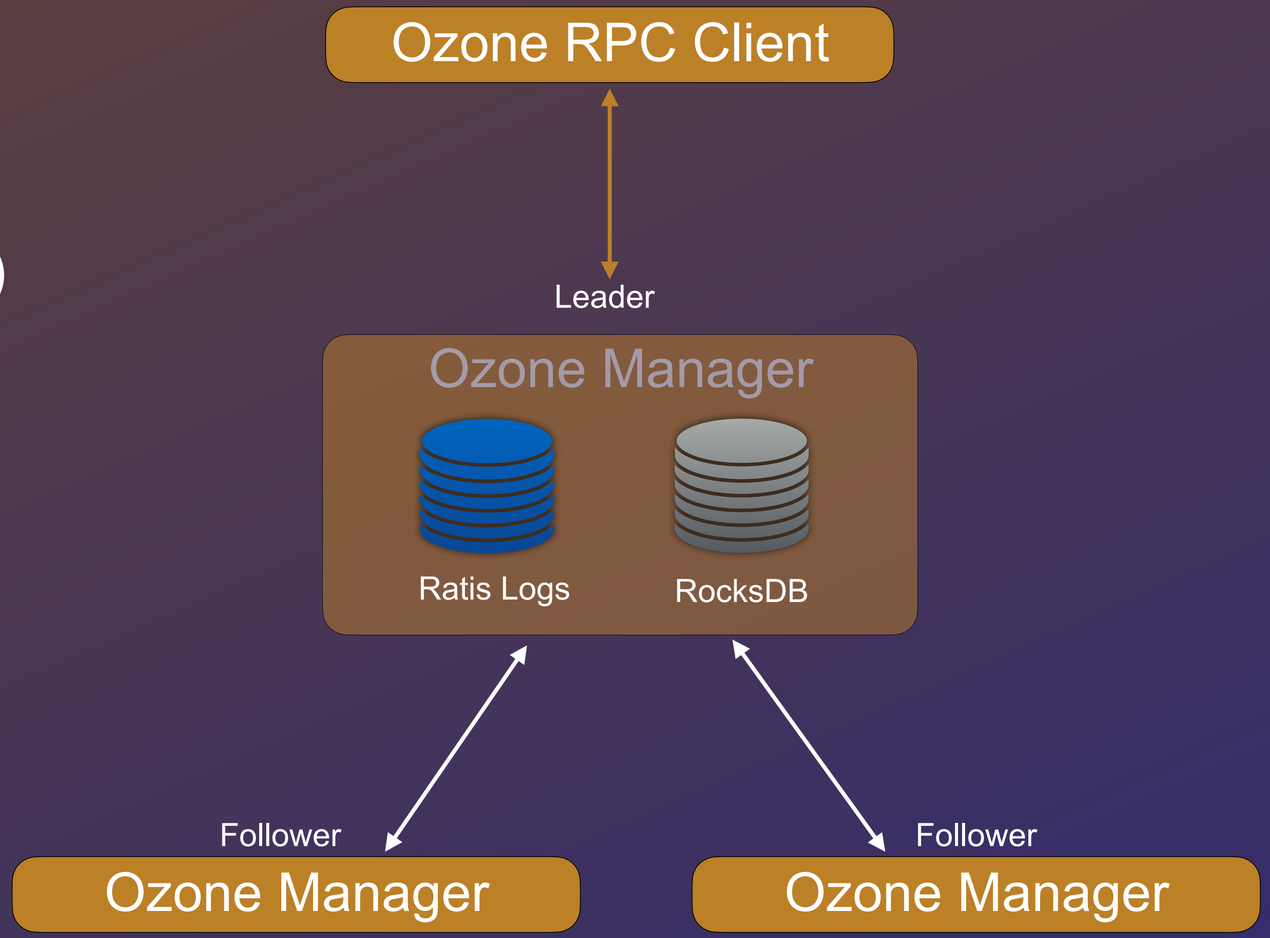
OM: Disk Failure/DB Corruption

- OM Stores data in 2 disks:
  1. Ratis Logs
  2. RocksDB (Ratis state machine)



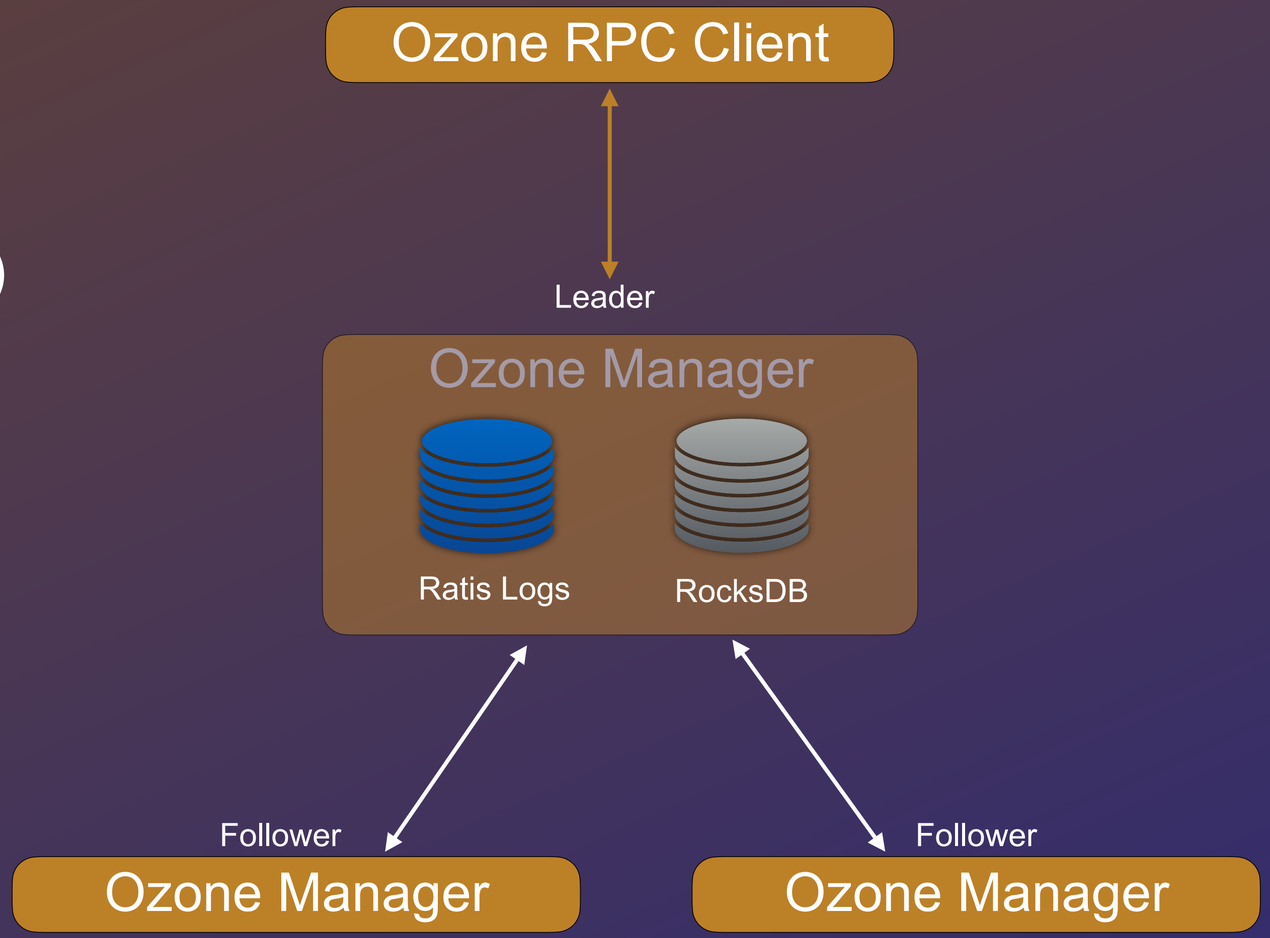
OM: Disk Failure/DB Corruption

- OM Stores data in 2 disks:
  1. Ratis Logs
  2. RocksDB (Ratis state machine)



OM: Disk Failure/DB Corruption

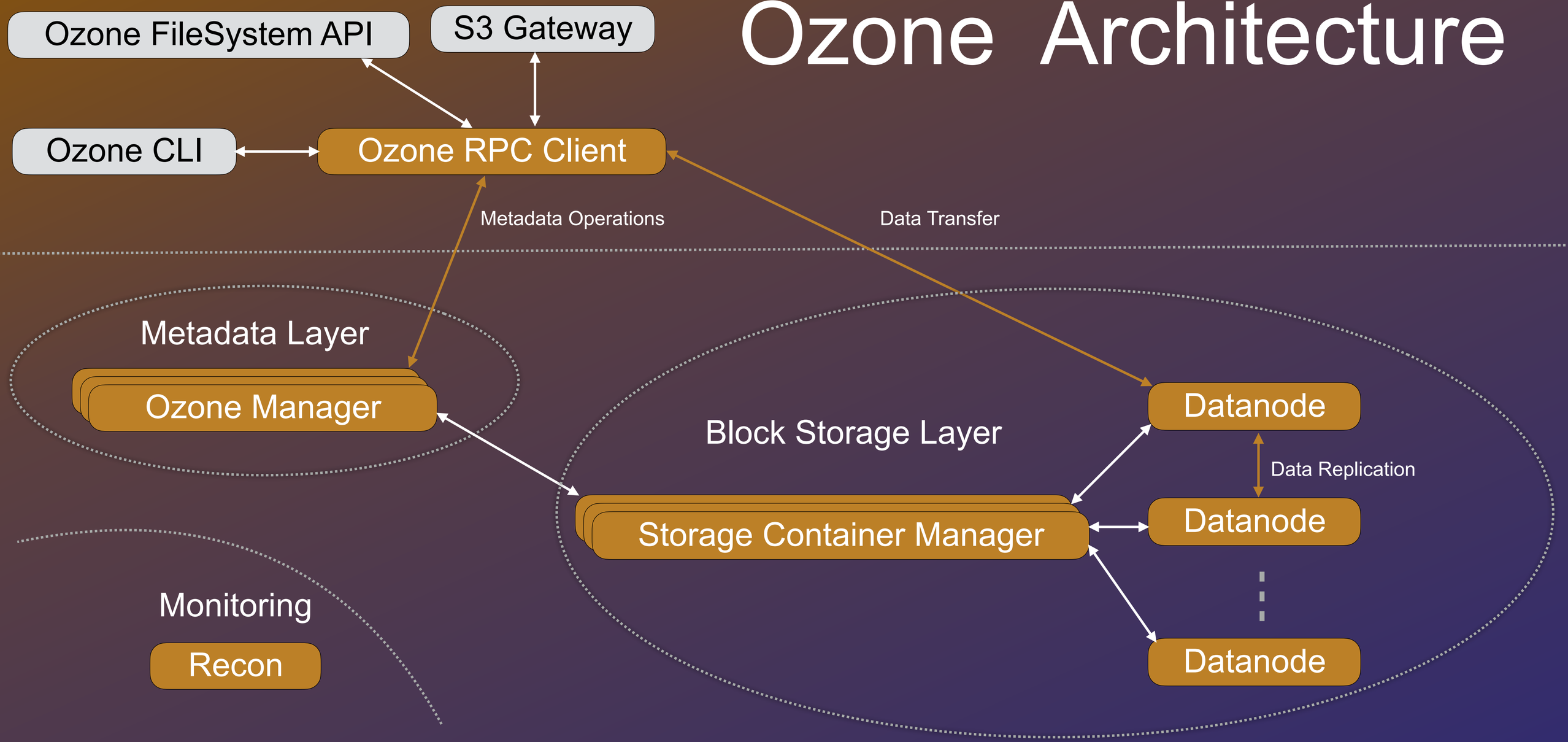
- OM Stores data in 2 disks:
  1. Ratis Logs
  2. RocksDB (Ratis state machine)
- Failure to write to disk or DB is fatal
  - All previous transactions must be applied before the current one.
  - RAID 1 recommended



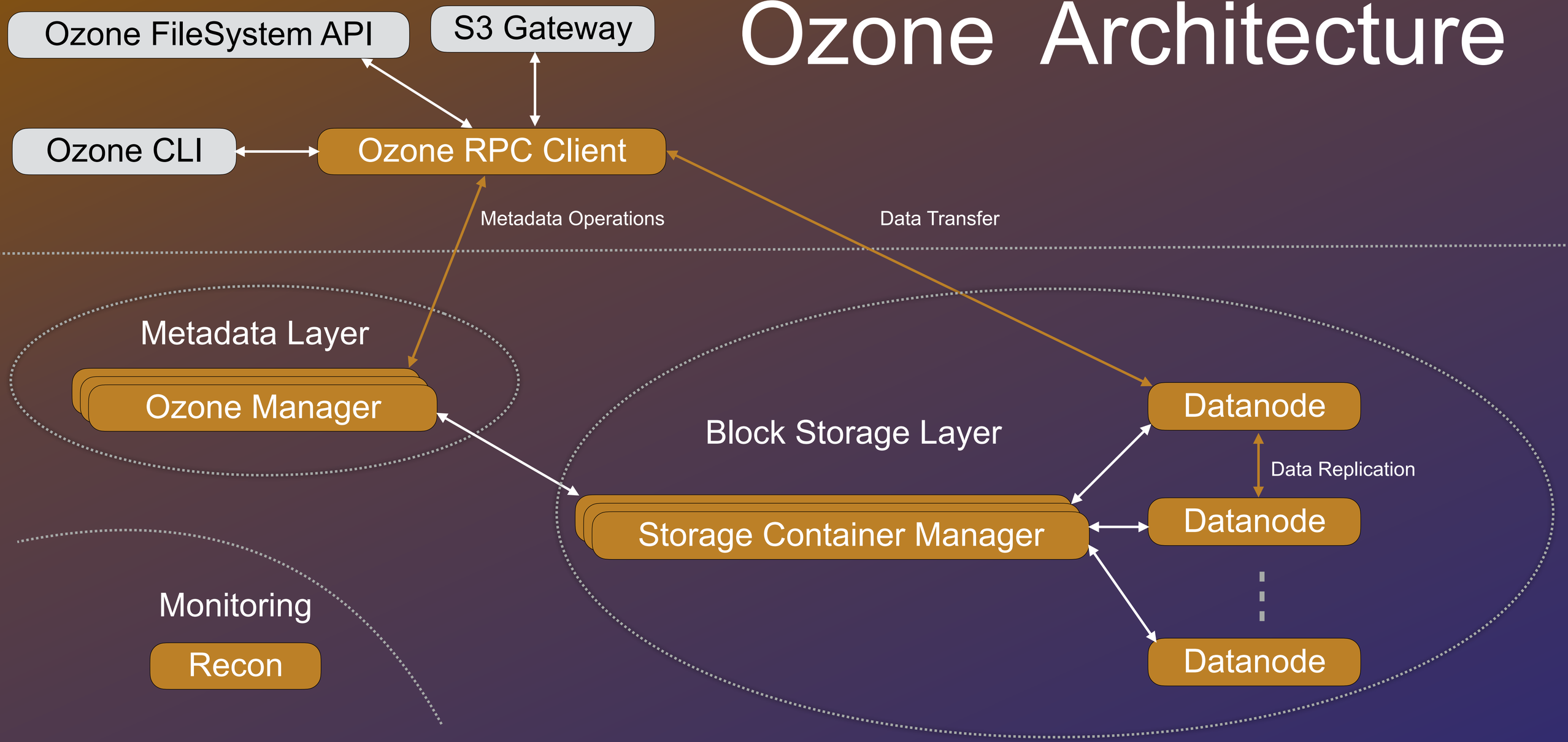
OM Faults: Relevant Config Keys

`ozone.om.db.dirs`	Disk for RocksDB
`ozone.om.ratis.storage.dir`	Disk for Ratis logs
`raft.server.leaderelection.leader.step-down.wait-time`	Time after the last heartbeat where the leader will step down.
`raft.server.rpc.timeout.min`	Lower bound of the random timeout for a follower to initiate a leader election.
`raft.server.rpc.timeout.max`	Upper bound of the random timeout for a follower to initiate a leader election.

Ozone Architecture

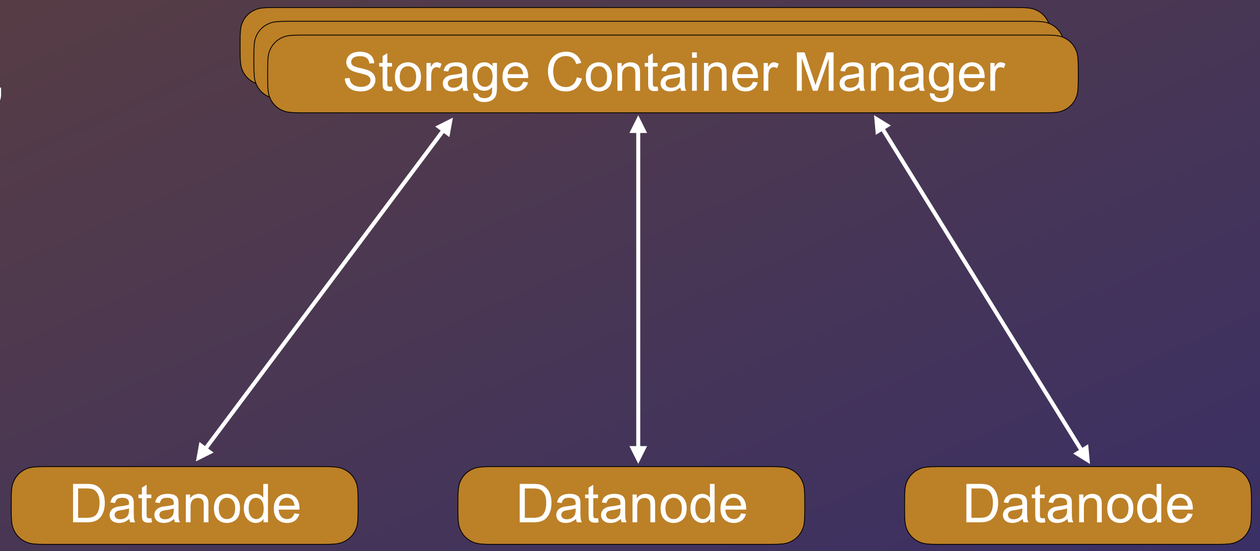


Ozone Architecture



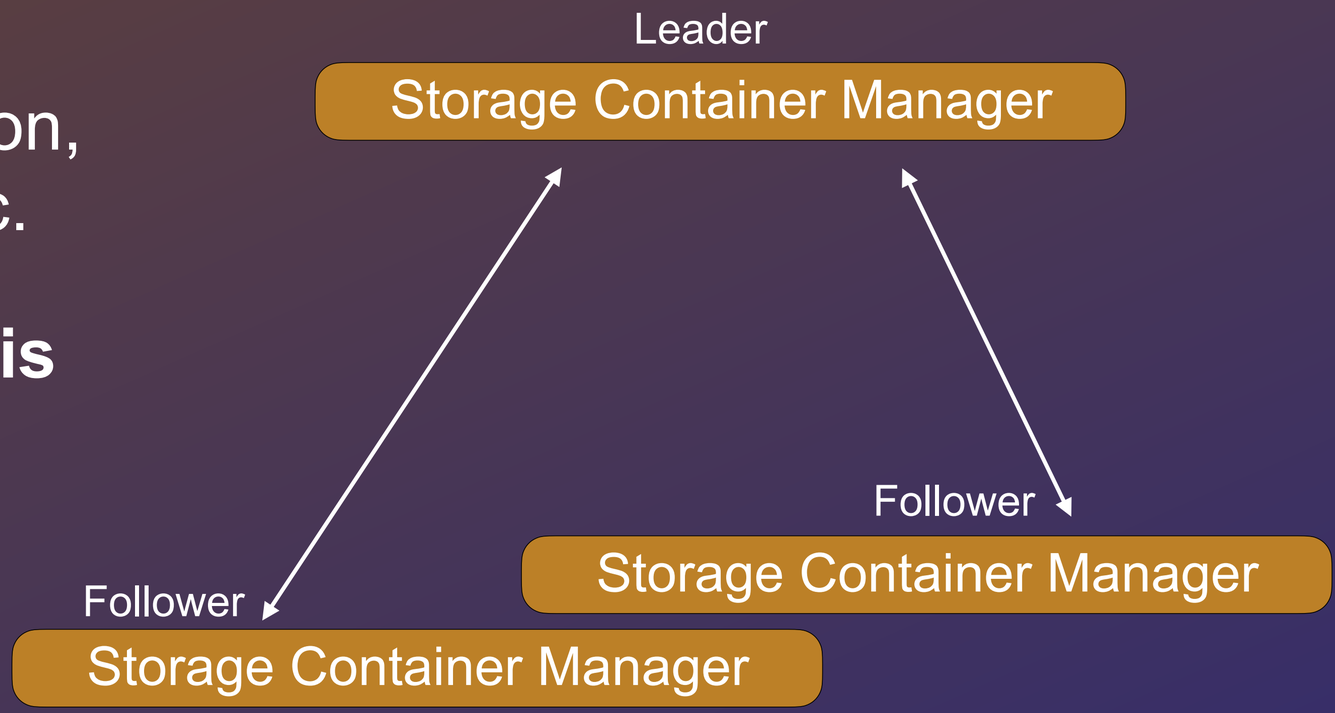
Storage Container Manager (SCM)

- Tracks metadata of storage containers (unit of replication)
- Write pipelines, replica information, cluster balancing, replication, etc.



Storage Container Manager (SCM)

- Tracks metadata of storage containers (unit of replication)
- Write pipelines, replica information, cluster balancing, replication, etc.
- Data replicated with **Apache Ratis** similar to OM
- SCM failures are handled in the same way as OM

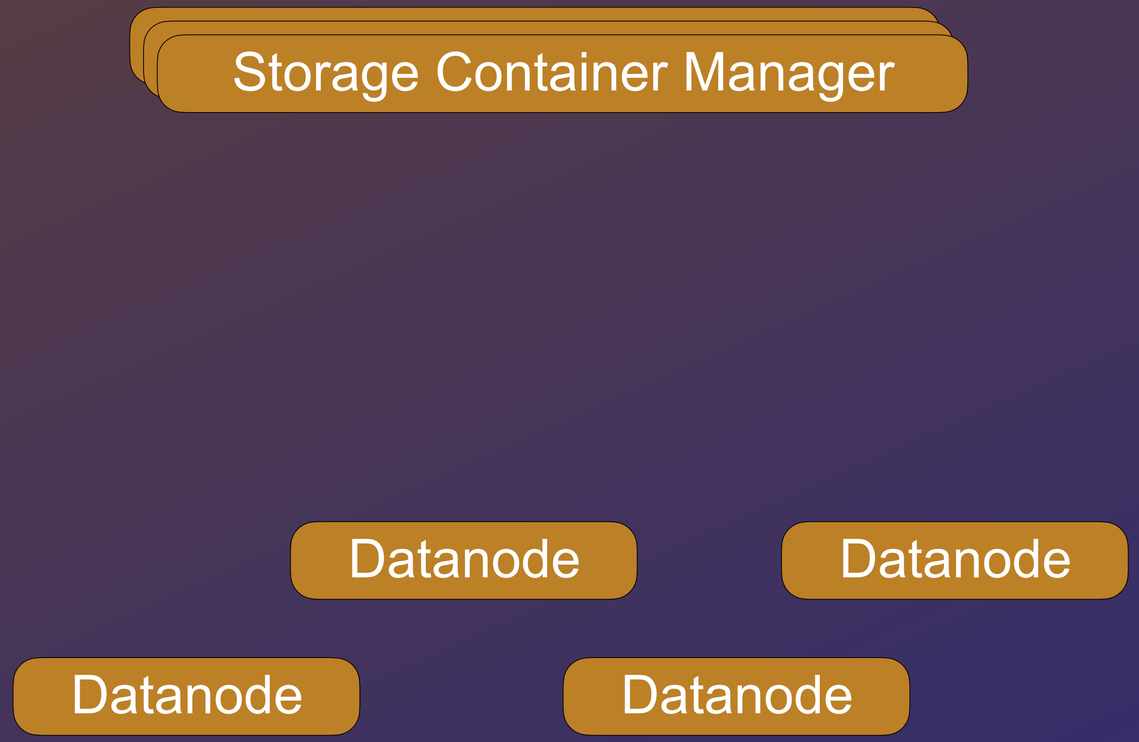


SCM Faults: Relevant Config Keys

`ozone.scm.db.dirs`	Disk for RocksDB
`ozone.scm.ha.ratis.storage.dir`	Disk for Ratis logs
`raft.server.leaderelection .leader.step-down.wait-time`	Time after the last heartbeat where the leader will step down.
`raft.server.rpc.timeout.min`	Lower bound of the random timeout for a follower to initiate a leader election.
`raft.server.rpc.timeout.max`	Upper bound of the random timeout for a follower to initiate a leader election.

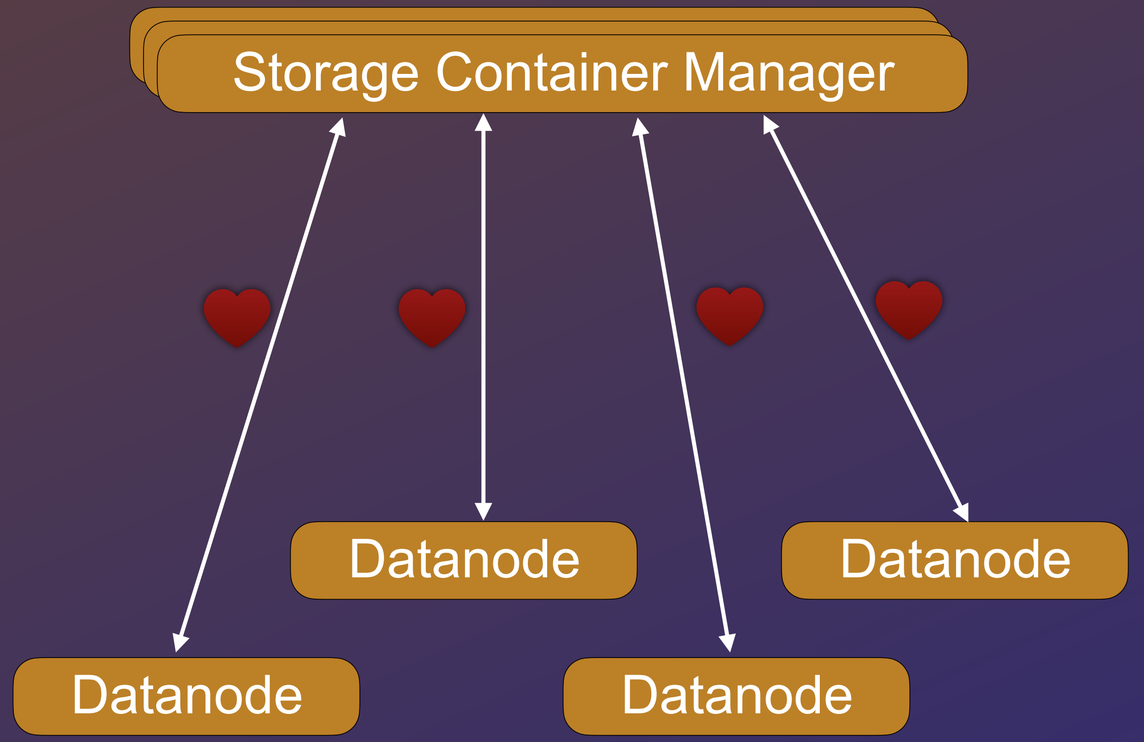
SCM + Datanodes

- Datanodes periodically heartbeat to all SCMs
- Leader SCM makes decisions based on datanode health.
- Leader SCM responds to heartbeats with commands for data nodes
  - Replicate data, delete blocks, etc.



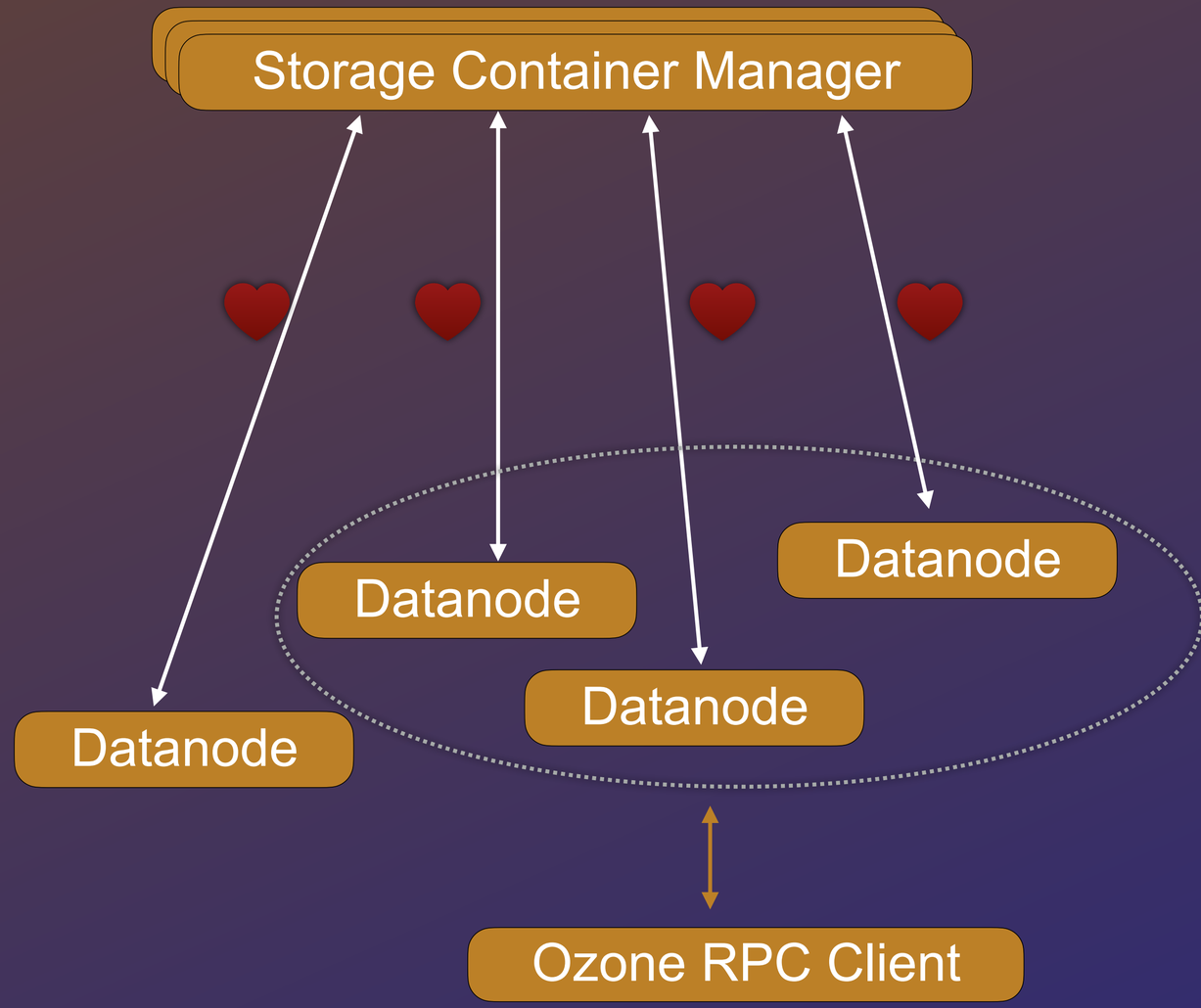
SCM + Datanodes

- Datanodes periodically heartbeat to all SCMs
- Leader SCM makes decisions based on datanode health.
- Leader SCM responds to heartbeats with commands for data nodes
  - Replicate data, delete blocks, etc.



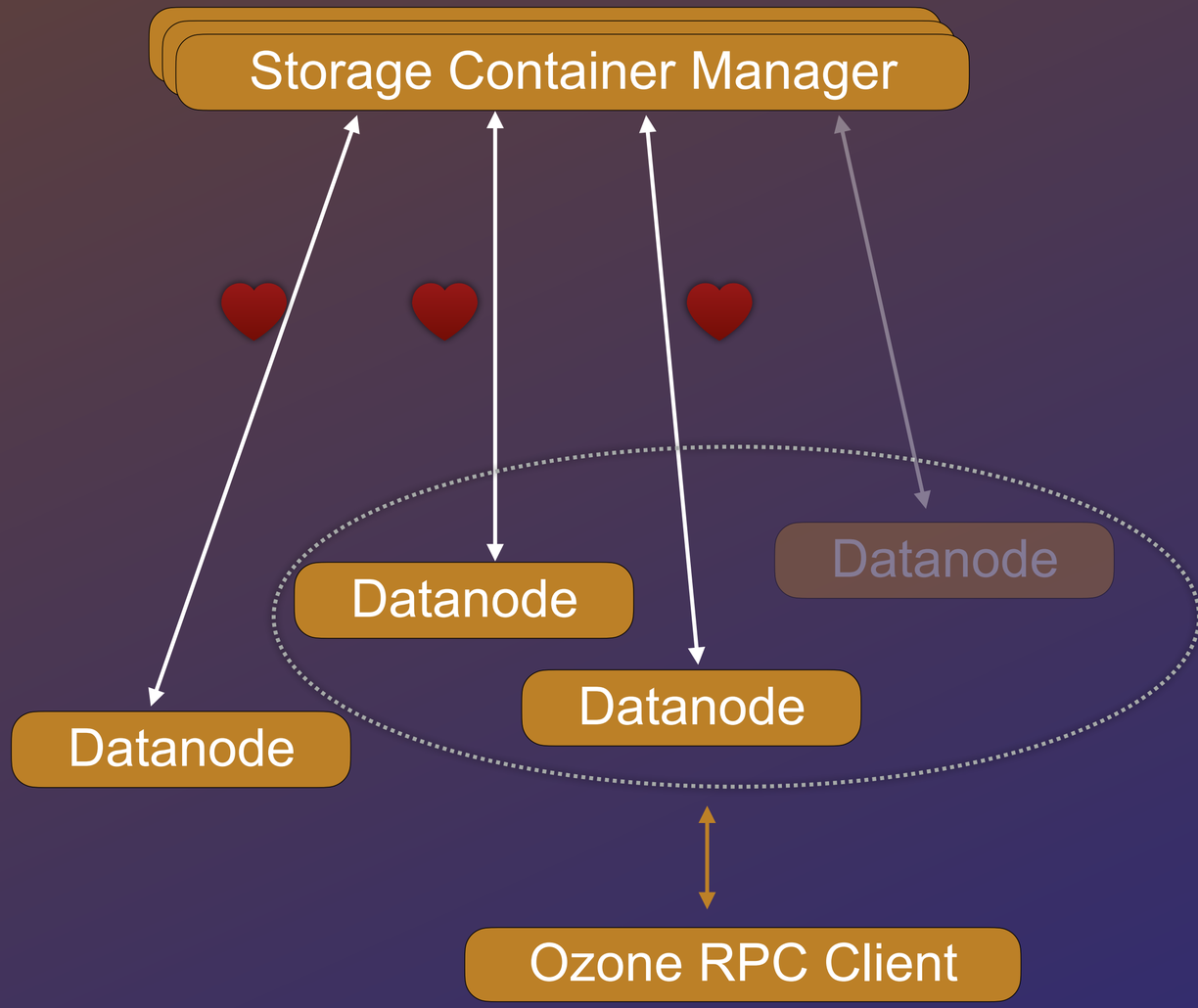
SCM + Datanodes + Clients: Network Failure

- When a datanode misses a heartbeat to SCM, SCM marks it as **stale.**



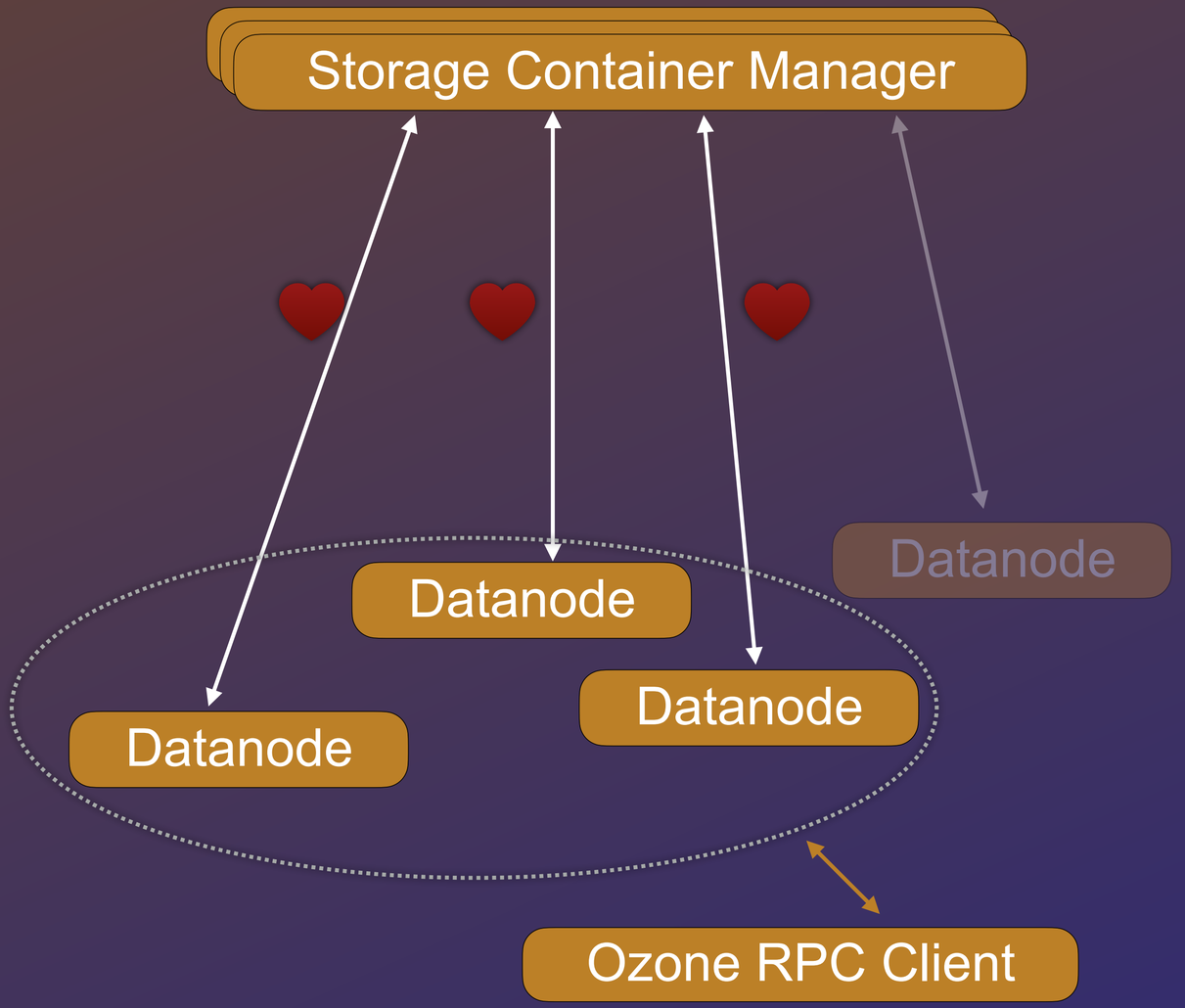
SCM + Datanodes + Clients: Network Failure

- When a datanode misses a heartbeat to SCM, SCM marks it as **stale.**



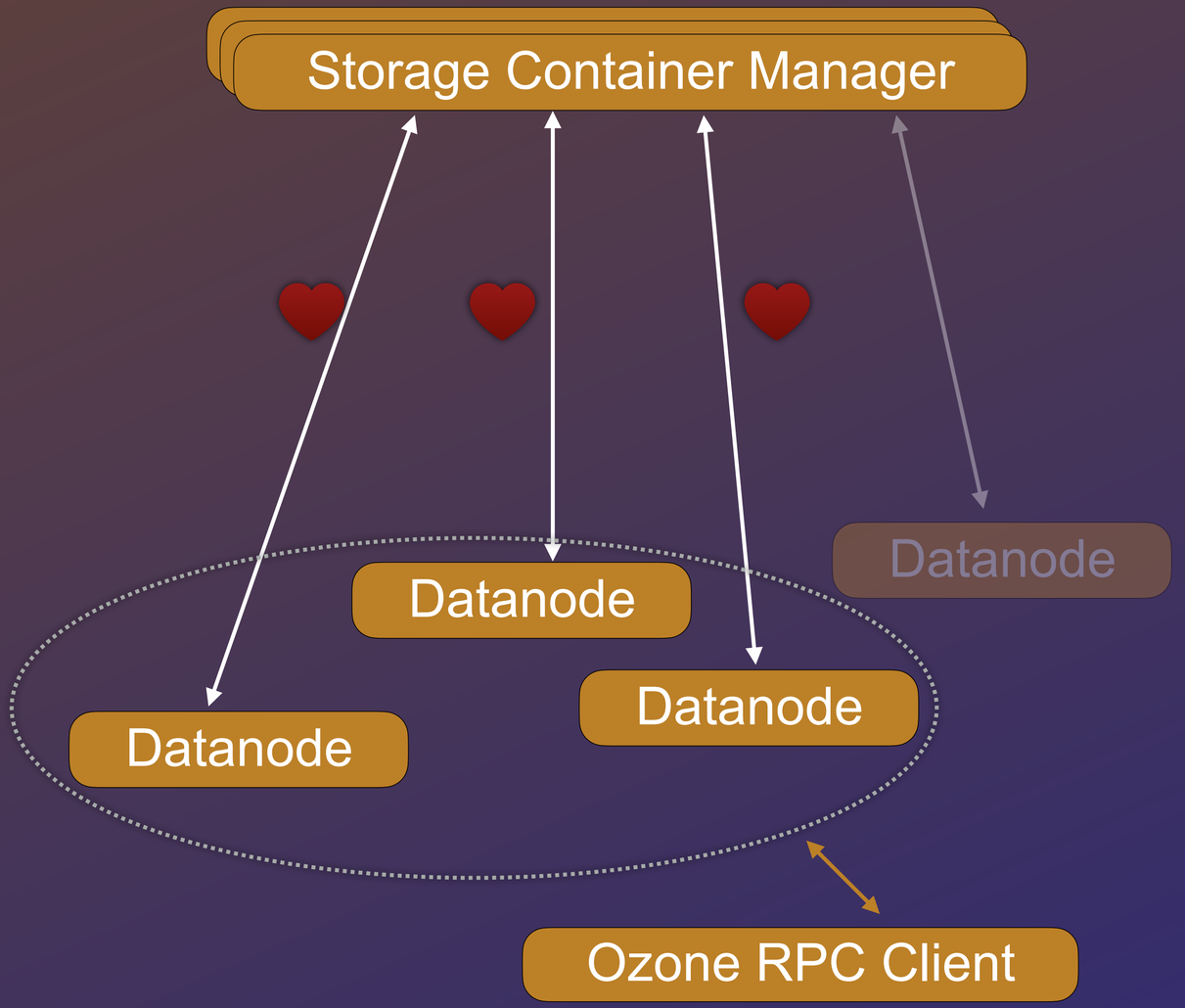
SCM + Datanodes + Clients: Network Failure

- When a datanode misses a heartbeat to SCM, SCM marks it as **stale.**
- Stop new writes to this datanode



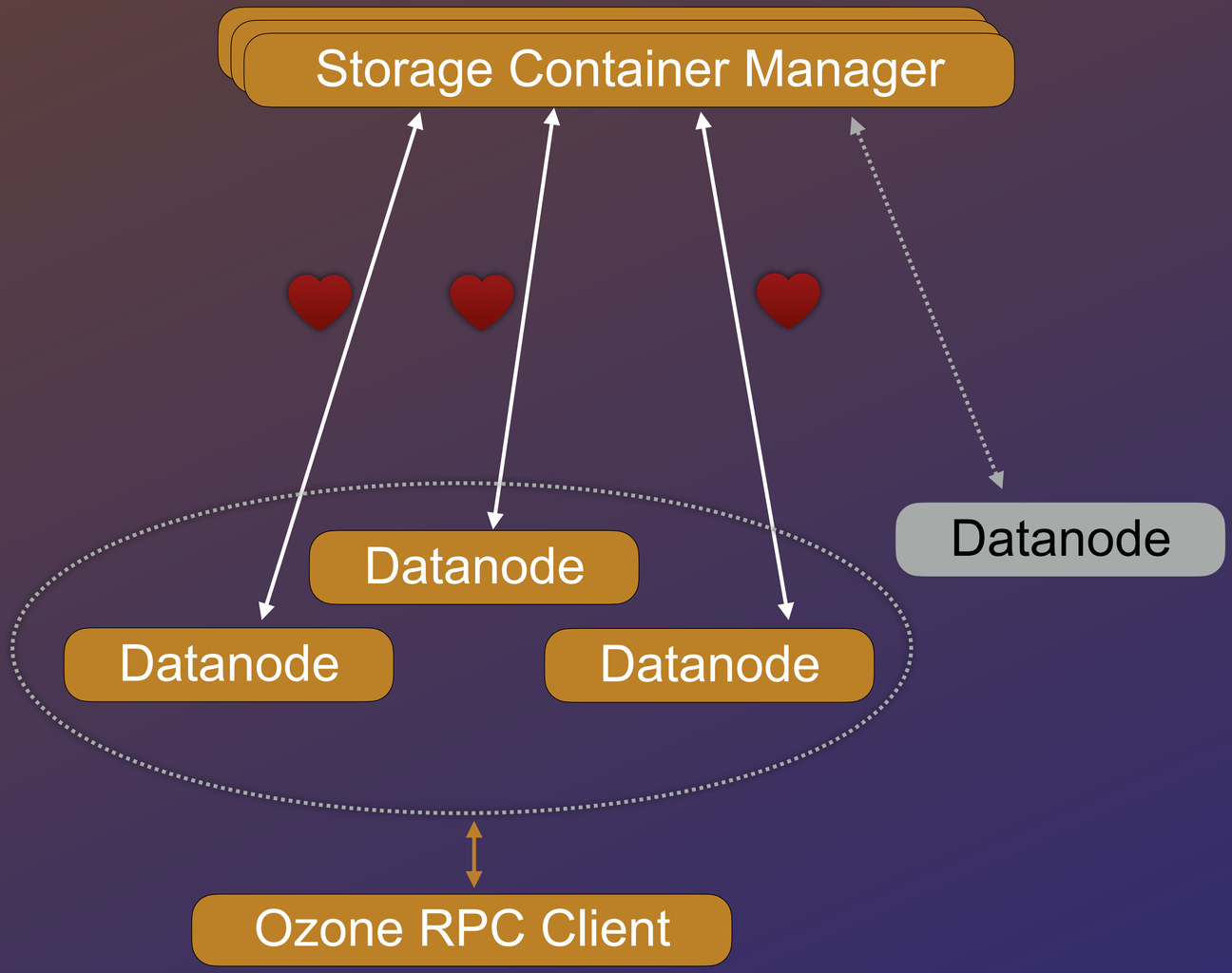
SCM + Datanodes + Clients: Network Failure

- When a datanode misses a heartbeat to SCM, SCM marks it as **stale.**
- Stop new writes to this datanode
- Data replication is not yet triggered



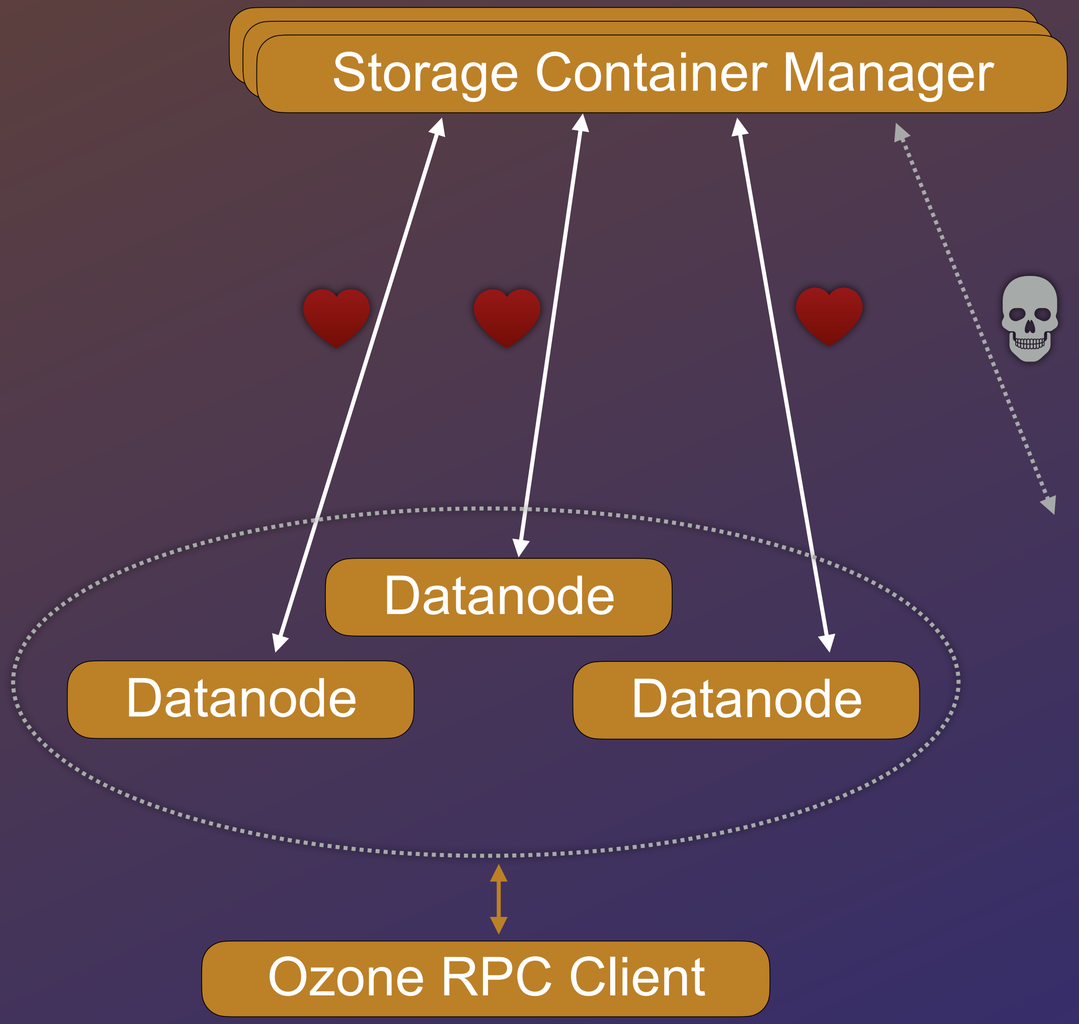
SCM + Datanodes: Node Failure

- When a stale datanode misses further heartbeats, SCM marks it as **dead**.



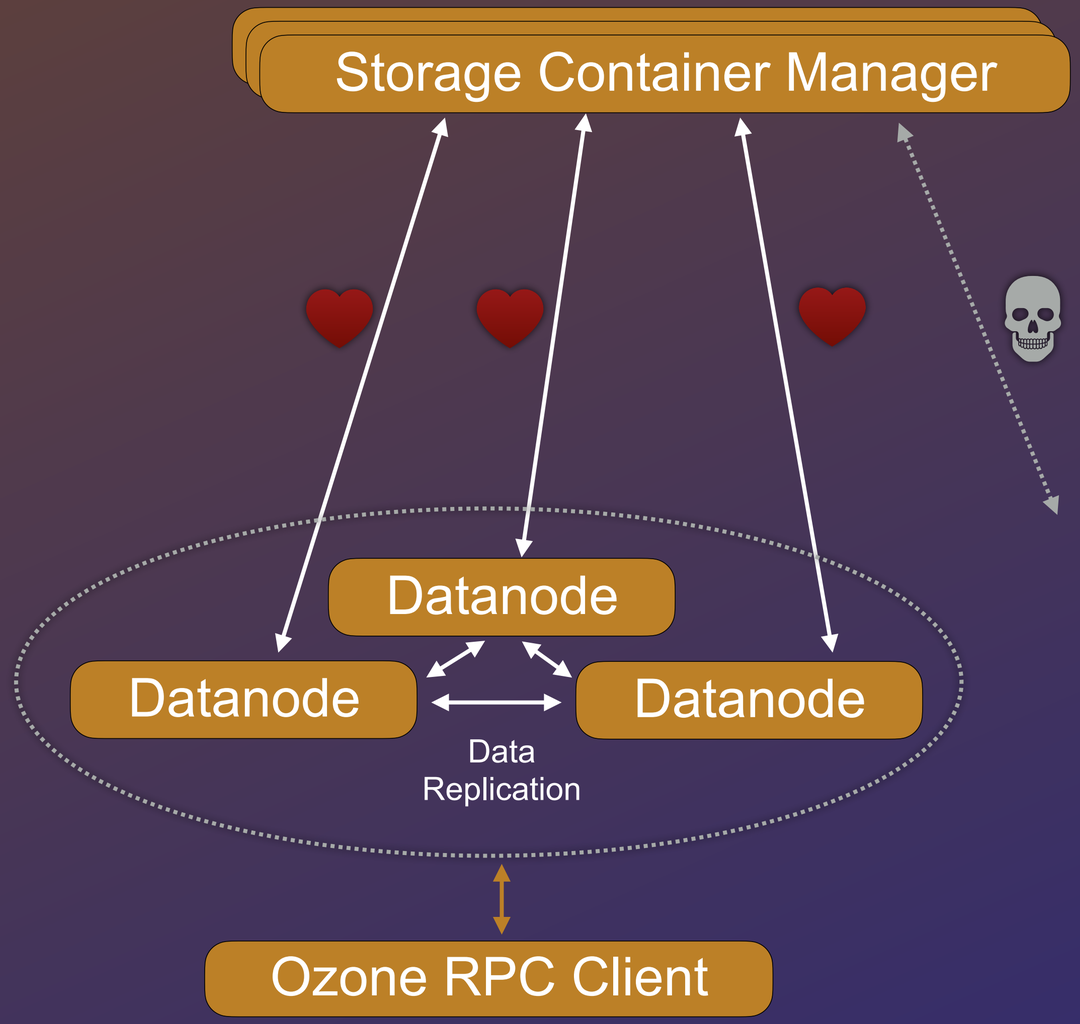
SCM + Datanodes: Node Failure

- When a stale datanode misses further heartbeats, SCM marks it as **dead**.



SCM + Datanodes: Node Failure

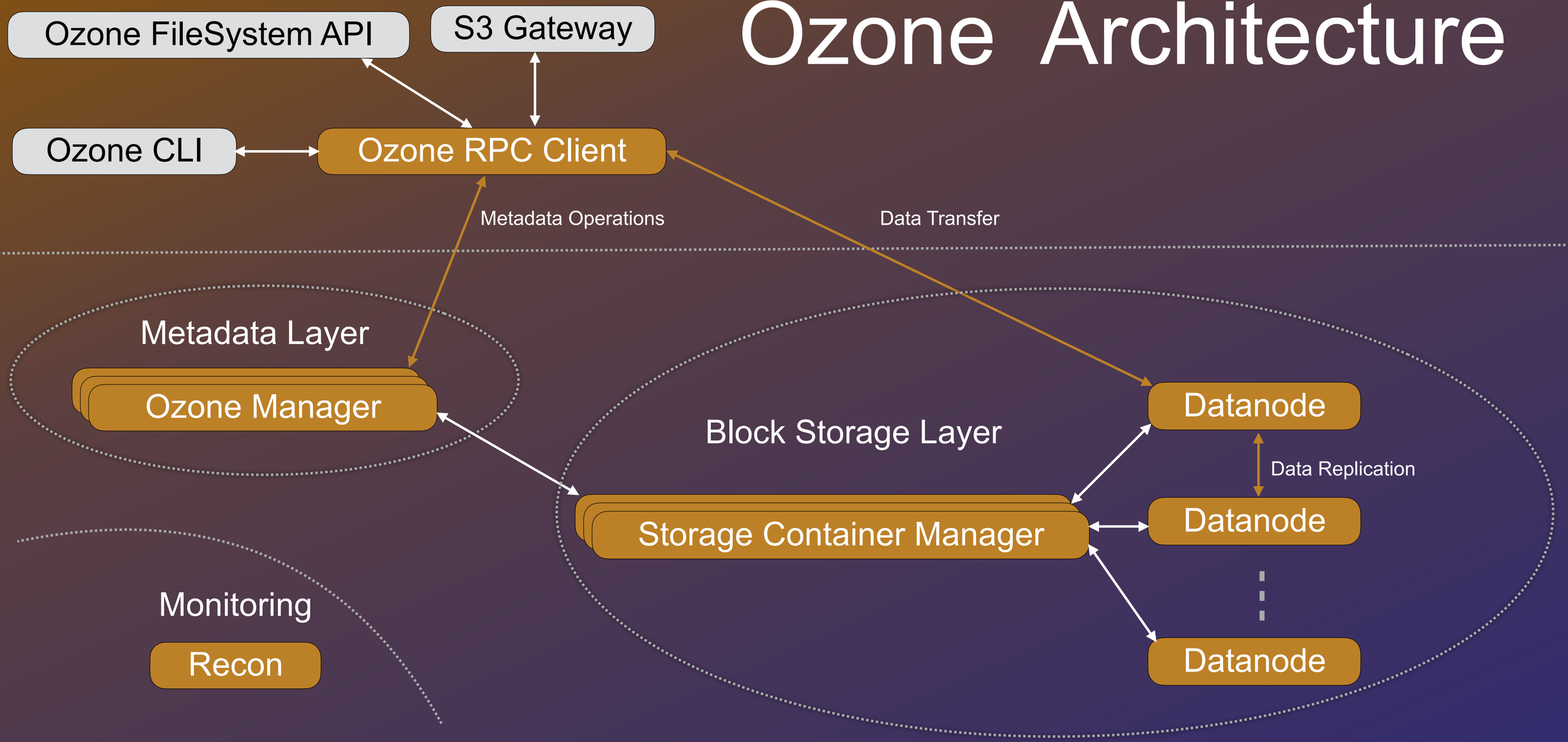
- When a stale datanode misses further heartbeats, SCM marks it as **dead**.
- Trigger replication/reconstruction from other nodes.



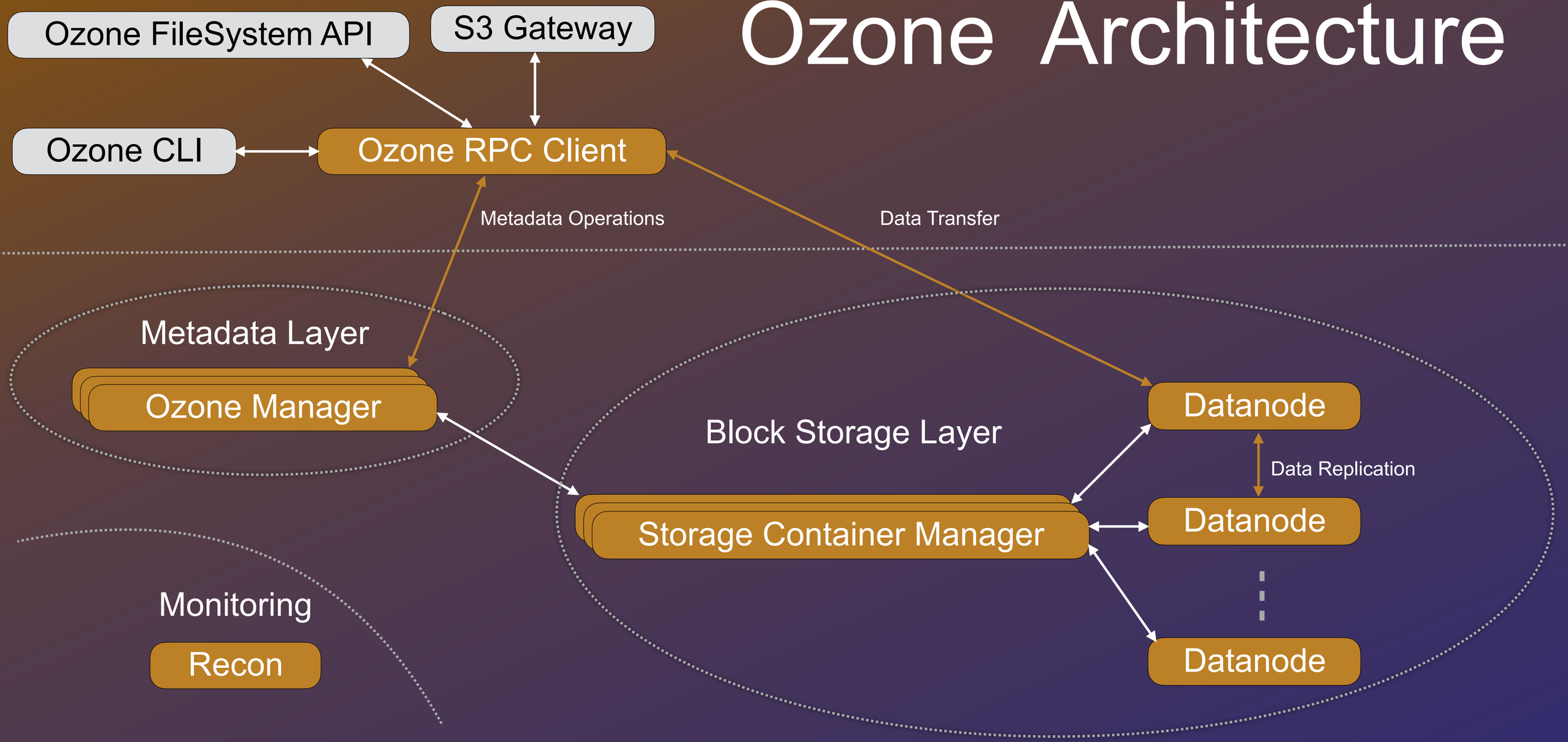
SCM to Datanode Communication

`hdds.heartbeat.interval`	Interval between datanode to SCM heartbeats
`ozone.scm.stale.node.interval`	Timeout for SCM to mark a datanode stale if no heartbeat is received
`ozone.scm.dead.node.interval`	Timeout for SCM to mark a datanode dead if no heartbeat is received

Ozone Architecture



Ozone Architecture



Datanodes

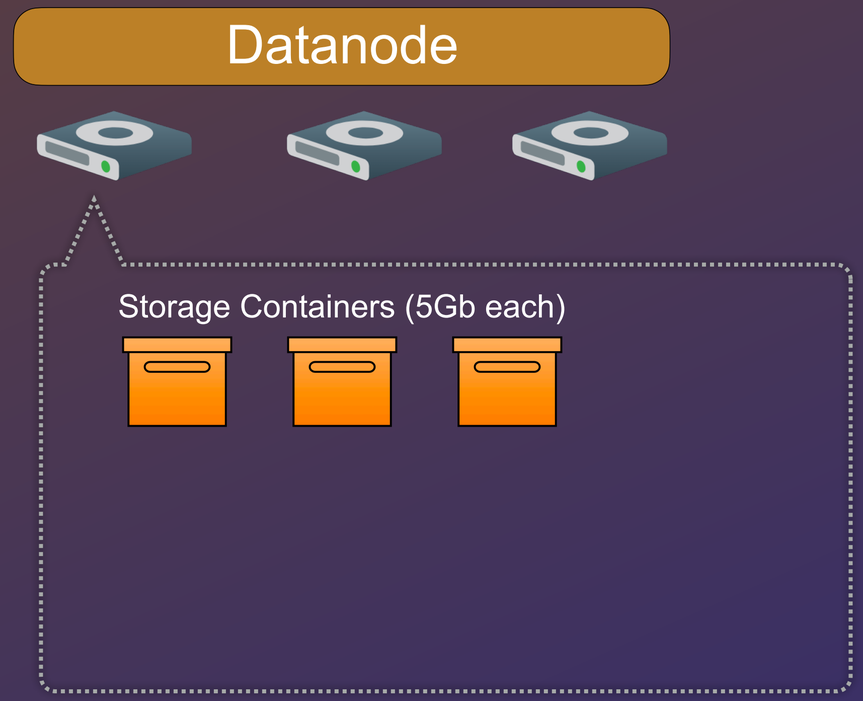
Datanode

Datanodes

- Comprised of multiple **volumes** (disks) that may fail independently

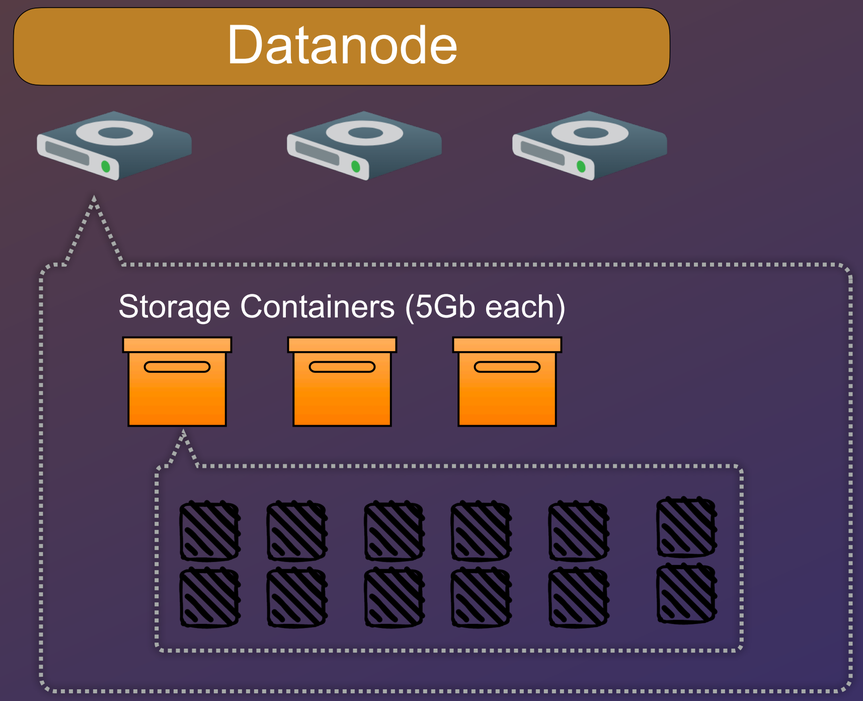
Datanodes

- Comprised of multiple **volumes** (disks) that may fail independently
- Each volume holds **storage containers**



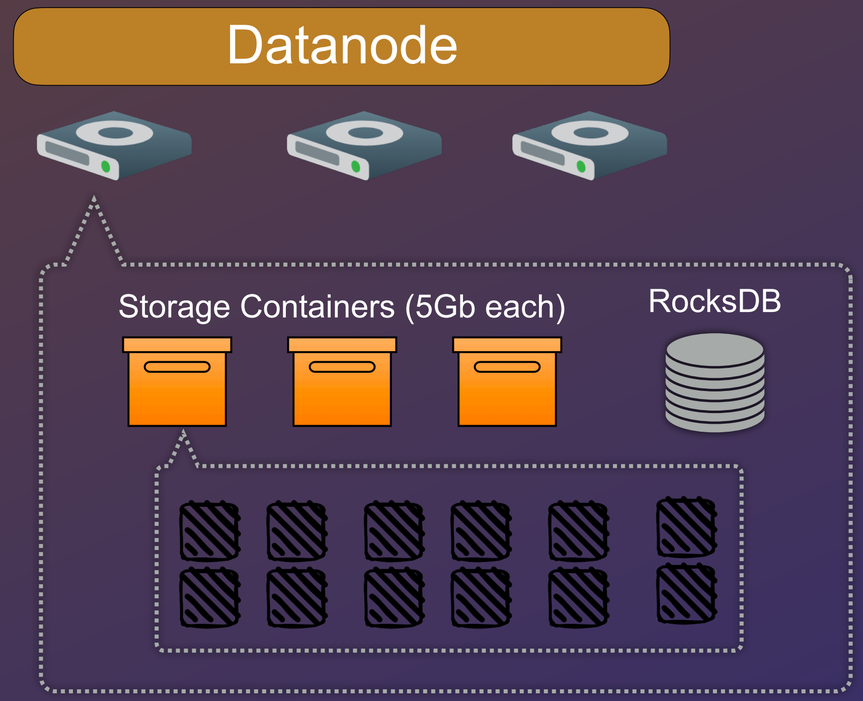
Datanodes

- Comprised of multiple **volumes** (disks) that may fail independently
- Each volume holds **storage containers**
- Each storage container is a 5gb collection of blocks

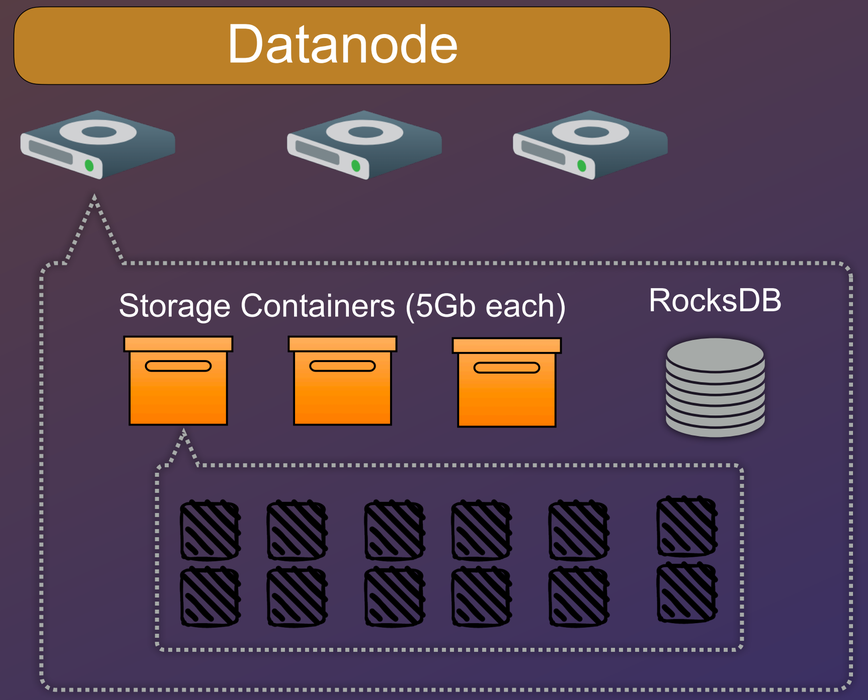


Datanodes

- Comprised of multiple **volumes** (disks) that may fail independently
- Each volume holds **storage containers**
- Each storage container is a 5gb collection of blocks
- RocksDB per volume holds metadata

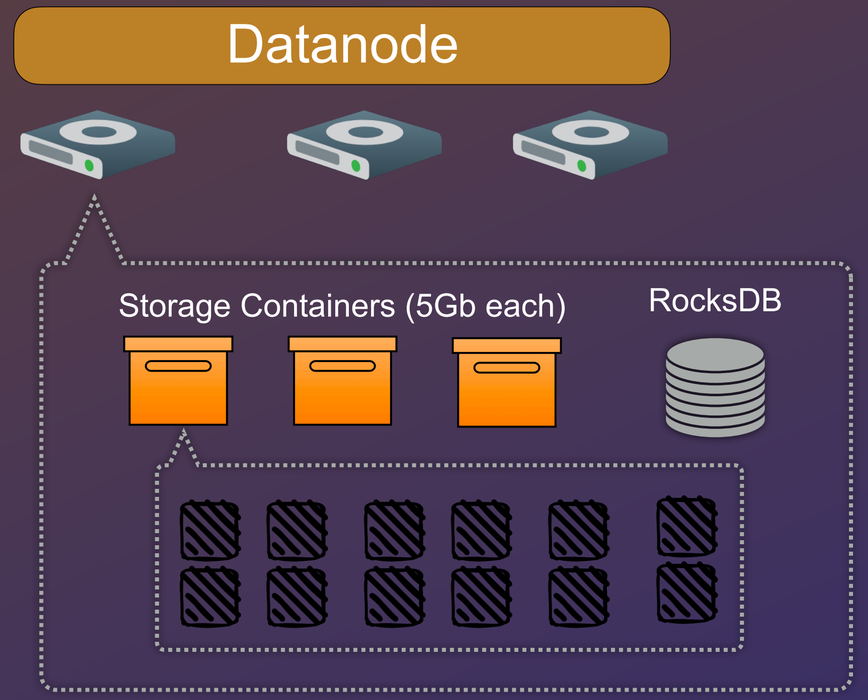


Datanode: Disk Failures



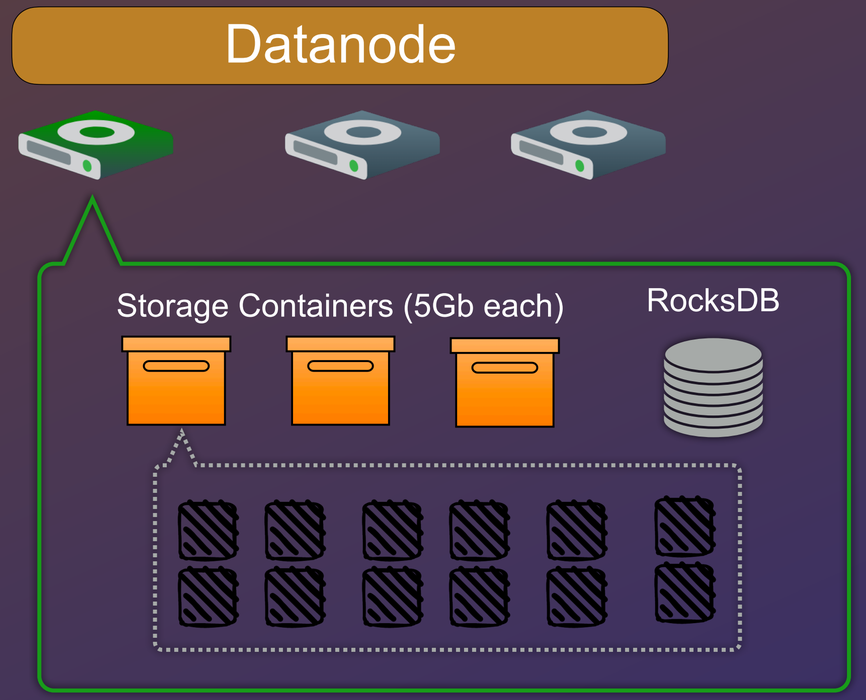
Datanode: Disk Failures

- **Volume Scanner:** Proactively identify bad disks



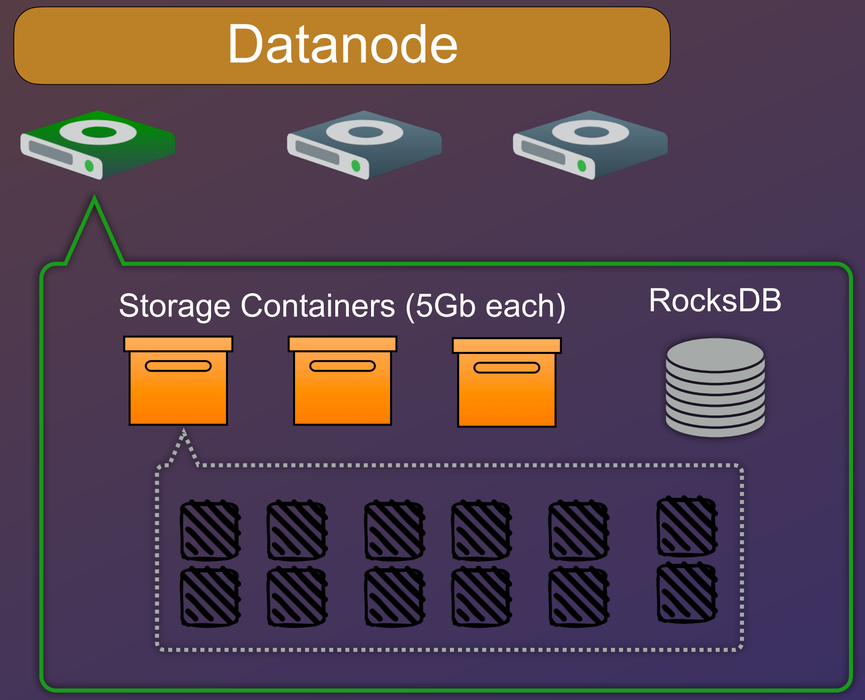
Datanode: Disk Failures

- **Volume Scanner:** Proactively identify bad disks
- Assess disk health, not disk contents (fast)



Datanode: Disk Failures

- **Volume Scanner:** Proactively identify bad disks
- Assess disk health, not disk contents (fast)
- Replication can be triggered from other sources



Volume Scanner: Directory Checks

- **Goal:** Identify configuration issues

Volume Scanner: Directory Checks

- **Goal:** Identify configuration issues
  - Disk mount point exists

Volume Scanner: Directory Checks

- **Goal:** Identify configuration issues
  - Disk mount point exists
  - Disk mount point has correct permissions

Volume Scanner: Disk Checks

- **Goal:** Touch disk hardware after the mount point is verified

Volume Scanner: Disk Checks

- **Goal:** Touch disk hardware after the mount point is verified

1. Save random byte string in memory `110110101…`

Volume Scanner: Disk Checks

- **Goal:** Touch disk hardware after the mount point is verified

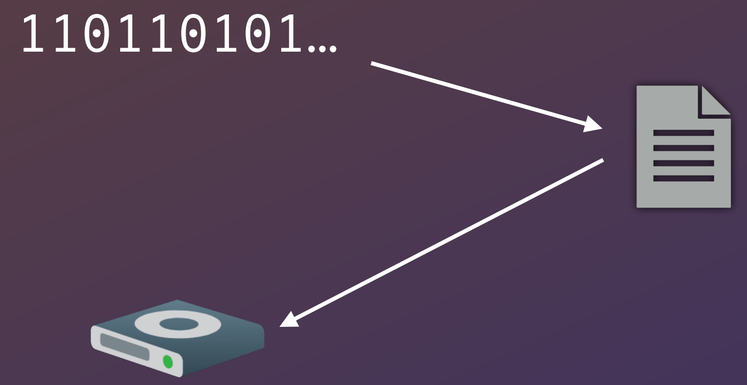
1. Save random byte string in memory

2. Write to tmp file

Volume Scanner: Disk Checks

- **Goal:** Touch disk hardware after the mount point is verified

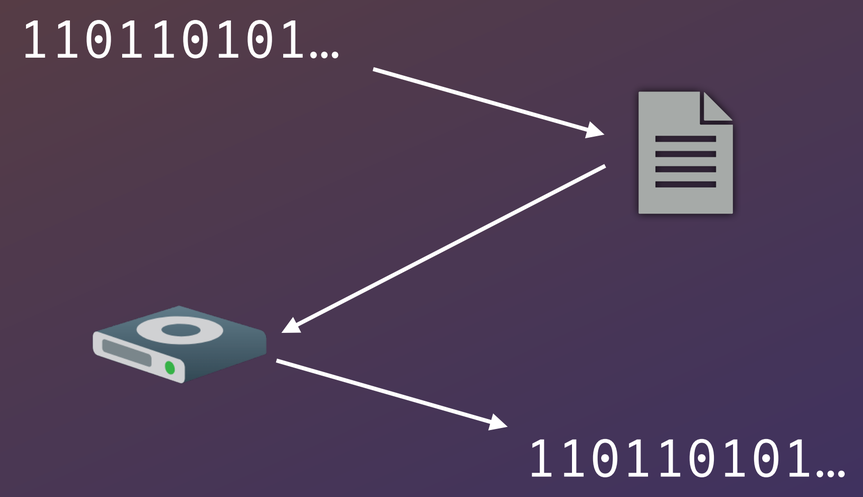
1. Save random byte string in memory
2. Write to tmp file
3. Sync file



Volume Scanner: Disk Checks

- **Goal:** Touch disk hardware after the mount point is verified

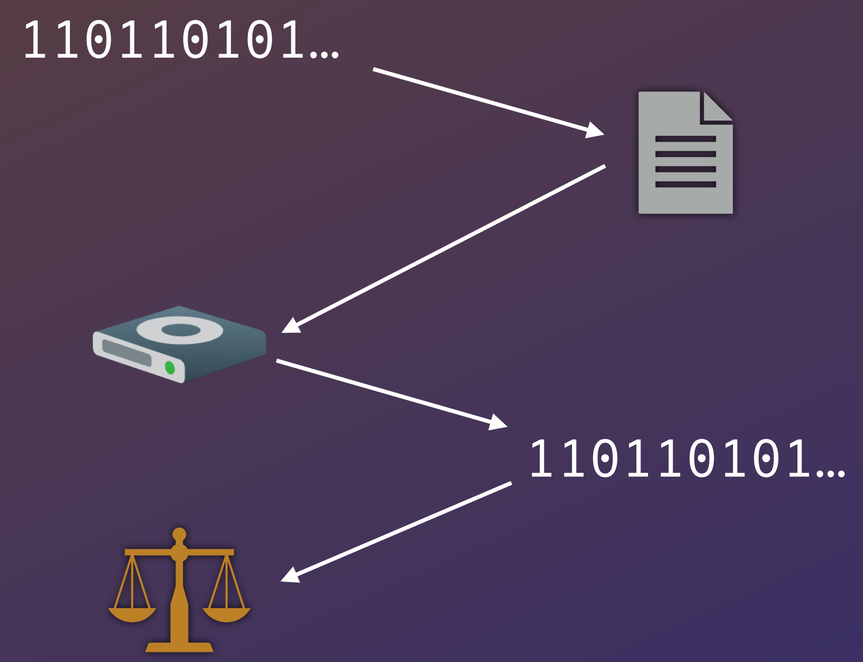
1. Save random byte string in memory
2. Write to tmp file
3. Sync file
4. Read back from file



Volume Scanner: Disk Checks

- **Goal:** Touch disk hardware after the mount point is verified

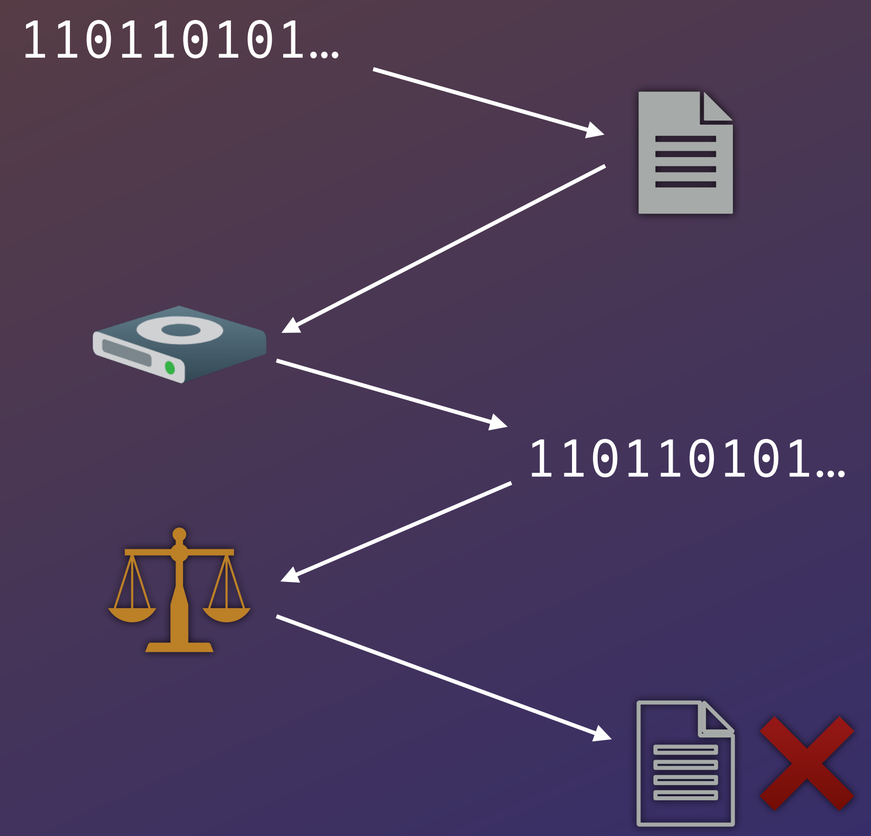
1. Save random byte string in memory
2. Write to tmp file
3. Sync file
4. Read back from file
5. Compare with saved byte string



Volume Scanner: Disk Checks

- **Goal:** Touch disk hardware after the mount point is verified

1. Save random byte string in memory
2. Write to tmp file
3. Sync file
4. Read back from file
5. Compare with saved byte string
6. Delete file



Volume Scanner: Disk Checks

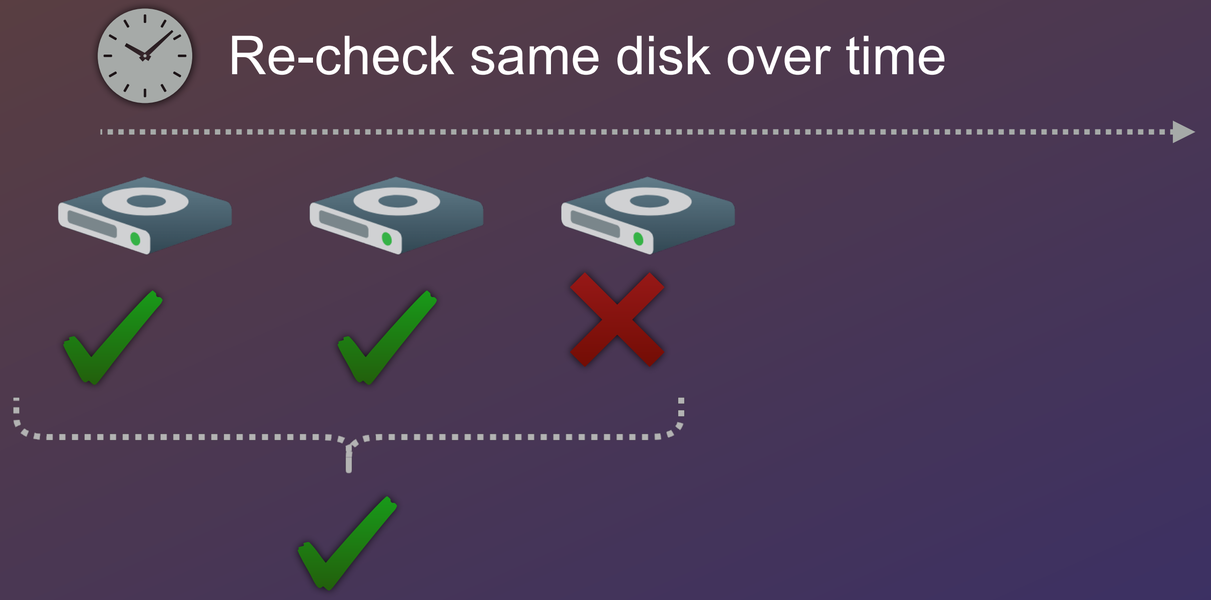
`hdds.datanode.disk.check.io.file.size`	How large of a file to use in the disk I/O check
`hdds.datanode.disk.check.timeout`	Maximum time a disk scan can take before it is considered failed

Volume Scanner: Iterating Disk Checks

- Don’t mistake intermittent IO errors for full disk failure
- **Disk check sliding window**: 2 of the last 3 disk checks must pass

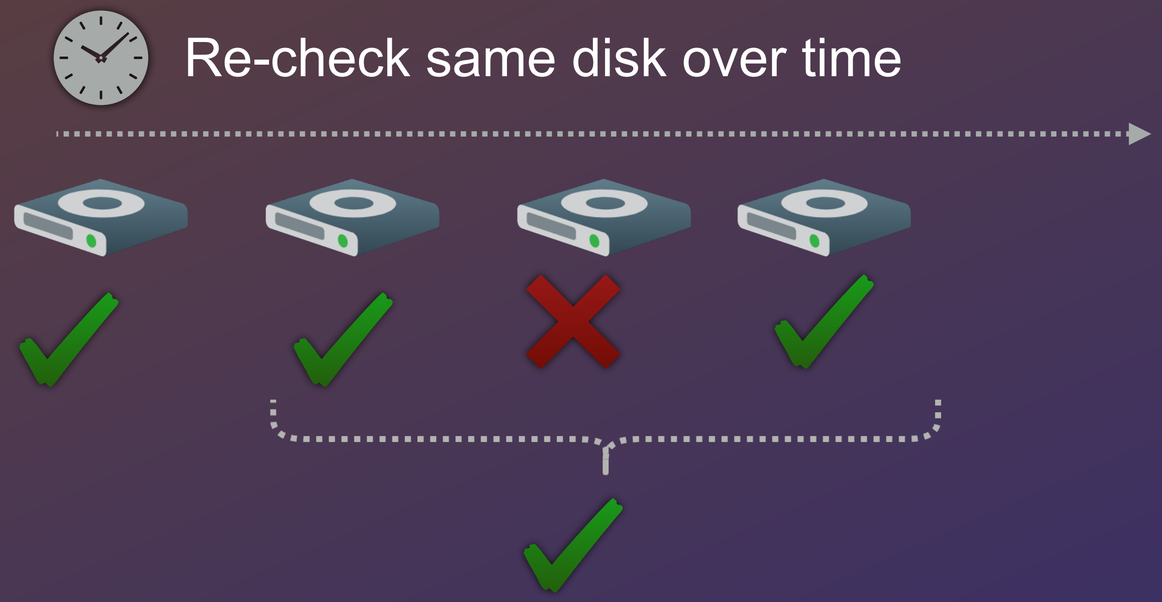
Volume Scanner: Iterating Disk Checks

- Don’t mistake intermittent IO errors for full disk failure
- **Disk check sliding window**: 2 of the last 3 disk checks must pass



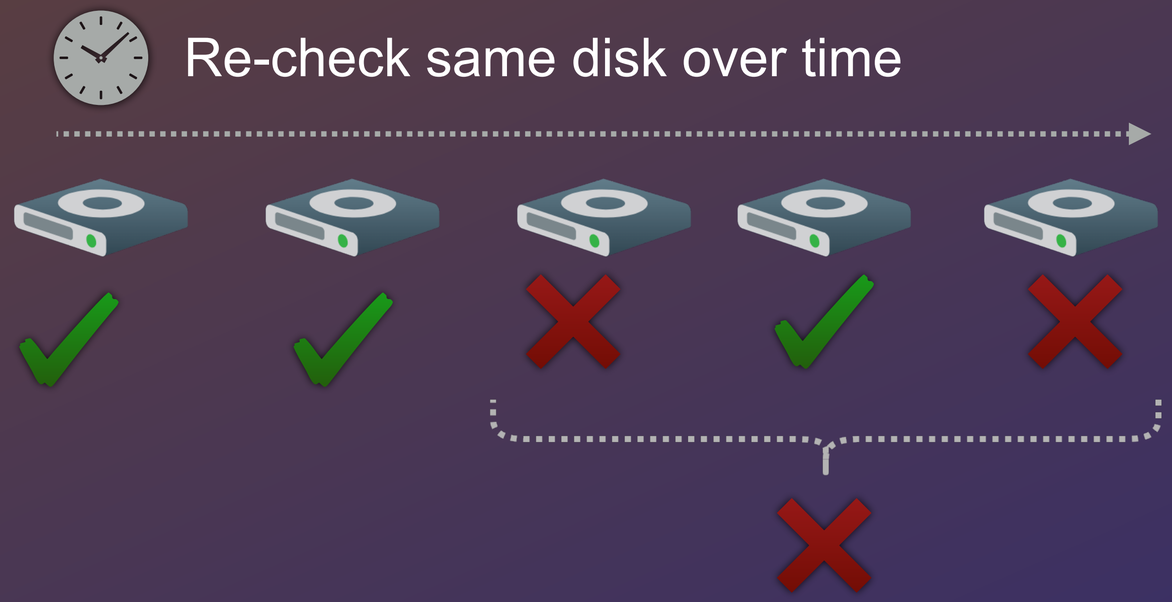
Volume Scanner: Iterating Disk Checks

- Don’t mistake intermittent IO errors for full disk failure
- **Disk check sliding window**: 2 of the last 3 disk checks must pass



Volume Scanner: Iterating Disk Checks

- Don’t mistake intermittent IO errors for full disk failure
- **Disk check sliding window**: 2 of the last 3 disk checks must pass

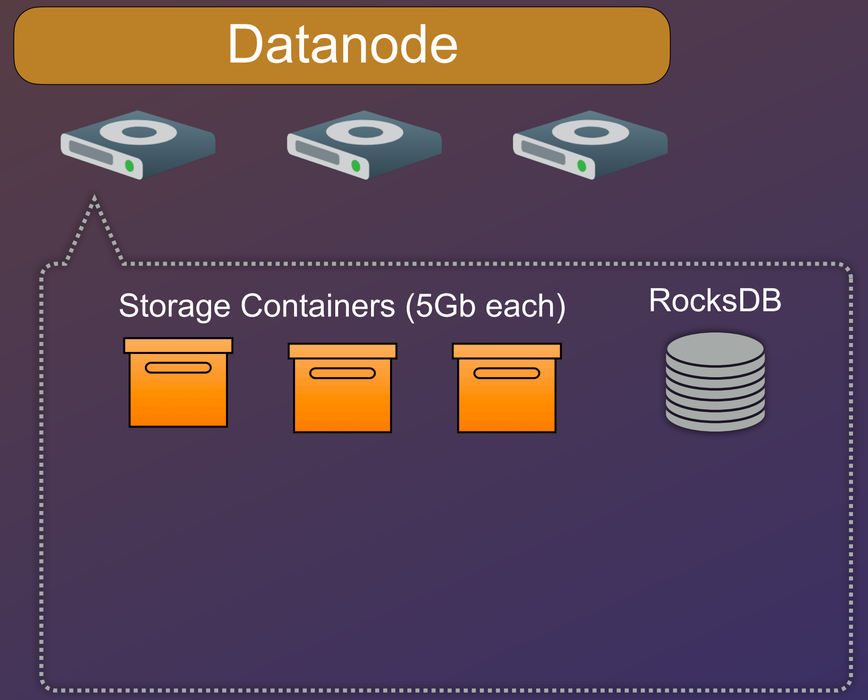


Volume Scanner: Iterating Disk Checks

`hdds.datanode.disk.check.io.test.count`	Size of disk check sliding window
`hdds.datanode.disk.check.io.failures.tolerated`	Number of checks in the window that can fail without volume being failed
`hdds.datanode.periodic.disk.check.interval.minutes`	How frequently the background disk checker runs

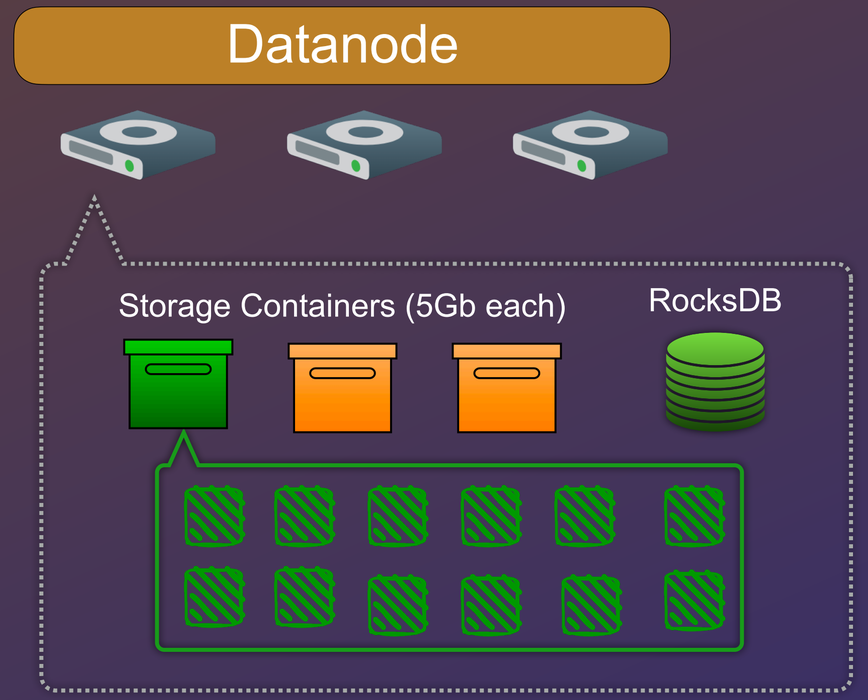
Datanode: Corruption

- **Container Scanner:** Proactively identify corrupted containers



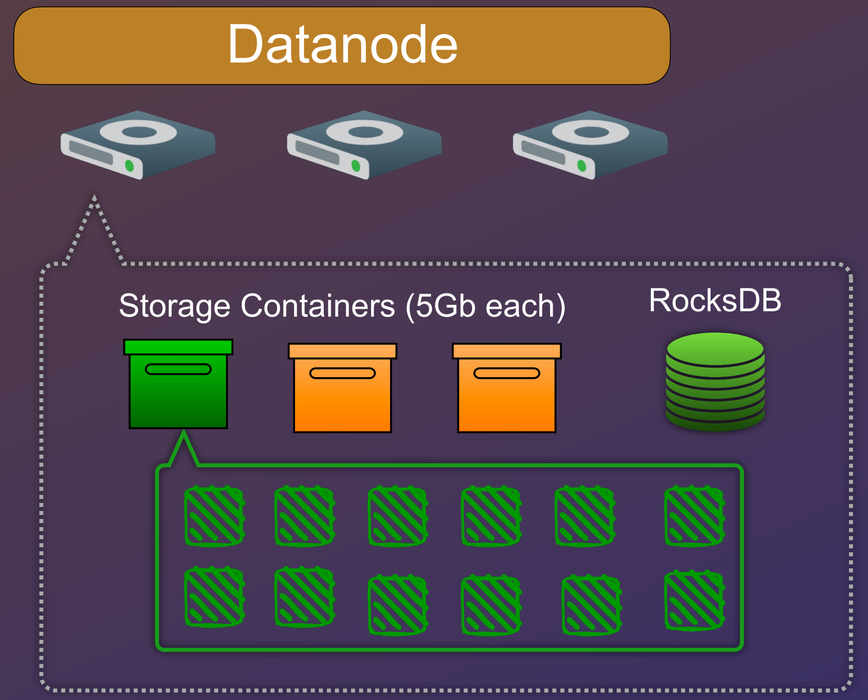
Datanode: Corruption

- **Container Scanner:** Proactively identify corrupted containers
- Assess disk contents, not disk health



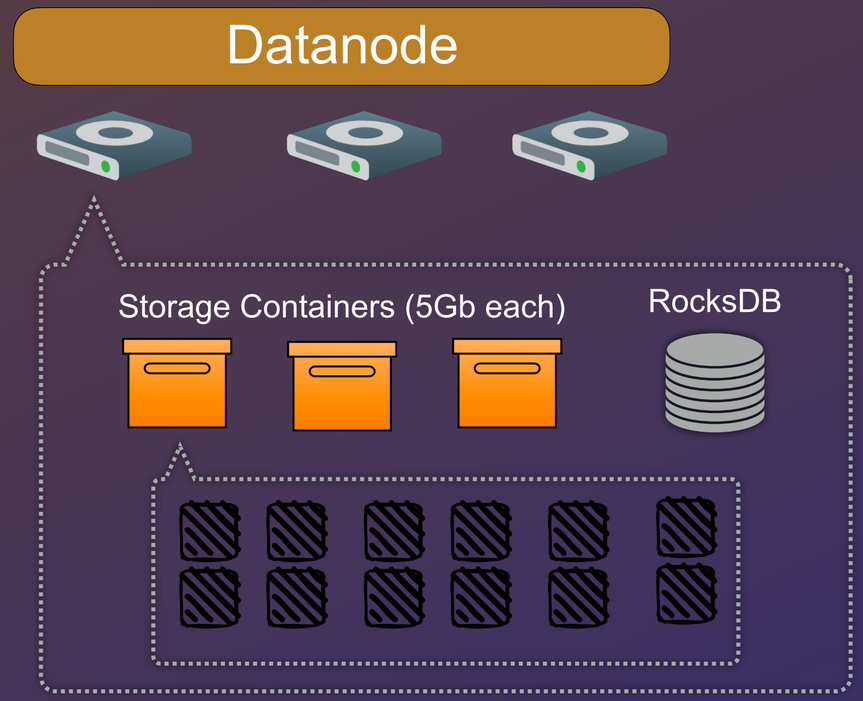
Datanode: Corruption

- **Container Scanner:** Proactively identify corrupted containers
- Assess disk contents, not disk health
- Replication can be triggered from other sources



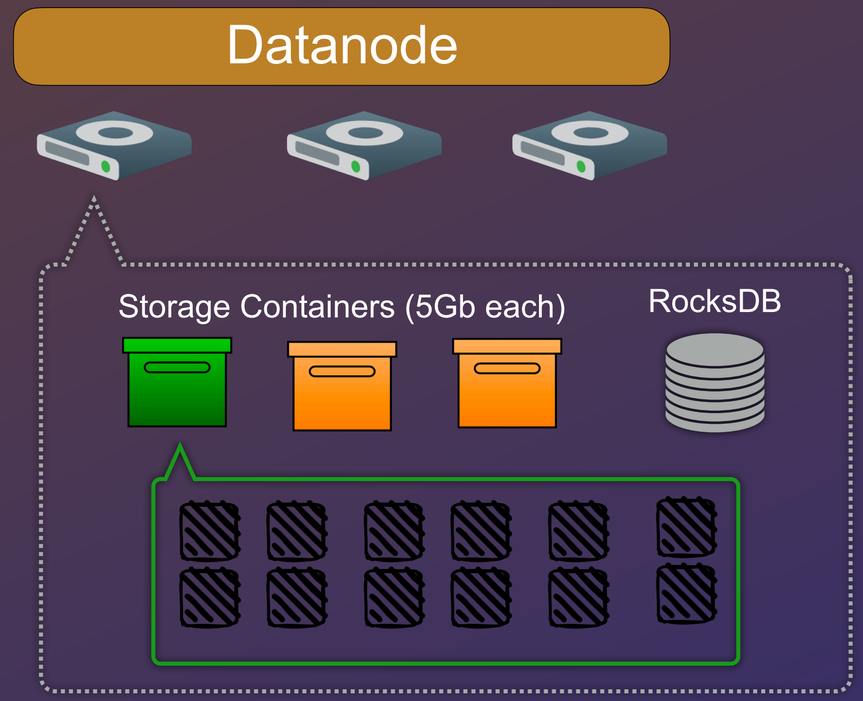
Container Scanner: Metadata Checks

- Fast check, can be done more frequently



Container Scanner: Metadata Checks

- Fast check, can be done more frequently
- Is the container's directory and metadata still present and readable?

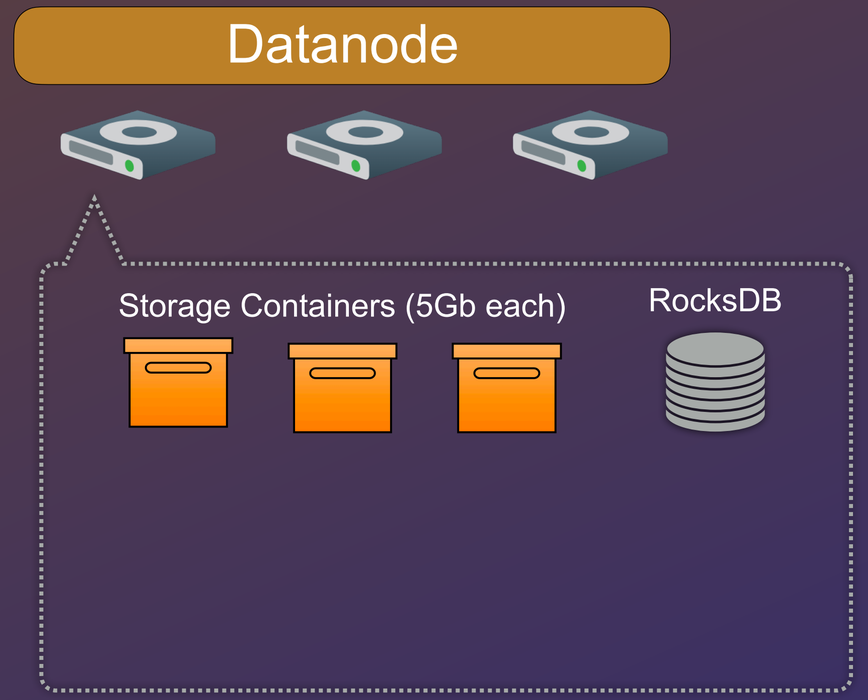


Container Scanner: Metadata Checks

`hdds.container.scrub.metadata.scan.interval`	Frequency the background container metadata scanner runs

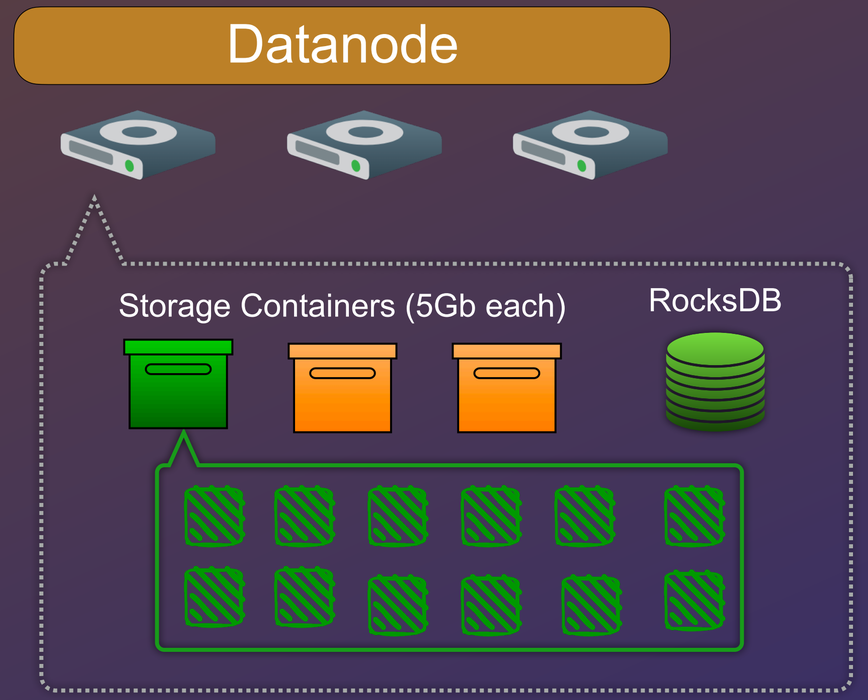
Container Scanner: Data Checks

- Slow check
  - Eventually touches every byte on disk
  - May run continuously in the background



Container Scanner: Data Checks

- Slow check
  - Eventually touches every byte on disk
  - May run continuously in the background
- Superset of metadata checks
  - Do block checksums match?
  - Does RocksDB metadata match blocks?



Container Scanner: Data Checks

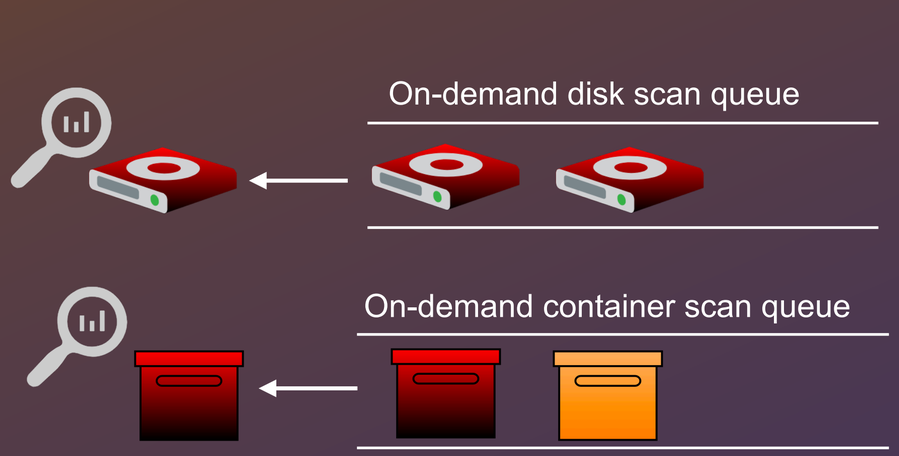
`hdds.container.scrub.data.scan.interval`	Frequency the background container data scanner runs
`hdds.container.scrub.volume.bytes.per.second`	Bandwidth throttle for background container data scanner

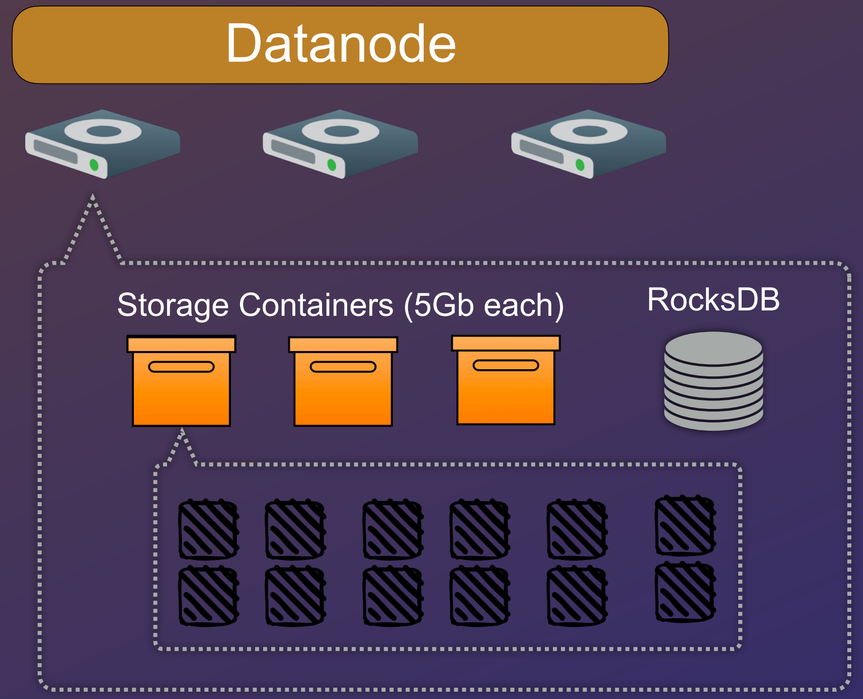
Container + Volume Scanner: Scheduling Scans

- **Background Scans:** Ensure data is intact even when not accessed.
- **On-Demand Scans:** Triggered when failure is encountered.
  - Background scans may take a while to get to this data otherwise.

Container + Volume Scanner: On-Demand Scans

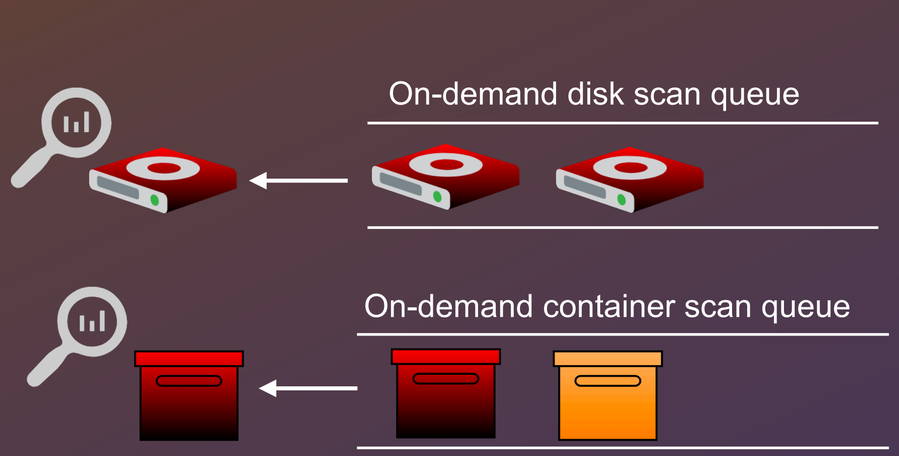
Ozone RPC Client

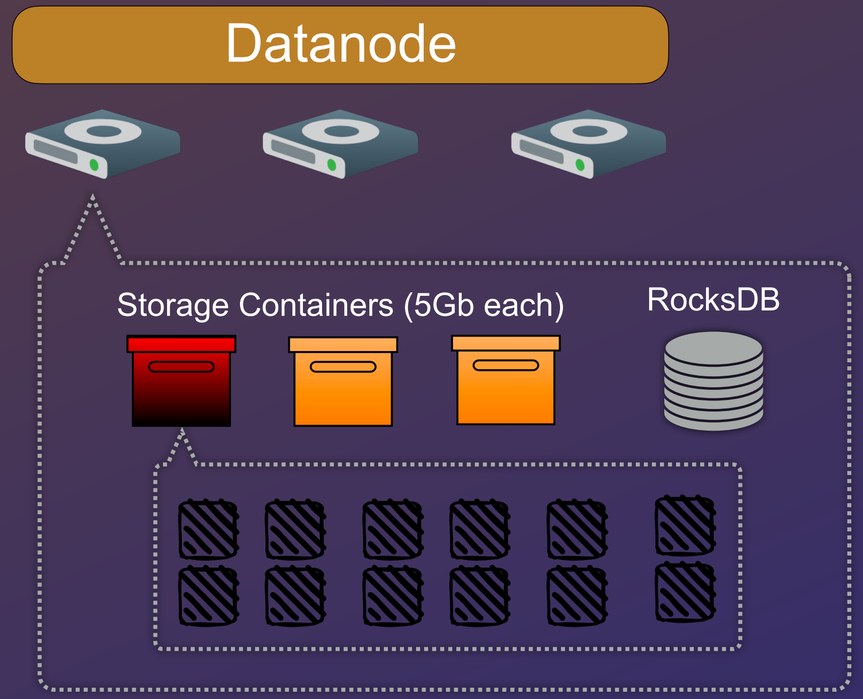




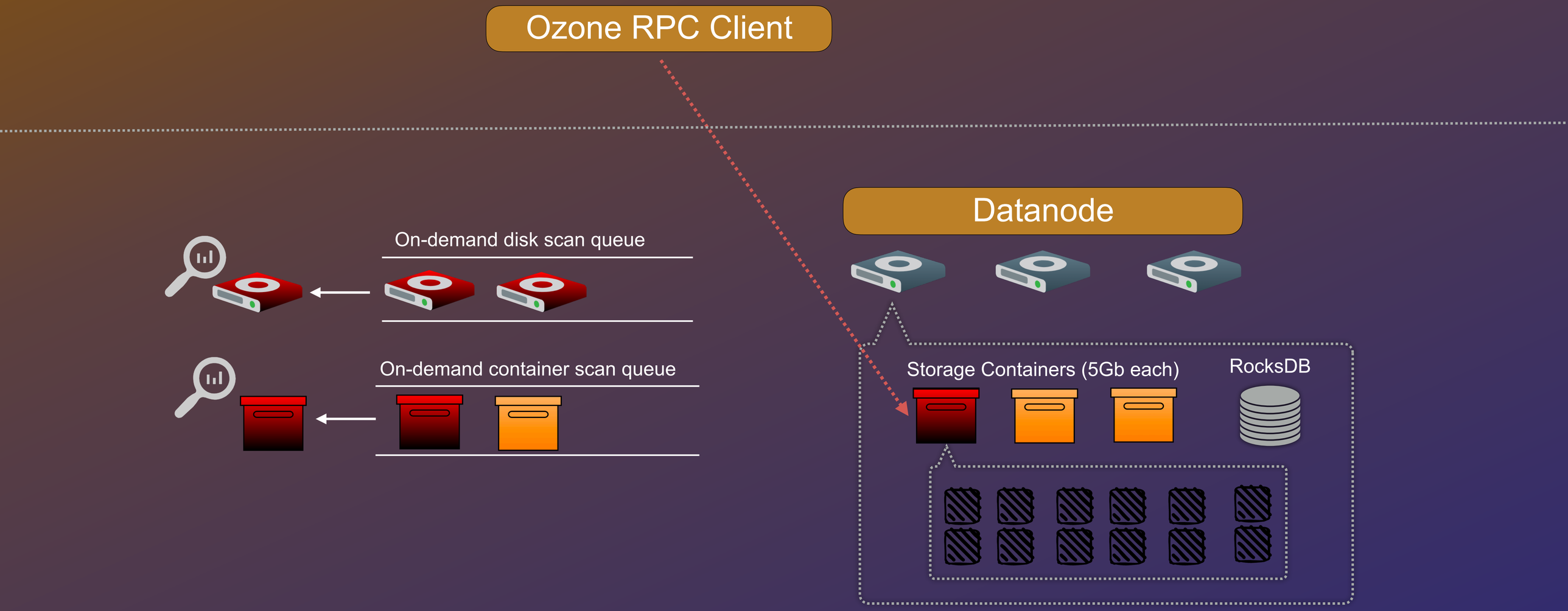
Container + Volume Scanner: On-Demand Scans

Ozone RPC Client



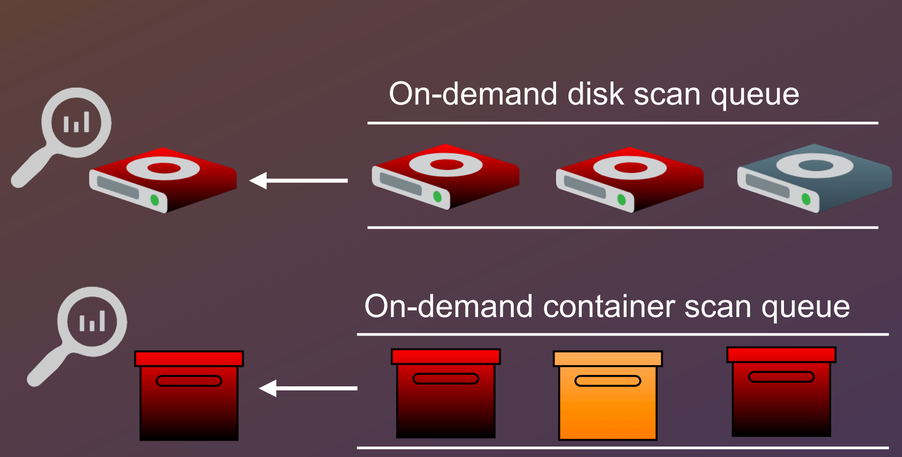


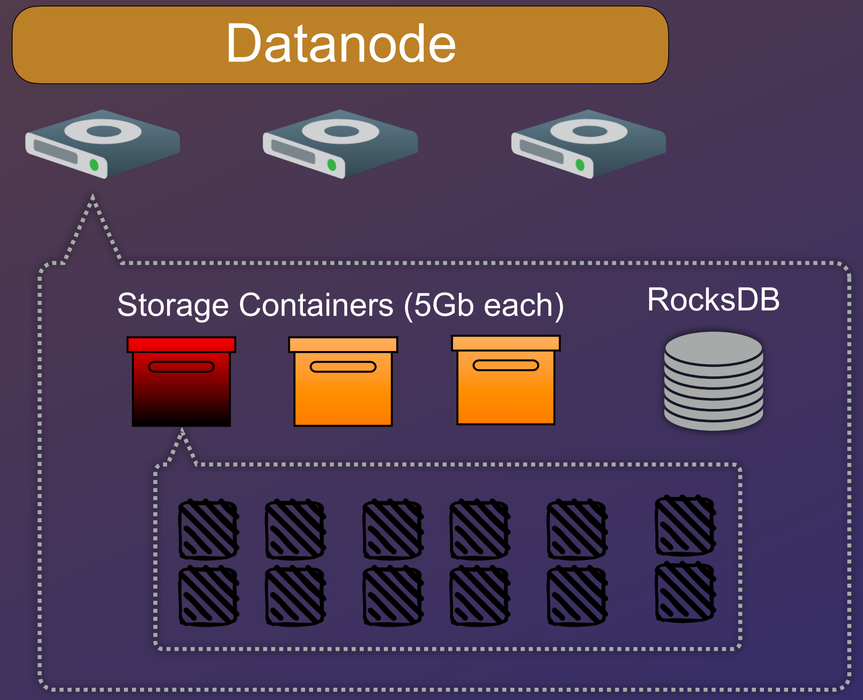
Container + Volume Scanner: On-Demand Scans



Container + Volume Scanner: On-Demand Scans

Ozone RPC Client

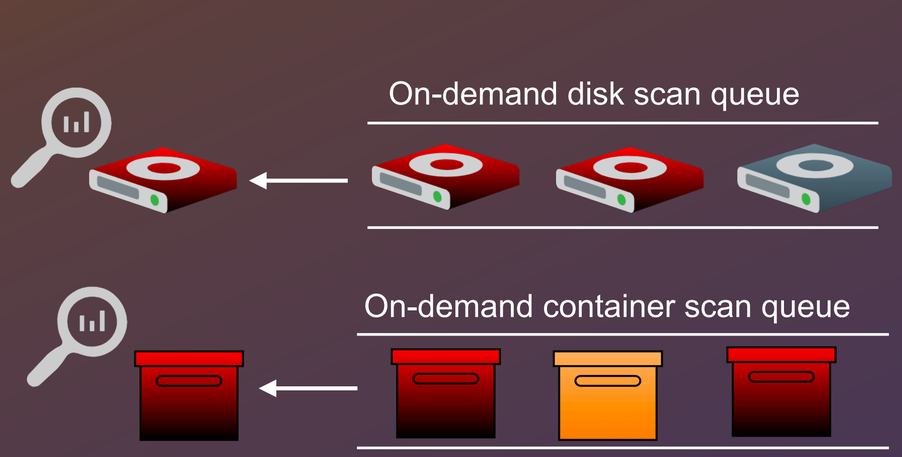


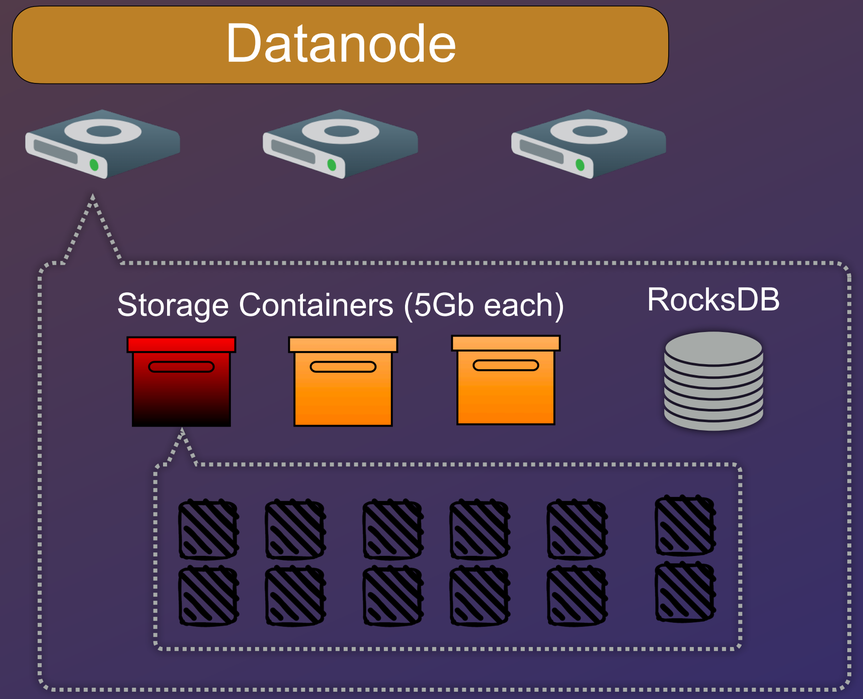


Container + Volume Scanner: On-Demand Scans

Client retries on a different datanode's replica

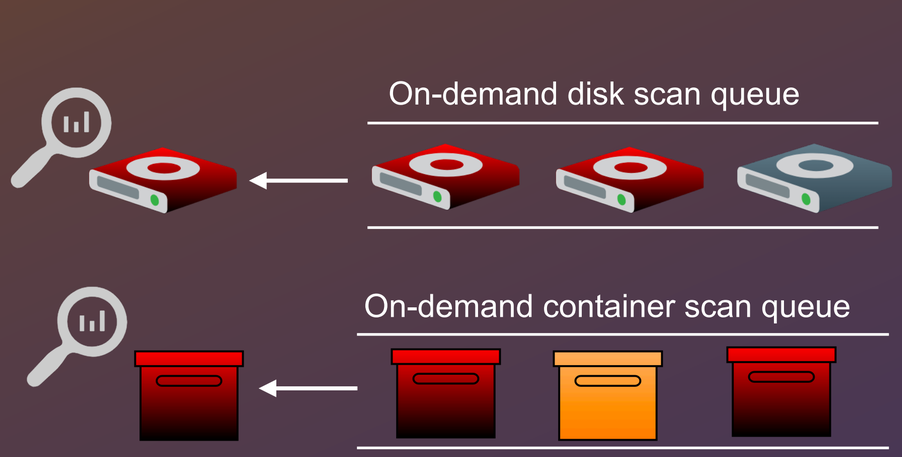
Ozone RPC Client

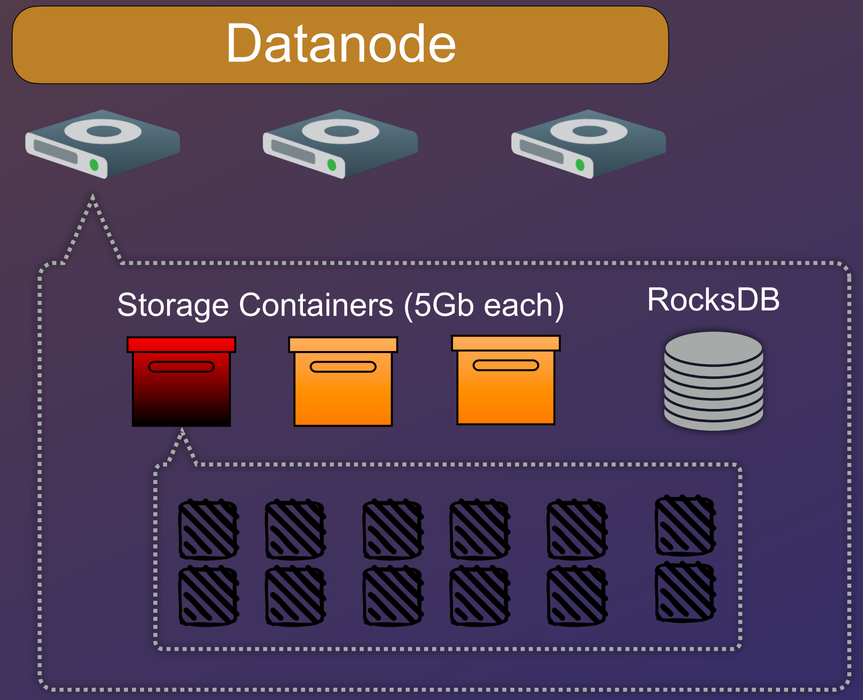




Container + Volume Scanner: On-Demand Scans

Client retries on a different datanode's replica

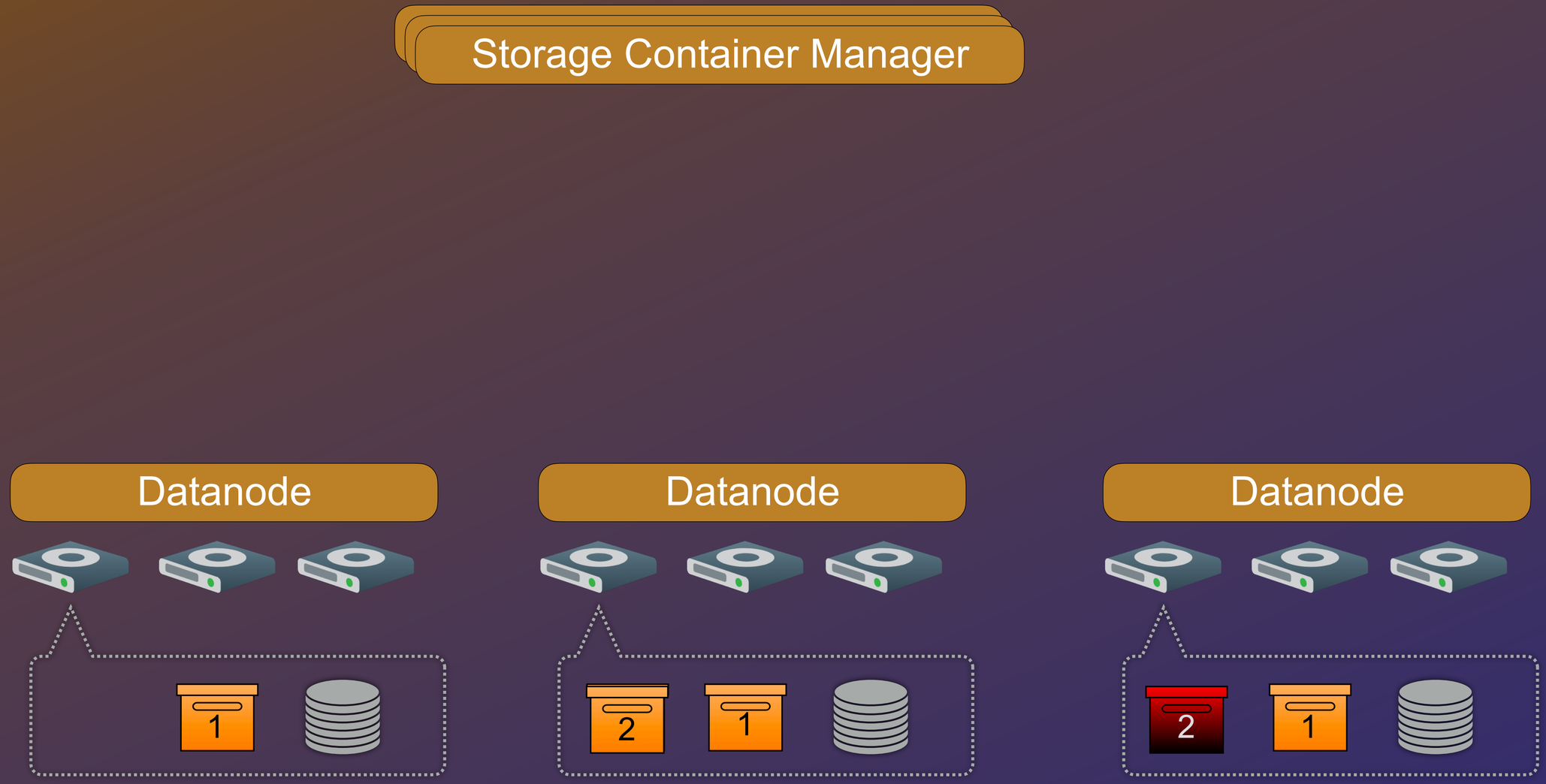




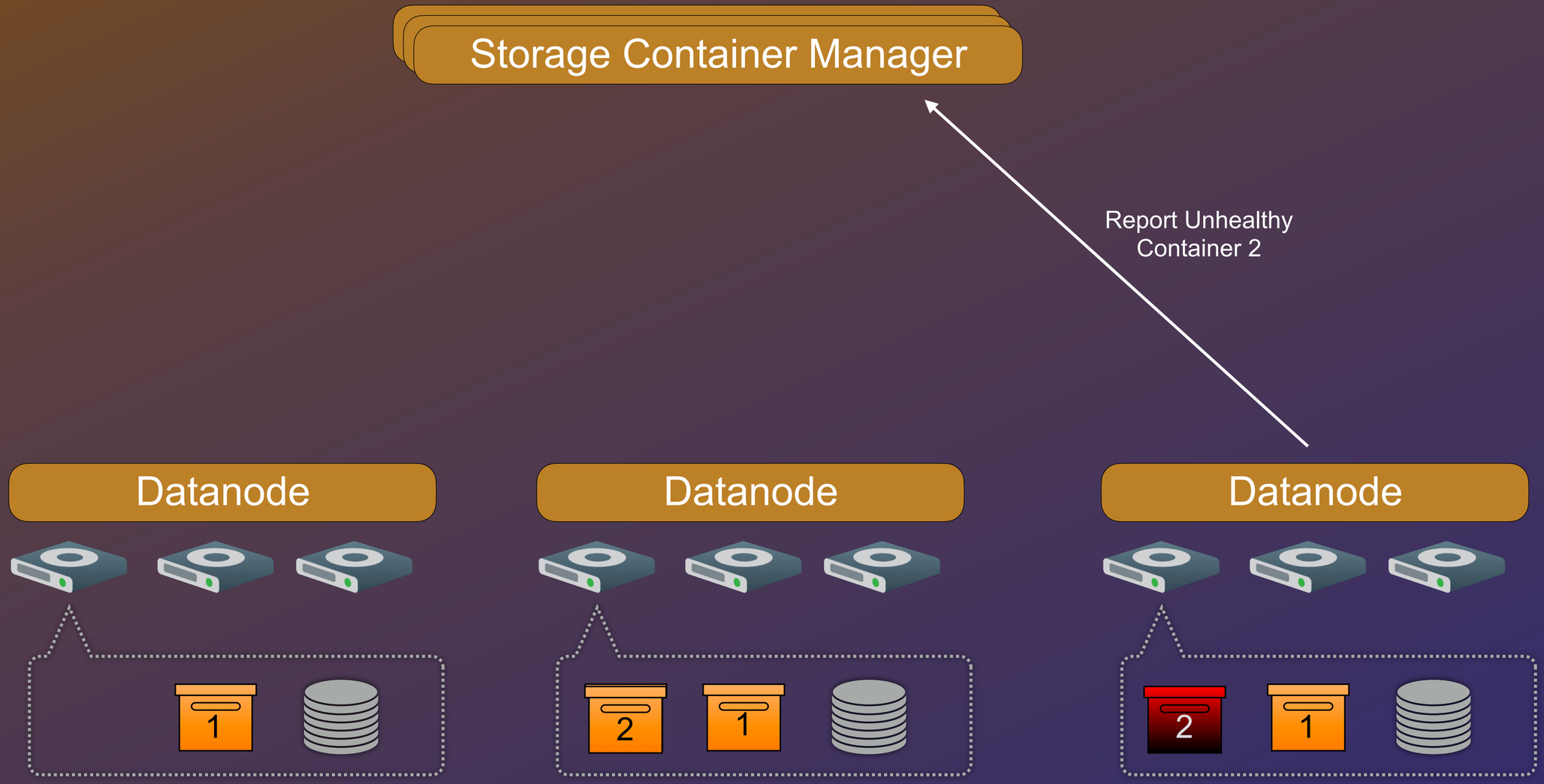
Container + Volume Scanner: On-Demand Scans

`hdds.container.scrub.data.scan.interval`	Frequency the background container data scanner runs
`hdds.container.scrub.on.demand.volume.bytes.per.second`	Bandwidth throttle for on-demand container data scanner
`hdds.container.scrub.min.gap`	Minimum gap between consecutive scans of the same container
`hdds.datanode.disk.check.min.gap`	Minimum gap between consecutive scans of the same volume

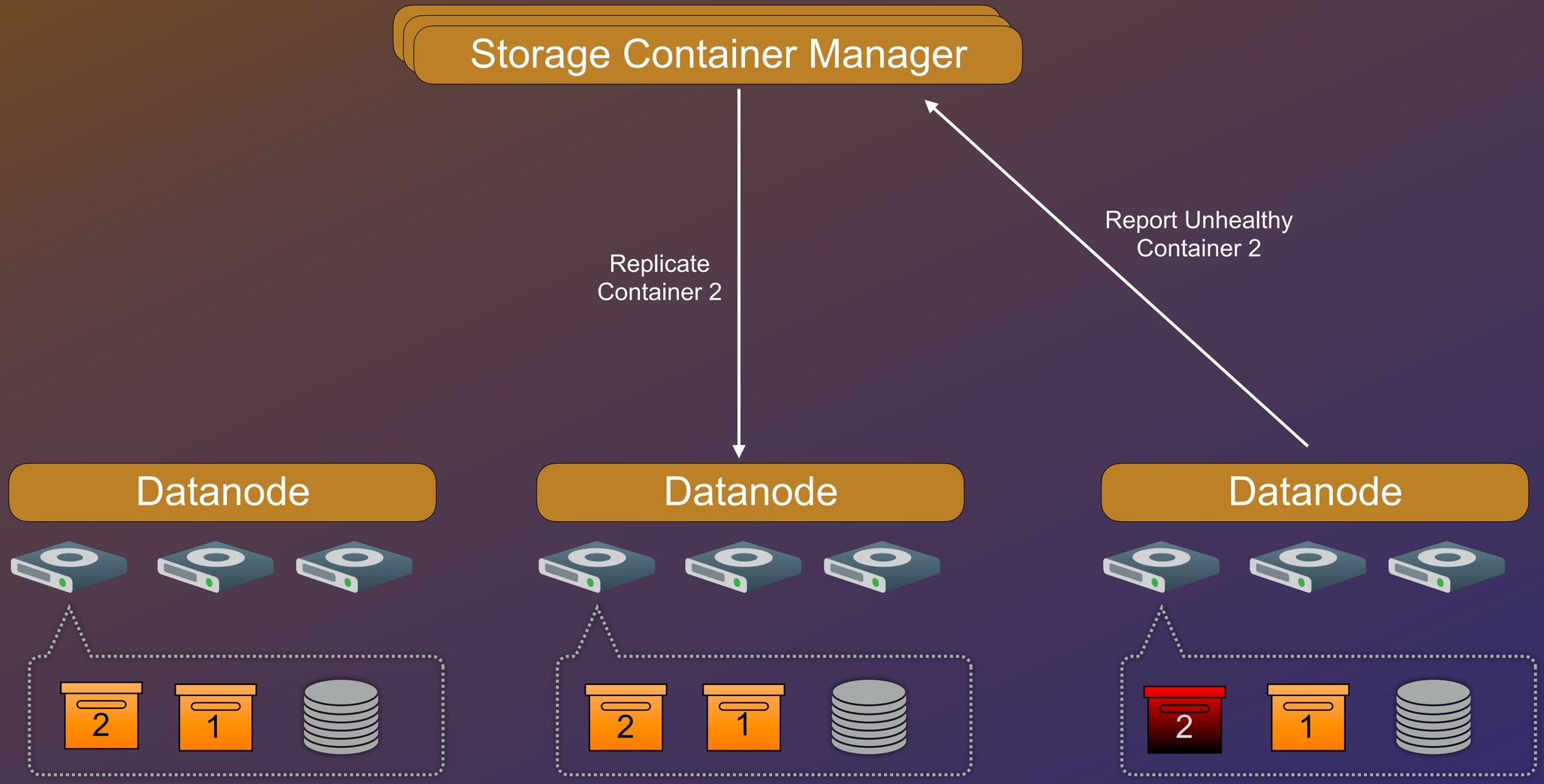
Recovering From Container Corruption



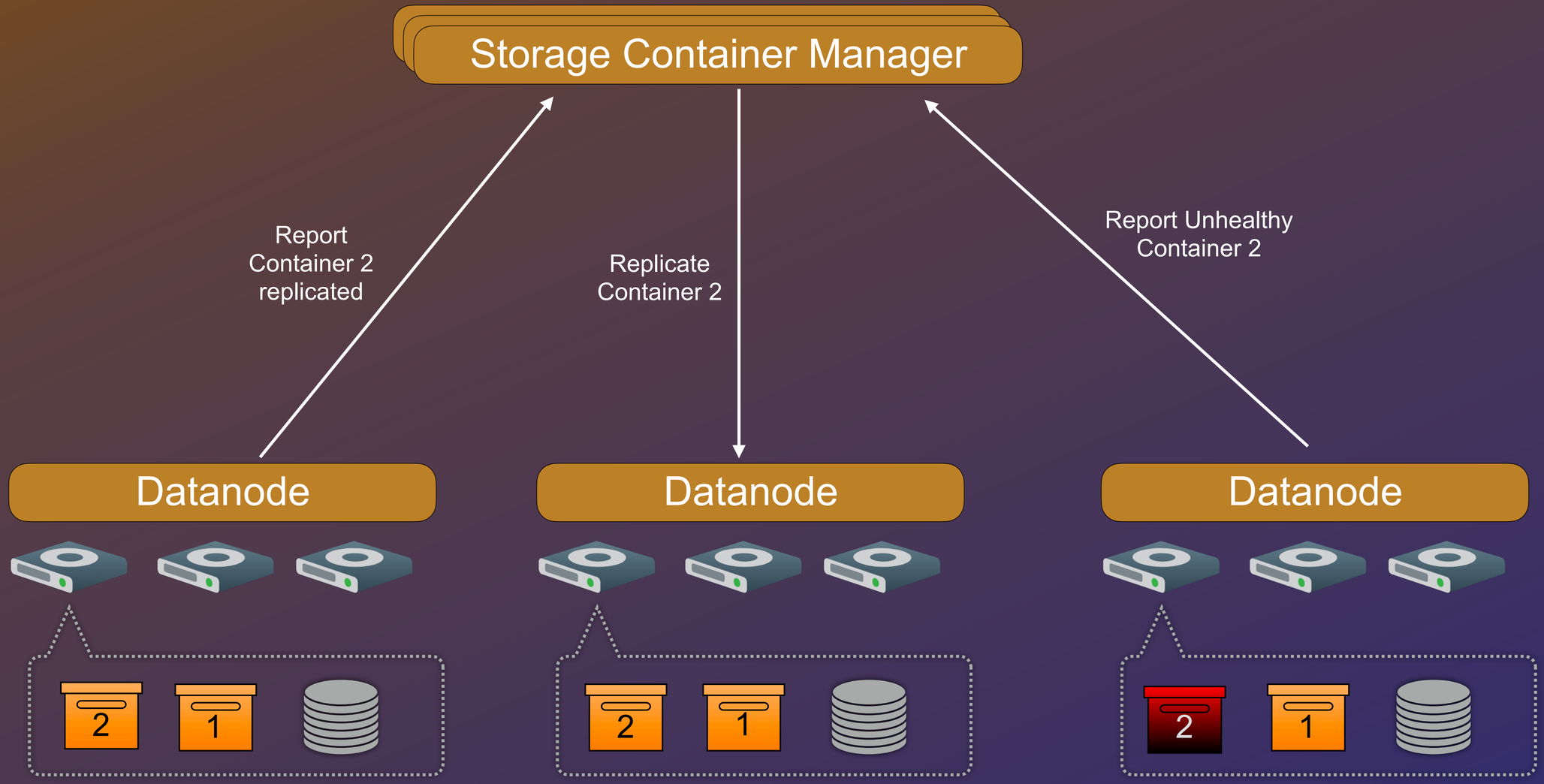
Recovering From Container Corruption



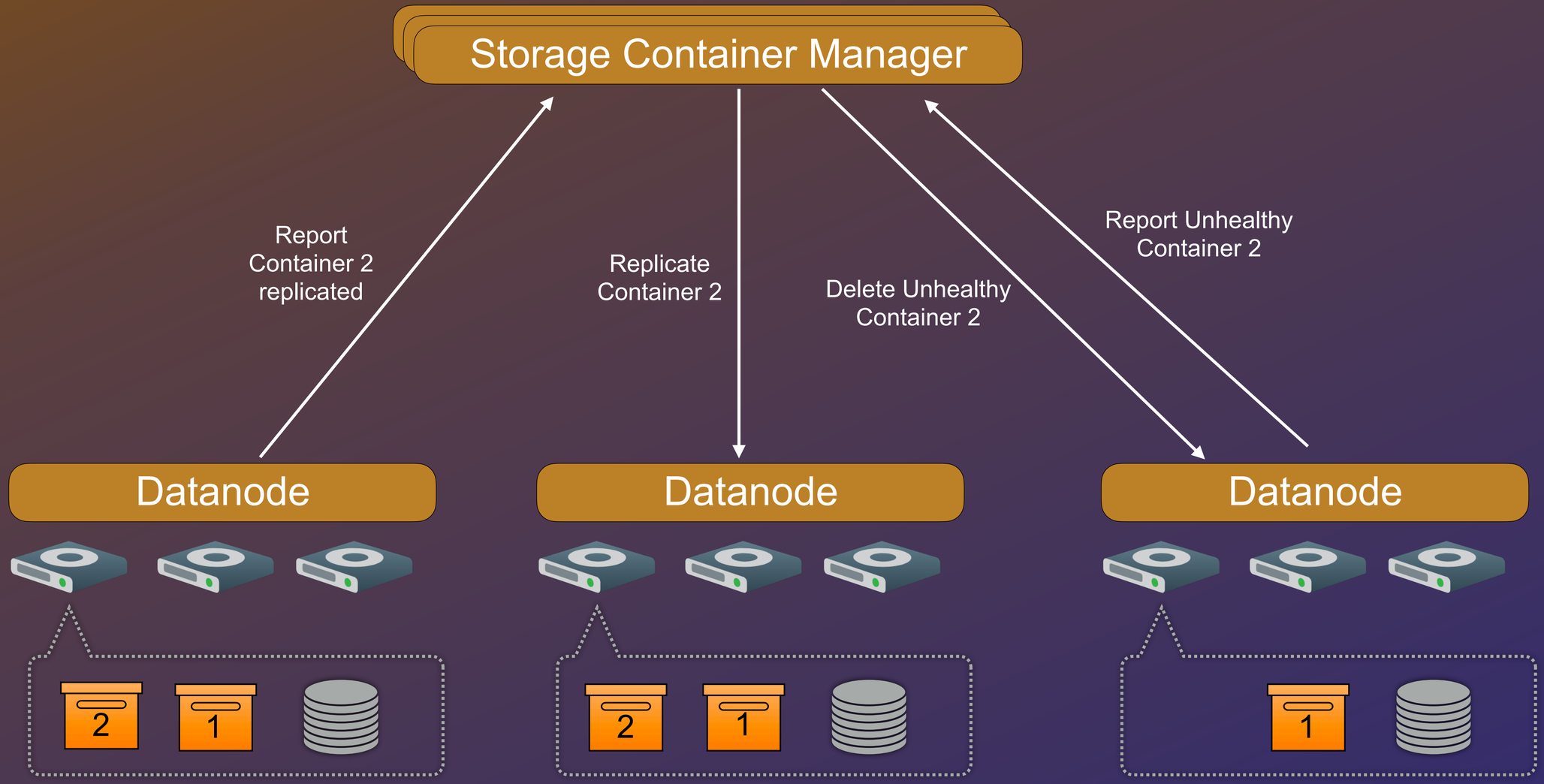
Recovering From Container Corruption



Recovering From Container Corruption



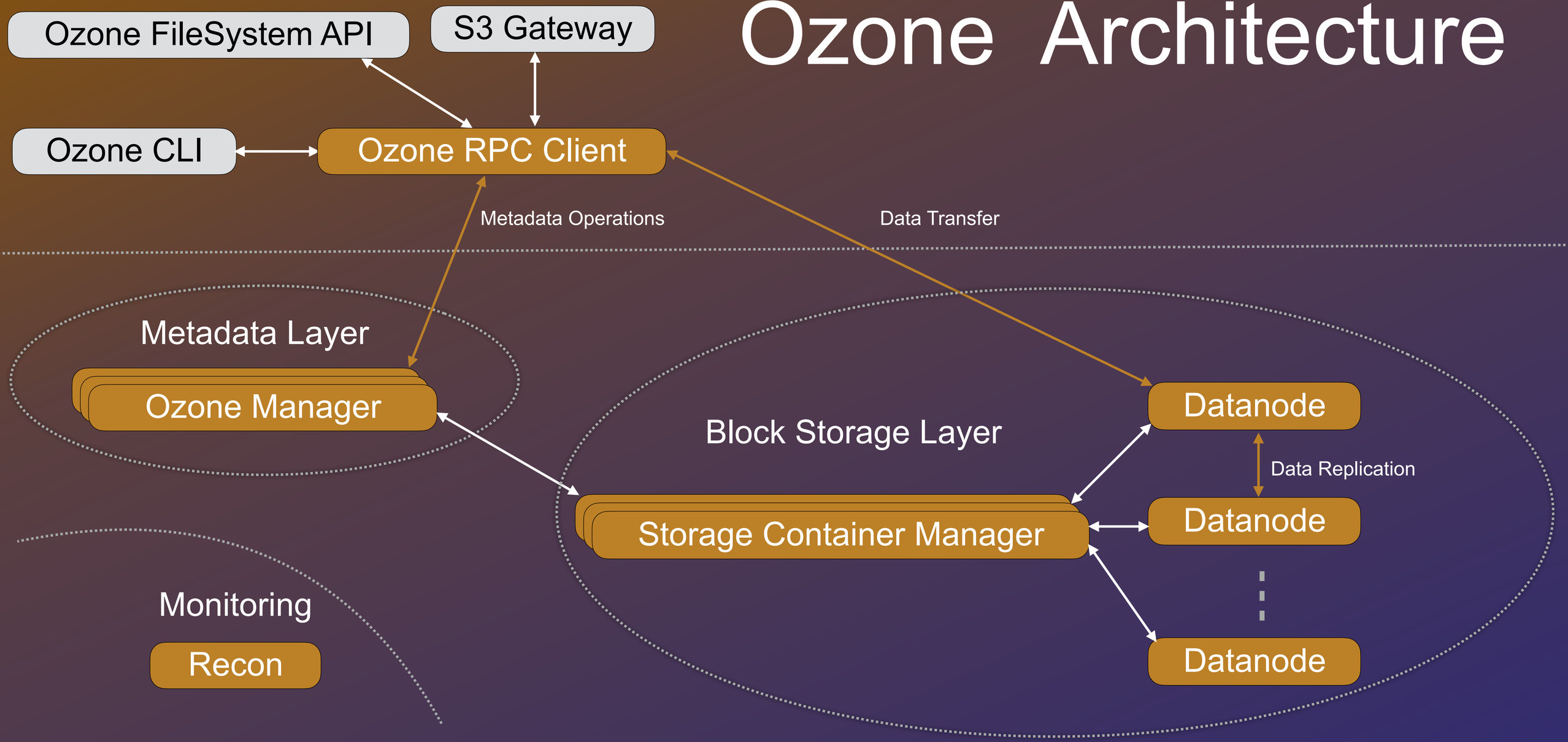
Recovering From Container Corruption



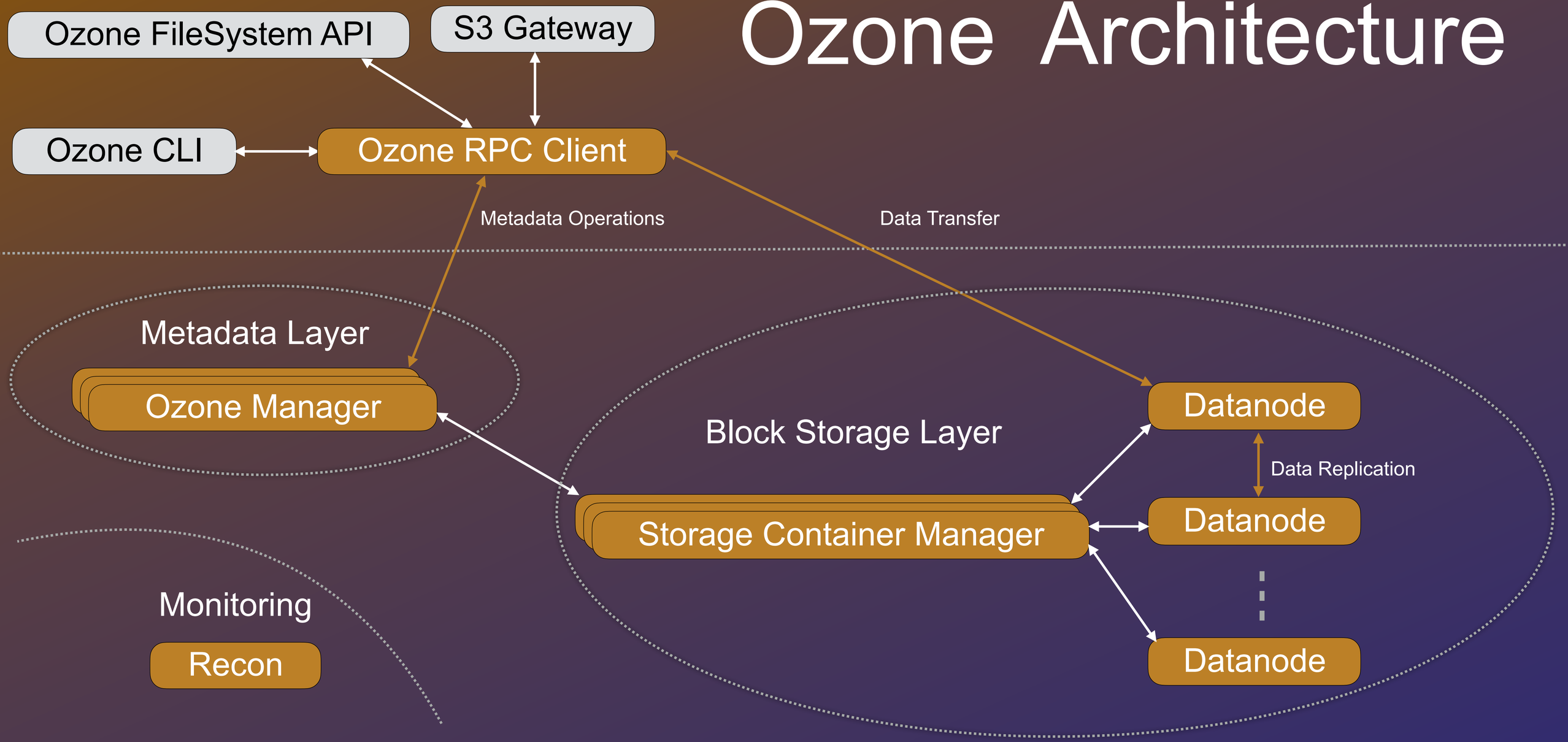
DN Faults: Relevant Config Keys

`hdds.scm.replication.thread.interval`	Frequency SCM's replication manager runs
`hdds.scm.replication.datanode.replication.limit`	Limits the number of replication commands queued per data node
`hdds.scm.replication.inflight.limit.factor`	Limits the total amount replication happening among the cluster

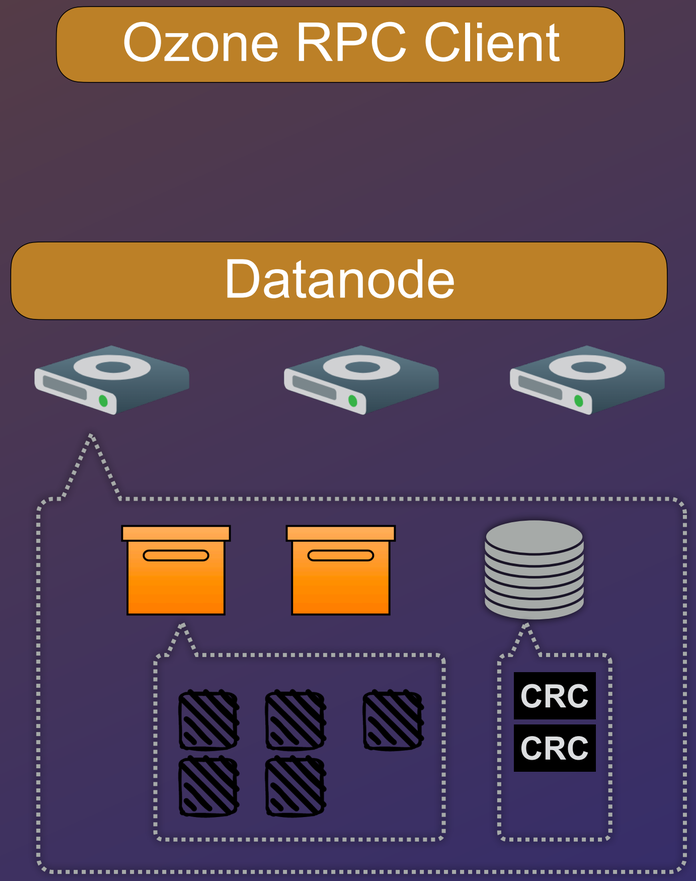
Ozone Architecture



Ozone Architecture

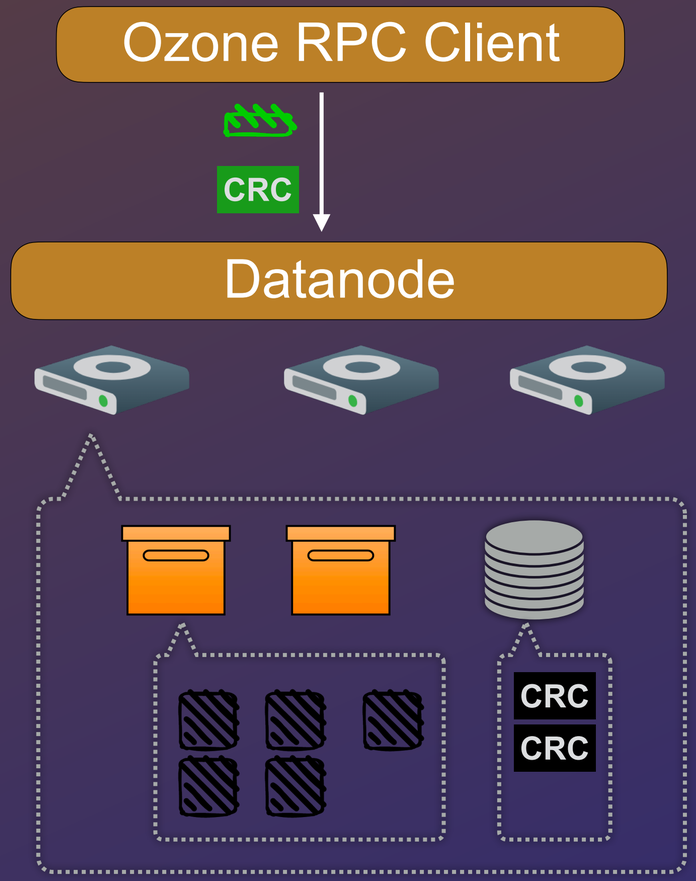


Client + Datanode: Write Corruption



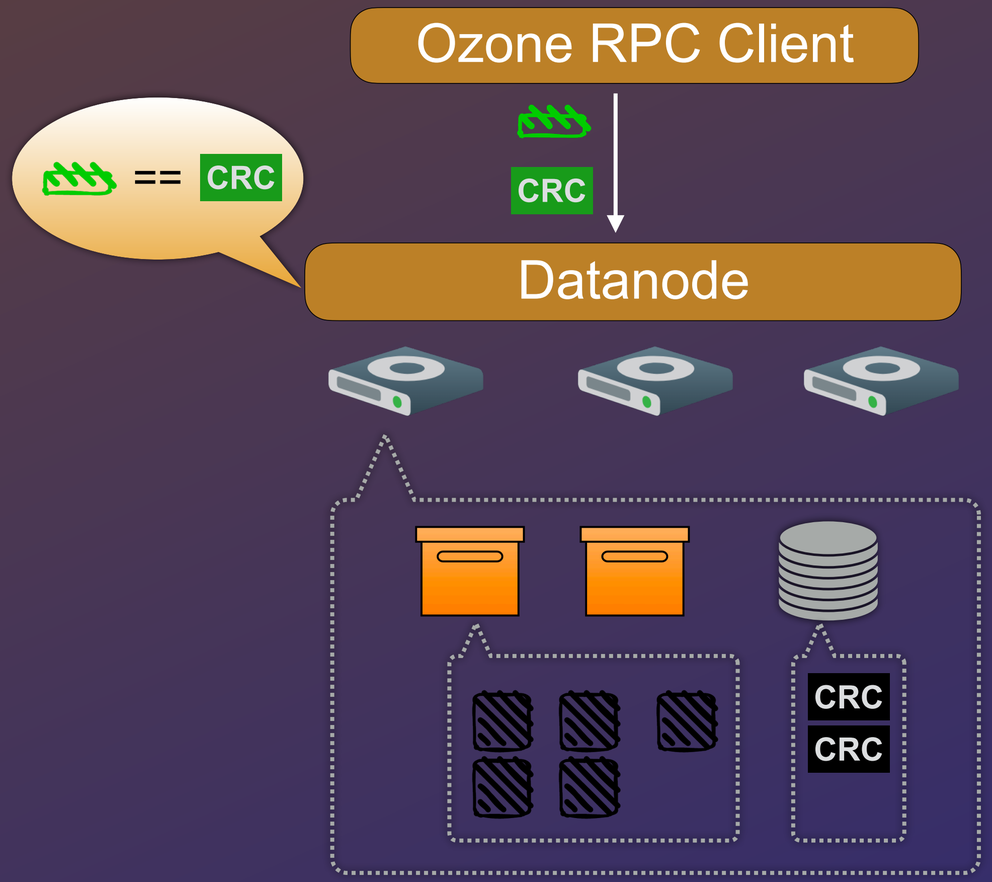
Client + Datanode: Write Corruption

1. Client calculates checksum for data
2. Client sends data + checksum to datanode



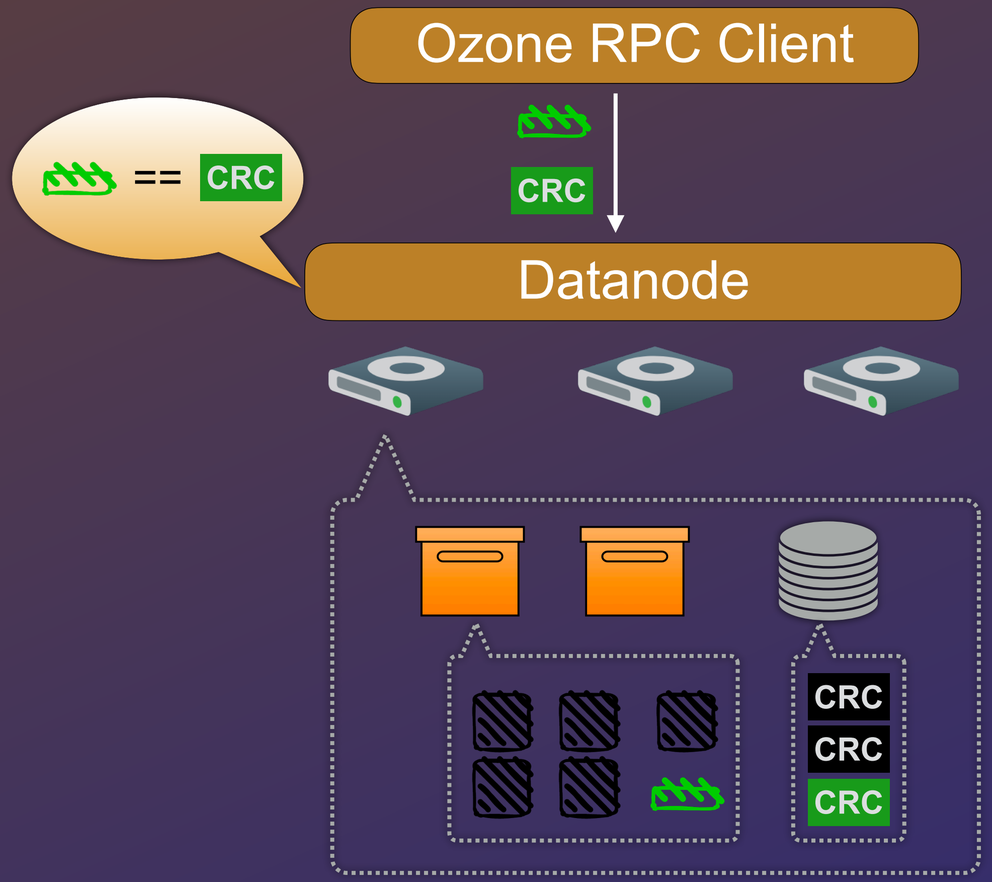
Client + Datanode: Write Corruption

1. Client calculates checksum for data
2. Client sends data + checksum to datanode
3. Datanode verifies checksum matches data



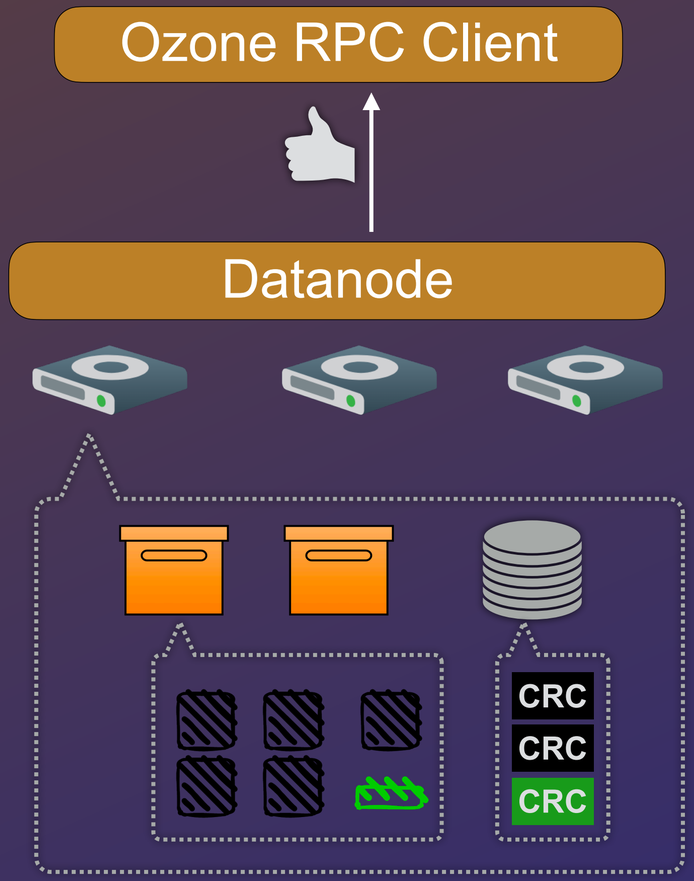
Client + Datanode: Write Corruption

1. Client calculates checksum for data
2. Client sends data + checksum to datanode
3. Datanode verifies checksum matches data
4. Datanode stores checksum and data

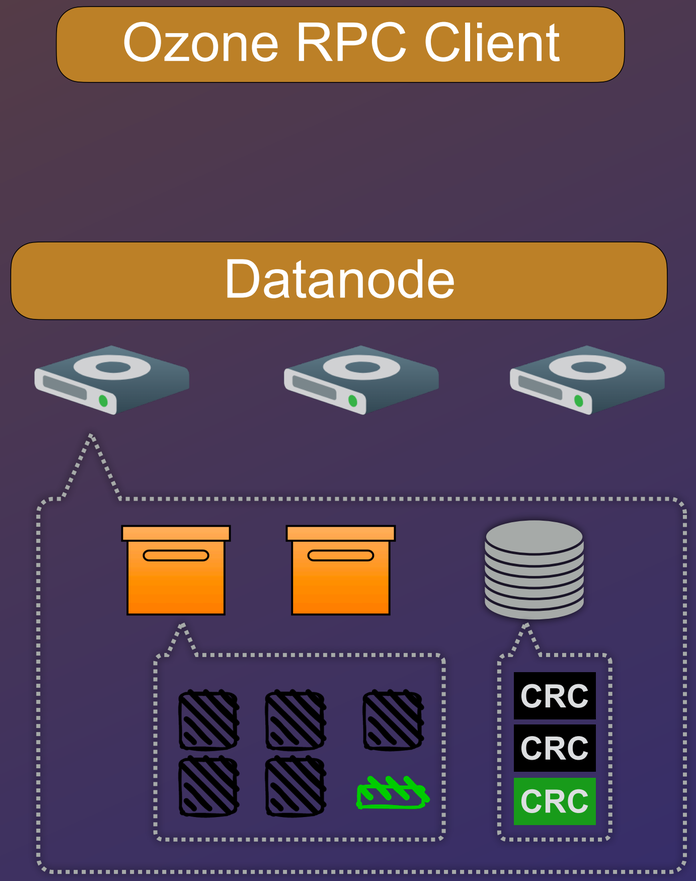


Client + Datanode: Write Corruption

1. Client calculates checksum for data
2. Client sends data + checksum to datanode
3. Datanode verifies checksum matches data
4. Datanode stores checksum and data
5. Datanode acks to client

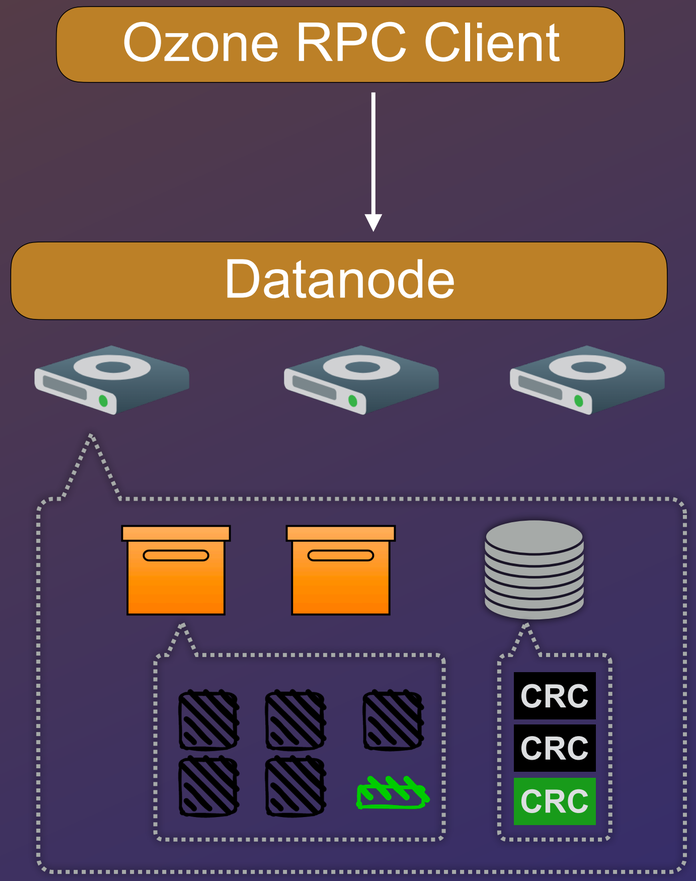


Client + Datanode: Read Corruption



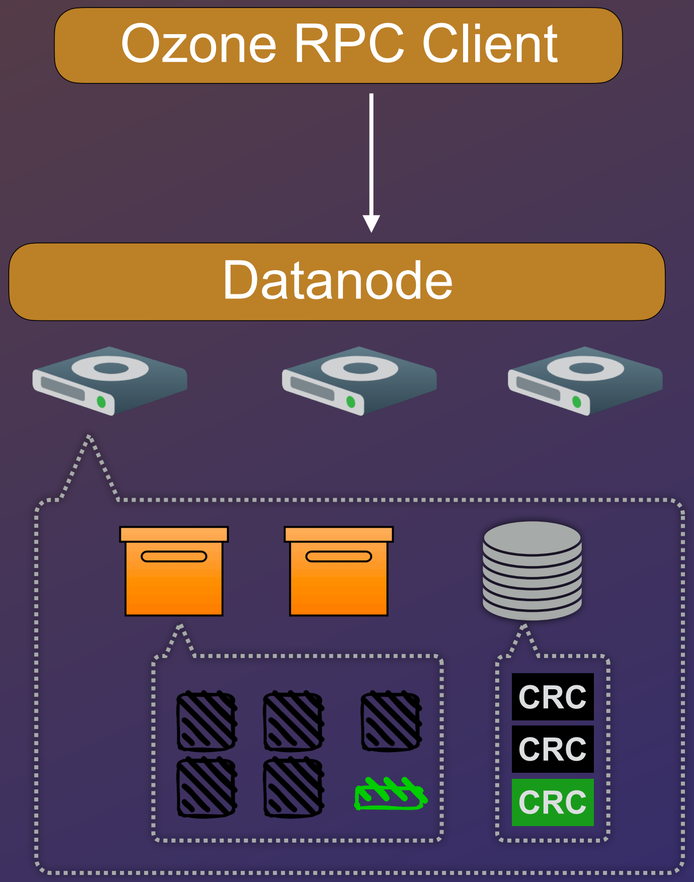
Client + Datanode: Read Corruption

1. Client requests data



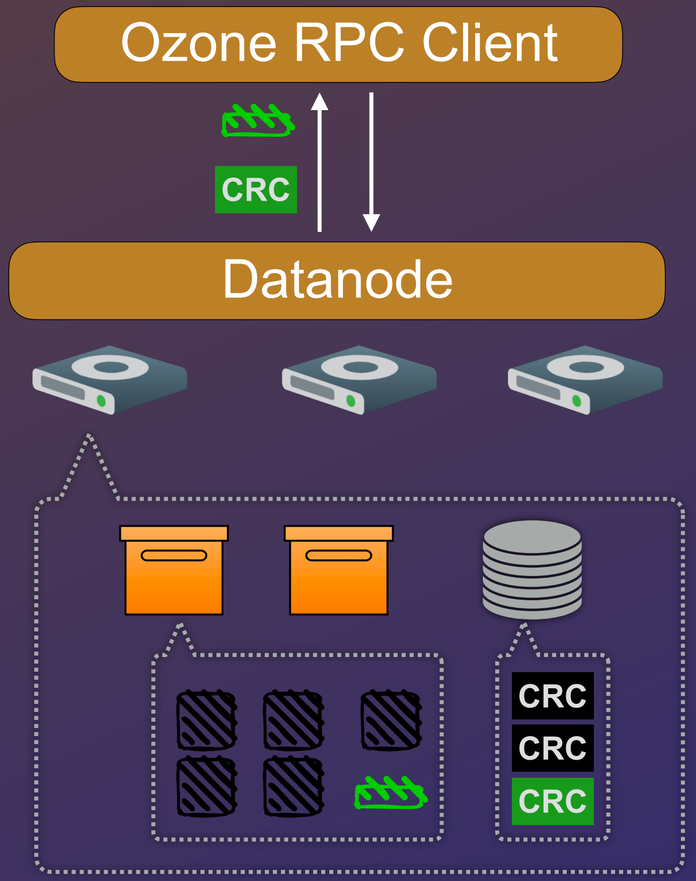
Client + Datanode: Read Corruption

1. Client requests data
2. Datanode retrieves existing checksums and data



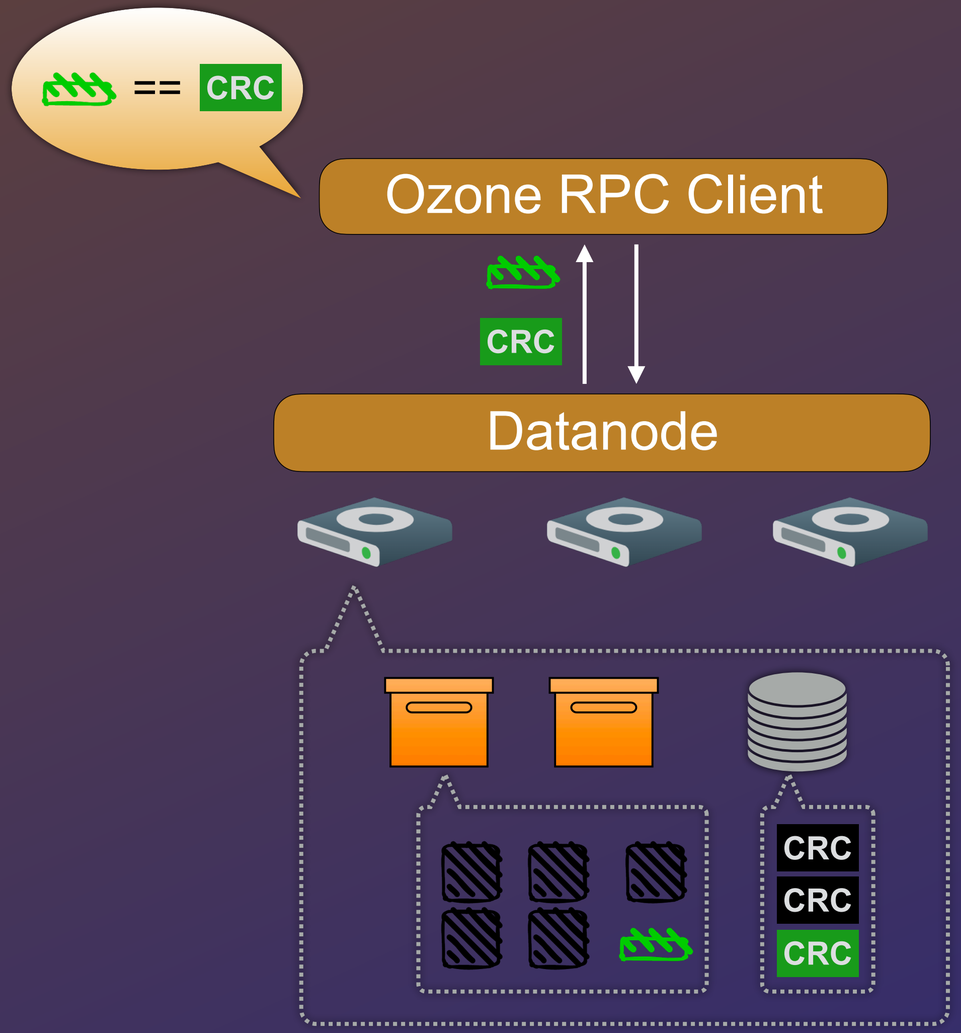
Client + Datanode: Read Corruption

1. Client requests data
2. Datanode retrieves existing checksums and data
3. Datanode sends checksums and data to client



Client + Datanode: Read Corruption

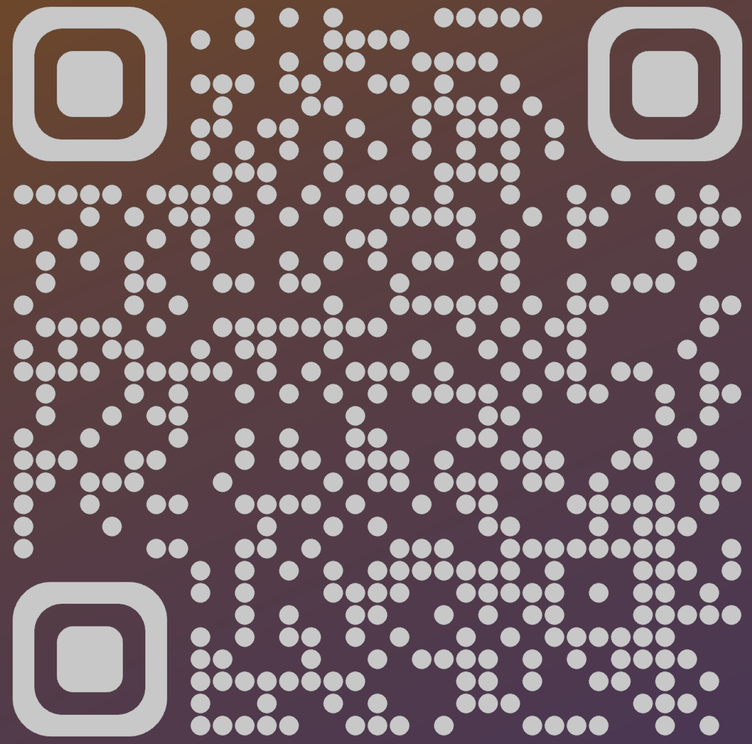
1. Client requests data
2. Datanode retrieves existing checksums and data
3. Datanode sends checksums and data to client
4. Client verifies checksum matches data



Future Improvements

- Write checksum performance improvements (WIP) HDDS-7228
- Identifying and avoiding slow datanodes (soft failures)
- Store checksum inside block files (for zero copy support)
- More robust disk failure detection
- Periodically verify RocksDB read checksums for metadata
- Use metrics from deployments to better tune default configs

Future of Data Storage User Group Meetup



- October 25, 2023
- 4-7 PM PST
- Santa Clara, California + Online

Uber

CLOUDERA

ICEBERG