



Recognize, Reconcile, Repeat: The Path to Uniform Replicas in Apache Ozone

Ethan Rose

*Senior Software Engineer, Cloudera Inc
Apache Ozone Committer, PMC Member*

Architecture

Goals

Tools

Implementation

Future

Ozone Architecture

Ozone Architecture

Metadata Layer

Ozone Manager

Ozone Architecture

Metadata Layer

Ozone Manager

Block Storage Layer

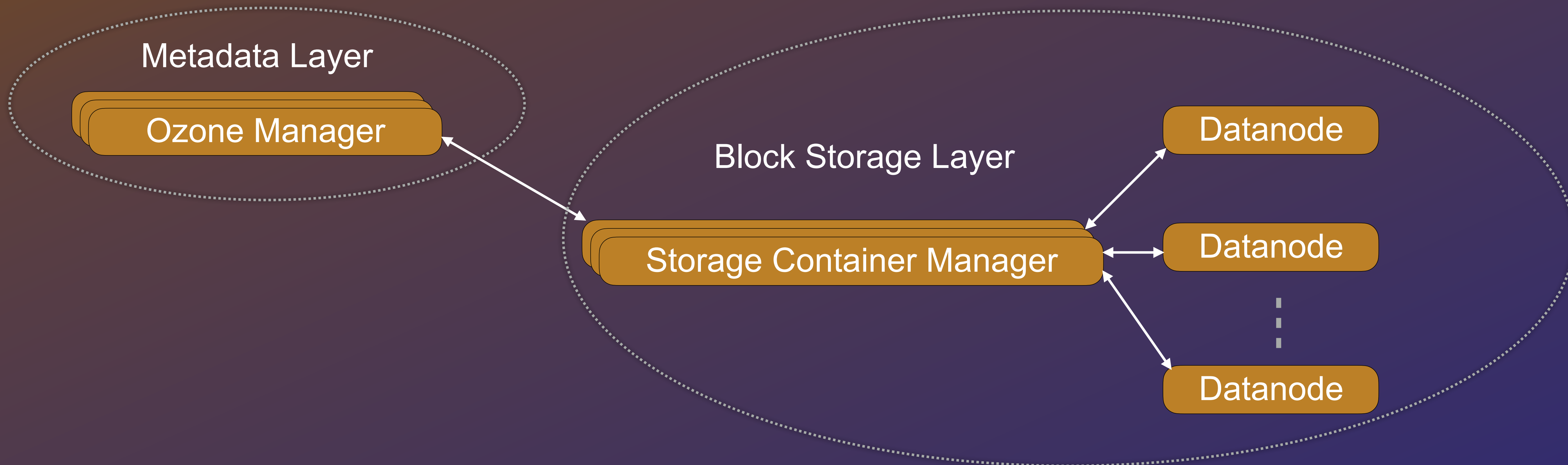
Storage Container Manager

Datanode

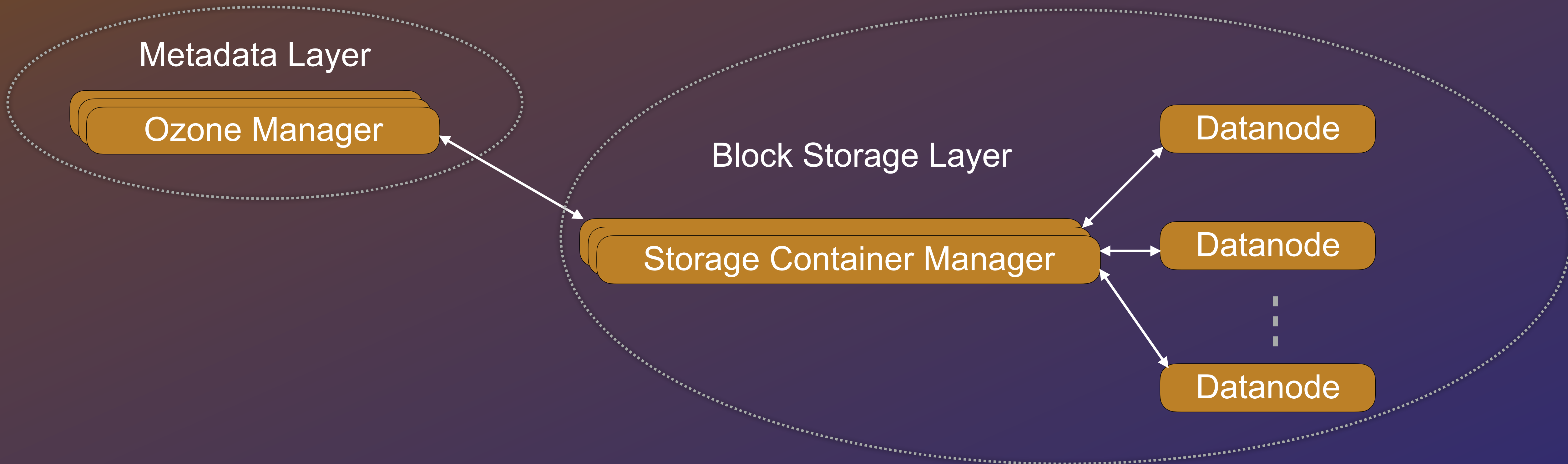
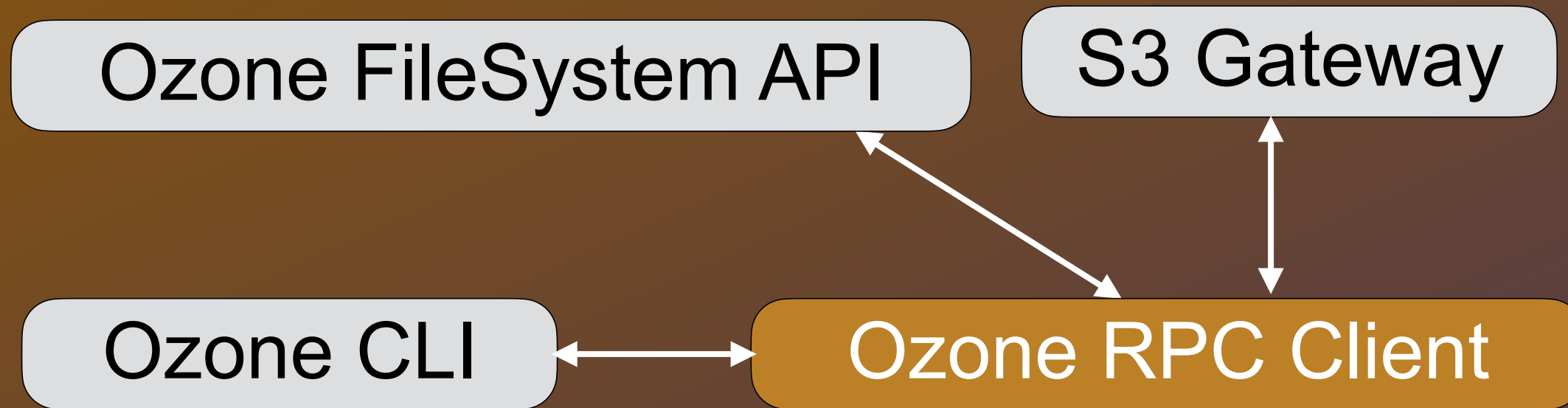
Datanode

Datanode

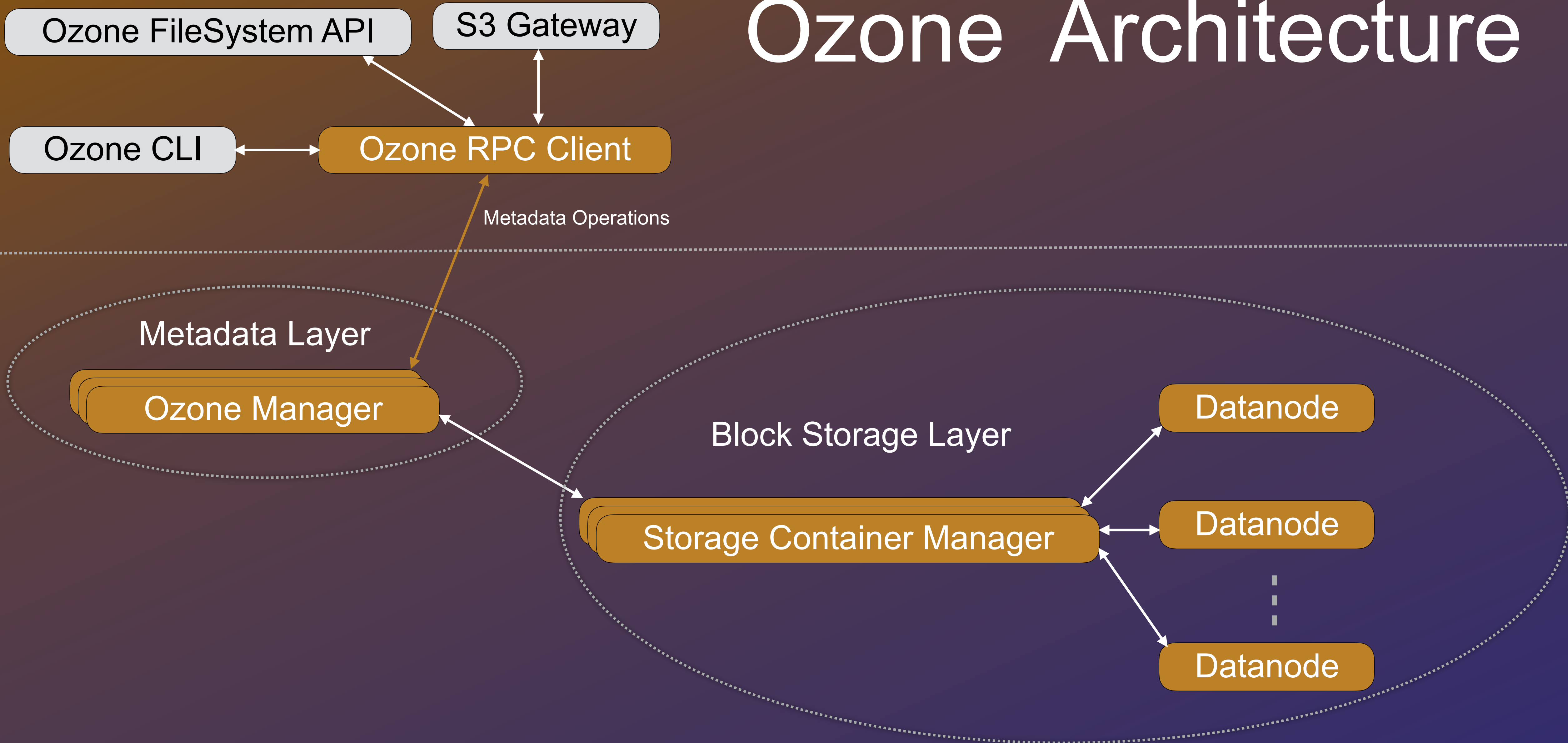
Ozone Architecture



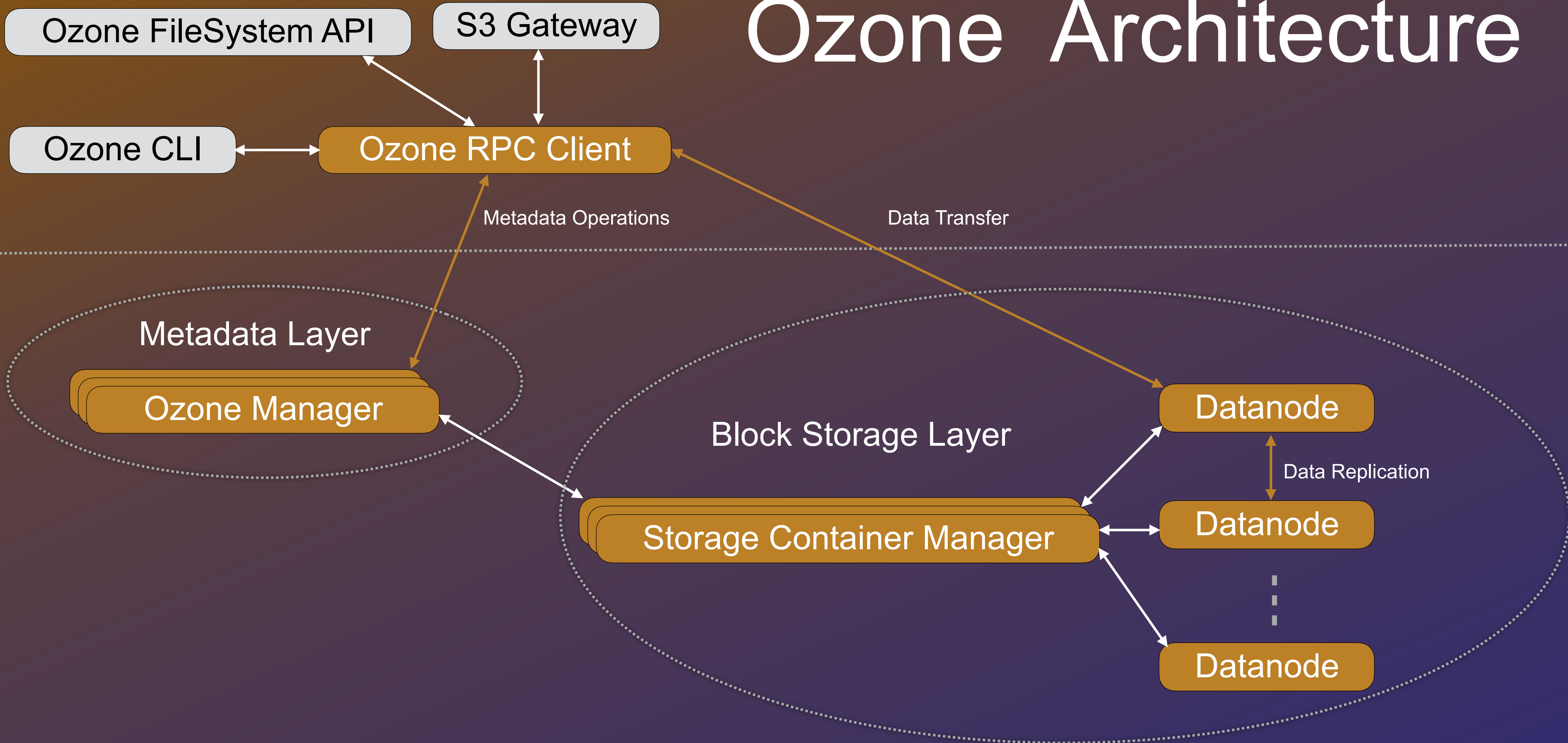
Ozone Architecture



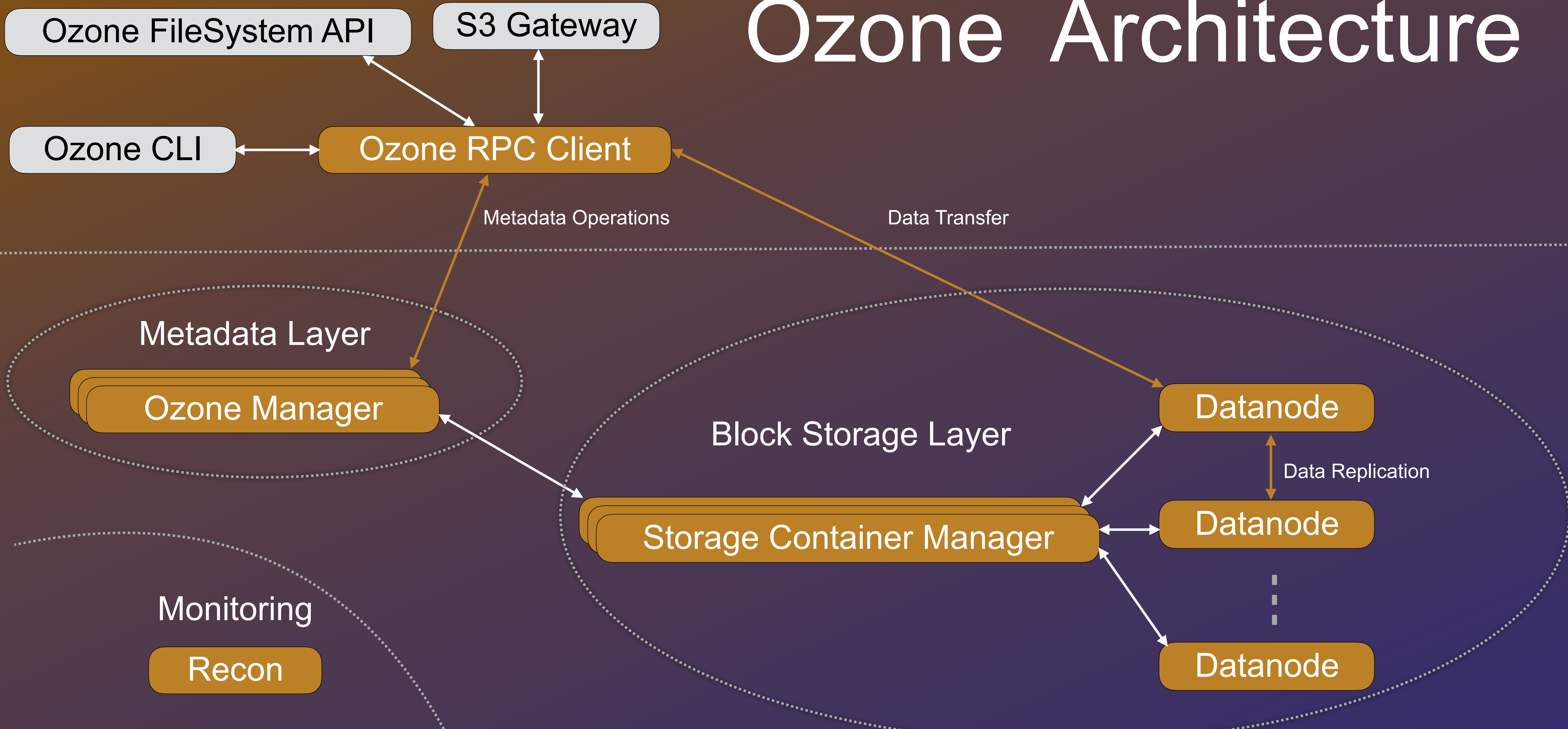
Ozone Architecture



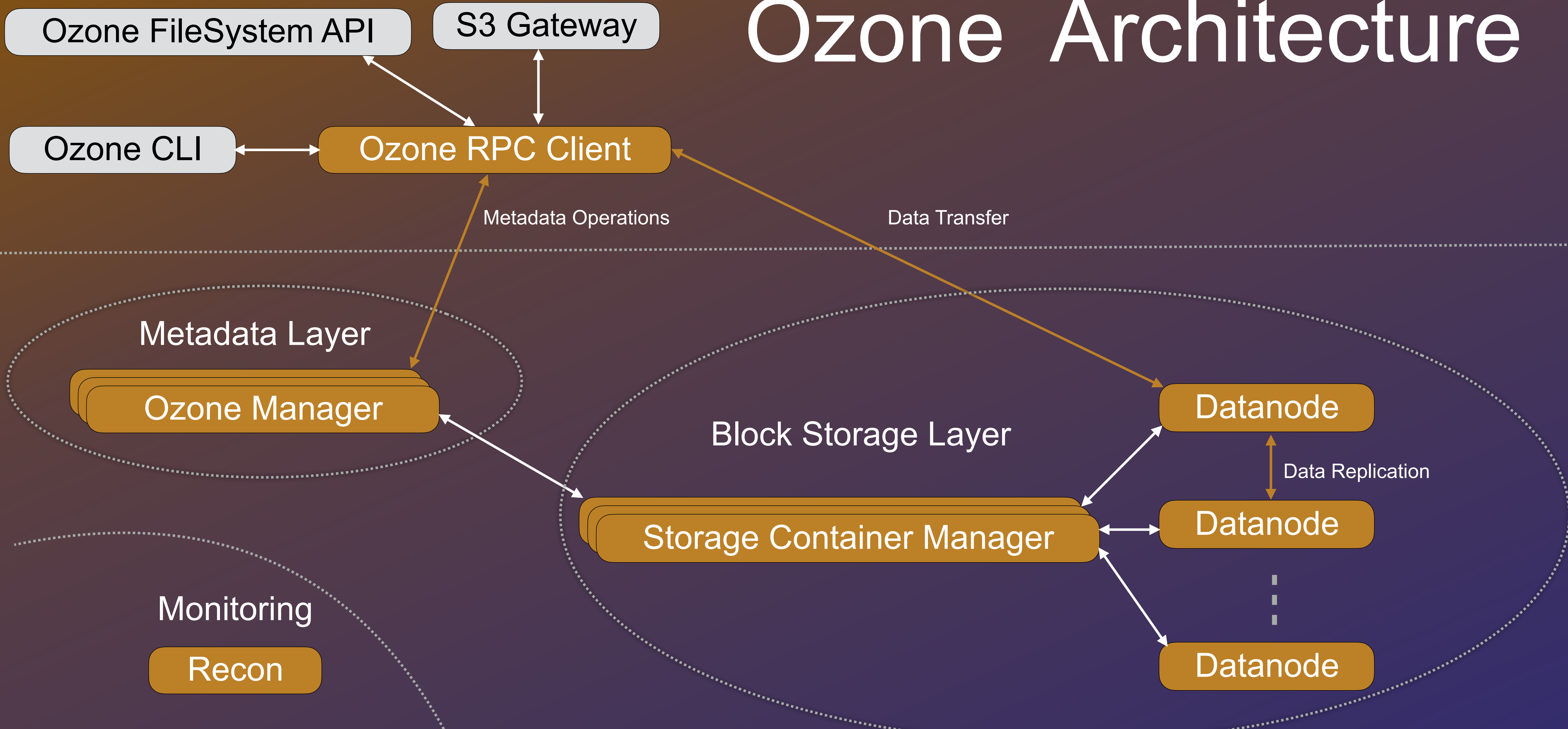
Ozone Architecture



Ozone Architecture



Ozone Architecture



Datanodes

Datanode

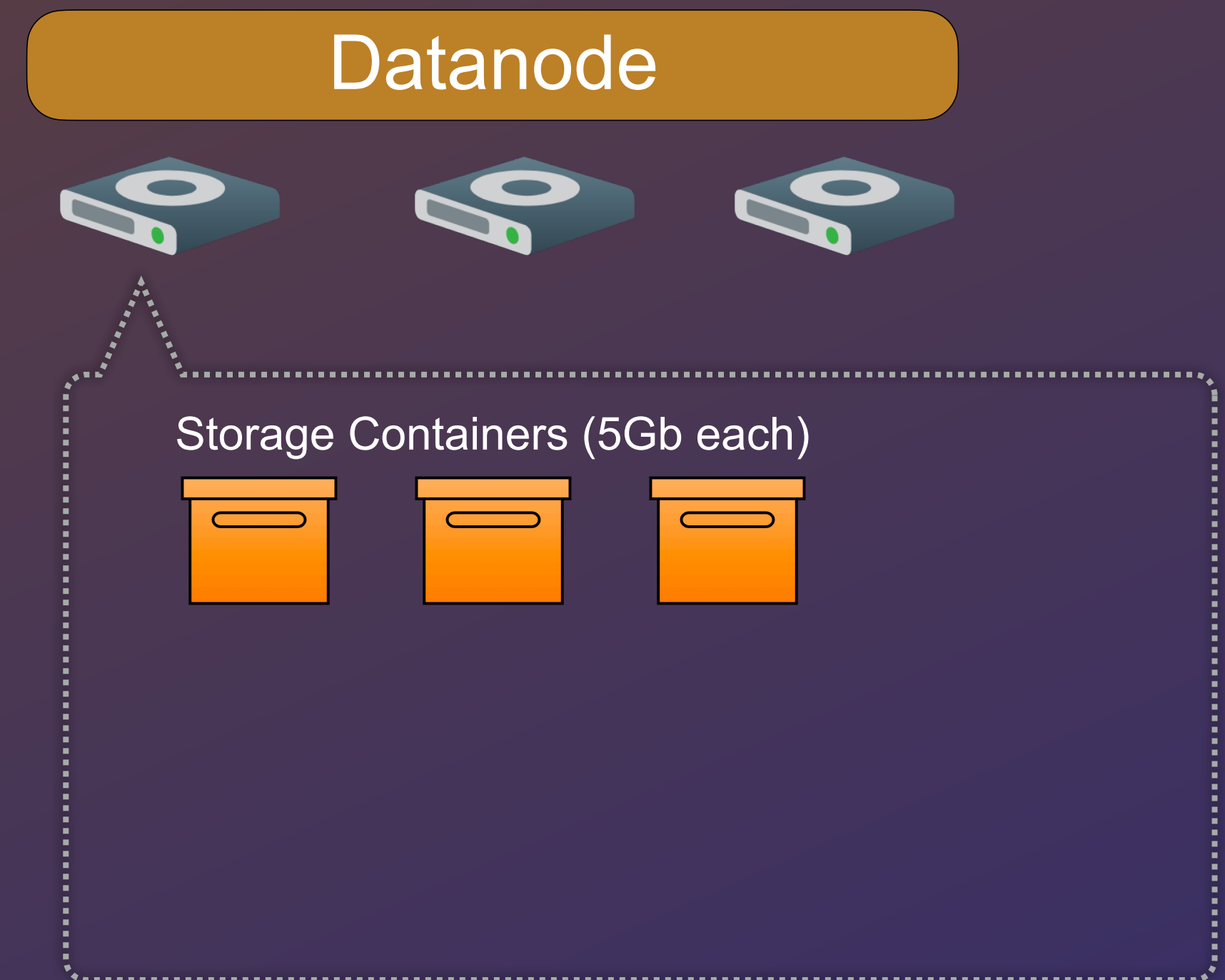
Datanodes

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



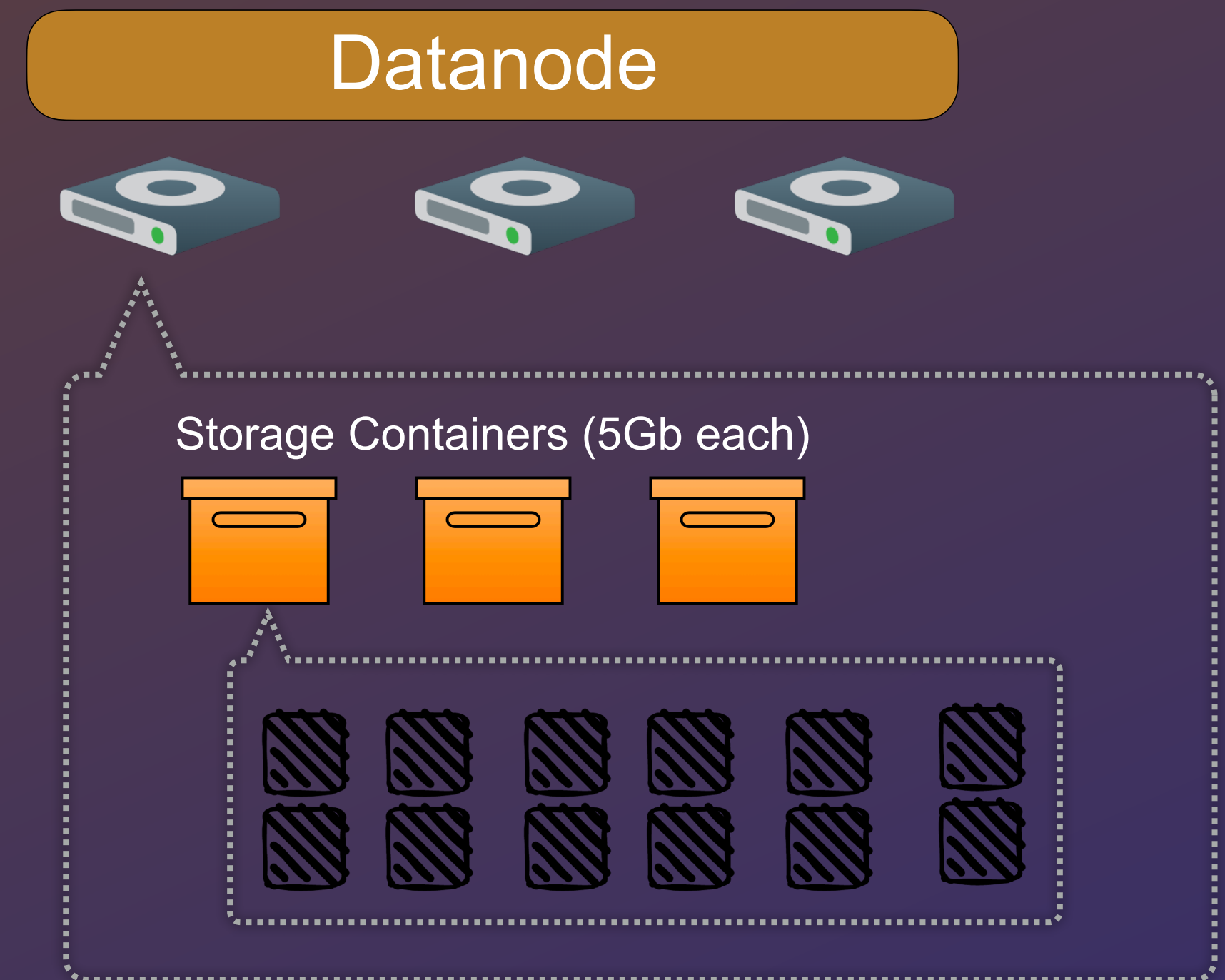
Datanodes

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



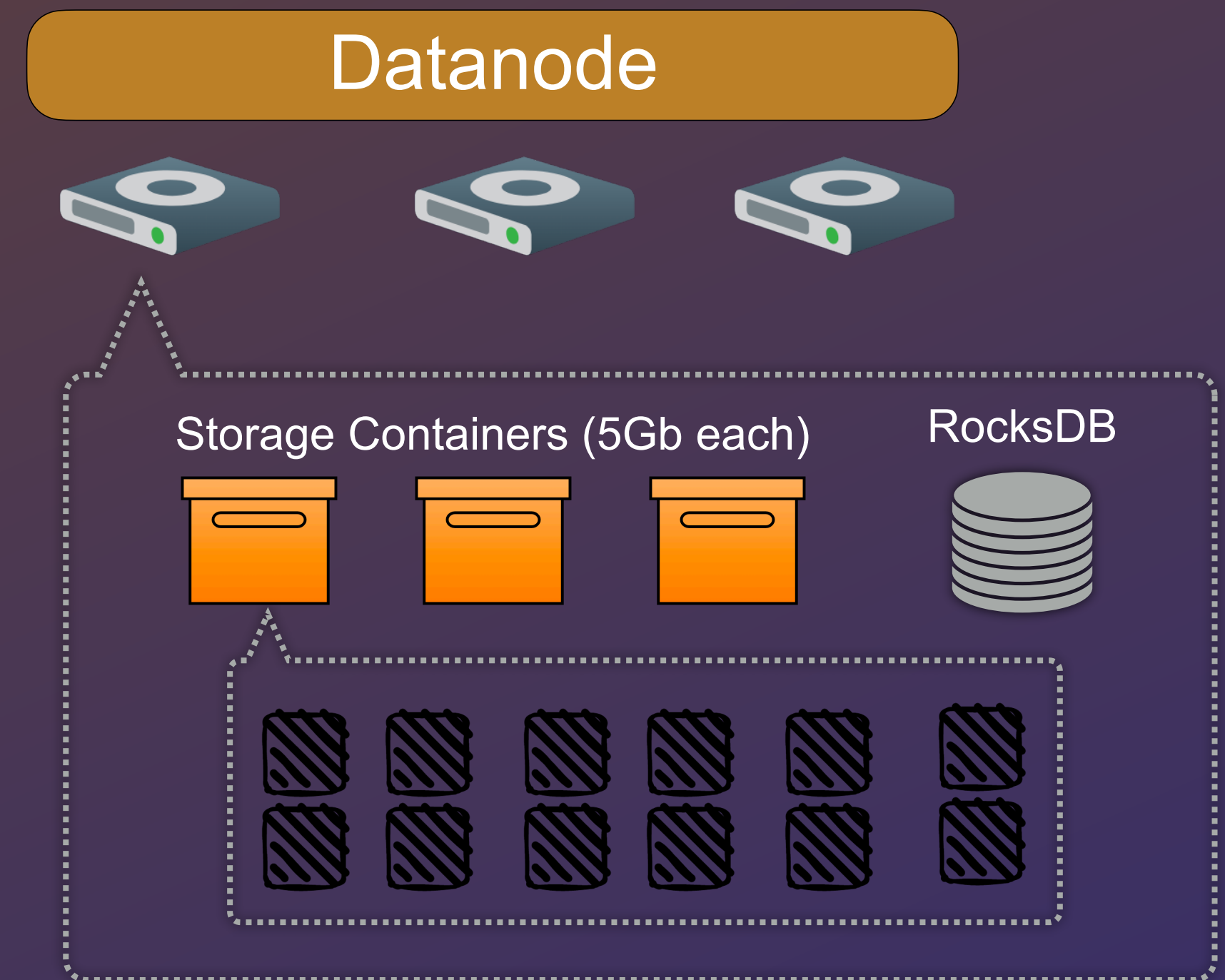
Datanodes

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

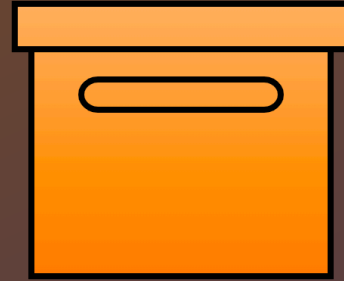


Datanodes

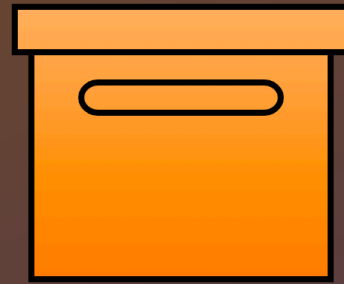
- Comprised of multiple **volumes** (disks) that may fail independently
- Each volume holds **storage containers**
- Each storage container is a 5gb collection of blocks
- RocksDB per volume holds metadata



Storage Container States



Storage Container States

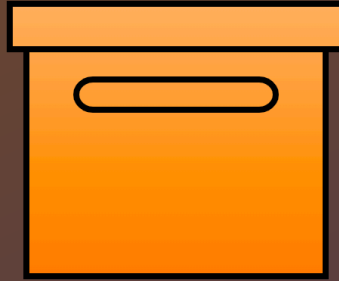





Read

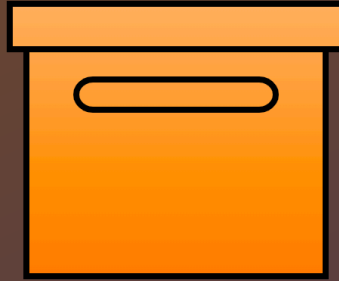






Write

Replicate

Storage Container States

	OPEN
Read	
Write	
Replicate	

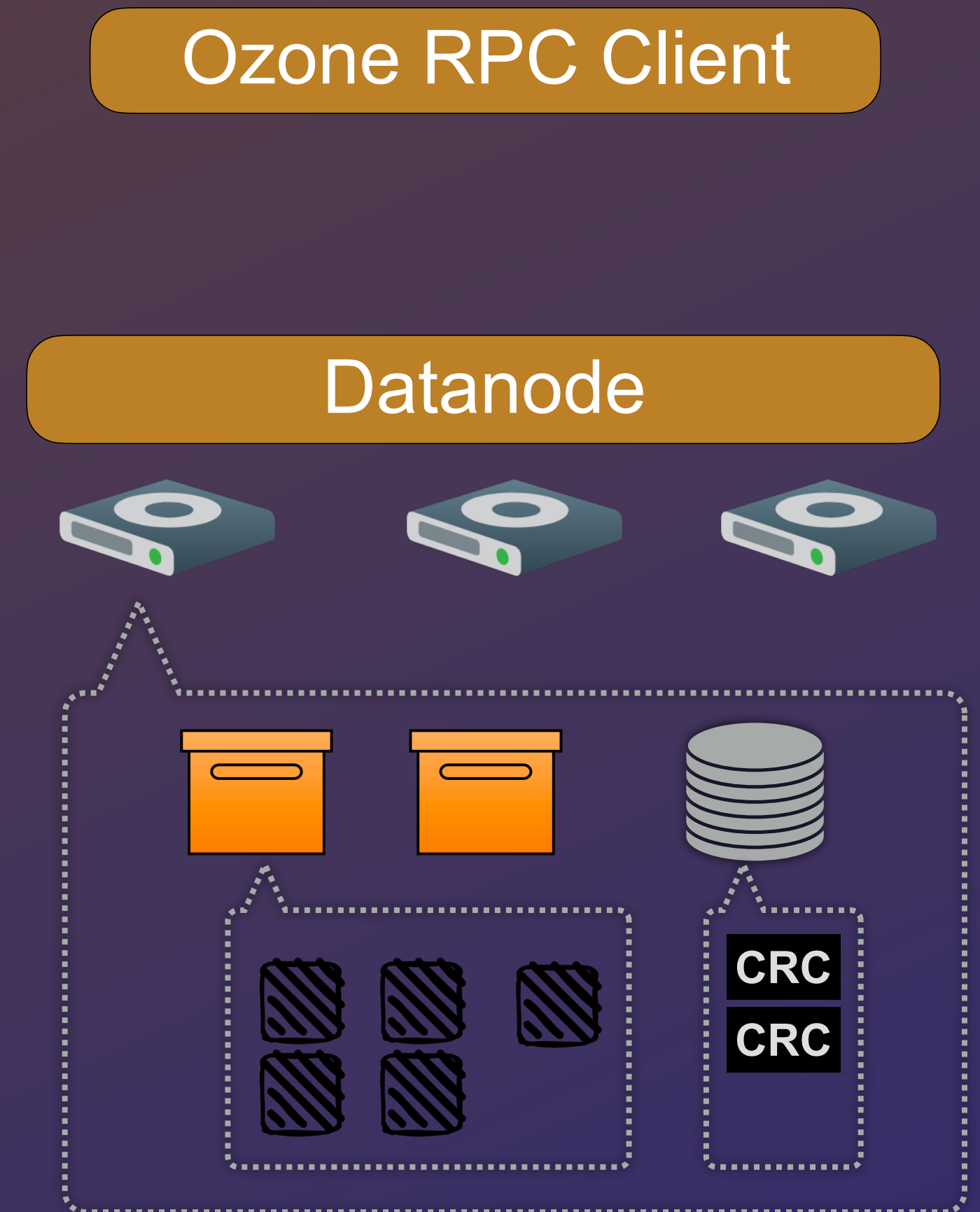
Storage Container States

	OPEN	CLOSED
Read		
Write		
Replicate		

Storage Container States

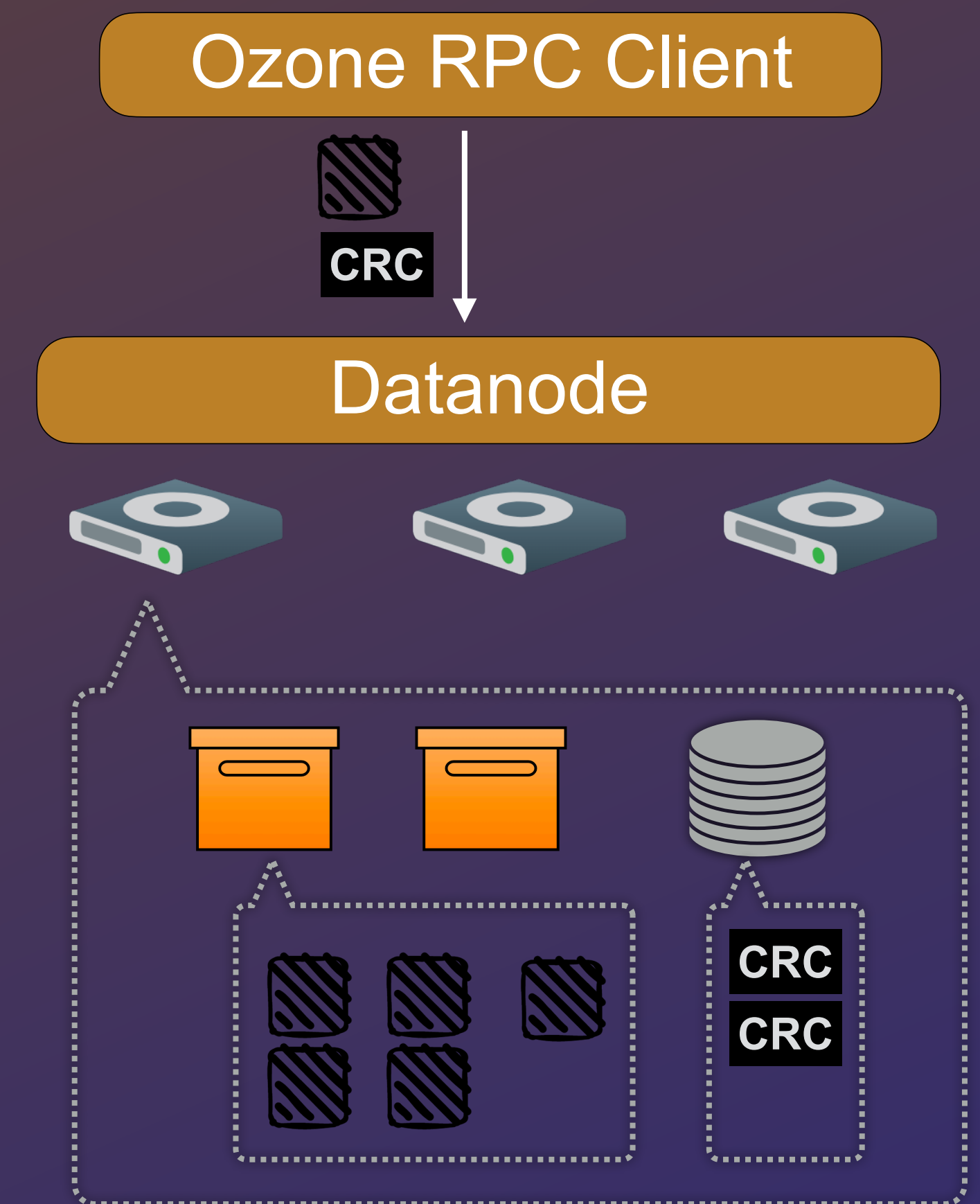
	OPEN	CLOSED	UNHEALTHY
Read			
Write			
Replicate			

Write Checksums



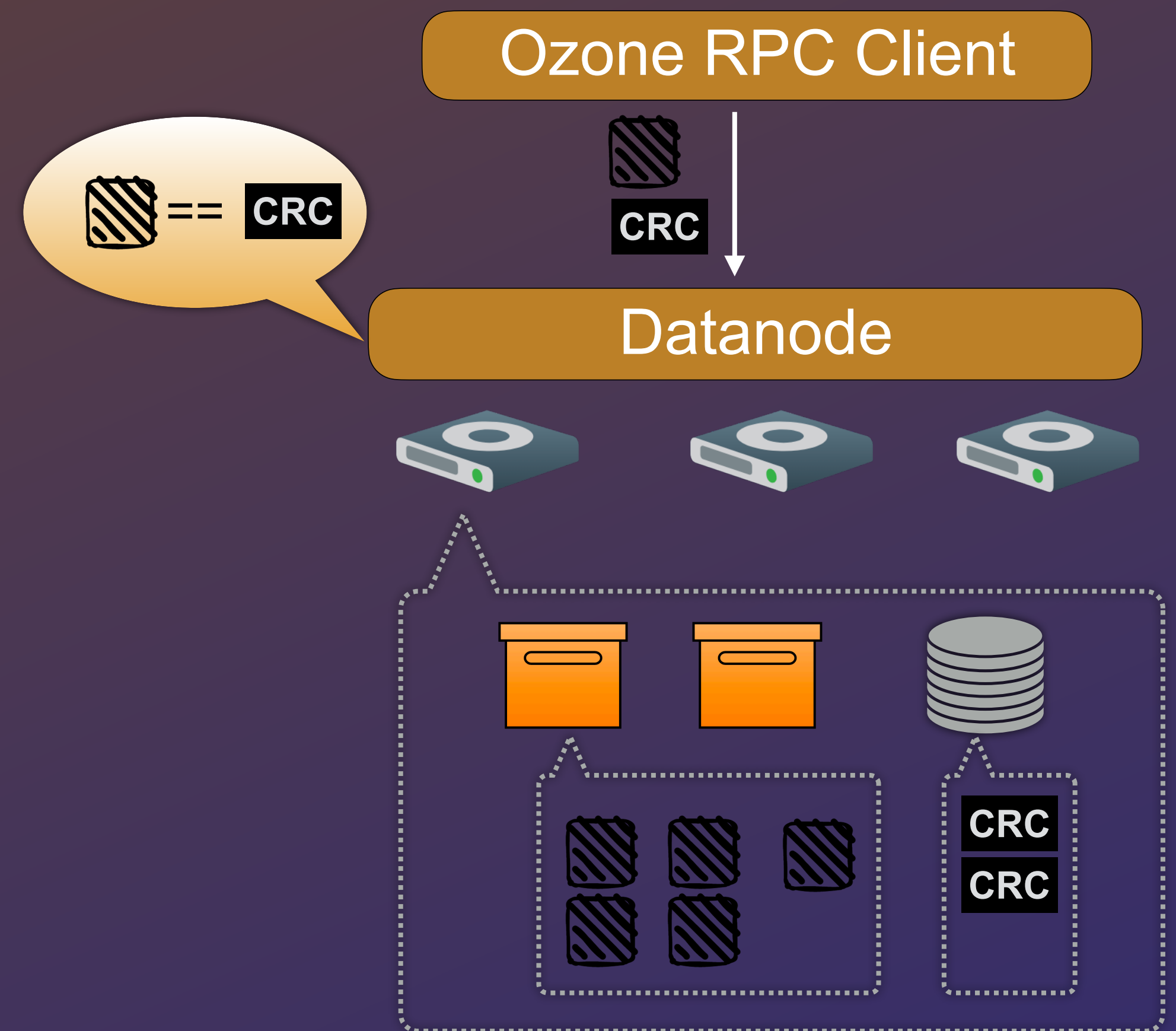
Write Checksums

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



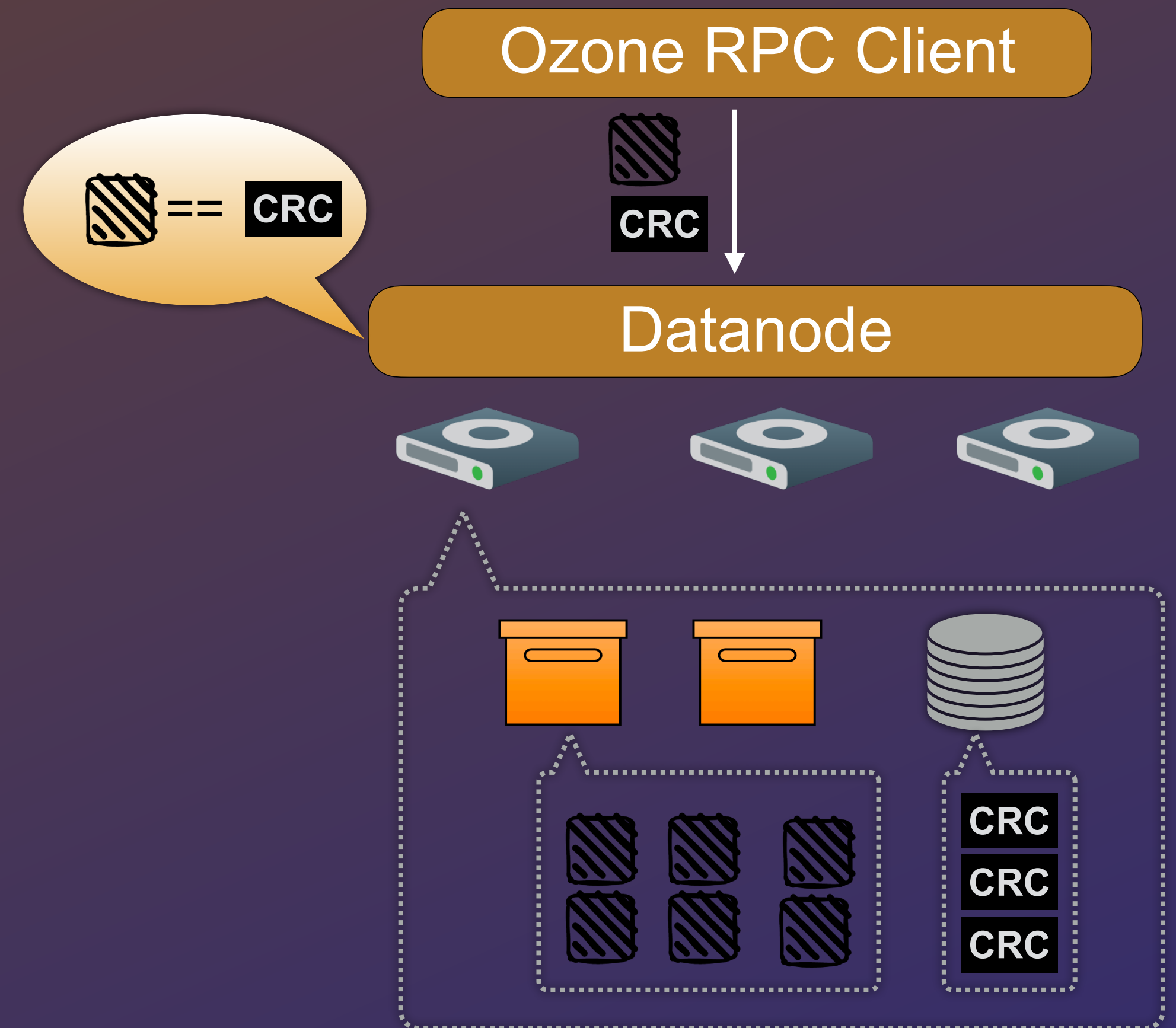
Write Checksums

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



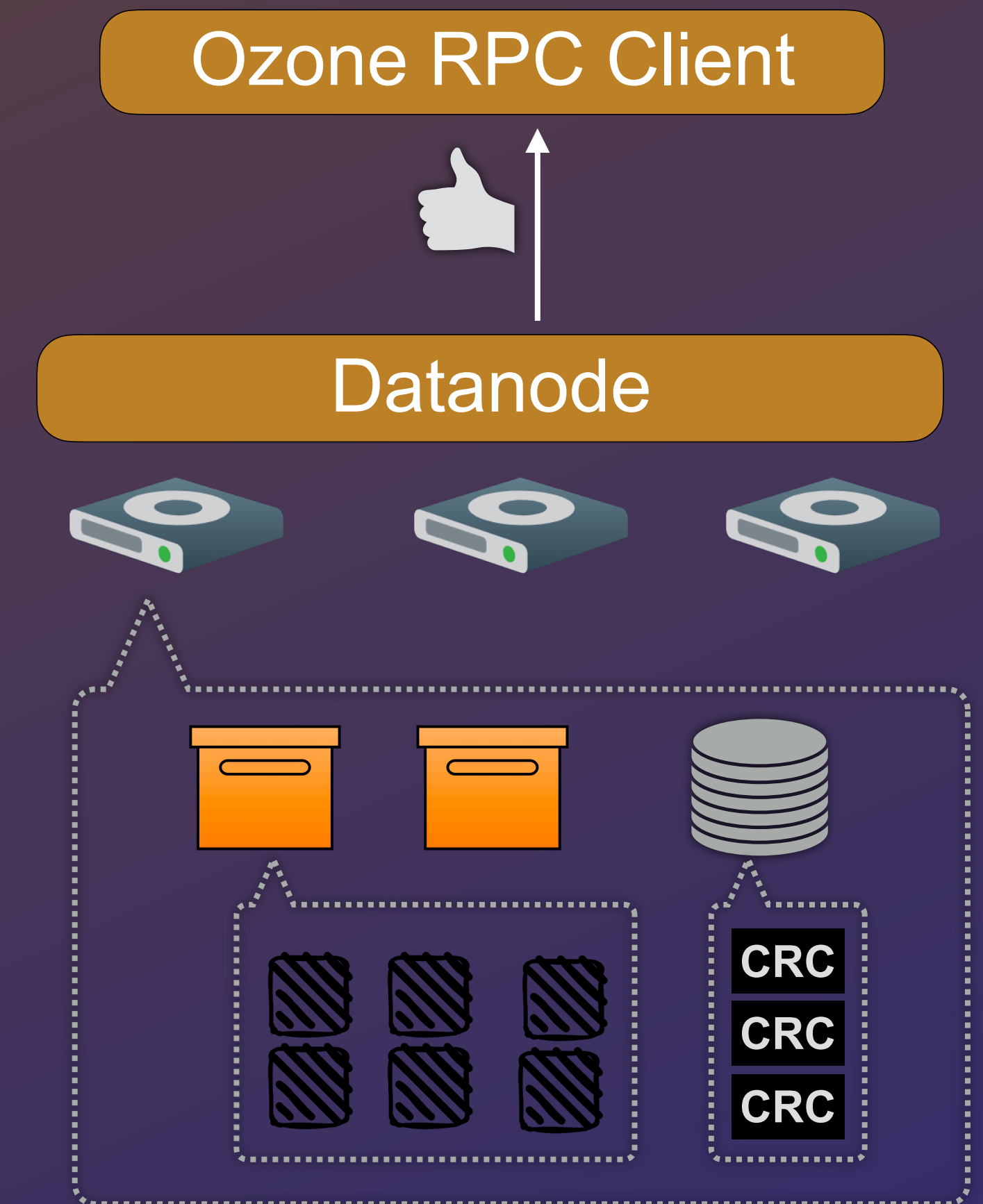
Write Checksums

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

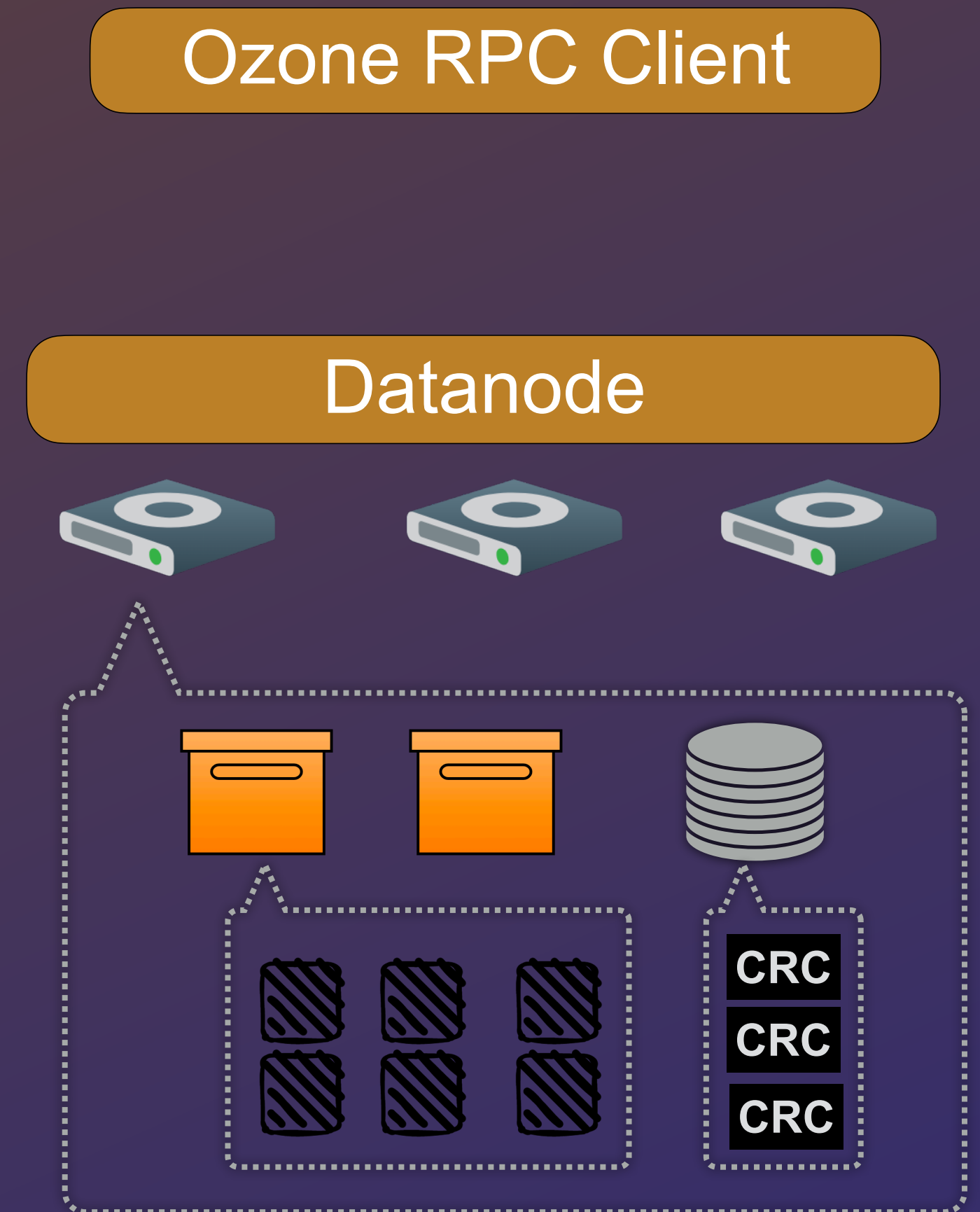


Write Checksums

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

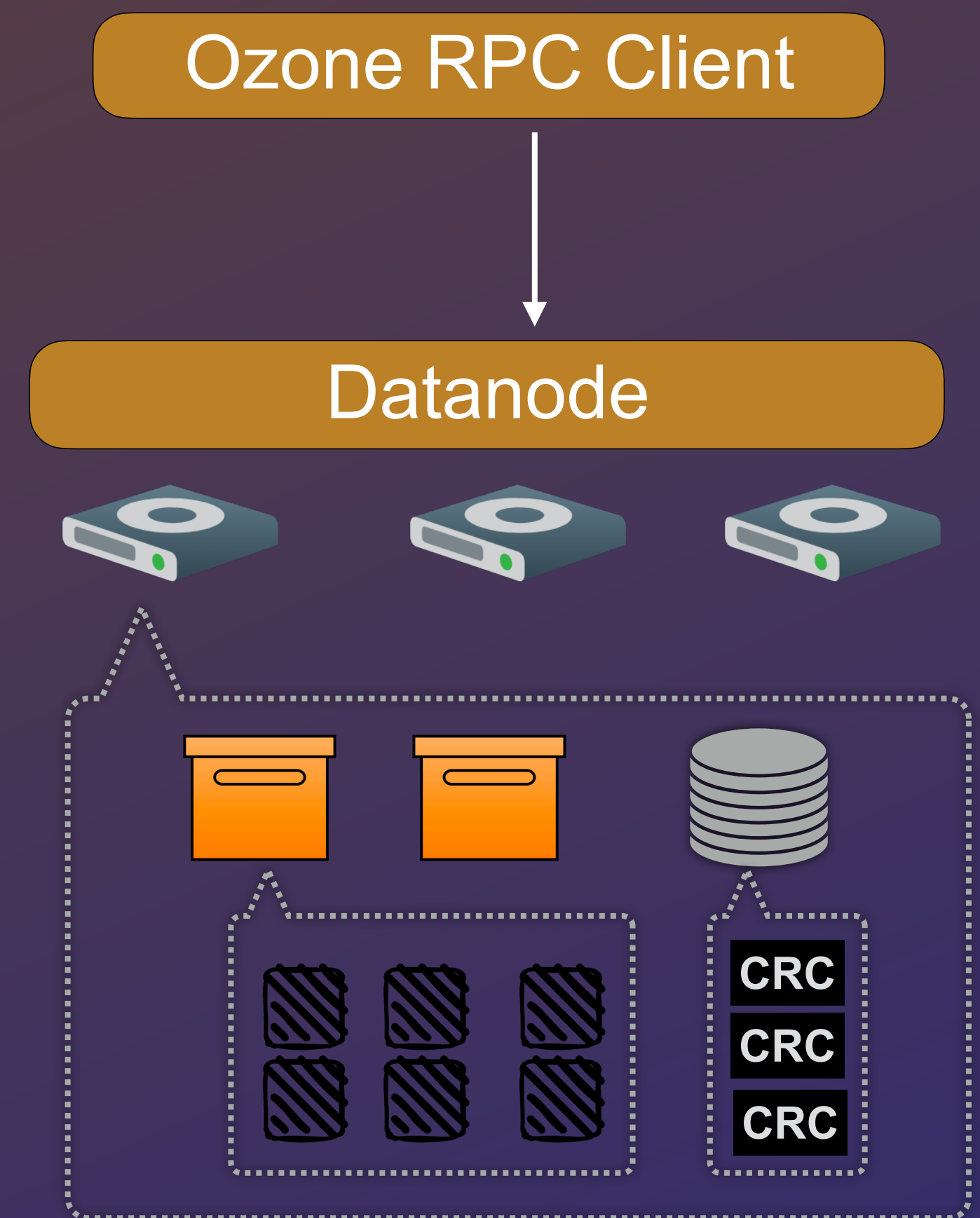


Read Checksums



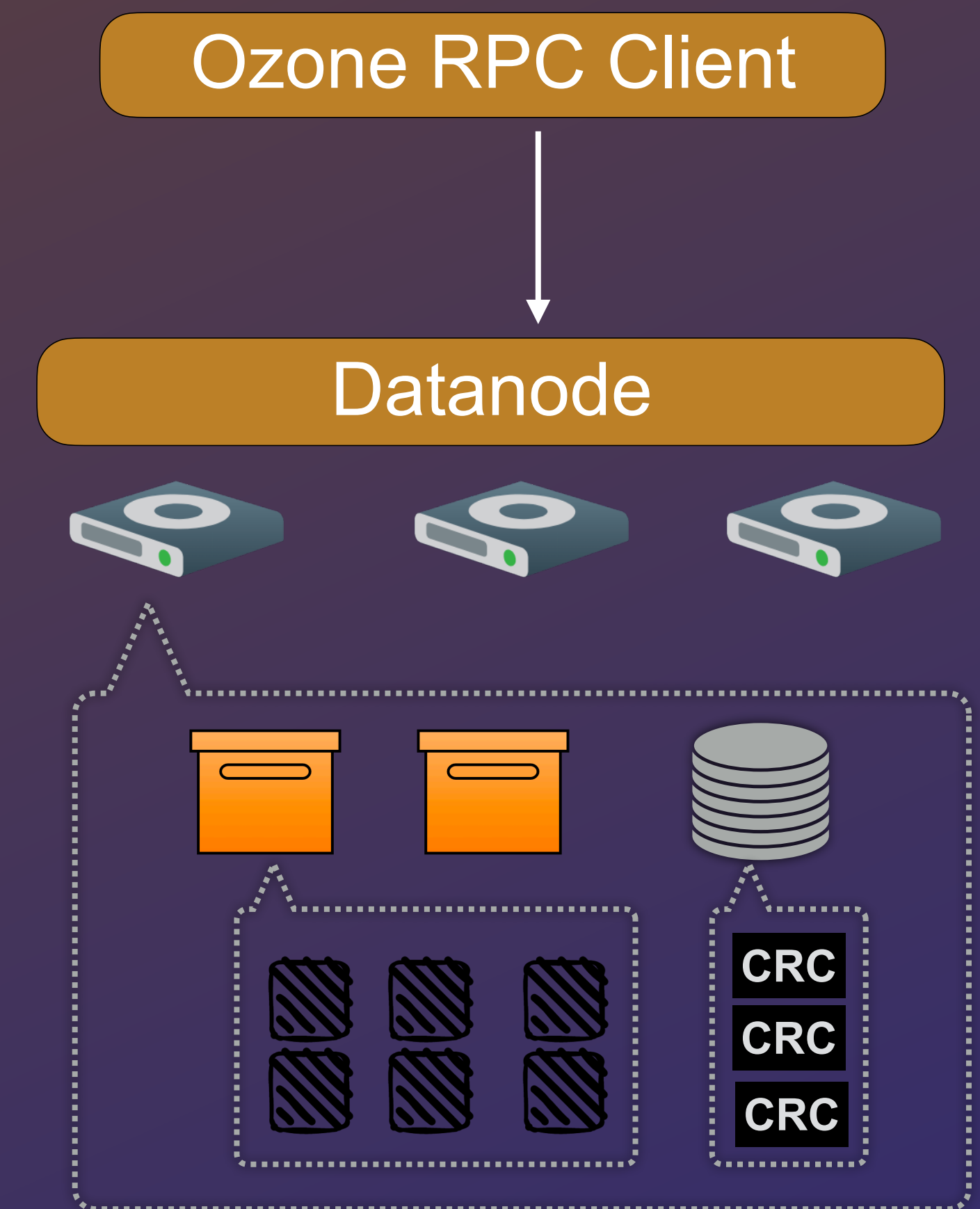
Read Checksums

1. Client requests data



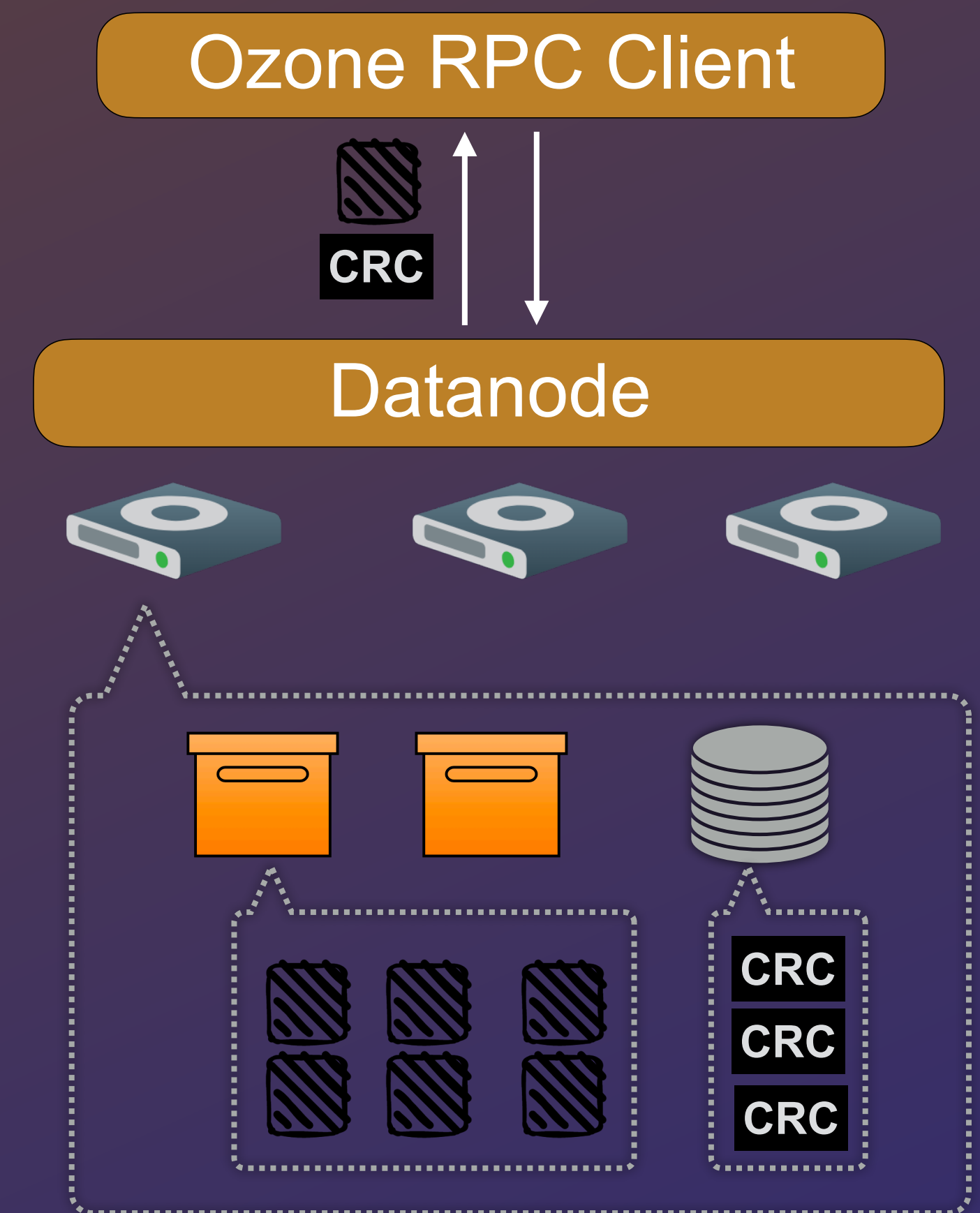
Read Checksums

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



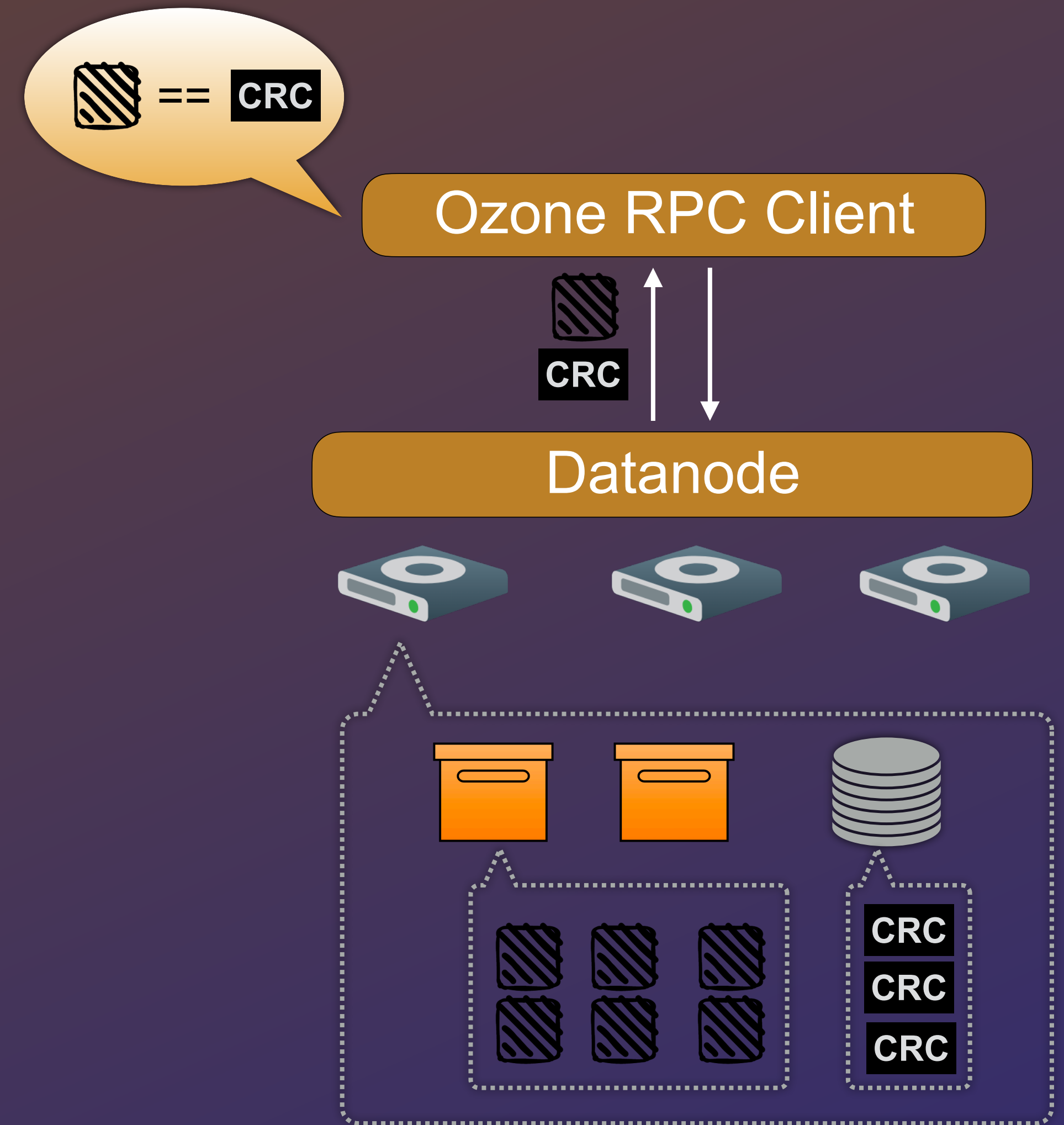
Read Checksums

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



Read Checksums

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



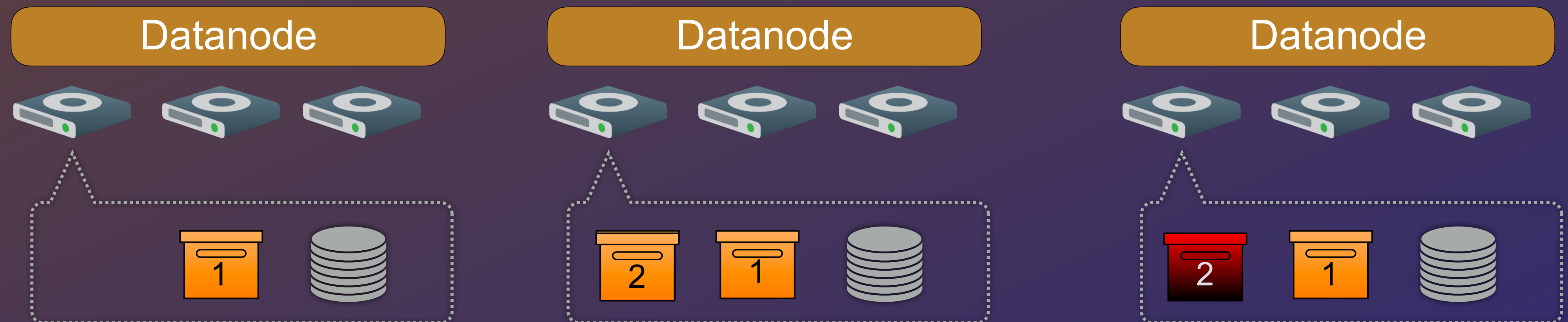
Container Scanner

- Background threads that periodically verifies checksums for each container
- Containers with checksum mismatches are marked UNHEALTHY
- UNHEALTHY containers are reported to SCM

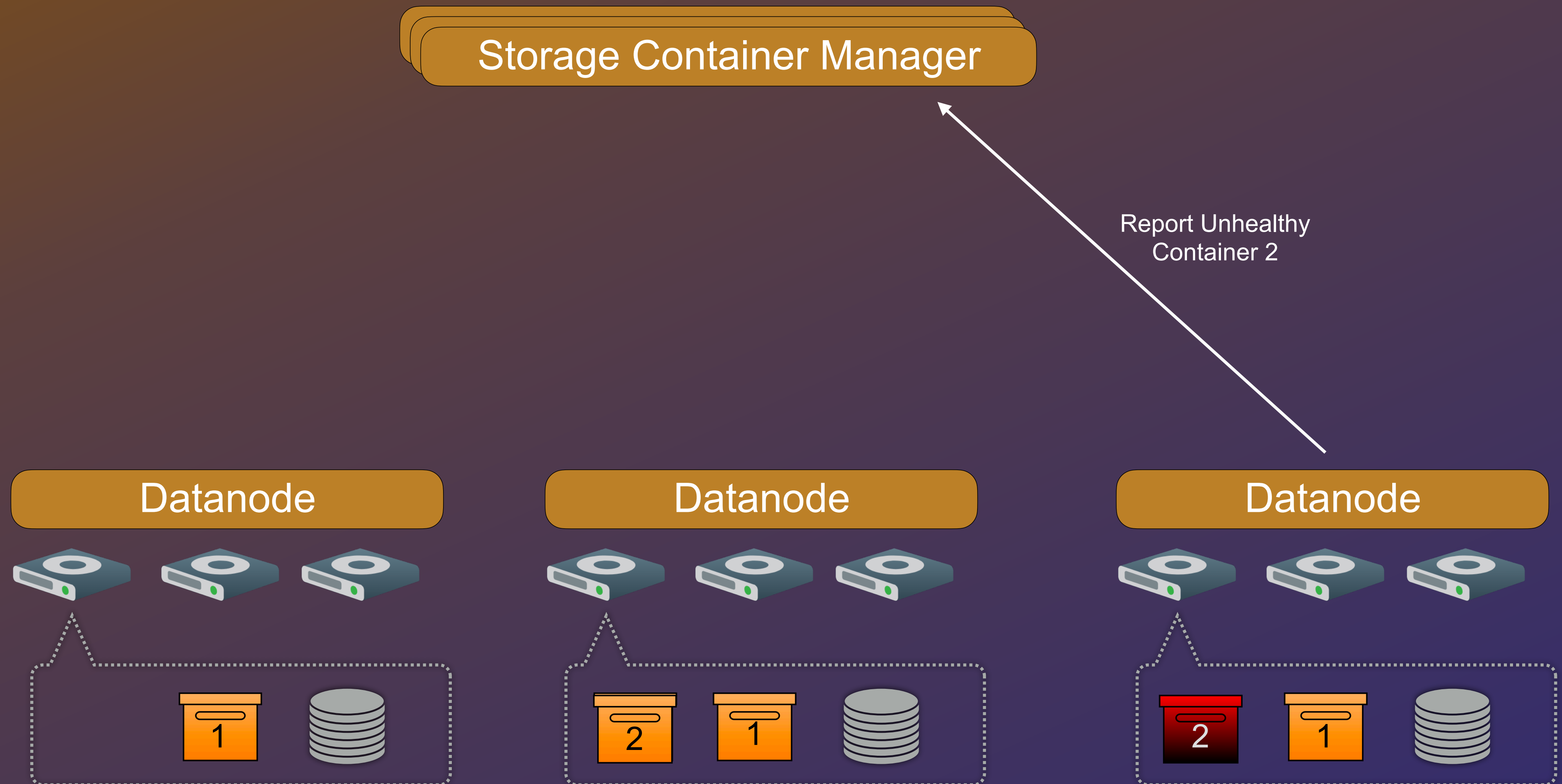


Recovering From an Unhealthy Container

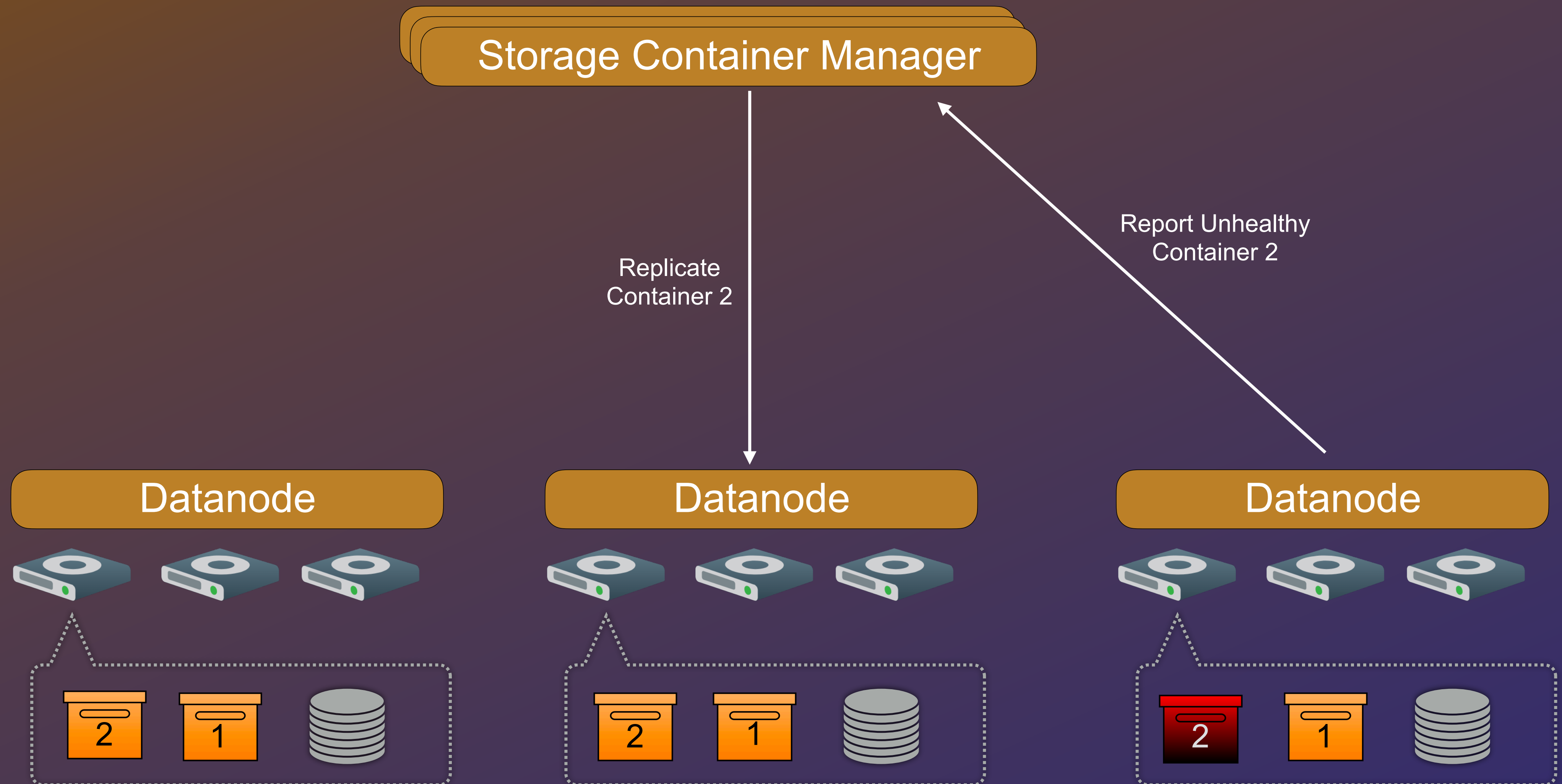
Storage Container Manager



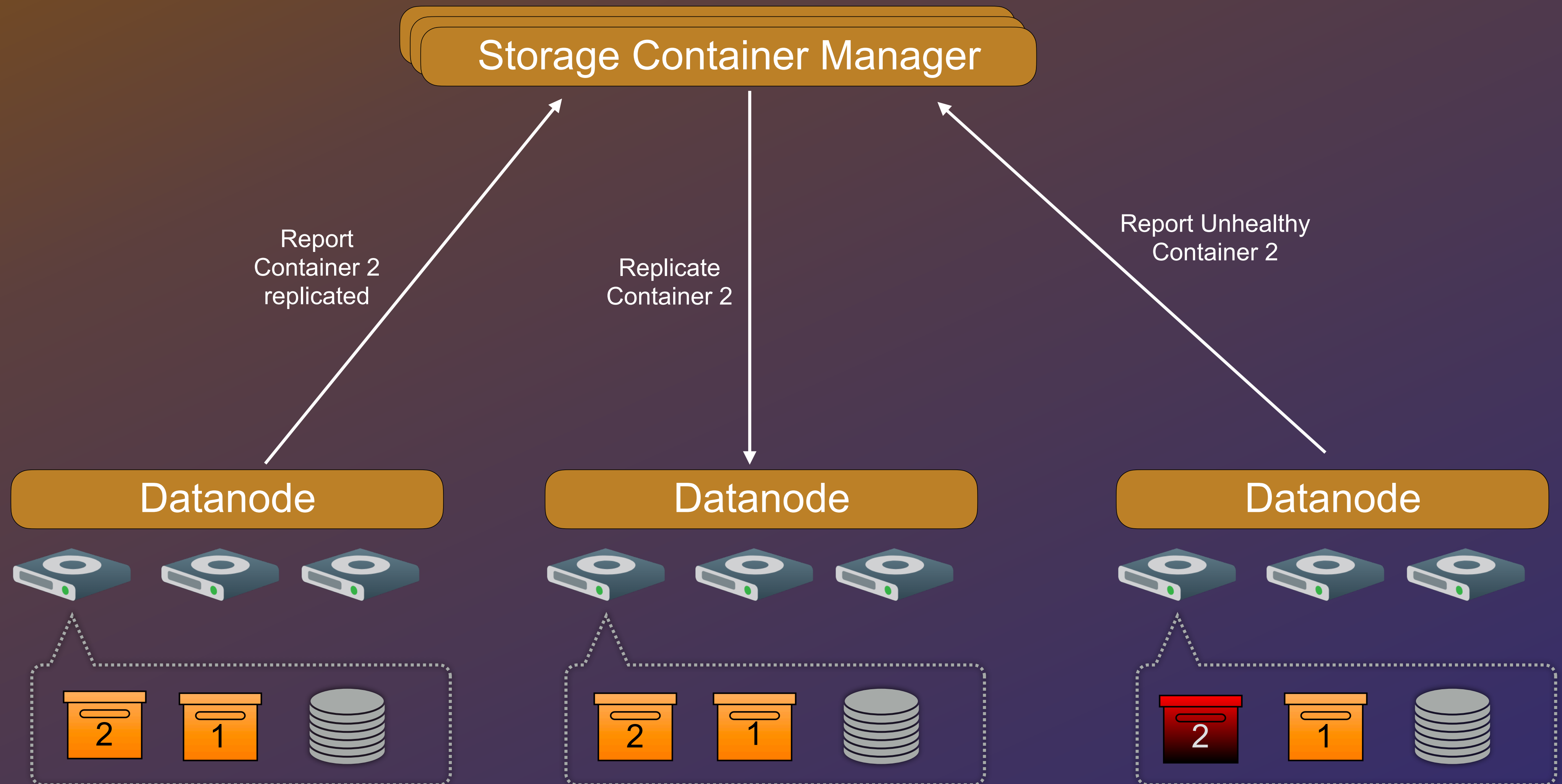
Recovering From an Unhealthy Container



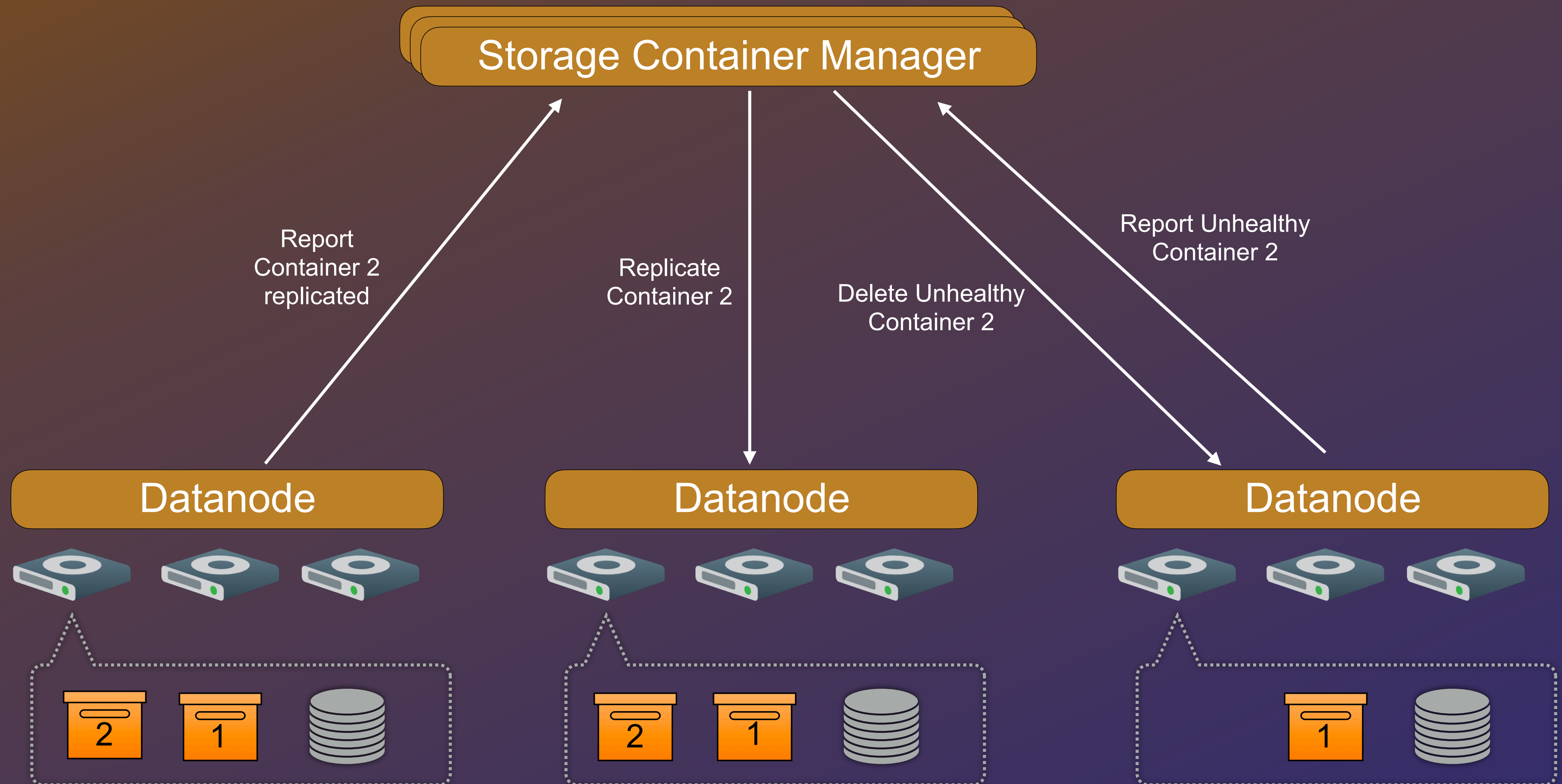
Recovering From an Unhealthy Container



Recovering From an Unhealthy Container



Recovering From an Unhealthy Container



Current Issues

SCM only sees container state without a representation of its data

False positives: UNHEALTHY container actually has good data

False negatives: CLOSED container is missing chunks or blocks

- Checksums are at chunk level only

No way out: If all replicas have issues, SCM cannot move the system to a better state.

Architecture
Goals
Tools
Implementation
Future

Goals

- SCM can take any set of containers and progress to a set of **HEALTHY CLOSED** containers.
- SCM can verify that all replicas in the set have matching data
- Datanodes can resolve discrepancies when replicas do not match

Guiding Principles

- Prioritize durability and availability
- Focus on the recovery path, not the failure path
- The system should always move itself to a safer state
- Datanodes should only make simple decisions

Architecture
Goals
Tools
Implementation
Future

Merkle Tree

- Each tree node contains a hash
- The hash is computed as a hash of its child nodes' hashes
- Provides:
 - One hash for the whole structure at the root
 - Efficient diff against other trees

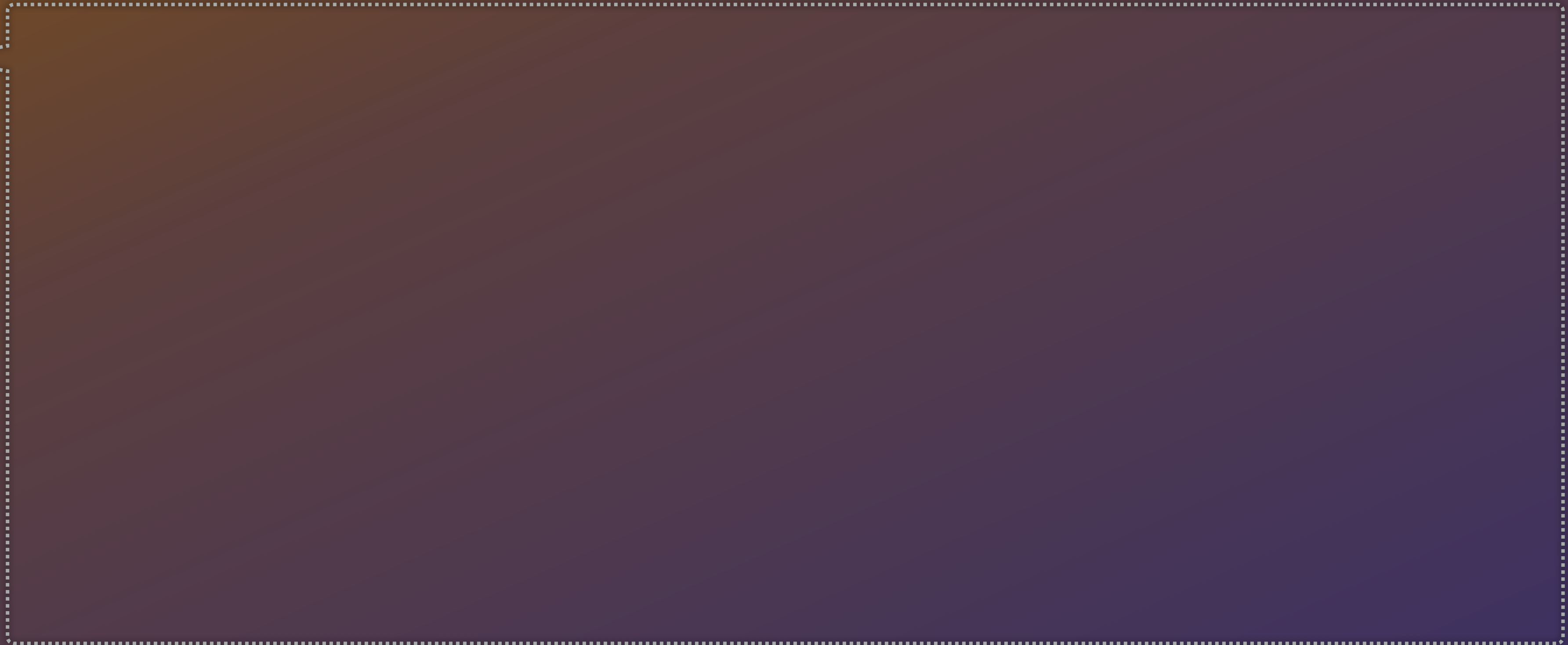
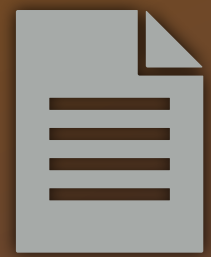
Container Merkle Tree: Build

Container Merkle Tree: Build



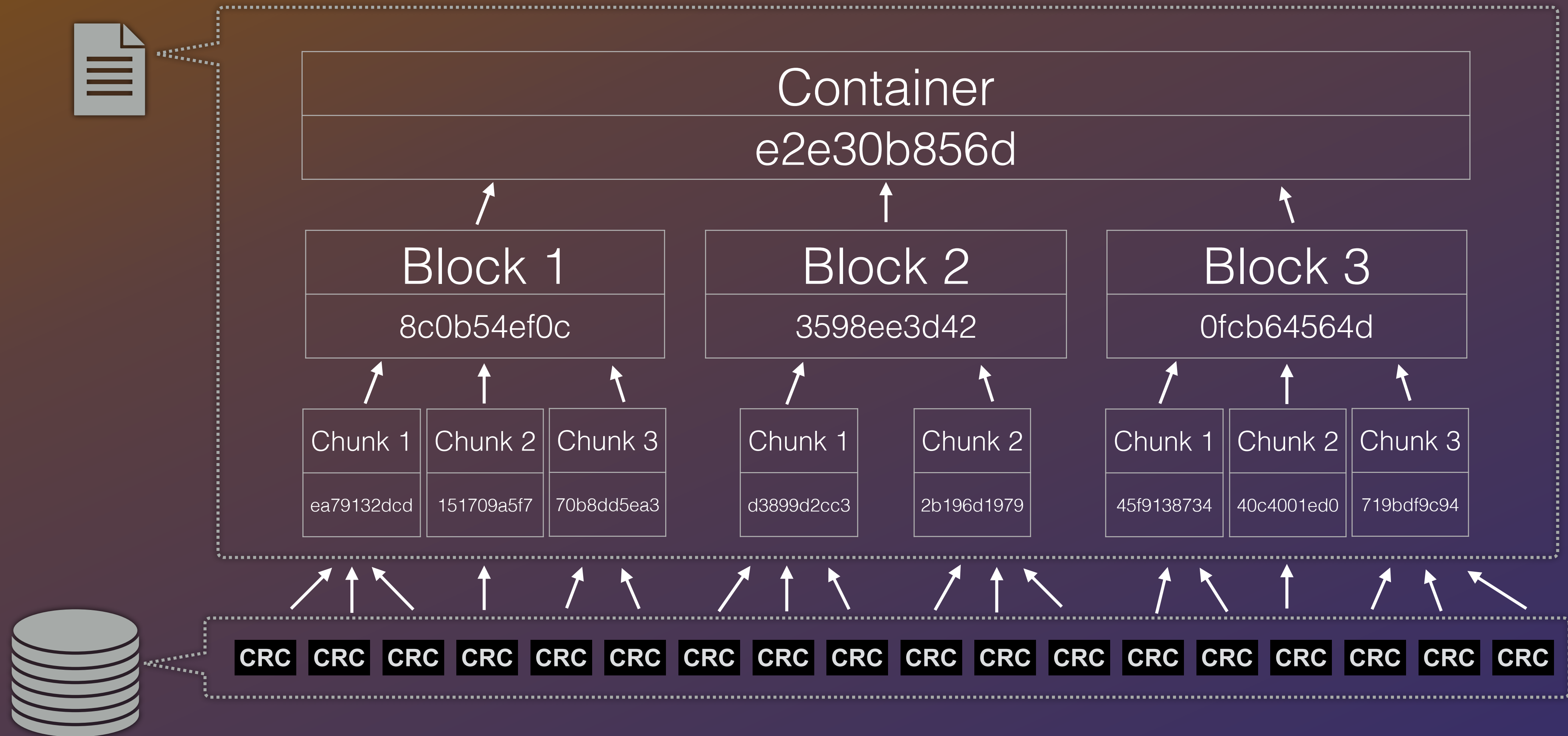
CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC

Container Merkle Tree: Build



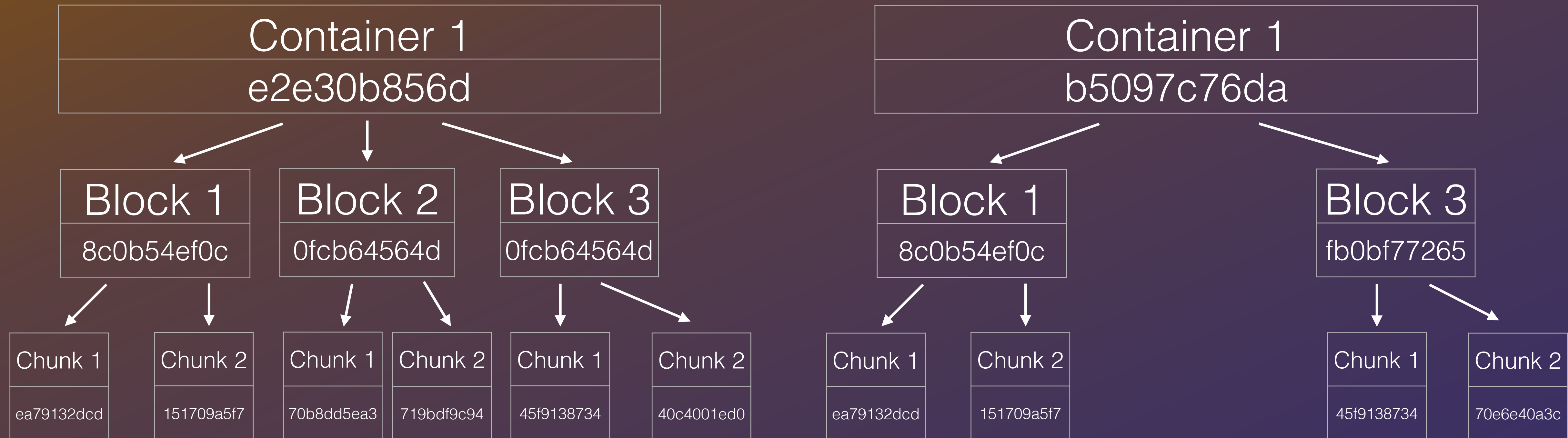
CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC CRC

Container Merkle Tree: Build



Container Merkle Tree: Diff

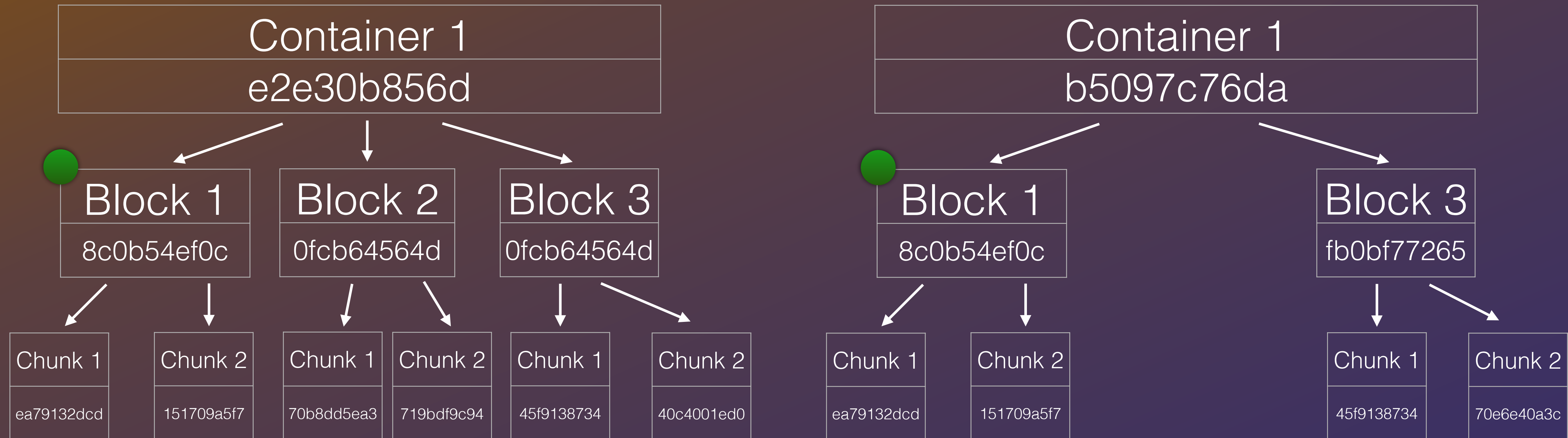
Container Merkle Tree: Diff



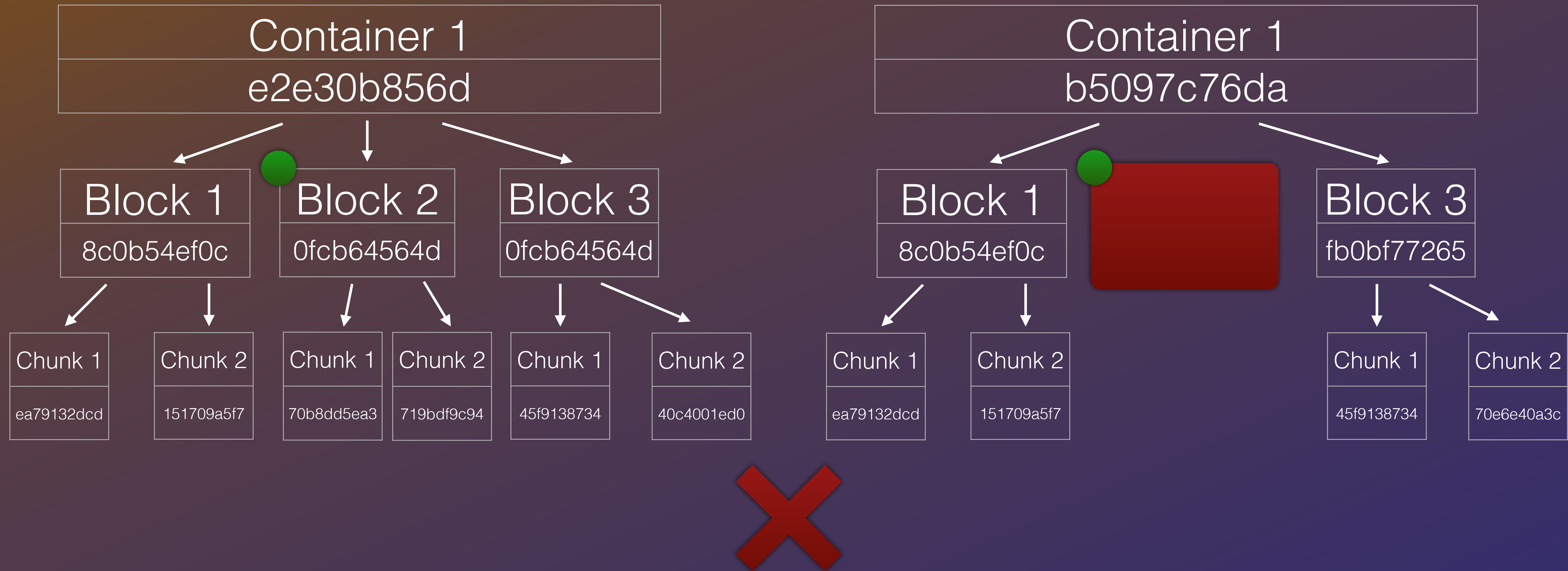
Container Merkle Tree: Diff



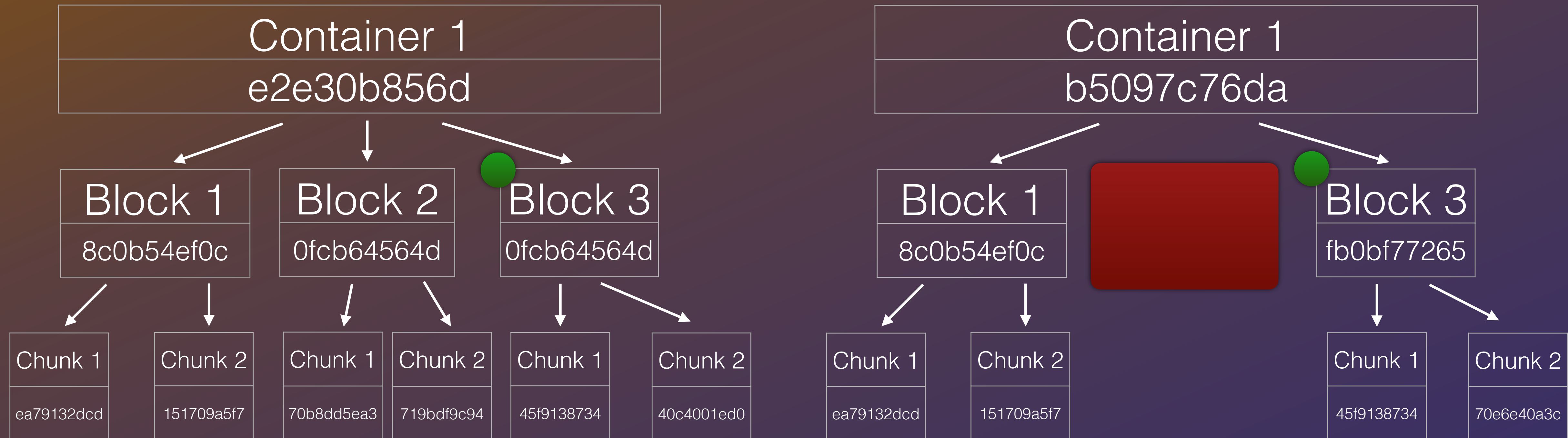
Container Merkle Tree: Diff



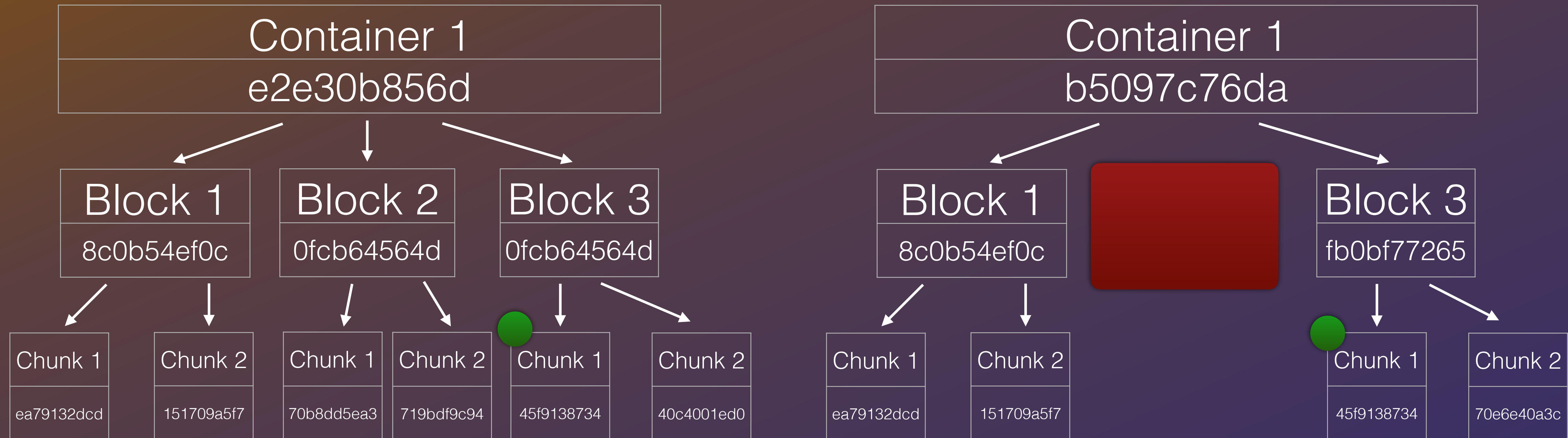
Container Merkle Tree: Diff



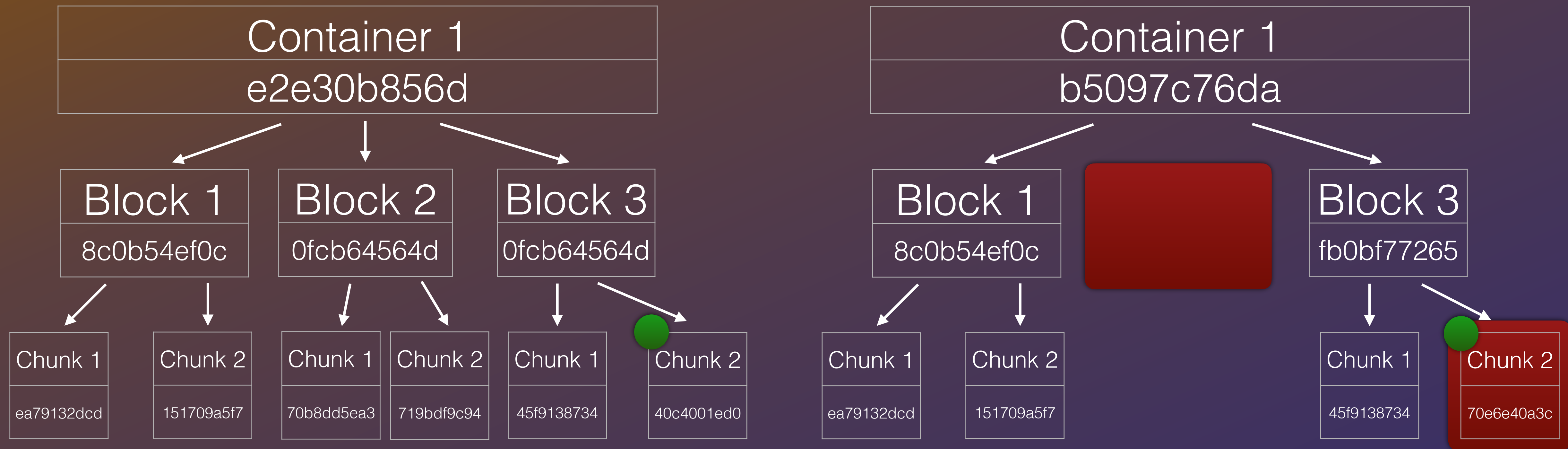
Container Merkle Tree: Diff



Container Merkle Tree: Diff



Container Merkle Tree: Diff



Conflict-Free Replicated Data Type (CRDT)

- Can be updated independently from peers
- Replicas *eventually converge* to the same state
- Provides:
 - A way to resolve independent changes to replicas

State-Based CRDT

- Replicas converge by sending their entire state to peers
- Each peer then *merges* its state with the peers state
- Merge must be:
 - Commutative:** $a + b = b + a$
 - Associative:** $a + (b + c) = (a + b) + c$
 - Idempotent:** $a + a = a$

Container State-Based CRDT

Container State-Based CRDT

- **Problem:** All peers would send 5GB container to each other.

Container State-Based CRDT

- **Problem:** All peers would send 5GB container to each other.

TOO BIG!

Container State-Based CRDT

- **Problem:** All peers would send 5GB container to each other.

TOO BIG!

- **Solution:** Send an efficient diff instead.

Container State-Based CRDT

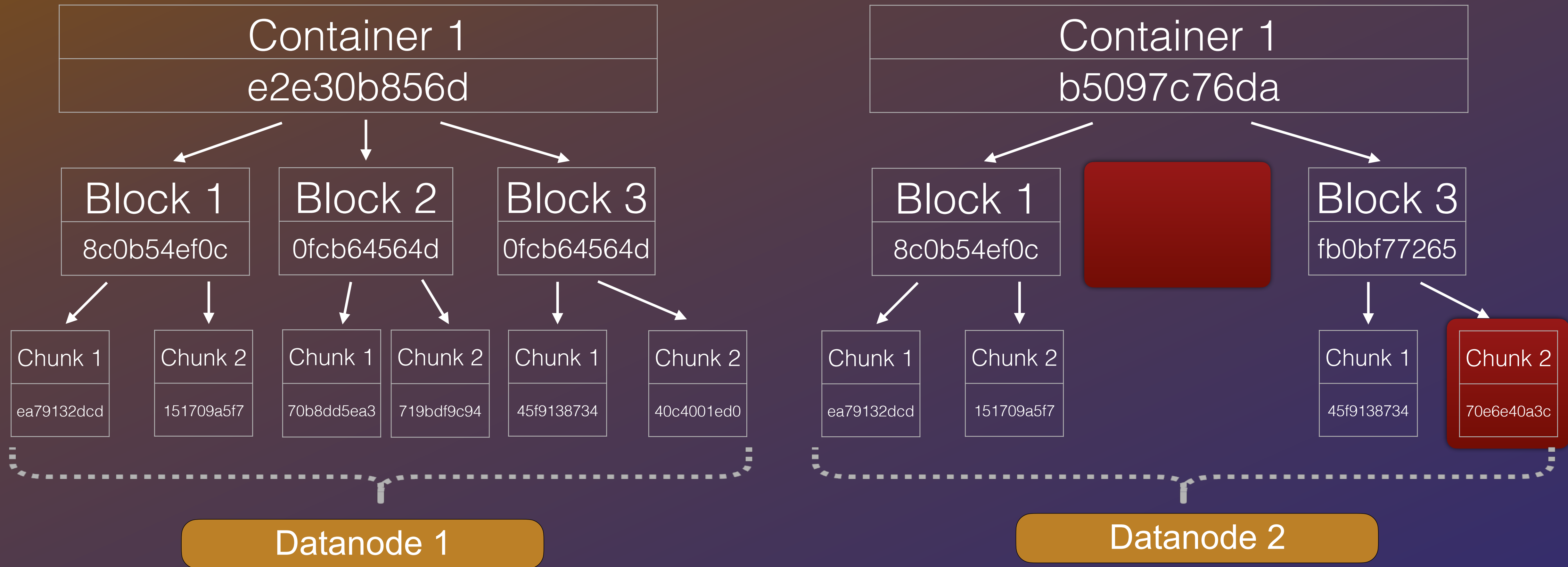
- **Problem:** All peers would send 5GB container to each other.

TOO BIG!

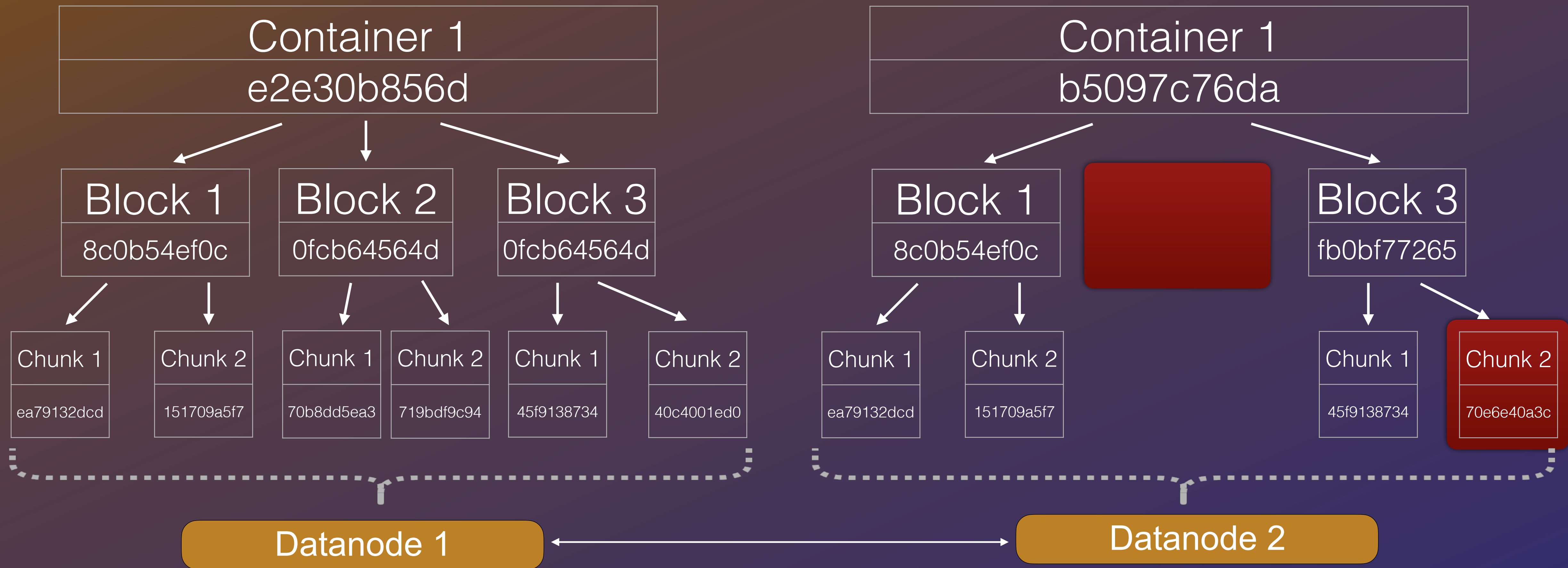
- **Solution:** Send an efficient diff instead.

The Merkle Tree

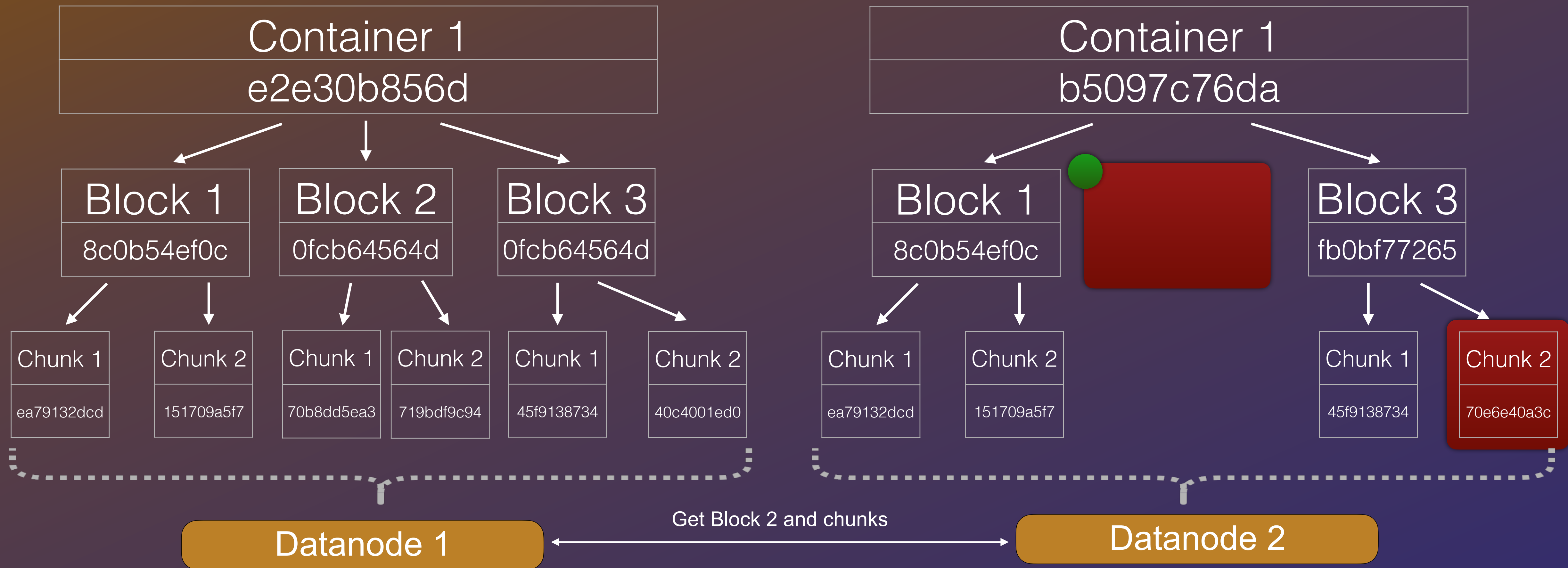
Merging Container CRDTs



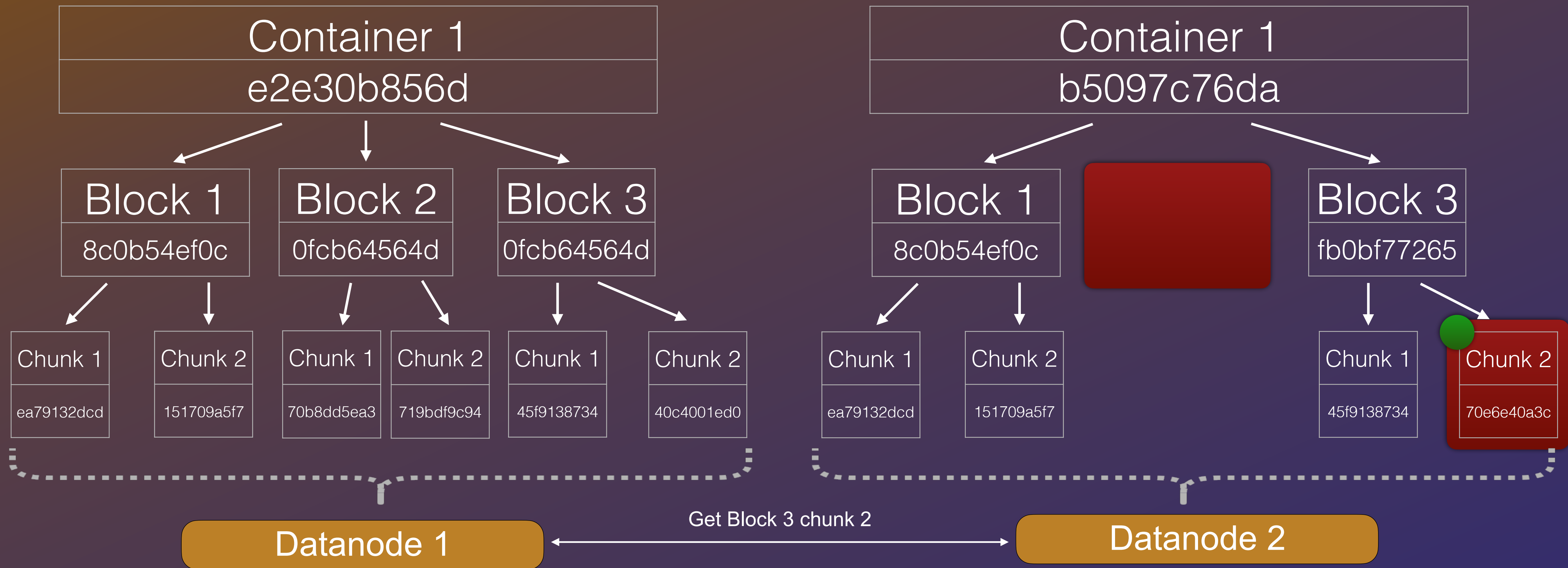
Merging Container CRDTs



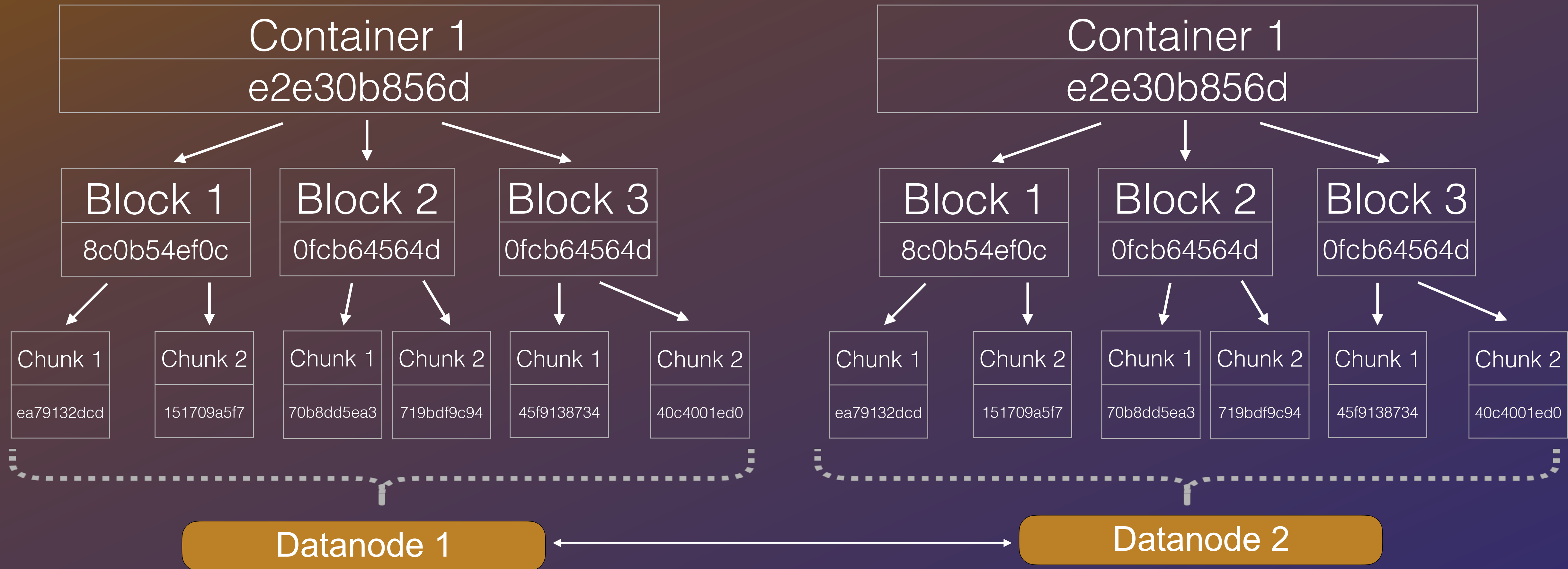
Merging Container CRDTs



Merging Container CRDTs



Merging Container CRDTs



Architecture
Goals
Tools
Implementation
Future

Recognize

Storage Container Manager

Datanode 1

Datanode 2

Datanode 3

Recognize

1. Datanode container scanner walks all data

Storage Container Manager

Datanode 1



Datanode 2

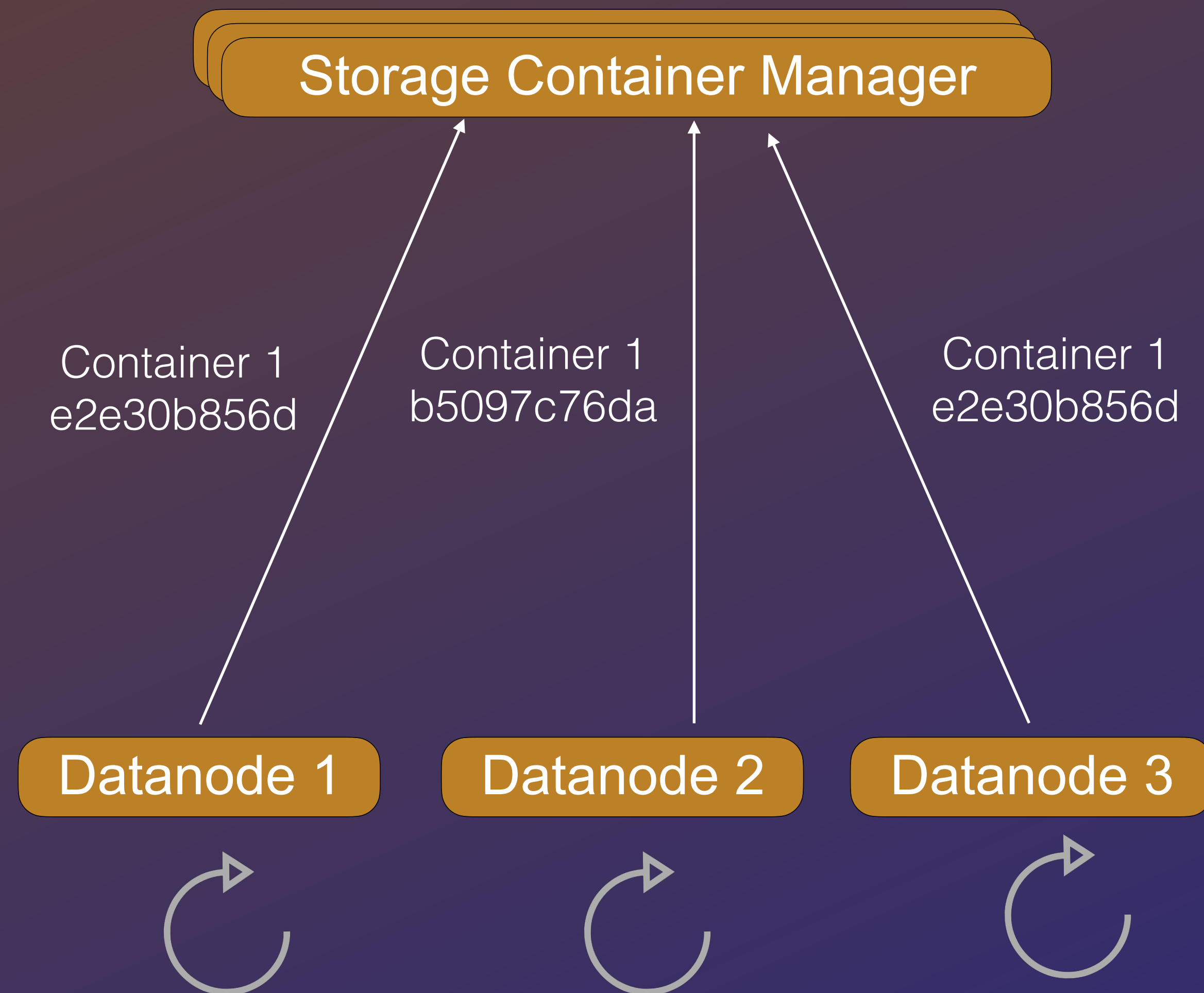


Datanode 3



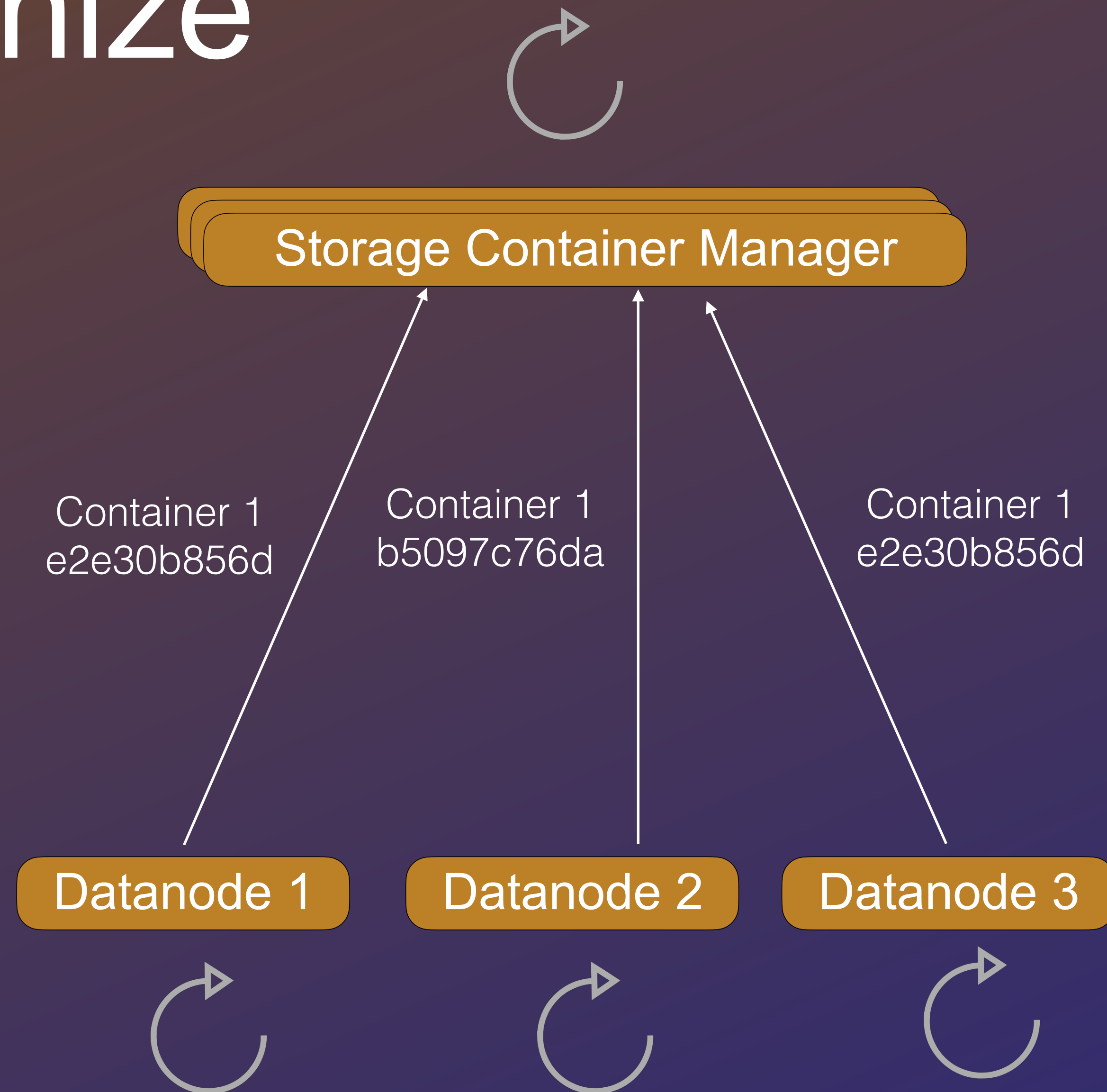
Recognize

1. Datanode container scanner walks all data
2. Datanode reports root hash of each container to SCM



Recognize

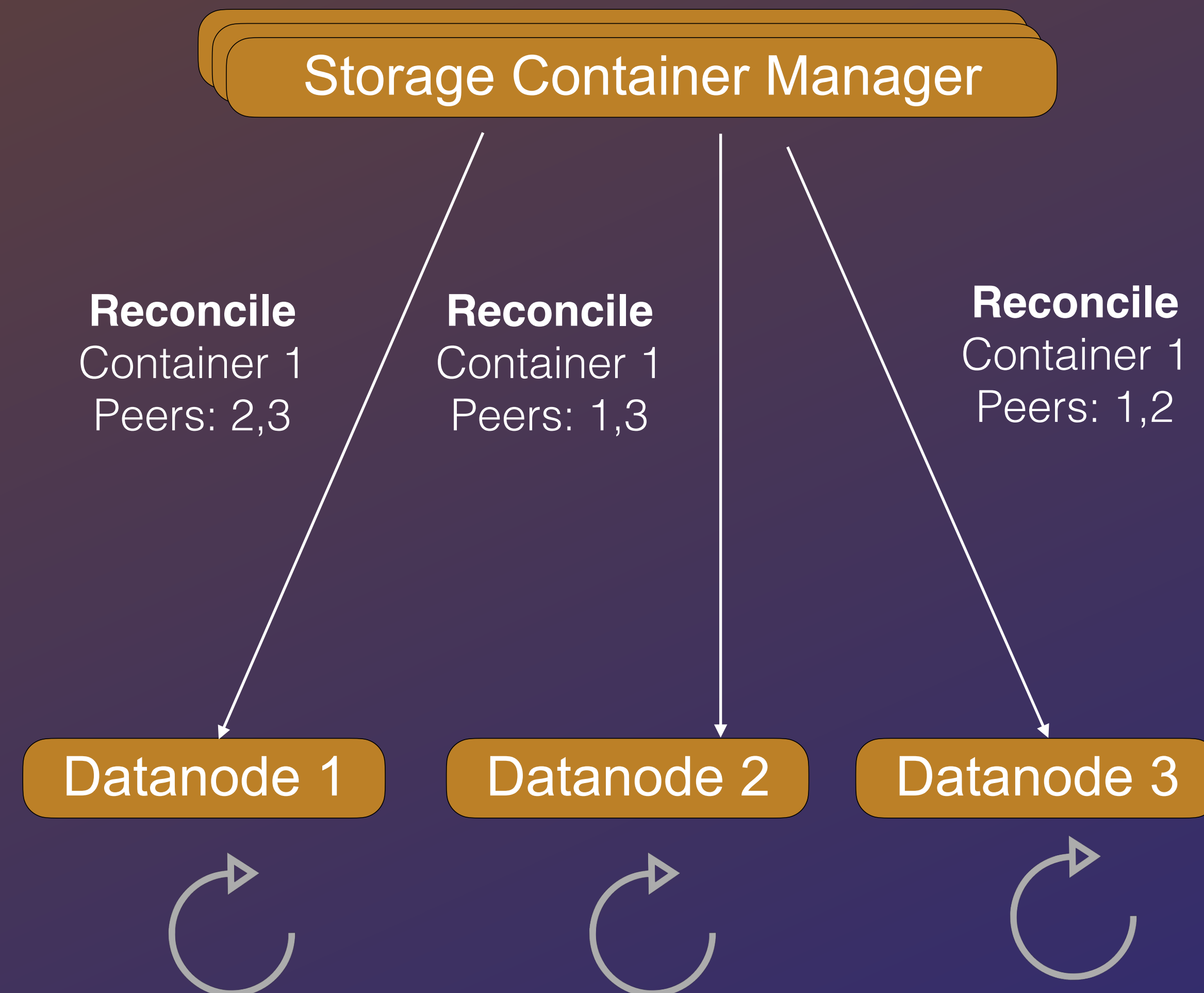
1. Datanode container scanner walks all data
2. Datanode reports root hash of each container to SCM
3. SCM sees replicas with mismatched hashes



Recognize



1. Datanode container scanner walks all data
2. Datanode reports root hash of each container to SCM
3. SCM sees replicas with mismatched hashes
4. SCM sends them reconcile commands



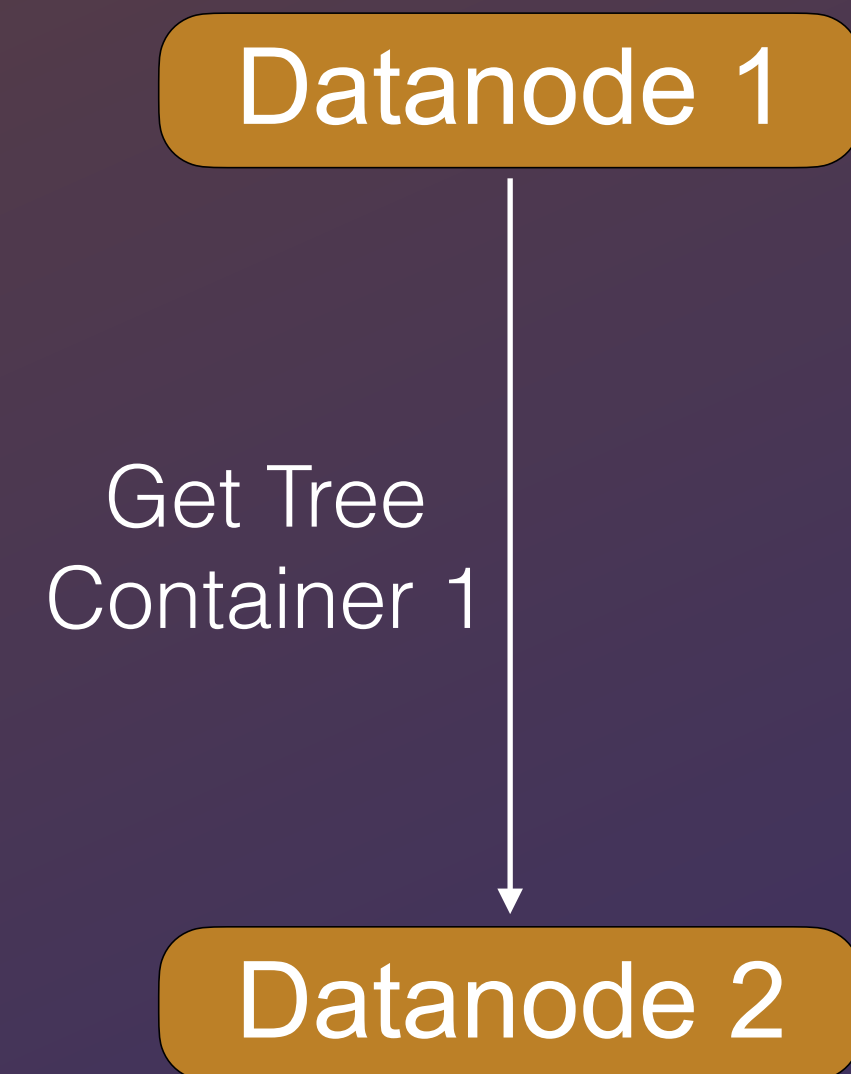
Reconcile

Datanode 1

Datanode 2

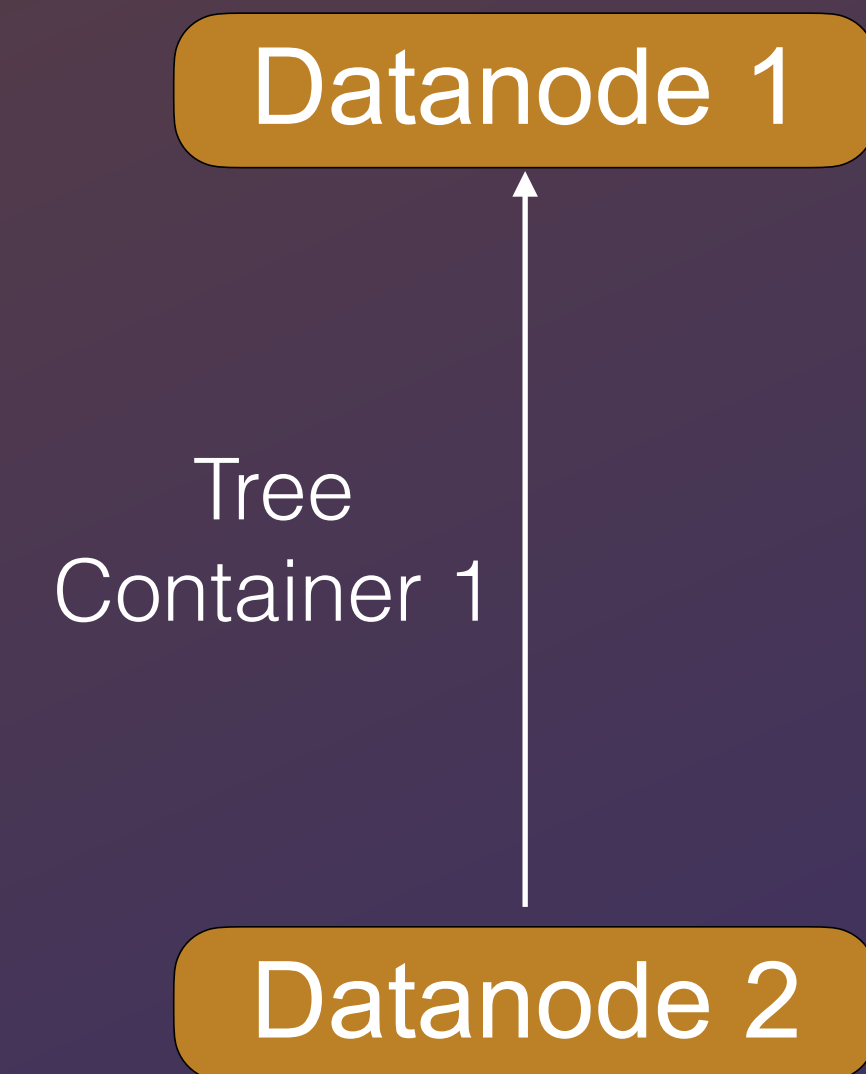
Reconcile

1. Datanode reads container merkle tree from first peer



Reconcile

1. Datanode reads container merkle tree from first peer



Reconcile

1. Datanode reads container merkle tree from first peer
2. Datanode diffs peer tree with its own tree

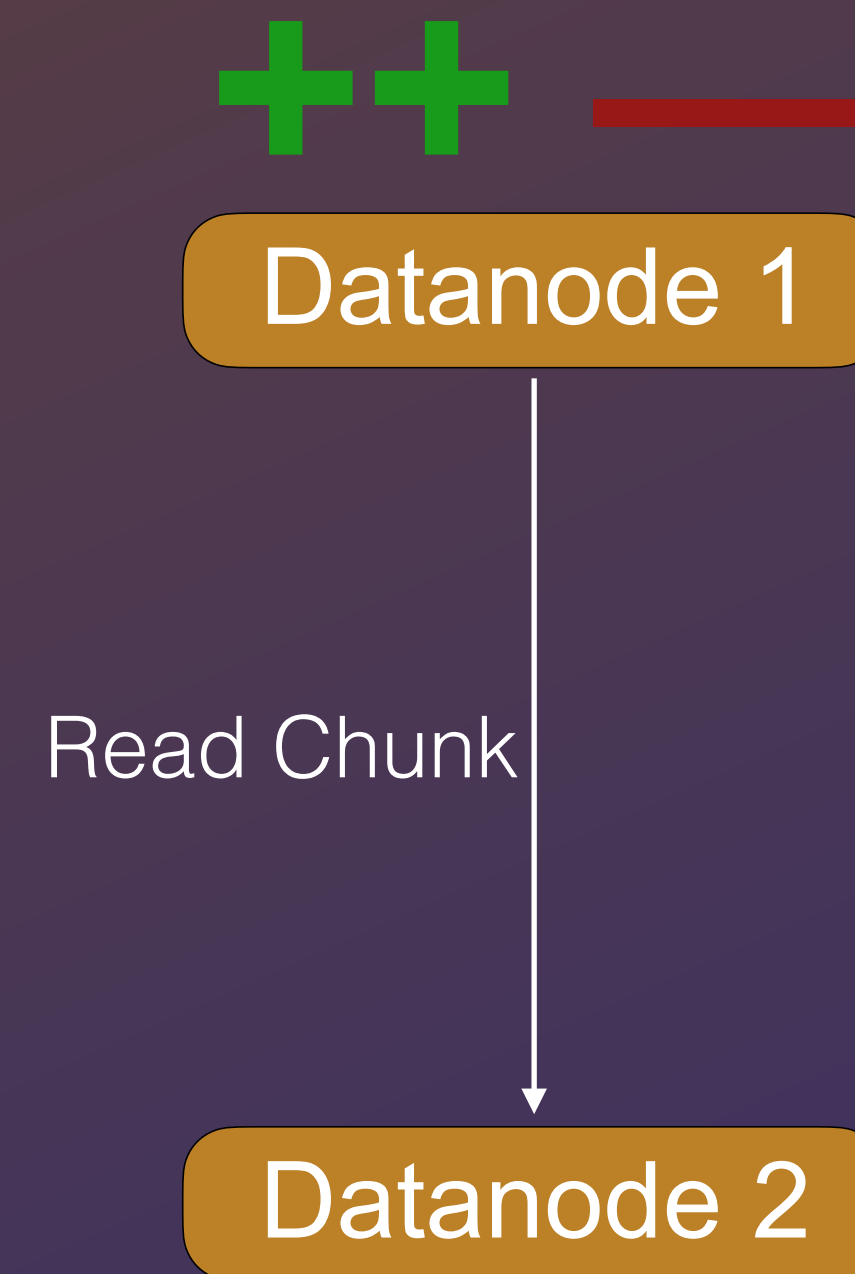


Datanode 1

Datanode 2

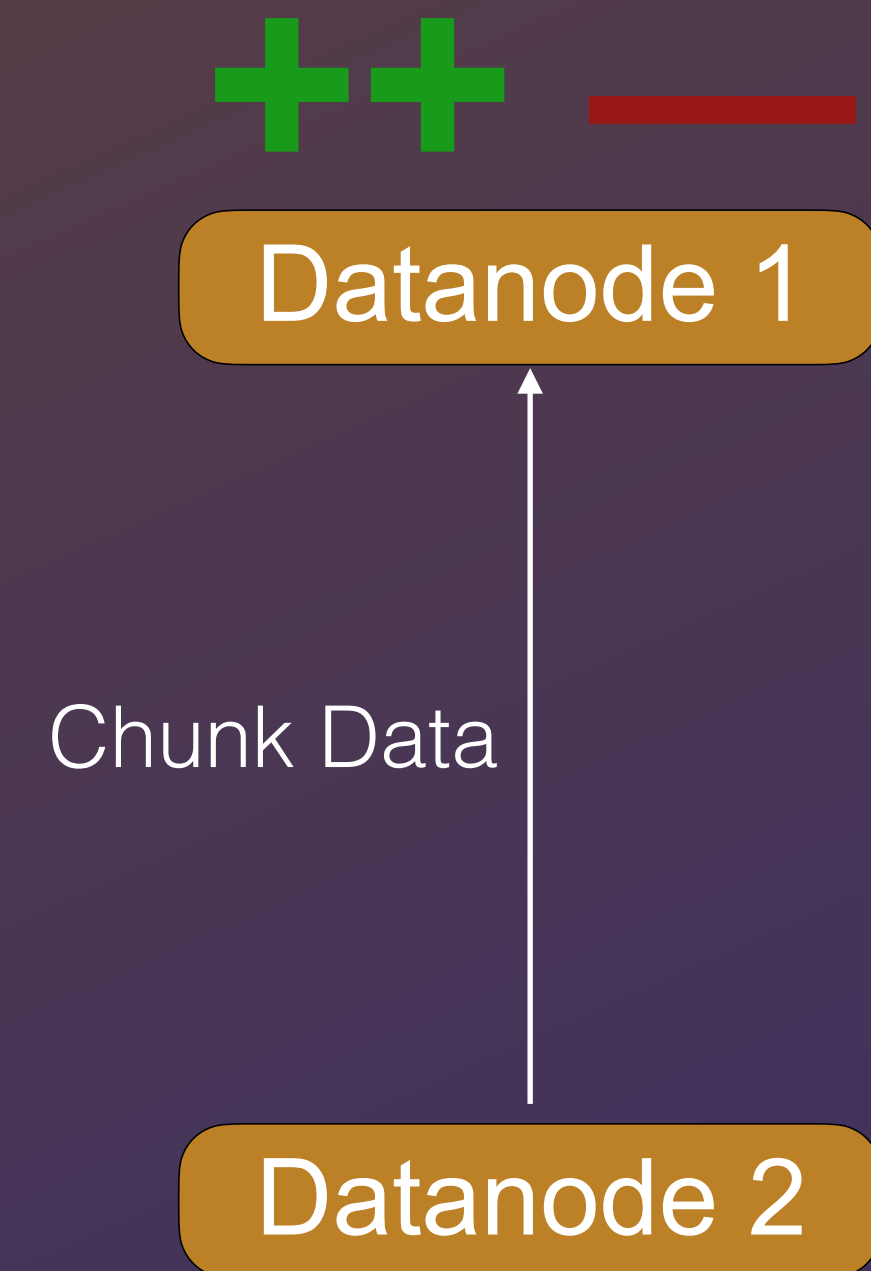
Reconcile

1. Datanode reads container merkle tree from first peer
2. Datanode diffs peer tree with its own tree
3. Datanode pulls missing data from peer



Reconcile

1. Datanode reads container merkle tree from first peer
2. Datanode diffs peer tree with its own tree
3. Datanode pulls missing data from peer



Reconcile

1. Datanode reads container merkle tree from first peer
2. Datanode diffs peer tree with its own tree
3. Datanode pulls missing data from peer
4. Datanode patches its own container



Datanode 1

Datanode 2

Repeat

Storage Container Manager

Datanode 1

Datanode 2

Datanode 3

Repeat

1. Datanodes repeat the reconcile step for all peers in any order

Storage Container Manager

Datanode 1

Datanode 2

Datanode 3



Repeat

1. Datanodes repeat the reconcile step for all peers in any order



Storage Container Manager

The diagram illustrates a distributed system architecture. At the top, a yellow rounded rectangle labeled 'Storage Container Manager' is shown with a slight 3D effect. Below it, three yellow rounded rectangles labeled 'Datanode 1', 'Datanode 2', and 'Datanode 3' are arranged horizontally. Curved white arrows connect the datanodes in a circular fashion: from Datanode 1 to Datanode 2, from Datanode 2 to Datanode 3, and from Datanode 3 back to Datanode 1.

Datanode 1

Datanode 2

Datanode 3

Repeat

1. Datanodes repeat the reconcile step for all peers in any order

Storage Container Manager

Datanode 1

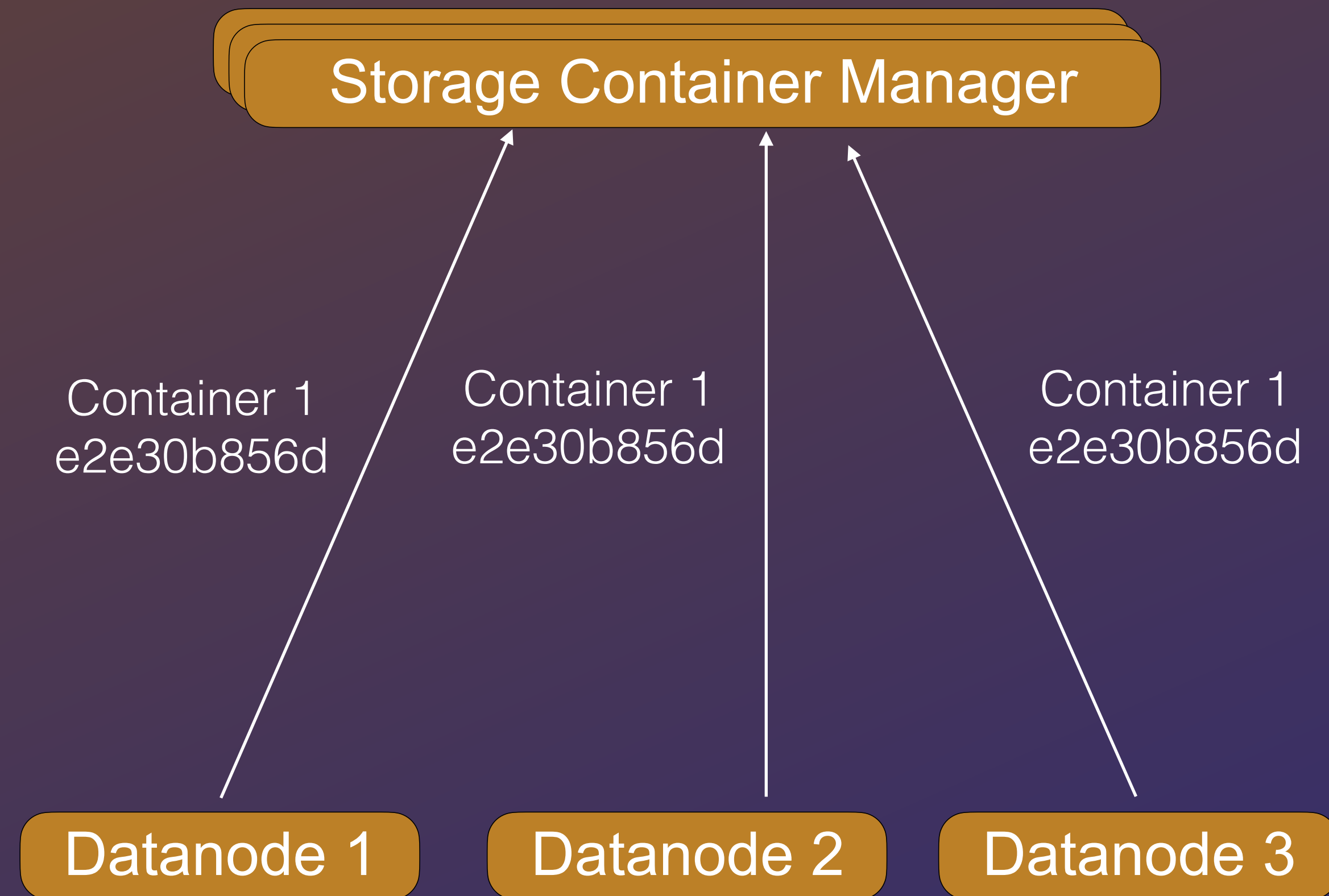
Datanode 2

Datanode 3



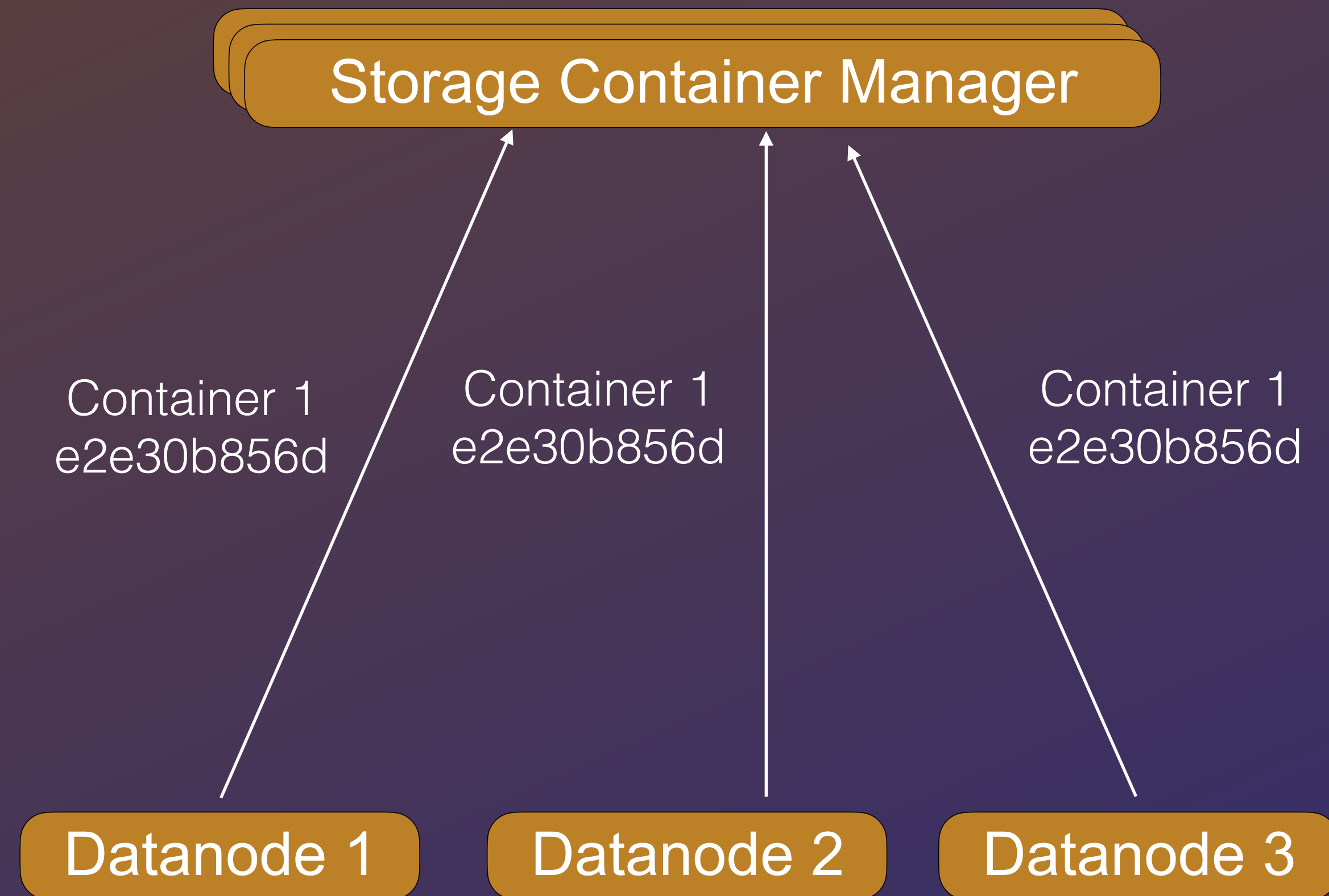
Repeat

1. Datanodes repeat the reconcile step for all peers in any order
2. Datanodes report new hash to SCM



Repeat

1. Datanodes repeat the reconcile step for all peers in any order
2. Datanodes report new hash to SCM
3. Either:
 - All datanodes end up with identical replicas
 - The reconcile step is retried



Architecture
Goals
Tools
Implementation
Future

SCM Replication Manager Integration

- **Phase 1** of reconciliation must be manually triggered from CLI
- **Phase 2** of development will trigger it automatically from SCM
- Simplifies SCM's replication manager when all replicas are bad

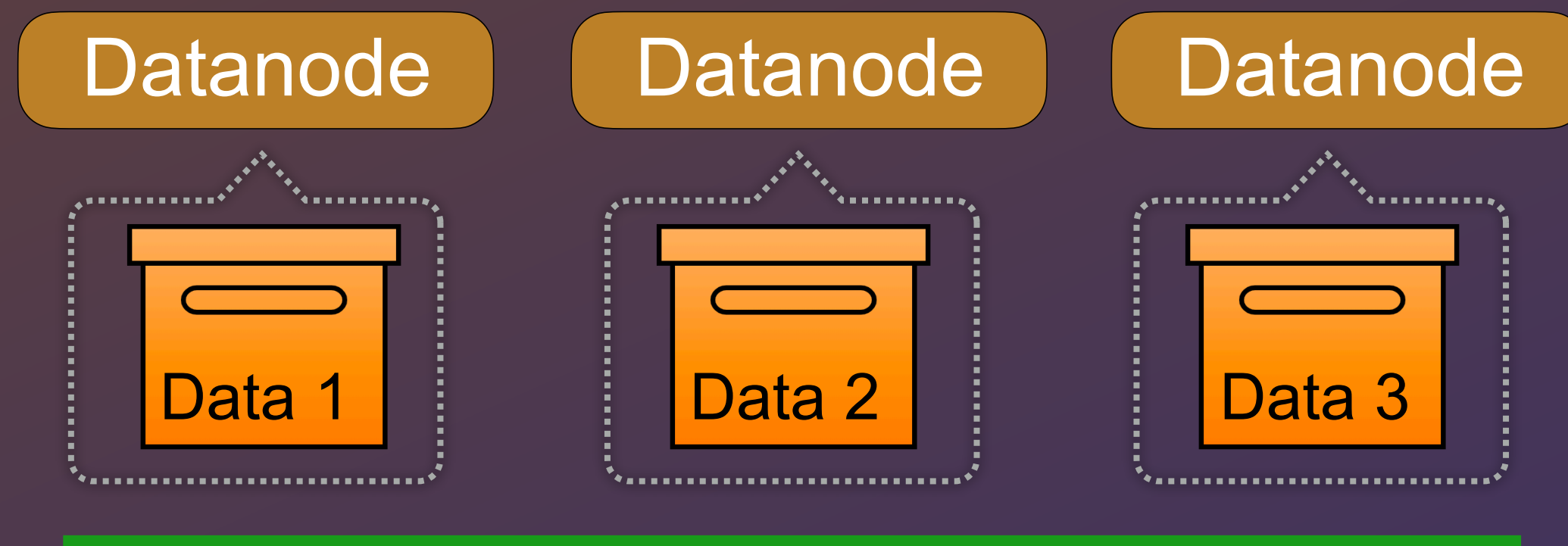
Erasure Coding (EC)

Erasure Coding (EC)

- Replicas are not identical

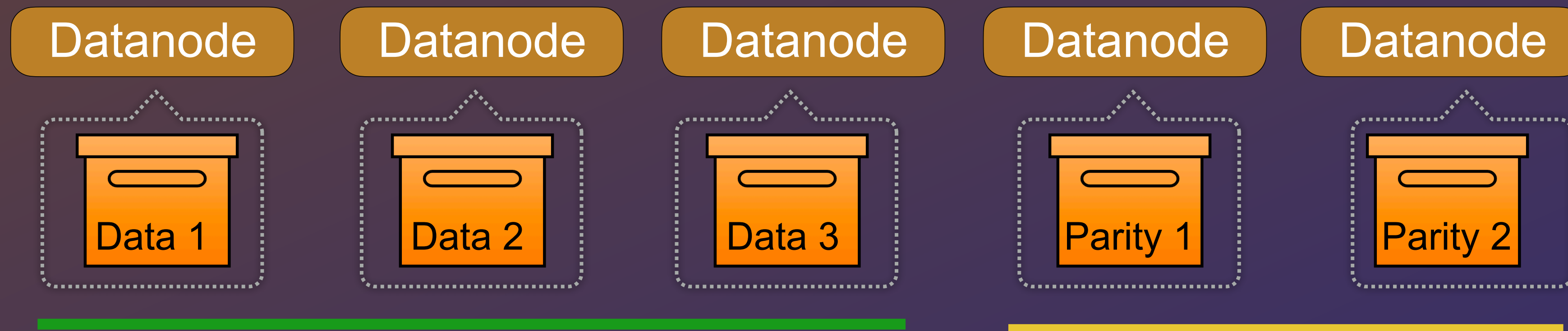
Erasure Coding (EC)

- Replicas are not identical



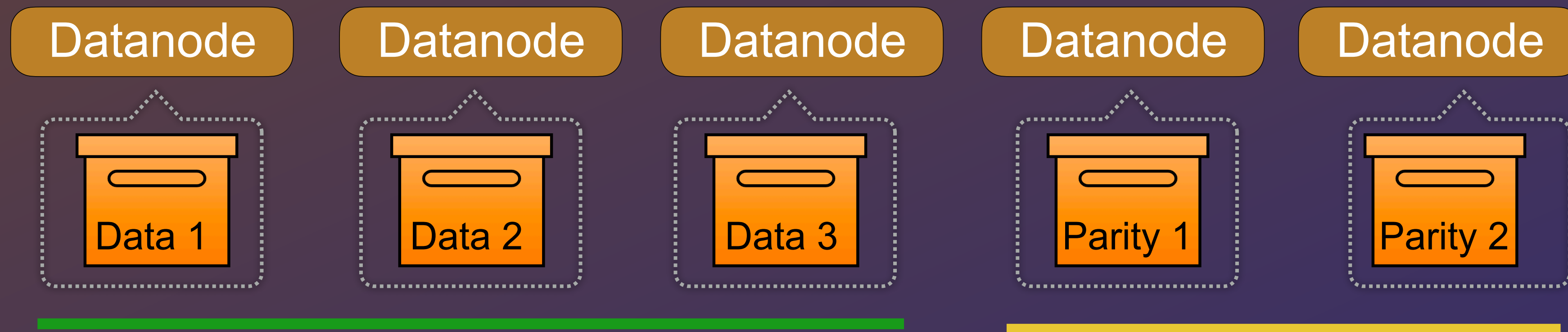
Erasure Coding (EC)

- Replicas are not identical



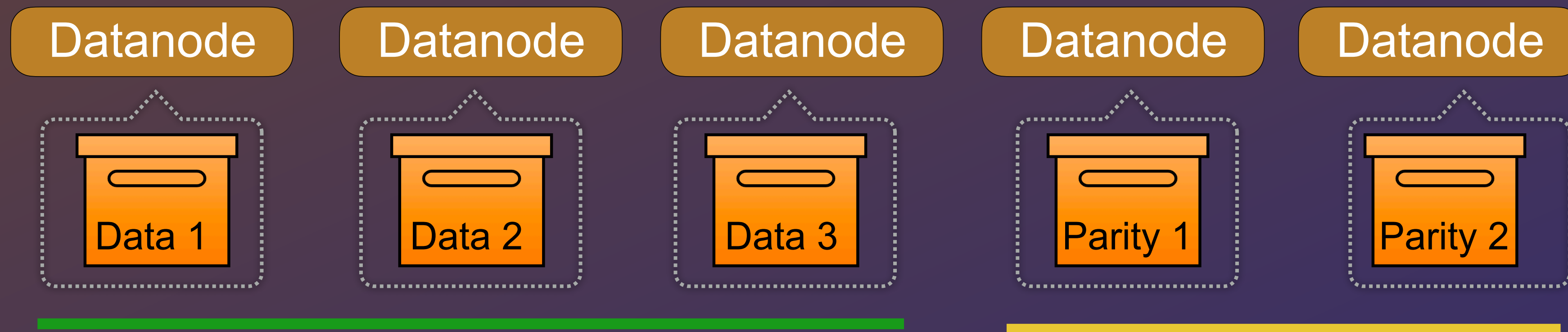
Erasure Coding (EC)

- Replicas are not identical
- Already has a merge algorithm: reconstruction



Erasure Coding (EC)

- Replicas are not identical
- Already has a merge algorithm: reconstruction
- How to proactively identify if a replica has diverged?



EC Container Merkle Tree?

EC Container Merkle Tree?

- Given:
 - **Chunk checksums:** Verifies integrity within a replica
 - **Stripe checksum:** Verifies integrity across replicas
 - Only useful after reconstruction, not before

EC Container Merkle Tree?

- Given:
 - **Chunk checksums:** Verifies integrity within a replica
 - **Stripe checksum:** Verifies integrity across replicas
 - Only useful after reconstruction, not before
- Ideas:
 - **Block counts** should be the same across replicas
 - Hash these into a container hash for early error detection?

Questions?

erose@apache.org

github.com/erose28

<https://issues.apache.org/jira/browse/HDDS-10239>

(Pull Request contains design doc)