Ozone: scaling HDFS to trillions of objects

Xiaoyu Yao & Anu Engineer

Apache Committers & PMC members

HDFS

- Apache HDFS is designed for large objects – not for many small objects.
- Small files create memory pressure on namenode.
  - Metadata in memory is the strength of the original GFS and HDFS design, but also the weakness in scaling to large number of files and blocks.
- Each file adds metadata to the namenode.
  - Datanodes send all block information to namenode in BlockReports.
- Both of these create scalability issues on Namenode.

HORTONWORKS

Ozone and HDFS - past

- HDFS-5477 – Separate Block Management.
- HDFS-8286 - Partial Namespace In Memory.
- HDFS-1052 Federation aims to address namespace and Block Space scalability issues.
  - Federation deployment and planning adds complexity
  - Requires changes to other components in the Hadoop stack
- HDFS-10467 Router based federation
- HDFS-7240 - Ozone borrows many ideas learned from these efforts and is a super set of these approaches.

HORTONWORKS

Ozone and HDFS - now

- HDFS-7240 merged into Apache Hadoop trunk
  - New sub-project named HDDS (Hadoop Distributed Data Store).
  - Loadable module running in/out of process with HDFS datanode.
  - HDDS release is independent release Hadoop.
  - Maven profile (-pHdds) that disable HDDS compile by default.
  - Thanks to all the contributors from the community that made Ozone real!

HORTONWORKS

Ozone Overview

- Ozone is an S3 like object store, which exposes REST API and Hadoop RPC interfaces.
- OzoneFS is a Hadoop compatible file system.
- Applications like Hive, Spark, YARN and MapReduce run natively on OzoneFS **without any modifications**.

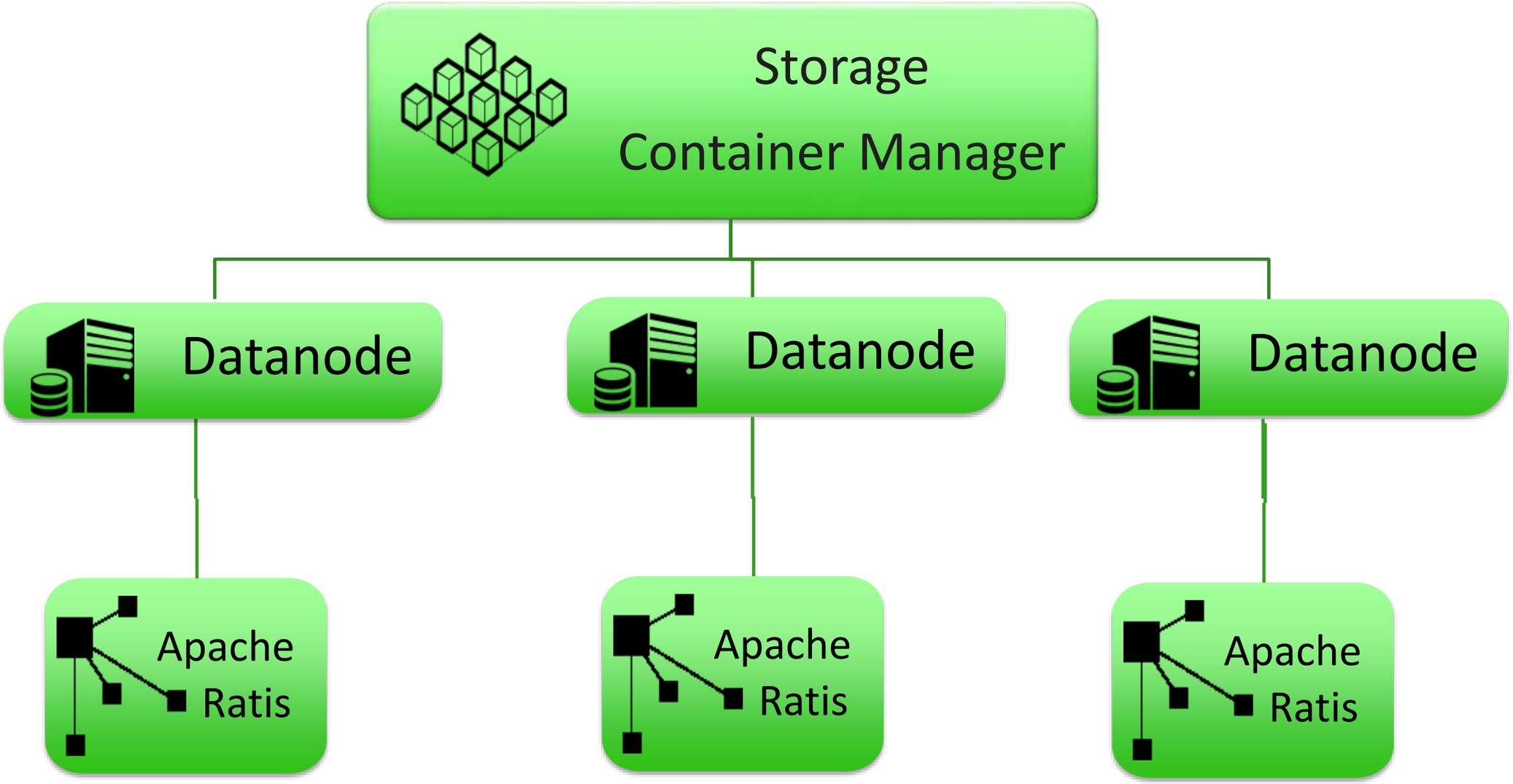
HORTONWORKS

Ozone and Block layer

- The most basic abstraction in Ozone is a block layer called –
  - **Hadoop Distributed Data Store or HDDS.**
- This talk is really about HDDS; how it works and how it will be used by other name space services.
  - For example, Ozone, HDFS and Quadra (CBlock) are simple namespace services that can be built using HDDS.
- HDDS is a Distributed, Highly available block storage framework. It is designed to be used by a system that provides it own namespace management.
- HDDS consists of a block manager called Storage Container Manager (SCM) and a set of nodes that act as the data stores. The following architectural diagram shows the components that makeup HDDS.

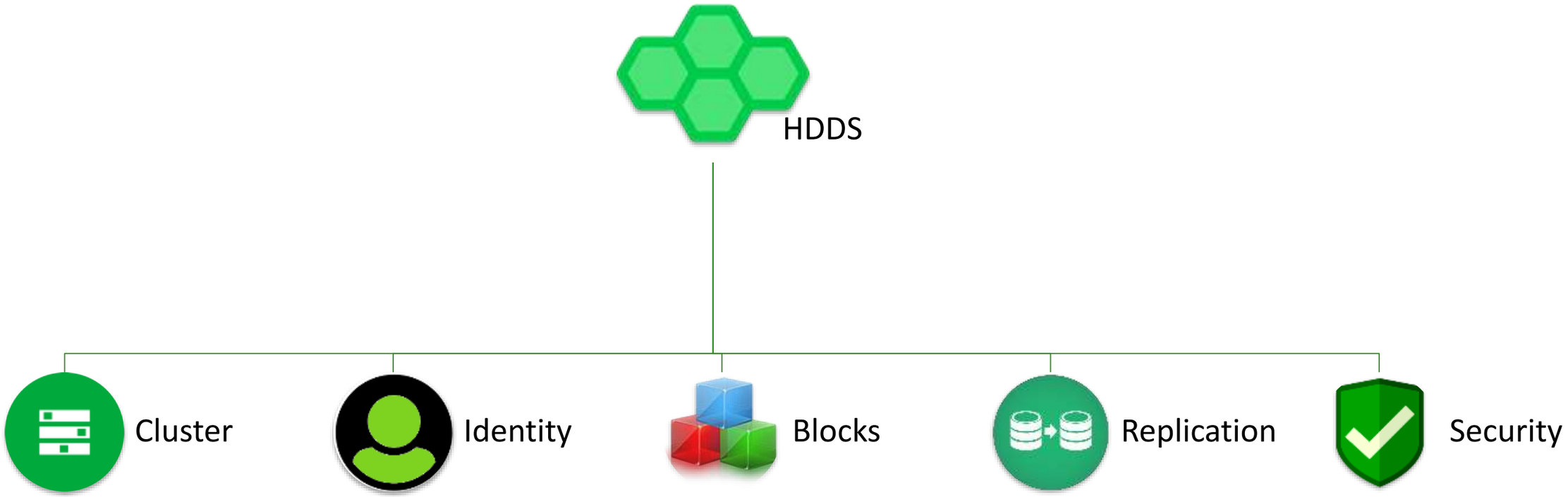
HORTONWORKS

Hadoop Distributed Data Store Architecture



HORTONWORKS

Hadoop Distributed Data Store Services



Booting a Cluster

HORTONWORKS

Booting up a Cluster

- HDDS acts as the central identity manager for the cluster.
- HDDS has a complete Certificate Infrastructure built-in
- The identity management service of HDDS provides the following identities
  - name service identities.
  - datanode identities.
- When SCM boots up it creates its own Certificate Authority
  - Yes, You can provision it to be an Intermediate authority – That is be a part of an existing CA infra-structure
- When datanodes join the cluster, SCM verifies datanode identity and issues a datanode certificate.
- Other name space services like Ozone Manager, Namenode' or Quadra Server will obtain service identity certificates from SCM.

HORTONWORKS

Booting up a Cluster – Initializing SCM

1. Generate Cluster ID & SCM ID

2. Generate Self-Signed Certificates

3. Wait for Data nodes to register

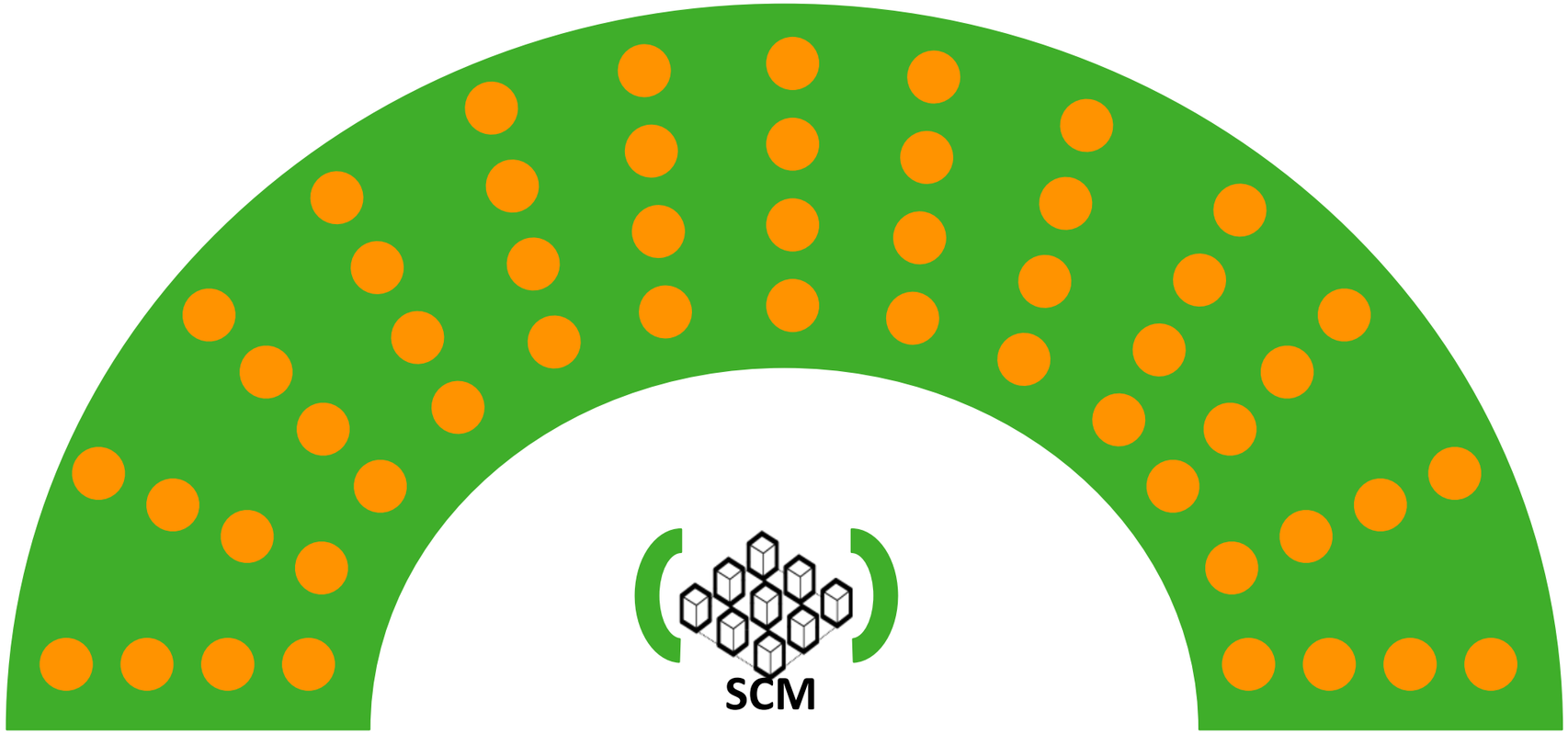
HORTONWORKS

Booting up a Cluster – Issuing certificates to Data nodes

1. Data nodes make a certificate request

2. SCM issues certificate after verifying the identity

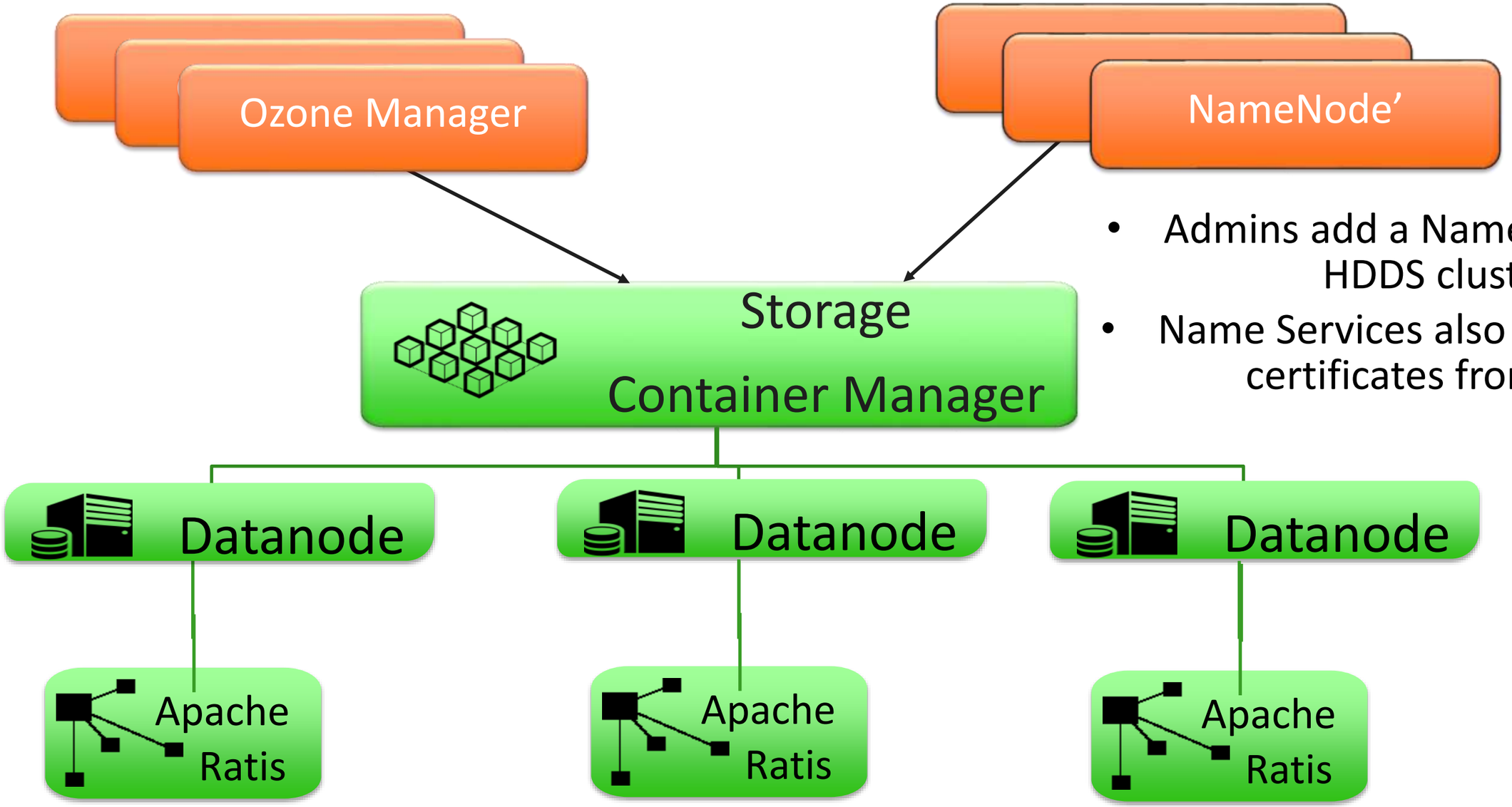
3. Data nodes are functional



- Kerberos is not required on datanodes for a secure cluster.

HORTONWORKS

Booting up a Cluster – Registering a name service



- Admins add a Name Service to HDDS cluster
- Name Services also get identity certificates from SCM

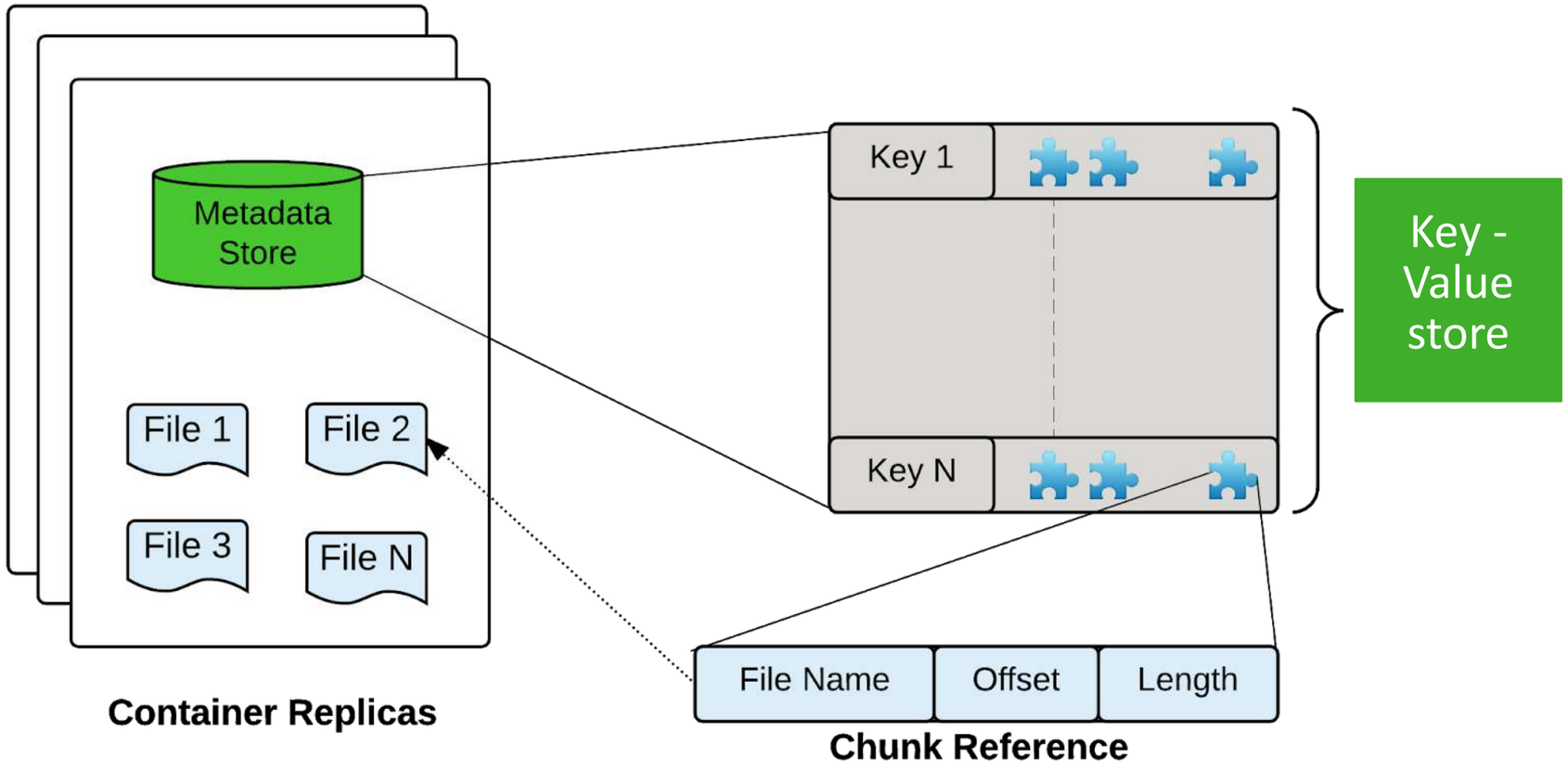
HORTONWORKS

Writing Data

HORTONWORKS

Writing Data – Understanding Storage Containers

- HDDS datanode is a plug-in service running in Datanodes.
- Container is basic unit of replication (2-16GB).
- Fully distributed block metadata in LSM-based K-V store.
- No centralized block map in memory like HDFS.



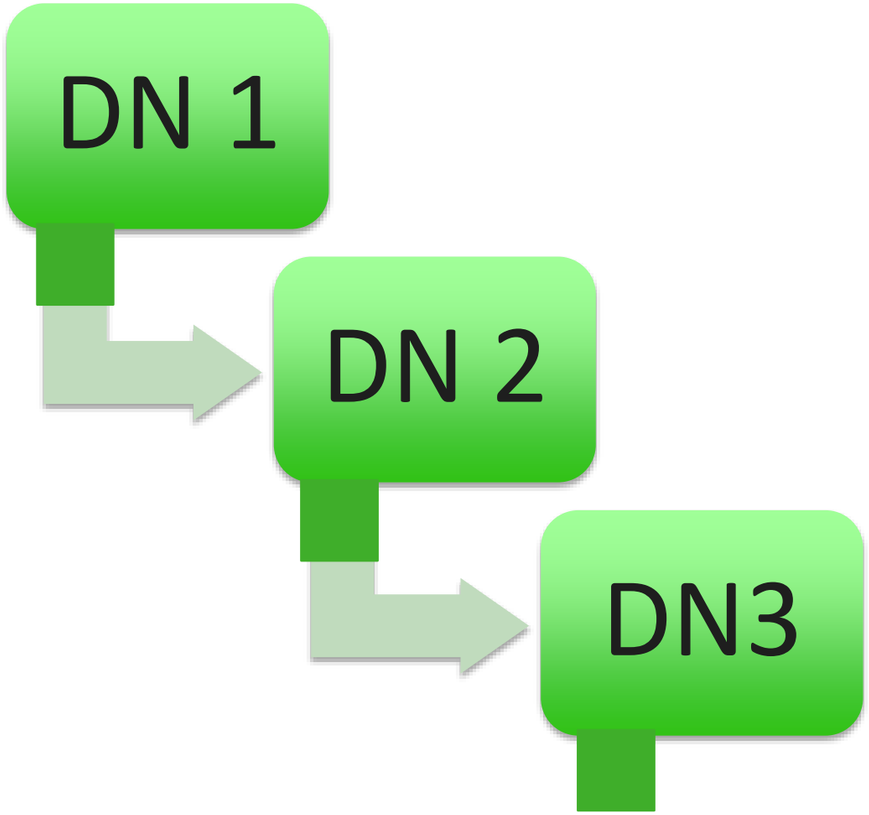
Writing Data – Understanding Open/Close Containers

- Close Container: keys and their blocks are immutable
  - Write is not allowed.
  - Container closed when is it is close to full or failure.

- Open Container: keys and their blocks are mutable
  - Write via Apace Ratis

Writing Data – Understanding Pipelines

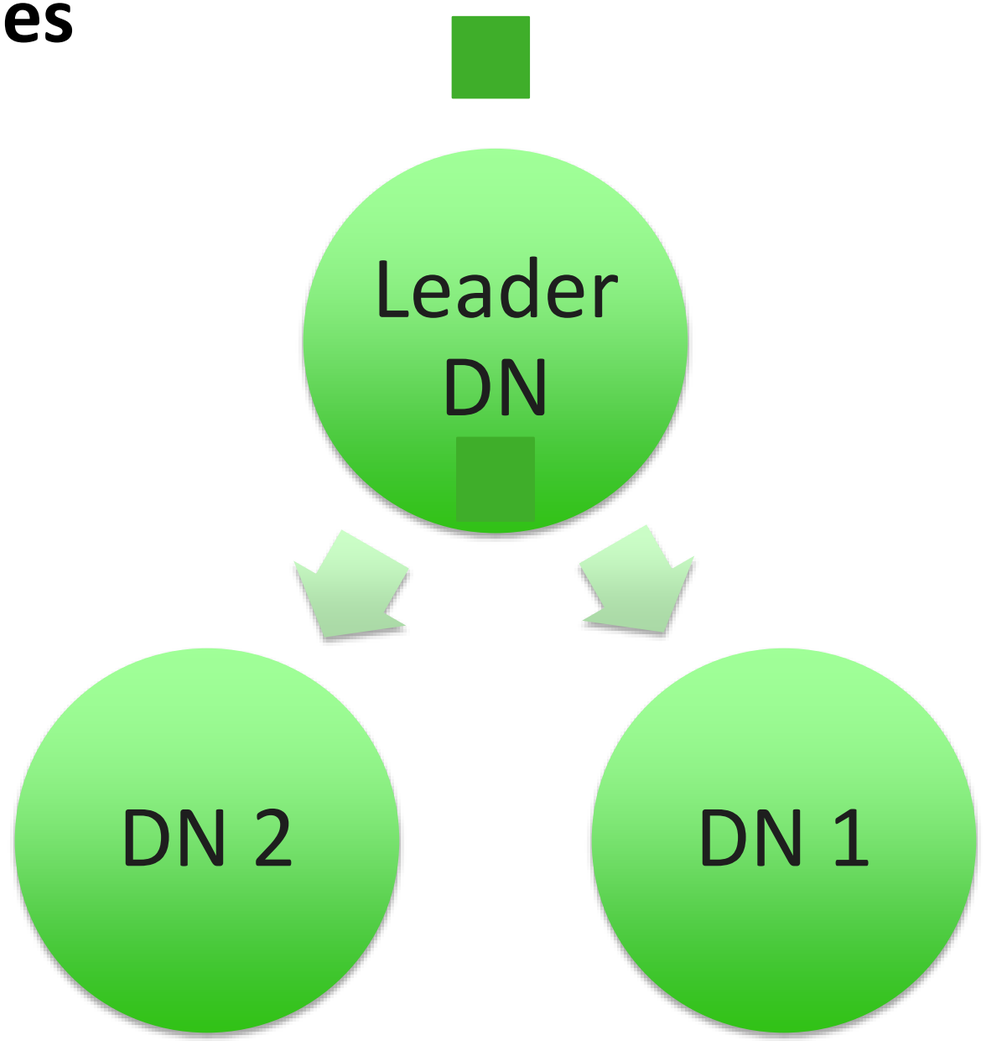
- **A pipeline is a replicated stream.**
- HDDS is designed to have different kind of streams – that allows support for different strategies.
- For example, if we are using HDFS replication, a Pipeline would like this.



HORTONWORKS

Writing Data – Understanding Pipelines

- **HDDS writes to Containers using Pipelines**.
- HDDS uses Apache Ratis based pipeline, which still appears as a stream.
  - Apache Ratis is a highly customizable Raft consensus protocol library.
  - Support high throughput data replication use cases like Ozone.
- For example, if we are using HDDS and Apache Ratis based replication, a Pipeline would like this.



Writing Data -- Blocks

Client

1. Write Key

1. Client makes a request to Ozone Manager to write a key.

Ozone Manager

2. Get Block

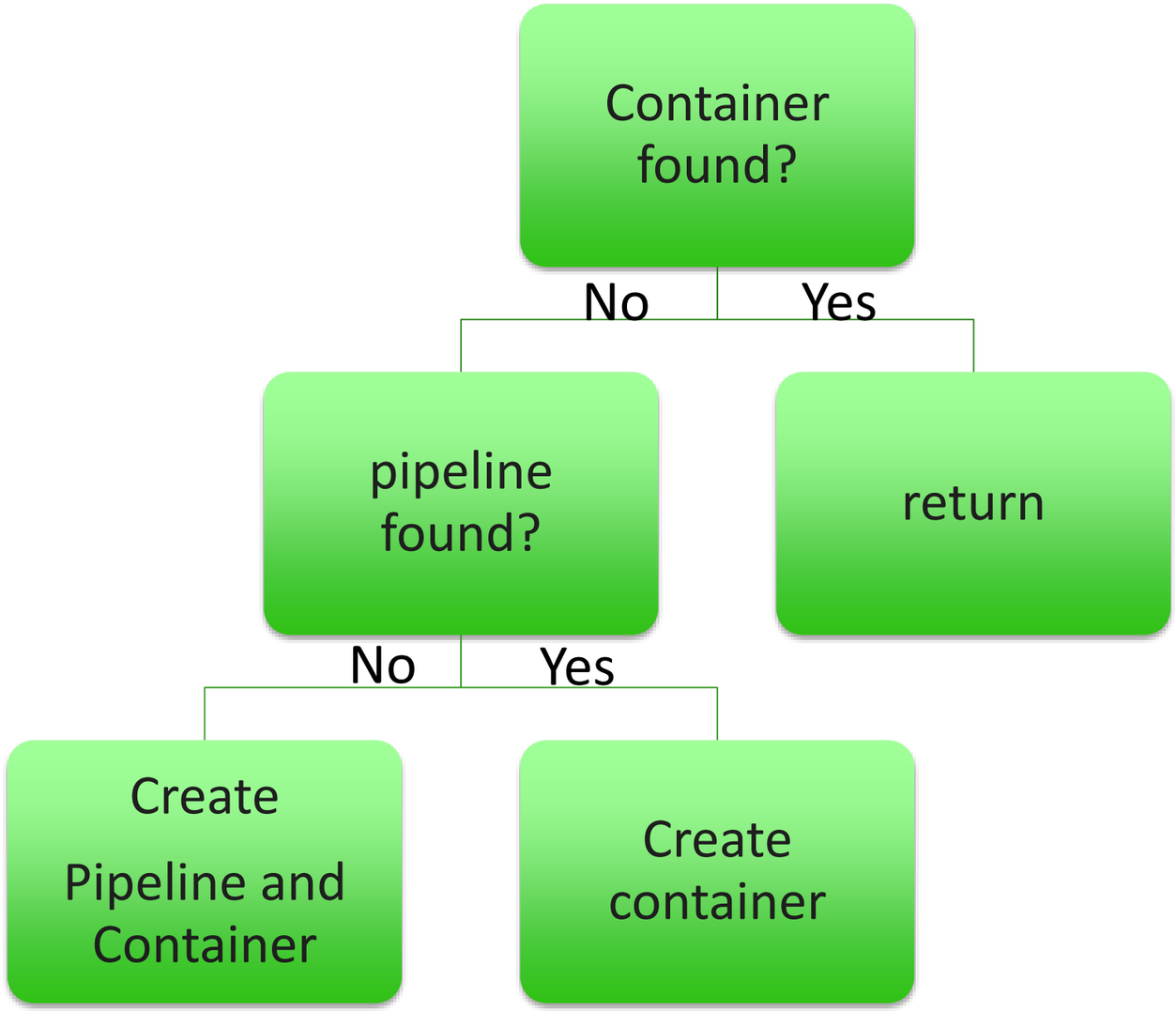
2. Ozone Manager requests SCM to allocate a block.

SCM

3. SCM allocates a block.

HORTONWORKS

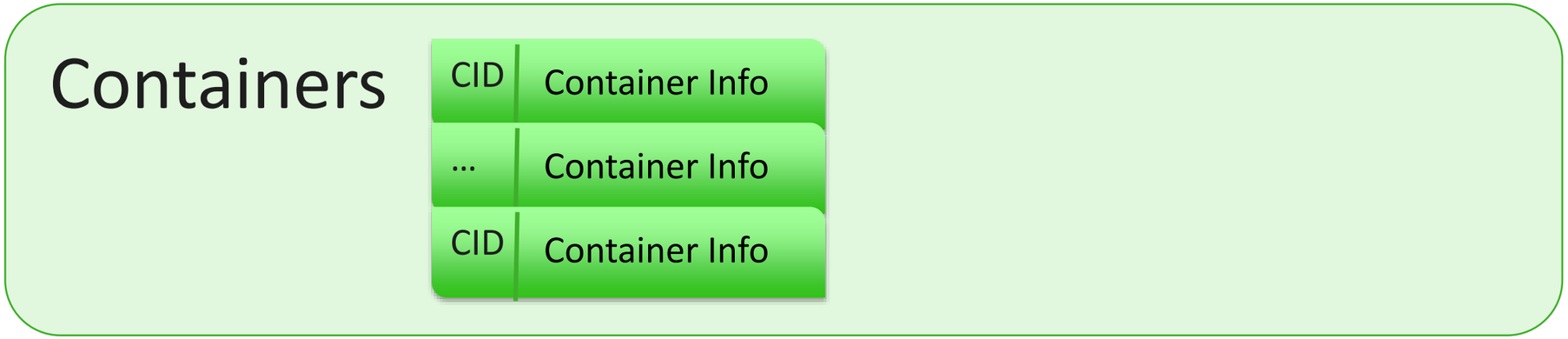
Writing Data – SCM Allocating Container/Pipeline for Blocks



HORTONWORKS

Writing Data - SCM records the Container state

Storage Container Manager persists the container states to LSM-based K-V Store

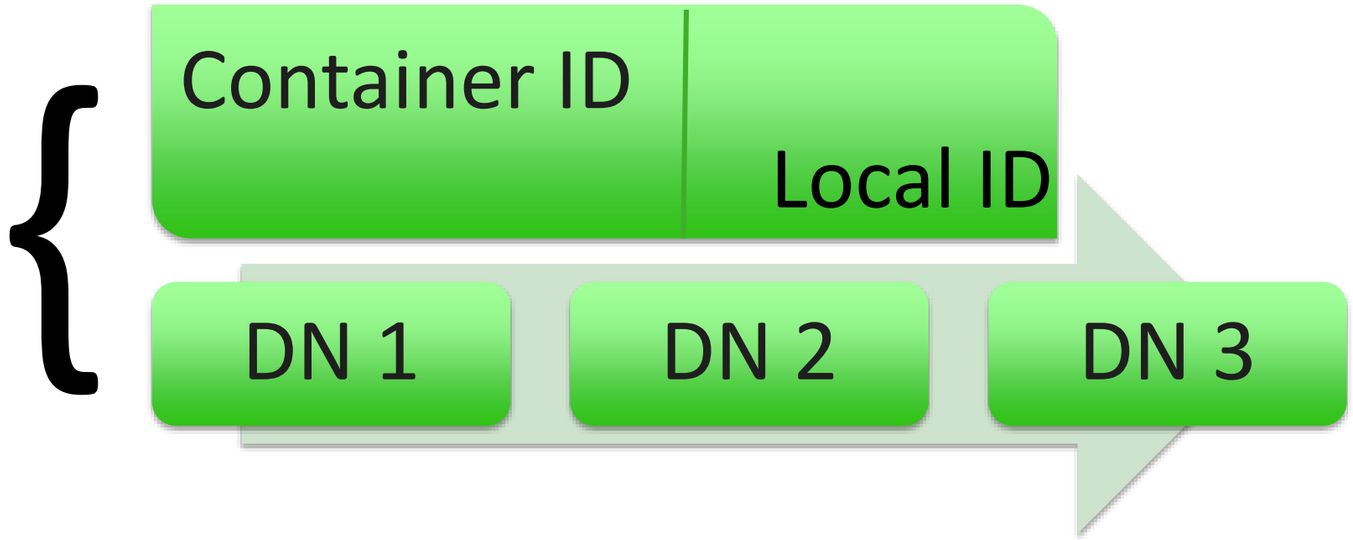


HORTONWORKS

Writing Data

SCM returns a Block ID and a Pipeline to Ozone Manager.

Client needs Block ID and Pipeline to write data



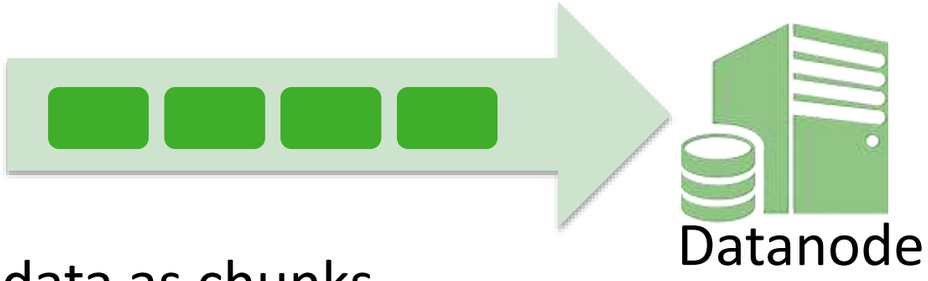
This is similar to HDFS block and data node locations.

HORTONWORKS

Block Commit Protocol

1. Client → PutKey → Ozone Manager → Allocate Block → SCM

2. Client → Datanode



- Writes data as chunks
- Update metadata of a block

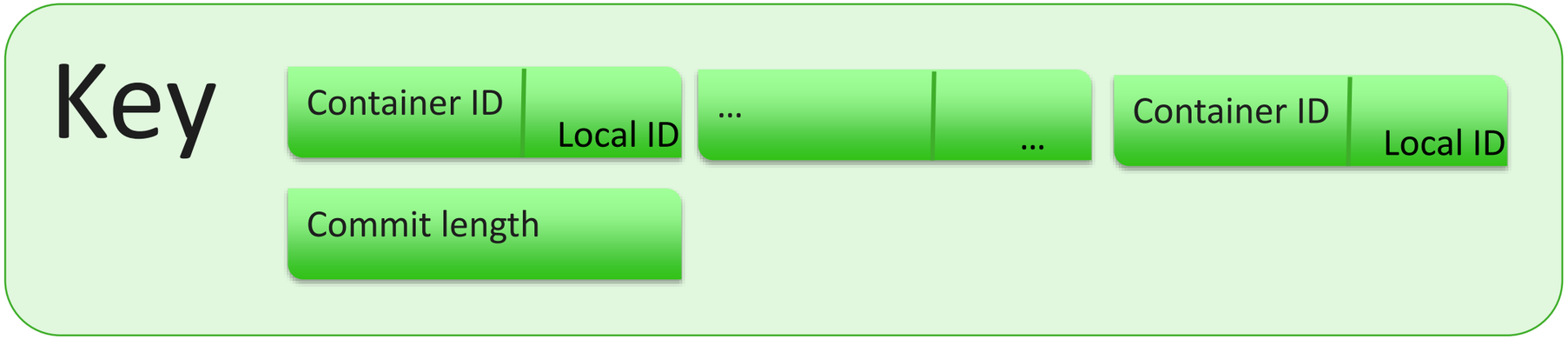
Block	Value
Block 001	{ List of Chunks }
Block 002	{ List of Chunks }

3. Client → Commit Key → Ozone Manager

HORTONWORKS

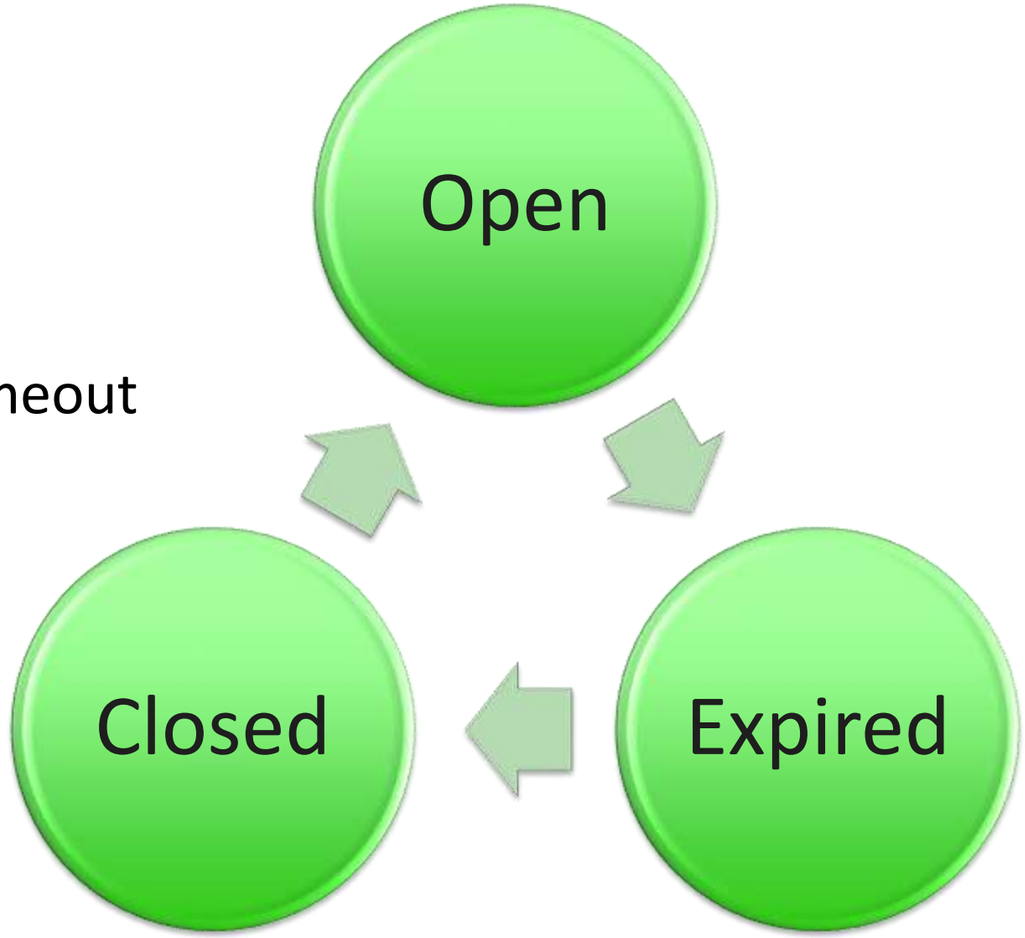
Writing Data – Ozone Manager records the Key info

Ozone Manager saves the Key Info to LSM-based K-V Metadata Store



Writing Data – Ozone Key State

- **Open Key:** Uncommitted key
  - commit length subject to change
- **Closed Key:** Commit length won't change
- **Expired Open Key:**
  - Open key is garbage collected after timeout



Reading Data

Reading Data

Client

1. Read Key

1. Client makes a request to Ozone Manager to write a key.

Ozone Manager

2. GetLatestKeyLocations

2. Ozone Manager returns key location info, i.e. list of blocks.

DN

3. Client connects to DN and reads blocks.

HORTONWORKS

Deleting Data

Deleting Data – Open/Close Containers

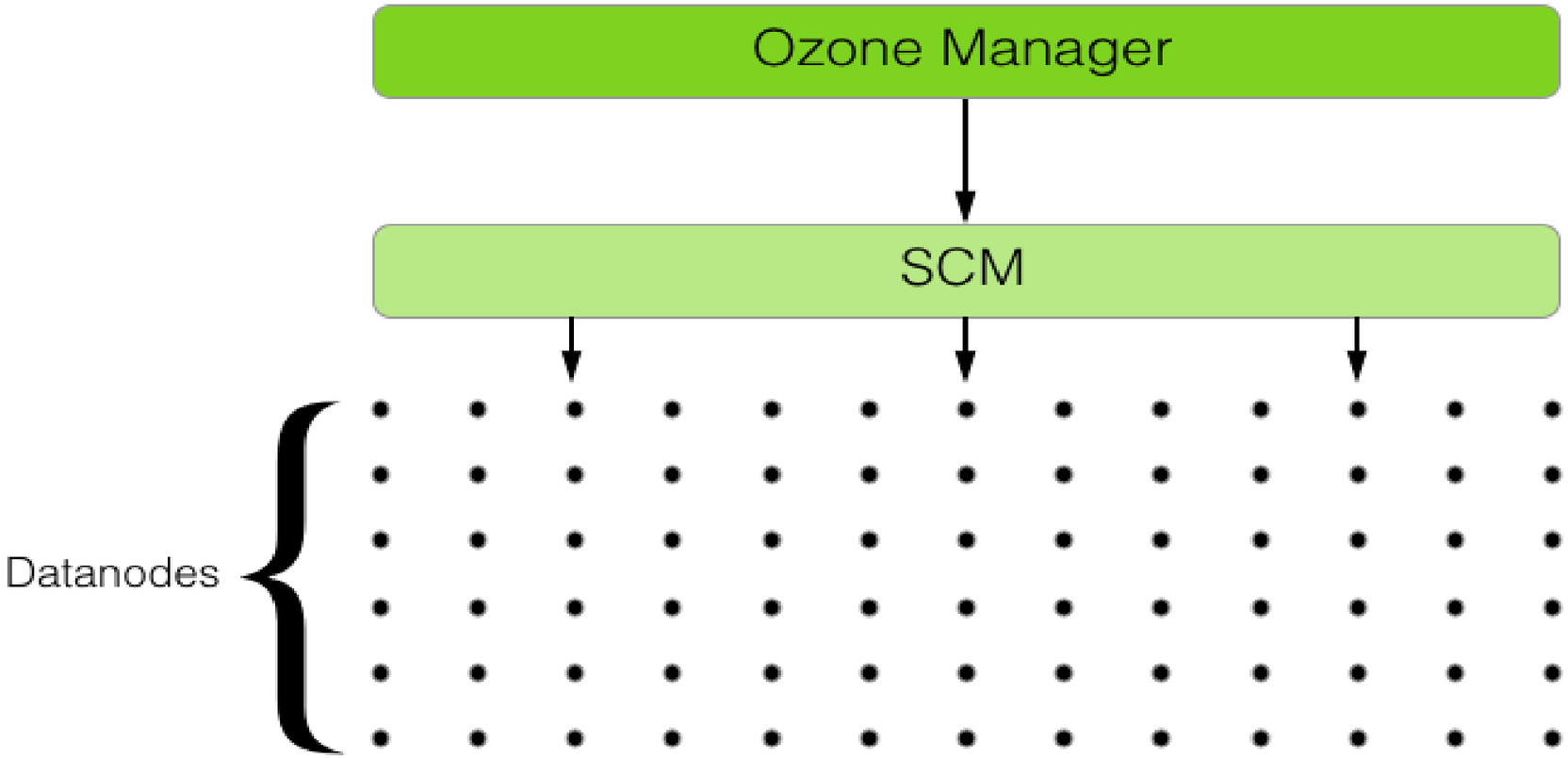
- Close Container: keys and their blocks are immutable
  - Delete key/blocks via SCM Delete log.
    - Each delete log has a unique ID
    - SCM sort delete logs by Datanodes and Containers
    - SCM send deleted logs to Datanode via heartbeat response
    - SCM clear delete log entry only if delete entry processing is marked done on all the datanodes.
- Open Container: keys and their blocks are mutable
  - Delete key/blocks via SCM Delete log.
    - SCM holds off sending delete logs to datanode on open containers until the container is closed.

HORTONWORKS

Scaling Ozone

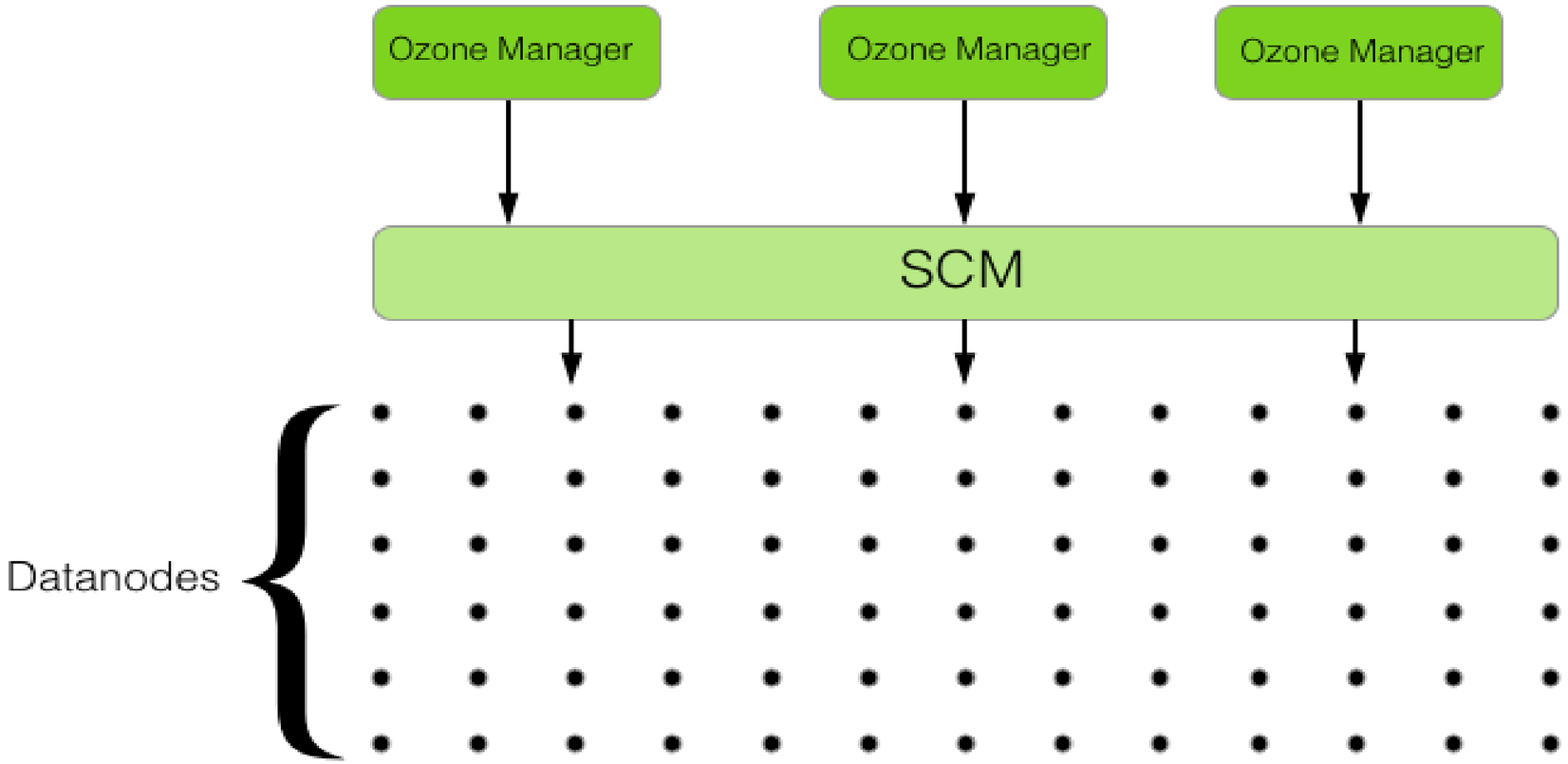
HORTONWORKS

Single Ozone Manager + Single SCM



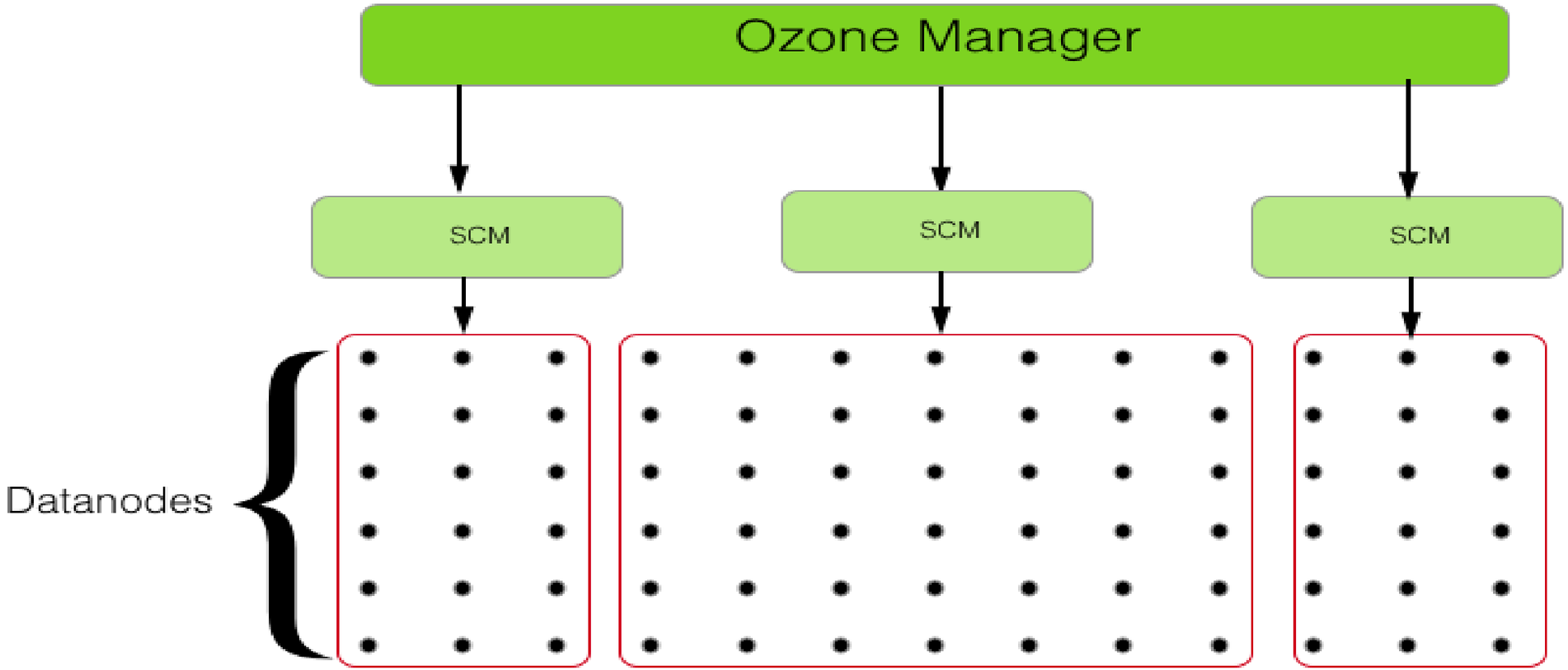
HORTONWORKS

Multiple Ozone Managers + Single SCM



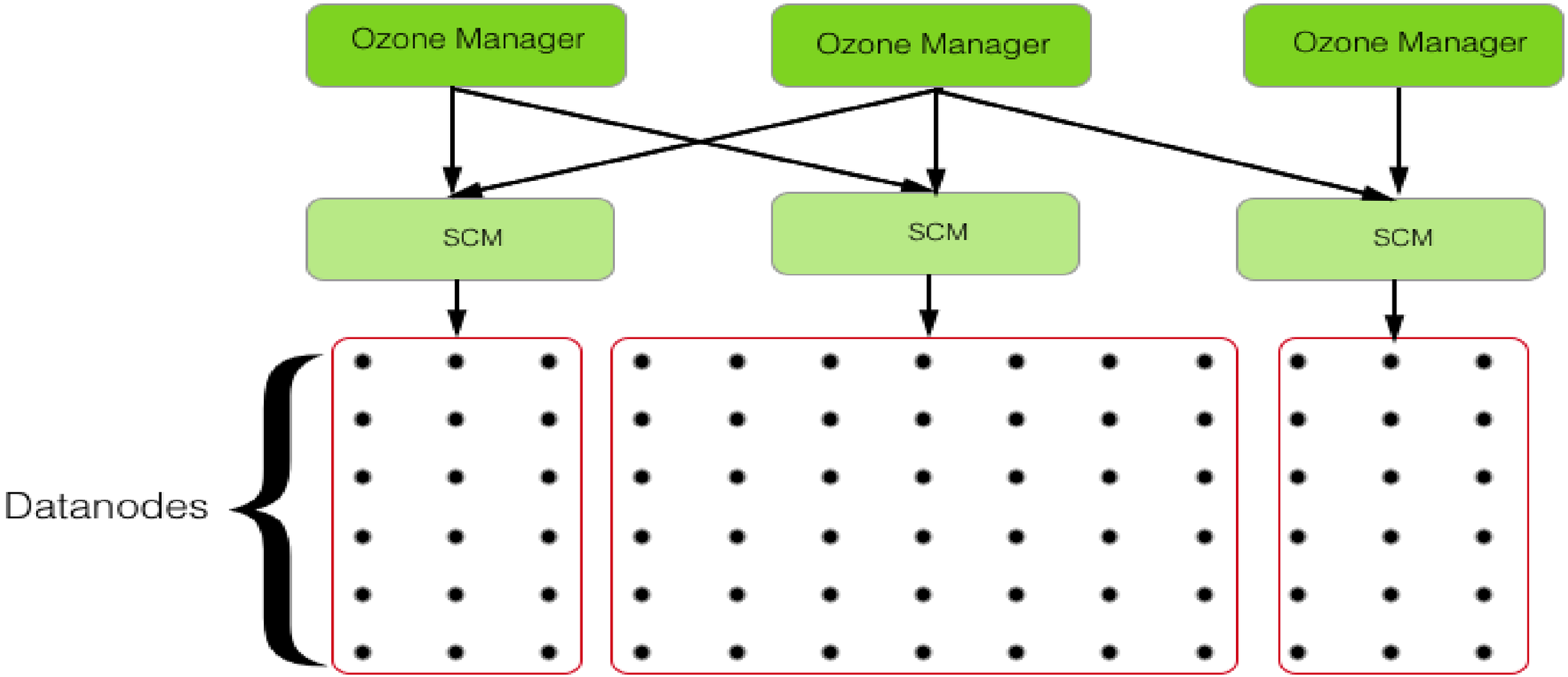
HORTONWORKS

Single Ozone Manager + Multiple SCMs



HORTONWORKS

Multiple Ozone Managers + Multiple SCMs



HORTONWORKS

Ozone Security

Ozone Security

- Authentication
  - Identity
    - Kerberos (OM/SCM/Client)
    - Certificate (SCM/OM/Datanodes)
  - Token
    - Delegation Token
    - Block Token

HORTONWORKS

Certificate-based Delegation Token

- OM token <-> HDFS namenode token
  - For MR jobs to access OM

	Ozone	HDFS
Issued by	OM	Namenode
Validated by	OM Certificate(Public Key)	Namenode
Signed by	HMAC-SHA256	HMAC-SHA1
	OM Private Key	Shared Key
Protocol	Hadoop RPC/REST	Hadoop RPC

Certificate-based Block Token

- OM block token <-> HDFS block token
  - To authenticate block access on datanodes from Ozone clients.

	Ozone	HDFS
Issued by	OM	Namenode
Validated by	HDDS Datanode **OM(Public Key)**	Datanode (Shared Key)
Signed by	HMAC-SHA256 OM(Private Key)	HMAC-SHA1 (Shared Key)
Protocol	gRPC	Hadoop RPC
Secure Block Token Transfer	DN Public Key	Hadoop RPC encryption

HORTONWORKS

Ozone HDDS in progress

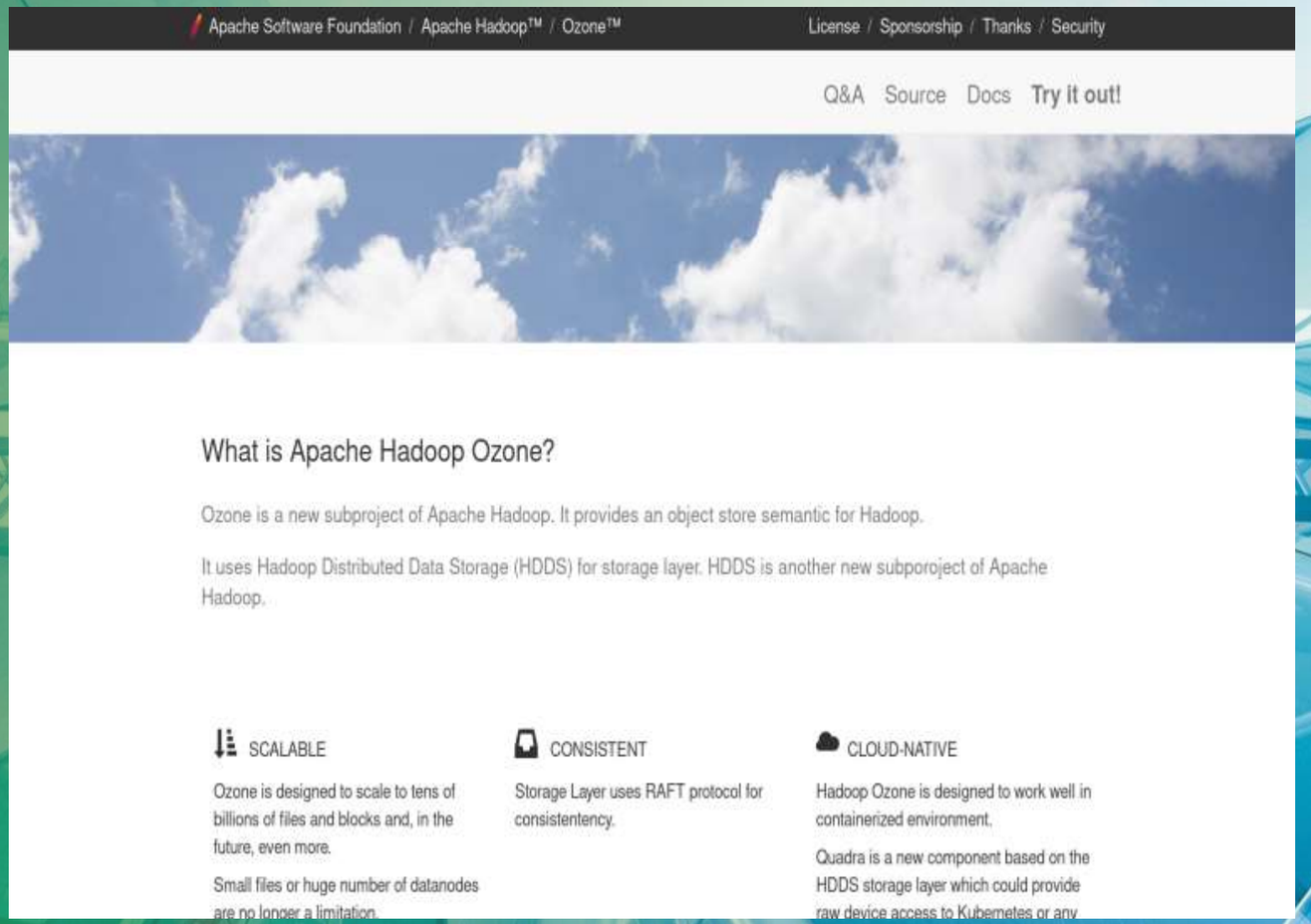
- High Availability
  - Apache Ratis-based OM/SCM metadata replication
  - HDDS-151 branch
- Pluggable Container Storage I/O
  - HDFS-48 branch
- Security
  - Kerberos/Delegation Token/Audit
  - HDDS-4 branch

HORTONWORKS

Questions?

- Visit Ozone website:
  - http://ozone.hadoop.apache.org

- Visit Ozone Booth@Ask Me Anything
  - Ozone Demo
  - Q & A



HORTONWORKS

Thank you