Aerospike for Developers

Architecture, Data Model and Types

AEROSPIKE

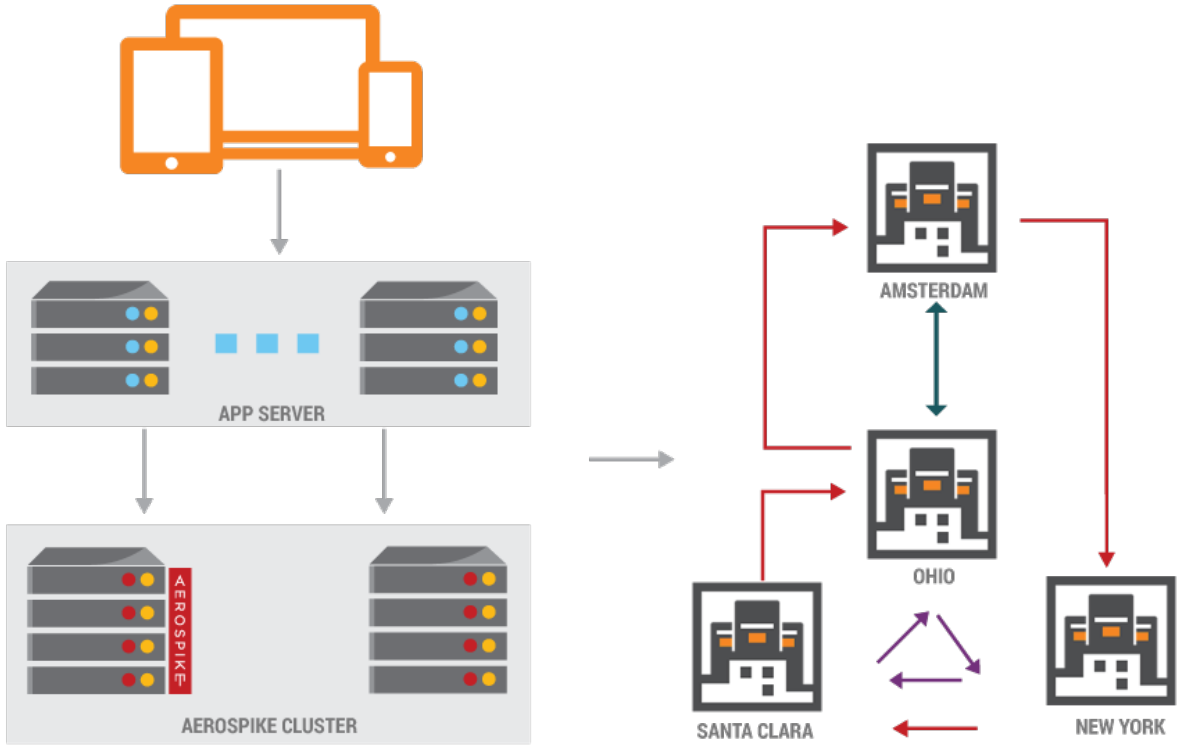
Goals

This module describes the Aerospike Data Model and Data Types. At the end of this module you will understand:

- The Aerospike Architecture
- The data model and types
  - Namespaces
  - Sets
  - Records
  - Bins and Bin types
  - Language mapping
  - Large Data Types

Architecture Overview

Architecture – The Big Picture



1) **No Hotspots** – DHT simplifies data partitioning

2) Smart Client – **1 hop** to data, no load balancers

3) **Shared Nothing Architecture**, every node is identical

4) Single row **ACID** – synch replication in cluster

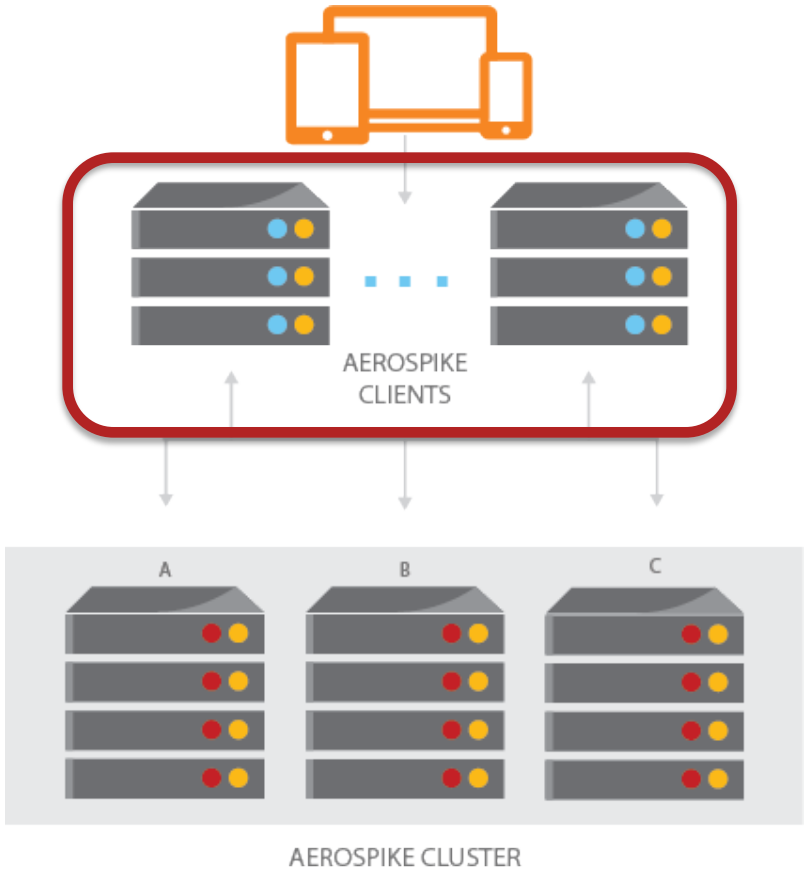
5) Smart Cluster, **Zero Touch** – auto-failover, rebalancing, rack aware, rolling upgrades

6) Transactions and long-running tasks prioritized in real-time

7) **XDR** – sync replication across data centers ensures **Zero Downtime**

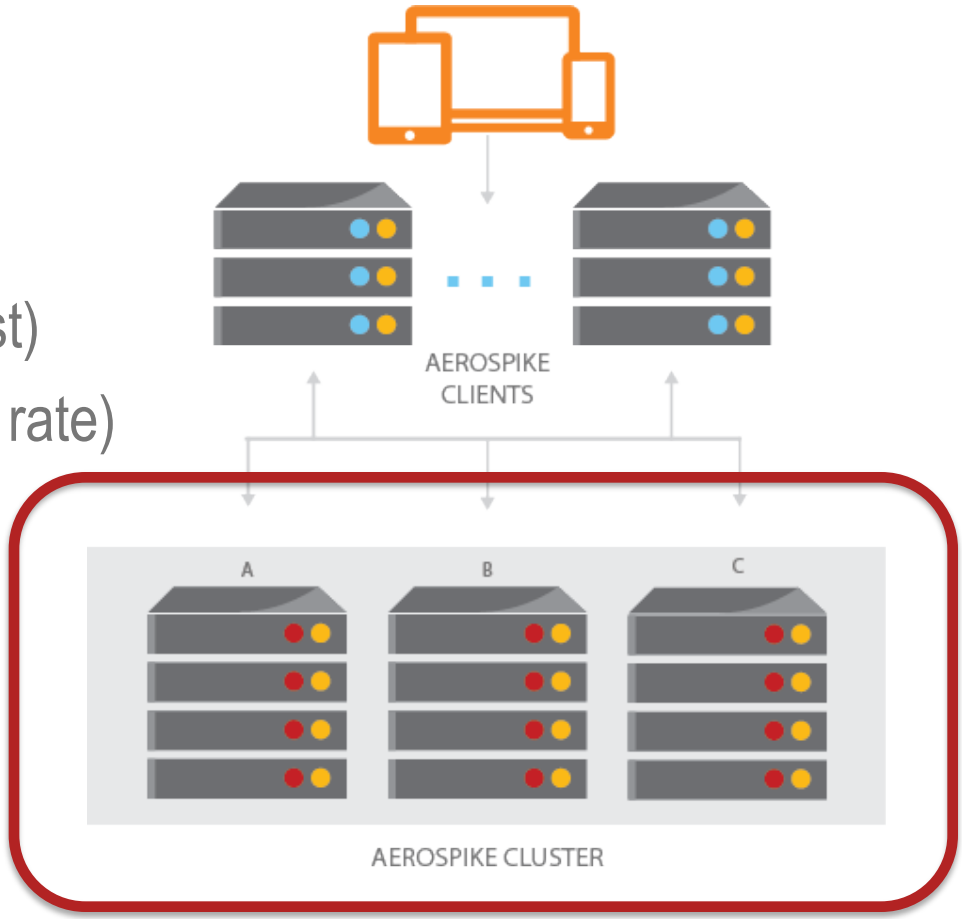
Smart Client™

- The Aerospike Client is implemented as a **library**, JAR or DLL, and consists of 2 parts:
  - Operation APIs – These are the operations that you can execute on the cluster – **CRUD+** etc.
  - First class **observer** of the Cluster – Monitoring the state of each node and aware on new nodes or node failures.



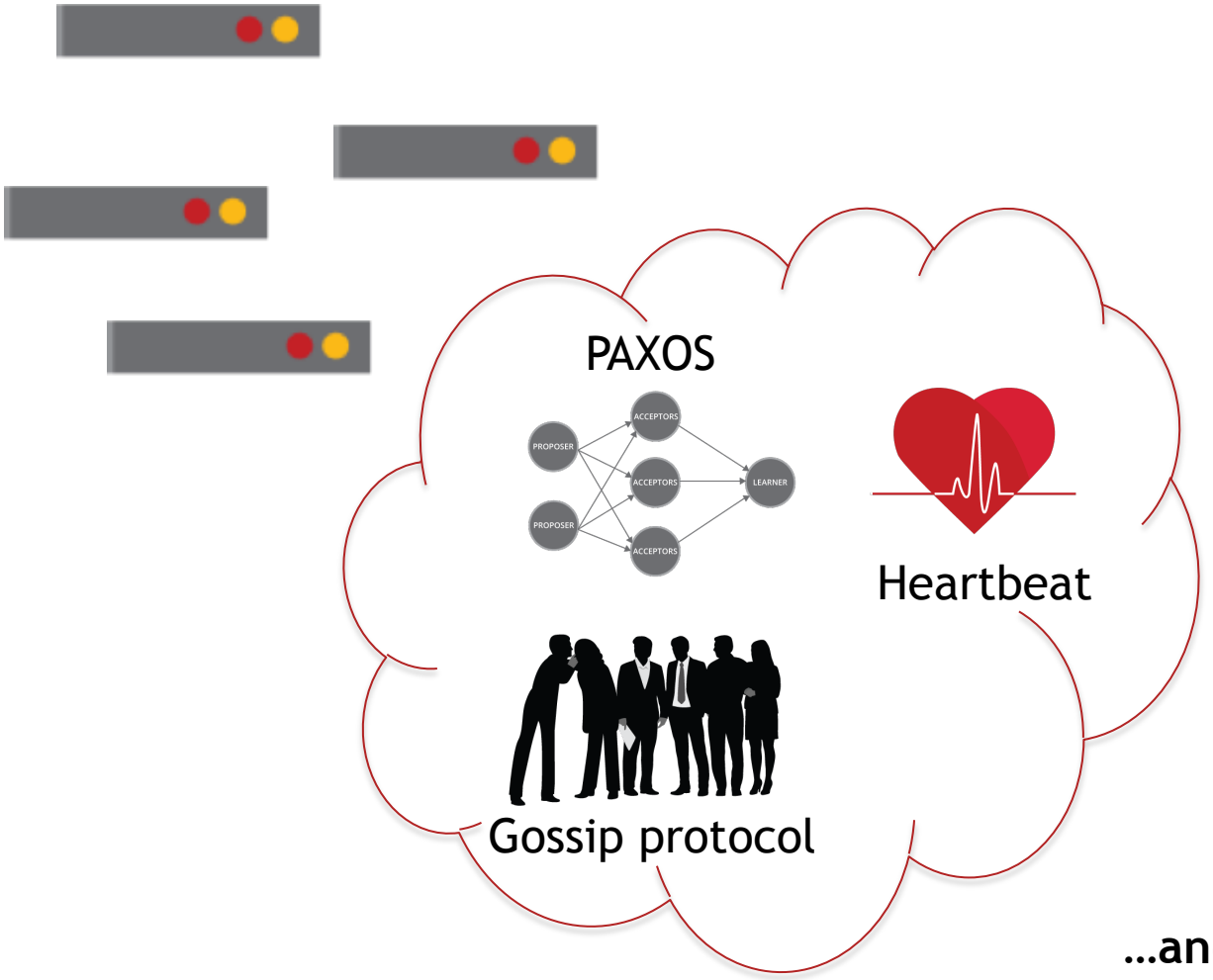
The Cluster (servers)

- Federation of **local servers**
  - XDR to remote cluster
- Automatic **load balancing**
- Automatic **fail over**
- Detects **new nodes** (multicast)
- **Rebalances** data (measured rate)
- Adds nodes **under load**
- **Rack awareness**
- **Locally attached** storage



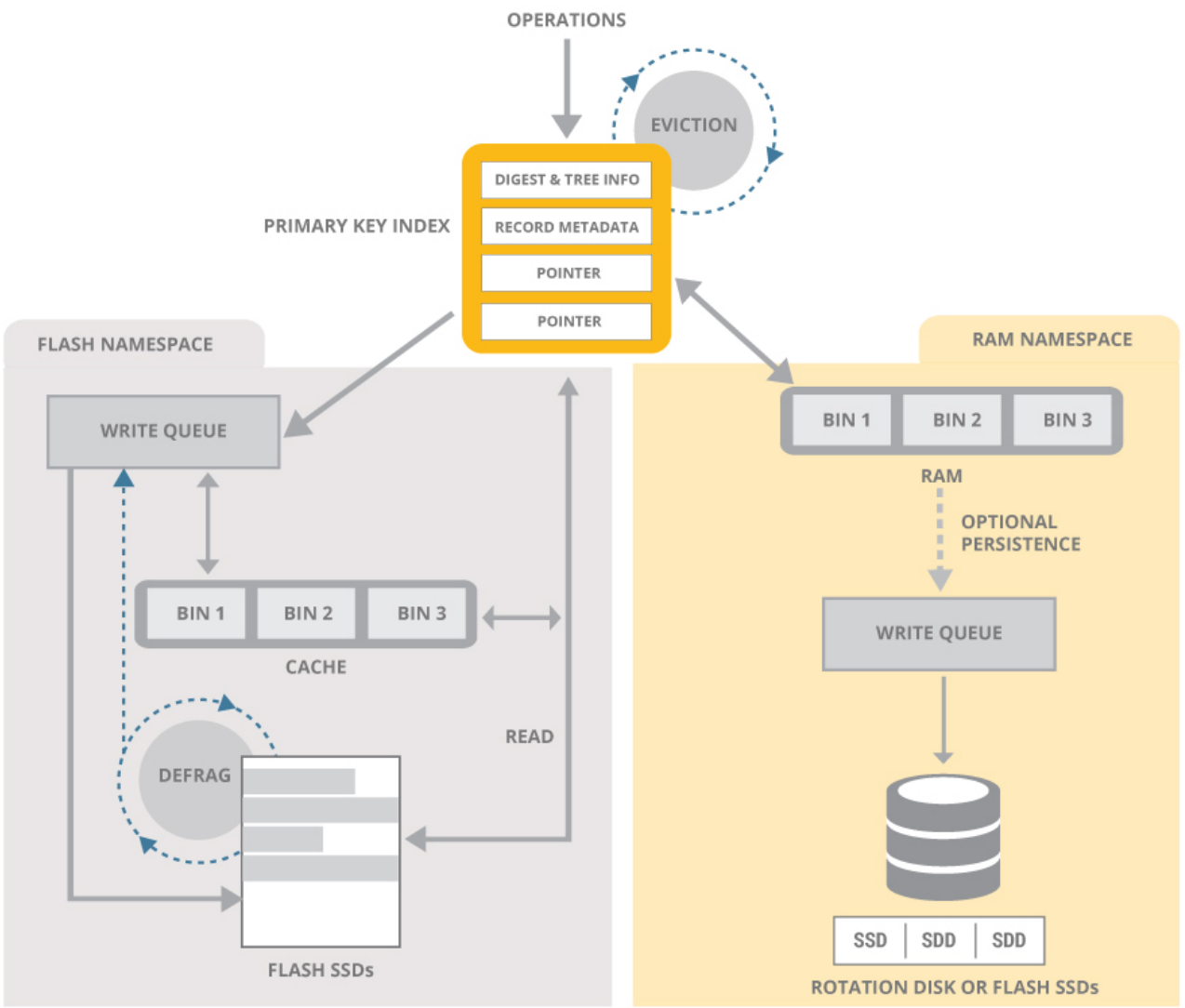
Cluster formation

Individual nodes go in…



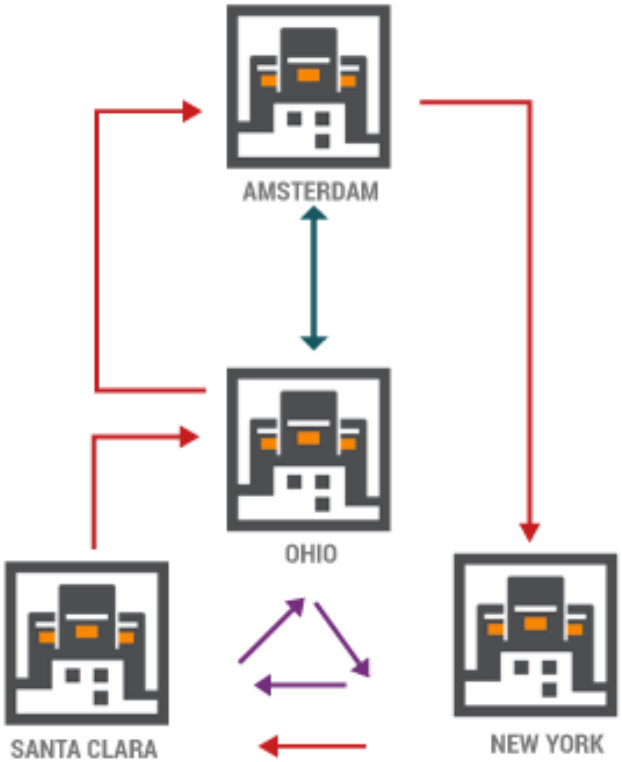
…and the Cluster forms

Data Storage Layer

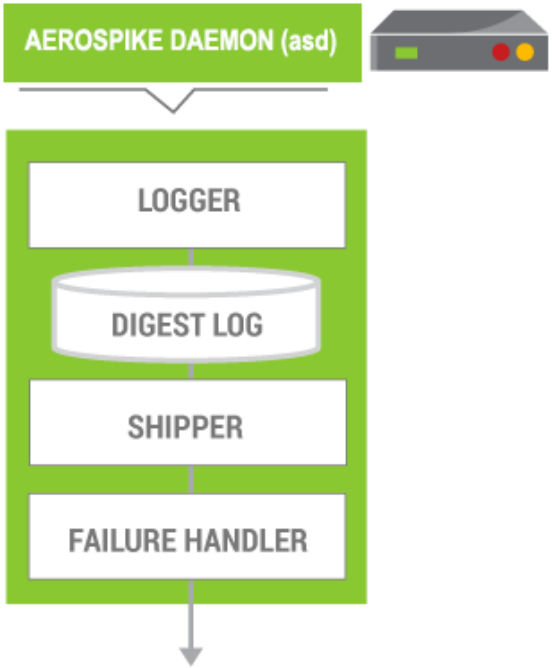


XDR Architecture

Distributed clusters

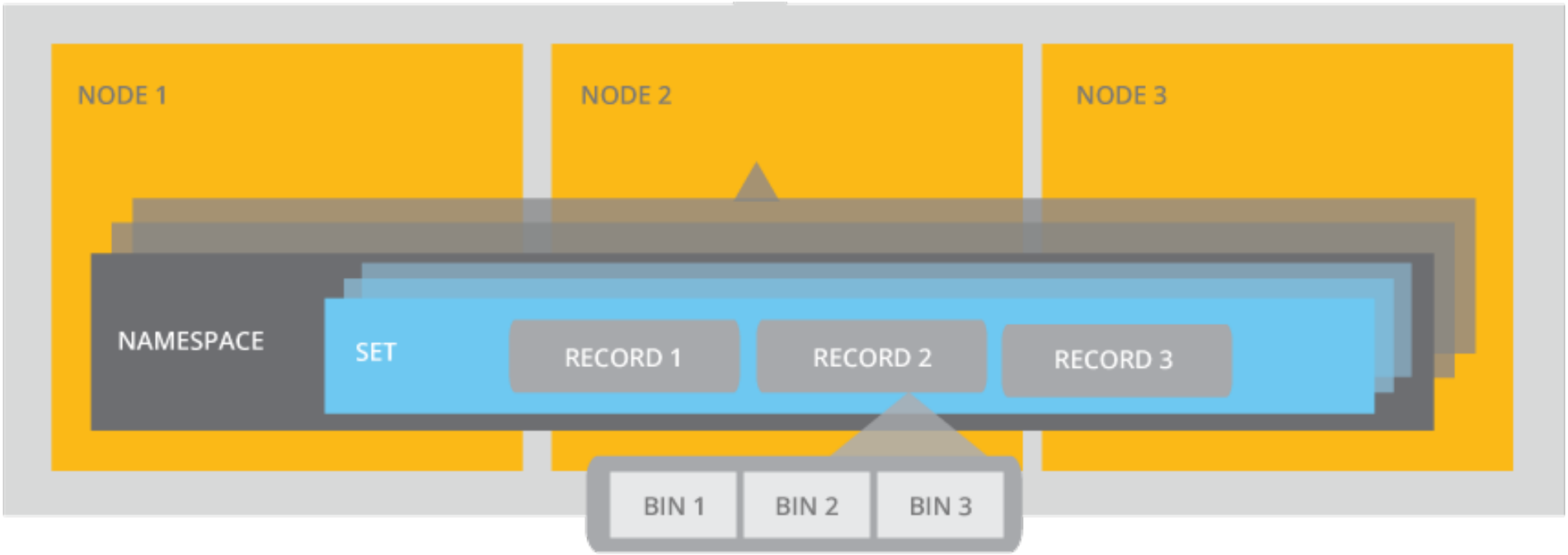


Each node in the cluster



Data Model

How Data is Organized



Aerospike	RDBMS
Namespace	Tablespace or Database
Set	Table
Record	Row
Bin	Column

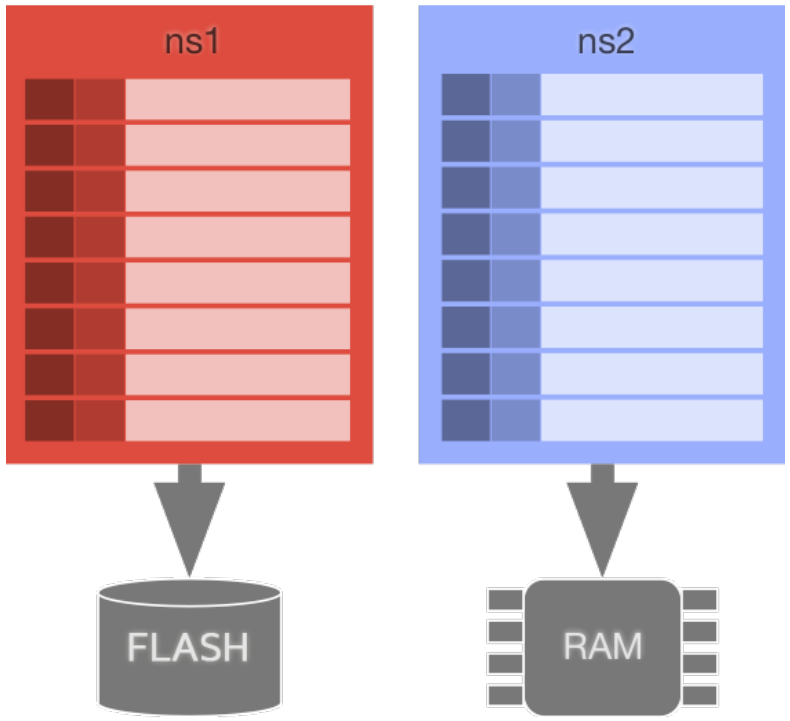
Bin type*
Integer
String
BLOB
List
Map

*details to follow

Namespace

Similar to a table space or database

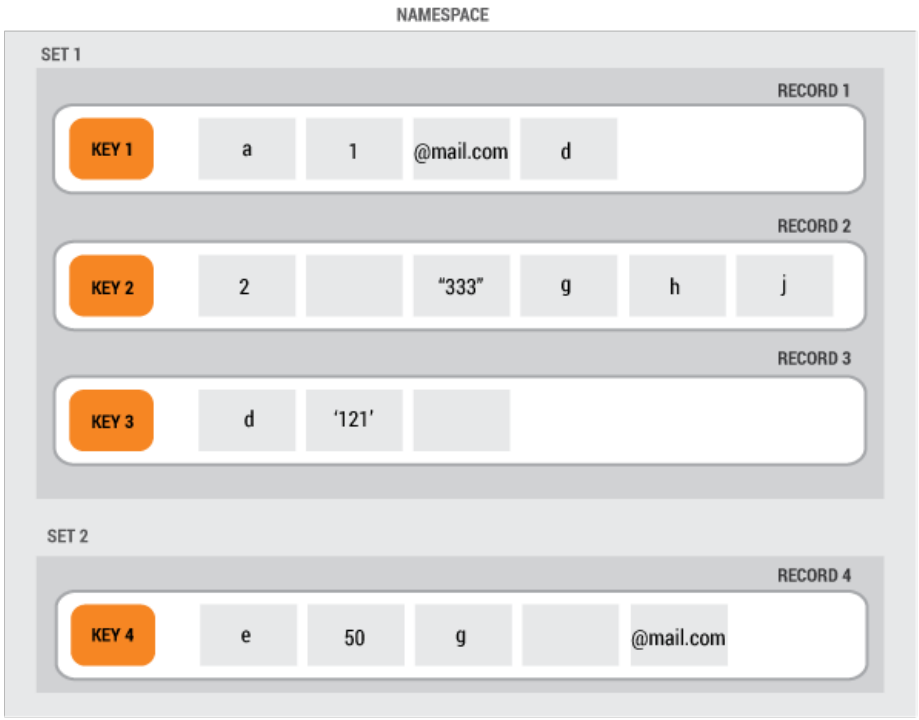
- Storage definition - DRAM or Flash
  - Storage block size (128k – 2MB)
  - Controls size of RAM and Storage
- Policy container
  - Replication factor
  - Default expiry
- Data container:
  - Namespace contain Sets
  - Sets contain Records
- Difficult to add or remove



Set

Sets are **similar** to tables

- But it has **no schema**
- Arbitrary grouping of records
- Inherits policy from namespace
- Prefix to **primary key**
- Set name <= 63 characters
- 1023 per namespace
- Cannot be deleted or renamed



Record

A record is a "**row**" of Key-value

- Value: one or more bins
- Bin has a name and type
  - Bin types: String, integer, blob, list, map, Large Data Types (LDTs)
- Bins can be added at any time
- Generation counter
  - Optimistic Concurrency
- Time-to-live
  - Auto expiration

EXP - Expiration
GEN - Generation

Bins

- Bins have a:
  - Name – 14 characters or less
  - Type – one of the following
- Bins are stored in the record
- A Bin can have a different type in another record

Id	lname	fname	address	favorites
1	Able	John	123 First	cats, dogs, mice
2	Baker	Kris	234 Second	
3	Charlie			
4	Delta	Moe	456 Fourth	stake, ice cream, apples

- Types
  - String
  - Integer
  - Blob
  - List
  - Map
- Large Data Types
  - Large Ordered List
  - Large Map
  - Large Set
  - Large Stack

Data (Bin) types

Simple Types

Simple types:

- String
  - Arbitrary length
    - limited only by record size (default 128k)
  - UTF-8
- Integer
  - 8 bytes (64 bits)
  - Unsigned
- Bytes (BLOB)
  - Array of bytes
    - limited only by record size

Caution: Language serialized

```
// C# string
string username = "iamontheinet";

// Java string
String cat = "cat";

// Go string
var username string

// C# integer
long ts = 0;

// Java integer
long mouse = 100L;

// Go integer
var feature int

 // C# integer
 byte[] sevenItems = . . .

 // Java byte array
 byte[] things = . . .
```

List

List is a **collection** of **values** ordered in insert order. A list may contain values of any of the supported data-types.

- C# List – ArrayList,
- Java List – ArrayList, , LinkedList, CopyOnWriteList, etc
- Go Slice
- Python List
- Can contain:
  - Integers, strings, BLOBS, Lists and maps
- JSON array

```
// C# List
List<object> things = . . .

// Java List
List<String> interests = . . .
```

Map

Map is a **collection** of **key-value pairs**. Each key may only appear once in the collection and is associated with a value. The key and value of a map may be of any of the supported data-types.

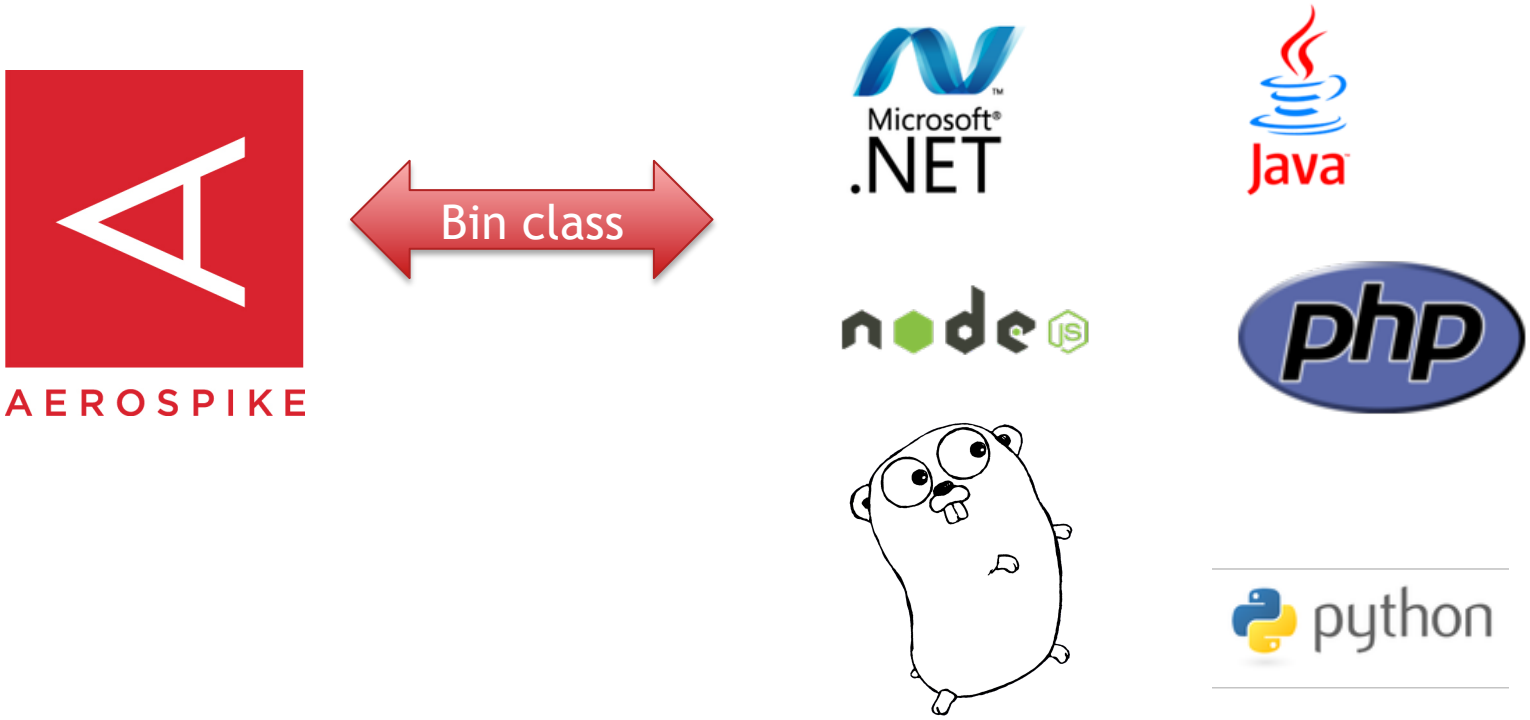
- C# Dictionary
- Java **Map** – HashTable, HashMap, etc.
- Python **Dictionary**
- Can contain:
  - Integers, strings, BLOBS, Lists and maps
- JSON object

```
// C# map
Dictionary<string, string> aMap = . . .

// Java map
Map<String, String> aMap = . . .
```

Type Mapping

Types are mapped to the equivalent language type. Methods on the Bin class convert the Aerospike type to the native language type.

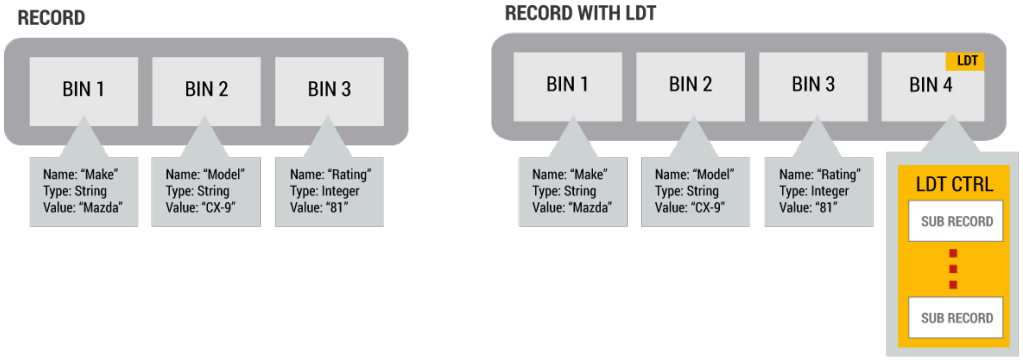


Large Data Types

Large Data Types

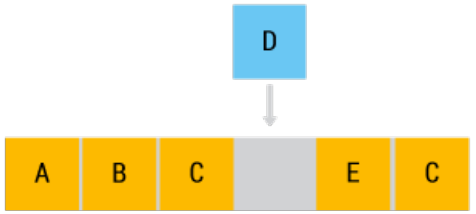
Large Data Types (LDTs) allow individual records to contain a **very large** amount of data, where the limit is based (mostly) on available storage and not on the maximum size of a record.

- Top record
  - Control bin
  - Configuration
- Elements as a **sub record**
  - Access time for an element is same
- Operations on the collection
  - add() an element to the collection
  - get() an element from the collection
  - remove() an element from the collection
  - filter() is used to scan a collection
- 4 Large Types
  - Large Ordered List
  - Large Map
  - Large Set
  - Large Stack

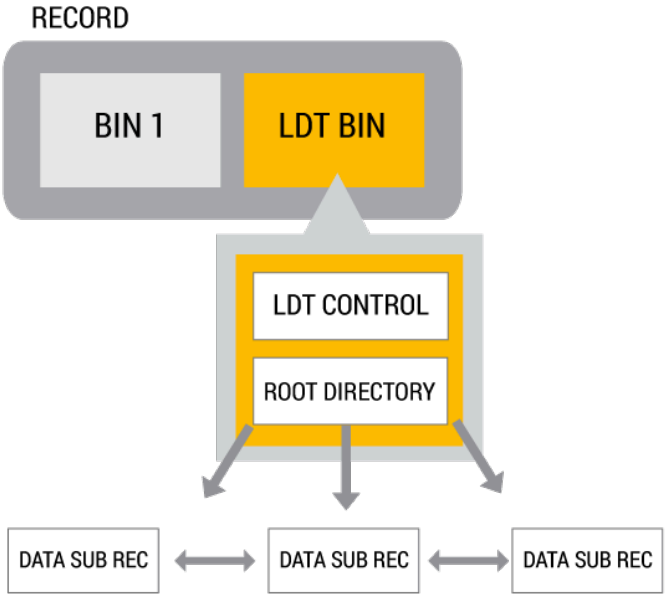


Large Ordered List

Large **Ordered List** is optimized for searching and updating sorted lists.



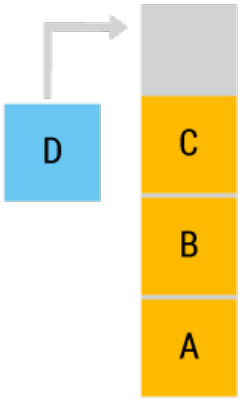
The Large List is ordered data. If the element being stored does not implicitly have an unique value, that can be ordered by a user supplied comparison function.



Large Stack

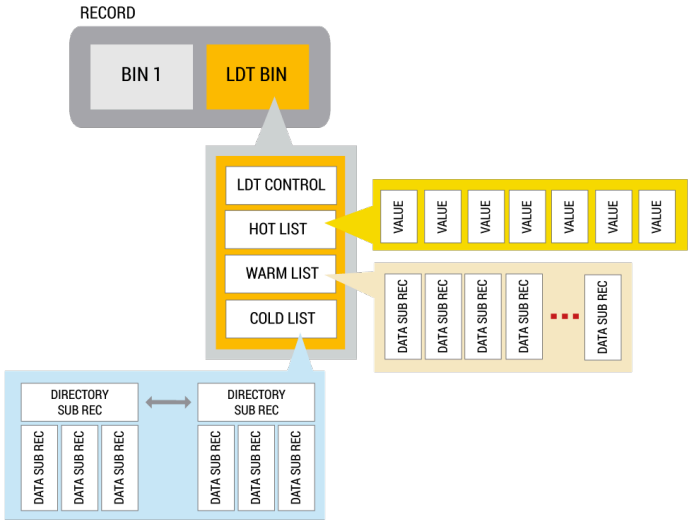
Large Stack follows the familiar mode of "Last In, First Out" (**LIFO**).

The most recent item pushed on the stack is the first item that is popped or peeked from the stack.



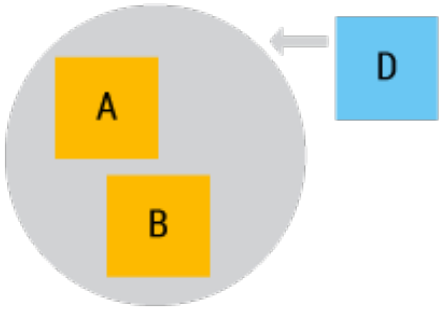
Stack in 3 segments:

- Hot list
- Warm list
- Cold list



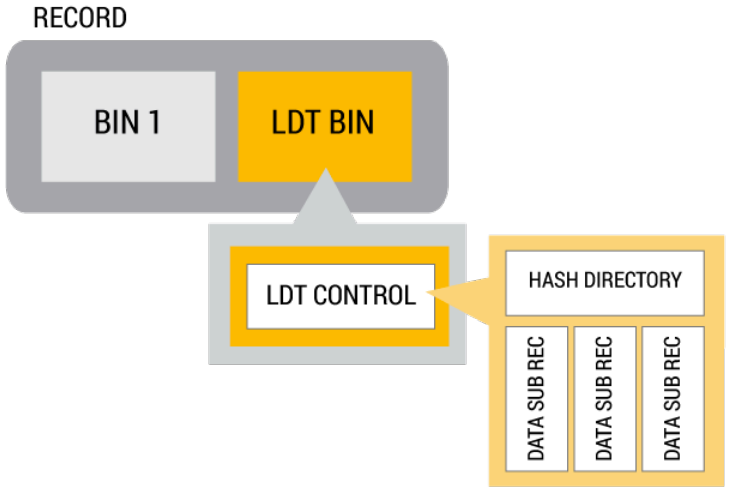
Large Set

A Large Set is a collection of **unique** elements.



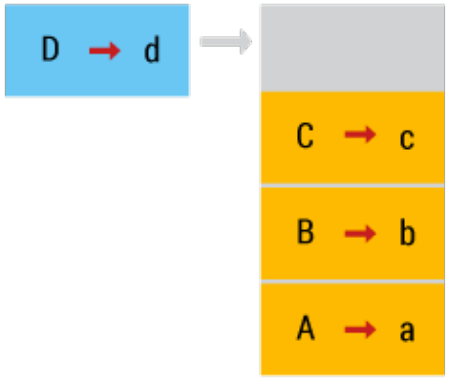
The elements types can be:

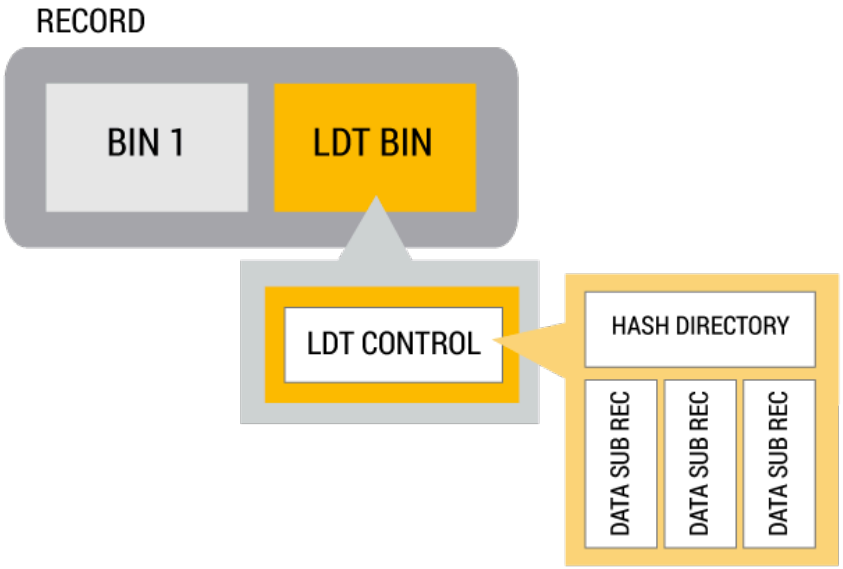
- Number
- String
- Map
- List



Large Map

The Large Map (lmap) implements the basic **name/value** properties of a map on a large scale.





At a glance

Term	RDBMs	Definition	Notes
Cluster	Database	An Aerospike cluster services a single database service.	While a company may deploy multiple clusters, applications will only connect to a single cluster.
Node	-	A single instance of an Aerospike database. A node will act as a part of the whole cluster.	For production deployments, a host should only have a single node. For development, you may place more than one node on a host.
Namespace	Database	An area of storage related to the media. Can be either RAM or flash (SSD based).	
Set	Table	An unstructured grouping of data that have some commonality.	Similar to “tables” in a relational database, but does not require a schema.
Record	Row	A key and all data related to that key.	
Bin	Column	One part of data related to a key.	Bins in Aerospike are typed, but the same bin in different records can have different types. Bins are not required. Single bin optimizations are allowed.

Data Modeling

- Focus on how you will query the data. For example:
  - All Tweets from a given user
  - Give me the last 10 Tweets for a given user
  - Last 10 Tweets for all users
  - How many users Tweeted in the last X minutes

Summary

You have learned about:

- The Aerospike architecture
- The Data Model
  - Namespaces
  - Sets
  - Records
  - Bins and Bin types
  - Large Data Types