Management

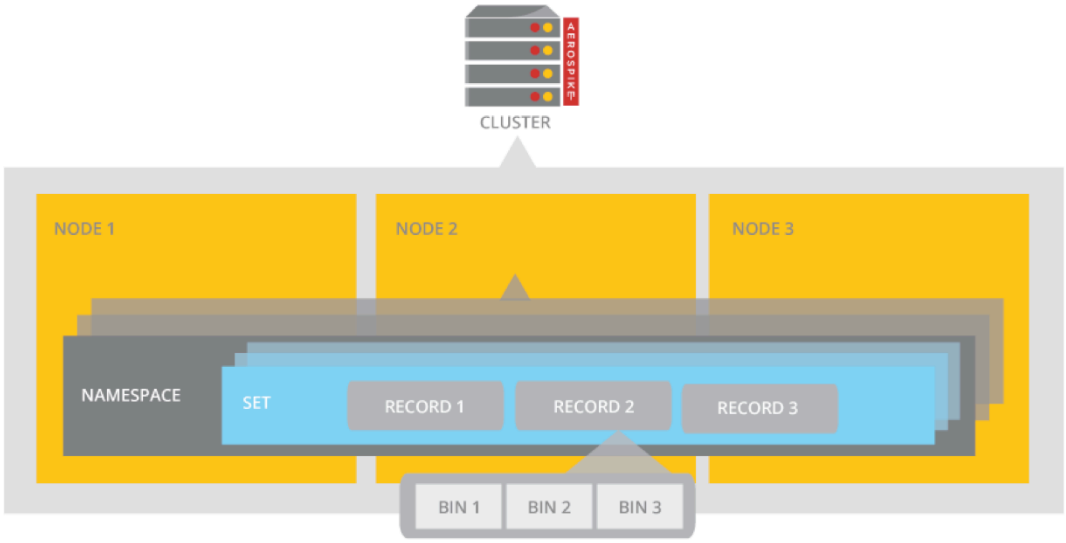
Objectives

At the end of this module, you will be able to:

- Use the Aerospike `aql` command line interface to manage a cluster.
  - View properties.
  - Create secondary indexes.
  - Run basic queries.
- Add data using the Aerospike loader from a CSV file.

This module will give you the basic tools to load and work with data. From here you should be able to do some simple testing of the data within a cluster.

Data Hierarchy



aql - Command Line Interface

aql – Command Line Interface

AQL is the Aerospike command line tool for managing the database with SQL–like commands. Major functions include:

- Record Operations
- Querying Records
- Data Management
- Index Management
- Query Scan Management
- Settings
- Statistics
- UDF Management

aql - Syntax

Common Usage:

```
aql OPTIONS
```

Option	Default	Description
-h	127.0.0.1	Seed host to connect to. AQL will learn about the other nodes in the cluster from this one.
-p	3000	Seed port.
-c	[none]	Command to run.
-f	[none]	Execute the commands in the specified file.
-o	table	One of “json” or “table”. The output format of queries.
--help	[none]	Display usage information.

In most cases, you will enter the `aql` CLI and enter additional commands (unless running a script with the –f command). You will then see the aql prompt “aql> “.

The –h parameter is used to connect to any server in the cluster. This is referred to as the “seed node.” AQL will learn about the other nodes from this one, so you do not need to know about any of the others.

The –p parameter gives the service port for the Aerospike cluster. It is 3000 by default.

The –c parameter allows you to run a command and immediately exit. This can be used if you want to use aql in a shell script.

The –f parameter allows you to run commands from a file, similar to a SQL script.

The –o parameter will allow for out put to look like json rather than tabular format.

For this class, since you are on the database node, you can simply enter “aql”.

aql – Record Operations

You may find that you need to do simple writes or deletes into the database. AQL will allow you to do this from the command line:

Insert A Record

```
aql> INSERT INTO <namespace>[.<set>] (PK, <bins>) VALUES (<key>, <values>)
```

Example

```
aql> INSERT INTO test.testset (PK, gender, age) VALUES ('homer', 'm', 42)
```

Delete A Record

```
aql> DELETE FROM <namespace>[.<set>] WHERE PK=<key>
```

Example

```
aql> DELETE FROM test.testset WHERE PK='homer'
```

Here "PK" is the primary key.

Note that the key is not by default stored as data. You must explicitly store the key, if you need that value.

aql – Data Management

List all namespaces in the cluster

```
aql> show namespaces
+----------------+
| namespaces     |
+----------------+
| "test"         |
| "bar"          |
+----------------+
```

List all sets in the cluster

```
aql> show sets
+-----------+----------------+----------------------+---------+-----------+------------+---------------------+
| n_objects | set-enable-xdr | set-stop-write-count | ns_name | set_name  | set-delete | set-evict-hwm-count |
+-----------+----------------+----------------------+---------+-----------+------------+---------------------+
| 100000    | "use-default"  | 0                    | "test"  | "flights" | "false"    | 0                   |
+-----------+----------------+----------------------+---------+-----------+------------+---------------------+
1 row in set (0.001 secs)
OK
```

List all bins in the namespace

```
aql> show bins
+-------+------------------+-------+-----------+
| quota | bin              | count | namespace |
+-------+------------------+-------+-----------+
| 32768 | "ORIGIN"         | 18    | "test"    |
| 32768 | "FL_DATE_BIN"    | 18    | "test"    |
| 32768 | "YEAR"           | 18    | "test"    |
| 32768 | "DAY_OF_MONTH"   | 18    | "test"    |
| 32768 | "AIRLINE_ID"     | 18    | "test"    |
...
```

These simple commands will be useful in determining whether or not data was loaded into the system.

For namespaces, you cannot add or remove namespaces from an Aerospike cluster without restarting the whole cluster. So this set should be static once the cluster has started.

For sets, you can see

- A count on the number of objects (n_objects).
- Whether or not this set will be synchronized with another cluster.
- A counter on the number of times the set has hit the “stop-write” (read-only) mode. If this increments constantly, this generally means the database is very full and a danger sign.
- The namespace for the set.
- The set name.
- Determines whether or not the set has been marked for deletion.
- How many times the sets has evicted data. If this keeps incrementing, this is another sign the cluster is having space issues.

For bins:

- The total number of bins possible
- The bin name
- The total count of bins
- The namespace for the bins

aql – Index Management

- Create an index on a bin for exact matches
  This will create an index that will allow for exact matches but not range queries. Bins that are not of the correct type will not be indexed.

```
aql> CREATE INDEX <index> ON <namespace>[.set] (<bin>) STRING
```

- Create an index on a bin for numeric range queries
  This will create an index that can be used for things such as range queries on timestamps or other numeric values.

```
aql> CREATE INDEX <index> on <namespace>[.<set>] (<bin>) NUMERIC
```

- Listing indexes
  In order to see a full list of indexes for a namespace or cluster.

```
aql> SHOW INDEXES [<namespace>]
+--------+------+-----------+----------+-------+------------+------------+--------------+
| ns     | bins | set       | num_bins | state | indexname  | sync_state | type         |
+--------+------+-----------+----------+-------+------------+------------+--------------+
| "test" | "b"  | "newtest" | 1        | "RW"  | "numindex" | "synced"   | "INT SIGNED" |
| "test" | "c"  | "newtest" | 1        | "RW"  | "strindex" | "synced"   | "TEXT"       |
+--------+------+-----------+----------+-------+------------+------------+--------------+
```

- Dropping an index
  To remove an existing index:

```
aql> DROP INDEX <namespace> <index>
```

There are 2 different types of indexes, exact match and string.

Note that since Aerospike allows for different records to have bins of different types, that it is possible to have a conflict. For example, one record may have an ID bin that is a string but the same ID bin in another record may have it as an integer. If you index based on string, the integer ID will NOT get indexed.

When you see the indexes, you will have the following:

- The namespace for each index
- The bin being indexed
- The set
- The number of bins
- The state ("RW" means read/write)
- The name of the index
- Whether or not the index has been synchronized across the cluster
- The type of the index.

Example – Create An Index

Run the following command on the server:

```
aql> create index ORIGIN on test.flights (ORIGIN) string
```

Now look at the AMC and see the newly defined index.

Note that you can add this index without defining the set ("flights") or having any data in it.

We will be inserting the appropriate data shortly.

Interlude – Aerospike Loader

Aerospike Loader

The Aerospike Loader is an open source project to help you load data from a CSV file.

The Aerospike Loader is available at:

https://github.com/aerospike/aerospike-loader

It has been pre-loaded on the instance.

Aerospike Loader

The loader uses a json config file. Let's look at an example in: `/home/aerotraining/packages/aerospike-loader/example`. This is airline data that covers domestic flights in North America in January, 2012.

flights_from.csv

```
3,2012,1,2012/11/11,19805,AA,1,12478,JFK,New York, NY,LAX,Los Angeles, CA,855,1142,347,330,2475
4,2012,2,2012/01/02,19805,AA,1,12478,JFK,New York, NY,LAX,Los Angeles, CA,921,1210,349,325,2475
5,2012,3,2012/01/03,19805,AA,1,12478,JFK,New York, NY,LAX,Los Angeles, CA,931,1224,353,319,2475
6,2012,4,2012/01/04,19805,AA,1,12478,JFK,New York, NY,LAX,Los Angeles, CA,904,1151,347,309,2475
7,2012,5,2012/01/05,19805,AA,1,12478,JFK,New York, NY,LAX,Los Angeles, CA,858,1142,344,306,2475
8,2012,6,2012/01/06,19805,AA,1,12478,JFK,New York, NY,LAX,Los Angeles, CA,911,1151,340,321,2475
9,2012,7,2012/01/07,19805,AA,1,12478,JFK,New York, NY,LAX,Los Angeles, CA,902,1203,361,337,2475
10,2012,8,2012/01/08,19805,AA,1,12478,JFK,New York, NY,LAX,Los Angeles, CA,855,1129,334,318,2475
...
```

flights_from.json

```
{
  "version" : "1.0",
  "input_type" : "csv",
  "csv_style": {
        "delimiter": "," ,
        "n_columns_datafile": 19,
        "ignore_first_line": false
  },

  "key": {
        "column_position": 1,
        "type": "integer"
  },
  "binlist": [
        {"name": "YEAR",
     "value": {"column_position": 2, "type" : "integer"}
    },
       {"name": "DAY_OF_MONTH",
     "value": {"column_position": 3, "type" : "integer"}
    },
        {"name": "FL_DATE_BIN",
     "value": {"column_position": 4, "type" : "timestamp", "encoding":"yyyy/MM/dd", "dst_type":
"integer"}
    },
...
```

Notice that column 4 is a date format.

The json file for column 4 includes special formatting options to convert that date into an integer value. This is useful for range queries.

Loading Example Data

To load the data issue the following commands:

```
> cd /home/aerotraining/packages/aerospike-loader
> ./run_loader -n test -s flights -c ./example/
flights_from.json ./example/flights_from.csv
```

This will load 100,000 records into the database.

In this case we are loading data into the namespace “test” and the set “flights”

It will read the configuration from the json file

It will load the CSV.

aql – Querying Records

You may often find you need to query data to make sure that it was properly entered into the database.

Select all data in a set/namespace. This is similar to selecting all rows in a table.

```
aql> SELECT * FROM <namespace>[.<set>]
```

Try the following:

```
aql> SELECT * FROM test.flights
```

Note that because we are doing a dump of the entire set, this is for 100,000 records and will take a bit of time.

Hit ctrl-C to stop it; press it once.

aql – Querying Records

Select specific bins from a set/namespace. This will select only the specified bins.

```
aql> select <bin>[,<bin>[...]] FROM
<namespace>[.<set>]
```

Now try it on the sample data

```
aql> select CARRIER,ORIGIN,DEST from test.flights
```

By restricting the bins to just the ones that we are interested in, we can see the data in a much more compact format.

aql – Querying Records

Filtering on Indexed Bins

```
aql> select <bin>[,<bin>[...]] FROM <namespace>[.<set>] WHERE
<predicate>
```

The predicate can be and equality “<bin>=<value>” or if it is numeric a range “<bin> BETWEEN <lower> and <upper>”. Try the following:

```
aql> select CARRIER,FL_NUM,ORIGIN,DEST from test.flights where
 ORIGIN='JFK'


aql> select CARRIER,FL_NUM,ORIGIN,DEST from test.flights where
 DEST='JFK'
```

Because the index for the bin “DEST” is not defined, you will get an error. So try the following:

```
 aql> create index DEST on test.flights (DEST) string
 aql> select CARRIER,FL_NUM,ORIGIN,DEST from test.flights where DEST='JFK'
```

Note that you cannot currently use compound predicates ("and"/"or").

We can now specify that we only want records that match a specific value.

Using the Predicate “DEST = ‘JFK’” will not work because we have not defined the index for the bin. This is somewhat different from a standard relational database, where you can always query on a column.

You can create the index after the data has been loaded and Aerospike will index the data as fast as it can. With only 100k records in our database, this will not take long.

Note that you cannot use compound predicates such as “where ORIGIN=‘LAX’ AND DEST=‘JFK’”

aql – Query Scan Management

Query management takes resources, so must be enabled (and disabled when done).
To enable from the Linux command line:

```
> asinfo -v "set-config:context=service;query-job-tracking=true"
```

Then you can run commands from the aql command line:

- List queries

```
aql> SHOW QUERIES
```

- List scans
  Scans can be run from a client and often take many minutes or hours.

```
aql> SHOW SCANS
```

- Kill running jobs
  If you find that a query or scan is running a long time or needs to be cancelled, you can kill it from the aql command line:

```
aql> KILL_QUERY <query_id>
aql> KILL_SCAN <scan_id>
```

Exercise: Try running the command from aql "select * from test.flights" and see it in the query. You will need to do this from 2 different windows.

Query management can take a lot of resources. So you must turn on the feature.

Turn on the tracking feature when needed and turn off when you have done.

Objectives

What we have covered:

- Use the Aerospike `aql` command line interface to manage a cluster.
  - View properties.
  - Create secondary indexes.
  - Run basic queries.
- Add data using the Aerospike loader from a CSV file.

This module will give you the basic tools to load and work with data. From here you should be able to do some simple testing of the data within a cluster.