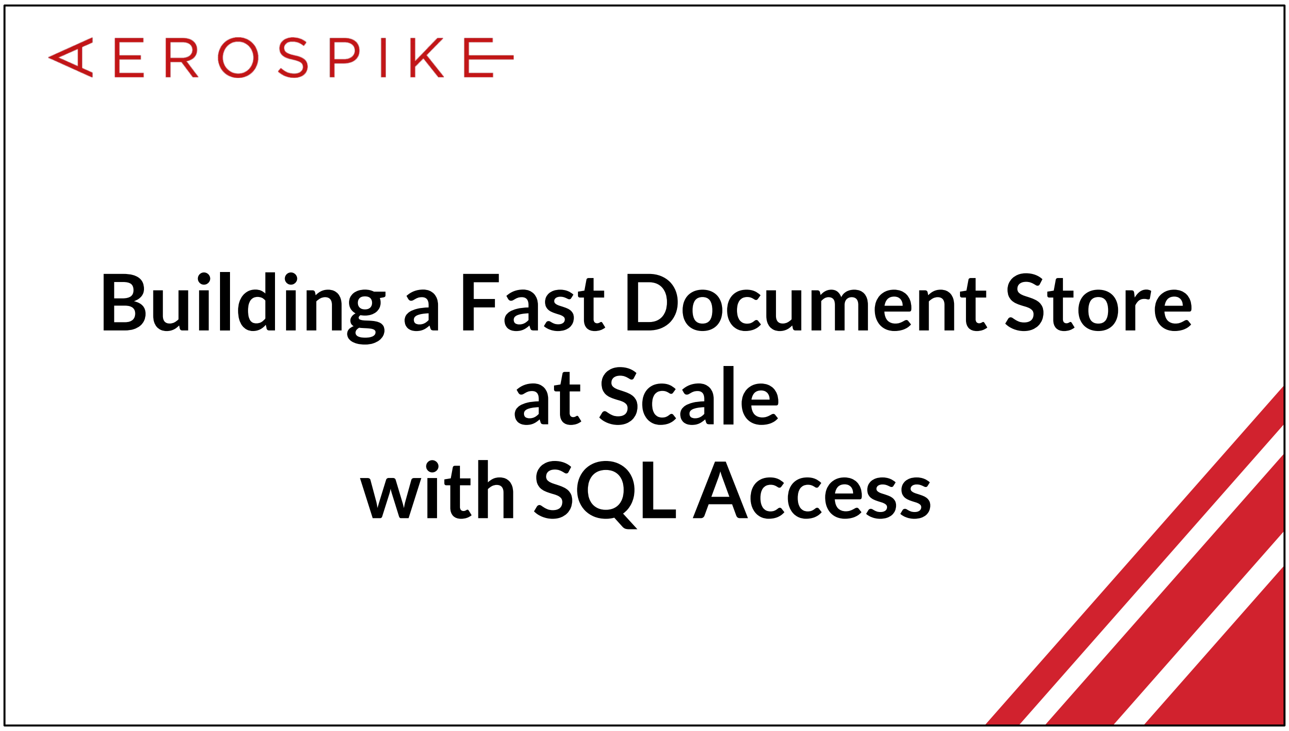


AEROSPIKE

Building a Fast Document Store at Scale with SQL Access

HELLO FROM AEROSPIKE!

Neel Phadnis

nphadnis@aerospike.com
github.com/neelp-git
linkedin.com/in/neel-phadnis

Director of Developer Engagement

HELLO
FROM
AEROSPIKE!

Art Anderson

aanderson@aerospike.com
github.com/artanderson
linkedin.com/in/artdanderson
@Art_Anderson_

Developer Advocate

Overview

What is Aerospike?

Building a Document Store

- Modeling JSON documents
- Document API with JSONPath
- Querying with filters and indexes
- SQL access with Trino/Presto
- Considerations for scale

What is Aerospike?

Real-time

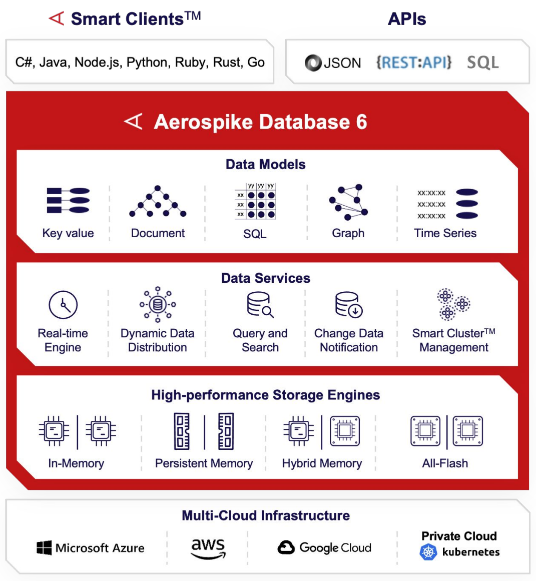
Low latency

High throughout

Scalable

Multi-model

Data platform



Real-time data platform for predictable low latency access at any scale with multiple data models including JSON documents.

Real-time = predictable low-latency + always-on resiliency
- Disk io bottleneck of traditional databases is overcome with hybrid memory architecture (HMA) to store indexes and data in DRAM and SSD, and optimized SSD io.
- Incremental garbage collection avoids GC slowdowns, minimal locking schemes to avoid hot spot bottlenecks.
- Hash-based placement ensures uniform data distribution to all nodes and uniform resource utilization.
- Cluster changes and failures are handled with data movement and request routing.
- Smart client library provides a single hop access to data and transparent handling of cluster transitions and data movements..

Easy and efficient scaling
- Higher SSD density vs DRAM means more data per node and smaller cluster size.
- Cluster can grow or shrink effortlessly without concerns for data distribution or loss.
- Clusters can be globally distributed.

- Should developers care about scaling?
    - Yes! Non-linear scaling will hit a wall in performance, reliability, and cost.
    - Large clusters are costly, and difficult to support.
    - Painful transition of database change, caching layer, consistency issues,, …
    - Always on call for proverbial 2am production issues…

Data model, distribution, and terminology

- Namespace = database: Many in a cluster. Governed by policies such as storage, replication, consistency, etc.
- Set = table: Many in a namespace.
- Partition: 4096 data partitions in a namespace, each with RF replicas.
- Record: Record key or digest is a RIPEMD160 hash of (set, id), and determines the record's partition.
- Bin = field: A record can have any number of bins. A bin can hold a value of any type.
- Schemaless!

A write updates all replicas ensuring they are consistent.

What is Aerospike?

- Real-time
- Low latency
- High throughout
- Scalable
- Multi-model
- Data platform

- Financial Services
- Gaming
- AdTech/MarTech
- IOT/Edge
- Telco

What is Aerospike?

Real-time

Low latency

High throughout

Scalable

Multi-model

Data platform

Hybrid Memory Architecture (HMA)
Store data with traditional in-memory, unique Hybrid-Flash, All-Flash, or Persistent Memory storage architecture to optimize for performance and scale.

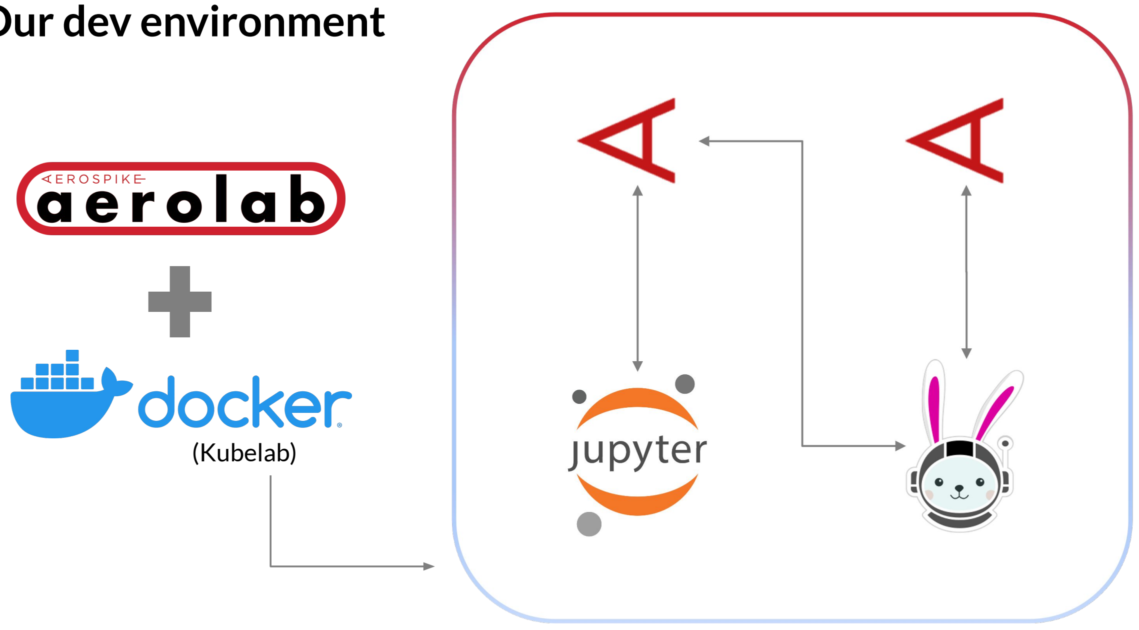
Intelligent Clients
Automatically distributes both data and traffic to all the nodes in a cluster.

Dynamic Cluster Management
Automatic storage and exchange of node information including changes and status with automated triggers for rebalancing.

Strong Consistency
No data loss. Commit-to-device support for demanding correctness environments.

Cross-Data Center Replication (XDR)
Global, asynchronous replication between clusters creating global data hubs

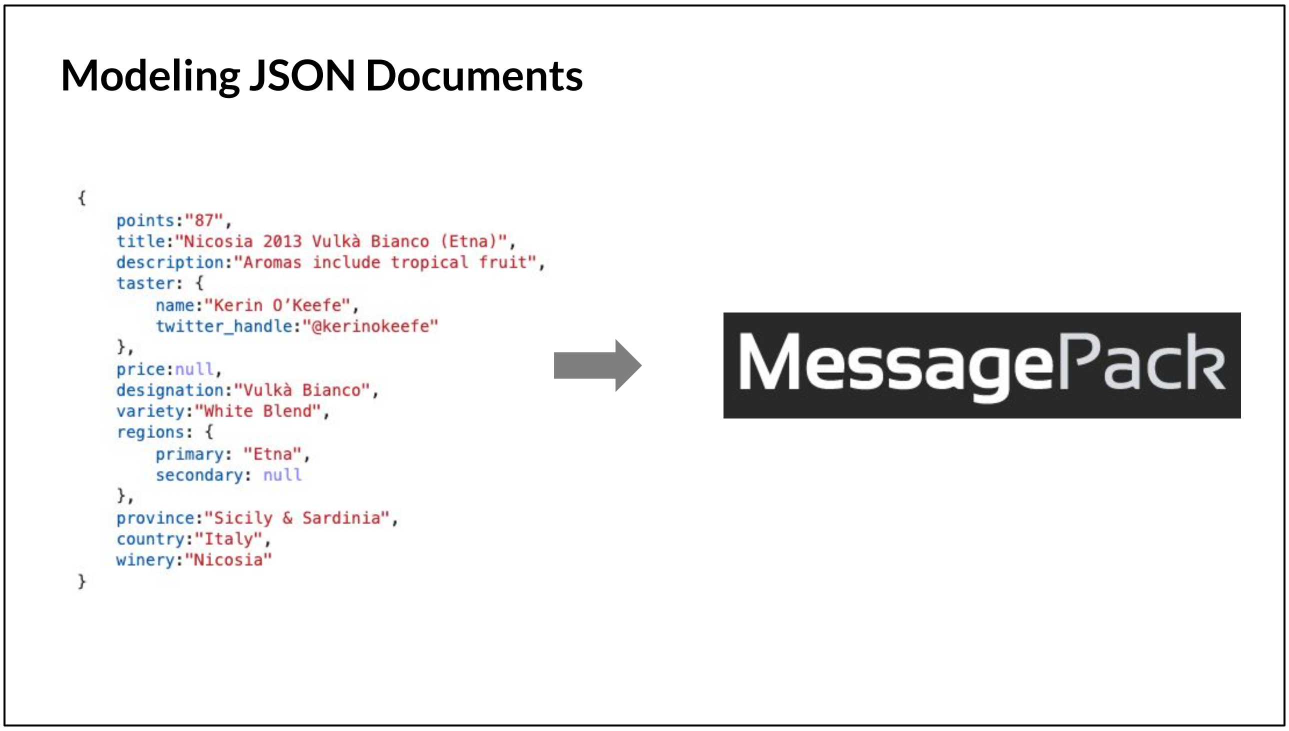
Our dev environment



Aerolab and Kubelab together create a development environment with two Aerospike clusters, a Trino server, and a Jupyter notebook server.

TRY IT YOURSELF!

https://github.com/aerospike-examples/document-sql-demo



Modeling JSON Documents

```
{
    points:"87",
    title:"Nicosia 2013 Vulkà Bianco (Etna)",
    description:"Aromas include tropical fruit",
    taster: {
        name:"Kerin O’Keefe",
        twitter_handle:"@kerinokeefe"
    },
    price:null,
    designation:"Vulkà Bianco",
    variety:"White Blend",
    regions: {
        primary: "Etna",
        secondary: null
    },
    province:"Sicily & Sardinia",
    country:"Italy",
    winery:"Nicosia"
}
```

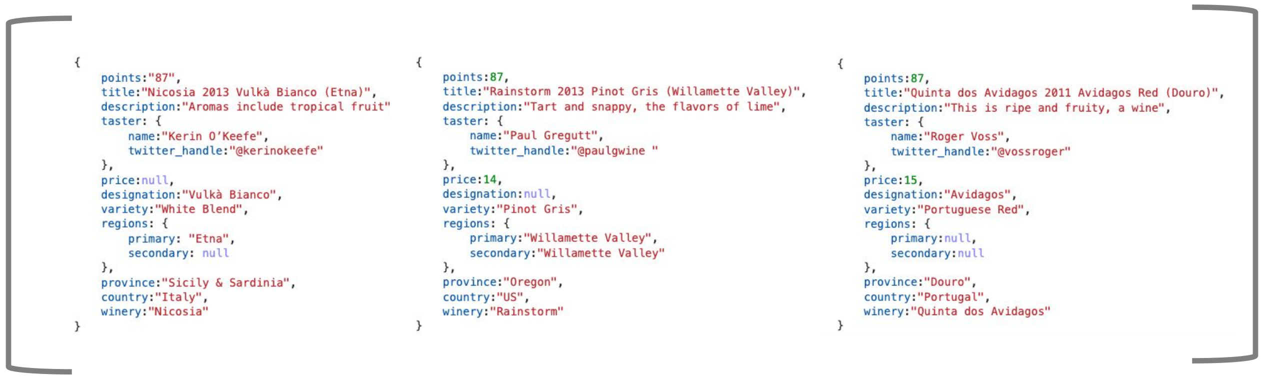
MessagePack

Storing JSON data
- Stored as a nested map, a collection data type (CDT).
- MessagePack is the storage and transfer format.

A JSON document can be representation of one or many objects.
- A Bin may store one or many JSON documents.
- Document can be organized in multiple records, sets, namespaces, and even clusters.

Key modeling considerations:
- Efficiency of access: Defining indexes, organizing in groups and sets.
- Efficiency of space: Grouping multiple objects in one record.
    - Record key design for key-based access. For example, stock ticker data for a day.

Modeling JSON Documents



```
{
    points:"87",
    title:"Nicosia 2013 Vulkà Bianco (Etna)",
    description:"Aromas include tropical fruit"
    taster: {
        name:"Kerin O'Keefe",
        twitter_handle:"@kerinokeefe"
    },
    price:null,
    designation:"Vulkà Bianco",
    variety:"White Blend",
    regions: {
        primary: "Etna",
        secondary: null
    },
    province:"Sicily & Sardinia",
    country:"Italy",
    winery:"Nicosia"
}
```

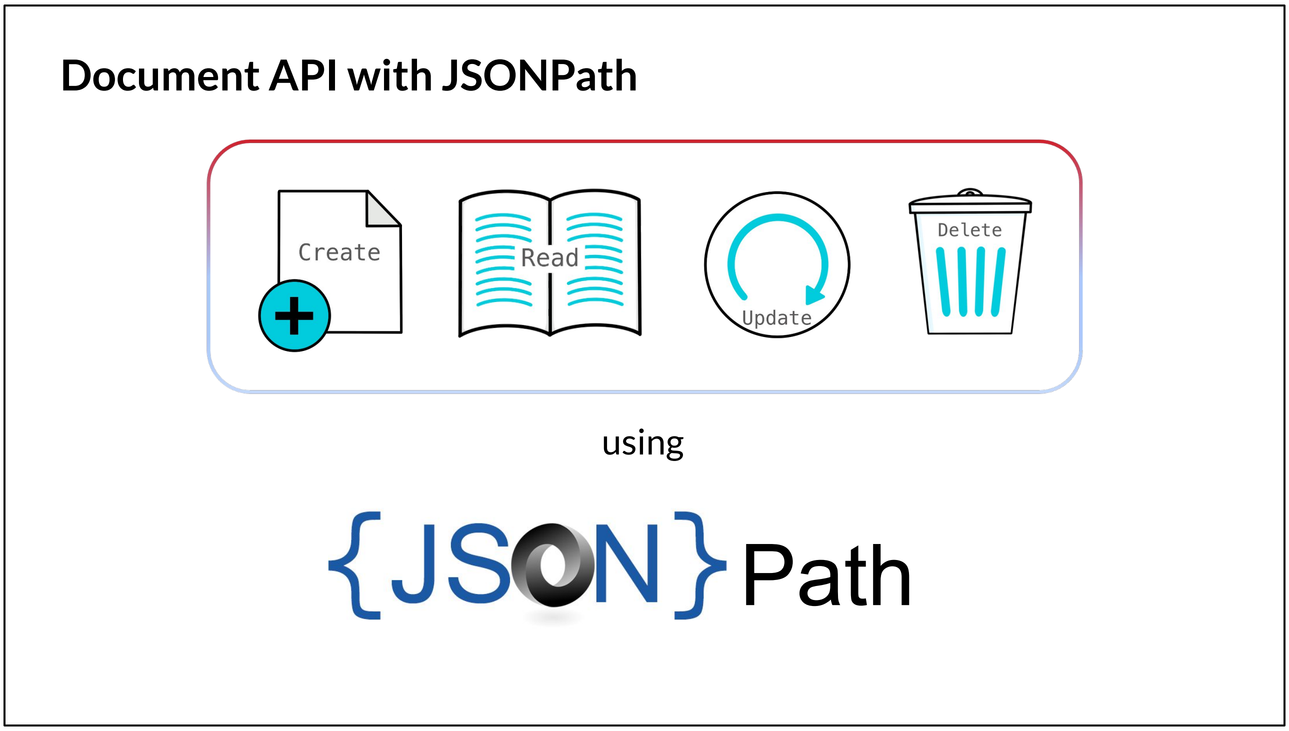
```
{
    points:87,
    title:"Rainstorm 2013 Pinot Gris (Willamette Valley)",
    description:"Tart and snappy, the flavors of lime",
    taster: {
        name:"Paul Gregutt",
        twitter_handle:"@paulgwine "
    },
    price:14,
    designation:null,
    variety:"Pinot Gris",
    regions: {
        primary:"Willamette Valley",
        secondary:"Willamette Valley"
    },
    province:"Oregon",
    country:"US",
    winery:"Rainstorm"
}
```

```
{
    points:87,
    title:"Quinta dos Avidagos 2011 Avidagos Red (Douro)",
    description:"This is ripe and fruity, a wine",
    taster: {
        name:"Roger Voss",
        twitter_handle:"@vossroger"
    },
    price:15,
    designation:"Avidagos",
    variety:"Portuguese Red",
    regions: {
        primary:null,
        secondary:null
    },
    province:"Douro",
    country:"Portugal",
    winery:"Quinta dos Avidagos"
}
```

CHECK IT OUT!

Steps/scenarios:

1. Load a JSON file holding a collection of wine reviews.
2. Store all reviews in one record (using Document API).
3. Store each review in a separate record (using Client API).



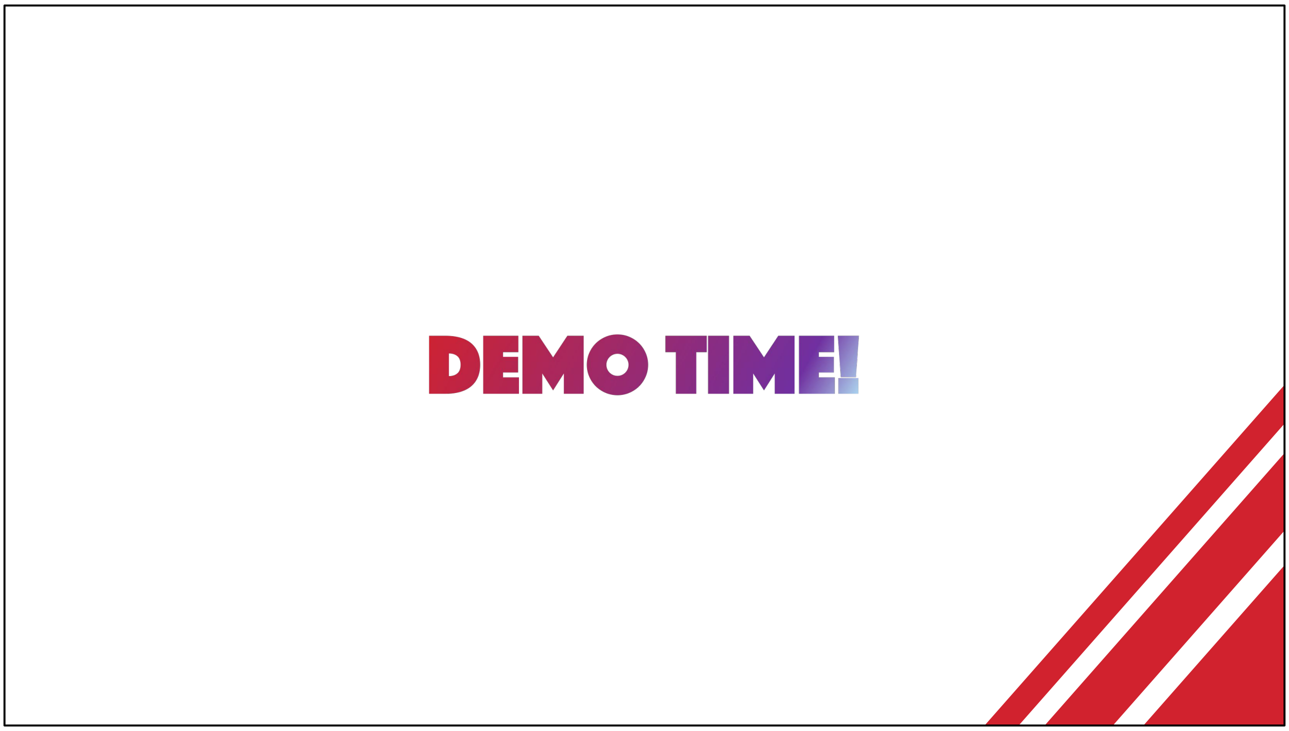
Document API with JSONPath

using

{JSON} Path

Document API supports JSON operations

- CRUD operations with JSONPath syntax
- Expressions, indexing, queries, nested element operations, batch operations supported with CDTs (internal representation of JSON)



DEMO TIME!

Steps/scenarios:

1. Read the JSON file and INSERT in a single bin.
2. READ a wine review.
3. QUERY multiple reviews with JSONPath.
4. UPDATE a review with JSONPath.
5. UPDATE multiple reviews with JSONPath.
6. DELETE one or more reviews with JSONPath.

Querying with filters and indexes

Apply filters to a query

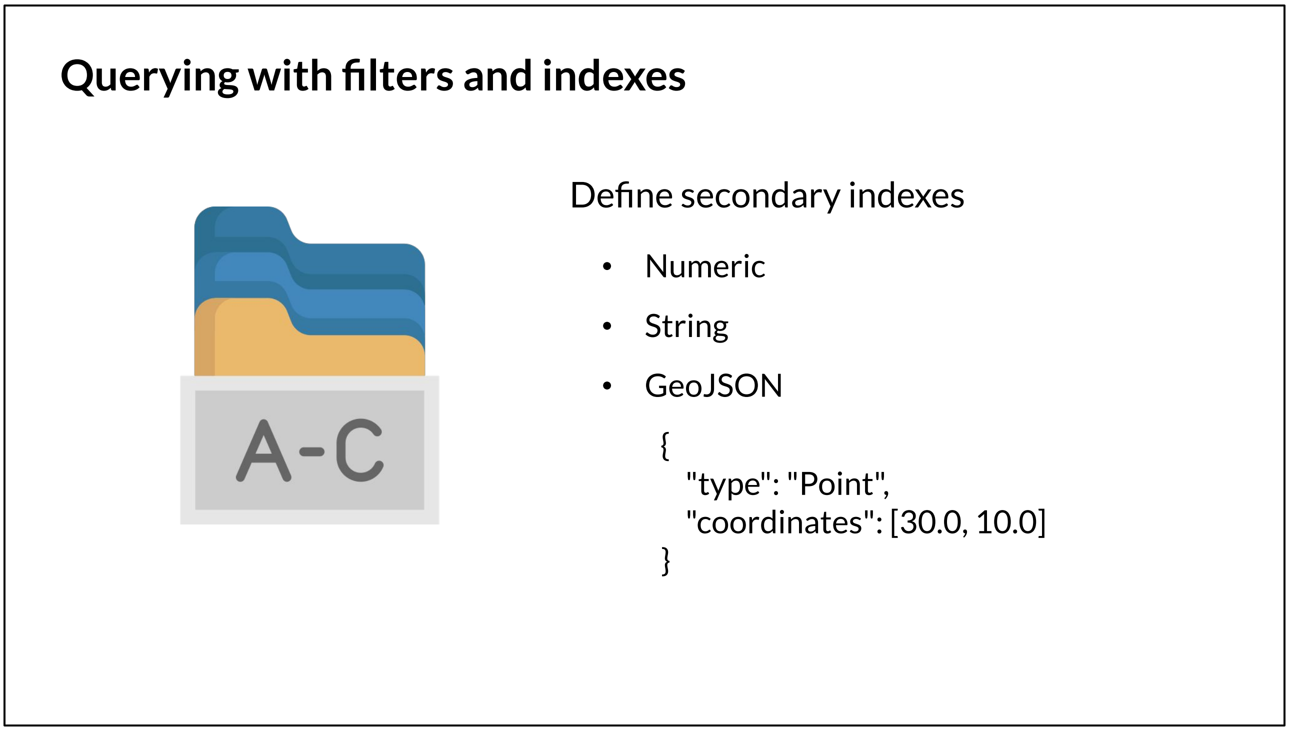
- Metadata
  - Digest
  - Size
  - Last-update-time
  - Time-to –live
  - Version
- Document fields

A document collection can be queried using
- a filter and/or
- a secondary index.

A filter is a binary expression on metadata (digest, size, LUT, TTL, version) and document fields.

A secondary index is defined on a numeric, string, or geojson values at any nested level.
- An array and object in JSON can also be indexed to retrieve the record.

image: Flaticon.com



Querying with filters and indexes

Define secondary indexes

- Numeric
- String
- GeoJSON

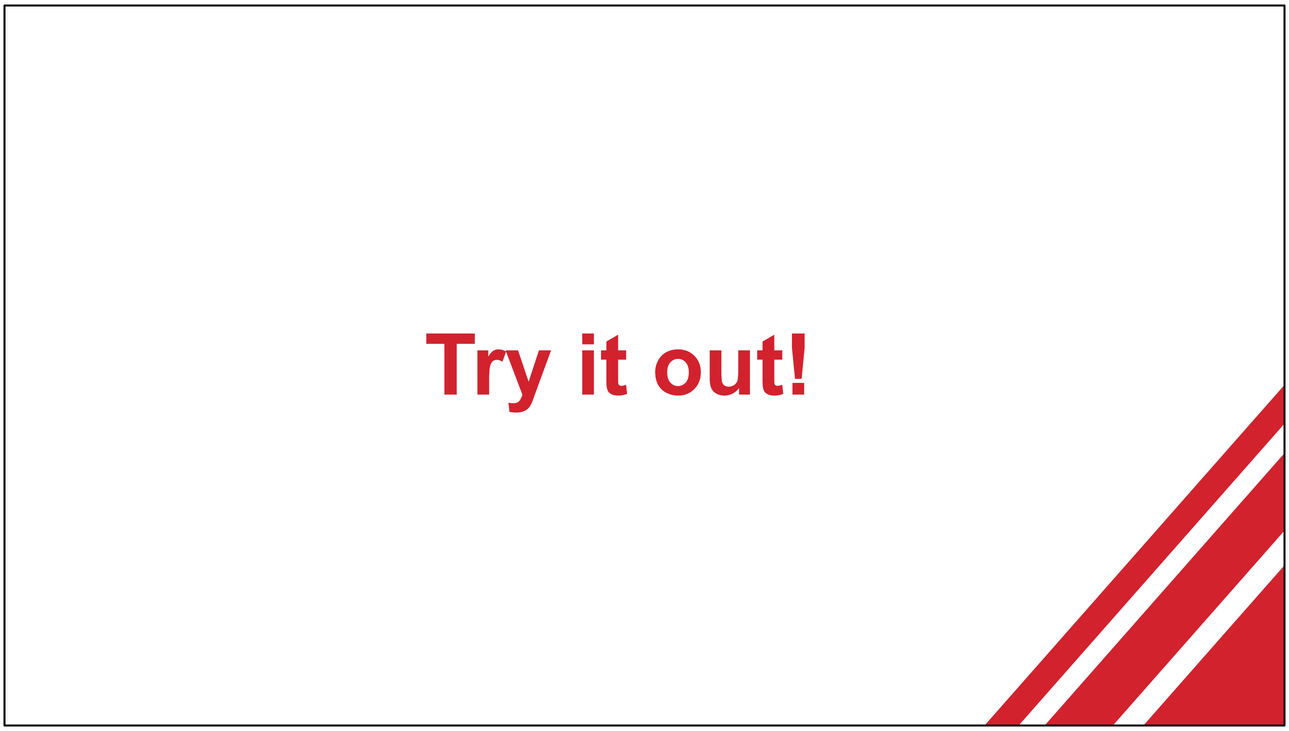
```
{
   "type": "Point",
   "coordinates": [30.0, 10.0]
}
```

A document collection can be queried using
- a filter and/or
- a secondary index.

A filter is a binary expression on metadata (digest, size, LUT, TTL, version) and document fields.

A secondary index is defined on a numeric, string, or geojson values at any nested level.
- An array and object in JSON can also be indexed to retrieve the record.

image: Flaticon.com



Try it out!

Steps/scenarios:

1. Create a secondary index on an integer/string field.
2. Query with a secondary-index predicate.
3. Query with a filter.
4. Query using both a filter and secondary-index predicate.
5. Query using a Map index.

SQL access with Trino*/Presto

- Aerospike Trino Connector provides fast processing of SQL queries on Aerospike data.
- Data from an operational cluster can be sync'd to analytics store in near real-time using XDR.
- Works with Trino's Cost Based Optimizer to optimize queries by pushing down predicates and projections, as well as leveraging indexes.

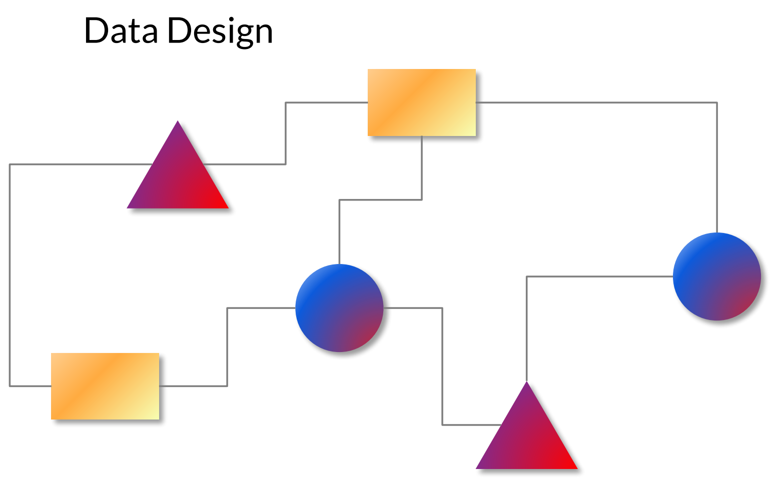
*Trino is a fork of Presto.

GIVE IT A SHOT!

Steps/scenarios:

1. SQL query with a predicate pushdown.
2. SQL query with JSON extract.
3. Aggregation query (eg, count, sum, min, max, etc).
4. Federated query with another catalog defined in a different database.

Considerations for scale



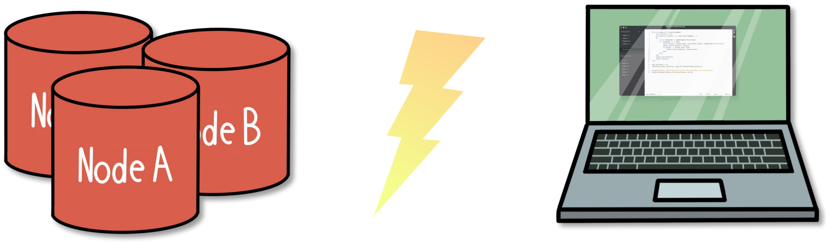
Aerospike cluster scales linearly over the number of objects and data size.

Considerations for building a large-scale document store:

- Data design
  - Grouping in a record and organization in sets
  - Global distribution
- Fast access
  - Indexes
  - Batch operations
  - Async operations
- Server operations
  - Collection data types
  - Expressions
  - Backend updates
  - Multi-ops
- Spark, Trino, and streaming connectors

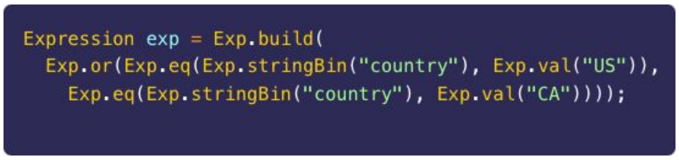
Considerations for scale

Fast Access



Considerations for scale

Operations



```
Expression exp = Exp.build(
  Exp.or(Exp.eq(Exp.stringBin("country"), Exp.val("US")),
    Exp.eq(Exp.stringBin("country"), Exp.val("CA"))));
```

Considerations for scale

Connectors

Starburst

kafka

APACHE Spark™

presto

Summary

Aerospike provides low-latency access to real-time applications at scale.

Built a demo store with JSON data for wine reviews.

Demonstrated CRUD operations with JSONPath syntax.

Performed programmatic queries using filters and indexes.

Issued SQL queries via Trino.

Highlighted key features for scaling up the document store.

Photo by Alan King: https://unsplash.com/@alanking on Unsplash: https://unsplash.com

More resources

Developer Hub AEROSPIKE

https://developer.aerospike.com

Documentation AEROSPIKE

https://docs.aerospike.com

GitHub

https://github.com/aerospike-examples/document-sql-demo

AEROSPIKE aerolab

https://github.com/aerospike/aerolab

THANK YOU!

ANY QUESTIONS?