Workshop 2 – Using Couchbase

Agenda

- What is NoSQL?
- What is a document database?
- Why NoSQL?
- Using the .NET SDK to interact with Couchbase

Couchbase
DEVELOPER COMMUNITY

Open source FTW

- https://github.com/couchbaselabs/aspnet-nosql-workshop

- If you find a typo, mistake, or spot an improvement, please send a pull request!

NoSQL: Document Database

- NoSQL is an umbrella term
- We’ll be looking at a subset called “document databases”
- It's like a key/value database:
  - The key is some unique identifier
  - The value is in a known format (typically JSON)

Document

Key: Foo::123::456

Value:
```
{
        "name" : "Matt",
        "twitter" : "@mgroves",
        "favoriteMovies" : [
            "Star Wars",
            "Willy Wonka",
            "Glitter"
        ],
        "type" : "user"
}
```

Why ~~NoSQL~~ document databases?

- Architecture
- Performance
- Scaling
- Flexibility

Document database use cases

- Big data
- Profile management
- Content management
- Customer 360 view
- IoT
- Fraud detection
- Catalogs
- Personalization
- Digital communication
- Caching
- Mobile (with Couchbase Mobile)

https://www.couchbase.com/use-cases

NoSQL: Saving data

- NoSQL operations
  - Insert
  - Update
  - Upsert
- SQL (N1QL) options
  - INSERT
  - UPDATE
  - DELETE
  - MERGE
  - etc

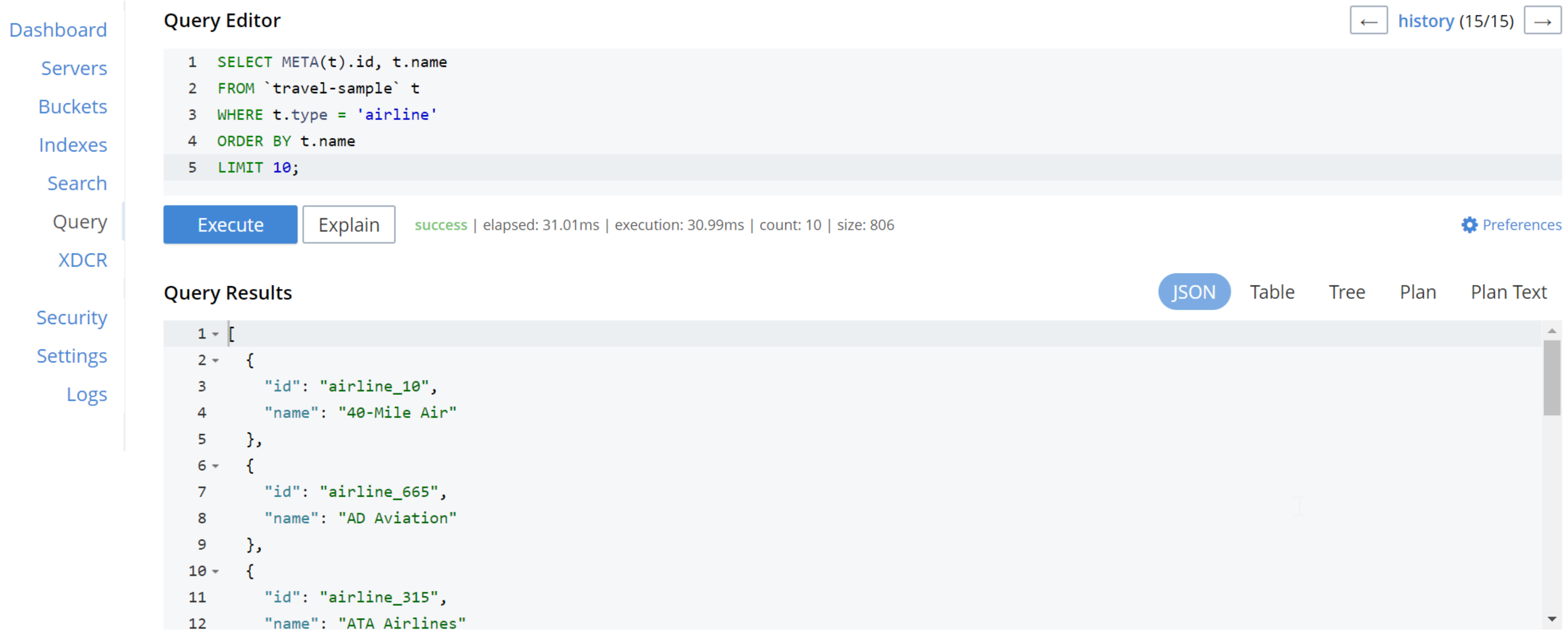
NoSQL: Retrieving Data

- NoSQL operations
  - Get (by key)

- SQL (N1QL) options
  - SELECT

Query Workbench

http://localhost:8091



```
SELECT META(t).id, t.name
FROM `travel-sample` t
WHERE t.type = 'airline'
ORDER BY t.name
LIMIT 10;
```

success | elapsed: 31.01ms | execution: 30.99ms | count: 10 | size: 806

```
[
  {
    "id": "airline_10",
    "name": "40-Mile Air"
  },
  {
    "id": "airline_665",
    "name": "AD Aviation"
  },
  {
    "id": "airline_315",
    "name": "ATA Airlines"
```

Exercise

How to install .NET SDK

NuGet: Install-Package CouchbaseNetClient

Connect to Couchbase

```
var config = new ClientConfiguration();
config.Servers = new List<Uri> {
    new Uri("couchbase://localhost")
};
_cluster = new Cluster(config);
_cluster.Authenticate(new PasswordAuthenticator("matt","password"));
_bucket = _cluster.OpenBucket("workshop");
```

Insert a document

```
IDocument<dynamic> doc = new Document<dynamic>
{
    Id = Guid.NewGuid().ToString(),
    Content = new
    {
        firstName = "Connie",
        lastName = "James",
        city = "Columbus, Ohio",
        country = "USA",
        type = "person"
    }
};
_bucket.Insert(doc);
```

Getting document(s) by key(s)

```
IOperationResult<dynamic> aDocument = _bucket.Get<dynamic>("key");

dynamic result = await _bucket.GetAsync<dynamic>(id);
```

Demo

Workshop 1: N1QL

- Create a primary index on default bucket

```
CREATE PRIMARY INDEX on `default`
```

- Add document(s) to default bucket
  - Use “Create Document” button in Documents view
- SELECT documents

```
SELECT d.* FROM `bucketname` d
```

Workshop 2: Using SDK in "Hello World"

- Write a console program that:
  - Insert document(s)
  - Select documents with N1QL

Questions