JAVA DEVELOPER WORKSHOP

Manuel Hurtado

Solutions Engineer at Couchbase

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

AGENDA

Lab Environment

Introduction to Data Modeling

Travel-sample data model

Connecting to Couchbase

Key/value and CRUD

Working with Parts of a Document

Querying for Documents with N1QL

Batching with RXJava

Full Text Search

Spring Boot

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

LAB ENVIRONMENT

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 0: Installing Couchbase

DOWNLOAD

https://www.couchbase.com/downloads

DEPLOYMENT OTIONS

https://developer.couchbase.com/documentation/server/current/install/get-started.html

 **Non-cloud On-Premises**	 **Virtualized/Containerized Environment**	 **Public Cloud**
Run Couchbase within your bare metal data center.	Run Couchbase within a virtualized or containerized environment such as VMware or Docker.	Deploy and manage Couchbase in a public cloud such as Amazon Web Services(AWS), Microsoft Azure, or Google Cloud Platform.

Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 0: Installing Couchbase

DOCKER

- Documentation: https://developer.couchbase.com/documentation/server/current/install/docker-deploy-single-node-cluster.html
- Single server deployment for this lab
- TIP: create local directory and map to enable persistence upon recreating containers

```
docker run -d -p 8091-8094:8091-8094 -p 11210:11210 --name cb1 -v
/[FULL_PATH]/cb1:/opt/couchbase/var couchbase
```

Lab

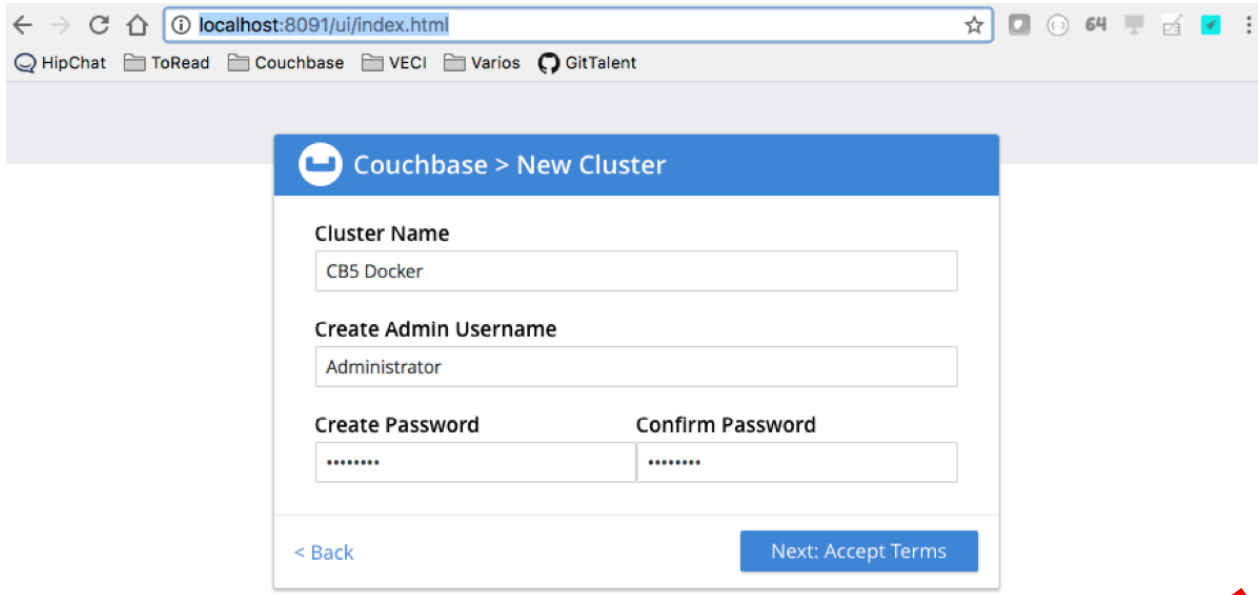
Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 1: Initial Setup

Access UI Web Console: http://localhost:8091/ui/index.html

Enter Cluster Name

Administration User: Adminsitrator:password



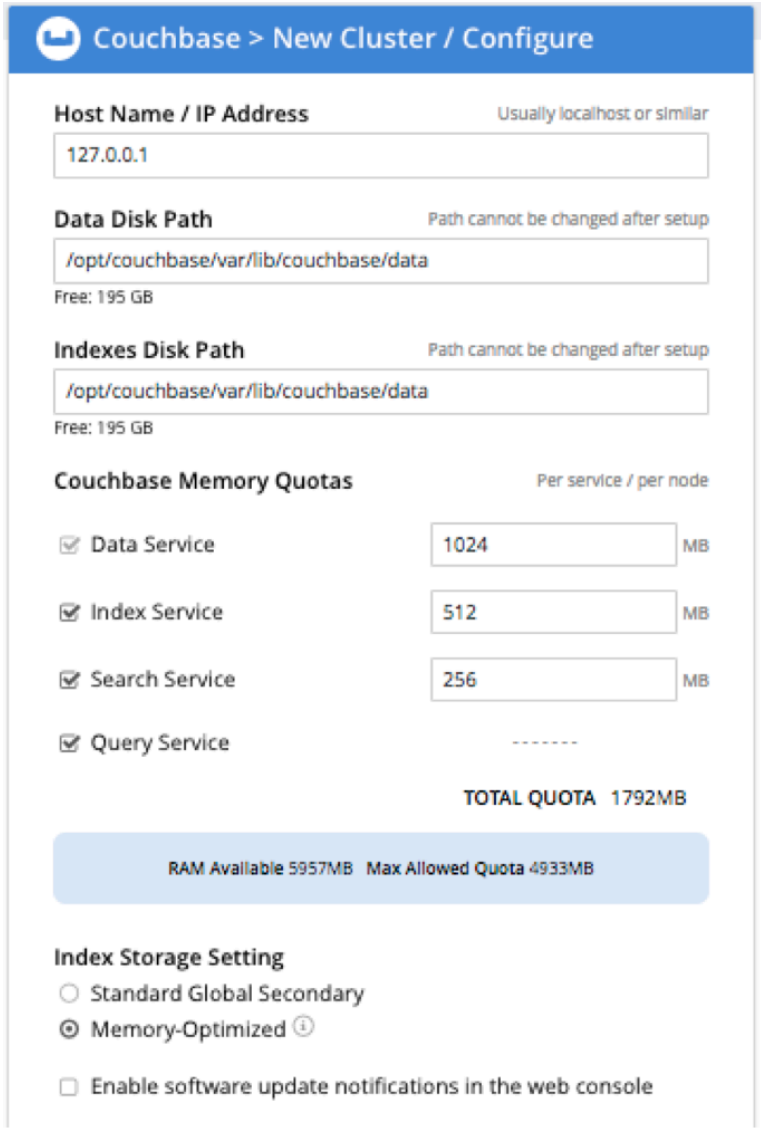
Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 1: Initial Setup

Memory Quotas:

- Data 1024 Mb
- Index: 246 Mb
- Search: 2456 Mb

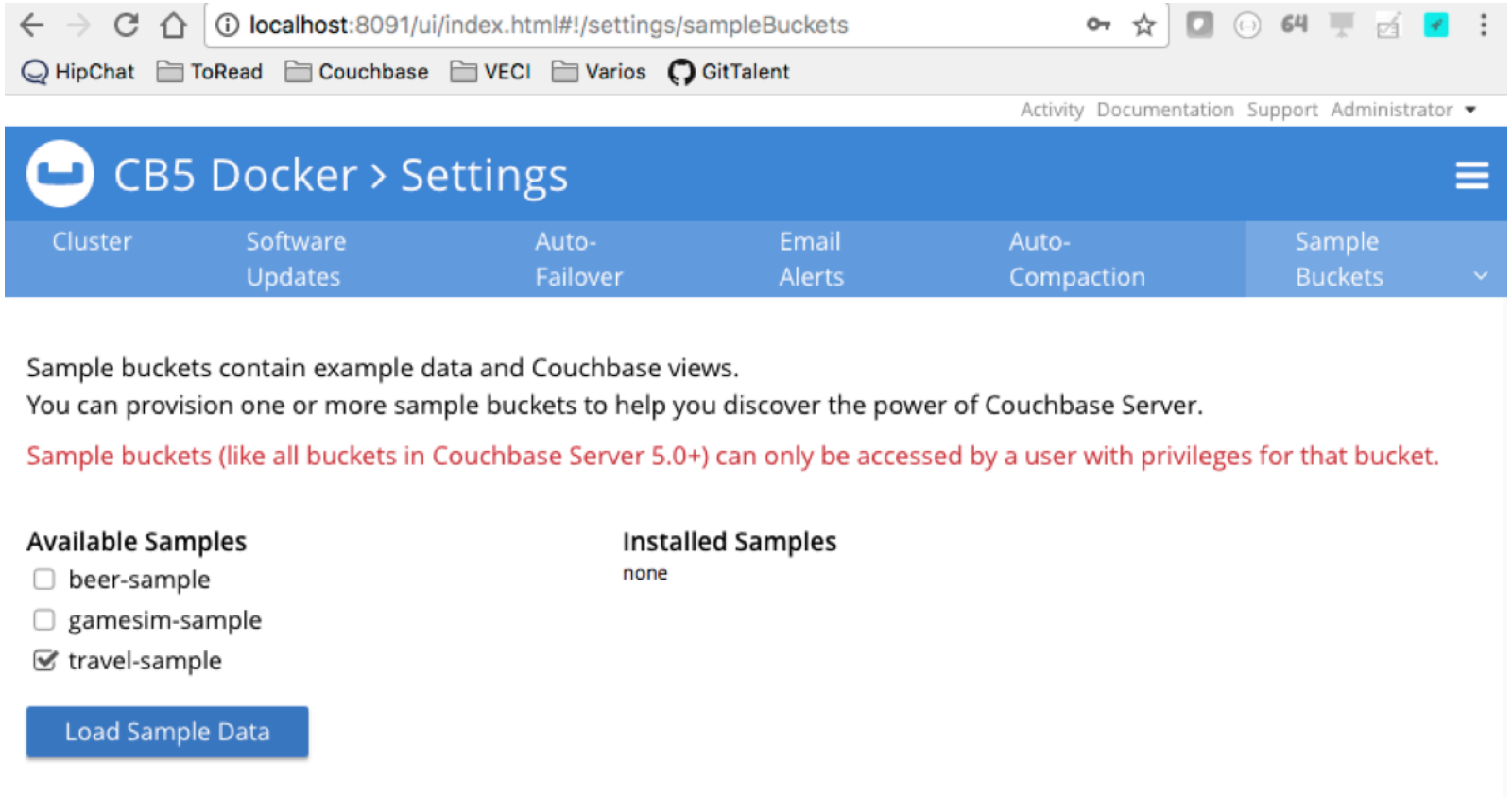


Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 1: Initial Setup
Install sample data set

Install travel-sample data set: Admin Console > Settings > Sample Buckets

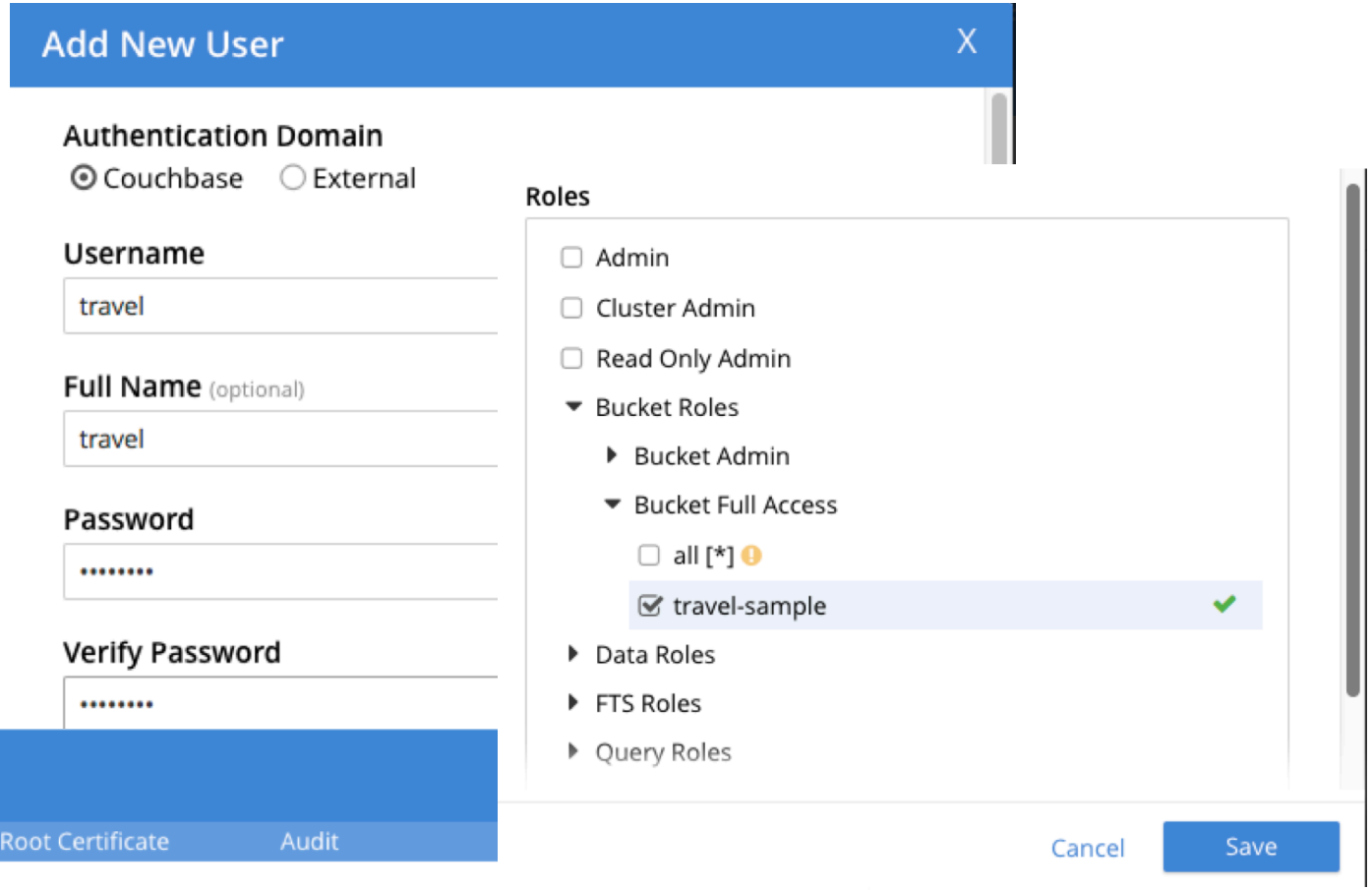


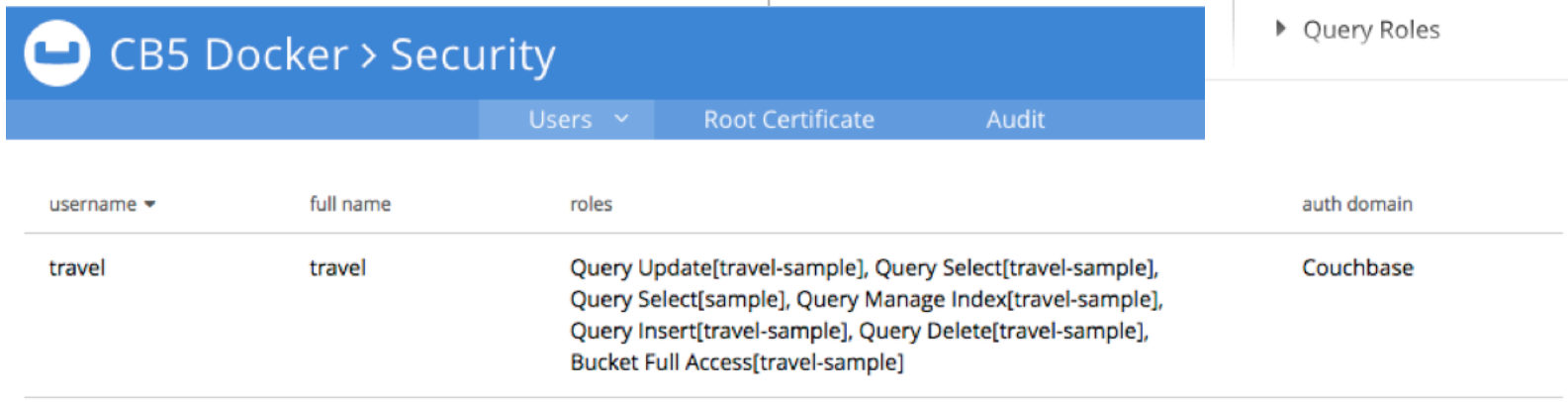
Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 1: Initial Setup
Create User

- Username: travel
- Password: password
- Roles:
  - Bucket Full Access[travel-sample]
  - Query Update[travel-sample]
  - Query Select[travel-sample]
  - Query Manage Index[travel-sample]
  - Query Insert[travel-sample]
  - Query Delete[travel-sample]





Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 1: Initial Setup
Clone repo

Download the workshop code from this repository:

```
git clone https://github.com/mahurtado/CouchbaseJavaWorkshop.git
```

Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

INTRODUCTION TO DATA MODELING

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

The Plan – Introduction to Data Modeling

- Designing a Document Key
- Embedding Data as Complex JSON in Documents
- Create References Between Documents

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Building a Document Key

- Key design is as important as document design.
- There are three broad types of key:
  - Human readable/deterministic: e.g. an email address
  - Computer generated/random: e.g. UUID
  - Compound: e.g. UUID with a deterministic portion

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Human Readable Document Keys

```
key: nic@couchbase.com
```

```
{
    "name": ”Nic Raboy",
    ”email”: ”nic@couchbase.com",
    "address": ”450 Mission Street",
    "city": ”San Francisco”,
    “state”: “CA”
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Generated Document Keys

```
key: f8d8d616-6ce0-4344-9793-4ec5a676c223
```

```
{
    "name": ”Nic Raboy",
    ”email”: ”nic@couchbase.com",
    "address": ”450 Mission Street",
    "city": ”San Francisco”,
    “state”: “CA”
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Compound Document Keys

```
key: profile::f8d8d616-6ce0-4344-9793-4ec5a676c223
```

```
{
    "name": ”Nic Raboy",
    ”email”: ”nic@couchbase.com",
    "address": ”450 Mission Street",
    "city": ”San Francisco”,
    “state”: “CA”
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Counter Document Keys

```
key: msg::66565
```

```
{
    ”date": ”2018-01-02 23:55:55",
    ”user”: ”nic@couchbase.com",
    ”msg”: “This is a test message”
}
```

Counters documentation:
https://developer.couchbase.com/documentation/server/5.0/sdk/core-operations.html#story-h2-5

```
// Java
// Create the counter
 bucket.counter(“myCounter”, 0, 20);
// Use counter for new key
String id = ”msg::" + bucket.counter((“myCounter”, 1).content();;
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Mixed Document Key Example

```
key: f8d8d616-6ce0-4344
```

```
{
    "name": ”Nic Raboy",
    ”email”: ”nic@couchbase.com",
    "address": ”450 Mission Street",
    "city": ”San Francisco”,
    “state”: “CA”
}
```

```
key: facebook::1234
```

```
{
    “profile”: “f8d8d616-6ce0-4344”
}
```

```
key: twitter::1234
```

```
{
    “profile”: “f8d8d616-6ce0-4344”
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Relational Databases

Customers		
customer	firstname	lastname
2342	Nic	Raboy

Products	
product	name
7182	Nintendo Switch

Orders		
order	total	customer
1234	$299.99	2342

Invoice		
order	product	quantity
1234	7182	2

```
SELECT
    c.firstname, c.lastname, p.name, o.total
FROM invoice AS i
JOIN order AS o ON i.order = o.order
JOIN product AS p ON i.product = p.product
JOIN customer AS c ON o.customer = c.customer
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

The Embedded Approach

Customers		
customer	firstname	lastname
2342	Nic	Raboy

Products	
product	name
7182	Nintendo Switch

Orders		
order	total	customer
1234	$299.99	2342

Invoice		
order	product	quantity
1234	7182	2

```
{
    "order": 1234,
    "customer": {
        "customer": "2342",
        "firstname": "Nic",
        "lastname": "Raboy"
    },
    "products": [
        {
            "product": 7182,
            "name": "Nintendo Switch"
        }
    ],
    "total": "$299.99"
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

The Referenced Approach

Customers		
customer	firstname	lastname
2342	Nic	Raboy

Products	
product	name
7182	Nintendo Switch

Orders		
order	total	customer
1234	$299.99	2342

Invoice		
order	product	quantity
1234	7182	2

```
{
    "firstname": "Nic",
    "lastname": "Raboy"
}
```

```
{
    "name": "Nintendo Switch"
}
```

```
{
    "order": 1234,
    "product": 7182,
    "quantity": 2
}
```

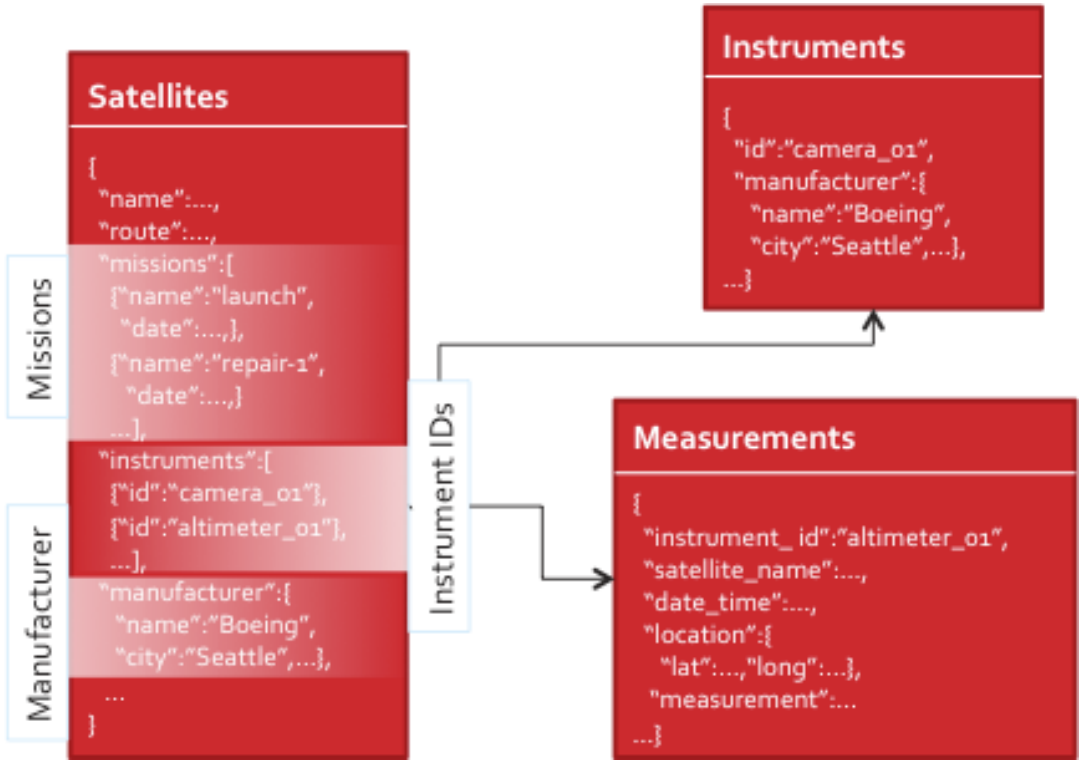
```
{
    "customer": 2342
    "total": "$299.99"
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Embedding vs Referencing

- Embed When
  - Both docs **are** frequently access together.
  - You need strict consistency between both docs.

- Reference When
  - Both docs **aren't** frequently access together.
  - You need strict consistency for the referenced information
  - Need to optimize for performance and referenced doc is large



Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Evolving Data Models

Change your code and you are good to go!

- Schema is App Driven in Couchbase Server
  - OK to maintain multiple versions of the schema
- Schema Changes with Couchbase Server
  - No Explicit Server Action Required by DBA
  - No Downtime required to update existing data to new shape

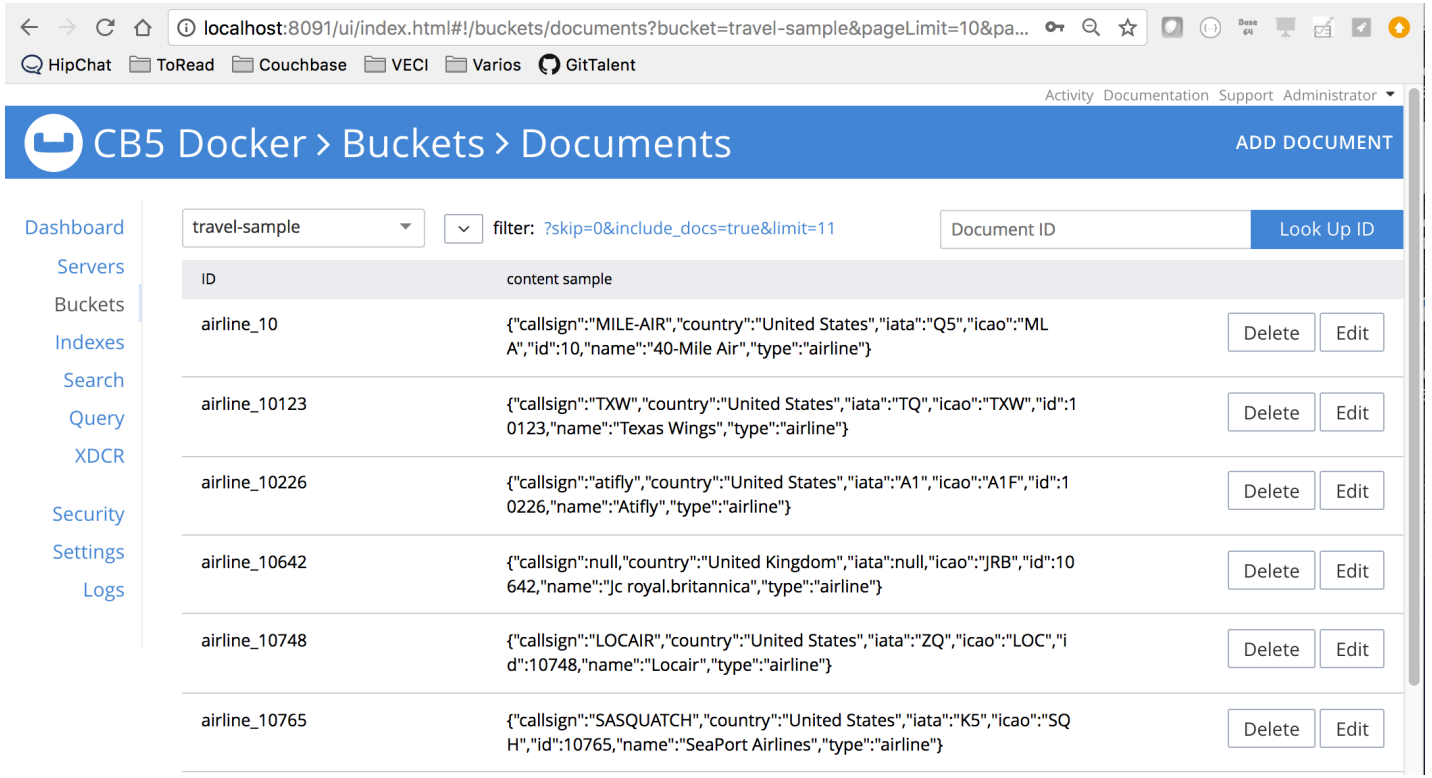
Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

TRAVEL-SAMPLE DATA MODEL

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 2: Browse documents from the UI Console

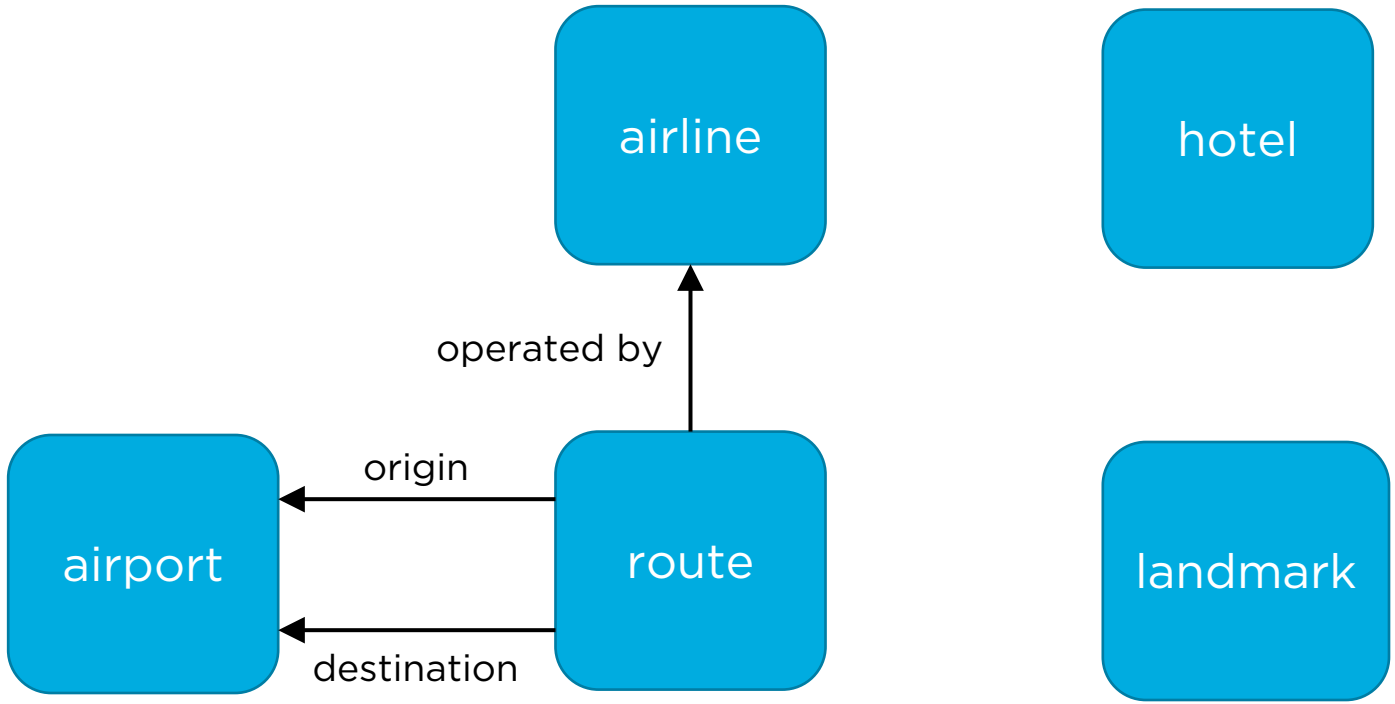
- Console > Buckets > travel-sample > Documents
- Try filter > start with > r
- Try Lookup ID: **route_10003**
- Create document. Add Document >
  - Document ID: k_1
  - Value {"name": "luis"}
- Delete document. ID: k_1



Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Travel-sample data model



Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Travel-sample data model: airline

- Sample key: airline_10123

airline

```
{
   "callsign": "TXW",
   "country": "United States",
   "iata": "TQ",
   "icao": "TXW",
   "id": 10123,
   "name": "Texas Wings",
   "type": "airline”
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Travel-sample data model: airport

airport

- Sample key: airport_1256

```
{
   "airportname": "Les Loges",
   "city": "Nangis",
   "country": "France",
   "faa": null,
   "geo": {
      "alt": 428,
      "lat": 48.596219,
      "lon": 3.006786},
   "icao": "LFAI",
   "id": 1256,
   "type": "airport",
   "tz": "Europe/Paris"
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Travel-sample data model: route

- Sample key: route_10004

route


```
{
    "airline": "AF",
    "airlineid": "airline_137",
    "destinationairport": "AMS",
    "distance": 9442.50092891188,
    "equipment": "777",
    "id": 10004,
    "sourceairport": "TPE",
    "stops": 0,
    "type": "route",
```



```
"schedule": [
 {
   "day": 0,
   "flight": "AF545",
   "utc": "13:22:00"
 },
 {
   "day": 0,
   "flight": "AF350",
   "utc": "01:21:00"
 },
 ... ,
 {
   "day": 6,
   "flight": "AF250",
   "utc": "02:32:00"
 }]
}
```


Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Travel-sample data model: hotel

- Sample key: hotel_10063

hotel


```
{
   "address": "6 rue aux Juifs",
   "alias": "Les Rouges Gorges",
   "checkin": null,
   "checkout": null,
   "city": "Giverny",
   "country": "France",
   "description": "The rustic style ...
",
   "directions": "27620 Giverny",
   "email": null,
   "fax": null,
   "free_breakfast": false,
   "free_internet": true,
   "free_parking": true,
   "geo": {
      "accuracy": "APPROXIMATE",
      "lat": 49.078069,
      "lon": 1.520866
   },
```

```
"geo": {
      "accuracy": "APPROXIMATE",
      "lat": 49.078069,
      "lon": 1.520866
   },
   "id": 10063,
   "name": "The Robins",
   "pets_ok": false,
   "phone": null,
   "price": "60 / 70 euros",
   "public_likes": [],
   "reviews": [ ... ],
   "state": "Haute-Normandie",
   "title": "Giverny",
   "tollfree": null,
   "type": "hotel",
   "url":
"http://givernyguesthouse.com/robin.h
tm",
   "vacancy": true
}
```


Confident ed.

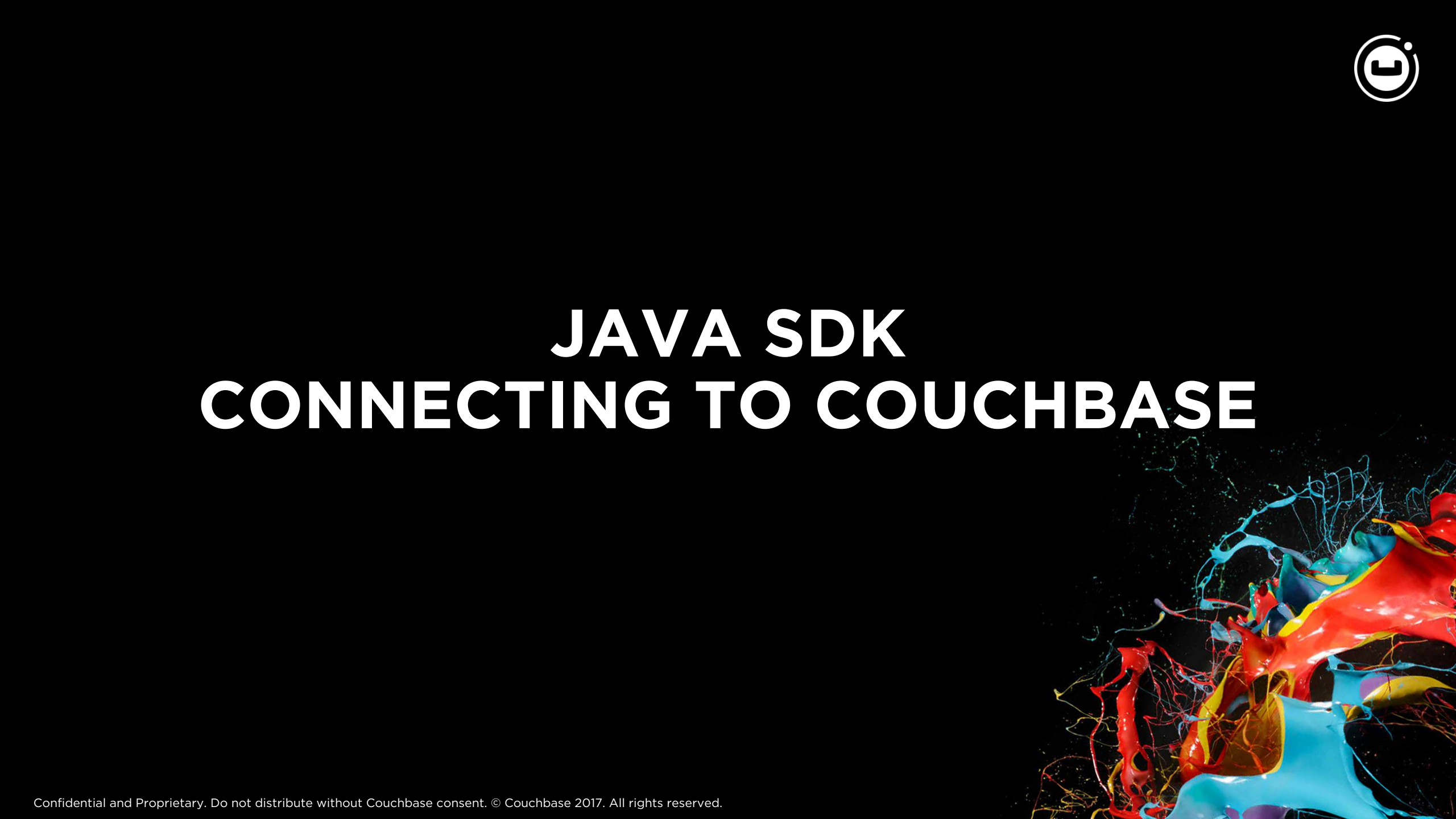
Travel-sample data model: landmark

landmark

- Sample key: landmark_10023


```
{
   "activity": "eat",
   "address": "3 King Street, ME6 1EY",
   "alt": null,
   "city": "Gillingham",
   "content": "Chinese restaurant just off the High Street.",
   "country": "United Kingdom",
   "directions": null,
   "email": null,
   "geo": {
      "accuracy": "RANGE_INTERPOLATED",
      "lat": 51.38697,
      "lon": 0.54852  },
   "hours": null,
   "id": 10023,
   "image": null,
   "name": "Beijing Inn",
   "phone": null,
   "price": null,
   "state": null,
   "title": "Gillingham (Kent)",
   "tollfree": null,
   "type": "landmark",
   "url": http://beijinginn.co.uk/div/
}
```


Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.



JAVA SDK
CONNECTING TO COUCHBASE

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Including the SDK

```
// Gradle
dependencies {
    ...
    compile group: 'com.couchbase.client', name: 'java-client', version:'2.5.1'
    ...
}
```

```
// Maven
<dependencies>
    <dependency>
        <groupId>com.couchbase.client</groupId>
        <artifactId>java-client</artifactId>
        <version>2.5.1</version>
    </dependency>
</dependencies>
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Connection Basics

```
// Java
import com.couchbase.client.java.CouchbaseCluster;
import com.couchbase.client.java.CouchbaseBucket;

CouchbaseCluster cluster = CouchbaseCluster.create("couchbase://<host>");
cluster.authenticate("<username>", "<password>");
CouchbaseBucket bucket = cluster.openBucket("<bucket-name>");
```

- Not like a JDBC Pool!
- Reuse Bucket object
- Singleton pattern

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 3: Couchbase Connection

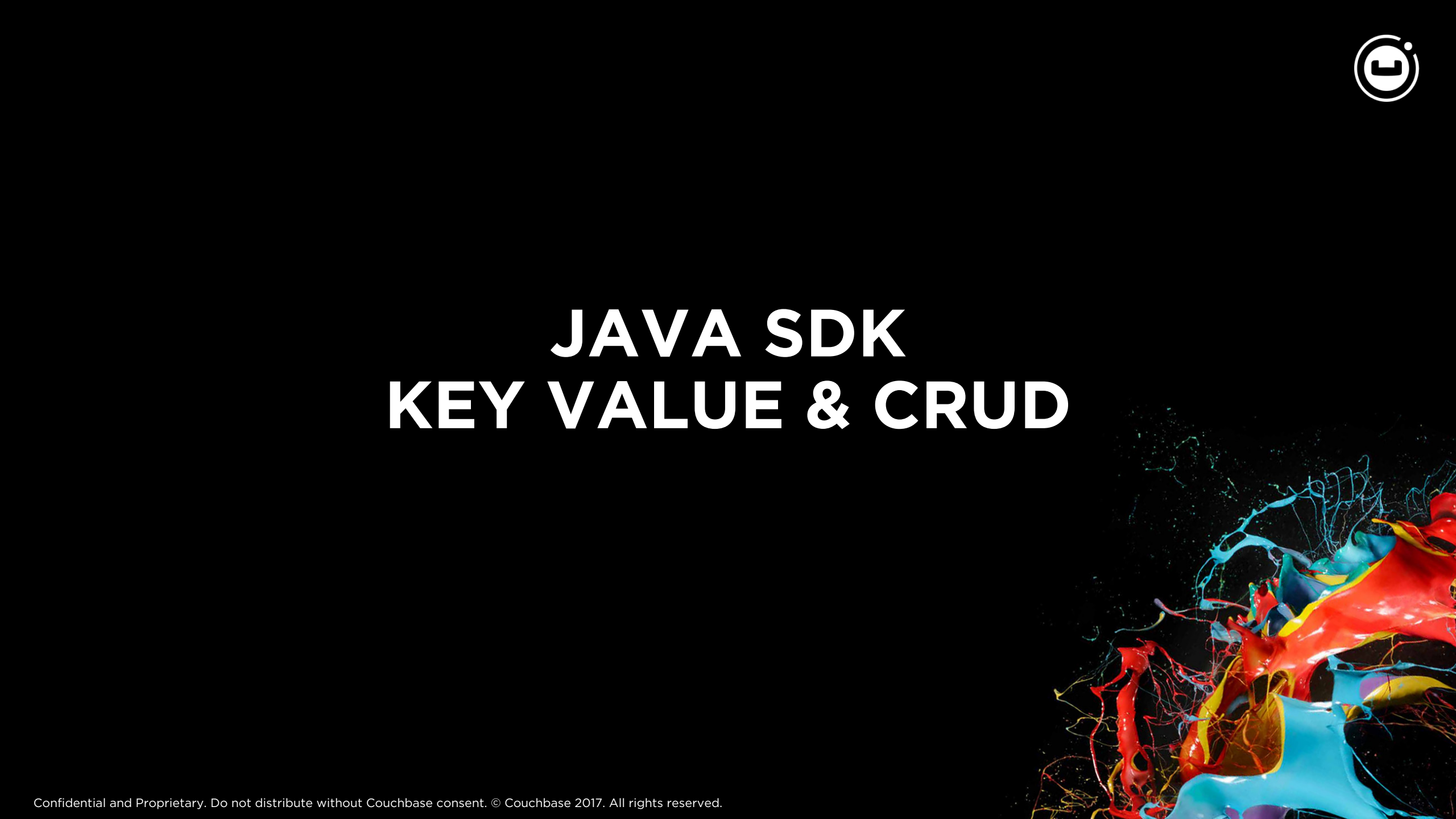
- Implement method initConnection()
- Read config values from System properties:
  - cbworkshop.clusteraddress
  - cbworkshop.user
  - cbworkshop.password
  - cbworkshop.bucket
- Initialize bucket
- Run application
- Check output:

```
Nov 20, 2017 1:44:25 PM com.couchbase.client.core.CouchbaseCore <init>
INFO: CouchbaseEnvironment: {sslEnabled=false, sslKeystoreFile='null',
sslTruststoreFile='null', sslKeystorePassword=false, sslTruststorePassword=false,
sslKeystore=null, sslTruststore=null, bootstrapHttpEnabled=true,
bootstrapCarrierEnabled=true, bootstrapHttpDirectPort=8091, bootstrapHttpSslPort=18091,
. . .}

Nov 20, 2017 1:44:33 PM com.couchbase.client.core.config.DefaultConfigurationProvider$8
call
INFO: Opened bucket travel-sample

Nov 20, 2017 1:44:35 PM com.couchbase.client.core.node.CouchbaseNode signalConnected
INFO: Connected to Node 127.0.0.1/localhost
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.



JAVA SDK
KEY VALUE & CRUD

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Creating Documents

- Data can be flat or complex
- Document keys can be custom, automatically generated, or incrementing
- The `insert` operator will create new documents if the key does not already exist
- The `upsert` operator will create or replace

```
JsonObject data = JsonObject.create()
    .put("firstname", "Nic")
    .put("lastname", "Raboy");
JsonArray address = JsonArray.create()
    .add(JsonObject.create().put("city", "Mountain View").put("state", "CA"))
    .add(JsonObject.create().put("city", "San Francisco").put("state", "CA"));
data.put("address", address);
bucket.insert(JsonDocument.create("person-1", data));
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Retrieving Documents by Key

- Data can be retrieved using a key-value lookup or with a N1QL query
- Lookups are significantly faster than indexed queries with N1QL
- Java has can be asynchronous (non-blocking) with RxJava or synchronous

```
// Java
bucket.get("person-1").content();
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 4: Create Object

- Implement method: create(String[] words)
- Compose a JSON document like this:
- Read parameters from command line:
  - Document key
  - from
  - to
- Compose key with prefix “msg::” + user provided
- Set timestamp to CurrentTimeMillis
- Set type to “msg”
- Use insert
- Try several times. See results in console
- Try same key
- Repeat with upsert

Key:
msg::some_text

```
{
   "timestamp": 1511184840248,
   "from": "luis",
   "to": "daniel",
   "type": "msg"
}
```

Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 5: Read Object

- Implement method: create(String[] words)
- Read parameter from command line:
  - Document key
- Read the document
- Write Json to STDOUT
- Test with values:
  - airline_10226
  - route_10009
  - hotel_10904

```
# read airline_10226
{"country":"United States","iata":"A1","callsign":"atifly","name":"Atifly","icao":"A1F","id":10226,"type":"airline"}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 6: Update Object

- Implement method: update(String[] words)
- Read parameter from command line:
  - Document key (prefix with “airline_” from code)
- Read the document
- Modify attribute “name”: set same value UPPERCASE
- Use replace to modify

```
# read airline_10642
{"country":"United Kingdom","iata":null,"callsign":null,"name":"Jc royal.britannica","icao":"JRB","id":10642,"type":"airline"}
# update 10642
# read airline_10642
{"country":"United Kingdom","iata":null,"callsign":null,"name":"JC ROYAL.BRITANNICA","icao":"JRB","id":10642,"type":"airline"}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

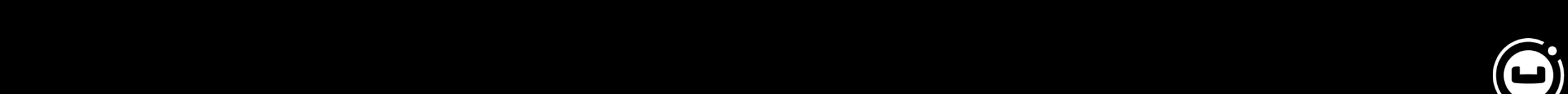
Lab 7: Delete Object

- Implement method: delete(String[] words)
- Read parameter from command line:
  - Document key (prefix with “msg::” from code)
- Delete document
- Tip: use create, then delete same key

```
# create 1001 luis ana
# read msg::1001
{"from":"luis","to":"ana","type":"msg","timestamp":1511192232720}
# delete 1001
# read msg::1001
java.lang.NullPointerException
#       at com.cbworkshop.MainLab.read(MainLab.java:95)
        at com.cbworkshop.MainLab.process(MainLab.java:60)
        at com.cbworkshop.MainLab.main(MainLab.java:30)
```

Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.



JAVA SDK
WORKING WITH PARTS OF A DOCUMENT

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

The Plan – Working with Parts of a Document

- Get Parts of a NoSQL JSON Document
- Update JSON Properties in a Document
- Batch Subdocument Operations Together

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Large Documents

```
key: nraboy
```

```
{
    "profile": {
        "firstname": "Nic",
        "lastname": "Raboy"
    },
    "data": [
        // 20MB of data
    ]
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Get Part of a Document

```
// Java
DocumentFragment<Lookup> result = bucket
    .lookupIn("nraboy")
    .get(”profile")
    .execute();
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Update Part of a Document

```
// Java
SubdocOptionsBuilder builder = new SubdocOptionsBuilder();
builder.createParents(true);
bucket().mutateIn(”nraboy").upsert(”profile.firstname", "Nicolas",
builder).execute();
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Chain Subdocument Operations

```
DocumentFragment<Mutation> result = bucket
    .mutateIn(”nraboy")
    .replace(”profile.firstname", ”Nic")
    .insert(”profile.gender", ”Male", builder)
    .remove(”data")
    .withDurability(PersistTo.MASTER, ReplicateTo.NONE)
    .execute();
```

```
DocumentFragment<Lookup> result = bucket
.lookupIn("myKey")
.get("sub.value")
.exists("fruits")
.execute();

String subValue = result.content("sub.value", String.class);
boolean fruitsExist = result.content("fruits", Boolean.class);
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 8: SubDocument API example

- Implement method: subdoc(String[] words)
- Read parameter from command line:
  - Document key (prefix with “msg::” from code)
- Using SubDocument API:
  - Change the value of the attribute from to value “administrator”
  - Add a new attribute: “reviewed”, with value CurrentTimeMillis

```
# create 1005 juan santiago
# read msg::1005
{"from":"juan","to":"santiago","type":"msg","timestamp":1511196994278}
# subdoc 1005
# read msg::1005
{"reviewed":1511197006619,"from":"Administrator","to":"santiago","type":"msg","timestamp":1511196994278}
```

Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

INTRODUCTION TO N1QL: SQL FOR JSON

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

The Plan – Querying for Documents with N1QL

- Understand the Power of N1QL
- Use Query Workbench and the Couchbase CLI
- Create Indexes for N1QL Queries
- Query for Data with N1QL
- Create Data with N1QL
- Remove Data with N1QL

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

What is N1QL?

- **Non-first (N1)** Normal Form **Query Language (QL)**
  - It is based on ANSI 92 SQL
  - Its query engine is optimized for modern, highly parallel multi-core execution
- SQL-like Query Language
  - Expressive, familiar, and feature-rich language for querying, transforming, and manipulating JSON data
- N1QL extends SQL to handle data that is:
  - **Nested:** Contains nested objects, arrays
  - **Heterogeneous:** Schema-optional, non-uniform
  - **Distributed:** Partitioned across a cluster

Power of SQL

Flexibility of JSON

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

N1QL (EXPRESSIVE)

- Access to every part of JSON document
- Scalar & Aggregate functions
- Subqueries in the FROM clause
- Aggregation on arrays

Expressive, familiar, and feature-rich language for querying, transforming, and manipulating JSON data

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

N1QL (FAMILIAR)

- SELECT * FROM bucket WHERE …
- INSERT single & multiple documents
- UPDATE any part of JSON document & use complex filter
- DELETE
- MERGE two sets of documents using traditional MERGE statement
- EXPLAIN to understand the query plan
  - EXPLAIN SELECT * FROM bucket WHERE …

Expressive, familiar, and feature-rich language for querying, transforming, and manipulating JSON data

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

N1QL (FEATURE-RICH)

- Access to every part of JSON document
- Functions (Date, Pattern, Array, Conditional, etc)
  - https://developer.couchbase.com/documentation/server/current/n1ql/n1ql-language-reference/functions.html
- JOIN, NEST, UNNEST
- Covering Index
- Prepared Statements
- USE KEYS, LIKE

Expressive, familiar, and feature-rich language for querying, transforming, and manipulating JSON data

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

N1QL – Example 1

```
SELECT   customers.id,
         customers.NAME.lastname,
         customers.NAME.firstname
         SUM(orderline.amount)
FROM     orders UNNEST orders.lineitems AS orderline
         JOIN customers ON KEYS orders.custid
WHERE customers.state = 'NY'
GROUP BY customers.id,
         customers.NAME.lastname
HAVING   SUM(orderline.amount) > 10000
ORDER BY SUM(orderline.amount) DESC
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

N1QL – Example 2

Which airlines has each airport flying from?

```
SELECT r.sourceairport, ARRAY_AGG(DISTINCT a.name) as airlines
FROM `travel-sample` r
JOIN `travel-sample` a ON KEYS r.airlineid
WHERE r.type="route"
GROUP BY r.sourceairport
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Query Workbench and the Couchbase CLI

- Query Workbench
  - http://localhost:8091/ui/index.html#!/query/workbench
- Couchbase CLI (Mac)
  - /Applications/Couchbase Server.app/Contents/Resources/couchbase-core/bin/cbq
- Couchbase CLI (Windows)
  - C:\Program Files\CouchbaseServer\bin\cbq.exe
- Couchbase CLI (Linux)
  - /opt/couchbase/bin/cbq

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

CRUD IN N1QL

C

```
INSERT INTO `default`
(KEY, VALUE)
VALUES (“person-1”, {
    “firstname”: “Nic”,
    “lastname”: “Raboy”
})
```

R

```
SELECT name FROM `travel-sample` WHERE type="airline" AND icao="JRB"
```

U

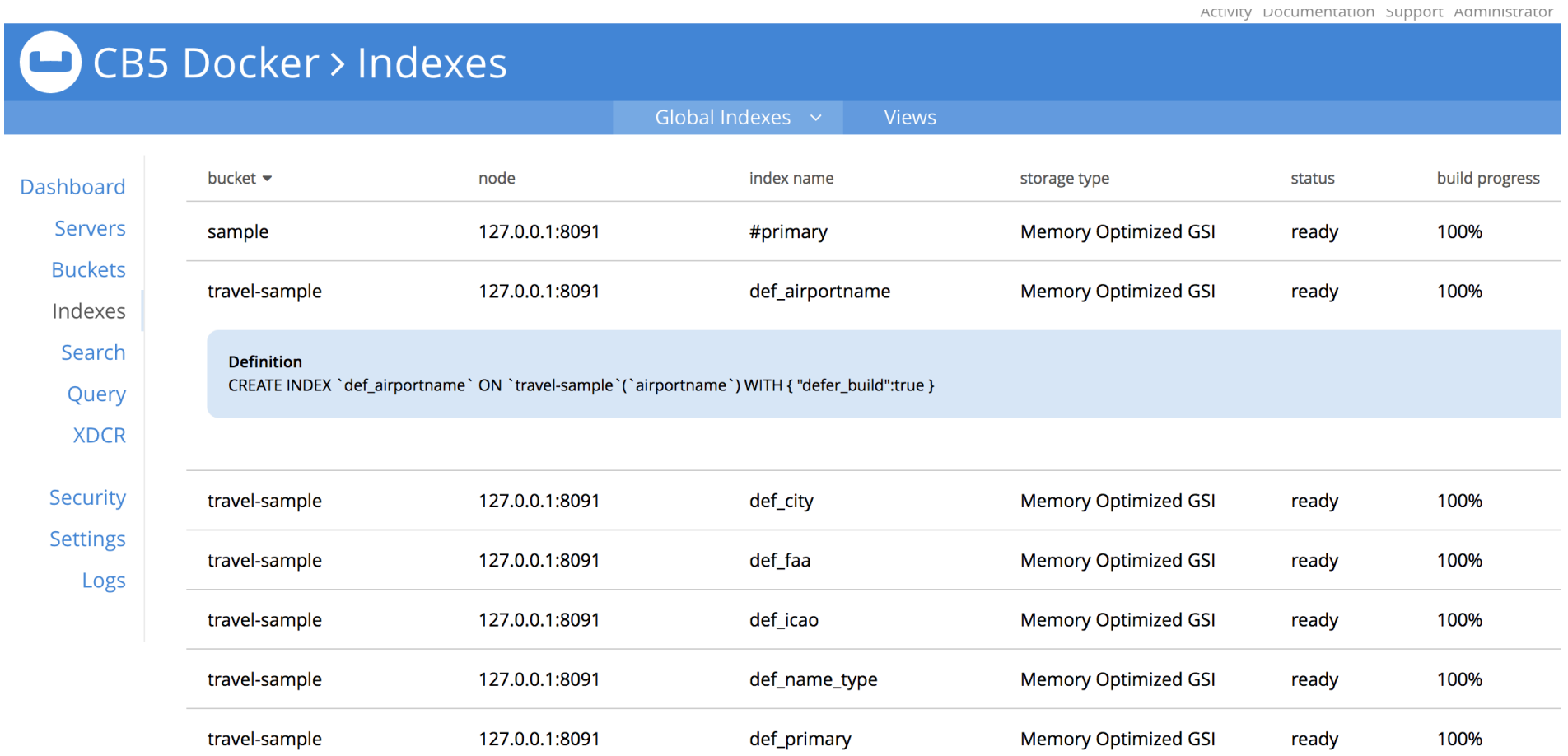
```
UPDATE `travel-sample` USE KEYS ["hotel_10138"]
SET pets_ok=true, free_internet=true
```

D

```
DELETE FROM `default`
WHERE META().id = ‘person-1’
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

GSI Indexes



Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Primary Index

- The primary index is the index on the document key on every document in a keyspace
- The primary index is used for:
  - Complete keyspace scan – equivalent of table scan
  - Query does not have any predicates (filters)
  - Query has predicate on document key
  - No other index or access path can be used
- Primary index is optional

```
CREATE PRIMARY INDEX ON `travel-sample`;


SELECT *
FROM `travel-sample`;

SELECT *
FROM `travel-sample`
WHERE META().id LIKE "user::%”;
```

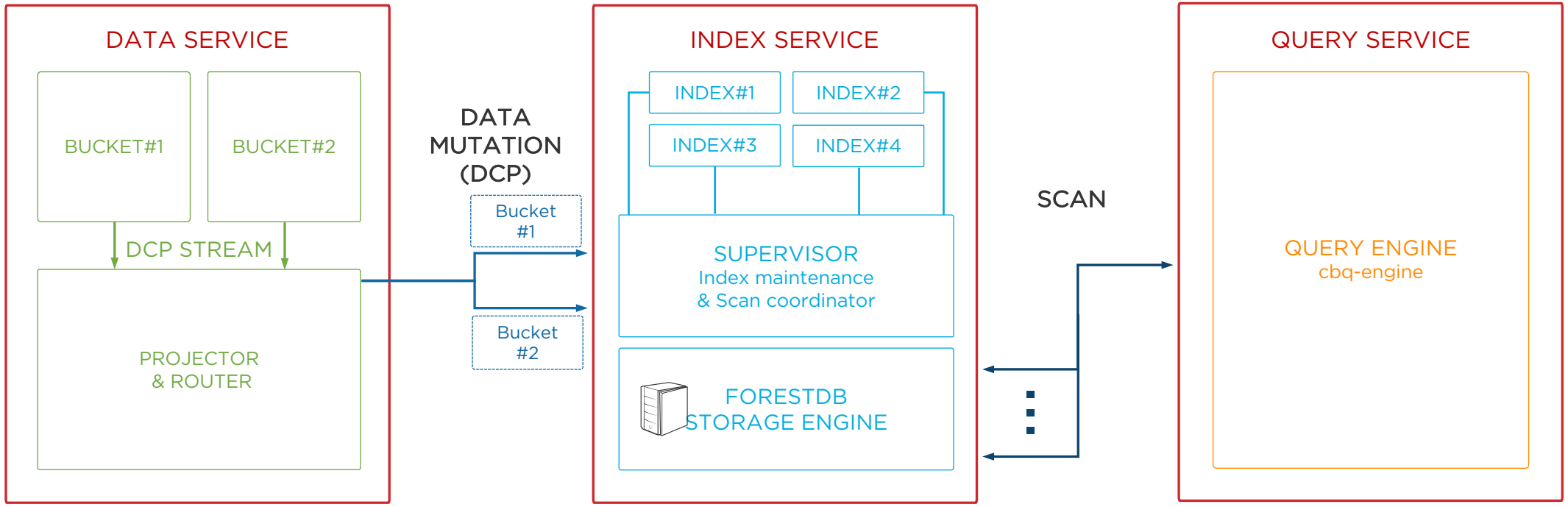
Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Secondary Index

- Simple Index
- Composite Index
- Functional Index
- Partial Index
- Replica Index
- Covering Index

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Couchbase Services



Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

JAVA SDK
N1QL QUERIES

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Query Raw String vs. DSL

Raw string query

```
N1qlQueryResult queryResult =
bucket.query(N1qlQuery.simple("SELECT * FROM `travel-sample` LIMIT 10"));
```

Using the DSL

```
import static com.couchbase.client.java.query.Select.select;
import static com.couchbase.client.java.query.dsl.Expression.*;
...
N1qlQueryResult query =
bucket.query(select("*").from("`travel-sample`").limit(10));
```

Iterate result

```
for(N1qlQueryRow row : queryResult){
      System.out.println(row.value().toString());
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 9: Simple Query

- Implement method: query (String[] words)
- Execute the query: “`SELECT * FROM `travel-sample` LIMIT 10`”
- Print the results to STDOUT
- Use both implementations, raw and dsl

```
# query
{"travel-sample":{"country":"United States","iata":"Q5","callsign":"MILE-AIR","name":"40-Mile Air","icao":"MLA","id":10,"type":"airline"}}
{"travel-sample":{"country":"United States","iata":"TQ","callsign":"TXW","name":"Texas Wings","icao":"TXW","id":10123,"type":"airline"}}
{"travel-sample":{"country":"United States","iata":"A1","callsign":"atifly","name":"Atifly","icao":"A1F","id":10226,"type":"airline"}}
{"travel-sample":{"country":"United Kingdom","iata":null,"callsign":null,"name":"JC ROYAL.BRITANNICA","icao":"JRB","id":10642,"type":"airline"}}
{"travel-sample":{"country":"United States","iata":"ZQ","callsign":"LOCAIR","name":"Locair","icao":"LOC","id":10748,"type":"airline"}}
{"travel-sample":{"country":"United States","iata":"K5","callsign":"SASQUATCH","name":"SeaPort Airlines","icao":"SQH","id":10765,"type":"airline"}}
{"travel-sample":{"country":"United States","iata":"KO","callsign":"ACE AIR","name":"Alaska Central Express","icao":"AER","id":109,"type":"airline"}}
{"travel-sample":{"country":"United Kingdom","iata":"5W","callsign":"FLYSTAR","name":"Astraeus","icao":"AEU","id":112,"type":"airline"}}
{"travel-sample":{"country":"France","iata":"UU","callsign":"REUNION","name":"Air Austral","icao":"REU","id":1191,"type":"airline"}}
{"travel-sample":{"country":"France","iata":"A5","callsign":"AIRLINAIR","name":"Airlinair","icao":"RLA","id":1203,"type":"airline"}}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Query with parameters

Raw string query

```
N1qlQueryResult queryResult =
bucket.query(N1qlQuery.simple("SELECT * FROM `travel-sample` LIMIT 10"));
```

Using the DSL

```
import static com.couchbase.client.java.query.Select.select;
import static com.couchbase.client.java.query.dsl.Expression.*;
...
N1qlQueryResult query =
bucket.query(select("*").from("`travel-sample`").limit(10));
```

Iterate result

```
for(N1qlQueryRow row : queryResult){
      System.out.println(row.value().toString());
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 10: Query with parameters

- Implement method: queryAirports(String[] words)
- Read parameters from command line:
  - sourceairport
  - destinationairport
- Write a query to find airlines (airline names) flying from sourceairport to destinationairport. Use JOIN
- Use a parametrized query
- TIP: Highest traffic airport codes: ATL, ORD, LHR, CDG, LAX, DFW, JFK

```
# queryairports JFK LHR
{"name":"British Airways"}
{"name":"Delta Air Lines"}
{"name":"American Airlines"}
{"name":"US Airways"}
{"name":"Virgin Atlantic Airways"}
{"name":"Air France"}
```

Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Query Consistency

- not_bounded
  - Fastest
  - Returns data that is currently indexed and accessible by the index or the view.
- at_plus
  - Fast
  - A query submitted with at_plus consistency level requires all mutations, up to the moment of the scan_vector (the logical timestamp passed in with at_plus), to be processed before the query execution can start.
- request_plus
  - Slowest, but still fast
  - Requires all mutations, up to the moment of the query request, to be processed before the query execution can start.

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Query Consistency

```
// Java
N1qlQueryResult result =
  bucket.query(
    N1qlQuery.simple(
      statement,
      N1qlParams.build().consistency(ScanConsistency.REQUEST_PLUS)
    )
  );
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

JAVA SDK
BATCHING WITH RXJAVA

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Batching with RxJava

- Implicit batching is performed by utilizing a few operators:
  - **Observable.just()** or **Observable.from()** to generate an Observable that contains the data you want to batch on.
- **flatMap()** to send those events against the Couchbase Java SDK and merge the results asynchronously.
- **last()** if you want to wait until the last element of the batch is received.
- **toList()** if you care about the responses and want to aggregate them easily.
- If you have more than one subscriber, using **cache()** to prevent accessing the network over and over again with every subscribe.

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Batching with RxJava

- The following example creates an observable stream of 5 keys to load in a batch,
- asynchronously fires off get() requests against the SDK,
- waits until the last result has arrived,
- and then converts the result into a list and blocks at the very end

```
Cluster cluster = CouchbaseCluster.create();
Bucket bucket = cluster.openBucket();

List<JsonDocument> foundDocs = Observable
    .just("key1", "key2", "key3", "key4", "key5")
    .flatMap(new Func1<String, Observable<JsonDocument>>() {
        @Override
        public Observable<JsonDocument> call(String id) {
            return bucket.async().get(id);
        }
    })
    .toList()
    .toBlocking()
    .single();
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Batching with RxJava

- If you wrap the code in a helper method, you can provide very nice encapsulated batching semantics

```
public List<JsonDocument> bulkGet(final Collection<String> ids) {
    return Observable
        .from(ids)
        .flatMap(new Func1<String, Observable<JsonDocument>>() {
            @Override
            public Observable<JsonDocument> call(String id) {
                return bucket.async().get(id);
            }
        })
        .toList()
        .toBlocking()
        .single();
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Batching with RxJava

- Example of batching mutations

```
// Generate a number of dummy JSON documents
int docsToCreate = 100;
List<JsonDocument> documents = new ArrayList<JsonDocument>();
for (int i = 0; i < docsToCreate; i++) {
    JsonObject content = JsonObject.create()
        .put("counter", i)
        .put("name", "Foo Bar");
    documents.add(JsonDocument.create("doc-"+i, content));
}

// Insert them in one batch, waiting until the last one is done.
Observable
    .from(documents)
    .flatMap(new Func1<JsonDocument, Observable<JsonDocument>>() {
        @Override
        public Observable<JsonDocument> call(final JsonDocument docToInsert) {
            return bucket.async().insert(docToInsert);
        }
    })
    .last()
    .toBlocking()
    .single();
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Batching with RxJava - Batching mutations

- The following code generates a number of fake documents and inserts them in one batch.
- Note that you can decide to either collect the results with **toList()** as shown before or just use **last()** as shown here to wait until the last document is properly inserted.

```
// Generate a number of dummy JSON documents
int docsToCreate = 100;
List<JsonDocument> documents = new ArrayList<JsonDocument>();
for (int i = 0; i < docsToCreate; i++) {
    JsonObject content = JsonObject.create()
        .put("counter", i)
        .put("name", "Foo Bar");
    documents.add(JsonDocument.create("doc-"+i, content));
}

// Insert them in one batch, waiting until the last one is done.
Observable
    .from(documents)
    .flatMap(new Func1<JsonDocument, Observable<JsonDocument>>() {
        @Override
        public Observable<JsonDocument> call(final JsonDocument docToInsert) {
            return bucket.async().insert(docToInsert);
        }
    })
    .last()
    .toBlocking()
    .single();
```

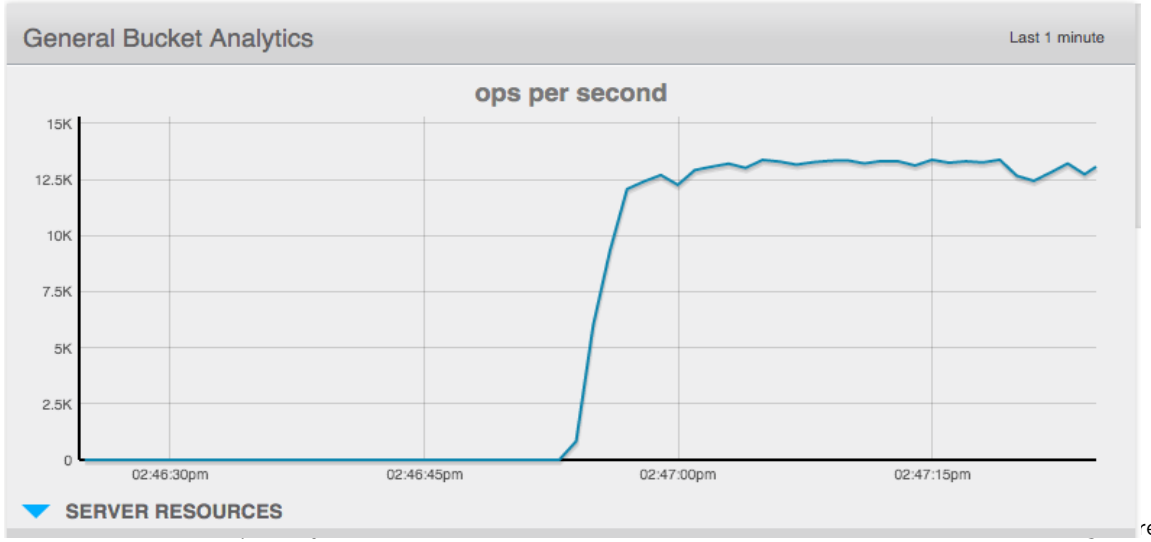
Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

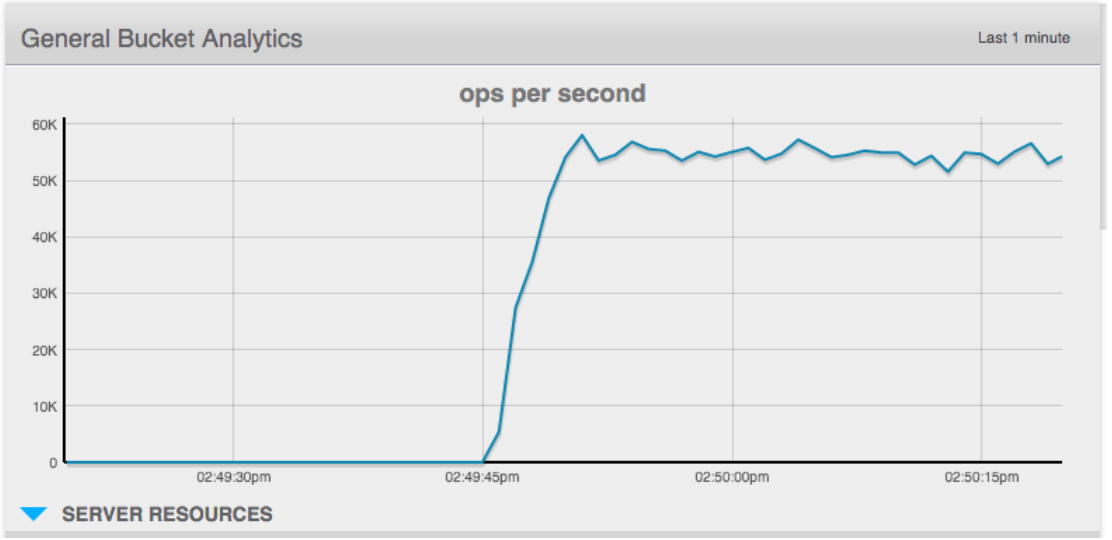
Sync Vs. ASync

- ✓ Here are two code samples, both synchronous, that showcase serialized and batched loading of documents.

```
// Serialized workload of loading documents
while(true) {
    List<JsonDocument> loaded = new ArrayList<JsonDocument>();
    int docsToLoad = 10;
    for (int i = 0; i < docsToLoad; i++) {
        JsonDocument doc = bucket.get("doc-" + i);
        if (doc != null) {
            loaded.add(doc);
        }
    }
}
```

```
// Same workload, utilizing batching effects
while(true) {
    int docsToLoad = 10;
    Observable
        .range(0, docsToLoad)
        .flatMap(new Func1<Integer, Observable<JsonDocument>>() {
            @Override
            public Observable<JsonDocument> call(Integer i) {
                return bucket.async().get("doc-"+i);
            }
        })
        .toList()
        .toBlocking()
        .single();
}
```





Bulk Read Async

```
List<String> priceKeys;
List<JsonObject> res = Observable
    .from(priceKeys)
    .flatMap(k -> bucket.async().get(k))
    .map(doc -> doc.content())
    .toList()
    .toBlocking()
    .single();
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Bulk Write Async

Async version:

```
List<JsonDocument> docs = ...;
Observable
    .from(docs)
    .flatMap(doc -> bucket.async().insert(doc))
    .last()
    .toBlocking()
    .single();
```

Sync version:

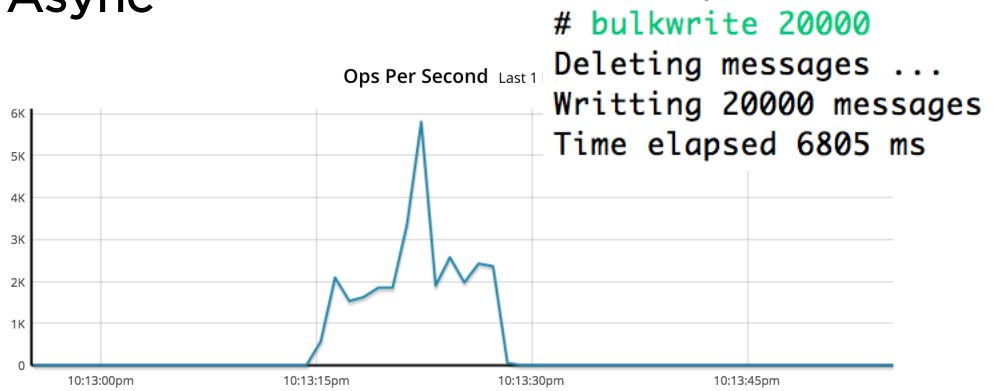
```
for(JsonDocument doc : docs){
   bucket.insert(doc);
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

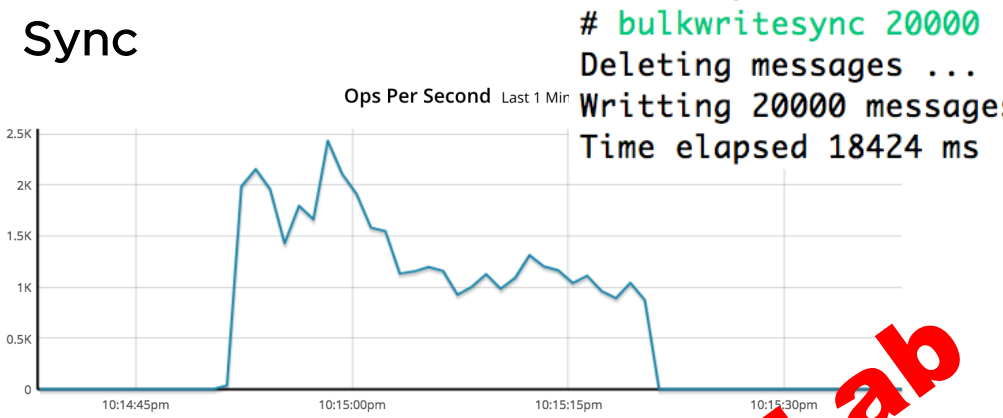
Lab 11: Bulk Write Performance

- Implement method: bulkWrite (String[] words) : Async version
- Implement method: bulkWriteSync (String[] words) : Sync version
- Read parameters from command line:
  - size: Number of messages to insert. Keys will be from msg::1 to msg::[size]
- Delete all messaes in the bucket: DELETE FROM `travel-sample` WHERE type="msg"
- Create an array of JsonDocuments of messages
- Insert the messages into the bucket (in async / sync way)
- Print the time elapsed to STDOUT
- Compare results sync vs. async. Check both time and operations per second in the console

Async



Sync



Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Query Async mode

```
bucket.async()
    .query(select("*").from(i("`travel-sample`")).limit(5))
    .flatMap(result ->
        result.errors()
        .flatMap(e ->
                Observable.<AsyncN1qlQueryRow>error(new CouchbaseException("N1QL
Error/Warning: " + e)))
                .switchIfEmpty(result.rows())
        )
    .map(AsyncN1qlQueryRow::value)
    .subscribe(
        rowContent -> System.out.println(rowContent),
        runtimeError -> runtimeError.printStackTrace()
    );
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 12: Simple Query – Async version

- Implement method: queryAsync (String[] words)
- Execute the query: "`SELECT * FROM \`travel-sample\` LIMIT 5`"
- Print the results to STDOUT
- Use asynchronous implementation

```
# queryasync
# {"travel-sample":{"country":"United States","iata":"Q5","callsign":"MILE-AIR","name":"40-Mile Air","icao":"MLA","id":10,"type":"airline"}}
{"travel-sample":{"country":"United States","iata":"TQ","callsign":"TXW","name":"Texas Wings","icao":"TXW","id":10123,"type":"airline"}}
{"travel-sample":{"country":"United States","iata":"A1","callsign":"atifly","name":"Atifly","icao":"A1F","id":10226,"type":"airline"}}
{"travel-sample":{"country":"United Kingdom","iata":null,"callsign":null,"name":"JC ROYAL.BRITANNICA","icao":"JRB","id":10642,"type":"airline"}}
{"travel-sample":{"country":"United States","iata":"ZQ","callsign":"LOCAIR","name":"Locair","icao":"LOC","id":10748,"type":"airline"}}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

JAVA SDK
FULL TEXT SEARCH

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

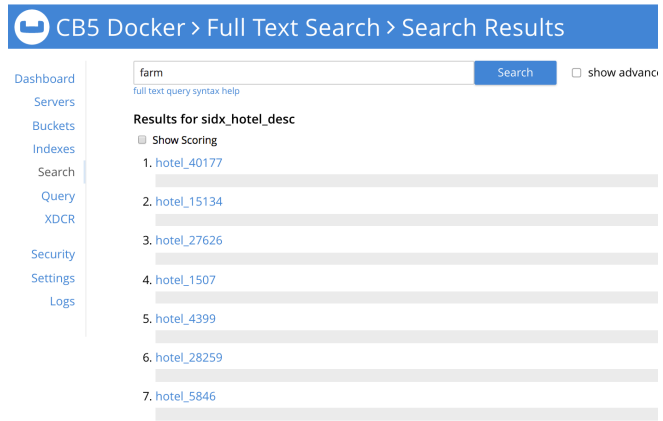
Full Text Search vs N1QL

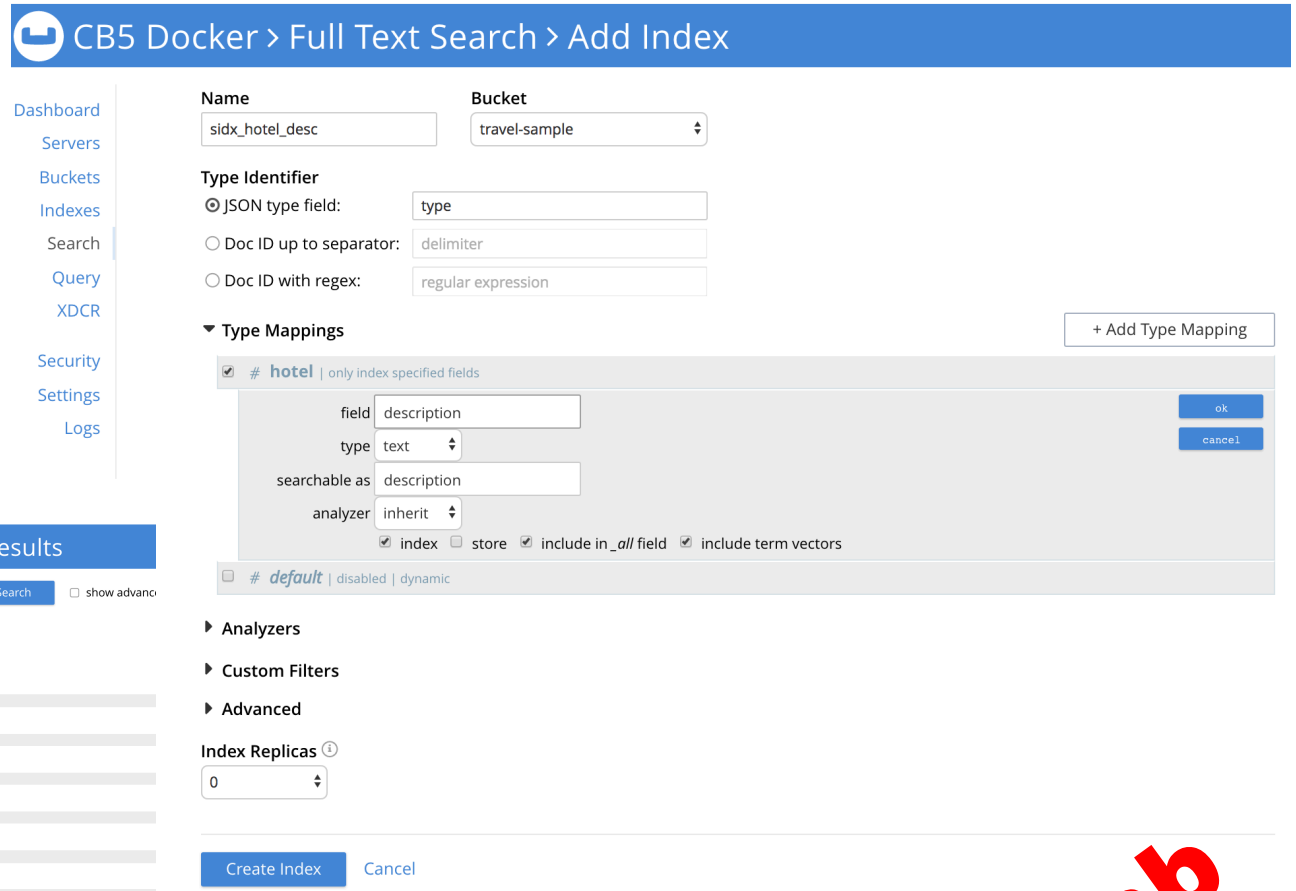
- Wildcard Queries are Slow
- What happens to variations of the wildcard term in N1QL / SQL?
  - '%golang%', does 'Golang' or 'GoLang' pop up?
  - '%golang%', do spelling errors like 'goang' pop up?
- FTS is for Natural Language Querying

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 13: FTS – Create index

- Build an FTS Search index on travel-sample for hotels (type="hotel")
- Index only the attribute "description"
- Index name: sidx_hotel_desc
- Bucket: travel-sample
- Type Mappings: uncheck default
- Add type mapping: description
- Check "only index specified fields"
- Insert child field: description
- Search by term: farm





Lab

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Full Text Search: Java example

```
String term = ... ;
MatchQuery fts = SearchQuery.match(term);
SearchQueryResult result = bucket.query(new SearchQuery("sidx_hotel_desc", fts));
for (SearchQueryRow row : result) {
      System.out.println(row);
}
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Lab 14: FTS – Java example

- Implement method: search(String[] words) : Async version
- Read parameters from command line:
  - term: String to look for in the index
- Write code to search using the index `sidx_hotel_desc`
- Print results to STDOUT

```
# search pool
DefaultSearchQueryRow{index='sidx_hotel_desc_13e37e3ca69ee59e_18572d87', id='hotel_20420', score=5.412489826449426, explanation={}, locations=DefaultHitLocations{siz
DefaultSearchQueryRow{index='sidx_hotel_desc_13e37e3ca69ee59e_18572d87', id='hotel_40383', score=3.8272084673560847, explanation={}, locations=DefaultHitLocations{si
DefaultSearchQueryRow{index='sidx_hotel_desc_13e37e3ca69ee59e_18572d87', id='hotel_4396', score=3.8272084673560847, explanation={}, locations=DefaultHitLocations{siz
DefaultSearchQueryRow{index='sidx_hotel_desc_13e37e3ca69ee59e_13aa53f3', id='hotel_3785', score=3.209767754362897, explanation={}, locations=DefaultHitLocations{size
DefaultSearchQueryRow{index='sidx_hotel_desc_13e37e3ca69ee59e_6ddbfb54', id='hotel_25303', score=3.1616432709856555, explanation={}, locations=DefaultHitLocations{si
DefaultSearchQueryRow{index='sidx_hotel_desc_13e37e3ca69ee59e_f4e0a48a', id='hotel_3786', score=3.124082787542814, explanation={}, locations=DefaultHitLocations{size
DefaultSearchQueryRow{index='sidx_hotel_desc_13e37e3ca69ee59e_6ddbfb54', id='hotel_20423', score=2.738063243874787, explanation={}, locations=DefaultHitLocations{siz
DefaultSearchQueryRow{index='sidx_hotel_desc_13e37e3ca69ee59e_18572d87', id='hotel_3714', score=2.6000760548180764, explanation={}, locations=DefaultHitLocations{siz
DefaultSearchQueryRow{index='sidx_hotel_desc_13e37e3ca69ee59e_54820232', id='hotel_16631', score=2.440399959929308, explanation={}, locations=DefaultHitLocations{siz
DefaultSearchQueryRow{index='sidx_hotel_desc_13e37e3ca69ee59e_f4e0a48a', id='hotel_25671', score=2.386057638178849, explanation={}, locations=DefaultHitLocations{siz
```

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Types of Full Text Search Queries

- Match Query
  - A match query analyzes the input text and uses that analyzed text to query the index. An attempt is made to use the same analyzer that was used when the field was indexed.
- Match Phrase Query
- Fuzzy Query
- Wildcard Query
- Conjunction Query
- Range Queries
- https://developer.couchbase.com/documentation/server/current/fts/fts-query-types.html

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

JAVA SDK
SPRING BOOT

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Travel-sample demo on couchbaselabs

- This application uses Couchbase Server 5.0 together with Spring Boot, Angular2 and Bootstrap.
- Documentation: https://github.com/couchbaselabs/try-cb-java/tree/5.0-updates
- From command line execute:

```
git clone -b 5.0-updates https://github.com/couchbaselabs/try-cb-java.git
```

- To run the app execute:

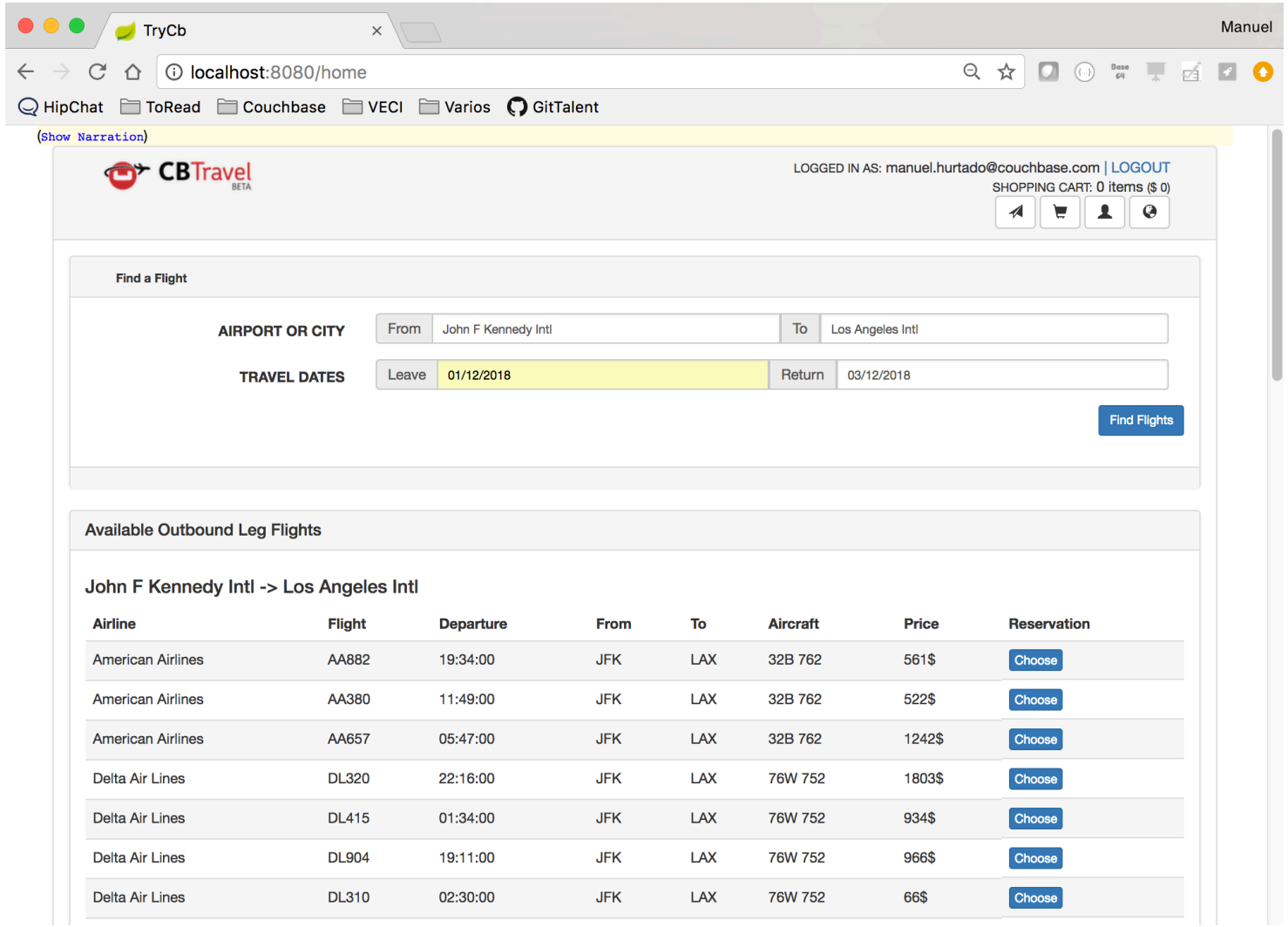
```
cd try-cb-java/
```

```
mvn spring-boot:run -Dstorage.host=127.0.0.1 -Dstorage.bucket=travel-sample -
Dstorage.password=password -Dstorage.username=travel
```

- Access application: http://localhost:8080
- Details: https://blog.couchbase.com/traveling-with-couchbase-using-the-java-sdk/

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Travel-sample demo on couchbaselabs



Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

THANK YOU

Confidential and Proprietary. Do not distribute without Couchbase consent. © C