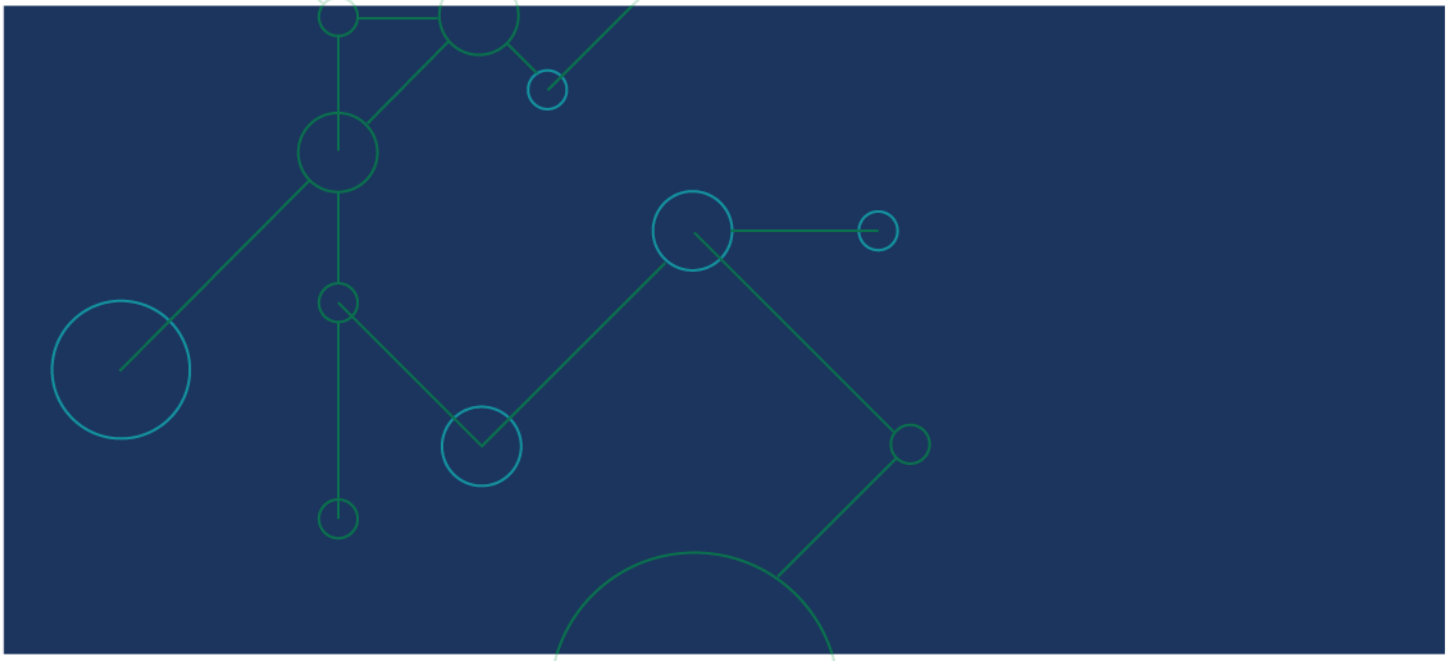
WHITE PAPER

Connecting to Cosmos DB Mongo API from Qlik Sense using the Qlik MongoDB Connector

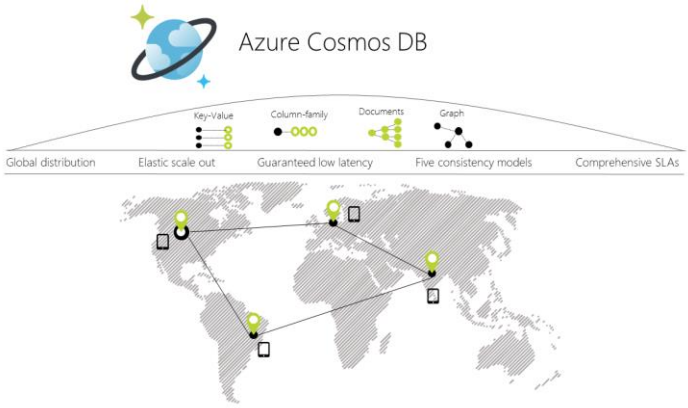
Qlik Partner Engineering: October 2018

David Freriks: Technology Evangelist



COSMOS DB OVERVIEW

Cosmos DB is a fast growing, multi-modal database service in Microsoft Azure offering several API's. Currently, the SQL API is the most popular and widely used API. When you create a Cosmos DB account, you must decide which API you want to use as the data will get stored in corresponding data model/format.



COSMOS DB KEY CAPABILITIES

Turnkey global distribution

- You can distribute your data to any number of Azure regions, with the click of a button. This enables you to put your data where your users are, ensuring the lowest possible latency to your customers.
- Using Azure Cosmos DB's multi-homing APIs, the app always knows where the nearest region is and sends requests to the nearest data center. All of this is possible with no config changes. You set your write-region and as many read-regions as you want, and the rest is handled for you.
- As you add and remove regions to your Azure Cosmos DB database, your application does not need to be redeployed and continues to be highly available thanks to the multi-homing API capability.

Multiple data models and popular APIs for accessing and querying data

The underlying atom-record-sequence (ARS) based data model that Azure Cosmos DB is built on natively supports multiple data models, including but not limited to document, graph, key-value, table, and column-family data models.

APIs for the following data models are supported with SDKs available in multiple languages:

- SQL API: A schema-less JSON database engine with rich SQL querying capabilities.

- MongoDB API: A massively scalable *MongoDB-as-a-Service* powered by Azure Cosmos DB platform. Compatible with existing MongoDB libraries, drivers, tools, and applications.
- Cassandra API: A globally distributed Cassandra-as-a-Service powered by Azure Cosmos DB platform. Compatible with existing Apache Cassandra libraries, drivers, tools, and applications.
- Gremlin API: A fully managed, horizontally scalable graph database service that makes it easy to build and run applications that work with highly connected datasets supporting Open Gremlin APIs (based on the Apache TinkerPop specification, Apache Gremlin).
- Table API: A key-value database service built to provide premium capabilities (for example, automatic indexing, guaranteed low latency, global distribution) to existing Azure Table storage applications without making any app changes.

The SQL API can be interacted with using ODBC, REST, or native code bases such as .Net (Core and Standard), Java, Go, Node.js, or Python.

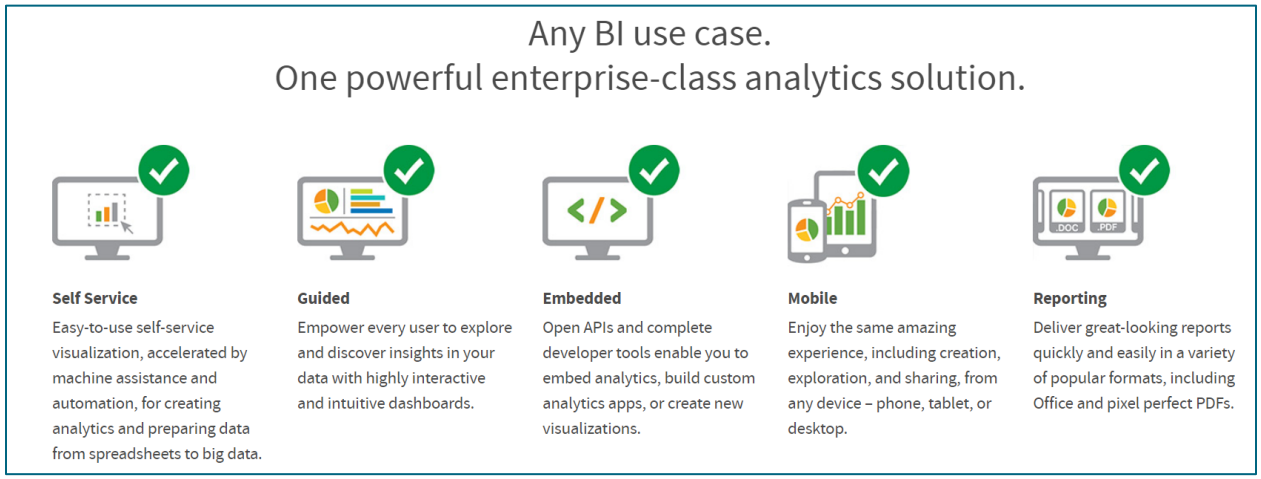
There are many connectivity methods validated with Qlik Partner Engineering:

- SQL API using ODBC Connector in Qlik Sense
- MongoDB API using the Qlik Sense MongoDB Connector (Beta currently, Oct 2018)
- MongoDB SQL API using REST Connector in Qlik Sense
- Mongo DB API using the gRPC connector for Qlik Core

The focus of this document is the details of connecting to the Qlik Sense MongoDB Connector.

ABOUT QLIK SENSE

Qlik Sense gives you data superpowers. Easily combine all your data sources, no matter how large, into a single view. Qlik's Associative engine indexes every possible relationship in your data so you can gain immediate insights and explore in any direction your intuition takes you. Unlike query-based tools, there's no pre-aggregated data and predefined queries to hold you back. That means you can ask new questions and create analytics without having to build new queries or wait for the experts.



Interactive analysis, without boundaries

Ask any question and quickly explore across all your data for insight, using global search and interactive selections. All analytics update instantly with each click, no matter how deep you go, furthering analysis or pivoting your thinking in new directions. There's is no limit to exploration and no data left behind.

Simply smarter visualizations

Innovative visualizations put your data in the right context to answer any question. Explore the shape of data and pinpoint outliers. Use advanced analytics integration and geographic calculation to broaden insight. And it's fully interactive - easily pan, zoom, and make selections to find insights visually.

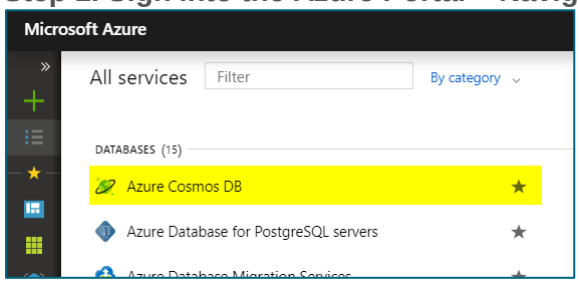
Create and explore on any device

Explore, create, and collaborate on any device, directly at the point of decision. Qlik Sense is built from the ground up with responsive mobile design and touch interaction. Build analytics apps once, and they'll work everywhere, on desktop, tablet, or mobile devices. To learn more, go to https://www.qlik.com/us/products/qlik-sense.

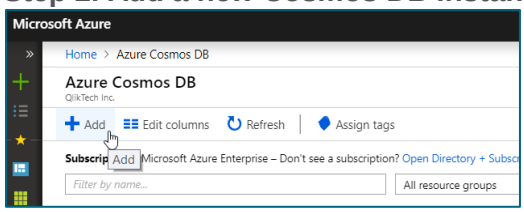
COSMOS DB SETUP

Setting up Cosmos DB is straightforward, once you pick your path. For this validation exercise, we are testing the MongoDB API as our query engine for Qlik.

Step 1. Sign into the Azure Portal – Navigate to Cosmos DB

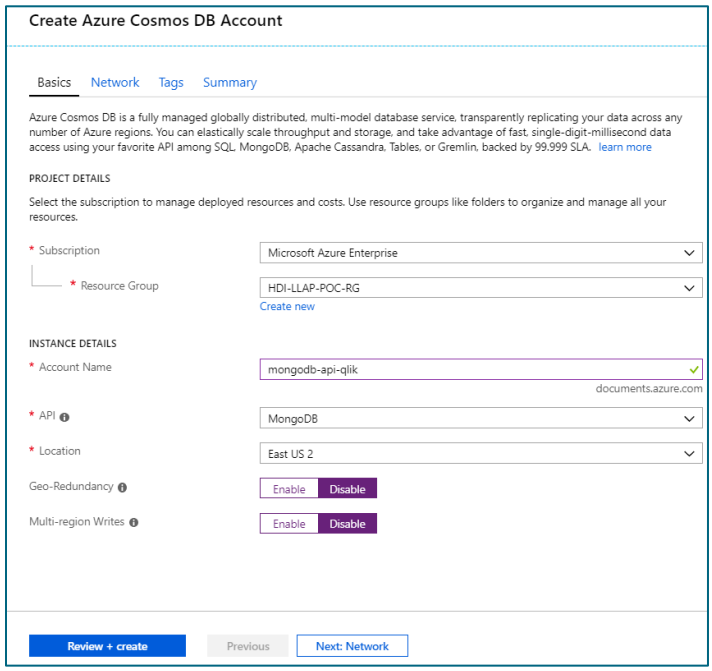


Step 2. Add a new Cosmos DB instance



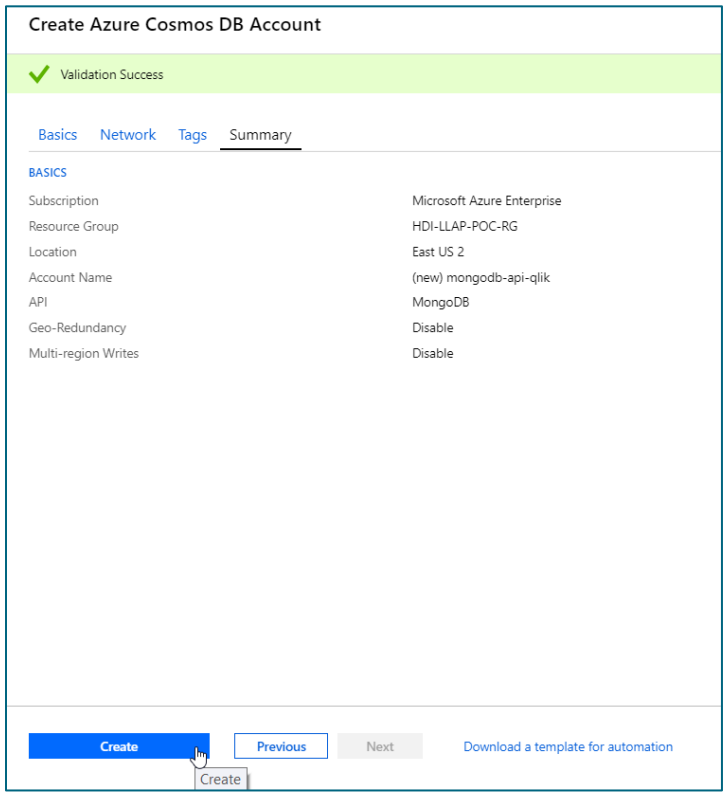
Step 3. Complete Project Details

For this use case, we are selecting the MongoDB API.



Step 4. Validate and Create!

The process will run for a few minutes and we have our database!

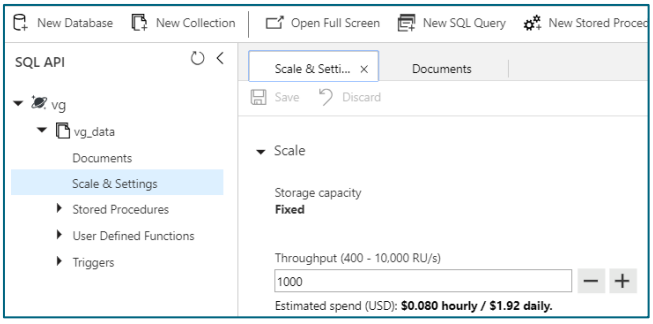


Conversation - What is a RU?

Cosmos DB is priced and scaled using a model called a “Request Unit” or RU. The RU defines how much throughput (and corresponding hardware resources needed to provide it) you can use with the Cosmos DB instance. There are two modes, **Fixed** or **Unlimited** which govern the capacity of the document collection during use. You can learn more about how it works here. This setting will govern the insert, update and query capacity (per second) of the Cosmos DB collection, so consideration must be given to expected workloads during the setup process as certain element cannot be changed after creation.

Step 5. Create Database and Collection

Before we can load data, we need to create the repository inside Cosmos DB that we’ll need to write data into. We start by opening Data Explorer in the menu. (Side note, with the utility we are using to upload data, the collection does not have to be precreated, but if you want to specify RU,



you will have to precreate.) We need to create a New Database and a New Collection underneath that database. Here is where you will set your RU capacity. For this data set and test, we'll go with 4000 RU.

Step 6. Loading Data

There are many ways to load data into Cosmos DB, we will follow the procedure detailed on the Azure help site Cosmos DB Help. The data set we will use as a test data set is a json collection of documents from Twitter and can be found Tweets.zip .

Once we have downloaded both the MongoDB community server and installed, as well as the data, we are ready to proceed. Unzip the tweets.zip into a directory and take note of the directory path.

MongoDB – mongorestore.exe

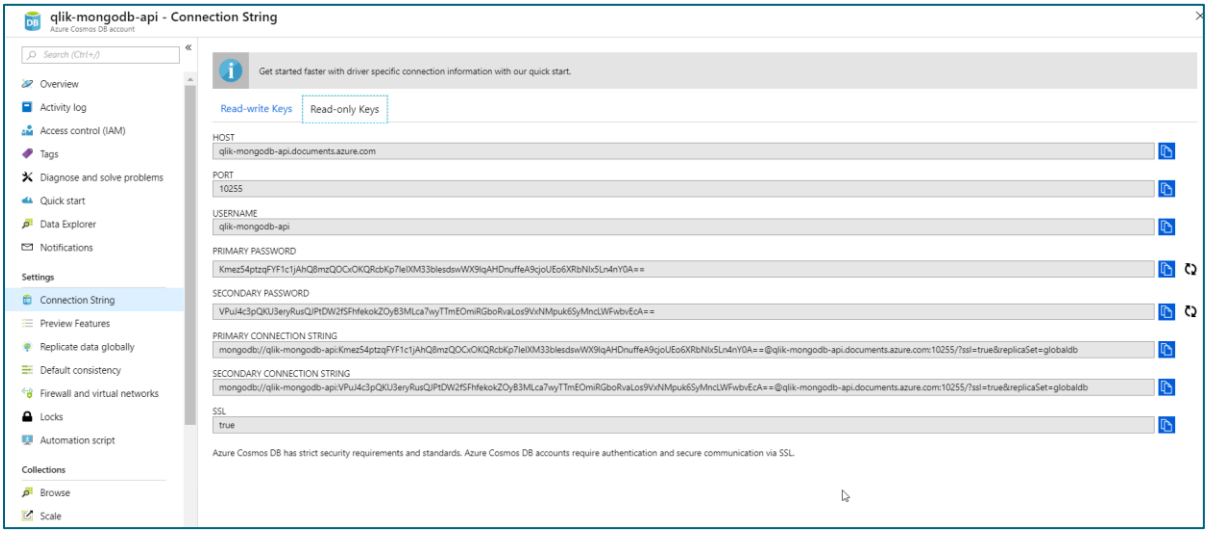
This process is command line based, so open a *command prompt* window. 

Change director to the install folder of where MongoDB server installed (C:\Program Files\MongoDB\Server\4.0\bin). From here we will create a command string to upload the tweets to Cosmos DB.

The format of the string is:

mongorestore.exe --host <your_hostname>:10255 -u <your_username> -p <your_password> --db <your_database> --collection <your_collection> --ssl --sslAllowInvalidCertificates <path_to_backup>

These all can be found on the Connection String section under Settings of the Cosmos DB account.



We need the following:

HOST, PORT, USERNAME, and PRIMARY PASSWORD which Azure has been kind enough to allow copying directly with the  option.

Taking those elements and with a trusty copy/paste into notepad, we can build our import string. For example:

mongorestore.exe --host qlik-mongodb-api.documents.azure.com:10255 -u qlik-mongodb-api -p DLHyL5G0Yzocjyzjy1yytntDXGryrSE4kHYE1vbkTaBWFnKZDmAOiCT49Zi9Sae0yfjXcZQqPdgnumpM9JigxA== --db twitter --collection tweets --ssl --sslAllowInvalidCertificates .\dump\twitter\tweets.bson

Sizing / Capacity

We're not quite ready to execute the command yet. Earlier in our setup we capped our RU usage at 4000, and because of this we need to throttle our import stream to prevent upload errors. We do this by adding modifiers *--numInsertionWorkersPerCollection 1 --batchSize 4*. These commands are discussed in detail on the help site reference earlier to understand how to calculate RU for a use case such as this…

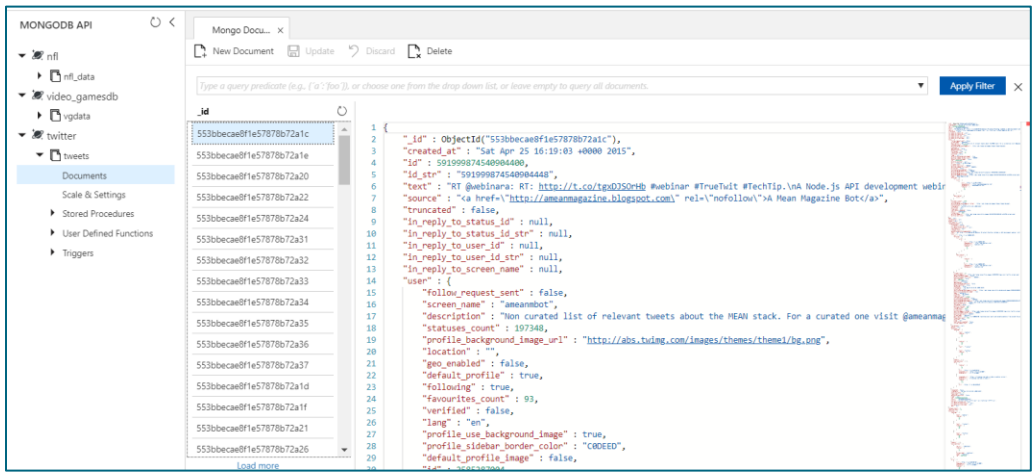
Final Command

mongorestore.exe --host qlik-mongodb-api.documents.azure.com:10255 -u qlik-mongodb-api -p DLHyL5G0Yzocjyzjy1yytntDXGryrSE4kHYE1vbkTaBWFnKZDmAOiCT49Zi9Sae0yfjXcZQqPdgnumpM9JigxA== --db twitter --collection tweets --ssl --sslAllowInvalidCertificates .\dump\twitter\tweets.bson --numInsertionWorkersPerCollection 1 --batchSize 4

We can execute this command, and after a few seconds we will have uploaded into our collection.

```
2018-10-11T10:13:08.321-0700    checking for collection data in dump\twitter\tweets.bson
2018-10-11T10:13:08.422-0700    reading metadata for twitter.tweets from dump\twitter\tweets.metadata.json
2018-10-11T10:13:08.424-0700    restoring twitter.tweets from dump\twitter\tweets.bson
2018-10-11T10:13:09.848-0700    [###.....................]  twitter.tweets  502KB/3.81MB  (12.9%)
2018-10-11T10:13:12.848-0700    [##########..............]  twitter.tweets  1.65MB/3.81MB  (43.4%)
2018-10-11T10:13:15.848-0700    [################........]  twitter.tweets  2.62MB/3.81MB  (68.8%)
2018-10-11T10:13:18.848-0700    [######################..]  twitter.tweets  3.58MB/3.81MB  (94.0%)
2018-10-11T10:13:19.985-0700    [########################]  twitter.tweets  3.81MB/3.81MB  (100.0%)
2018-10-11T10:13:19.985-0700    restoring indexes for collection twitter.tweets from metadata
```

We can look in our Data Explorer and find that indeed the data has loaded!



We have now completed our Cosmos DB setup.

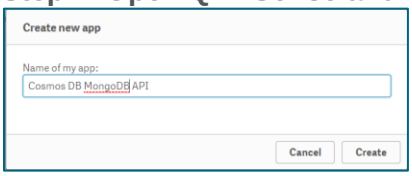
QLIK SENSE CONFIGURATION

Install & Configure Qlik Sense

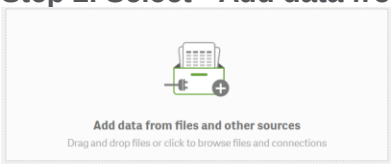
This is not covered in this guide, as we pre-assume a running Qlik Sense system. If you need to setup Qlik Sense – please refer to this guide (Install Qlik Sense on Azure) or download Qlik Sense desktop (Qlik Sense Desktop).

Creating the Qlik Sense App

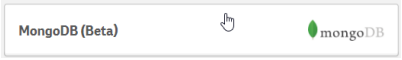
Step 1. Open Qlik Sense and create a new App



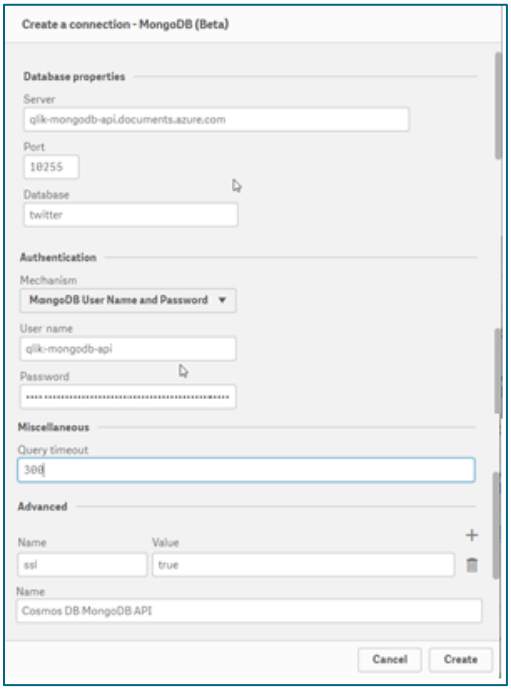
Step 2. Select - Add data from Files and other sources



Step 3. Select MongoDB (beta)



Step 4. Fill in the information we captured earlier…and Create

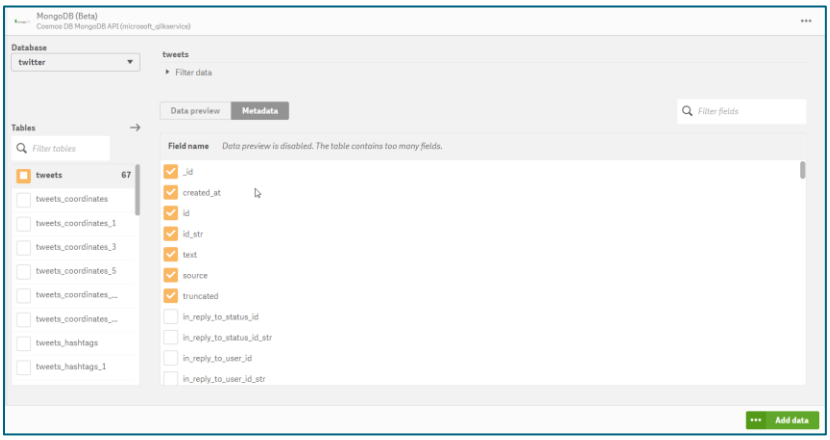


Note: Up the timeout setting from 30 to 300, the MongoDB driver creates the schema dynamically upon accessing the data – which can sometime exceed the timeout on a complex collection.

Step 5. Select a dataset (tweets) and Add Data

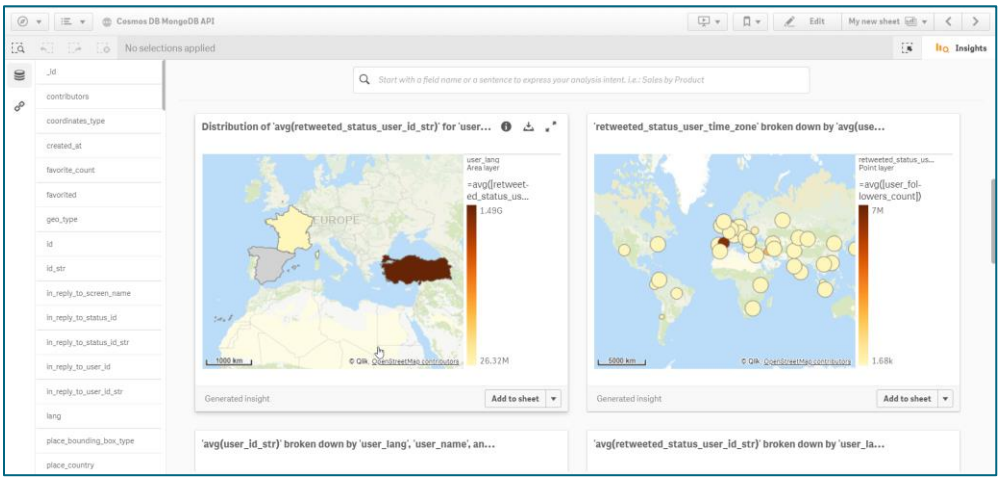
Note – again, since the schema is being dynamically created, it may take a moment… Qlik will detect only one table is being used and bypass our data prep engine. You can select “Data Manager” from the dropdown if you’re interested in manipulating the raw data.

The table by default is very wide with a lot of nulls, to speed up import and data prep (optional) deselect most of the “retweeted_*” content, but it is interesting if you want to visualize how a tweet propogates across the world.



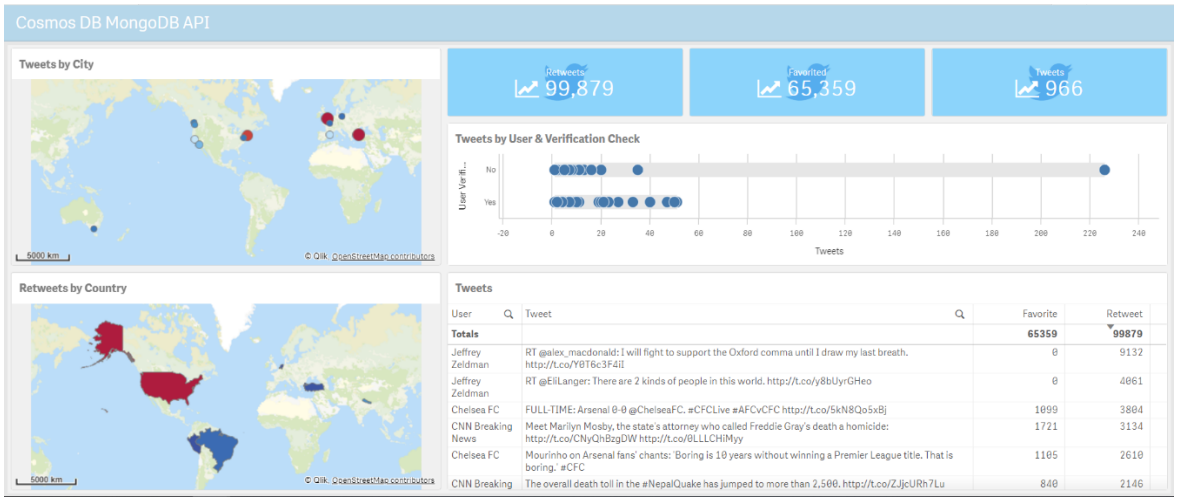
Step 6. Generate Insights…

Use the “Generate Insights” capability of Qlik to get an idea of what the engine finds interesting. Add a dimension or a measure to further refine the insights.



Step 7. Explore!

By either using insights or directly building on the canvas, we can build our app exploring video games sales!



SUMMARY & CONCLUSION

This document shows how to use Qlik Sense with the Cosmos DB MongoDB API account with a step-by-step tutorial. The MongoDB driver is built into Qlik Sense and makes connectivity simple.