How to Guide:

Connect Qlik to ArangoDB

In this tutorial we will use Qlik Sense Desktop with the Qlik REST connector for ArangoDB using ArangoDB 3.4.

In this tutorial we will use Qlik Sense Desktop with the Qlik REST connector and ArangoDB 3.4.

This tutorial assumes you have access to a running instance of ArangoDB that already has a collection with data you want to export to Qlik. Note that the connector for ArangoDB only supports exporting data from a single collection.

Installing the connector

The Qlik connector can be installed as a Foxx service using the ArangoDB web interface or the Foxx CLI.

To install the service using the web interface:

1. Open the ArangoDB web interface in your browser (e.g. `http://localhost:8529` if ArangoDB is running locally).
2. Enter your ArangoDB credentials and select the database from which you want to export data to Qlik.
3. Select the *Services* tab on the right, press *Add Service* and select the *qlik-connector* service.
4. Enter a mount point (e.g. `/qlik`) and press *Install*.

To install the service using the Foxx CLI use the following command (assuming user `root`, database `_system` and mount point `/qlik`):

```
foxx install -u root -P -H http://localhost:8529 -D _system /qlik \
https://github.com/arangodb-foxx/qlik-connector/archive/master.zip
```

Configuring the connector

Before the Qlik connector is ready to use it needs to be configured:

- **Collections**: the names of collections in the current database the connector should have access to (multiple values can be separated by comma but only one collection can be imported at a time).

- **Username** and **password**: credentials that will be used to protect the connector against unauthorized access. Note that these are different from the credentials used to access ArangoDB itself and will only be used by Qlik to authenticate against the connector.

To configure the service from the web interface:

1. Select the *Services* tab and select the mount point where the Qlik connector service was installed.
2. Select the *Settings* tab from the top bar.
3. Fill in the configuration values and press the *Apply* button to save.

To configure the service using the Foxx CLI use the following command (assuming collection `data`, username `qlik` and password `qlik123`):

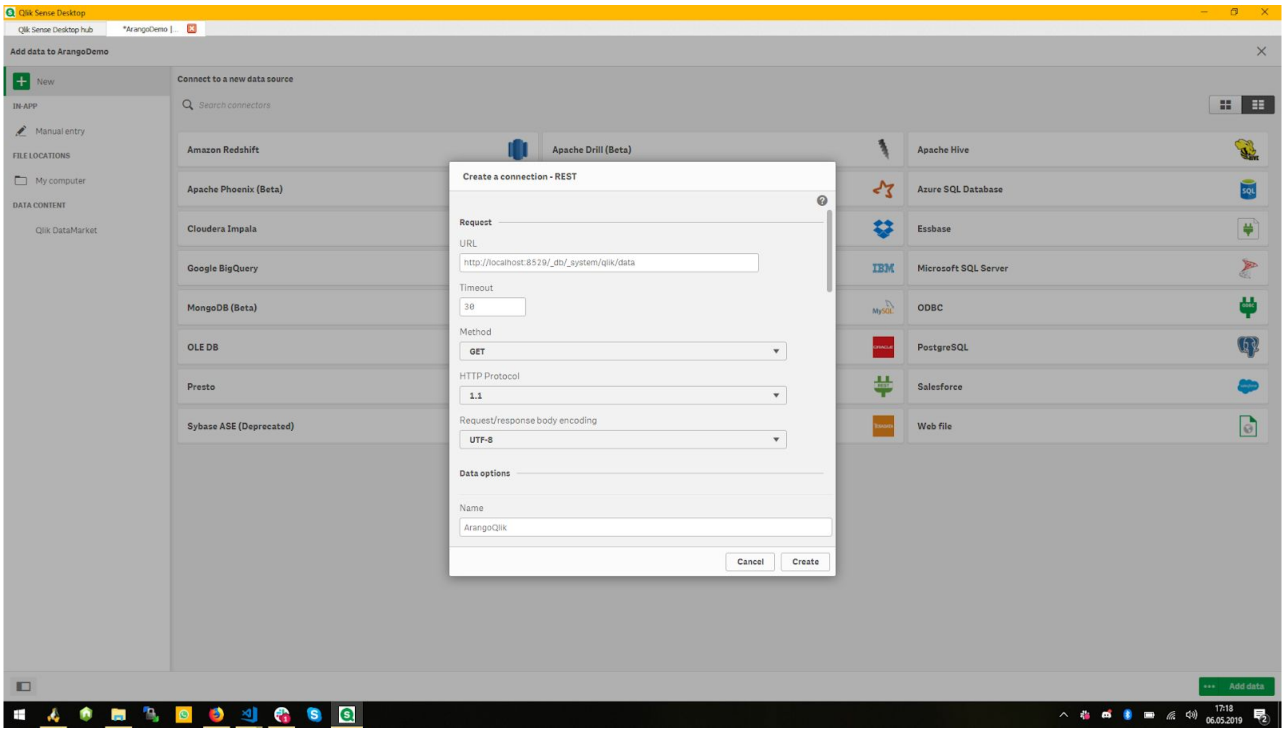
```
foxx config -u root -P -H http://localhost:8529 -D _system /qlik \
collections=data username=qlik password=qlik123
```

Using the connector in Qlik Sense Desktop

We'll start out by creating a new Qlik app using the *Create new app* button and calling it "ArangoDemo". After creating the app for us, Qlik automatically prompts us to open the app.

Press the large *Add data from files and other sources* button at the center of the screen to connect our Qlik app with ArangoDB. If you are using an existing Qlik app that already contains data, you can also use the *Add data* option from the app menu in the upper right corner of the screen.

On the *Connect to a new data source* screen select the *REST* data source. The data source list is sorted alphabetically but you can also find the connector more easily by entering "REST" in the search box.


Create a connection - REST

Request

URL: http://localhost:8529/_db/_system/qlik/data

Timeout: 30

Method: GET

HTTP Protocol: 1.1

Request/response body encoding: UTF-8

Data options

Name: ArangoQlik

The **URL** should be the full path of the Qlik connector, including the name of the collection you want to load. Note that we can only load one collection at a time and the connector needs to be configured to expose that collection.

For example, if we have a local copy of ArangoDB running on our machine, the Qlik connector was installed at `/qlik` and the name of the collection we want to import is `data`, our URL looks like this: `http://localhost:8529/_db/_system/qlik/data`

In the **Authentication** section we need to select **Basic** as the *Authentication Schema*. The **Username** and **Password** again need to be taken from the configuration of our Qlik connector. Remember that these are not the credentials used to connect to ArangoDB itself. They merely protect the Qlik connector against unauthorized access.

In the **Pagination** section we also need to select **Offset** as the *Pagination type*. This section is necessary to allow us to fetch more than the initial 100 documents from the collection. Use the following settings:

- **'Start' parameter name**: `start`
- **'Start' initial value**: `0`
- **'Count' parameter name**: `count`
- **'Count' initial value**: `200`
- **'Total records' path**: `meta/totalCount`
- **Data indicator path**: `data`

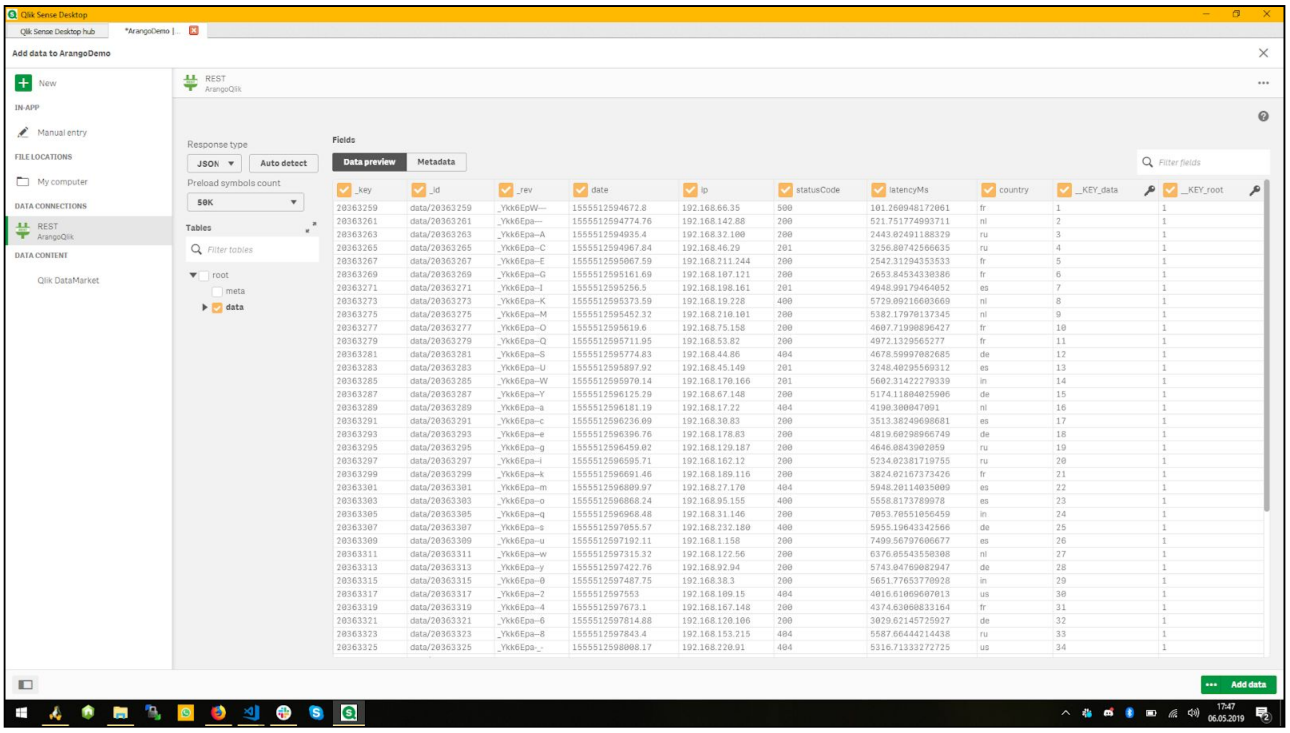
If your documents are very large, you can use a smaller value for the **'Count' initial value**. A smaller number means fewer documents will be fetched in each request but

Qlik will need to make more requests to import all the data. A larger number will result in fewer requests but more data in each request.

We'll call the connection "ArangoQlik". Press the *Create* button to proceed to the next step.

Data selection

If the connection succeeded, you should now see an empty table. You may have to explore the data a bit by drilling down the data tree to the left of the preview table. Try expanding the "root" entry and checking the box next to the "data" entry below it.



Response type: JSON — Auto detect

Preload symbols count: 50K

Tables

- root
  - meta
  - ✓ data

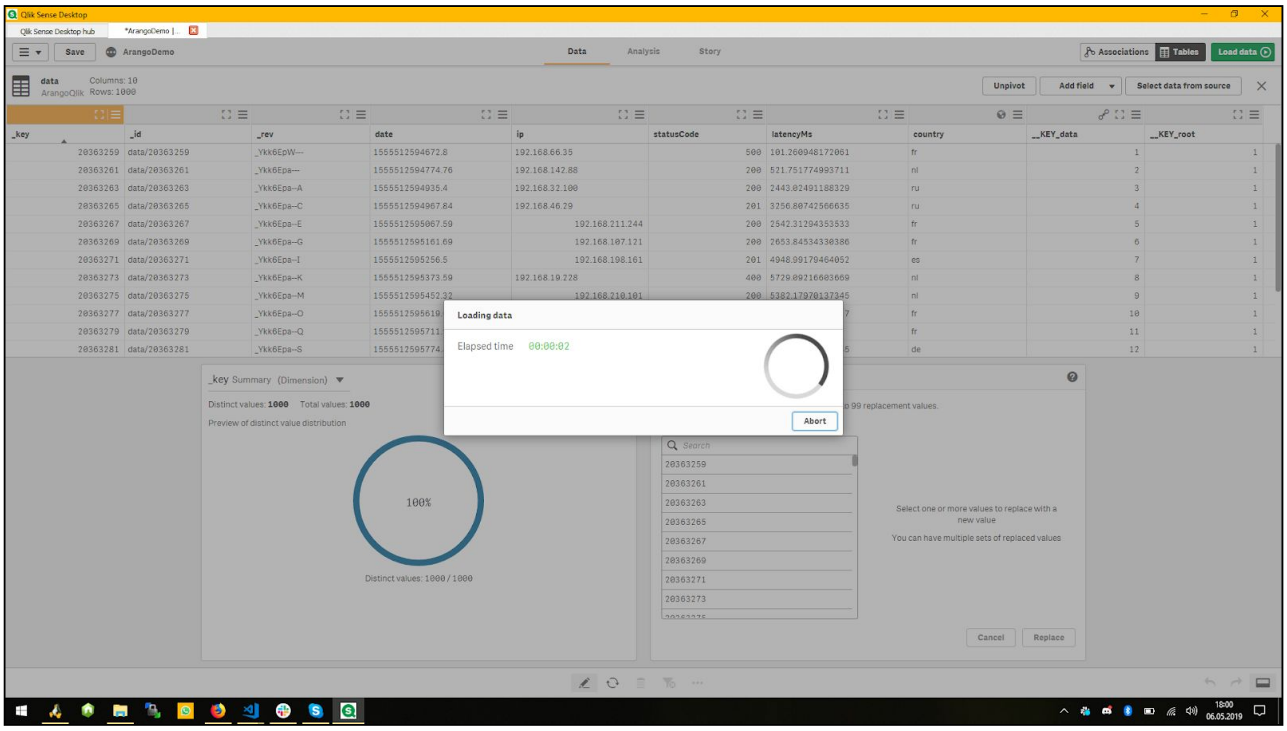
✓ _key	✓ _id	✓ _rev	✓ date	✓ ip	✓ statusCode	✓ latencyMs	✓ country	✓ __KEY_data	✓ __KEY_root
20363259	data/20363259	_Ykk6EpW---	1555512594672.8	192.168.66.35	500	101.260948172061	fr	1	1
20363261	data/20363261	_Ykk6Epa---	1555512594774.76	192.168.142.88	200	521.751774993711	nl	2	1
20363263	data/20363263	_Ykk6Epa--A	1555512594935.4	192.168.32.100	200	2443.02491188329	ru	3	1
20363265	data/20363265	_Ykk6Epa--C	1555512594967.84	192.168.46.29	201	3256.80742566635	ru	4	1
20363267	data/20363267	_Ykk6Epa--E	1555512595067.59	192.168.211.244	200	2542.31294353533	fr	5	1
20363269	data/20363269	_Ykk6Epa--G	1555512595161.69	192.168.107.121	200	2653.84534330386	fr	6	1
20363271	data/20363271	_Ykk6Epa--I	1555512595256.5	192.168.198.161	201	4948.99179464052	es	7	1
20363273	data/20363273	_Ykk6Epa--K	1555512595373.59	192.168.19.228	400	5729.09216603669	nl	8	1
20363275	data/20363275	_Ykk6Epa--M	1555512595452.32	192.168.210.101	200	5382.17970137345	nl	9	1
20363277	data/20363277	_Ykk6Epa--O	1555512595619.6	192.168.75.158	200	4607.71990896427	fr	10	1
20363279	data/20363279	_Ykk6Epa--Q	1555512595711.95	192.168.53.82	200	4972.1329565277	fr	11	1
20363281	data/20363281	_Ykk6Epa--S	1555512595774.83	192.168.44.86	404	4678.59997082685	de	12	1
20363283	data/20363283	_Ykk6Epa--U	1555512595897.92	192.168.45.149	201	3248.40295569312	es	13	1
20363285	data/20363285	_Ykk6Epa--W	1555512595970.14	192.168.170.166	201	5602.31422279339	in	14	1
20363287	data/20363287	_Ykk6Epa--Y	1555512596125.29	192.168.67.148	200	5174.11804025906	de	15	1
20363289	data/20363289	_Ykk6Epa--a	1555512596181.19	192.168.17.22	404	4190.300047091	nl	16	1
20363291	data/20363291	_Ykk6Epa--c	1555512596236.09	192.168.30.83	200	3513.38249698681	es	17	1
20363293	data/20363293	_Ykk6Epa--e	1555512596396.76	192.168.178.83	200	4819.60298966749	de	18	1
20363295	data/20363295	_Ykk6Epa--g	1555512596459.02	192.168.129.187	200	4646.0843902059	ru	19	1
20363297	data/20363297	_Ykk6Epa--i	1555512596595.71	192.168.162.12	200	5234.02381719755	ru	20	1
20363299	data/20363299	_Ykk6Epa--k	1555512596691.46	192.168.189.116	200	3824.02167373426	fr	21	1
20363301	data/20363301	_Ykk6Epa--m	1555512596809.97	192.168.27.170	404	5948.20114035009	es	22	1
20363303	data/20363303	_Ykk6Epa--o	1555512596868.24	192.168.95.155	400	5558.8173789978	es	23	1
20363305	data/20363305	_Ykk6Epa--q	1555512596968.48	192.168.31.146	200	7053.70551056459	in	24	1
20363307	data/20363307	_Ykk6Epa--s	1555512597055.57	192.168.232.180	400	5955.19643342566	de	25	1
20363309	data/20363309	_Ykk6Epa--u	1555512597192.11	192.168.1.158	200	7499.56797606677	es	26	1
20363311	data/20363311	_Ykk6Epa--w	1555512597315.32	192.168.122.56	200	6376.05543550308	nl	27	1
20363313	data/20363313	_Ykk6Epa--y	1555512597422.76	192.168.92.94	200	5743.04769082947	de	28	1
20363315	data/20363315	_Ykk6Epa--0	1555512597487.75	192.168.38.3	200	5651.77653770928	in	29	1
20363317	data/20363317	_Ykk6Epa--2	1555512597553	192.168.109.15	404	4016.61069607013	us	30	1
20363319	data/20363319	_Ykk6Epa--4	1555512597673.1	192.168.167.148	200	4374.63060833164	fr	31	1
20363321	data/20363321	_Ykk6Epa--6	1555512597814.88	192.168.120.106	200	3029.62145725927	de	32	1
20363323	data/20363323	_Ykk6Epa--8	1555512597843.4	192.168.153.215	404	5587.66444214438	ru	33	1
20363325	data/20363325	_Ykk6Epa-_-	1555512598008.17	192.168.220.91	404	5316.71333272725	us	34	1

You can remove the checkbox next to individual columns to avoid that data from being imported. You can also rename fields by clicking on the column name. Note that qlik automatically generates primary keys in addition to the fields returned by the connector. Removing these fields may prevent Qlik from loading the data correctly.

Once you're satisfied with your changes, proceed to the next step by pressing the *Add data* button. Depending on the size of the collection, this may take a number of seconds.

Associations and tables

If the data you imported contains nested objects, Qlik may suggest associations. Otherwise you may want to further massage the imported data in the *Tables* tab.



data ArangoQlik Columns: 10 Rows: 1000

_key	_id	_rev	date	ip	statusCode	latencyMs	country	__KEY_data	__KEY_root
20363259	data/20363259	_Ykk6EpW---	1555512594672.8	192.168.66.35	500	101.260948172061	fr	1	1
20363261	data/20363261	_Ykk6Epa---	1555512594774.76	192.168.142.88	200	521.751774993711	nl	2	1
20363263	data/20363263	_Ykk6Epa--A	1555512594935.4	192.168.32.100	200	2443.02491188329	ru	3	1
20363265	data/20363265	_Ykk6Epa--C	1555512594967.84	192.168.46.29	201	3256.80742566635	ru	4	1
20363267	data/20363267	_Ykk6Epa--E	1555512595067.59	192.168.211.244	200	2542.31294353533	fr	5	1
20363269	data/20363269	_Ykk6Epa--G	1555512595161.69	192.168.107.121	200	2653.84534330386	fr	6	1
20363271	data/20363271	_Ykk6Epa--I	1555512595256.5	192.168.198.161	201	4948.99179464052	es	7	1
20363273	data/20363273	_Ykk6Epa--K	1555512595373.59	192.168.19.228	400	5729.09216603669	nl	8	1
20363275	data/20363275	_Ykk6Epa--M	1555512595452.32	192.168.210.101	200	5382.17970137345	nl	9	1
20363277	data/20363277	_Ykk6Epa--O	1555512595619.			7	fr	10	1
20363279	data/20363279	_Ykk6Epa--Q	1555512595711.				fr	11	1
20363281	data/20363281	_Ykk6Epa--S	1555512595774.			5	de	12	1

o 99 replacement values.

20363259
20363261
20363263
20363265
20363267
20363269
20363271
20363273

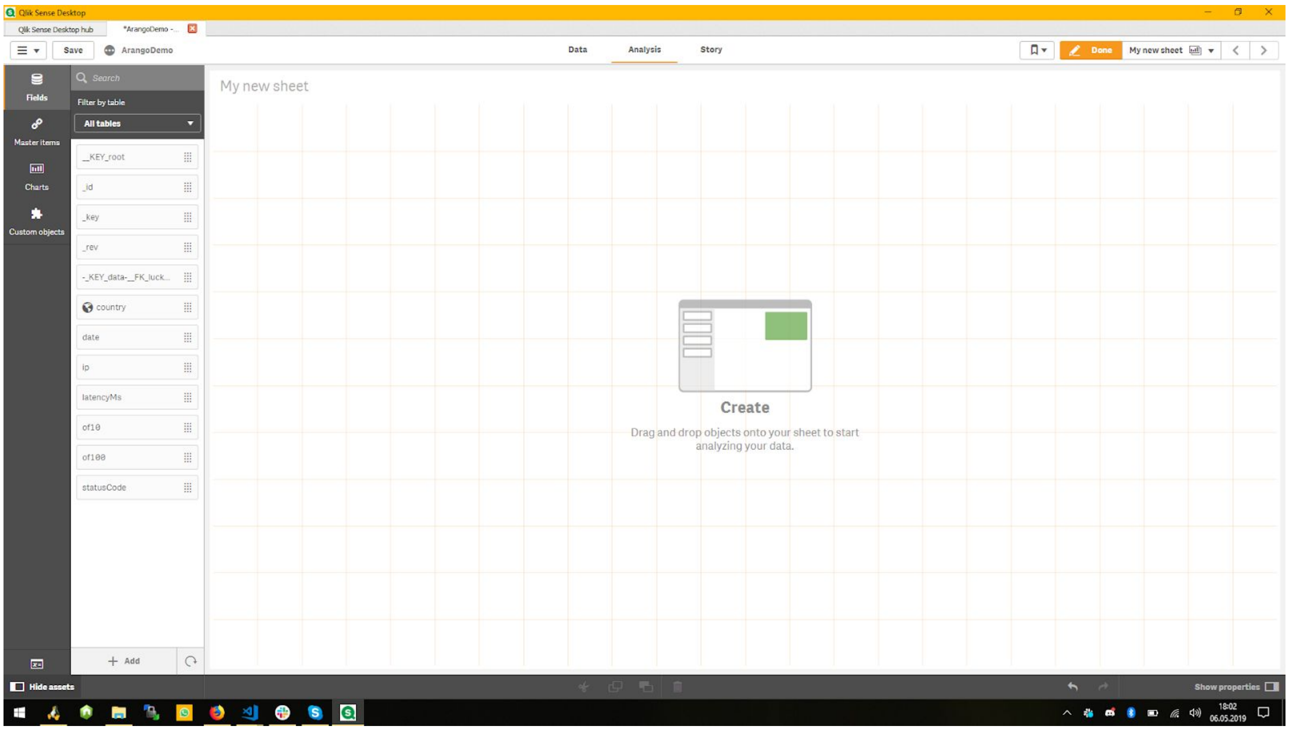
Select one or more values to replace with a new value

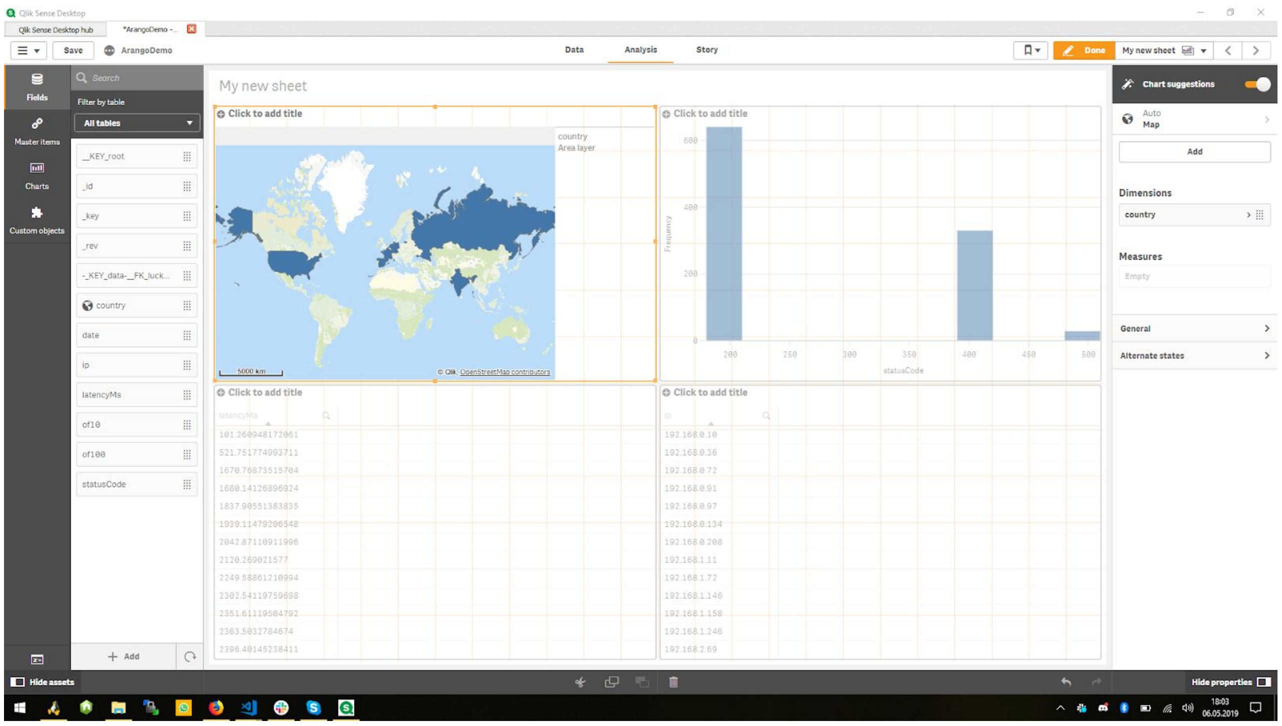
You can have multiple sets of replaced values

Finally you can load the data by pressing the *Load data* button in the upper right. Note that this will again take several seconds or minutes depending on the size of the collection as Qlik transforms and indexes the imported data set.

Creating a data sheet

With all the data imported into Qlik, you can now visualize it by dragging fields onto the data sheet.





latencyMs
101.260948172061
521.751774993711
1670.76873515704
1680.14126896924
1837.90551383835
1939.11479206548
2042.87110911996
2120.269021577
2249.58861210994
2302.54119759698
2351.61119504792
2363.5032784674
2396.40145238411

ip
192.168.0.10
192.168.0.36
192.168.0.72
192.168.0.91
192.168.0.97
192.168.0.134
192.168.0.208
192.168.1.11
192.168.1.72
192.168.1.146
192.168.1.158
192.168.1.246
192.168.2.69

Extending the connector with filters

As the Qlik REST connector is a normal Foxx service you can modify the source code to create your own connector. In this section we will extend the existing connector with an option to filter the collection dynamically before handing it over to Qlik.

Using the web interface open the settings tab also used to configure the service earlier and press the download icon in the upper right. This downloads a zip bundle of the service's source code.

Extract the zip archive to any folder on your computer and open the file `index.js` in a code editor.

Defining the operators

There are a lot of useful operators in AQL but for this example we'll stick to the basics. Add the following in the source code before the line starting with `const COLLECTIONS`:

```
const OPERATORS = new Map([
  ["lt", aql.literal("<")],
  ["lte", aql.literal("<=")],
  ["gt", aql.literal(">")],
  ["gte", aql.literal(">=")],
  ["eq", aql.literal("==")],
  ["neq", aql.literal("!=")],
  ["in", aql.literal("in")],
  ["nin", aql.literal("not in")]
]);
```

This gives us a mapping of safe but human-readable names to AQL operators. The `aql.literal` function converts the strings to something we can use in an AQL query template without having to worry about being misinterpreted as a bind variable.

Extending the query parameters

We want to allow users to specify multiple filters. The easiest way to do this with the existing GET route is by adding a query parameter that takes a JSON value.

Find one of the lines starting with `.queryParam(` and add the following immediately before that line:

```
  .queryParam(
    "filters",
    joi
      .array()
      .items(
        joi
          .object()
          .keys({
            fieldName: joi.string().required(),
            operator: joi.only(...OPERATORS.keys()).required(),
            value: joi.any().required()
          })
          .required()
      )
      .optional(),
    "Filter expressions to match the documents against."
  )
```

In plain English this matches an optional JSON array containing objects with three attributes:

- **fieldName**: a string value we will use to decide which field to compare
- **operator**: one of the operator names we defined earlier
- **value**: a value the field will be compared to using the operator

For example, this would limit the results to documents with a `statusCode` field set to either `400` or `500`:

```
[{ "fieldName": "statusCode", "operator": "in", "value": [400, 500] }]
```

Applying the filter

Find the following lines in the source code:

```
const { start, count } = req.queryParams;

const { query, bindVars } = aql`
  FOR doc IN ${collection}
  LIMIT ${start}, ${count}
  RETURN doc
`;
```

Replace those lines with the following code:

```
const { start, count, filters: rawFilters } = req.queryParams;

const filters = rawFilters
  ? rawFilters.map(
      ({ fieldName, operator, value }) =>
        aql`FILTER doc[${fieldName}] ${OPERATORS.get(operator)} ${value}`
    )
  : [];

const { query, bindVars } = aql`
  FOR doc IN ${collection}
  ${aql.join(filters)}
  LIMIT ${start}, ${count}
  RETURN doc
`;
```

Installing the modified connector

To reflect the changes to the source code in the installed service, first create a zip archive of your working copy with the saved changes to the `index.js` file.

To upgrade the service using the web interface:

1. Open the service's *Settings* tab and press the *Replace* button.
2. Open the *Upload* tab and press the *Upload File* button to select the zip file.
3. Press the *Replace* button, don't modify any of the options.
4. Confirm the dialog by pressing the *Replace* button.

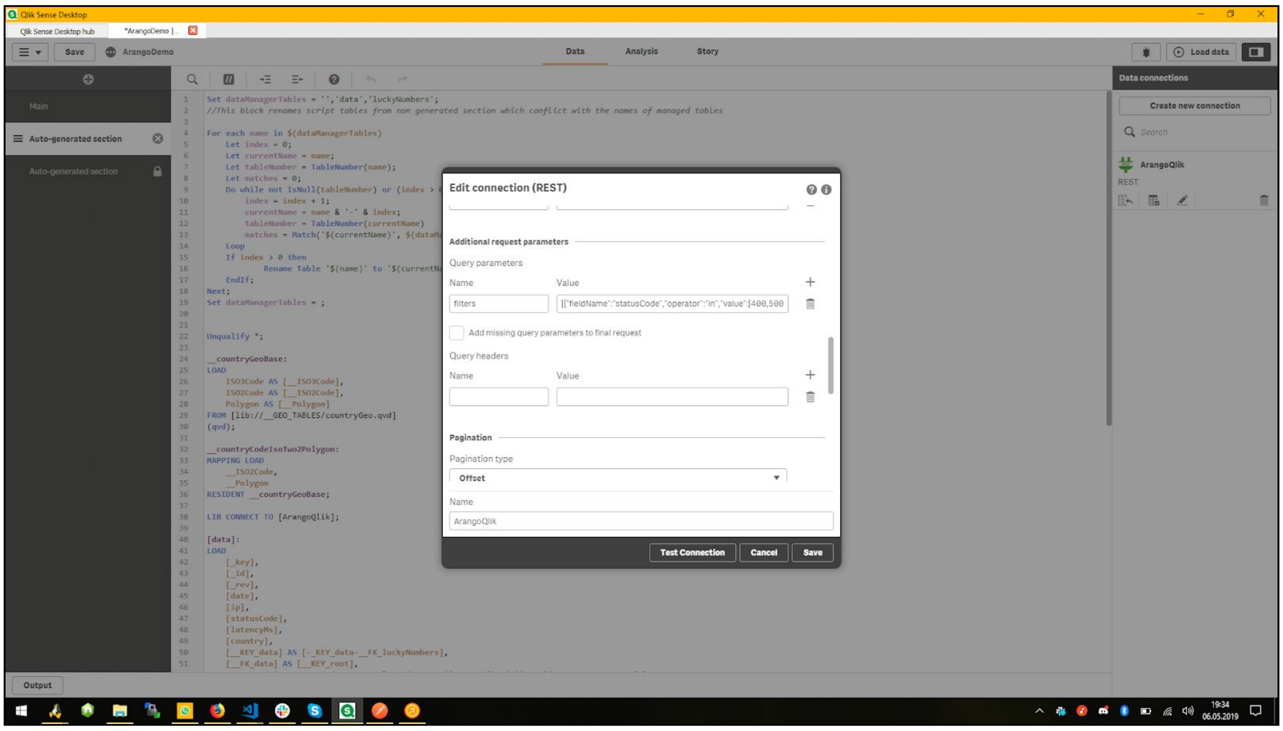
To upgrade the service using Foxx CLI (assuming filename `qlik.zip` and that the file is in the current directory):

```
foxx upgrade -u root -P -H http://localhost:8529 -D _system /qlik
qlik.zip
```

Using the filter in Qlik Sense Desktop

Open the *Edit Connection* dialog or follow the instructions to add a new data source using the Qlik REST connector but in the section **Additional request parameters** add the following for *Query parameters*:

- **Name**: `filters`
- **Value**: `[{"fieldName":"statusCode","operator":"in","value":[400,500]}]`



```
Set dataManagerTables = '','data','luckyNumbers';
//This block renames script tables from non generated section which conflict with the names of managed tables

For each name in $(dataManagerTables)
    Let index = 0;
    Let currentName = name;
    Let tableNumber = TableNumber(name);
    Let matches = 0;
    Do while not IsNull(tableNumber) or (index > 
        index = index + 1;
        currentName = name & '-' & index;
        tableNumber = TableNumber(currentName)
        matches = Match('$(currentName)', $(dataM
    Loop
    If index > 0 then
            Rename Table '$(name)' to '$(currentN
    EndIf;
Next;
Set dataManagerTables = ;


Unqualify *;

__countryGeoBase:
LOAD
    ISO3Code AS [__ISO3Code],
    ISO2Code AS [__ISO2Code],
    Polygon AS [__Polygon]
FROM [lib://__GEO_TABLES/countryGeo.qvd]
(qvd);

__countryCodeIsoTwo2Polygon:
MAPPING LOAD
    __ISO2Code,
    __Polygon
RESIDENT __countryGeoBase;

LIB CONNECT TO [ArangoQlik];

[data]:
LOAD
    [_key],
    [_id],
    [_rev],
    [date],
    [ip],
    [statusCode],
    [latencyMs],
    [country],
    [__KEY_data] AS [-_KEY_data-__FK_luckyNumbers],
    [__FK_data] AS [__KEY_root],
```

Edit connection (REST)

Additional request parameters

Query parameters

Name	Value
filters	[{"fieldName":"statusCode","operator":"in","value":[400,500

Add missing query parameters to final request

Query headers

Name	Value

Pagination

Pagination type: Offset

Name: ArangoQlik

Test Connection | Cancel | Save

Note that you can substitute whatever filters make sense for your data instead.

Once you're satisfied with your filter expression, press the *Save* button to confirm the changes or the *Create* button if you are adding a new data source instead.