Introduction to

Hands-On Workshop

Lab 4 - Triggers

Overview

In Lab 1 of this workshop you created a MongoDB cluster and loaded some data. A really powerful aspect of MongoDB Atlas are Triggers, which allow you listen for changes to your data and react in real time.

For the purpose of this workshop, whenever a new restaurant is added to the collection, we'll also add that restaurant to a health inspector collection for review.

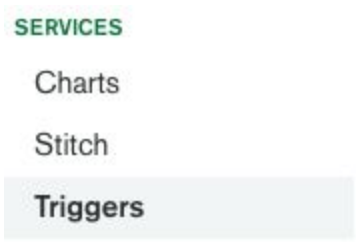
Prerequisites

You've completed Lab 1, in which you used **MongoDB Atlas** to deploy a MongoDB cluster.

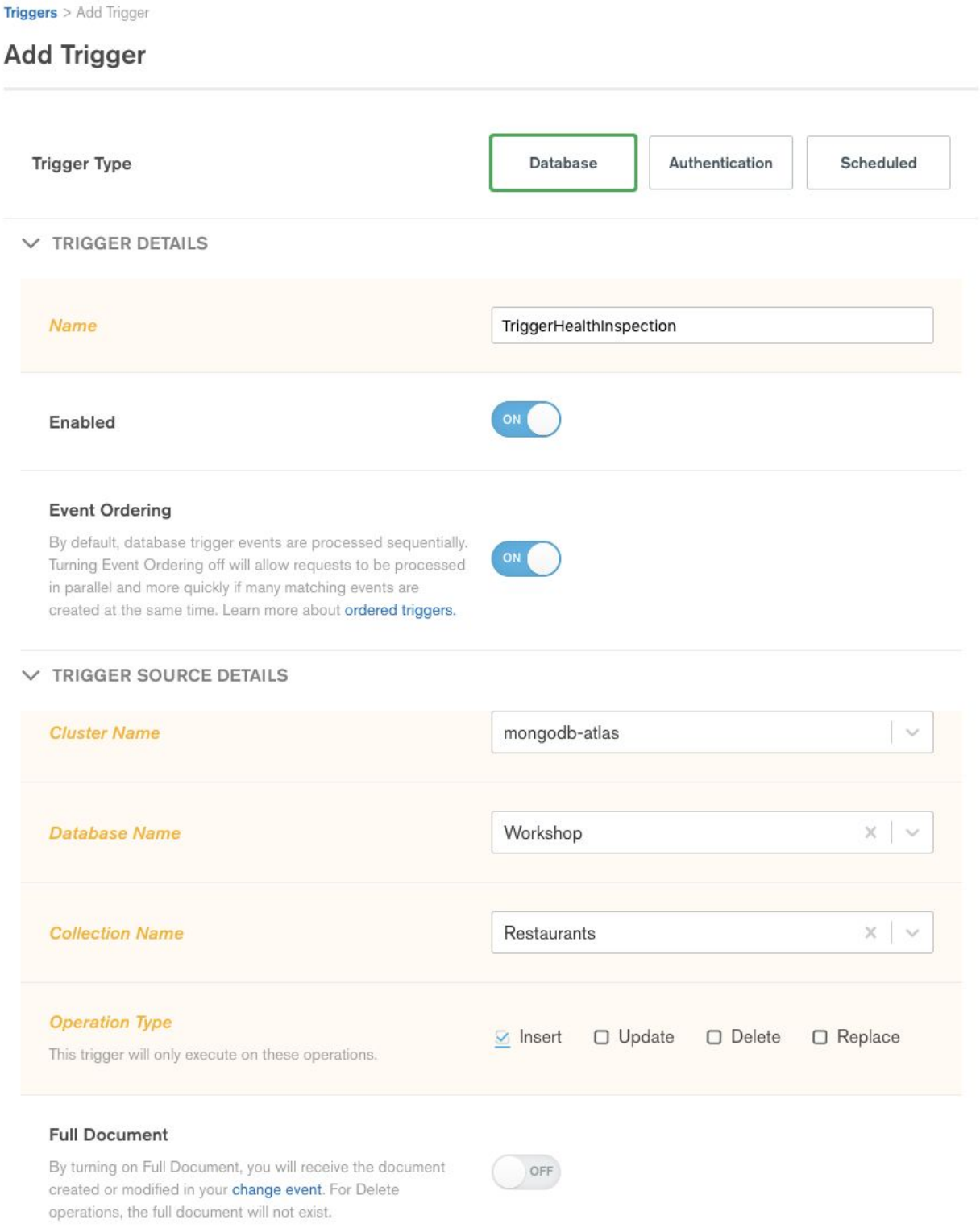
Hands-On Exercises

Exercise 1 - Create a Trigger

In the Atlas UI, click the **Triggers** menu on the left:



Click **Add a Trigger**. Supply the information as shown below:



Finally, we need to provide the function that will execute when the trigger fires. Click **Function**, and in the Function drop-down list select "**+ New Function**". Give your trigger a helpful name, such as "trgNewHealthInspection".

Review the sample code that's provided to see what's possible. When ready, replace the sample code with the following, which simply gets the new document from the changeEvent and inserts it into a "NewRestaurants" collection.

> *Note, make sure the variables **linkedCluster** and **databaseName** match the values you selected above when configuring the trigger.*

```
exports = function(changeEvent) {

  // Get the new document from the changeEvent...
  var fullDocument = changeEvent.fullDocument;

  // Connect to the Workshop database
  // (make sure these values match your environment)
  var linkedCluster = "mongodb-atlas";
  var databaseName = "Workshop";
  var db = context.services.get(linkedCluster).db(databaseName);

  // Save the restaurant to the new collection
  collection = db.collection("NewRestaurants");
  var status = collection.insertOne(fullDocument);
};
```

Function

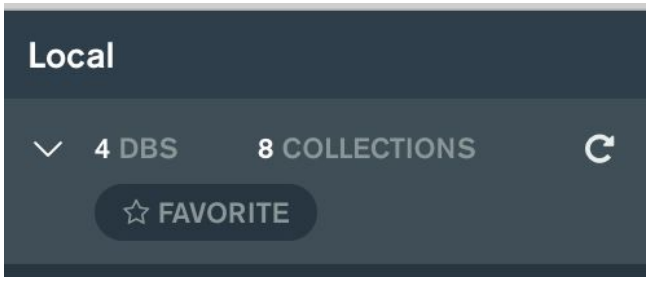
The following code will be executed for each event. The default function provided has no side effects. You can edit this code later.

```
1
2  exports = function(changeEvent) {
3
4    // Get the new document from the changeEvent...
5    var fullDocument = changeEvent.fullDocument;
6
7    // Connect to the Workshop database
8    // (make sure these values match your environment)
9    var linkedCluster = "mongodb-atlas";
10   var databaseName = "Workshop";
11   var db = context.services.get(linkedCluster).db(databaseName);
12
13   // Save the restaurant to the new collection
14   collection = db.collection("NewRestaurants");
15   var status = collection.insertOne(fullDocument);
16 };
17
18
```

Save the trigger, then **Review & Deploy** the changes.

Exercise 2 - Test the Trigger

If you completed the exercise building the new restaurant application with Stitch, you can now use that application to test the trigger functionality that you just implemented. Simply add a new restaurant, and then verify the results using either Compass or Atlas. We're expecting to see a new collection named "NewRestaurants", which should contain any new restaurants added since the trigger was enabled. If using Compass, note the refresh button in the top-left corner of the UI, which will refresh the list of available collections (not to be confused with the right-hand Refresh button that only refreshes the current query).



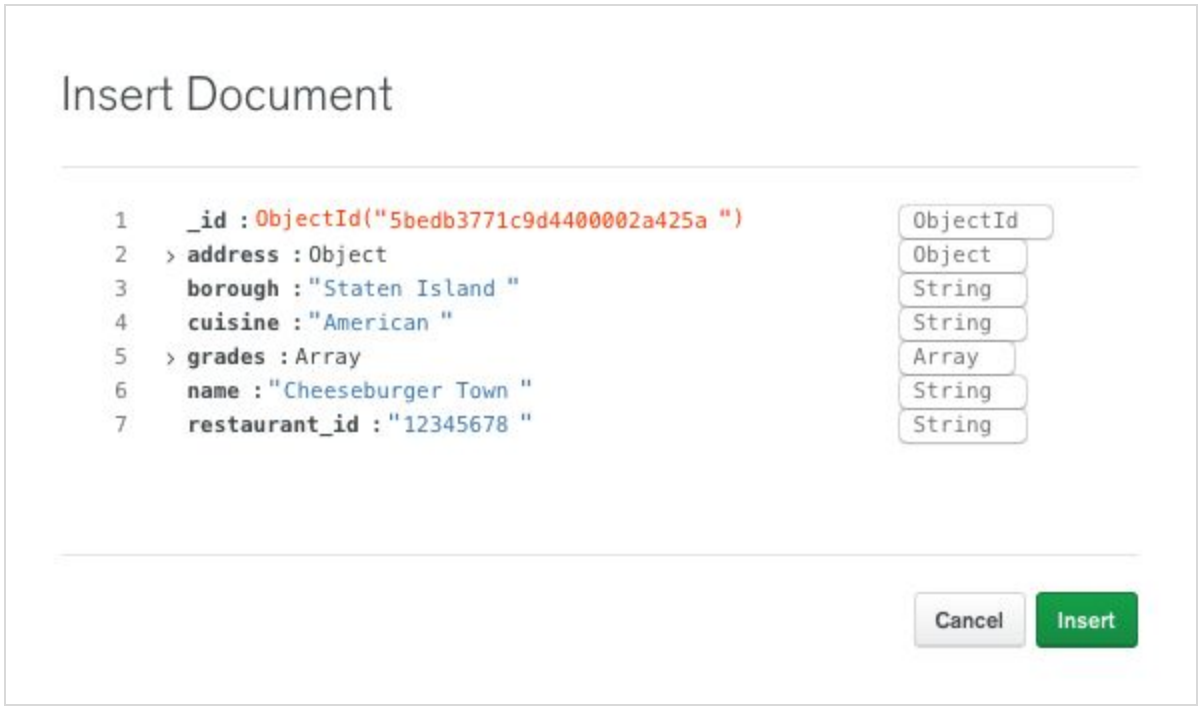
You can also view these results within Atlas by navigating to **Clusters** and then clicking the Collections button in the appropriate Cluster.

If the Stitch application is unavailable, you can still test the trigger using the data explorer in Atlas. Navigate back to your cluster by selecting **Clusters** in the menu on the left:

Then click the **Collections** button.

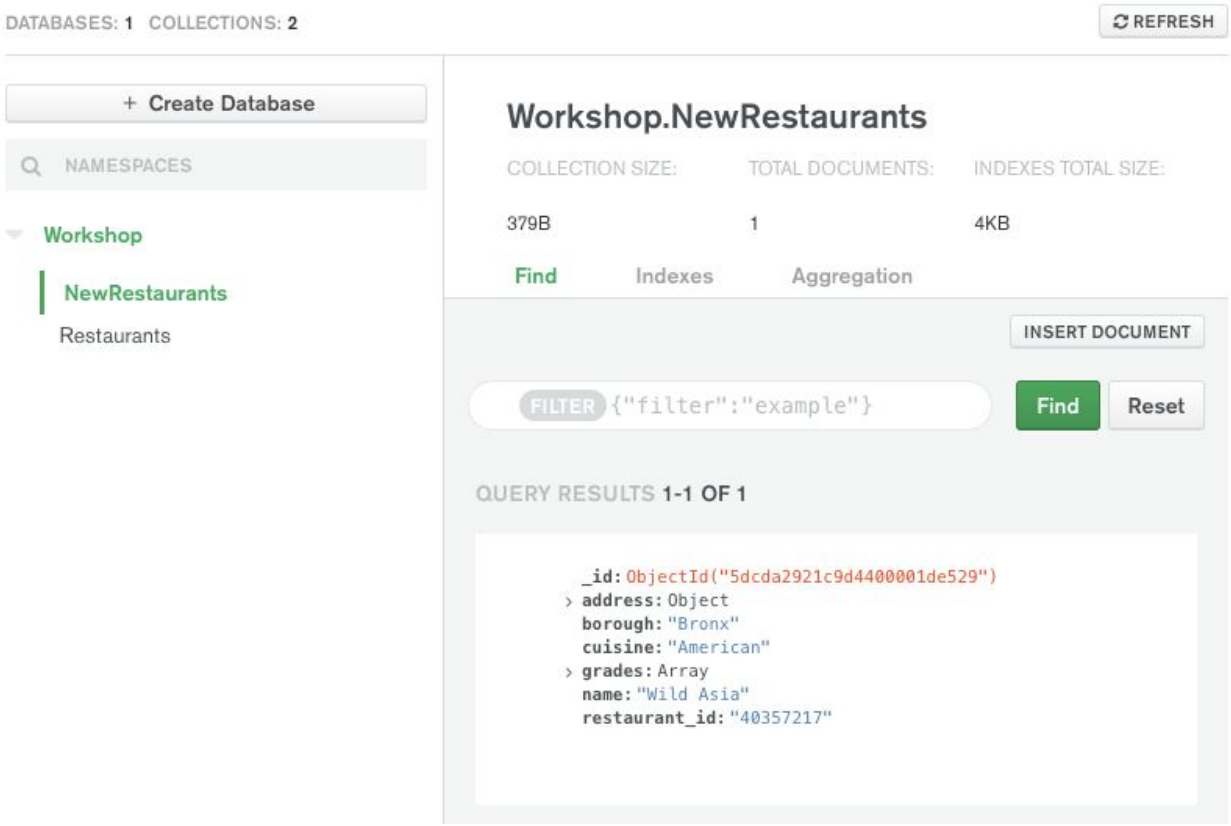
Hover over a document in the Restaurants collection and click the clone icon document icon  that appears in the upper right:

Optionally change some of the fields:



Click **Insert**.

Click the REFRESH button in the right corner to see your new collection:



Congratulations!

You've completed this lab on Atlas Triggers, and can now build reactive applications and services that respond to database changes as they occur. Note that Atlas Triggers are built on top of Change Streams, which allows clients of all types (i.e. any official MongoDB driver) to subscribe to changes occurring within the database.