Let it Sync! Using Monstache to sync MongoDb and Elasticsearch in real-time.

Many of us might have heard about elasticsearch. It is an open-source NoSQL search engine that is commonly used to search and analyze data. With that being said, we know that elasticsearch is not recommended to be used as a primary database, hence we always need a database to be used with Elasticsearch and keep them synced!

In order to sync elasticsearch with relational databases, there are tools like JDBC and logstash, and many tutorials and articles on how to integrate that but elasticsearch does not provide the required MongoDB JDBC support, which leaves us with very few tools which can be used to sync MongoDB and elasticsearch:

Mongo Connector:

According to its Github:

“ mongo-connector creates a pipeline from a MongoDB cluster to one or more target systems, such as Solr, Elasticsearch, or another MongoDB cluster. “

But the drawback of Mongo Connector is that it does not have very good support for Elasticsearch 6+. Not to forget that its repository has not been updated for more than a year.

Trasporter:

Transporter is a good tool to export data from MongoDB to elasticsearch, but it does not provide real-time syncing.

Mongoosastic:

According to its Github:

“Mongoosastic is a mongoose plugin that can automatically index

your models into elasticsearch.”

But, it is only useful when changes in MongoDB are done through the server, any changes done directly to MongoDB will not reflect in Elasticsearch

Sync in real-time with Monstache!

Monstache is a sync daemon written in Go that syncs MongoDB collections into Elasticsearch in real-time. It is possible using monstache to index entire MongoDB collections into elasticsearch, and after indexing, monstache will also keep everything synced.

How it syncs in real-time?

Monstache reads oplogs of the MongoDB that is connected to it, to sync every operation that is performed on MongoDB.

You will need to ensure that MongoDB is configured to produce an oplog by deploying a replica set. If you haven’t already done so, follow the 5 step procedure to initiate and validate your replica set. For testing it locally, your replica set may contain a single member.

Before we move forward, let me mention some features of monstache listed here:

Supports up to and including the latest versions of Elasticsearch and MongoDB

Single binary with a light footprint

Support for MongoDB change streams and aggregation pipelines

Pre built Docker containers

Optionally filter the set of collections to sync

Direct read mode to do a full sync of collections in addition to tailing the oplog

Transform and filter documents before indexing using Golang plugins or JavaScript ( What I like the most! )

Source:

<https://www.folio3.com/mobile/blog/sync-mongodb-monstache-elasticsearch/>

[Monstache¶](https://rwynn.github.io/monstache-site/)

Getting Started

Installation

[Monstache](https://github.com/rwynn/monstache) is just a single binary without dependencies on runtimes like Ruby, Python or PHP. Monstache is written in Go but you don't need to install the Go language unless you decide to write your own Go plugins.

If you simply want to run Monstache you just need to [download the latest version](https://github.com/rwynn/monstache/releases).

Usage:

Monstache uses the MongoDB [oplog](https://docs.mongodb.com/manual/core/replica-set-oplog/) as an event source. You will need to ensure that MongoDB is configured to produce an oplog by [deploying a replica set](http://docs.mongodb.org/manual/tutorial/deploy-replica-set/).

I deployed replica set on my [windows](https://github.com/YashPaliwal000/MongoDB-Replication/blob/main/CreateReplicaMongoDB%20WIN10.txt) environment and on docker ([docker-compose-file](https://github.com/YashPaliwal000/MongoDB-Replication/blob/main/docker-compose.yml))

**Note**

If you have enabled security in MongoDB you will need to give the user in your connection string certain privileges:

For MongoDB versions prior to 3.6 the user in the connection string will need to be able read the local database (to read from the oplog) and any user databases that you wish to synch data from.

When using the resume or clustering features the user will need to be able to write to and create indexes for the monstache database, or more generally, whatever you configure the option config-database-name to be.

When using change streams you will need to implement the changes in the documentation for [access control](https://docs.mongodb.com/manual/changeStreams/). Monstache defaults to opening the change stream against the entire deployment.

Without any explicit configuration monstache will connect to Elasticsearch and MongoDB on localhost on the default ports and begin tailing the MongoDB oplog. Any changes to MongoDB while Monstache is running will be reflected in Elasticsearch.

To see the indexes created by Monstache you may want to issue the following command which will show the indices in Elasticsearch. By default, the index names will match the db.collection name in MongoDB.

curl localhost:9200/\_cat/indices?v

Monstache uses the [TOML](https://github.com/toml-lang/toml) format for its configuration. You can run monstache with an explicit configuration by passing the -f flag.

monstache -f /path/to/config.toml

Simple example config.toml

##########################################################

# connect to MongoDB using the following URL

mongo-url ="mongodb://localhost:27020,localhost:27021/?replicaSet=rhngdb"

# connect to the Elasticsearch REST API at the following node URLs

elasticsearch-urls = ["http://localhost:9200"]

# use 4 go routines concurrently pushing documents to Elasticsearch

elasticsearch-max-conns = 10

# do not start processing at the beginning of the MongoDB oplog

# if you set the replay to true you may see version conflict messages

# in the log if you had synced previously. This just means that you are replaying old docs which are already

# in Elasticsearch with a newer version. Elasticsearch is preventing the old docs from overwriting new ones.

replay = false

# resume processing from a timestamp saved in a previous run

resume = true

enable-oplog = true

# override the name under which resume state is saved

resume-name = "productresume"

# if you need to seed an index from a collection and not just listen and sync changes events# you can copy entire collections or views from MongoDB to Elasticsearch

direct-read-namespaces = ["hngdb.product","hngdb.store"]

# if you want to use MongoDB change streams instead of legacy oplog tailing use change-stream-namespaces

# change streams require at least MongoDB API 3.6+

# if you have MongoDB 4+ you can listen for changes to an entire database or entire deployment

# in this case you usually don't need regexes in your config to filter collections unless you target the deployment

# to listen to an entire db use only the database name. For a deployment use an empty string.

change-stream-namespaces = ["hngdb.product","hngdb.store"]

index-as-update = true

# print detailed information including request traces

verbose = true

# do not exit after full-sync, rather continue tailing the oplog

exit-after-direct-reads = false

[[mapping]]

namespace = "hngdb.product" #target collection

index = "product\_idx" # index created in ES.

[[mapping]]

namespace = "hngdb.store"

index = "store\_idx"

################################################################

**Use cases I worked on.**

1.sync collection from mongodb to Elastic search.

Using monstache it was easy to sync any collection to index in ES.Index created automatically in ES(if enabled in ES).By default, the index names will match the db.collection name in MongoDB or we can name index by using mapping inside config.toml. Or we can target already created index in ES.

The above example config.toml create two index of respective mapping.

Index Mapping

When indexing documents from MongoDB into Elasticsearch the default mapping is as follows:

For Elasticsearch prior to 6.2

Elasticsearch index name <= MongoDB database name . MongoDB collection name

Elasticsearch type <= MongoDB collection name

Elasticsearch document \_id <= MongoDB document \_id

For Elasticsearch 6.2+

Elasticsearch index name <= MongoDB database name . MongoDB collection name

Elasticsearch type <= \_doc

Elasticsearch document \_id <= MongoDB document \_id

If these default won't work for some reason you can override the index and type mapping on a per collection basis by adding the following to your TOML config file:

[[mapping]]namespace= "test.test

"index= "index1

"type= "type1"

[[mapping]]namespace= "test.test2"

index= "index2"

type= "type2"

With the configuration above documents in the test.test namespace in MongoDB are indexed into the index1 index in Elasticsearch with the type1 type.

If you need your index and type mapping to be more dynamic, such as based on values inside the MongoDB document, then see the sections [Middleware](https://rwynn.github.io/monstache-site/advanced/) and [Routing](https://rwynn.github.io/monstache-site/advanced/).

2.sync two or more collection with respective index.

We can direct read from two collection from different database and sync with respective indices.

Note: we can open change-stream for collection in mongodb 3.6+ and 4+ can listen for changes to an entire database or entire deployment.But if we are syncing collection having existing data we have to use direct read initially to get data in ES .For details how change-stream work read

NOTE:Change streams are available for [replica sets](https://docs.mongodb.com/manual/replication/) and [sharded clusters](https://docs.mongodb.com/manual/sharding/)

<https://docs.mongodb.com/manual/changeStreams/>

3.Sync three different collection from different DB with single Index and get data in specific format in index.like getting product data from one collection and store data from other collection and sync this in ES in single index.expectation was getting single document in index from two documents from mongodb collections.

NOTE: 1.change-stream cannot be open for views.You can open change streams against a collection,a database,or a deployment.

2.we can direct-read in ES from view in mongodb.(What I used for this sync)

For getting data from different collection I used aggregation pipeline in mongodb. Mongodb support aggregations so using aggregation I created a pipeline .

db.product.aggregate([{$lookup: {

from: 'store',

localField: 'skuId',

foreignField: 'sku\_code',

as: 'storedetails'

}}])

And using this pipeline created a view in mongodb and sync this view to index in ES.

Now for this sync to work properly we have to make changes in config.toml file.

Monstache provide Mongodb View replication.

We can use the relate config to do this.

[[relate]]

namespace = "hngdb.product"

with-namespace = "hngdb.viewtest"

keep-src = false

[[relate]]

namespace = "hngdb.store"

with-namespace = "hngdb.viewtest"

src-field = "sku\_code"

match-field = "skuId"

keep-src = false

Consider we have a collections product and store. A product has an associated store and a product is linked to a store via a field skuId which points to the sku\_code of the associated store in the store collection.

We can create a view in MongoDB that uses a $lookup to pull the store information in and present a view of product with the store information included.

Our config.toml now looks like:

#############################################################

mongo-url = "mongodb://localhost:27020,localhost:27021/?replicaSet=rhngdb"

elasticsearch-urls = ["http://localhost:9200"]

elasticsearch-max-conns = 10

replay = false

resume = true

enable-oplog = true

resume-name = "productresume"

change-stream-namespaces = ["hngdb.product","hngdb.store"]

direct-read-namespaces = [ "hngdb.viewtest"]

index-as-update = true

verbose = true

exit-after-direct-reads = false

[[mapping]]

namespace = "hngdb.viewtest"

index="myviewtest1"

[[relate]]

namespace = "hngdb.product"

with-namespace = "hngdb.viewtest"

keep-src = false

[[relate]]

namespace = "hngdb.store"

with-namespace = "hngdb.viewtest"

src-field = "sku\_code"

match-field = "skuId"

keep-src = false

###############################################################

4.Getting specific field from mongodb to ES.

Expectations: If there are 10 fields in a collection we have to sync let say only three of them to ES.

One of way getting this is by creating a view in mongodb and relating the view to ES by following the above approach.

other way to do this is with a script, Scripts are run after the results are returned.

[[script]]

Another way which is more efficient but a little more difficult to setup is using an aggregation pipeline which runs at query time. In this case MongoDB would be doing the projection instead of monstache.

Monstache supports aggregation pipelines via the [[pipeline]] config.

[https://rwynn.github.io/monstache-site/config/#pipeline](https://rwynn.github.io/monstache-site/config/)

You would want to construct a pipeline that uses a [project](https://docs.mongodb.com/manual/reference/operator/aggregation/project/) stage.

Note that for change streams this pipeline would be operating on [change events](https://docs.mongodb.com/manual/reference/change-events/) and you would only want to only change the projection of the fullDocument field within that change event.

For more details please read.

<https://github.com/rwynn/monstache/issues/149>

So this time I did it using [[pipeline]] config .

And change config.toml by adding [[pipeline]] just below mapping part.

################################################################

............

exit-after-direct-reads = false

[[mapping]]

namespace = "mydb.test1"

index = "finaltestidx"

[[pipeline]]

namespace = "mydb.test1"

script = """

module.exports = function(ns, changeStream) {

if (changeStream) {

return [{$project: {

\_id: 1,

operationType: 1,

documentKey:1,

"fullDocument.\_id": 1,

"fullDocument.sku\_code": 1,

"fullDocument.sku\_name": 1,

ns:1

}}]

} else {

return []

}

}

"""

###############################################################

5.Only make updates in ES when specific field created,updated or deleted in mongodb.

Expactations: we wanted to listen for create, delete, and update events on the namespace hngdb.product, but we only wanted to sync changes when the sku\_code or sku\_name field changed on the doc.

we can use the following configuration.

If, for example, a field named sku\_description changed on the document, then this change would be ignored by monstache.

[[pipeline]]

namespace = "mydb.test"

script = """

module.exports = function(ns, changeStream) {

if (changeStream) {

return [

{

$match: {

$or: [

{ "updateDescription": {$exists: false} },

{ "updateDescription.updatedFields.sku\_code": {$exists: true}},

{ "updateDescription.updatedFields.sku\_name": {$exists: true}}

]

}

}

];

} else {

return [];

}

}

"""

6.Lets Dokerize .

Expectations:Run monstache as a container.

Docker image of [monstache](https://hub.docker.com/r/rwynn/monstache) is available

Docker Pull Command

docker pull rwynn/monstache

Here I encounterd error due to docker networking. But figured out by going through docs.

I run mongodb ,Elasticsearch,Kiabana and monstache as container it was working as expected.

Docker-compose-file I am using

####################################################

version: "2.1"

services:

mongodb1:

image: mongo:latest

restart: always

container\_name: mongodb1

volumes:

- mongodata1:/data/db

expose:

- "27017"

ports:

- 27011:27017

entrypoint: [ "/usr/bin/mongod", "--replSet", "rsmongo", "--bind\_ip\_all"]

networks:

mynetwork:

aliases:

- mongodb

mongodb2:

image: mongo:latest

restart: always

container\_name: mongodb2

volumes:

- mongodata2:/data/db

expose:

- "27017"

ports:

- 27012:27017

entrypoint: [ "/usr/bin/mongod", "--replSet", "rsmongo", "--bind\_ip\_all"]

networks:

mynetwork:

aliases:

- mongodb

mongodb3:

image: mongo:latest

restart: always

container\_name: mongodb3

volumes:

- mongodata3:/data/db

expose:

- "27017"

ports:

- 27013:27017

entrypoint: [ "/usr/bin/mongod", "--replSet", "rsmongo", "--bind\_ip\_all" ]

networks:

mynetwork:

aliases:

- mongodb

elasticsearch:

restart: always

image: docker.elastic.co/elasticsearch/elasticsearch:6.8.23

ports:

- 9200:9200

- 9300:9300

depends\_on:

- mongodb

environment:

- ES\_JAVA\_OPTS=-Xms2048m -Xmx2048m

- discovery.type=single-node

healthcheck:

test: "wget -q -O - http://localhost:9200/\_cat/health"

interval: 1s

timeout: 30s

retries: 300

ulimits:

memlock:

soft: -1

hard: -1

networks:

mynetwork:

aliases:

- elasticsearch

kibana:

restart: always

image: docker.elastic.co/kibana/kibana:6.8.23

expose:

- "5601"

ports:

- 5601:5601

depends\_on:

- elasticsearch

environment:

- SERVER\_PORT=5601

- SERVER\_NAME=kibana.example.org

- ELASTICSEARCH\_HOSTS=http://elasticsearch:9200

networks:

mynetwork:

aliases:

- kibana

monstache:

image: rwynn/monstache:5.7.7

container\_name: monstache\_container

working\_dir: /monstache

command: -f ./config.toml

volumes:

- ./monstache:/monstache/

ports:

- "8080:8080"

restart: always

links:

- elasticsearch

networks:

mynetwork:

aliases:

- monstache

networks:

mynetwork:

volumes:

mongodata1:

mongodata2:

mongodata3:

#################################################

Initially I used all latest Images of ES,Kibana,monstache and mongo.

Above compose is used for working with ES 6.8 for working on some specific setting and mapping of index which was earlier used in ES 6.x.

7.Create Index in ES with given setting and mapping and sync mongodb collection with this index.

Expectations: already created index has fields with different name like mongodb collection has field sku\_name and ES index has name as skuName so I have to do appropriate mapping of these fields .

So I was provided a setting and mapping of index.while creating the index the first problem here I face is missing [synonym.txt](https://stackoverflow.com/questions/61227043/cannot-load-and-use-synonym-txt-file-on-elasticsearch) error. we are using synonym.txt with ES in production .So it was missing in my machine so I remove the synonym part[this file was used for full text search feature ].

Then mapper\_parsing\_exception error .so I downgrade my ES from 7.16 to 6.8.23.

[https://www.elastic.co/guide/en/elasticsearch/reference/7.16/removal-of-types.html](mapper_parsing_exception)

So for this mapping I created a aggregation pipeline in mongodb

[{$project: {

\_id: 1,

chemical\_free: 'fullDocument.$chemicalFree',

sku\_name: '$skuName',

beauty\_tip: '$beautyTip',

brand\_code: '$brandCode',

sku\_description: '$skuDescription',

skin\_type: '$skinType',

skin\_tone: '$skinTone'

}}]

And created a view in mongodb and replicate the view with ES index.

Config.toml for this sync.

mongo-url = "mongodb://mongodb1:27017"

elasticsearch-urls = ["http://elasticsearch:9200"]

elasticsearch-max-conns = 10

replay = false

resume = true

enable-oplog = true

resume-name = "producttest"

change-stream-namespaces = ["mydb.product"]

direct-read-namespaces = ["mydb.test3"]

index-as-update = true

verbose = true

exit-after-direct-reads = false

[[mapping]]

namespace = "mydb.test3"

index = "test3"

type = "product"

[[relate]]

namespace = "mydb.product"

with-namespace = "mydb.test3"

keep-src = false