# **Topcoder - Add Endpoints To Populating Marathon Matches And SRMs Into ElasticSearch**

# version 1.0

**Deployment And Verification Guide**

# Deployment

Please follow the instructions below to deploy the needed services to verify the submission :

* **ElasticSearch** :

Follow setup section in docs/Verification\_Topcoder - Elasticsearch Feeder Service For Challenges version 1.0 to use AWS based ElasticSearch.

For local ElasticSearch, we can use a local ElasticSearch docker container like the one used in

<https://github.com/topcoder-platform/challenges-logstash-conf/blob/dev/docker-compose.yml>

...

elasticsearch:

image: elasticsearch:2.3

ports:

- 9200:9200

- 9300:9300

...

* **Informix database :**

TopCoder Informix database docker container can be used as in https://github.com/appirio-tech/tc-common-tutorials/blob/master/docker/direct-app/docker-compose.yml

tc-informix:

image: "appiriodevops/tc-database-scripts:latest"

hostname: cockpit.cloud.topcoder.com

container\_name: iif\_innovator\_c

environment:

LICENSE: accept

ports:

- "2021:2021"

- "2022:2022"

- "27017:27017"

- "27018:27018"

- "27883:27883"

tty: true

...

For convenience, a docker-compose.yml is provided. Check submission/docker-compose.yml

version: '2'  
services:  
 elasticsearch:  
 image: elasticsearch:2.3  
 ports:  
 - 9200:9200  
 - 9300:9300  
 tc-informix:  
 image: "appiriodevops/tc-database-scripts:latest"  
 hostname: cockpit.cloud.topcoder.com  
 container\_name: iif\_innovator\_c  
 environment:   
 LICENSE: accept  
 ports:  
 - "2021:2021"  
 - "2022:2022"  
 - "27017:27017"  
 - "27018:27018"  
 - "27883:27883"  
 tty: true

Please map your DOCKER\_IP to cockpit.cloud.topcoder.com in your hosts file, [https://github.com/appirio-tech/tc-common-tutorials/tree/master/docker/direct-app#verificaition](https://github.com/appirio-tech/tc-common-tutorials/tree/master/docker/direct-app" \l "verificaition)

By default, the tc-elasticsearch-feeder-service uses cockpit.cloud.topcoder.com as a base hostname in src/main/resources/elasticsearch-feeder-service.yaml.

In command line application, go to submission directory and run the needed service containers

$ docker-compose up

We can also run each of those services in separate command line terminal

$ docker-compose up elasticsearch

Open another terminal

$ docker-compose up tc-informix

# Data Setup

The TC Informix database already contained test data for Marathon Matches (MM) and Single Round Matches (SRM)

[https://github.com/topcoder-platform/tc-database-scripts/blob/dev/informixoltp/05\_informixoltp\_test\_data.sql#L486-L755](https://github.com/topcoder-platform/tc-database-scripts/blob/dev/informixoltp/05_informixoltp_test_data.sql" \l "L486-L755)

There are two Marathon matches:

- roundId = 13673

- roundId = 13675

There are two SRMs:

- roundId = 13672

- roundId = 13676

# Verification

The tc-elasticsearch-feeder-service that in use is

repository: https://github.com/topcoder-platform/tc-elasticsearch-feeder-service

branch: dev

commit: cfc98832aa10b35c229107f49b7df992a3ac189a

In command line app, go to submission/ tc-elasticsearch-feeder-service

Build the microservice package by run command

$ mvn clean compile package

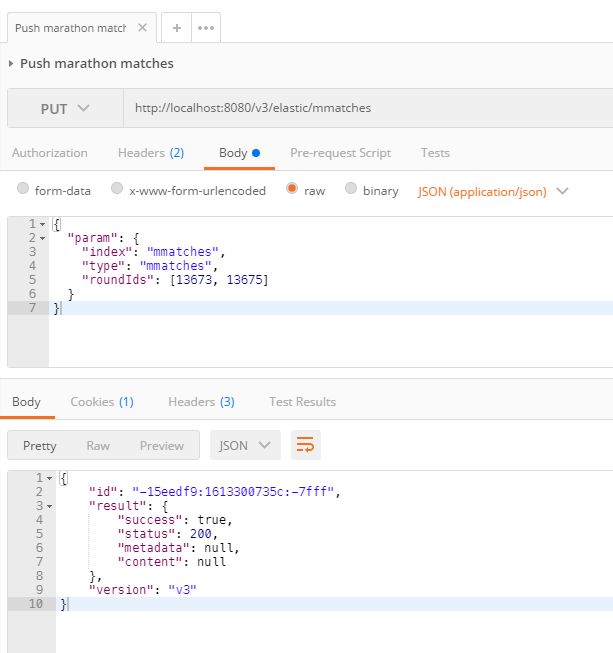
Once build is done, go to submission/tc-elasticsearch-feeder-service/local and run the service

$ ./run.sh

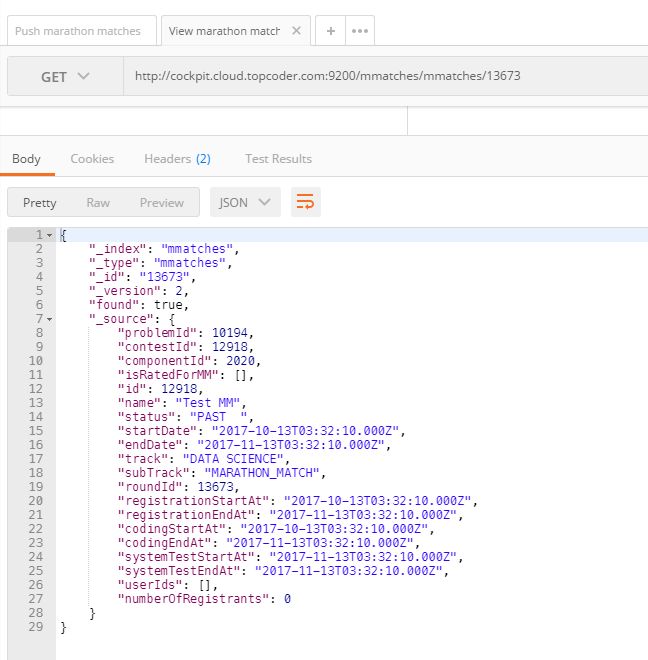
After the microservice started, open Postman and import docs/mm-srm-feeder.postman\_collection.json

**Marathon Match**

In Postman, select “Push marathon matches” from marathon-match-feeder folder in left pane.



View marathon match via ElasticSearch



As we can see in the picture above, the MM does not have registrants.

Connect to Informix and execute the following statements on informix\_oltp database to create some registrants for the marathon match with roundId 13673

DATABASE informixoltp;

-- 124766 - twight

-- 124834 - lightspeed

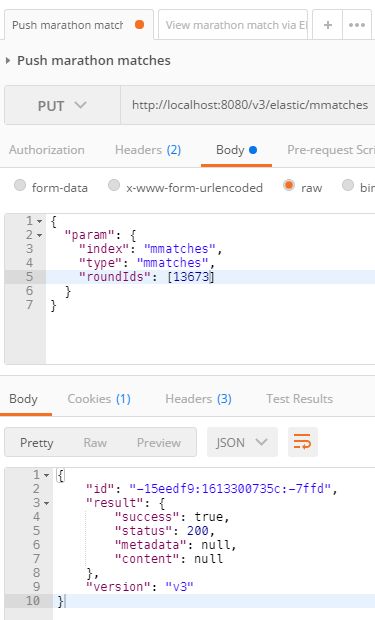
INSERT INTO round\_registration (round\_id, coder\_id) VALUES (13673, 124766);

INSERT INTO round\_registration (round\_id, coder\_id) VALUES (13673, 124834);

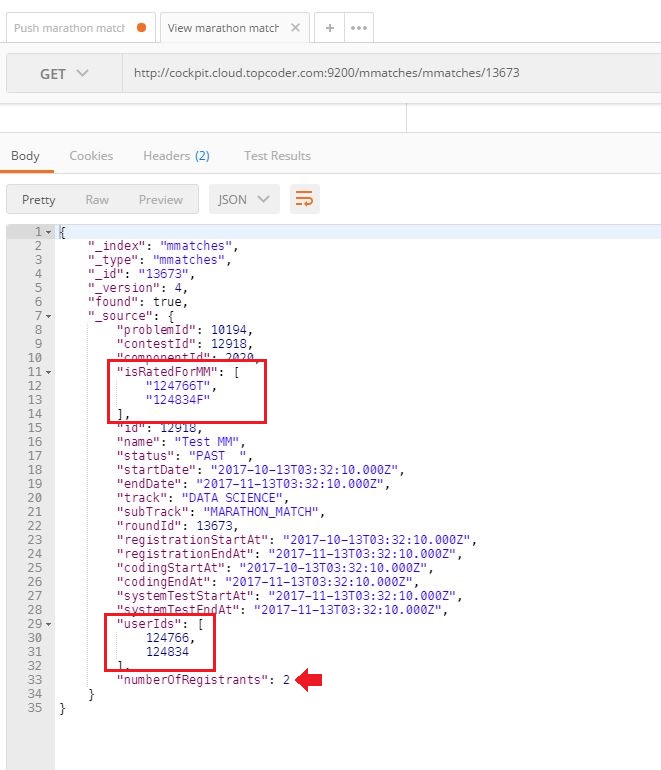
INSERT INTO long\_comp\_result (round\_id, coder\_id, rated\_ind) VALUES (13673, 124766, 1);

INSERT INTO long\_comp\_result (round\_id, coder\_id, rated\_ind) VALUES (13673, 124834, 0);

Re push the marathon match

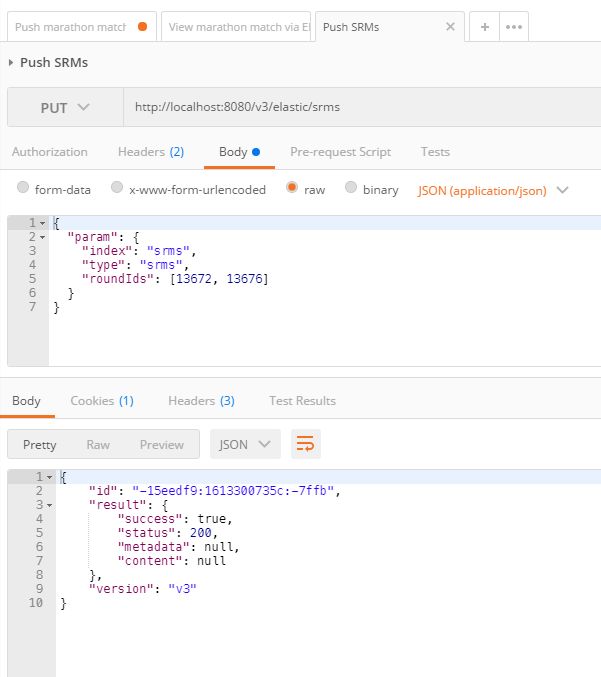


Now, re-send the request in “View marathon match via ElasticSearch” tab

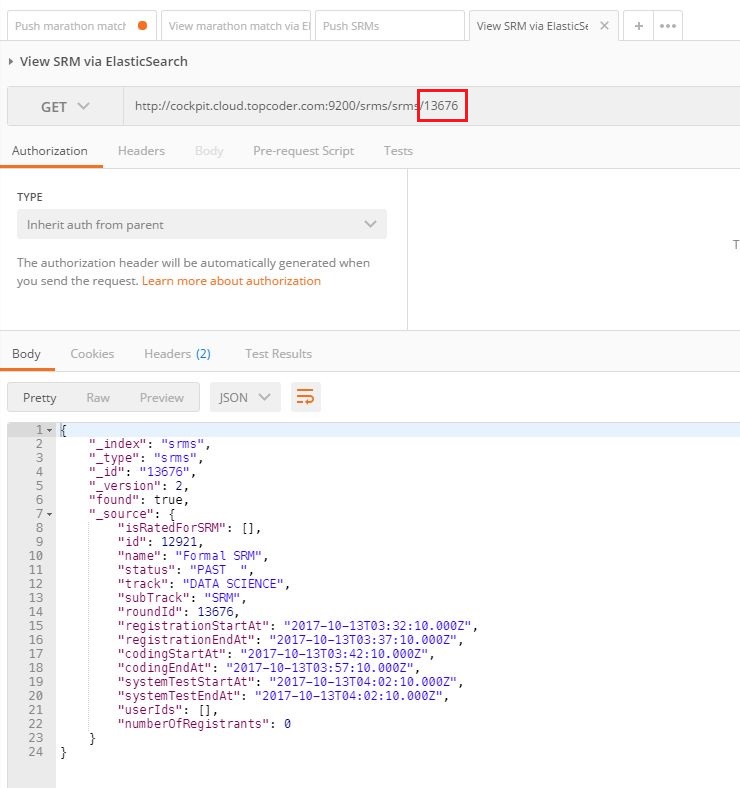


**Single Round Match (SRM)**

In Postman, select “Push SRMs” from srm-feeder folder in left pane.



View SRMs via ElasticSearch



As we can see in the picture above, the SRM does not have registrants.

Connect to Informix and execute the following statements on informix\_oltp database to create some registrants for the SRM with roundId 13672

DATABASE informixoltp;

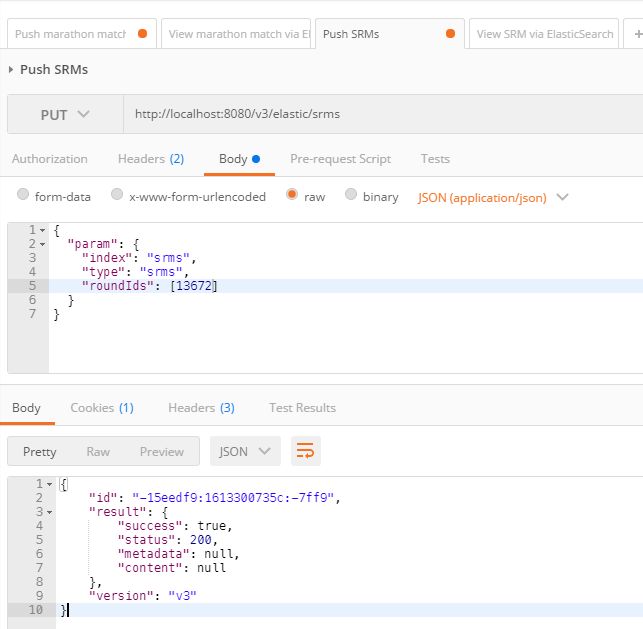
INSERT INTO round\_registration (round\_id, coder\_id) VALUES (13672, 124766);

INSERT INTO round\_registration (round\_id, coder\_id) VALUES (13672, 124834);

INSERT INTO room\_result (round\_id, room\_id, coder\_id, rated\_flag) VALUES (13672, (SELECT room\_id FROM room WHERE round\_id=13672), 124766, 0);

INSERT INTO room\_result (round\_id, room\_id, coder\_id, rated\_flag) VALUES (13672, (SELECT room\_id FROM room WHERE round\_id=13672), 124834, 1);

Re-push the SRM



Now, re-send the request in “View SRM via ElasticSearch” tab

