**Topcoder - Create CronJob For Populating Marathon Matches and SRMs To Elasticsearch version 1.0**

**Verification Guide**

# 

# Setup

To setup the environment it is necessary to navigate to source folder and type

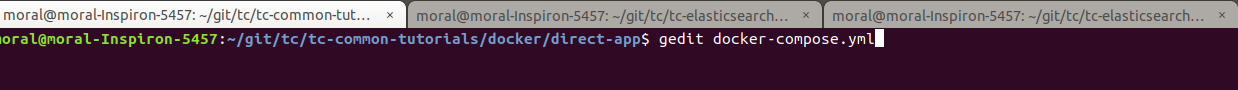
**mvn clean package**

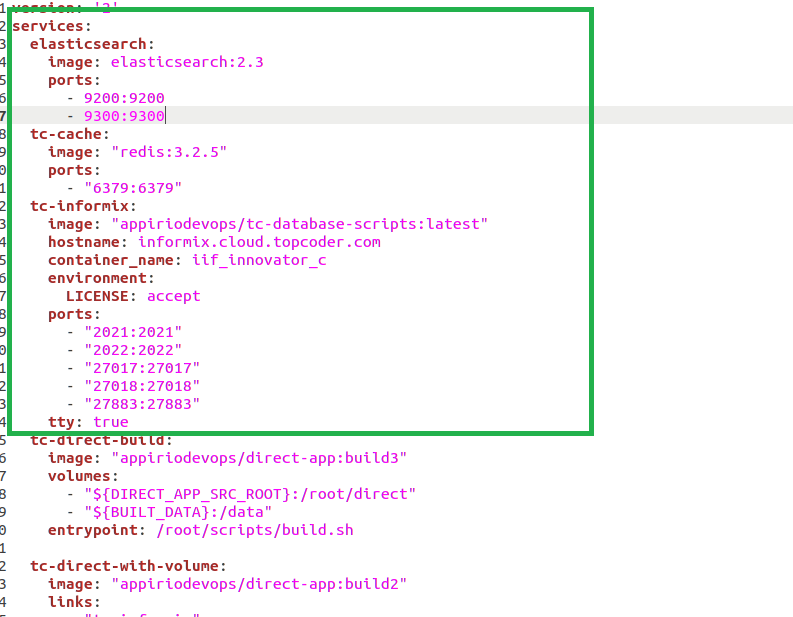
This will compile and create the jar file under “*target*” folder.

To create new marathon matches and single round matches it is necessary to run the tc-direct application. A docker image is provided in

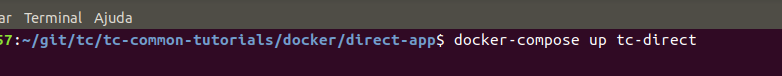
<https://github.com/appirio-tech/tc-common-tutorials/tree/master/docker/direct-app#build-and-run-with-docker-compose>

Go to folder “*tc-common-tutorials/docker/direct-app*” and edit file *docker\_compose.yml* as following to add the *elasticsearch* image and start all necessary services at once.





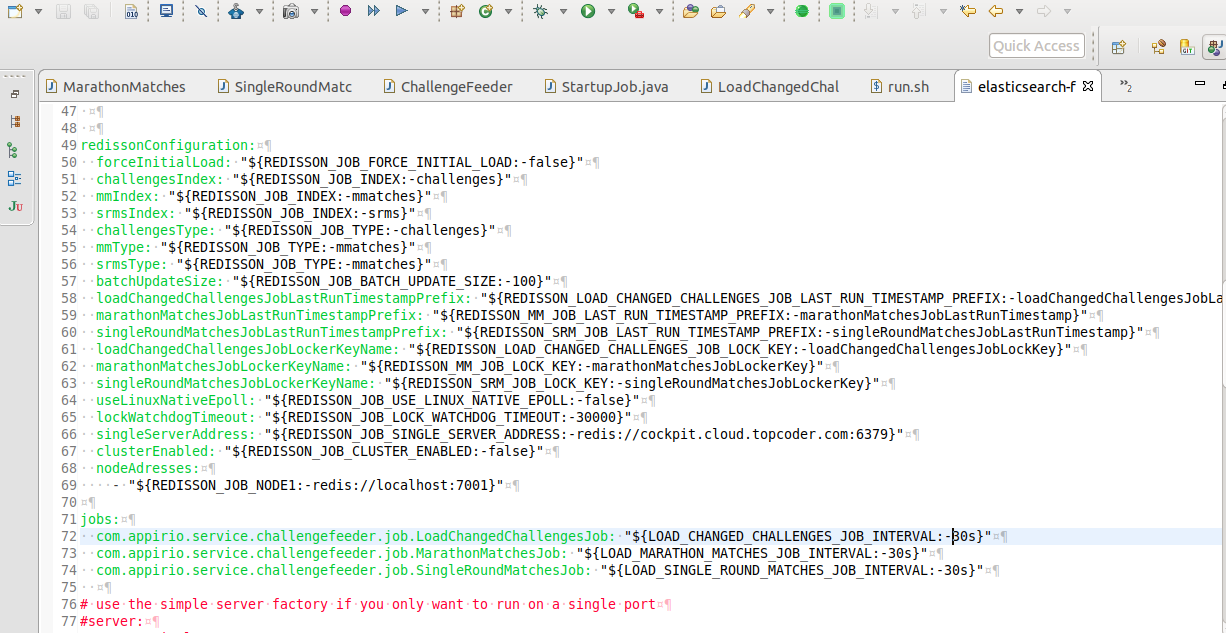
Then run



Then tc-direct, elasticsearch and tc-cache (redis) will be running.

For more detailed instructions on how to create input data (mm and srm) please check “docs/Verification\_Topcoder - Elasticsearch Feeder Service For Challenges version 1.0.docx”.

The feeder service and its related jobs use the variable defined in file *src/main/resources/challenge-feeder-service.yaml*, all the config values can be overridden by environment variables.

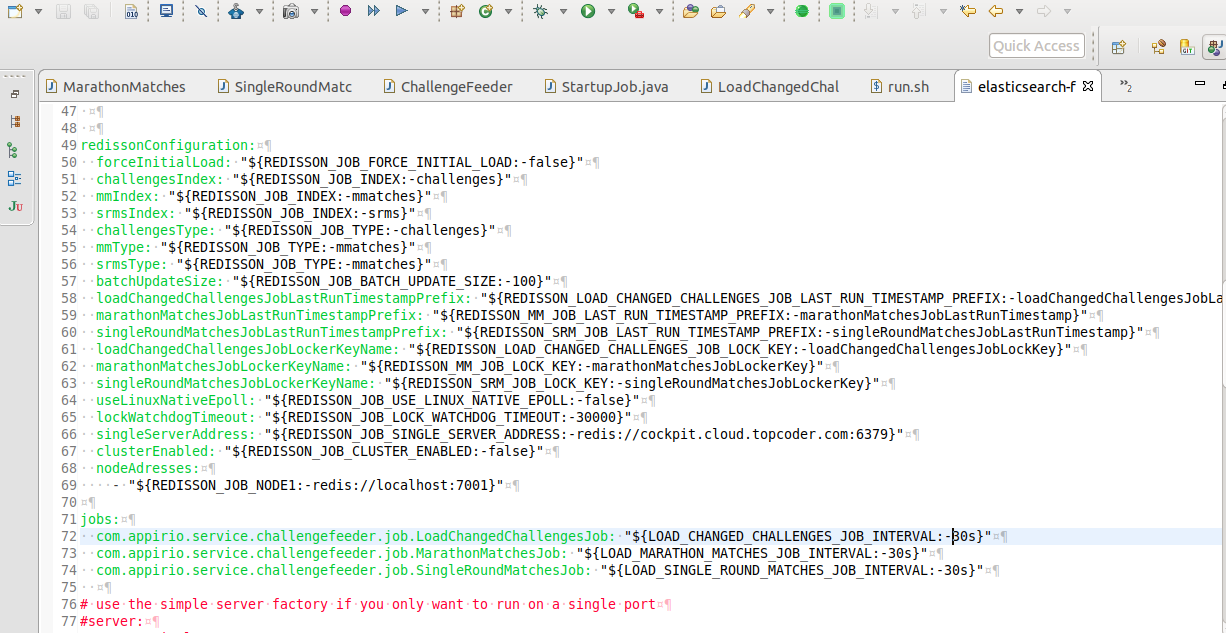


In the current challenge some environment variables were renamed to be more generic and others were added and are related to the new jobs.

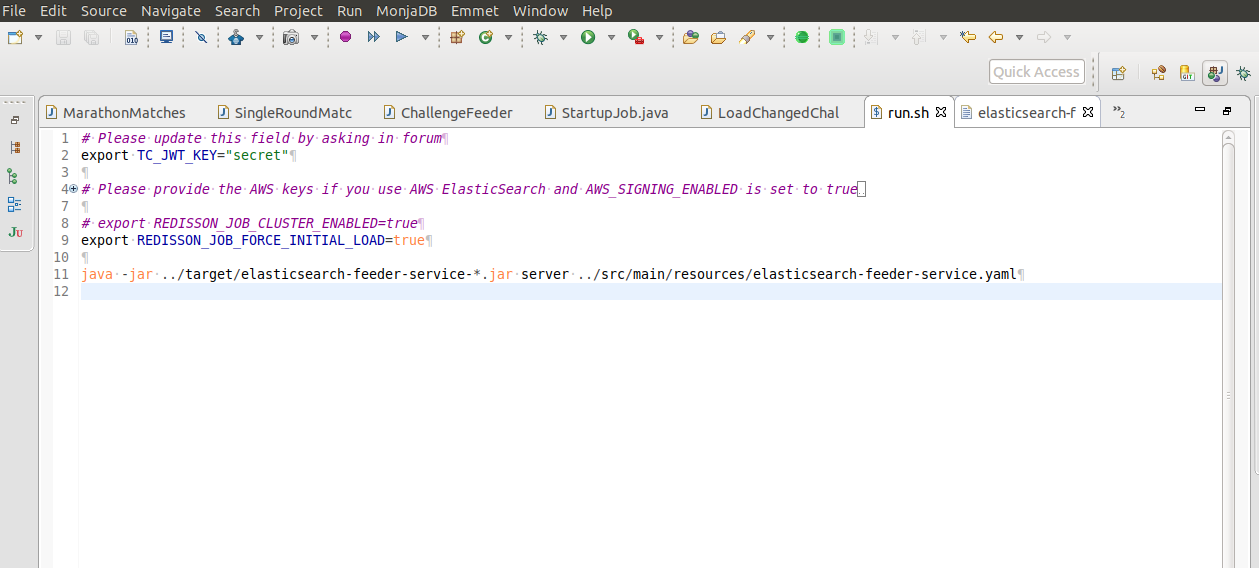
The application will be started by default on port 8080. This can be changed in file *src/main/resources/challenge-feeder-service.yaml.*

Current challenge added variables:

* marathonMatchesJobLastRunTimestampPrefix
  + This will set the index entry prefix while managing las execution timestamp for job ***MarathonMatchesJob***. It can be overridden by env variable *REDISSON\_MM\_JOB\_LAST\_RUN\_TIMESTAMP\_PREFIX*.
* singleRoundMatchesJobLastRunTimestampPrefix
  + This will set the index entry prefix while managing las execution timestamp for job ***SingleRoundMatchesJob***. It can be overridden by env variable *REDISSON\_SRM\_JOB\_LAST\_RUN\_TIMESTAMP\_PREFIX*.
* marathonMatchesJobLockerKeyName
  + This is the index entry for managing the lock for distributed execution of job ***MarathonMatchesJob***. It can be overridden by env variable *REDISSON\_MM\_JOB\_LOCK\_KEY*.
* singleRoundMatchesJobLockerKeyName
  + This is the index entry for managing the lock for distributed execution of job ***SingleRoundMatchesJob***. It can be overridden by env variable *REDISSON\_SRM\_JOB\_LOCK\_KEY*.
* com.appirio.service.challengefeeder.job.MarathonMatchesJob
  + This property sets ***MarathonMatchesJob*** execution interval. By default it is set to 30 seconds. It can be overridden by env variable *LOAD\_MARATHON\_MATCHES\_JOB\_INTERVAL*.
* com.appirio.service.challengefeeder.job.SingleRoundMatchesJob
  + This property sets ***SingleRoundMatchesJob*** execution interval. By default it is set to 30 seconds. It can be overridden by env variable *LOAD\_SINGLE\_ROUND\_MATCHES\_JOB\_INTERVAL*.

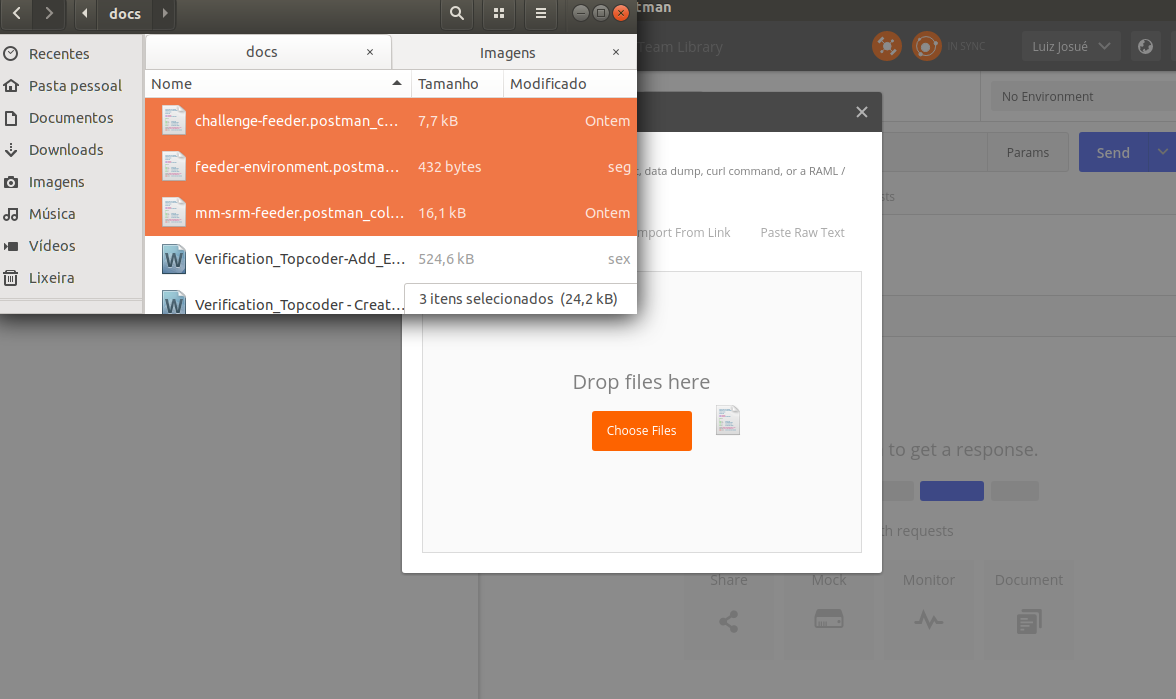


In order to start the application, navigate to “*local*” folder and edit file *run.sh* as follow:

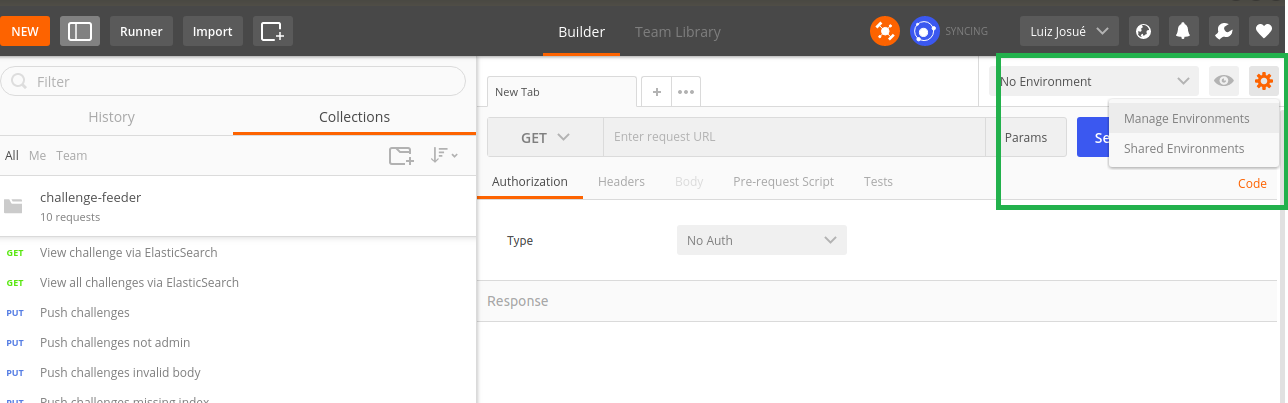


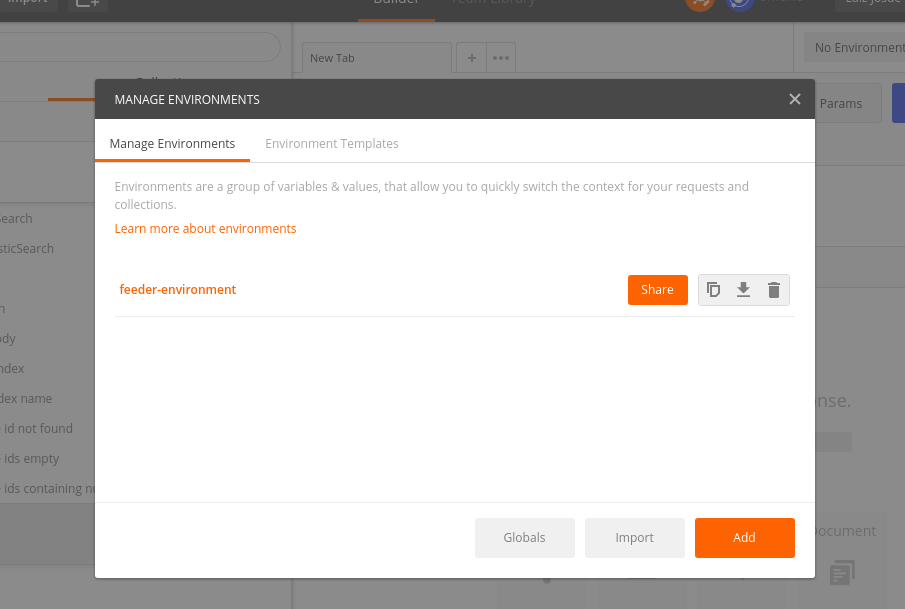
There you will find variables *REDISSON\_JOB\_CLUSTER\_ENABLED* and *REDISSON\_JOB\_FORCE\_INITIAL\_LOAD*. The first one tells the application to use redis in clustered or standalone mode. By default the application runs in ***standalone*** mode. The variable *REDISSON\_JOB\_FORCE\_INITIAL\_LOAD* tells the application to perform a full load at the first time the jobs are executed. By default it is set to ***false*** but in run.sh it is set to true to make the current challenge reviewing process easier.

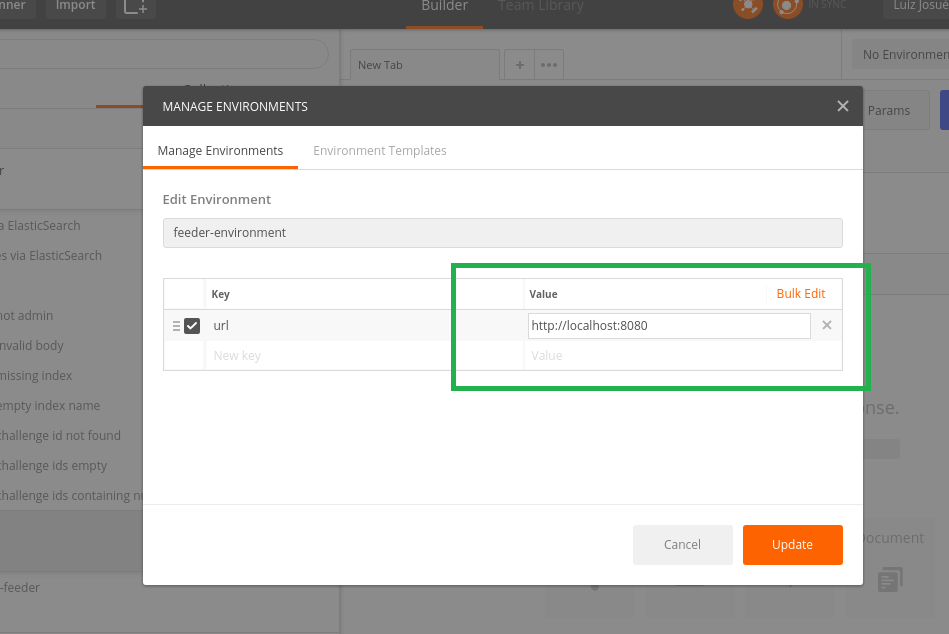
In order to verify the data loaded into elasticsearch it is necessary to use the Postman files in docs folder: *challenge-feeder.postman\_collection.json, mm-srm-feeder.postman\_collection.json and feeder-environment.postman\_environment.json*. Import them:



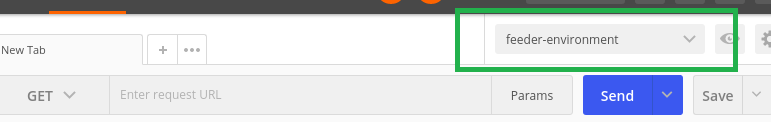
Configure the feeder-environment by clicking on “Manage environments” as follows:







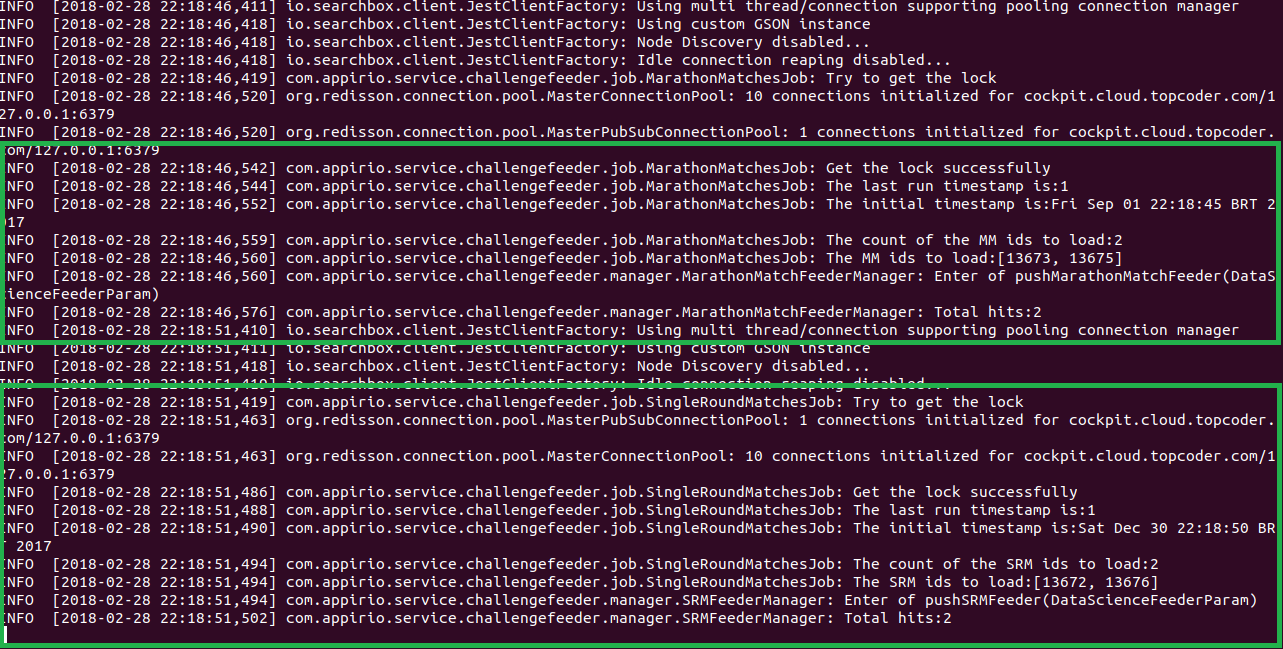
And then select it:



You can check that the indices are empty before running application. Hit send for any of the postman requests: “List all MMs via ElasticSearch” or “List all SRMs via ElasticSearch” and check it.

# Verify

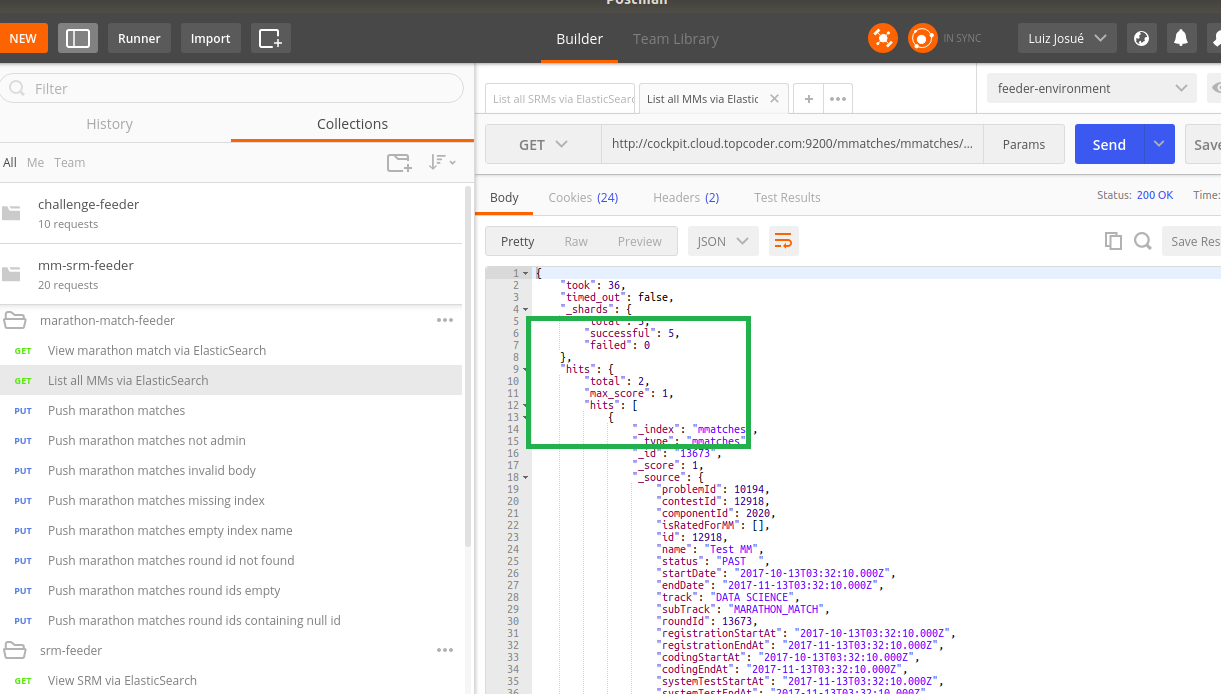
After running *“local/run.sh”* the jobs will then start and you can check console output.

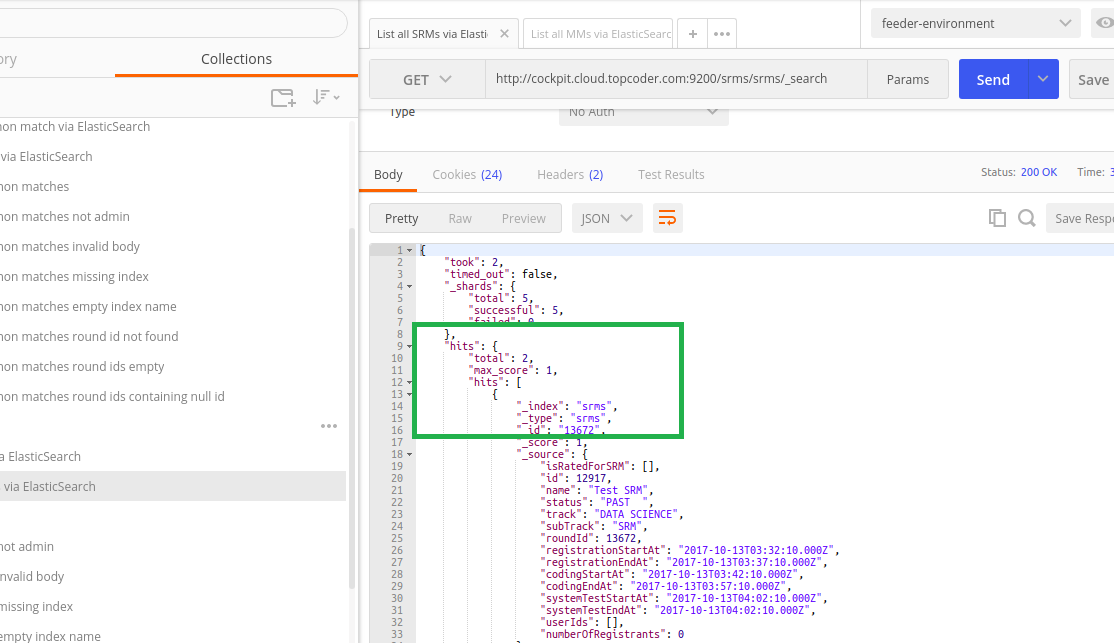


The above picture shows that ***MarathonMatchesJob*** found two rounds (ids: 13673 and 13675) and then pushed them into elasticsearch index. It also shows that ***SingleRoundMatchesJob*** has found two rounds (ids: 13672 and 13676) and the job will push them into elasticsearch index.

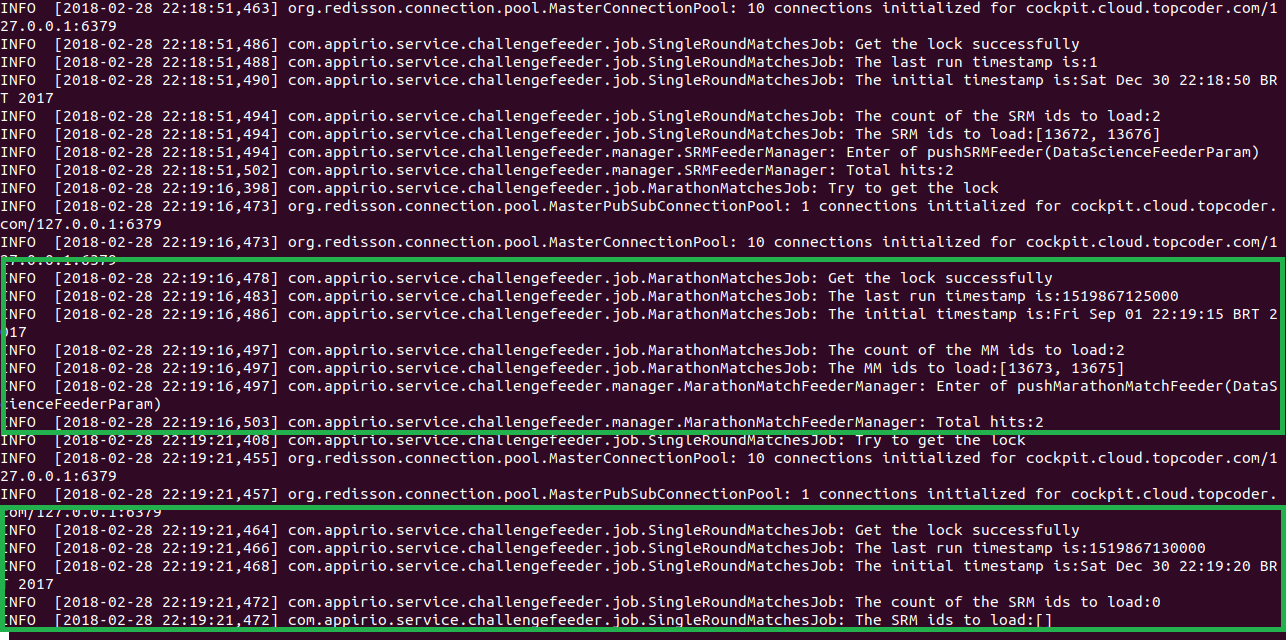
The names of the indices used are “*mmatches”* and “*srms”*. They can be configured in *challenge-feeder-service.yaml* by properties *mmIndex* and *srmsIndex* respectively.

Then check them using the postman requests: “List all MMs via ElasticSearch” or “List all SRMs via ElasticSearch”





You can check in the console output that the jobs are still running:



The ***MarathonMatchesJob*** keeps showing two hits because their dates are more recent than the **180 days** defined in the variable: ***marathonMatchesDaysToSubtract***. You can also see that ***SingleRoundMatchesJob*** does not have any other hit in the last **60 days**.

You can change variables ***marathonMatchesDaysToSubtract*** and ***singleRoundMatchesDaysToSubtract*** to verify that the job is only retrieving the rounds in the last x days specified by these variables.