**Topcoder Elasticsearch Feeder Service - Jobs Cleanup And Improvement version 1.0**

**Verification Guide**

# 

# Setup

Local setup:

1. Following <https://github.com/appirio-tech/tc-common-tutorials/tree/master/docker/direct-app> to run all dependent services(such as mock services)

1.1 you need to build from the latest code of direct app (see <https://github.com/appirio-tech/tc-common-tutorials/tree/master/docker/direct-app#build-and-run-with-docker-compose>)  
1.2 also the elasticsearch service should be up, run like (docker-compose up -d elasticsearch)  
2. Run the elasticsearch feeder service using mm-features branch, you probably should change the ports (8080, 8081 in src/main/resources/elasticsearch-feeder-service.yaml) due to port conflict.  
3. Create some challenges through direct app, so the elasticsearch feeder will get the data populated into elasticsearch and they will be used by challenge service.

When you try to build the docker image from the last code of tc-direct, make sure that you update the last appiriodevops/tc-database-scripts:latest docker, and after the informix database is started, select the tcs\_catalog database run:

select \* from client.

If the enable\_effort\_hours is null, update it to false or true with this sql:

update client set enable\_effort\_hours = 0;

See Verification\_Topcoder - Elasticsearch Feeder Service For Challenges version 1.0.docx about how to create the challenge in the tc-direct app for testing.

select the informixoltp database and run:

update round\_segment set start\_time = current where segment\_id = 1 and round\_id = 13672;

update round\_segment set start\_time = current, end\_time = current + 1000 units minute where segment\_id = 1 and round\_id=13675;

update round set forum\_id = 111 where round\_id = 13675;

delete from round\_prize;

insert into round\_prize values (13675, 1, 200);

insert into round\_prize values (13675, 2, 100);

delete from round\_registration;

insert into round\_registration values (13675, 132456, current, 0, null);

insert into round\_registration values (13675, 132458, current, 0, null);

delete from round\_event;

insert into round\_event values (13675, 1, 'event name 1', 'reg url 1');

insert into round\_event values (13675, 2, 'event name 2', 'reg url 2');

delete from long\_submission;

delete from long\_component\_state;

insert into long\_component\_state values (1, 13675, 132456, 2041, 99, 130, 2, 2);

insert into long\_submission (long\_component\_state\_id, submission\_number, example, open\_time, submit\_time, submission\_points) values (1, 2, 2, 132000, 1517544546857, 99);

delete from long\_comp\_result;

insert into long\_comp\_result (round\_id, coder\_id, placed) values (13675, 132456, 1);

delete from algo\_rating where coder\_id = 132456;

insert into algo\_rating values (132456, 1000, 0, null, 0, 3, current);

See Verification\_Topcoder - Create CronJob For Populating Marathon Matches and SRMs To Elasticsearch 1.0.docx for more details about how to create the the marahon matches and single round match data in the database

Use the redis-cli to insert the data.

you can down load the redis client on https://redis.io/

For windows OS, you can download the Redis-x64-3.2.100.zip here: https://github.com/MicrosoftArchive/redis/releases

open the command line and run

redis-cli -h 192.168.99.100 -p 6379

192.168.99.100 should be the docker IP that holds the redis cache service.

Then run the following commands:

hset com.appirio.service.challengefeeder.job.LoadChangedChallengesListingJob.map.cache "\"challengeslisting.job.enable\"" "\"true\""

hset com.appirio.service.challengefeeder.job.LoadChangedChallengesDetailJob.map.cache "\"challengesdetail.job.enable\"" "\"true\""

hset com.appirio.service.challengefeeder.job.LegacyMMToChallengeListingJob.map.cache "\"challengeslisting.job.enable\"" "\"true\""

hset com.appirio.service.challengefeeder.job.LoadChangedMMChallengeDetailJob.map.cache "\"challengesdetail.job.enable\"" "\"true\""

hset com.appirio.service.challengefeeder.job.MarathonMatchesJob.map.cache "\"mmatches.job.enable\"" "\"true\""

hset com.appirio.service.challengefeeder.job.SingleRoundMatchesJob.map.cache "\"srms.job.enable\"" "\"true\""

There is another easy way to configure the enable job flag, just copy the following code to src\main\java\com\appirio\service\challengefeeder\job\BaseJob.java and run the main method in your IDE

**public** **static** **void** main(String[] args) **throws** Exception {

Config redissonConfig = **new** Config();

redissonConfig.useSingleServer().setAddress("redis://192.168.99.100:6379");

RedissonClient client = Redisson.*create*(redissonConfig);

client.getMap("com.appirio.service.challengefeeder.job.LoadChangedChallengesListingJob.map.cache").put("challengeslisting.job.enable", "true");

client.getMap("com.appirio.service.challengefeeder.job.LoadChangedChallengesDetailJob.map.cache").put("challengesdetail.job.enable", "true");

client.getMap("com.appirio.service.challengefeeder.job.LegacyMMToChallengeListingJob.map.cache").put("challengeslisting.job.enable", "true");

client.getMap("com.appirio.service.challengefeeder.job.LoadChangedMMChallengeDetailJob.map.cache").put("challengesdetail.job.enable", "true");

client.getMap("com.appirio.service.challengefeeder.job.MarathonMatchesJob.map.cache").put("mmatches.job.enable", "true");

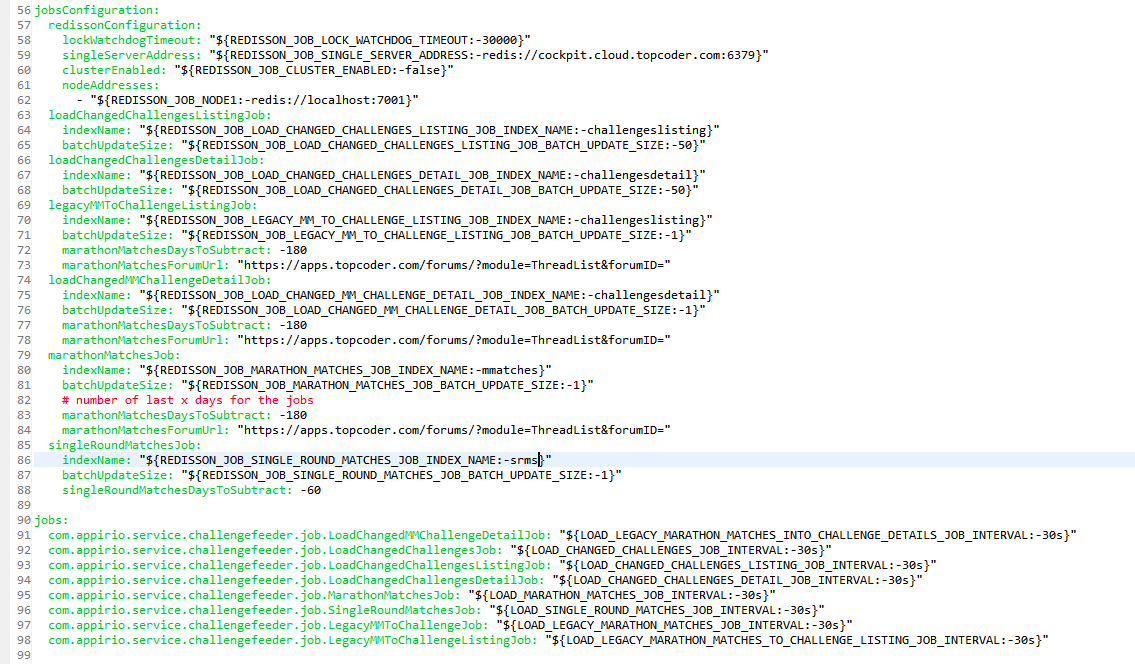
client.getMap("com.appirio.service.challengefeeder.job.SingleRoundMatchesJob.map.cache").put("srms.job.enable", "true");

client.shutdown();

}

192.168.99.100 should be the docker IP that holds the redis cache service

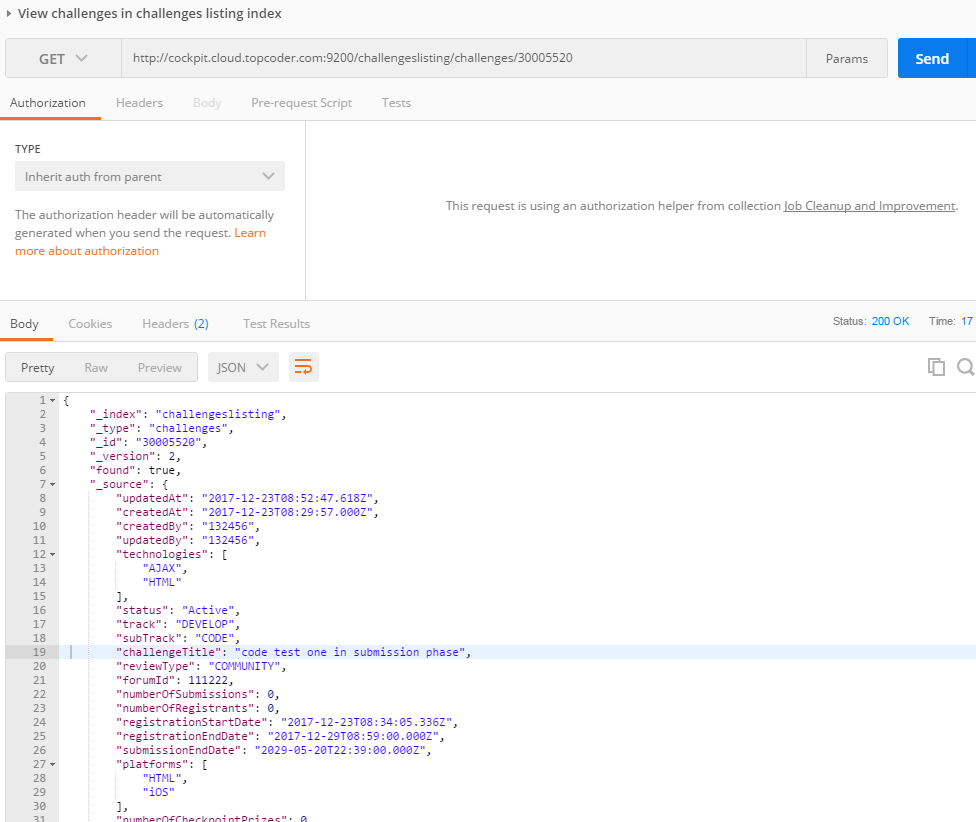
The configuration values for jobs is(src\main\resources\elasticsearch-feeder-service.yaml):



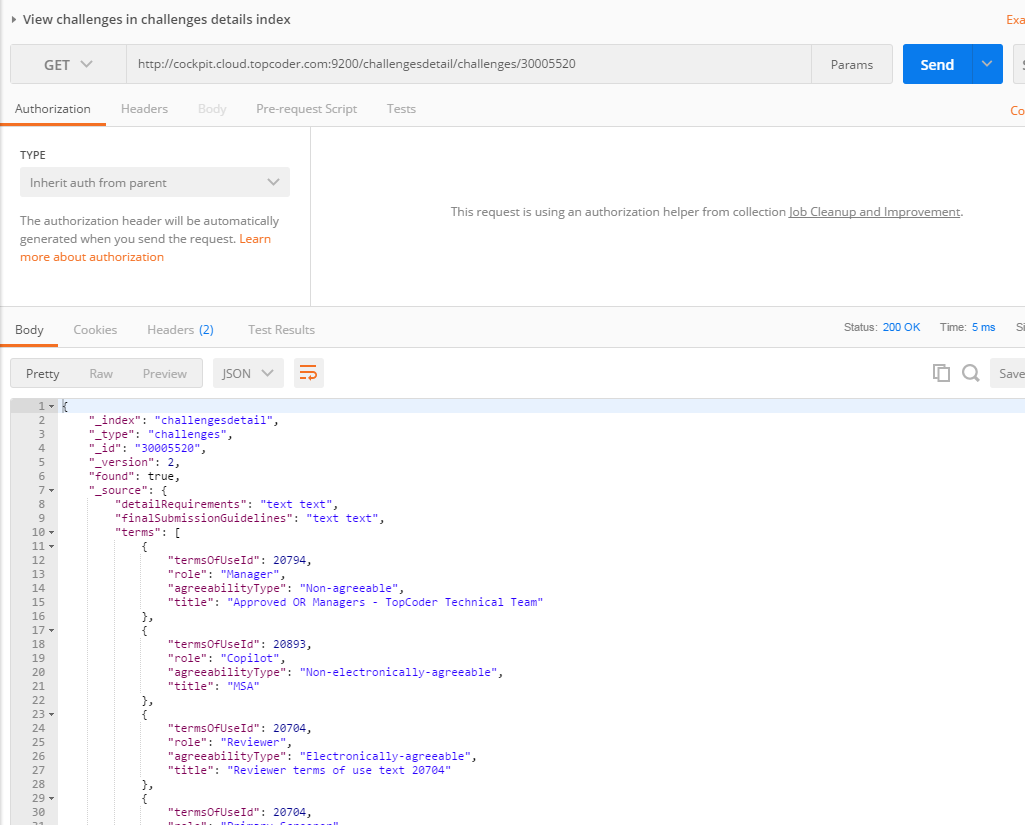
# Verify

Import the Job Cleanup and Improvement.postman\_collection.json into the postman.

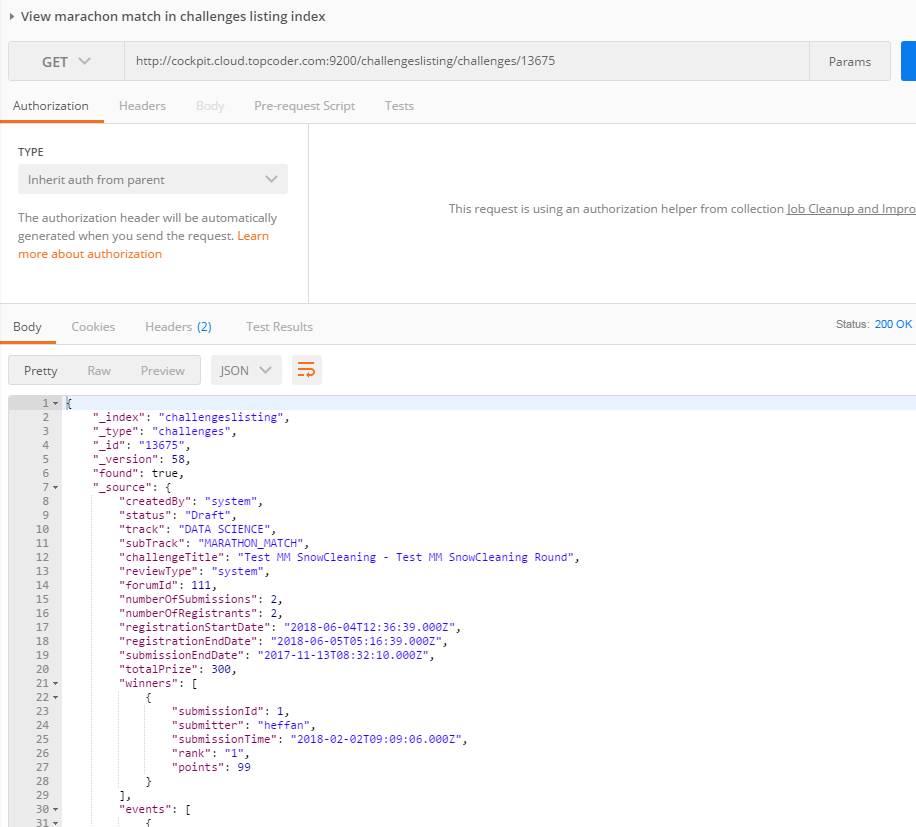
View challenges in challenges listing index



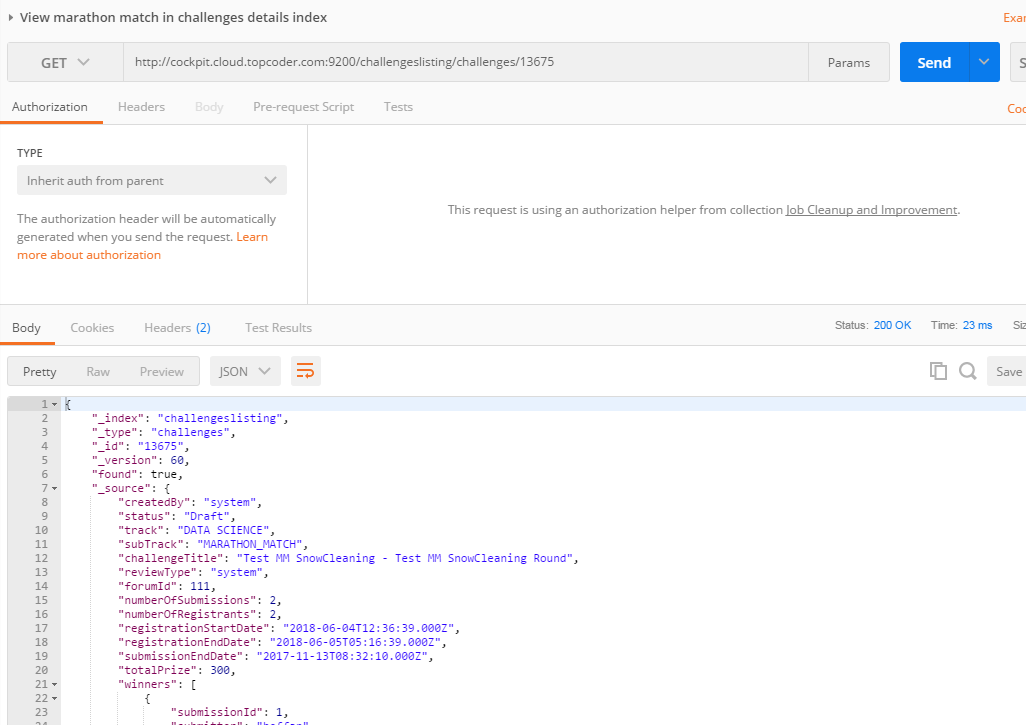
View challenges in challenges details index



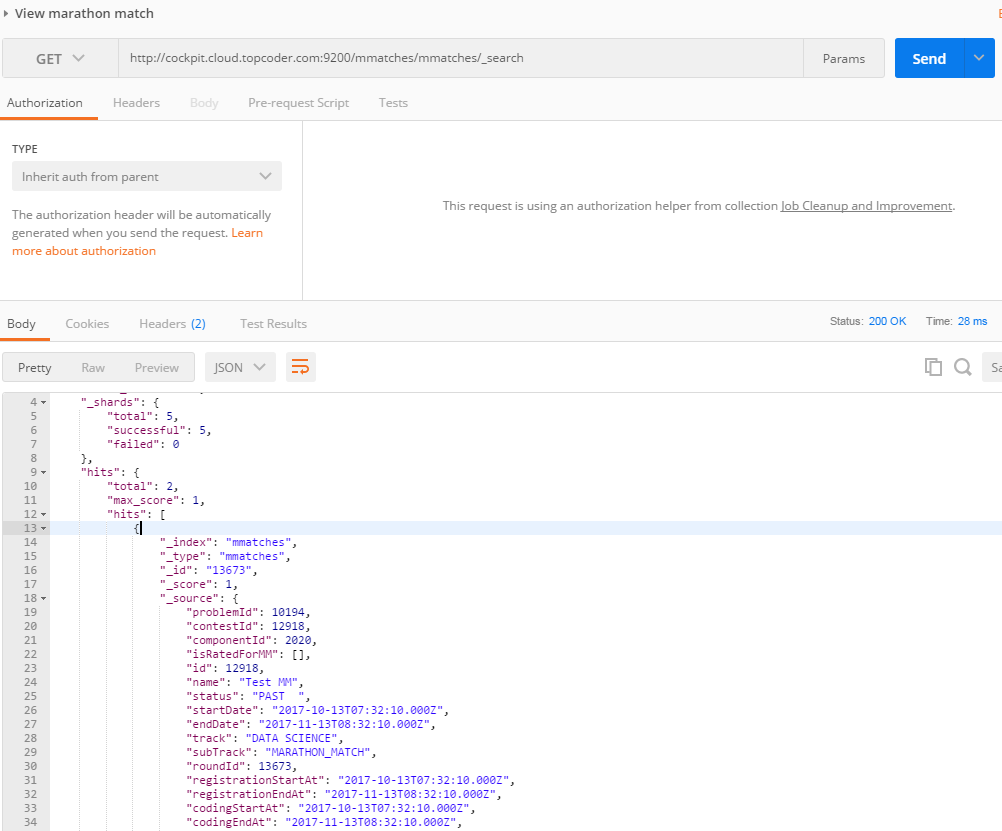
View marachon match in challenges listing index



View marathon match in challenges details index



View marathon match



View single round match

