**Etl process: py script**

### Setup in cron ###  
**List cron**: crontab -l  
**Entry in cron**: 0 \* \* \* \* . /opt/app/etl\_job/etl\_job.sh   
(wil run hourly.)  
**Update scheduler as required in cron** : crontab -e  
**Checking cron log**: sudo cat /var/log/cron

### etl script ###  
**Location of deploy file**: /opt/app/etl\_job

**Command to run etl script**: python script.py

**Configuration file**: config.ini

**High level steps** ---

A. Fetch data from MySQL ‘Master’ table.  
B. Convert records into elasticsearch input..

C. Push data into ES.

D. Insert value into log table.

**New Table information** ---

**Table Name**: etl\_log

**Configurations:**

**File name**: config.ini

**File Information:**

[mysqlDB]

host = host name to connect mysql DB

db = database name

user = username

pass = user password

table\_name = Table to fetch data from

[elasticSearch]

host= ES IP

index= name of the index to push data into.

**Logging** --- Logging would be maintained in MySQL table ‘etl\_log’.

**Incremental Load**: ETL job will refer to the log table to find new inserted rows in the table by finding last job run datetime. It will then check in tables to all the datetime columns to find new rows to insert into ES.

**Full Load**: To full load, truncate the log table i.e etl\_log and run the script (script.py).

**Java Spring MVC backend service**

**Location of deploy file:**

/opt/app/java\_service/es-util.jar

**Configuration file:**

/opt/app/java\_service/application.properties

**Log file location:** /opt/logs/es-util.log

**Command to run:**

nohup java -jar /opt/app/java\_service/es-util.jar --spring.config.location=file:///opt/app/java\_service/application.properties >/opt/logs/es-util.log &

**NOTE:** THIS SHOULD BE DEPLOYED ONLY ONCE SINCE IT IS A SERVICE UNLESS SERVER IS RESTARTED

**High Level steps:**

a. Get records based on search term from elasticsearch

b. Comparison for aggregation - percMatch, store diff, weight match

c. Refactor response based on above aggregation

**Configurations in application.properties:**

elasticsearch.pmatch.threshold=65.0D ~ (%match for product\_name)

consider.weight.aggregation=false ~ (should weight be matched in aggregation)

consider.diffstore.aggregation=true ~ (should store be different for aggregation)

elasticsearch.index.name=fiverr\_products ~(index name in elastisearch)

elasticsearch.server.host=XXXX ~ (hostname for elasticsearch)

elasticsearch.server.port=9200 ~ (port for elasticsearch)

elasticsearch.fetch.size=5000 ~ (fetch size of records from elasticsearch - max 10000 allowed)

server.port=8081 ~ (port for API application)

**NOTE:** ONCE UPDATED, KILL THE EXISTING PROCESS AND RUN AGAIN BY ABOVE COMMAND

API -

POST

http://XXXX:8081/es-service/api/product

BODY

{

"text": "filtered skimmed milk"

}