Ingest data from a relational database into Elastic

This guide shows how to use the Logstash JDBC input plugin to import data from a relational database into Elastic Application. It shows how Logstash can be used to efficiently copy records and receive updates from a relational database, and then deliver them to Elasticsearch.

MySQL has been used to test the code and procedures described here. They need to be compatible with other relational databases.

You can use the Logstash Java Database Connectivity (JDBC) input plugin to pull data from a variety of relational databases, including MySQL and Postgres or MariaDB if you want to go Opens-Source. The JDBC input plugin, in theory, runs a loop that periodically polls the relational database for entries that have been inserted or updated since the loop's previous iteration.

#### Prerequisites

For this tutorial, you need a source MySQL instance for Logstash to read from. A free version of MySQL is available from the [MySQL Community Server section](https://dev.mysql.com/downloads/mysql/) of the MySQL Community Downloads site. In addition, you need an Elastic deployment (on premise or cloud it does not matter). I used an already installed elk instance (on premise). The elk instance consist in Logstash, Elasticsearch and Kibana. So yes the first step is to have an already elk cluster or if you do not have it, install it from the official elastic.co documentation.

Get the JDBC driver for MySQL. There are no database connection drivers included in the Logstash JDBC input plugin. For the tasks in the next section, you will need a JDBC driver for your relational database. Use the JDBC input plugin to set up a Logstash pipeline.

Download and unpack the JDBC driver for MySQL from the MySQL Community Downloads site's [Connector/J section](https://dev.mysql.com/downloads/connector/j/) (link on click).Make a note of the driver's location, since it will be required in the following phases.

!!! Disclaimer:

This demonstration is done only for **TESTING PURPOSE**. OTHER SETTING AND CONFIGURATIONS MAY BE APPLIED FOR A PRODUCTION ENVIRONMENT. THIS DOCUMENTATION **WILL NOT COVER A PRODUCTION INSTALATION INSTRUCTIONS** AND WILL NOT BE RESPONSIBLE FOR IT IF IT IS DONE.

# *Prepare a MYSQL database*

For this purpose, I deployed in Azure a VM with Ubuntu 20.04 serving as Operating System. So, the steps for installing MYSQL were:

1. sudo apt update (update the operating system);
2. sudo apt install mysql-server (This will install MySQL, but will not prompt you to set a password or make any other configuration changes.) ;

# *Configure MYSQL DB:*

1. sudo mysql\_secure\_installation (This will take you through a series of prompts where you can make some changes to your MySQL installation’s security options. The first prompt will ask whether you’d like to set up the Validate Password Plugin, which can be used to test the password strength of new MySQL users before deeming them valid.) ;
2. Connect to the DB with mysql -u root –p and the password you choose ;
3. Create a user to connect remotely to the ELK Instance. (!!! Notice, ELK installation is not provided here, you can find how to install it on [elatic.co](https://www.elastic.co/guide/en/elastic-stack/current/index.html))

Following command will create a user with **full privileges,** this is **NOT** recommended for PRODUCTION ENVIRONMENT, this was done only for testing purposes.

1. If you followed step 2, you are logged in DB, if not followed step 2, please DO:
2. CREATE USER 'newuser'@'host' IDENTIFIED BY 'password';

* Replace newuser with you user;
* Replace host with the elk instance IP ( or where you installed Logstash );
* Replace password with a proper password

1. GRANT ALL PRIVILEGES ON \* . \* TO 'newuser'@'host';

* Replace newuser with the user created.
* Replace host with the elk instance IP(or where you installed Logstash );

**While this is helpful for explaining some MySQL concepts, it may be impractical for most use cases and could put your database’s security at high risk.**

* FLUSH PRIVILEGES; (Once you have finalized the permissions that you want to set up for your new users, always be sure to reload all the privileges.)
* You can check if your user is created with this command: SELECT User, Host FROM mysql.user;

Text

Description automatically generated

# *Prepare a source MYSQL DB*

Let's have a look at a simple database that you'll use to import data and transmit to Elastic instance. A MySQL database with timestamped records is used in this example. The timestamps make it simple to figure out what has changed in the database since the last data transfer to ELK instance.

For this example, let’s create a new database es\_db with table es\_table, as the source of our Elasticsearch data.

1. **Run the following SQL statement:**

CREATE DATABASE es\_db;

USE es\_db;

DROP TABLE IF EXISTS es\_table;

CREATE TABLE es\_table (

id BIGINT(20) UNSIGNED NOT NULL,

PRIMARY KEY (id),

UNIQUE KEY unique\_id (id),

client\_name VARCHAR(32) NOT NULL,

modification\_time TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

Let's look at the key principles in the SQL statement above:

* **es table ;**

(The name of the table in which the data is stored.) ;

* **id ;**

(The record's unique identifier. To ensure that each id occurs only once in the current database, id is specified as both a PRIMARY KEY and a UNIQUE KEY. For updating or putting the document into Elasticsearch, this is transformed to \_id. ) ;

* **client name**

( The data that will be ingested into Elasticsearch at the end. This sample simply has one data column to keep things simple. );

* **modification time ;**

( When the record was inserted or last modified, this is the timestamp. This timestamp can be used further down to see what has changed since the last data transfer into Elasticsearch.) ;

1. Now let’s insert something into the DB created:

use es\_db

INSERT INTO es\_table (id, client\_name)

VALUES (1,"TEST1"),

(2,"TEST2"),

(3,"TEST3");

1. See if the SQL statement inserted into the table:

SELECT \* FROM es\_table;

The output should look like this:

A picture containing calendar

Description automatically generated

# *Configure the Logstash JDBC input plugin pipeline:*

Now let’s create a simple Logstash pipeline to ingest data from the DB with the jdbc plugin.

1. Go to the Logstash configuration directory, In Linux this is located here:

/etc/logstash/conf.d Now create a new file named jdbc.conf or how you want, but keep the .conf after the name.

1. Now let’s write a configuration:

input {

jdbc {

jdbc\_driver\_library => "/home/elk/mysql-connector-java-8.0.27/mysql-connector-java-8.0.27.jar"

jdbc\_driver\_class => "com.mysql.jdbc.Driver"

jdbc\_connection\_string => "jdbc:mysql://databaseIP/es\_db"

jdbc\_user => "user"

jdbc\_password => "password"

jdbc\_paging\_enabled => true

tracking\_column => "unix\_ts\_in\_secs"

use\_column\_value => true

tracking\_column\_type => "numeric"

schedule => "\*/5 \* \* \* \* \*"

statement => "SELECT \*, UNIX\_TIMESTAMP(modification\_time) AS unix\_ts\_in\_secs FROM es\_table WHERE (UNIX\_TIMESTAMP(modification\_time) > :sql\_last\_value AND modification\_time < NOW()) ORDER BY modification\_time ASC"

}

}

filter {

mutate {

copy => { "id" => "[@metadata][\_id]"}

remove\_field => ["id", "@version", "unix\_ts\_in\_secs"]

}

}

output {

elasticsearch {

hosts => ["localhost:9200"]

user => "elastic"

password => "elasticuser"

index => "fromdb"

}

stdout {

codec => "rubydebug"

}

}

Now you have to do something here:

* Replace the jdbc connector path with your path, I left mine there for orientation;
* Replace user and password with the credentials you entered in the previous section when you created the DB user;
* Replace databaseIP with the ip of the vm where you installed mysql and es\_db with the name of the table you created;

Now you should lunch the configuration with this command:

/usr/share/logstash/bin/logstash /etc /Logstash/conf.d/{“nameOfTheconfigurationfile”}

The output section of the jdbc.conf has the following configurations:

1. Set the Elasticsearch instance, mine is localhost
2. Set the user if you have enabled authentication and the username password for Elasticsearch and the index where you want to ingest data.

Commands if something is not working, this will start all services needed to run the operation:

1. Make sure to enable and start the Logstash, Elasticsearch Kibana and MySQL deamon:

* systemctl start Logstash ; systemctl enable Logstash
* systemctl start Elasticsearch ; systemctl enable Elasticsearch
* systemctl start Kibana ; systemctl enable Kibana
* systemctl start mysql.service ; systemctl enable mysql.service

1. Go in Kibana, and see the index created in Stack management 🡪 Go to Index pattern and create a pattern from the index created 🡪 Go to discovery to see it . Great no you will see all documents you inserter into **es\_table.**

# *Last Considerations*

The purpose of adding a JDBC connector in the Logstash was to test multiple connections in ELK. This was need to start work on the Log Book Architecture. The Logbook currently is maintained in OneNote, so the intention is to replace OneNote with a WEB form where users can introduce data and after that the data can be send to a DB, then with Logstash to Elasticsearch and finally the inputs from the user can be seen in Kibana UI.