

SYNC YOUR MICROSOFT SQL DATA TO ELASTICSEARCH USING LOGSTASH

We will learn how to sync your Microsoft Sql data to Elasticsearch 7.x. How to schedule and update them with zero effort

AIMS:

- Logstash configuration to sync Microsoft Sql data
- How to build on purpose docker image
- Include plugins in docker image

SUMMARY

In many years our search perception change. Now google search or another search bar in everywhere. You can see your in phone or you can see on computers etc. So many developer needs searches and they have to do it fast. In traditional for searches in Microsoft Sql we can write queries work on small data set but what about in big datasets? You can choose, change queries and their performance and you can replicate your server etc. But this would be bad solution in my opinion but why? Because Microsoft Sql have many tasks, insert data, update data, delete data etc. Of course, it was created for this but you can separate your search from database and you can give that responsibility to another. I mean you can give that job to who is job do search.

Contents

WHAT IS MICROSOFT SQL.....	3
WHAT IS ELASTICSEARCH	3
WHAT IS LOGSTASH	4
WHAT IS DOCKER.....	4
IN ACTION	4
My Own Test Environment.....	5
Docker Setup	5
Windows	6
Ubuntu.....	6
Install Microsoft SQL Server on docker	6
Install Elasticseach on docker	8
What you need for logstash.....	9
Build docker image with logstash plugin.....	9
Create docker compose file for logstash container	11
Logstash configuration file.....	13
RESULTS.....	15

WHAT IS MICROSOFT SQL

Microsoft SQL Server is a relational database management system (RDBMS) that supports a wide variety of transaction processing, business intelligence and analytics applications in corporate IT environments. **Microsoft SQL Server is one of the three market-leading database technologies**, along with Oracle Database and IBM's DB2. (Rouse, 2019)



Like other RDBMS software, Microsoft SQL Server is built on top of SQL, a standardized programming language that database administrators (DBAs) and other IT professionals use to manage databases and query the data they contain. SQL Server is tied to Transact-SQL (T-SQL), an implementation of SQL from Microsoft that adds a set of proprietary programming extensions to the standard language. (Rouse, 2019)

WHAT IS ELASTICSEARCH

Elasticsearch is a **distributed, open source search and analytics engine** for all types of data, including textual, numerical, geospatial, structured, and unstructured. Elasticsearch is built on Apache Lucene and was first released in 2010 by Elasticsearch N.V. (now known as Elastic). Known for its **simple REST APIs, distributed nature, speed, and scalability**, Elasticsearch is the central component of the **Elastic Stack, a set of open source tools for data ingestion, enrichment, storage, analysis, and visualization**. Commonly referred to as **the ELK Stack (after Elasticsearch, Logstash, and Kibana)**, the Elastic Stack now includes a rich collection of lightweight shipping agents known as Beats for sending data to Elasticsearch. (Elasticsearch, 2019)



WHAT IS LOGSTASH

Logstash is an **open source, server-side data processing pipeline** that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favorite "stash." (Elasticsearch, 2019)



WHAT IS DOCKER

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine **(Now they can run Windows machine too and windows containers released)** regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code. (Opensource, 2019)



In a way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application. (Opensource, 2019)

And importantly, Docker is open source. This means that anyone can contribute to Docker and extend it to meet their own needs if they need additional features that aren't available out of the box. (Opensource, 2019)

IN ACTION

Now we can take a action for our scenario. First, we need environment, **you need Elasticsearch, Microsoft SQL server and Logstash.** I used **docker** for this environment but if you already have, you can skip these sections:

- Docker setup
- Windows
- Linux

- MacOS
- Install Microsoft SQL Server on Docker
- Install Elasticsearch on docker
- What you need for logstash
- Build docker image with logstash plugin

My Own Test Environment

My environment is running on Windows with Docker Desktop

Windows version is:

Windows specifications

Edition	Windows 10 Pro
Version	1803
Installed on	25.06.2018
OS build	17134.1006

Docker Desktop version is:

Docker Desktop Community 2.1.0.3

Docker version is:

Docker version 19.03.2, build 6a30dfc

We will use logstash and elasticsearch so their version is:

elasticsearch:7.3.0

logstash:7.3.1

Docker Setup

I use docker for this so if you don't have, I will try to explain installation. Windows part is easy, you can maybe face some challenge but you can do it. Ubuntu part is quite more easy than windows, you will see it. MacOS part... I don't know, I have never a Mac 😞 Mac have Docker desktop too, so it seems easy as Windows

Windows

First of all, go to link <https://www.docker.com/products/docker-desktop> you will see download Docker Desktop Mac or Windows click it. You will see login screen, **you need docker account for download**. You can register here <https://hub.docker.com/signup> and then do the same steps, come login screen, login to system and you will see “**Download Docker Desktop for Windows**” button click and download it.

While installing program you can maybe face it **HYPER-V not enabled problem** Go to your **BIOS settings** and **enable HYPER-V** and then try to install setup again. If you have more trouble please go to this link and read <https://docs.docker.com/docker-for-windows/install/>

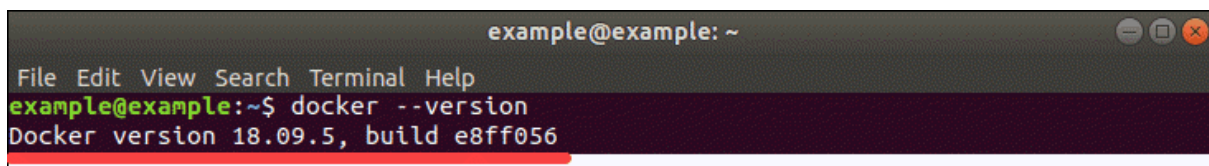
Ubuntu

Ubuntu part is easier than Windows. My ubuntu version is 18.04 first of all open command prompt and write these step by step:

```
sudo apt-get update
sudo apt-get remove docker docker-engine docker.io
sudo apt install docker.io
sudo systemctl start docker
sudo systemctl enable docker
```

And you can check your installation with
docker --version

You will see something like this (Simic, 2018)

A screenshot of a terminal window titled 'example@example: ~'. The terminal has a menu bar with 'File Edit View Search Terminal Help'. The prompt is 'example@example:~\$' and the command entered is 'docker --version'. The output is 'Docker version 18.09.5, build e8ff056'. The output line is highlighted with a red background.

```
example@example: ~
File Edit View Search Terminal Help
example@example:~$ docker --version
Docker version 18.09.5, build e8ff056
```

Install Microsoft SQL Server on docker

Yes, you can run your sql server in docker container. This is good you can skip all sql server setup cause you want only development now, if you in production you can think about it but you want it development NOW!

So open your command prompts and write docker command

```
docker run -e --name dmssql "ACCEPT_EULA=Y" -e "SA_PASSWORD=<StrongPassword>" -p 1433:1433  
-d mcr.microsoft.com/mssql/server:2017-CU8-ubuntu
```

Once you should be careful about **<StrongPassword>** area you have to choose right password for microsoft password policy. If you don't have strong password on your mind, just use <https://passwordsgenerator.net>

In container creation process first you have to download image, **this operation handled automatically by docker(pull operation)**. You should see something like this.

```
Unable to find image 'mcr.microsoft.com/mssql/server:2017-CU8-ubuntu' locally  
2017-CU8-ubuntu: Pulling from mssql/server  
4fa80d7b805d: Downloading [====>] 4.915MB/47.62MB  
484dd0f2fbdc: Download complete  
47004b22ec62: Download complete  
b70745c852a2: Download complete  
718060832ef2: Waiting  
5594e4e5950b: Waiting  
5b67719e2956: Waiting  
7d648891de3f: Waiting  
e0d1b3db20c8: Waiting  
ded313a21911: Waiting
```

When download process is done, you can look container logs for detect Microsoft Sql Server start properly. You can use this command:

```
docker container logs <container-name or id>
```

If you don't know your container name or id you can look them use this command:

```
docker container ps
```

220da27b6315	docker.elastic.co/elasticsearch/elasticsearch:7.3.0	"/usr/local/bin/dock..."	11 days ago
32451c0762ea	hyness/spring-cloud-config-server	"java -Djava.securit..."	13 days ago
3b2a1b5015ae	mcr.microsoft.com/mssql/server:latest	"/opt/mssql/bin/sqls..."	13 days ago
9baadcbcafb5	logstash-with-mssql-plugin:1.6	"/usr/local/bin/dock..."	13 days ago

You should see something like this row. If you don't see mssql. You can look all containers properly start or not:

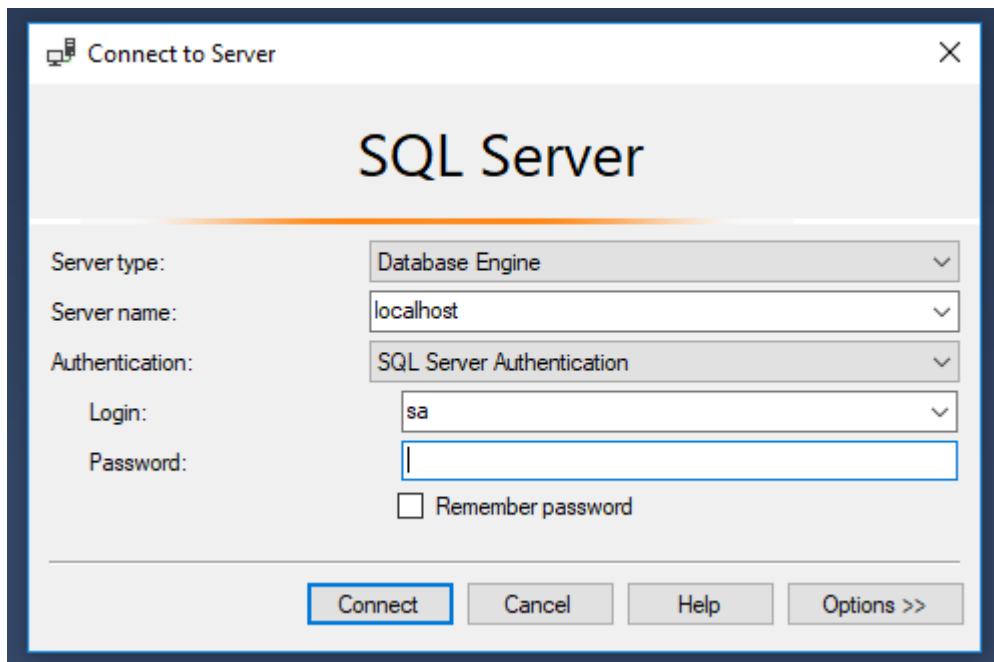
```
docker container ps -a
```

Now you can see mssql row. If you don't see in "docker container ps" command, your container probably not start properly and crash. You can look logs for what is going wrong inside.

If your mssql started properly which you see container in "docker container ps" command, you will see in logs something like this row. If you see, **you started mssql server successfully**

```
2019-09-23 17:20:24.63 spid9s      Converting database 'model' from version 862 to the current version 869.  
2019-09-23 17:20:24.64 spid9s      Database 'model' running the upgrade step from version 862 to version 863.  
2019-09-23 17:20:24.70 spid18s     SQL Server is now ready for client connections. This is an informational message  
user action is required.
```

Then you can log in inside use Sql Server Management Studio.



You have to write your strong password in Password area. You can change your password after login



Install Elasticseach on docker

Let's up and run our elasticsearch on docker. You will need this docker command line for this action:

```
docker run --name els -p 9201:9200 -p 9301:9300 -e "discovery.type=single-node"
docker.elastic.co/elasticsearch/elasticsearch:7.3.0
```

You will see pulling image screen again like previous steps (You can look “**Install Microsoft SQL Server on docker**” section) And if you look containers logs again you have to see something like row:

```
{ "type": "server", "timestamp": "2019-09-24T17:00:12,916+0000", "level": "INFO", "component": "o.e.n.Node", "cluster.name": "docker-cluster", "node.id": "X0BqW5HWSe2Qc6M2yR_Fw", "message": "started" }
{"type": "server", "timestamp": "2019-09-24T17:00:13,197+0000", "level": "INFO", "component": "o.e.g.GatewayService", "cluster.name": "docker-cluster", "node.id": "X0BqW5HWSe2Qc6M2yR_Fw", "message": "recovered [0] indices into cluster state" }
{"type": "server", "timestamp": "2019-09-24T17:00:13,848+0000", "level": "INFO", "component": "o.e.c.m.MetaDataIndexTemplateService", "cluster.name": "docker-cluster", "node.id": "X0BqW5HWSe2Qc6M2yR_Fw", "message": "adding template [.watch-history-10] for index patterns [.watcher-history-10*]" }
{"type": "server", "timestamp": "2019-09-24T17:00:13,979+0000", "level": "INFO", "component": "o.e.c.m.MetaDataIndexTemplateService", "cluster.name": "docker-cluster", "node.id": "X0BqW5HWSe2Qc6M2yR_Fw", "message": "adding template [.watches] for index patterns [.watches*]" }
{"type": "server", "timestamp": "2019-09-24T17:00:14,102+0000", "level": "INFO", "component": "o.e.c.m.MetaDataIndexTemplateService", "cluster.name": "docker-cluster", "node.id": "X0BqW5HWSe2Qc6M2yR_Fw", "message": "adding template [.triggered_watches] for index patterns [.triggered_watches*]" }
```

Again, if you **don't remember how you can look logs of containers**, you can look previous step for inspect logs (You can look “**Install Microsoft SQL Server on docker**” section)

What you need for logstash

We need **synchronize our Microsoft SQL Server data to Elasticsearch**, we search in the internet and we find logstash app. If you run logstash, i mean pure logstash, **have no capability transform MSSQL data to Elasticsearch**. We need **plugin for that** and we **have to install in our logstash image**. So we need to build a docker image. We need only “**logstash-input-jdbc**” plugin for this operation but I will install few more plugins:

- logstash-filter-jdbc_streaming
- logstash-filter-aggregate
- **logstash-input-jdbc**
- logstash-filter-translate

logstash-filter-jdbc_streaming: This guy taking you sql results and **keep in temporary cache** (Elasticsearch, 2019)

logstash-filter-aggregate: If you have multiple events on several entities, this plugin **can take them and push aggerated results one event** (Elasticsearch, 2019)



logstash-filter-translate: This plugin is general search and replace tool, looking columns find values, replace and put in destination columns in Elasticsearch (Elasticsearch, 2019)

logstash-input-jdbc: This is our plugin to transform sql data to Elasticsearch. **This plugin is designed for get data any database with JDBC interface**. (Elasticsearch, 2019)

We will use mainly logstash-input-jdbc plugin. This guy **need driver for connect to sql servers** (oracle, mssql, mysql, postgresql). We have **to download jdbc driver for him**. I create folder which name is jars and put drivers in that folder. I use “**mssql-jdbc-7.4.1.jre8**” driver for connect Microsoft SQL Server. You can download directly these drivers from <https://github.com/microsoft/mssql-jdbc/releases> address. If you download driver, now we can go to next step.

Build docker image with logstash plugin

First you **can create a folder with name “logstash”**, than inside that folder **create another folder and give name that folder to “jars”** Then **create a file, this file name have to be “Dockerfile”**, you **don’t need an extension for this file** so name it. After that you have seen something like this:

 jars	27.08.2019 18:34	File folder	
 Dockerfile	27.08.2019 18:34	File	1 KB

Now open Dockerfile using any text editor and start writing.

```
FROM docker.elastic.co/logstash/logstash:7.3.1
```

```
RUN logstash-plugin install logstash-filter-jdbc_streaming
```

```
RUN logstash-plugin install logstash-filter-aggregate
```

```
RUN logstash-plugin install logstash-input-jdbc
```

```
RUN logstash-plugin install logstash-filter-translate
```

```
COPY ./jars /usr/share/logstash/logstash-core/lib/jars
```

Let's explain line by line. First line, say docker to which image we are using for build our image, we use **"FROM"** command for this in docker World. After that we use **"RUN"** command for execute to something inside container. In next four line we are installing our plugins which we mentioned in previous section. Last line we are copy our jars folder content to inside Logstash container image, which our jdbc plugin use for later to connect our Microsoft SQL Server. You can put that folder more than one jars if you want multiple data streams.

After that explanation we have to build our image. We have to write docker command for this situation. Let's do that. Go to Logstash folder in command prompt and write:

```
docker build -t logstash-with-mssql-plugin:1.7 .
```

You have to be careful in writing command process, last character should be dot(.) with spacebar separate. If you do that successfully you should see something like this

```
Sending build context to Docker daemon 1.242MB
Step 1/6 : FROM docker.elastic.co/logstash/logstash:7.3.1
--> 1c5964893183
Step 2/6 : RUN logstash-plugin install logstash-filter-jdbc_streaming
--> Using cache
--> 20394e05d170
Step 3/6 : RUN logstash-plugin install logstash-filter-aggregate
--> Using cache
--> f38b88e924a8
Step 4/6 : RUN logstash-plugin install logstash-input-jdbc
--> Using cache
--> 612c5b556e48
Step 5/6 : RUN logstash-plugin install logstash-filter-translate
--> Using cache
--> f28b0ffa6608
Step 6/6 : COPY ./jars /usr/share/logstash/logstash-core/lib/jars
--> Using cache
--> e326b16e7e42
Successfully built e326b16e7e42
Successfully tagged logstash-with-mssql-plugin:1.7
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.
```

In command -t means tag and this is our image name actually. This info will be usefully for when we are creating compose file just keep that your mind 😊

Create docker compose file for logstash container

Now we come Logstash container creation process. This can be done using by docker run command but this will be long command. So, we need to see big picture, so we need something like code file. You can use that for docker-compose, let's get our hands dirty! **Create logstash-compose.yml file in our "logstash" folder.**

```
version: '3.4'

services:
  logstash:
    image: logstash-with-mssql-plugin:1.6
    container_name: logstash
    ports:
      - "5001:5001"
    environment:
      LS_JAVA_OPTS: "-Xmx256m -Xms256m"
    volumes:
      - c:/docker_logstash_volume/pipeline:/usr/share/logstash/pipeline/
      - c:/docker_logstash_volume/config/logstash.yml:/usr/share/logstash/config/logstash.yml
    network_mode: bridge
```

I give that as image because my code not fitting good in word, **don't worry you can access that codes in my GitHub account repository.** I will give you address end of the article. Let's explain this code.

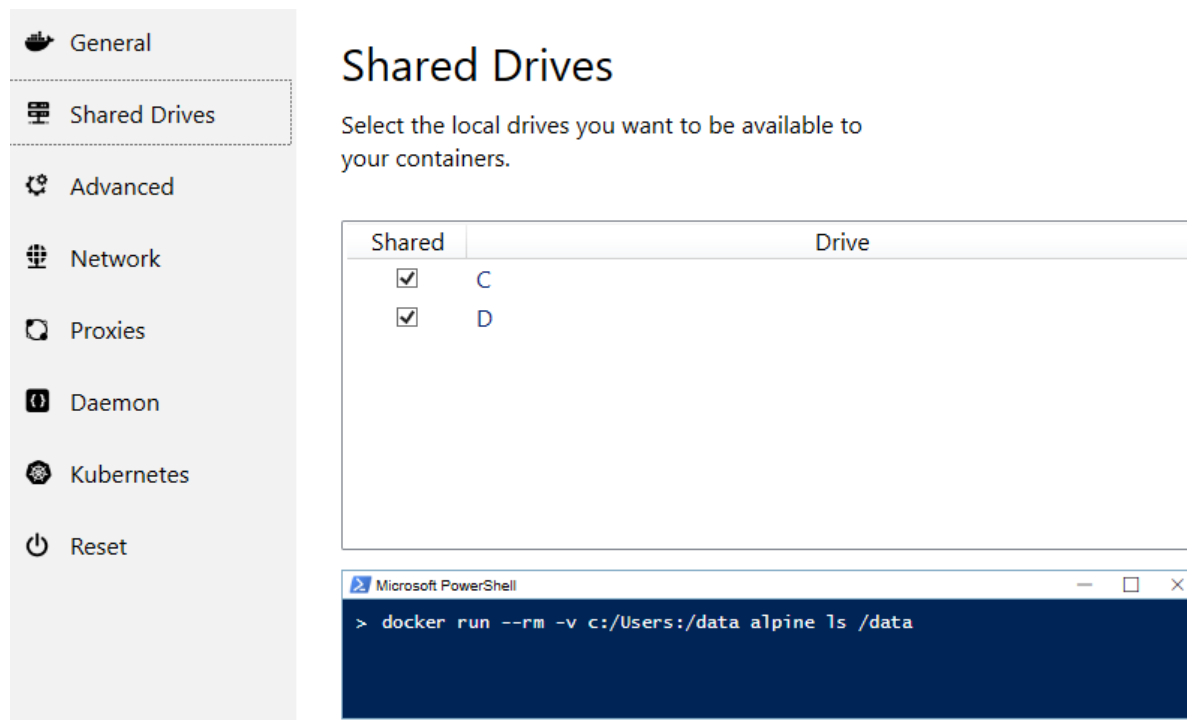
Version number tells that which compose version will be using this compose file. **You can access their differences** from <https://docs.docker.com/compose/compose-file/compose-versioning/>

In services section we are writing our container specs. We are giving service name first which represent as "logstash:" and then step in code for select image to our containers, you can see in our builder image on this section, I am talking about this section "image: logstash-with-mssql-plugin:1.6" Then we can give container_name our container for easy find in docker network. In docker world

container name represent as hostname like www.google.com and access that container from another container, you can use this container_name like hostname.

Of course, we need expose our ports for accessing Logstash from host computer. You can use for this purpose “ports” command in compose file. In environment section we can pass special configuration parameters to container. This can be different for every docker image also some image doesn’t need env variable too, so why we use this environment variable in our compose file. When Logstash start need initial heap in memory. You can set these options using LS_JAVA_OPTS variable. You can see detailed explanation in [here](https://www.elastic.co/guide/en/elasticsearch/reference/current/jvm-options.html)

Now we are in volume section, this section is critical because we need access some things inside container also need somethings persist (data config etc.) We can use volume section to share data, config etc. between host computer and container. If you in windows you can share your drive inside docker, go to settings→Shared Drivers section and choose your drive.



After that explanations now volume section has to be clearer to you. In Logstash container system I believe we should share two section. First is pipeline section which is critical for our data transformation to Elasticsearch, we will put Logstash.config file in here, we will see config file next section 😊 Second one is main config file logstash.yml, not need for this scenario but later more complex scenarios you can be need that file.

Last code is network_mode section, this is critical because we have two other container mssql and elasticsearch. We have to communicate these containers so need in bridge network, because docker run command create container in bridge network. Now let’s run our compose file. In command prompt come in “logstash” folder.

```
docker-compose -f logstash-compose.yml up -d
```

After you have to see something like this:

```
WARNING: The Docker Engine you're using is running in swarm mode.
Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the
current node.
To deploy your application across the swarm, use `docker stack deploy`.
Creating logstash ... done
```

Now we have to create our configuration file and then restart this container or recreate it, let's do that!

Logstash configuration file

In the beginning of article, our need is transforming Microsoft SQL data to Elasticsearch. One more step and this will be done for you. When we are creating Logstash container we also create shared volume and put "pipeline folder" into shared folder, now go to that folder. Create file inside that folder and give "logstash.conf" as name. In that file generally we have 3 section.

- Input
- Filter
- Output

Input Section: We define which data will transform to Elasticsearch. Also, we define driver_class, schedule etc.

Filter Section: If you need convert your data or filter your data by content. You can use this section.

Output Section: In this section you say to logstash take data from input and filter, put elsewhere (Elasticsearch another resource)

Let's look in code

```
input {
  jdbc {
    jdbc_driver_library => ""
    jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
    jdbc_connection_string => "jdbc:sqlserver://mssql;databasename=TestDB"
    jdbc_user => "sa"
    jdbc_password => "1234"
    jdbc_fetch_size => 1000
    schedule => "*/45 * * * *"
    lowercase_column_names => false
    type=>"customer"
    statement => "select * from customers"
  }
}
```

```
filter {
  mutate {
    convert => { "Timestamp" => "string" }
  }
  date {
    match => ["Timestamp", "ISO8601"]
    target => "@timestamp"
  }
}
output {
  stdout {
    codec => rubydebug
  }
  if [type] == "customer" {
    elasticsearch{
      hosts => ["http://elasticsearch:9200"]
      index => "testdb-index-customer"
      document_id => "%{CustomerId}"
    }
  }
}
```

Let's little explain the code, in input section we start with jdbc and this represent which plugin we will use, inside that tag you see `jdbc_driver_library=""` we don't need spesifiy driver_library because we put library files when we are creating container.

Second line is `"jdbc_driver_class"` we specify which jdbc class we are using and then we define `"jdbc_connection_string"` we are writing our Microsoft SQL Server connection string after that we need username and password `"jdbc_user"` and `"jdbc_password"` code block doing that work.

Our fetch size is classic fetch size. You know fetch size is important when you transfer big amount of data. If you have 10.000 row data and your fetch size is 1000 logstash will create 10 different connection while getting data.

In schedule code block we decide which time our transform operation will run, this code block use cron job pattern, if you don't know you can look in here <https://crontab.guru> After that you see `"lowercase_column_names"` using this code block you can specialize your columns name transformation. We are using `"type"` code because when we have multiple jdbc input and if we want specialize each input's output by one by one, you can use this code block and in output section you can use `"if"` block and separate to output. Last code block in jdbc section is `"statement"` you can write your sql queries here.

In filter code block we see two section. In my customer table I have `"Timestamp"` column and this area type is Datetime. First I convert this area to string and then taking this string area and converting date

using "ISO8601" standard and put this area to "@timestamp" area. Why I am doing this, because I don't want to use default Elasticsearch @timestamp and overwriting my "Timestamp" area.

Let's come our final code block "output" We take our data in input, we put some filter on in filter section and now we have to write our data elsewhere. In stdout section actually write data to console. You can look details in here <https://www.elastic.co/guide/en/logstash/7.3/plugins-outputs-stdout.html> Ooo we see type, remember we just talked. If our type customer use Elasticsearch. If you create another jdbc code block and if you want it transform to different way, you can use like this. I am saying that something like this in here.

If my type "customer", use "elasticsearch" and this "elasticsearch" hosts are ["http://elasticsearch:9200"] in here, data index will be "testdb-index-customer" and document_id will be our "CustomerId" column which represent like this in code block "%{CustomerId}"

Let's look our results 😊

RESULTS

After put our logstash.conf file in pipeline folder. We need restart our container. You can use these commands:

```
docker-compose -f logstash-compose.yml down
docker-compose -f logstash-compose.yml up -d
```

If you want follow logs don't use -d keyword in docker-compose up command. If you miss this step, look your container name and write this command:

```
docker container logs <container-name> --follow
```

If your transform success you should see in your console screen something like this.

```
{
  "@timestamp" => 2017-04-29T08:24:02.168Z,
  "port" => 27326,
  "domain" => "niiconstlting.com",
  "@version" => "1",
  "host" => "127.0.0.1",
  "message" => "niiconstlting.com",
  "type" => "proxy",
  "tags" => []
}
{
  "@timestamp" => 2017-04-29T08:24:02.168Z,
  "port" => 27326,
  "domain" => "niiconsudting.com",
  "@version" => "1",
  "host" => "127.0.0.1",
  "message" => "niiconsudting.com",
  "type" => "proxy",
  "tags" => []
}
```

References

Elasticsearch, C., 2019. <https://www.elastic.co>. [Online]
Available at: https://www.elastic.co/guide/en/logstash/current/plugins-filters-jdbc_streaming.html
[Accessed 28 09 2019].

Elasticsearch, C., 2019. <https://www.elastic.co>. [Online]
Available at: <https://www.elastic.co/guide/en/logstash/current/plugins-filters-aggregate.html>
[Accessed 28 09 2019].

Elasticsearch, C., 2019. <https://www.elastic.co>. [Online]
Available at: <https://www.elastic.co/guide/en/logstash/current/plugins-filters-translate.html>
[Accessed 28 09 2019].

Elasticsearch, C., 2019. <https://www.elastic.co>. [Online]
Available at: <https://www.elastic.co/guide/en/logstash/current/plugins-inputs-jdbc.html>
[Accessed 28 09 2019].

Elasticsearch, C., 2019. www.elastic.co. [Online]
Available at: <https://www.elastic.co/what-is/elasticsearch>
[Accessed 21 09 2019].

Elasticsearch, C., 2019. www.elastic.co. [Online]
Available at: <https://www.elastic.co/products/logstash>
[Accessed 21 09 2019].

Opensource, 2019. opensource.com. [Online]
Available at: <https://opensource.com/resources/what-docker>
[Accessed 21 09 2019].

Rouse, M., 2019. searchsqlserver.techtarget.com. [Online]
Available at: <https://searchsqlserver.techtarget.com/definition/SQL-Server>
[Accessed 21 09 2019].

Simic, S., 2018. phoenixnap.com. [Online]
Available at: <https://phoenixnap.com/kb/how-to-install-docker-on-ubuntu-18-04>
[Accessed 22 09 2019].

Contributors:

[Can Cömert](#)

This article supports Open-Article project.

Open article means, you can get these articles and convert them into your needs. If you see misspelling on articles, you can contribute to here or you can translate in your language. You can write your article and put our community.

<https://github.com/organizations/Open-Article>