Big Data

ELK Stack Exercise

von
Benjamin Ellmer (S2210455012)

Mobile Computing Master
FH Hagenberg

April 27, 2023

Step 1

Clone repo and run the containers using docker compose:

```
git clone https://github.com/Digital-Media/elk-stack-dock
cd elk-stack-dock
docker compose -f docker-compose.yml up -d
```

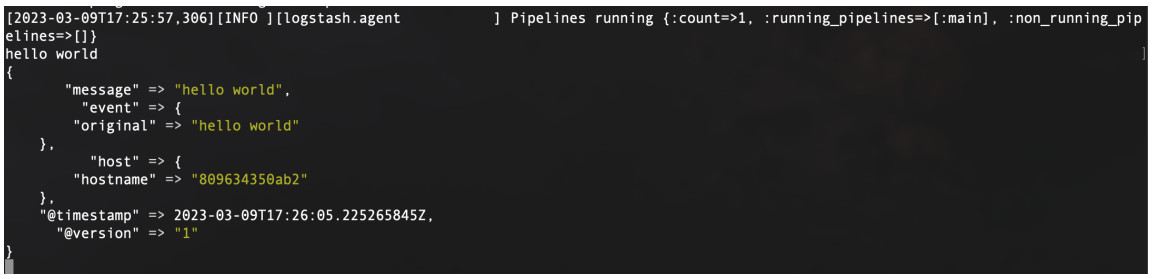
Open bash inside logstash container and run simple pipeline:

```
docker exec -it elk-stack-dock_logstash_1 /bin/bash
whereis logstash
cd /usr/share/logstash/bin
./logstash -e 'input { stdin { } } output { stdout {} }'
```

The pipeline fails because data is already in use

Rerun pipeline using the data2 directory

```
logstash --path.data data2 -e 'input { stdin { } } output {
    stdout {} }'
```



Check if jdbc is installed

```
logstash-plugin list --verbose | grep "jdbc"
```

```
logstash@b0edb9428e65:~/bin$ logstash-plugin list --verbose | grep "jdbc"
io/console on JRuby shells out to stty for most operations
logstash-integration-jdbc (5.4.1)
 ├── logstash-input-jdbc
 ├── logstash-filter-jdbc_streaming
 └── logstash-filter-jdbc_static
logstash@b0edb9428e65:~/bin$
```

Check pipeline configuration

```
logstash --path.settings /usr/share/logstash/config -t
```

```
logstash@b0edb9428e65:~$ /usr/share/logstash/bin/logstash --path.settings /usr/share/logstash/config -t
Using bundled JDK: /usr/share/logstash/jdk
Sending Logstash logs to /usr/share/logstash/logs which is now configured via log4j2.properties
[2023-03-09T18:21:20,652][INFO ][logstash.runner          ] Log4j configuration path used is: /usr/share/logstash/config/log4j2.propert
ies
[2023-03-09T18:21:20,667][INFO ][logstash.runner          ] Starting Logstash {"logstash.version"=>"8.6.1", "jruby.version"=>"jruby 9.3
.8.0 (2.6.8) 2022-09-13 98d69c9461 OpenJDK 64-Bit Server VM 17.0.5+8 on 17.0.5+8 +indy +jit [aarch64-linux]"}
[2023-03-09T18:21:20,672][INFO ][logstash.runner          ] JVM bootstrap flags: [-Xms1g, -Xmx1g, -Djava.awt.headless=true, -Dfile.enco
ding=UTF-8, -Djruby.compile.invokedynamic=true, -XX:+HeapDumpOnOutOfMemoryError, -Djava.security.egd=file:/dev/urandom, -Dlog4j2.isThre
adContextMapInheritable=true, -Xms256m, -Xmx256m, -Djruby.regexp.interruptible=true, -Djdk.io.File.enableADS=true, --add-exports=jdk.co
mpiler/com.sun.tools.javac.api=ALL-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.file=ALL-UNNAMED, --add-exports=jdk.compiler
/com.sun.tools.javac.parser=ALL-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.tree=ALL-UNNAMED, --add-exports=jdk.compiler/co
m.sun.tools.javac.util=ALL-UNNAMED, --add-opens=java.base/java.security=ALL-UNNAMED, --add-opens=java.base/java.io=ALL-UNNAMED, --add-o
pens=java.base/java.nio.channels=ALL-UNNAMED, --add-opens=java.base/sun.nio.ch=ALL-UNNAMED, --add-opens=java.management/sun.management=
ALL-UNNAMED]
[2023-03-09T18:21:21,310][INFO ][org.reflections.Reflections] Reflections took 83 ms to scan 1 urls, producing 127 keys and 444 values
[2023-03-09T18:21:22,065][INFO ][logstash.javapipeline    ] Pipeline `main` is configured with `pipeline.ecs_compatibility: v8` setting
. All plugins in this pipeline will default to `ecs_compatibility => v8` unless explicitly configured otherwise.
Configuration OK
[2023-03-09T18:21:22,066][INFO ][logstash.runner          ] Using config.test_and_exit mode. Config Validation Result: OK. Exiting Logs
tash
```

Step 2

Clone postgres repo

```
git clone https://github.com/Digital-Media/postgres
cd postgres
```

Create sql scripts in local folder named "scripts"

See schema.sql and data.sql in appendix.

Map the scipts folder to the container adding scripts to the volumes:

```
volumes:
  - ./scripts:/scripts
  - pgdata:/var/lib/postgresql/data
```

Start postgres container with docker compose:

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

Open psql inside postgres container create schema and insert data:

```
docker exec -it postgres psql -d postgres -U postgres
\i scripts/schema.sql
\i scripts/data.sql
```

Download jdbc driver to local machine

```
cd elk-stack-dock/logstash
mkdir jars
cd jars
curl -o postgresql-42.5.4.jar https://jdbc.postgresql.org/
   download/postgresql-42.5.4.jar
```

Add this line to the end of the logstash Dockerfile

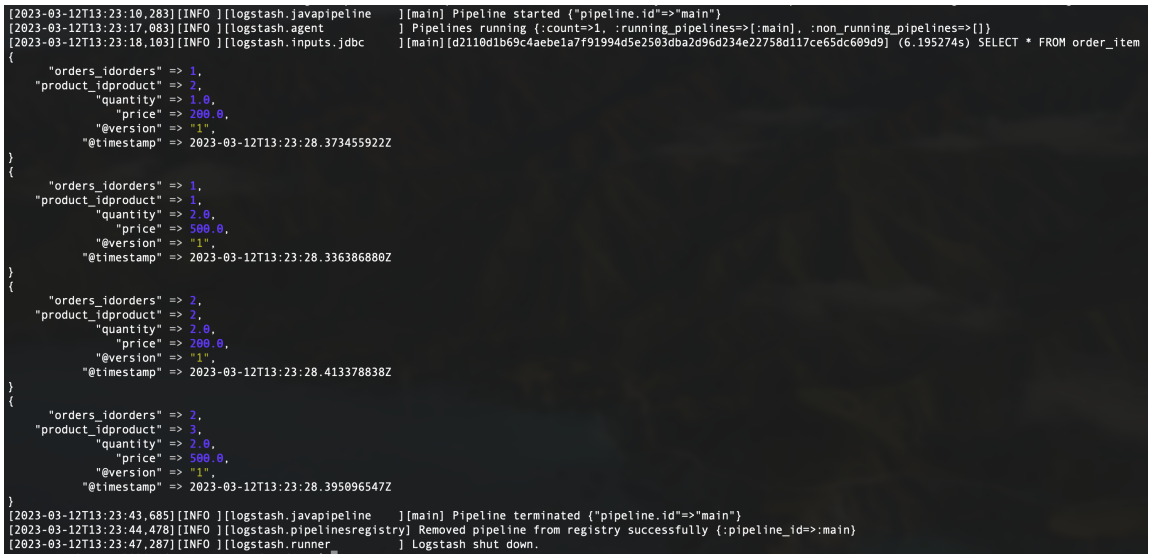
```
COPY jars/* /usr/share/logstash/logstash-core/lib/jars/
```

Run docker compose with build flag

```
docker compose -f elk-stack-dock/docker-compose.yml up -d
   --build
```

Test Postgres pipeline

```
logstash --path.data data2 -e 'input {
  jdbc {
    jdbc_connection_string => "jdbc:postgresql
       ://192.168.0.23:5432/postgres"
    jdbc_user => "postgres"
    jdbc_password => "geheim"
    jdbc_driver_class => "org.postgresql.Driver"
    statement => "SELECT * FROM order_item"
  }
} output {
  stdout {}
}'
```



Write the new pipeline in the pipeline configuration file:

See pipeline.conf in appendix for the pipeline

```
nvim elk-stack-dock/logstash/pipeline/logstash.conf
```

Restart the logstash container to apply the new configuration

```
docker stop elk-stack-dock_logstash_1 && docker start elk-
   stack-dock_logstash_1
```

Check the logstash docker logs:

```
docker logs -f elk-stack-dock_logstash_1
```

```
[2023-04-01T14:12:39,347][INFO ][logstash.javapipeline    ][main] Starting pipeline {:pipeline_id=>"main", "pipeline.workers"=>4, "pipeline.ba
tch.size"=>125, "pipeline.batch.delay"=>50, "pipeline.max_inflight"=>500, "pipeline.sources"=>["/usr/share/logstash/pipeline/logstash.conf"],
:thread=>"#<Thread:0x31335ebe@/usr/share/logstash/logstash-core/lib/logstash/java_pipeline.rb:131 run>"}
[2023-04-01T14:12:39,685][INFO ][logstash.javapipeline    ][main] Pipeline Java execution initialization time {"seconds"=>0.34}
[2023-04-01T14:12:39,942][INFO ][logstash.inputs.jdbc     ][main] ECS compatibility is enabled but `target` option was not specified. This may
 cause fields to be set at the top-level of the event where they are likely to clash with the Elastic Common Schema. It is recommended to set
the `target` option to avoid potential schema conflicts (if your data is ECS compliant or non-conflicting, feel free to ignore this message)
[2023-04-01T14:12:39,943][INFO ][logstash.javapipeline    ][main] Pipeline started {"pipeline.id"=>"main"}
[2023-04-01T14:12:39,952][INFO ][logstash.agent           ] Pipelines running {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=
>[]}
```

Check the data in kibana

```
GET orders/_search
```


```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 2,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "orders",
        "_id": "orders_2",
        "_score": 1,
        "_source": {
          "date_ordered": "22.02.2022",
          "married_to": null,
          "idorders": 2,
          "user_iduser": 2,
          "@timestamp": "2023-04-01T14:15:00.617472422Z",
```


```
GET orders/_doc/orders_1
```


```
{
  "_index": "orders",
  "_id": "orders_1",
  "_version": 1,
  "_seq_no": 1,
  "_primary_term": 1,
  "found": true,
  "_source": {
    "date_ordered": "12.02.2021",
    "married_to": null,
    "idorders": 1,
    "user_iduser": 1,
    "@timestamp": "2023-04-01T14:15:00.617102963Z",
    "total_sum": 1200,
    "@version": "1"
  }
}
```


Add a new sql entry (between 14:45 and 14:50) then check logs -> the data in es is updated every 5 minutes:

```
docker exec -it postgres psql -d postgres -U postgres
INSERT INTO orders (idorders, user_iduser, total_sum,
   date_ordered) VALUES (3, 2, 1400, '23.02.2022');
```

```
[2023-04-01T14:12:39,952][INFO ][logstash.agent           ] Pipelines running {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=
>[]}
[2023-04-01T14:15:00,609][INFO ][logstash.inputs.jdbc     ][main][5d69d240eb63fdb8d96c21c8e257955a67fce95a451f4c92529085a0ff2fe7c2] (0.013463s
) SELECT * FROM orders where idorders > 0
/usr/share/logstash/vendor/bundle/jruby/2.6.0/gems/manticore-0.9.1-java/lib/manticore/client.rb:284: warning: already initialized constant Man
ticore::Client::HttpPost
/usr/share/logstash/vendor/bundle/jruby/2.6.0/gems/manticore-0.9.1-java/lib/manticore/client.rb:536: warning: already initialized constant Man
ticore::Client::StringEntity
[2023-04-01T14:20:00,939][INFO ][logstash.inputs.jdbc     ][main][5d69d240eb63fdb8d96c21c8e257955a67fce95a451f4c92529085a0ff2fe7c2] (0.003809s
) SELECT * FROM orders where idorders > 2
[2023-04-01T14:25:00,147][INFO ][logstash.inputs.jdbc     ][main][5d69d240eb63fdb8d96c21c8e257955a67fce95a451f4c92529085a0ff2fe7c2] (0.002284s
) SELECT * FROM orders where idorders > 2
[2023-04-01T14:30:00,142][INFO ][logstash.inputs.jdbc     ][main][5d69d240eb63fdb8d96c21c8e257955a67fce95a451f4c92529085a0ff2fe7c2] (0.002445s
) SELECT * FROM orders where idorders > 2
[2023-04-01T14:35:00,121][INFO ][logstash.inputs.jdbc     ][main][5d69d240eb63fdb8d96c21c8e257955a67fce95a451f4c92529085a0ff2fe7c2] (0.003371s
) SELECT * FROM orders where idorders > 2
[2023-04-01T14:40:00,124][INFO ][logstash.inputs.jdbc     ][main][5d69d240eb63fdb8d96c21c8e257955a67fce95a451f4c92529085a0ff2fe7c2] (0.002377s
) SELECT * FROM orders where idorders > 2
[2023-04-01T14:45:00,144][INFO ][logstash.inputs.jdbc     ][main][5d69d240eb63fdb8d96c21c8e257955a67fce95a451f4c92529085a0ff2fe7c2] (0.002180s
) SELECT * FROM orders where idorders > 2
[2023-04-01T14:50:00,208][INFO ][logstash.inputs.jdbc     ][main][5d69d240eb63fdb8d96c21c8e257955a67fce95a451f4c92529085a0ff2fe7c2] (0.003399s
) SELECT * FROM orders where idorders > 2
[2023-04-01T14:55:00,262][INFO ][logstash.inputs.jdbc     ][main][5d69d240eb63fdb8d96c21c8e257955a67fce95a451f4c92529085a0ff2fe7c2] (0.002022s
) SELECT * FROM orders where idorders > 3
```

Logstash pipeline vs. MongoDB pipeline

MongoDB supports a way to import data from files e.g. json files using mongoimport. The same workflow that we implemented using logstash contains this steps using MongoDB and mongoimport:

1. Check which data is already synced to the MongoDB e.g. storing the last synced id
2. Export the "new" data from postgres to e.g. json file
3. Import the json file using mongoimport

In my opinion this is not the perfect solution, because it means, that some code has to be written and maintained if this should be done automatically. Of course, it is not very complex to sequentially fetch the new data, convert it to json and write it into MongoDB. But writing the code for the pipeline and hosting the pipeline is much more error prone than just using and configuring a tool like logstash.

Still I would not decide to use ElasticSearch just because of the advantage of logstash compared to mongoimport.

Step 3

Start Metricbeat

```
docker-compose \
    docker-compose.yml \
    -f extensions/metricbeat/metricbeat-compose.yml \
    up -d
```

Change the environments in the .env file

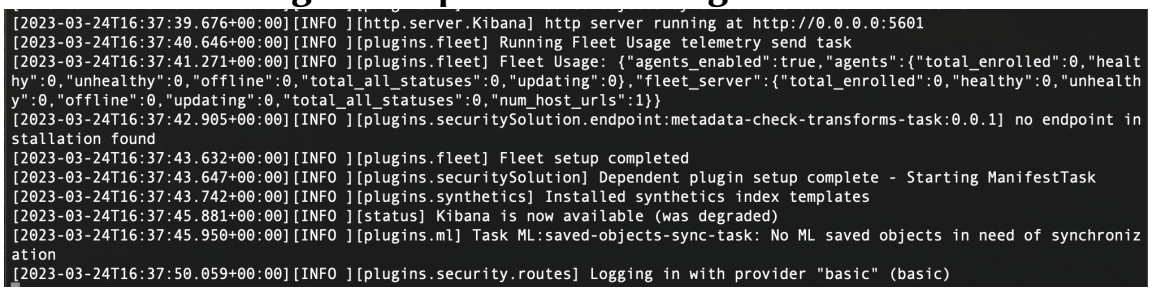
See the file .env file in the appendix.

```
nvim elk-stack-dock/.env
```

Run the setup container again

```
docker compose -f docker-compose.yml rm setup
docker volume rm elk-stack-dock_setup
docker-compose -f docker-compose.yml up setup
```

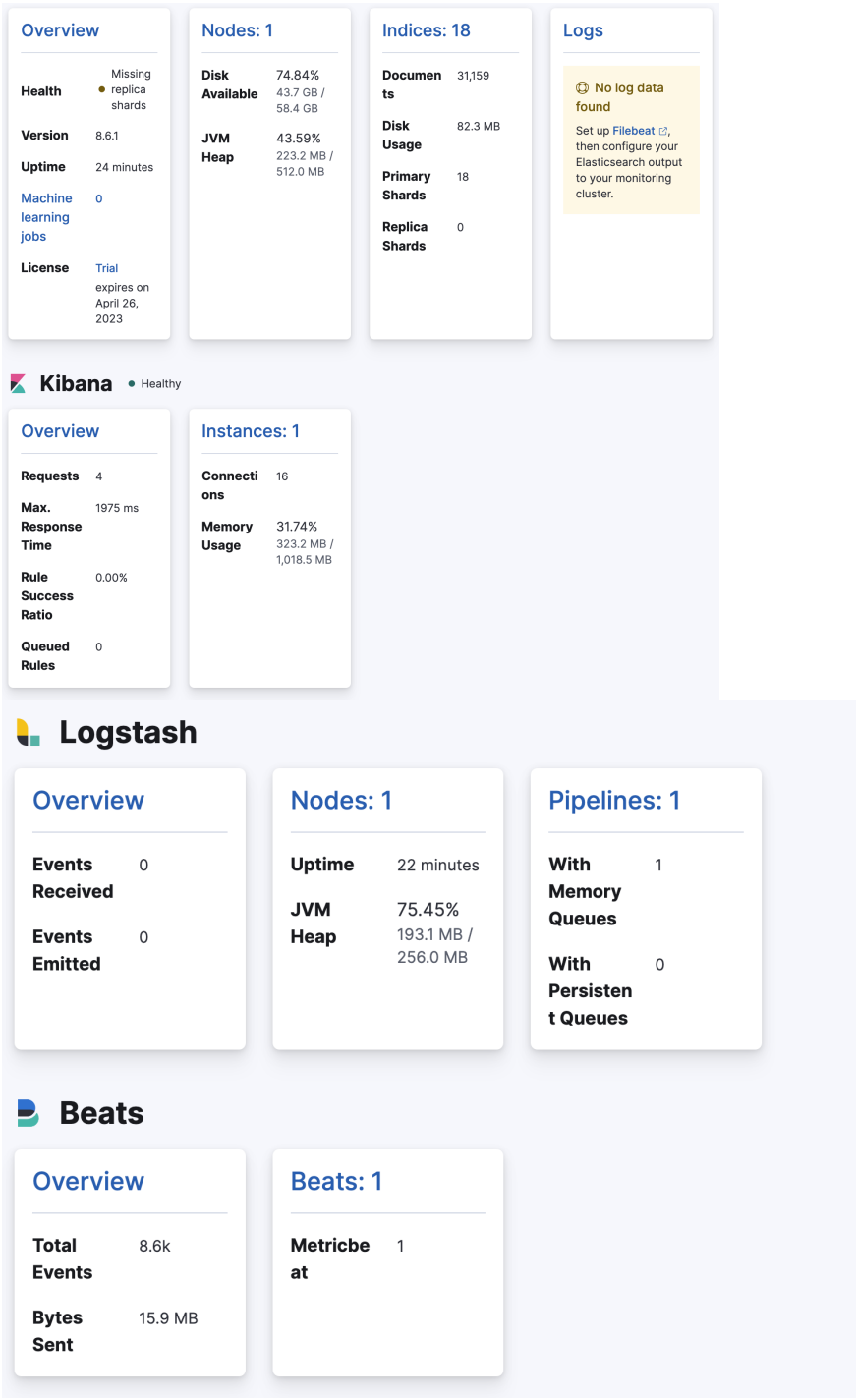
Check Kibana logs => http server running at...



Restart metricbeat (as soon as Kibana is ready)

```
docker-compose \
    -f docker-compose.yml \
    -f extensions/metricbeat/metricbeat-compose.yml \
    restart metricbeat
```

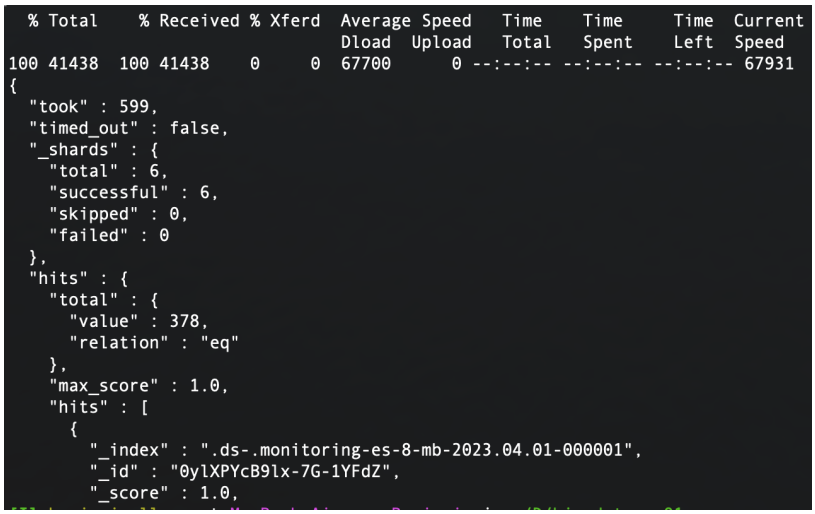
Check metricbeat data in Kibana



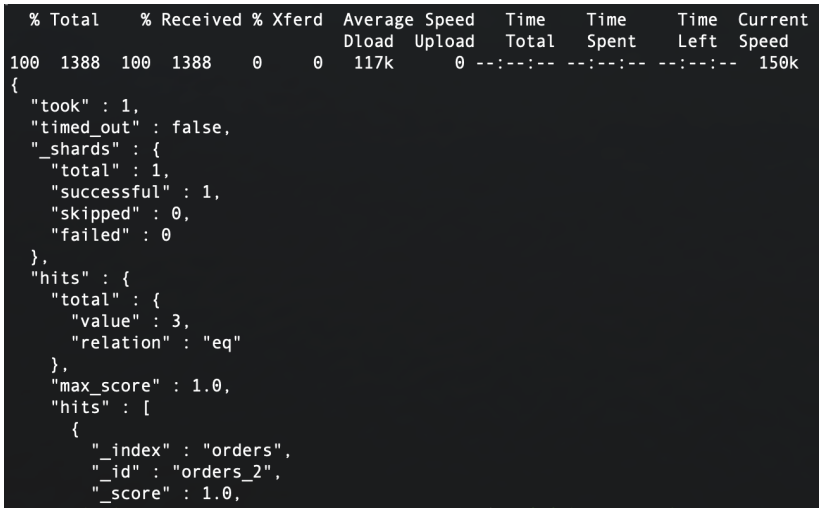
Retrieve data using curl

Please write me if you need the files response1.json and response2.json. I did not hand them in because we can upload only one document in moodle, and I guess the screenshots show the most important parts.

```
curl -X GET -u elastic:changeme "http://localhost:9200/
   _search?pretty" > response1.json && head -n20 response1.
   json
```




```
curl -X GET -u elastic:changeme "http://localhost:9200/
   orders/_search?pretty" > response2.json && head -n20
   response2.json
```



Match specific entry

```
↳ curl -X GET -u elastic:changeme "http://localhost:9200/_search?pretty" -H 'Content-Type: application/json' -d '{
    "query": {
      "match": {
        "_id": "orders_1"
      }
    }
  }'

{
  "took" : 8,
  "timed_out" : false,
  "_shards" : {
    "total" : 6,
    "successful" : 6,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "orders",
        "_id" : "orders_1",
        "_score" : 1.0,
        "_source" : {
          "date_ordered" : "12.02.2021",
          "married_to" : null,
          "idorders" : 1,
          "user_iduser" : 1,
          "@timestamp" : "2023-04-01T14:15:00.617102963Z",
          "total_sum" : 1200.0,
          "@version" : "1"
        }
      }
    ]
  }
}
```

Match entry that does not exist

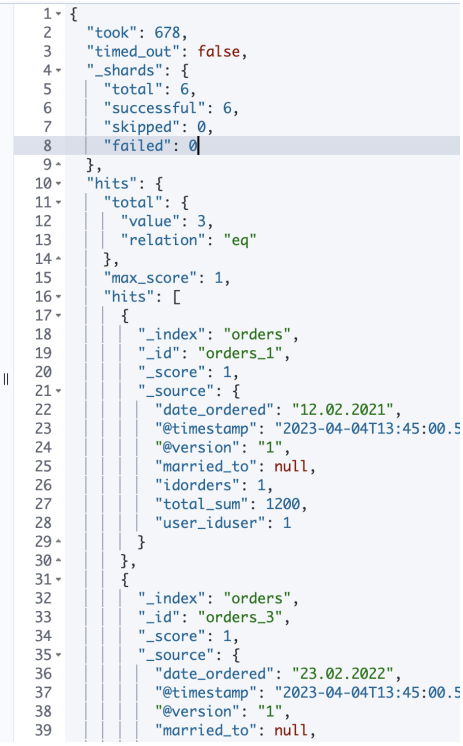
```
↳ curl -X GET -u elastic:changeme "http://localhost:9200/_search?pretty" -H 'Content-Type: application/json' -d '{
    "query": {
      "match": {
        "_id": "orders_10"
      }
    }
  }'

{
  "took" : 1119,
  "timed_out" : false,
  "_shards" : {
    "total" : 6,
    "successful" : 6,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 0,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  }
}
```

Query all entries in index without match condition

```
GET orders/_search
```

Query all entries in index with match condition

```
GET _search
{
  "query": {
    "match": {
      "_index": "orders"
    }
  }
}
```



Query specific entry without match condition

```
GET orders/_doc/orders_3
```


```
{
  "_index": "orders",
  "_id": "orders_3",
  "_version": 1,
  "_seq_no": 2,
  "_primary_term": 1,
  "found": true,
  "_source": {
    "date_ordered": "23.02.2022",
    "married_to": null,
    "idorders": 3,
    "user_iduser": 2,
    "@timestamp": "2023-04-01T14:
    "total_sum": 1400,
    "@version": "1"
  }
}
```


Query specific entry with match condition

```
GET _search
{
  "query": {
    "match": {
      "_id": "orders_1"
    }
  }
}
```


```
{
  "took": 5,
  "timed_out": false,
  "_shards": {
    "total": 6,
    "successful": 6,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 1,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "orders",
        "_id": "orders_1",
        "_score": 1,
        "_source": {
          "date_ordered": "12.02.2021",
          "@timestamp": "2023-04-04T13:45:00.551797422Z",
          "@version": "1",
          "married_to": null,
          "idorders": 1,
          "total_sum": 1200,
          "user_iduser": 1
        }
      }
    ]
  }
}
```

Sort asc

```
GET _search
{
  "query": {
    "match": {
      "_index": "orders"
    }
  },
  "sort": {
    "idorders": {
      "order": "asc"
    }
  }
}
```


```
{
  "took": 4,
  "timed_out": false,
  "_shards": {
    "total": 6,
    "successful": 6,
    "skipped": 5,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 3,
      "relation": "eq"
    },
    "max_score": null,
    "hits": [
      {
        "_index": "orders",
        "_id": "orders_1",
        "_score": null,
        "_source": {
          "date_ordered": "12.02.2021",
          "married_to": null,
          "idorders": 1,
          "user_iduser": 1,
          "@timestamp": "2023-04-01T14:15
          "total_sum": 1200,
          "@version": "1"
        },
        "sort": [
          1
        ]
      },
      {
        "_index": "orders",
        "_id": "orders_2",
        "_score": null,
        "_source": {
          "date_ordered": "22.02.2022",
          "married_to": null,
```


Sort desc

```
GET _search
{
  "query": {
    "match": {
      "_index": "orders"
    }
  },
  "sort": {
    "idorders": {
      "order": "desc"
    }
  }
}
```


```
{
  "took": 3,
  "timed_out": false,
  "_shards": {
    "total": 6,
    "successful": 6,
    "skipped": 5,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 3,
      "relation": "eq"
    },
    "max_score": null,
    "hits": [
      {
        "_index": "orders",
        "_id": "orders_3",
        "_score": null,
        "_source": {
          "date_ordered": "23.02.2022",
          "married_to": null,
          "idorders": 3,
          "user_iduser": 2,
          "@timestamp": "2023-04-01T14:50:00.210410546Z",
          "total_sum": 1400,
          "@version": "1"
        },
        "sort": [
          3
        ]
      },
      {
        "_index": "orders",
        "_id": "orders_2",
        "_score": null,
        "_source": {
          "date_ordered": "22.02.2022",
          "married_to": null,
```

Appendix

schema.sql

```
DROP TABLE IF EXISTS order_item;
DROP TABLE IF EXISTS orders;

CREATE TABLE IF NOT EXISTS orders (
  idorders bigint,
  user_iduser bigint,
  married_to bigint,
  total_sum decimal(10,2) NOT NULL,
  date_ordered varchar(15) NOT NULL,
  PRIMARY KEY (idorders)
);

CREATE TABLE IF NOT EXISTS order_item (
  orders_idorders bigint,
  product_idproduct bigint,
  quantity decimal(10,2) NOT NULL,
  price decimal(10,2) NOT NULL,
  PRIMARY KEY (orders_idorders,product_idproduct)
);
```

data.sql

```
INSERT INTO orders (idorders, user_iduser, total_sum,
   date_ordered) VALUES (1, 1, 1200, '12.02.2021');
INSERT INTO orders (idorders, user_iduser, total_sum,
   date_ordered) VALUES (2, 2, 1400, '22.02.2022');

INSERT INTO order_item (orders_idorders, product_idproduct,
    quantity, price) VALUES (1, 1, 2, 500.00);
INSERT INTO order_item (orders_idorders, product_idproduct,
    quantity, price) VALUES (1, 2, 1, 200.00);
INSERT INTO order_item (orders_idorders, product_idproduct,
    quantity, price) VALUES (2, 3, 2, 500.00);
INSERT INTO order_item (orders_idorders, product_idproduct,
    quantity, price) VALUES (2, 2, 2, 200.00);
```

logstash.conf

```
input {
  jdbc {
    jdbc_connection_string => "jdbc:postgresql
       ://192.168.0.23:5432/postgres"
    jdbc_user => "postgres"
    jdbc_password => "geheim"
    jdbc_driver_class => "org.postgresql.Driver"
  statement => "SELECT * FROM orders where idorders > :
     sql_last_value"
  tracking_column => "idorders"
  use_column_value => true
  tracking_column_type => "numeric"
  schedule => "*/5 * * * *"
  }
}

output {
  elasticsearch {
    hosts => "elasticsearch:9200"
  user => "elastic"
  password => "changeme"
  index => "orders"
  document_id => "orders_%{orders_idorders}"
  doc_as_upsert => true
  }
}
```

.env

```
ELASTIC_VERSION=8.6.1

ELASTIC_PASSWORD='changeme'
LOGSTASH_INTERNAL_PASSWORD='changeme'
KIBANA_SYSTEM_PASSWORD='changeme'
METRICBEAT_INTERNAL_PASSWORD='changeme'
FILEBEAT_INTERNAL_PASSWORD='changeme'
HEARTBEAT_INTERNAL_PASSWORD='changeme'
MONITORING_INTERNAL_PASSWORD='changeme'
BEATS_SYSTEM_PASSWORD='changeme'
```