# Kibana/ElasticSearch


- Prerequisites
- Defining our Kibana Services
- Starting and Stoping Kibana
- Connect to our Kibana Instance
- Loading Sample Data
    - Pushing directly to Elasticsearch
    - Using JQ for JSON manipulation
    - Manipulating ElasticSearch data
        - Examples
    - Using the TCP Input Plugin
    - Using the HTTP Input Plugin
- References


## Prerequisites

- Docker and cocker-compose is required.

## Defining our Kibana Services

**Write our docker-compose file for kibana 6.2.3**

> *vi docker-compose.yml*

**docker-compose.yml**

**docker-compose.ym**

```
version: "3.3"
services:
  elasticsearch:
    container_name: elasticsearch6
    image: docker.elastic.co/elasticsearch/elasticsearch:6.2.3
    hostname: elasticsearch
    environment:
      - discovery.type=single-node
      - "ES_JAVA_OPTS=-Xms512m -Xmx1024m"
      - ELASTIC_PASSWORD=changeme
      - bootstrap.memory_lock=true
    volumes:
      - ./data:/usr/share/elasticsearch/data
    ports:
      - 9200:9200
      - 9300:9300

  logstash:
    container_name: logstash6
    image: docker.elastic.co/logstash/logstash:6.2.3
    hostname: logstash
    volumes:
      - ./pipeline/:/usr/share/logstash/pipeline/
    ports:
      - 9600:9600
      - 5400:5400
      - 3000:3000
    depends_on:
      - elasticsearch

  kibana:
    container_name: kibana6
    image: docker.elastic.co/kibana/kibana:6.2.3
    hostname: kibana
    environment:
      - ELASTICSEARCH_URL=http://elasticsearch:9200
    ports:
      - 5601:5601
    depends_on:
      - elasticsearch
```

**Define our logstash pipeline**

> *mkdir pipeline*
>
> *vi pipeline/logstash.conf*

**logstash.yml**

```
input {
 tcp {
    port => 5400
    codec => json
 }
 http {
    id => "http"
    port => 3000
 }
}
output {
  stdout {
    codec => rubydebug
  }
  elasticsearch {
    hosts => "elasticsearch:9200"
    user => elastic
    password => changeme
  }
}
```

# Starting and Stoping Kibana

See the status of the containers by issuing the following command:

> *docker ps -a*

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|---|---|---|---|---|---|---|
| dcb436ad63ee | docker.elastic.co/kibana/kibana:6.2.3 | "/bin/bash /usr /loca…" | About an hour ago | Up About an hour | 0.0.0.0:5601->5601/tcp | kibana |
| 1b4e8b01e575 | docker.elastic.co/elasticsearch /elasticsearch:6.2.3 | "/usr/local/bin /dock…" | About an hour ago | Up About an hour | 0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp | elasticsearch |
| cf433b12ae8e | docker.elastic.co/logstash/logstash:6.2.3 | "/usr/local/bin /dock…" | About an hour ago | Up About an hour | 5044/tcp, 0.0.0.0:9600->9600/tcp | logstash |

**Start our containers**

> *docker-compose up -d*

**View our logs**

> *docker-compose logs -f*

**Stop our containers**

> *docker-compose stop*

# Connect to our Kibana Instance

> *Browse to http://localhost:5601/*

# Loading Sample Data

## Pushing directly to Elasticsearch

You can push to elasticsearch directly using curl as follows:

> curl -s -XPOST -H "Content-Type: application/json" -H "Cache-Control: no-cache" <ELASTICSEARCH_HOST>:9200/<INDEX>/1 --data-binary <FILE>

example:

> > curl -s -XPOST -H "Content-Type: application/json" -H "Cache-Control: no-cache" localhost:9200/spa/1 --data-binary myfile.json

Sample script

**spa.sh**

```
#!/usr/bin/env bash

SOURCE_URL=http://spa.jmehan.com/info/
ELASTIC_URL=localhost:9200/spa/1
TMP_FILE=spa.tmp


HEADER1="Content-Type: application/json"
HEADER2="Cache-Control: no-cache"


curl -L $SOURCE_URL -o $TMP_FILE
curl -s -XPOST -H "$HEADER1" -H "$HEADER2" $ELASTIC_URL --data-binary @$TMP_FILE


rm $TMP_FILE
```

## Using JQ for JSON manipulation

Merging files

```
jq -s '.[0] * .[1]' file.json file2.json
```

Extracting portions

```
jq '.[0].value file.json
```

Practice your jq using: https://jqplay.org/

## Manipulating ElasticSearch data

Besides adding data to ElasticSearch, you can search it and delete entries

## Examples

**Search**

```
GET myindex/_search
{
    "query": {
        "range" : {
            "temperature" : {
                "lte" : -10
            }
        }
    }
}
```

**Delete By Query**

```
POST myindex/_delete_by_query
{
  "query": {
    "range" : {
            "temperature" : {
                "lte" : -10
            }
    }
  }
}
```

# Using the TCP Input Plugin

We will use the logstash TCP plugin to push JSON data into elasticsearch.

> *vi test.json*

**test.json**

```
{"message":{"someField":"someValue"}}
```

> *nc -c localhost 5400 < test.json*

# Using the HTTP Input Plugin

> *curl -H "content-type: application/json" -XPUT 'http://127.0.0.1:3000/twitter/tweet/1' -d '{*
>
> *"user" : "kimchy",*
>
> *"post_date" : "2009-11-15T14:12:12",*
>
> *"message" : "trying out Elasticsearch"*
>
> *}'*

```
curl -XPUT 'http://127.0.0.1:3000/twitter/tweet/1' -d 'hello'
```

# References

| Reference | URL |
|---|---|
| Kibana | https://www.elastic.co/products/kibana |
| Kibana User Guide | https://www.elastic.co/guide/en/kibana/6.x/index.html |
| Install ElasticSearch on Docker | https://www.elastic.co/guide/en/elasticsearch/reference/current/docker.html |
| Install Kibana on Docker | https://www.elastic.co/guide/en/kibana/current/_pulling_the_image.html |
| Installing Logstash on Docker | https://www.elastic.co/guide/en/logstash/current/docker.html |
| Installing APM on Docker | https://www.elastic.co/guide/en/apm/server/6.2/running-on-docker.html#running-on-docker |
| Using the TCP input filter with logstash | https://stackoverflow.com/questions/35143576/sending-data-to-logstash-via-tcp |
| TCP Logstash input plugin | https://www.elastic.co/guide/en/logstash/5.5/plugins-inputs-tcp.html |
| HTTP Logstash input plugin | https://www.elastic.co/blog/introducing-logstash-input-http-plugin |
| Deploying ELK Stack to the Raspberry Pi | https://logz.io/blog/elk-stack-raspberry-pi/ |