Elastic Stack

Labs

zenika
ARCHITECTURE INFORMATIQUE

Labs *objectives* are:

- Learn how to use Elasticsearch search and aggregation queries using logs as this type of document is central to Elastic stack
- Learn how to parametrize beats and logstash tools to get logs from sources and push them toward Elasticsearch
- Learn how to use the Kibana process to make data visually monitored and analyzed

During this work, you will use *access logs* which are similar to Apache access logs and *application logs* to learn how to deal formatless messages or stacktraces

To do the labs, you will need to get from your trainer:

- The latest `Java 1.8` JRE or JDK
- The `elastic-stack.tar.gz` archive

The `elastic-stack.tar.gz` archive contains all you need for your working session

- The last supported Elasticsearch version
- Resources for Elasticsearch labs bootstrap
- The last supported Logstash version
- Logstash-start.conf
- The last supported Kibana version
- The last supported Filebeat version
- The last supported Metricbeat version
- The last supported Packetbeat version
- Some generated logs
- Some Kibanas exported dashboards
- Templates matching generated logs
- A web application to generate logs: `webapp.jar`
- Courses and labs slides

Bootstrap

1. Copy the `elastic-stack.tar.gz` archive in a disc path which does **not** contain any space character in its full path name
2. Extract its content in that place

3. Change directory to `elastic-stack`

```
/home/user$ cp elastic-stack.tar.gz /home/user/training

/home/user$ cd /home/user/training

/home/user/training$ tar xf elastic-stack.tar.gz

/home/user/training$ cd elastic-stack
```

Java

- Check the installed version of java on your machine

```
/home/user/training$ java -version
```

> If no JRE or JDK is installed or your machine, please install it now

- Check the JAVA_HOME environment variable is set

```
/home/user/training$ echo $JAVA_HOME
```

> If no JAVA_HOME is set, please set it to the JDK or JRE main driectory

```
/home/user/training$ export JAVA_HOME=/usr/lib/jvm/default-java
```

Launching Labs Tools

- You can use tools execution scripts to help you as shown here in those samples

```
# Stop the running elasticsearch if needed, launch a new one in background mode
# and open the elasticsearch welcome page
/home/user/training/elastic-stack$ ./elasticsearch-execution.sh

# Stop Elasticsearch without launching a new one
/home/user/training/elastic-stack$ ./elasticsearch-execution.sh stop

# Stop the running Kibana if needed, launch a new one in background mode
# and open the Kibana welcome page
/home/user/training/elastic-stack$ ./kibana-execution.sh

# Stop Kibana without launching a new one
/home/user/training/elastic-stack$ ./kibana-execution.sh stop

# Execute Logstash in foreground mode to enable stdin
# It will find the directory where all your configuration files are ready to use
/home/user/training/elastic-stack$ ./logstash-execution.sh

# Stopping Logstash is straighforward
/home/user/training/elastic-stack$ ^Ctrl-C
```

```
[...]
```

Dev Tools initiation

- Launch Elasticsearch and Kibana tools
- From the Kibana page, select the `Dev Tools` left pane option
- A Welcome console appears and explain the main available features
- Once read, click the `Get to work` button
- On the editor pane, you can see a generic request to search all documents from Elasticsearch

```
GET _search
{
  "query": {
    "match_all": {}
  }
}
```

- Try the autocompletion on the internal `.kibana` index (kibana created it to store its settings)

```
POST /.kibana/_search
{
  "quer...
```

- Try to learn the keyboard shortcuts by clicking on the `Help` button at the top right of the page

You should be ready now !

Lab 1 - Elasticsearch

Indexing documents

To help you indexing documents, you can find some resources under the `elastic-stack/tps/elasticsearch/resources` directory.

You will find in it four files related to logs:

	Logs Type	JSON formated logs file	Bulk formated commands file to massivelly insert logs
Logs	`access`	`access.json`	`access.bulk.json`
Logs	`application`	`application.json`	`application.bulk.json`

> For instance, in the `access.json` file you will see each line is a `json object` representing a line of log.
>
> We have made this mutation to replace the job `usually done by Logstash` to transform text to json.

- Take the first three following `access.json` file's log objects and index them in Elasticsearch `logs` index with `doc` type

log 1:

```
{
  "message": "127.0.0.1 - - [06/May/2015:11:20:42 +0200] \"GET / HTTP/1.1\" 200 155
  "@version": "1",
  "@timestamp": "2015-05-06T09:20:42.000Z",
  "type": "access",
  "host": "gqa-laptop",
  "path": "/media/OS/Work/Formation/formation-elk/Exercices/workspaces/logs/access_
  "clientip": "127.0.0.1",
  "ident": "-",
  "auth": "-",
  "timestamp": "06/May/2015:11:20:42 +0200",
  "verb": "GET",
  "request": "/",
  "httpversion": "1.1",
  "response": "200",
  "bytes": "1553",
  "durationMs": "108"
}
```

log 2:

```
{
  "message": "127.0.0.1 - - [06/May/2015:11:20:42 +0200] \"GET /angular-material.mi
  "@version": "1",
  "@timestamp": "2015-05-06T09:20:42.000Z",
  "type": "access",
  "host": "gqa-laptop",
  "path": "/media/OS/Work/Formation/formation-elk/Exercices/workspaces/logs/access_
  "clientip": "127.0.0.1",
  "ident": "-",
  "auth": "-",
  "timestamp": "06/May/2015:11:20:42 +0200",
  "verb": "GET",
  "request": "/angular-material.min.css",
  "httpversion": "1.1",
  "response": "200",
  "bytes": "158965",
  "durationMs": "12"
}
```

log 3:

```
{
  "message": "127.0.0.1 - - [06/May/2015:11:20:42 +0200] \"GET /main.css HTTP/1.1\"
  "@version": "1",
  "@timestamp": "2015-05-06T09:20:42.000Z",
  "type": "access",
  "host": "gqa-laptop",
  "path": "/media/OS/Work/Formation/formation-elk/Exercices/workspaces/logs/access_
  "clientip": "127.0.0.1",
  "ident": "-",
  "auth": "-",
  "timestamp": "06/May/2015:11:20:42 +0200",
  "verb": "GET",
  "request": "/main.css",
  "httpversion": "1.1",
  "response": "200",
  "durationMs": "6"
}
```

- Check the documents presence in the `logs` index and look at the answer details
- Observe the Elasticsearch generated `mapping` for the `logs` index and `doc` type

```
GET logs/_mapping
```

- Note that all is mapped as multi-fields mapping the initial field as text AND as keyword
- Take a look at the content format of the `application.bulk.json` file
- Use the `_bulk` api to index all the `application.bulk.json` logs (use the curl shell command)

```
~/training/elastic-stack$ curl -H 'Content-Type: application/json' -XPOST
localhost:9200/_bulk --data-binary  @tps/elasticsearch/resources/application.bulk
.json
```

- Check there really are some logs with an `application *type* field` in the `logs` index

- Try to count each type of logs based on the logs `type` field

> You should have *3* logs with type == `access`
>
> You should have *35* logs with type == `application`

- Observe the Elasticsearch generated `mapping` for the `doc` type again and note the differences

```
GET logs/_mapping
```

- Recreate the `logs` index from scratch

```
DELETE logs

PUT logs
{
  "settings": {
    "number_of_replicas": 0,
    "number_of_shards": 2
  }
}
```

- Add the `access.mapping.json` definition to the `logs` index

```
# With curl
~/training/elastic-stack$ curl -H 'Content-Type: application/json' -XPOST
localhost:9200/logs/_mapping/doc --data-binary  @tps/elasticsearch/resources/
access.mapping.json

# With Kibana
PUT logs/_mapping/doc
{
  "doc": {
    "properties": {
      "@timestamp": {
        "type": "date",
        "format": "strict_date_optional_time||epoch_millis"
      },
      "@version": {
        "type": "text"
      },
      "auth": {
        "type": "text"
      },
      "bytes": {
        "type": "integer"
      },
      "clientip": {
        "type": "keyword"
      },
      "durationMs": {
        "type": "integer"
      },
      "host": {
        "type": "keyword"
      },
      "httpversion": {
        "type": "keyword"
      },
      "ident": {
        "type": "text"
      },
      "message": {
        "type": "text"
      },
      "path": {
        "type": "text"
      },
      "request": {
        "type": "text"
      },
      "response": {
        "type": "integer"
      },
      "timestamp": {
        "type": "text"
      },
      "type": {
        "type": "text"
      },
      "verb": {
        "type": "keyword"
      }
    }
  }
}
```

- Observe the Elasticsearch new `mapping` for the `doc` type again and note the differences
- Re-index `application` and `access` logs

```
~/training/elastic-stack$ curl -H 'Content-Type: application/json' -XPOST
localhost:9200/_bulk --data-binary  @tps/elasticsearch/resources/access.bulk.json

~/training/elastic-stack$ curl -H 'Content-Type: application/json' -XPOST
localhost:9200/_bulk --data-binary  @tps/elasticsearch/resources/application.bulk
.json
```

Search for documents

- Search for all `logs`

> You should have a total count of *46* logs

- Search for all `application typed logs`

> You should have *35* logs with type == `application`

- Search for all `access typed logs`

> You should have *11* logs with type == `access`

- Search for `logs` containing the word `boot` with the `Lucene syntax`

> You should find only two logs matching but why when the following command shows 10 matches ?

```
~/training/elastic-stack$ grep  boot tps/elasticsearch/resources/*.bulk.json |wc -l
10
```

- Try these calls and look at the result

```
POST logs/_analyze
{
  "field": "logger",
  "text":  "org.boot.com"
}

POST logs/_analyze
{
  "field": "message",
  "text":  "org.boot.com /home/boot/com"
}
```

- Search for `logs` containing the words `spring boot` in their `message` field
    - Write it with the `Lucene syntax`
    - Write it with the `JSON syntax`
    - Why do we find the log with the following message: *Initializing Spring embedded WebApplicationContext*?
    - How to get logs with both `spring` **and** `boot` ?
- Search for `logs` in wich `@timestamp` is in this range : [2015-05-05, 2015-05-07 } and sort them with `ascending @timestamp`
- Search for `logs` with an `info level` and a `message` field containing the word `Tomcat`
    - Why do we find nothing with the following query ?

```
  {
      "query": {
        "term":{
          "level":"INFO"
        }
     }
 }
```

- To understand, try another analysis test command :

```
POST logs/_analyze
{
  "field": "level",
  "text":  "INFO"
}
```

- Get a histogram of the durations(`durationMs` field) by a 10ms interval and compute the overall min/max/avg of this field.
- Optional : Give the average duration(`durationMs` field) of the `access typed logs` by response code (`response` field)

Lab 2 Logstash

Start needed tools

```
# In a first terminal
~/training/elastic-stack$ tps/elasticsearch-execution.sh

# In a second one
~/training/elastic-stack$ tps/kibana-execution.sh

# In a third one
~/training/elastic-stack$ tps/metricbeat-execution.sh
```

Please note:

- the `metricbeat setup` command used in the script to setup `kibana dashboards and views` and `kibana and elasticsearch index template` for metricbeat-*
- the metricbeat `output` is set to default value here Elasticsearch at `localhost:9200`

Take a Kibana tour to see default metricbeat setup

- Connect to `localhost:5601`
- Go to the `Management` panel, choose the `Index Pattern` tab and note that the `metricbeat-*` index pattern has been created.
- Select the `metricbeat-*` index pattern
- Select the `@timestamp` field as time filter field
- Go to the `Dev Tools` panel and try this command: `GET _template`. What's important to see
- Go to the `Visualize` panel and see the numerous views created by metricbeat
- Go to the `Dashboard` panel and select the `[Metricbeat System] Host overview` dashboard
-

Start Logstash

```
# In a new terminal
~/training/elastic-stack$ ./logstash-execution.sh
```

Please note:

- The provided file `tps/logstash/config/logstash-start.conf` is used to configure logstash

- This configuration starts with a `StdIn Input` : type some text, hit the `enter key` on the keyboard and look at the `~/training/elastic-stack/logs/console.json.log` file after that

Stop/Start Logstash

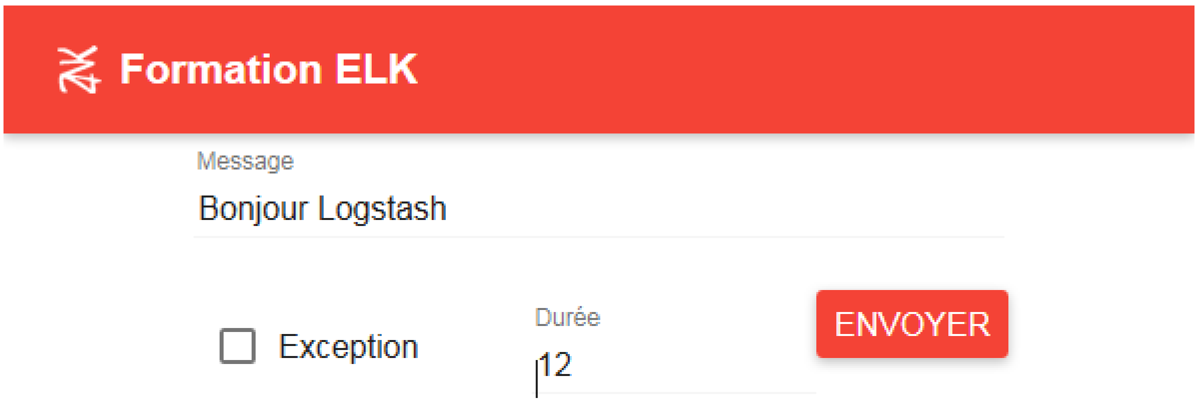
- Stop Logstash with `Ctrl-C`

```
# List the logstash plugins
~/training/elastic-stack$ ./tps/logstash/logstash-<version>/bin/logstash-plugin
 list

# Start logstash again
~/training/elastic-stack$ ./logstash-execution.sh
```

Start the log generation web-app

```
# Start the application and wait for your internet browser to connect to
'http://localhost:8080'
~/training/elastic-stack$ webapp-execution.sh
```



Bonjour Logstash

12

- Complete the form with custom `messages` and validate it multiple times with different `durations` (`Durée` field)

- Note the produced logs in the file `./tps/applicationWeb/logs/application.log`

- Note the produced logs in the file `./tps/applicationWeb/logs/access_log.<year>-<month>-<day>.log`

Configure Logstash to parse access logs

Note the Logstash `--config.reload.automatic` option is used by the execution script to reload configuration as soon as it is changed Note reloading will not work when changing from StdIn input to something else. It is due to the particular console input attachment...

- Start editing the file `tps/logstash/config/logstash-start.conf`

- Replace the StdIn Input with a File Input to read the `<pathto>/tps/applicationWeb/logs/access*.log` file of the web-app and give it the `access` type

- As reloading do not work for this particular step, please stop Logstash
- Change the StdOut output to use the "rubydebug" codec
- Start Logstash again
- Parse each line of the file:
  - The Grok format to use is `%{COMMONAPACHELOG} %{NUMBER:durationMs}`
  - Use the `date` filter to set the `@timestamp` field to the `timestamp` field value. The format to use is `dd/MMM/YYYY:HH:mm:ss Z`
- Do not forget to activate the webapp multiple times during this phase as log lines have already been read by Logstash file input
- Output the result in a file `<pathto>/tps/logs/out-%{type}.json`
- Check the Grok and Date formats are correct (no dateparserfailure or grokparsefailure)
- Use a filter option to suppress the `message` and the `timestamp` fields
- Replace the File Output with the Elasticsearch one
  - Use the `logstash-%{+YYYY.MM.dd}` index name for all of your logs
  - Check your access logs are in Elasticsearch
  - Look at the Logstash generated Index Template in Elasticsearch

```
    GET _template/logstash
```

- Please, ask the trainer for explanations
- Optional : Convert `durationMs` in an `integer` field using the `Mutate Convert` filter

Configure Logstash to parse application logs

- Add a File Input to read the `./tps/applicationWeb/logs/application*.log` file of the web-app and give it the `application` type
- Try to match each line of the file
  - Please use the Grok Debug or Grok Constructor sites to write and test your Grok matching regular expressions
  - The logs format without exception is as follow

```
timestamp level  logger - message
```

- The `timestamp` field is `ISO`
- You could use the predefined `USERNAME [a-zA-Z0-9._-]+` to match the `level` and the `logger` fields

- Do not forget to activate the webapp multiple times during this phase as log lines have already been read by Logstash file input
- Output the result in a file `./tps/logs/out-%{type}.json`
- Check the Grok and Date formats are correct
- Add a filter to suppress the `timestamp` field
- Replace the File Output with the Elasticsearch one

Configure Logstash to parse application exceptions

- Try to parse the special case of `Java stack traces`, when you choose the web-app to generate an exception

Add a Metricbeat Input and redirect metrics to Elasticsearch Output

- Add a `Beats Input` to catch metrics on the `5044 port`
- Modify the `Elasticsearch Output` to enable metric indexing without breaking the existing pipeline indexations
- Stop `metricbeat`
- Change the `metricbeat` configuration file to send datas on Logstash
  - Go to file `<pathto>/tps/metricbeat/metricbeat-<version>-linux-x86_64/metricbeat.yml`
  - Comment out all the Elasticsearch section
  - Enable the logstash output

```
#Sample extract

#================================ Outputs =====================================

# Configure what output to use when sending the data collected by the beat.

#-------------------------- Elasticsearch output ------------------------------

#output.elasticsearch:
  # Array of hosts to connect to.
  # hosts: ["localhost:9200"]

  # Optional protocol and basic auth credentials.
  # protocol: "https"
  # username: "elastic"
  # password: "changeme"

#----------------------------- Logstash output --------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]
```

- Execute `metricbeat` again
- Check you have new metrics in Elasticsearch via this new pipeline flow

Lab3 Kibana

Inject sample logs

- Stop Logstash
- From Kibana `Dev tools` add the following index template

```
PUT _template/agile-wake-up
{
  "order": 0,
  "index_patterns": "agile-wake-up*",
  "settings": {
    "index": {
      "refresh_interval": "5s"
    }
  },
  "mappings": {
    "dynamic_templates": [
      {
        "message_field": {
          "path_match": "message",
          "mapping": {
            "norms": false,
            "type": "text"
          },
          "match_mapping_type": "string"
        }
      },
      {
        "string_fields": {
          "mapping": {
            "norms": false,
            "type": "text",
            "fields": {
              "keyword": {
                "type": "keyword"
              }
            }
          },
          "match_mapping_type": "string",
          "match": "*"
        }
      }
    ],
    "properties": {
      "@timestamp": {
        "type": "date"
      },
      "horodatage": {
        "type": "date",
        "format": "epoch_millis"
      },
      "centralized-by": {
        "type": "keyword"
      },
      "stype": {
        "type": "keyword"
      },
      "protocole": {
        "type": "keyword"
      },
      "protocole-version": {
        "type": "keyword"
      },
      "verbe": {
        "type": "keyword"
      },
      "dureeMs": {
        "type": "long"
      },
      "taille": {
        "type": "long"
      },
      "status": {
        "type": "long"
      },
      "geoip": {
        "dynamic": true,
        "properties": {
          "ip": {
            "type": "ip"
          },
          "latitude": {
            "type": "half_float"
          },
          "location": {
            "type": "geo_point"
          },
          "longitude": {
            "type": "half_float"
          }
        }
      },
      "@version": {
        "type": "keyword"
      }
    }
  },
  "aliases": {}
}
```

- Delete or move your old `logstash-start.conf` file out of the `./tps/logstash/config` directory
- Copy the `./tps/kibana/resources/logstash-agile-wake-up.conf` under the `./tps/logstash/config` directory

- Replace the <path-to> parameters to match your current directory values

```
input {
    file {
        path => "<path-to>/elastic-stack/tps/logs/accesslogs.log"
        sincedb_path => "<path-to>/tps/logstash/accesslogs.db"
        type => "logs"
        start_position => "beginning"
        add_field => { "stype" => "access" }
    }

    beats {
      port => 5044
      add_field => [ "type", "metric" ]
   }
}

filter {

    if [type] == "logs" {

        mutate {
            rename => {"host" => "centralized-by"}
        }

        grok {
          match => [ "message",
          "%{IPORHOST:host}\|%{IPORHOST:clientip}\|%{USERNAME:user}\|%{NUMBER:horod
          remove_field => [ "message" ]
        }
        mutate {
          convert => {
            "taille" => "integer"
            "dureeMs" => "integer"
            "status" => "integer"
          }
        }
        geoip {
          source => "clientip"
          remove_field => [ "clientip"]
        }
      }
}

output {
  if [type] == "metric" {

    elasticsearch {
      hosts => "localhost:9200"
      manage_template => false
      index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
    }
  } else {

    elasticsearch {
      hosts => "localhost"
      index => "agile-wake-up"
    }
  }
  stdout {
    codec => rubydebug {}
  }
}
```

- Start logstash again

Kibana Hands On

- Configure the `agile-wake-up` index from the Kibana management panel
- This index holds access logs only
- From the `Management` panel, choose the `SavedObject` tab and `import` the provided dashboards:
    - /tps/logs/resources/kibana/visualisations-métier.json
    - /tps/logs/resources/kibana/visualisations-techniques.json
- From the `Dashboard` panel create a new Dashboard
- Add in it all the vizualizations for which the name do start with `agility-technical` words
- Save this dashboard under the `technical-agility-dashboard` name
- From the `Dashboard` panel create a new Dashboard
- Add in it all the vizualizations for which the name do start with `agility-` word when not followed by the `technical` word
- Save this dashboard under the `business-agility-dashboard` name
- Do not forget to change the date range at the top right corner of kibana to see datas...
- Now, try to make visualizations looking the same as the ones you can see on these dashboards. For each one, ask yourself:
    - Which are the availables fields and data types to create my visualization
    - Do it need a special initial query
    - What is the needed underlying aggregations to build the visualization
- Try to build a visualization where you show the different encountered status codes occurences per service url:
    - What diagram type suit the best ?
    - What are the aggregations to use ?

- Try to build a visualization like the `Agility - Durée moyenne par provenance` one but restricted to France
- Try to build a query which list services called from France showing:
  - time
  - country name
  - request url
- Add it to your business dashboard
- With the business dashboard, modify filtering criteria to see possibilities
  - Change the date range limits
  - Modify the dashboard query