CERN

**CH-1211 Geneva 23
Switzerland**

CERN TE/Group	TE-VSC-ICM

EDMS Document No.
2279500 v1.0

VACUUM DATA PIPELINE

LOG STREAMING USERGUIDE

Vacuum Data Pipeline

Prepared by :	***Checked by :***	***Approved by :***
N. LUNA		

HISTORY OF CHANGES

REV. NO.	DATE	PAGES	DESCRIPTIONS OF THE CHANGES
1.0	6-11-2019	ALL	Document Creation

TABLE OF CONTENTS

1. Introduction

This document shows the steps needed to be followed for building a log streaming data pipeline. In any case it is focused on explaining further details about log analysis or monitoring, but only a guide for setting up all the infrastructure.

Our pipeline will start reading data from several input log files, and after some processing of this data, users will be able to visualize the streamed messages thanks to a graphical web interface. With this architecture, we will obtain a basic infrastructure which could be used in a future for analysing data traffic in real-time and detecting unexpected situations in the monitoring process.

Specifically, this data pipeline is composed of four items:

- **Filebeat:**
  - It is a centralizing tool able to monitor log files. It is in charge of collecting and shipping messages from a predefined source file to another output entity, in our case, Logstash (but it could also be Elasticsearch, Kafka or an output file among others).
- **Logstash:**
  - It is an open source data collection engine with real-time pipelining capabilities. It is in charge of aggregating the data from an input source, processing it (through filters) and sending it to another output entity, in our case, Elasticsearch. But, as with Filebeat, you could also configure another output entity different from Elasticsearch.
- **Elasticsearch (ES):**
  - It is a distributed real-time search and analytics platform. It is in charge of indexing and storing any type of data, as well as supporting fast searches within the data thanks to the REST API it exposes.
- **Kibana:**
  - It is a platform for analysing data, a web interface designed to work together with Elasticsearch, that is why it is considered as the Elasticsearch front-end. It allows interactively explore, visualize, and share insights into the data.

By having all these elements connected among them, it is easy to see the flow of messages from one endpoint to the other, allowing the user analyse and squeeze the information hidden in the data, which, in many cases, and due to the huge amount of data generated in some scenarios, it is a very difficult task. As additional information, it is good to know that the last three tools, Logstash, Elasticsearch and Kibana belong to what is known as ELK server.

Next, we show in each section how do the aforementioned technologies can be installed, configured and launched, starting from Filebeat and finishing with Kibana. One important point to mention is that even if the data pipeline order is Filebeat – Logstash – ES - Kibana, for letting the pipeline work, we should first launch Logstash and then Filebeat, so that when executing the latter, the former is already up and ready to receive the data. This ordering aspect will be tackled in following sections.

2. Architecture overview

In this section, and before explaining how to deal with the aforementioned technologies, we will show how our data pipeline would look like in a real scenario. In this way, we can have a predefined idea of what we are going to see along the document although in a simplified way, so that it will be easy to understand.

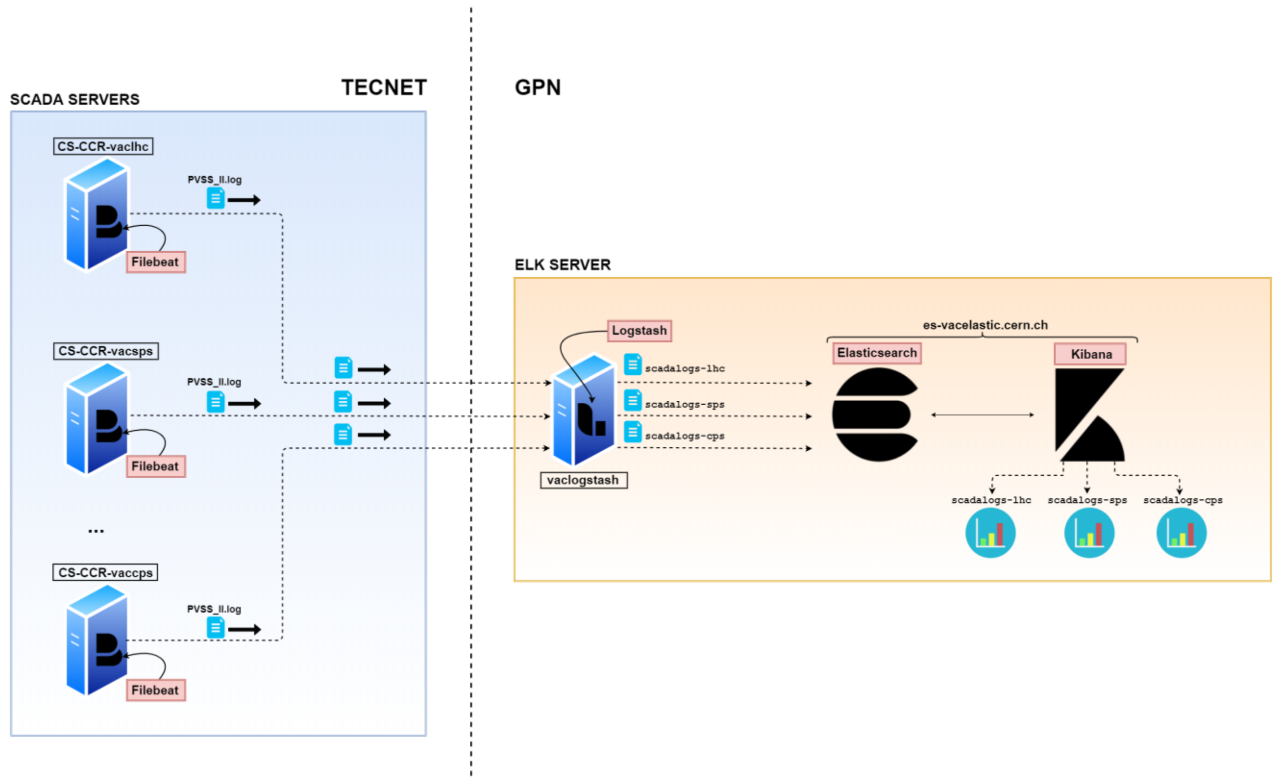


Figure 1 – Data pipeline overview. Simplified scenario

In Figure 1, we can see, from left to right, how data is flowing. First, Filebeat would be installed in each of the servers and will be in charge of collecting the data generated by them and sending the messages to next server, where another tool will be installed: Logstash. This is a simplified view of the whole system, since in production we make use of two Logstash machines to ensure data redundancy if one of the machines goes down.

During this process, messages will be univocally labelled so that when server containing Logstash receives that set of messages all mixed, he will know from which input log file that message comes from.

Hence, depending on that label, Logstash will redirect the message to one Elasticsearch index or the other (in this case, indexes are `scadalogs-lhc`, `scadalogs-sps`, `scadalogs-cps`). Once data is stored in different indexes in Elasticsearch, we will be able to open Kibana, visualize the data, perform operations or apply filters on it.

Notice that in the case of Elasticsearch and Kibana, we have no physical servers to deal with the data but some services provided by CERN that are ready to use. More details about the complete process will be addressed in the following sections.

3. Filebeat

3.1 Installation

In order to install Filebeat, the following steps should be followed (choose a path different from *home* folder):

1. Install *wget* command on target machine (add *sudo* in front of it in case permissions are needed):

➢ **`[sudo] yum install wget`**

2. Download rpm package:

➢ **`[sudo] wget http://linuxsoft.cern.ch/mirror/artifacts.elastic.co/packages/oss-7.x/yum/7.4.1/filebeat-oss-7.4.1-x86_64.rpm`**

3. Install Filebeat rpm package:

➢ **`[sudo] rpm --install ./filebeat-oss-7.4.1-x86_64.rpm`**

For checking if it has been correctly installed, run the following command and verify you get Filebeat version 7.4.1 as output:

➢ **`[sudo] yum list installed | grep filebeat`**

3.2 Configuration

In order to configure Filebeat, the following step should be followed:

4. Go to /etc/filebeat folder, open filebeat.yml. Just edit what we will explain next, <u>the rest of lines that are not mentioned next in this point and that were already in the file should be commented</u>. Besides, take care of indentation when modifying this file):

Look for the Filebeat inputs section in that file and replace the lines starting from "**`filebeat.inputs:`**" up to "**`#- c:\programdata\elasticsearch\logs`**" (both included) with the following lines:

```
filebeat.inputs:
# Each - is an input. Most options can be set at the input level, so
# you can use different inputs for various configurations.
# Below are the input specific configurations.
- type: log
```

```
  # Change to true to enable this input configuration.
  enabled: true
  # Paths that should be crawled and fetched. Glob based paths.
  paths:
  # SET HERE THE FILE PATH TO STREAM (in this case, folder /tests has been created, but you can set another path for your .log file)
    - /tests/PVSS_II.log
  fields:
    app_id: vac_vaclhc_1_scada_logs
# This is another input file to stream:
- type: log
  # Change to true to enable this input configuration.
  enabled: true
  # Paths that should be crawled and fetched. Glob based paths.
  paths:
  # SET HERE THE FILE PATH TO STREAM (in this case, folder /tests has been created, but you can set another path for your .log file)
    - /tests/PVSS_II_sps.log
  fields:
    app_id: vac_vacsps_1_scada_logs
  # Allowing multiline logs:
  multiline.pattern: '^WCC'
  multiline.negate: true
  multiline.match: after
```

In the previous code block, we have established two different log input sections (starting each with the statement `- type: log`). Inside each section, we can see how a different source file is defined in the **`paths`** (`PVSS_II.log` and `PVSS_II_sps.log`). In another scenario, you could also have added several input sections, or even set only one input section whose **`paths`** field includes several files to read at once, all with the same extension (e.g - `/var/log/*.log`).

Below, you can see the **`fields`** field. This is a way to define a univocal identifier for every streamed log, and afterwards, letting Elasticsearch classify those messages into indexes according to that specific field. You can set whatever name in that field parameter, but then, be sure that such name is the same as the one in section 4.2, point 7 (`if [fields][app_id] == "<field name>"`) . More details will be shown along this document.

To finish the second input block, we can see some lines that allow receiving multiline logs. Those configured lines help Filebeat know the incoming message is composed of several lines so that he will not take them as individual different logs but all lines together as a single input message. That configuration shows that consecutive lines that do not match the pattern (starting with "WCC") will be appended to the previous line that does match.

Look for the output section of Filebeat in *filebeat.yml* and configure it as follows:

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

In that previous piece of code, we are just pointing the host name to which we want to connect (in our case, a Logstash server), together with the port in which Logstash will listen (5044 is used for stream SCADA applications, while 5045 is used for streaming web applications, e.g vacDM, vacMON, vacCC). Reached that point, we have already configured Filebeat. Here again, say that in production scenario, we will need to add a second host in the output section to ensure redundancy:

```
hosts: ["vaclogstash:5044", "vaclogstash2:5044"]
```

3.3 Execution

As already mentioned in section 1, ordering is very important when launching the different elements in the data pipeline. Concretely, Filebeat must be executed once Logstash is up and running. For this reason, we will skip this step here and continue with the tutorial, and after setting everything for running Logstash in section 4, we will be able then to start Filebeat.

4. Logstash

4.1 Installation

In this point, we should move to another production server so that we can independently deal with Logstash. Again, choose a different path from *home* folder to follow the next steps:

1. Install *Java 8* on target machine (add *sudo* in front of it in case permissions are needed):
   - `[sudo] yum install java-1.8.0-openjdk`
2. Install *wget* command:
   - `[sudo] yum install wget`
3. Download *rpm* package:
   - `[sudo] wget http://linuxsoft.cern.ch/mirror/artifacts.elastic.co/packages/oss-7.x/yum/7.4.1/logstash-oss-7.4.1.rpm`
4. Install Logstash *rpm* package:
   - `[sudo] rpm --install logstash-oss-7.4.1.rpm`

For checking if it has been correctly installed, run the following command and verify you get Logstash version 7.4.1 as output:

➢
```
[sudo] yum list installed | grep logstash
```

4.2 Configuration

In order to configure Logstash, the following steps should be followed:

5. Go to */etc/systemd/system*, and JUST IN CASE you don't see a file named *logstash.service* in that directory, run the following from your home directory (then file *logstash.service* should automatically be generated in */etc/systemd/system*):

➢
```
[sudo] /usr/share/logstash/bin/system-install /etc/logstash/startup.options systemd
```

6. Go to */etc/default*, and JUST IN CASE you don't see a file named *logstash* in that directory, create it with that same name and with the following content:

```
LS_HOME="/usr/share/logstash"

LS_SETTINGS_DIR="/etc/logstash"

LS_PIDFILE="/var/run/logstash.pid"

LS_USER="logstash"

LS_GROUP="logstash"

LS_GC_LOG_FILE="/var/log/logstash/gc.log"

LS_OPEN_FILES="16384"

LS_NICE="19"

SERVICE_NAME="logstash"

SERVICE_DESCRIPTION="logstash"
```

7. Create a configuration file for Logstash, called *logstash.conf* and place it in */etc/logstash/conf.d*:

A sample configuration file will be the following:

```
input {

        beats {

                port => 5044

        }

}

output {

        if [fields][app_id] == " vac_vaclhc_1_scada_logs " {

                elasticsearch {
```

```
                hosts => ['https://es-vacelastic.cern.ch:443/es']

                user => 'vacelastic'

                password => 'XXXXXXXXX'

                cacert => "/etc/pki/tls/certs/CERN-bundle.pem"

                manage_template => false

                index => "vacelastic-logs-scada-v1-vac_vaclhc_1-%{+YYYY.MM.dd}"

                ilm_enabled => false

            }

        }

        else if [fields][app_id] == " vac_vacsps_1_scada_logs " {

            elasticsearch {

                hosts => ['https://es-vacelastic.cern.ch:443/es']

                user => 'vacelastic'

                password => 'XXXXXXXXX'

                cacert => "/etc/pki/tls/certs/CERN-bundle.pem"

                manage_template => false

                index => "vacelastic-logs-scada-v1-vac_vacsps_1-%{+YYYY.MM.dd}"

                ilm_enabled => false

            }

        }

}
```

In order to explain the previous code, user needs to know how Logstash configuration is internally behaving. Within Logstash, two main parts need to be declared in the configuration file: these are the **input** section and the **output** section (see code above). Optionally, a third section can be added, the **filter** section (but in this case, we have skipped that part). In this way, Logstash will read the data received from the input section, will process it (filter part), and will send it to the determined output.

Hence, if we take a look at the previous configuration, we can see that our input section corresponds to `beats`. Beats refers to a "family name", specifically, to Filebeat, but apart from it, you also have other types of shippers within this family, such as Metricbeat, Heartbeat or Audiobeat.

Regarding the output section, this has been divided into two subparts with an *if-else if* statement, but in both cases, our output is Elasticsearch, which is the next step in the data pipeline. The difference between those two parts is just the index name that was defined (`index =>`). As briefly mentioned before in section 1, Elasticsearch is an indexation engine, so we need to tell him the name of the index where we want to store our data. One important thing to take into account is that the index name should follow the following pattern:

```
vacelastic-logs-<source>-v1-<app_name>-%{+YYYY.MM.dd}"
```

where:

- `<source>` refers to the logs origin (SCADA or web applications). It can take the following values: `scada | web`
- `<app_name>` is the application name itself:
    - Some values (if `<source>` = `scada`) can be: `vac_vaclhc_1 | vac_vacsps_1 …`
    - Some values (if `<source>` = `web`) can be: `vacmon-spring | vacmon-tomcat`

Since we have two input files, we would like Elasticsearch to create two different indexes. But, how can we differentiate between messages coming from one file or the other? Thanks to the **`fields`** field that we explained before in section 3.2: within the *if-else if* statement, if the message identifier is `vac_vaclhc_1_scada_logs`, that specific log will be stored in index `vacelastic-logs-scada-v1-vac_vaclhc_1-%{+YYYY.MM.dd}`, while if the message identifier is `vac_vacsps_1_scada_logs`, associated index will be `vacelastic-logs-scada-v1-vac_vacsps_1-%{+YYYY.MM.dd}`.

Regarding the password, it will be published in the vacuum password repository. Go there and replace it in both `'xxxxxxxxx'` in the code in **7.**.

Say that this is just one of infinite options anyone can test. Logstash is very flexible and can even allow you setting several input and output sections, but for simplicity purposes, we have just defined one input and one output sections in this tutorial.

8. In order to get messages ordered in Kibana, we need to set Logstash to one worker. Go to */etc/logstash* and open *logstash.yml* file. Look for the pipeline workers parameters and replace it with the following:

```
pipeline.workers: 1
```

4.3 Execution

In order to execute Logstash, the following steps should be followed:

9. Open the default port where Logstash listens to incoming Filebeat connections (5044):

```
[sudo] yum install firewalld
[sudo] systemctl start firewalld
[sudo] systemctl enable firewalld
[sudo] systemctl status firewalld
[sudo] firewall-cmd --zone=public --add-port=5044/tcp --permanent
[sudo] firewall-cmd --reload
```

For checking port has been correctly opened, execute the following command and check port 5044 is displayed as an output:

➢ **`[sudo] firewall-cmd --list-ports`**

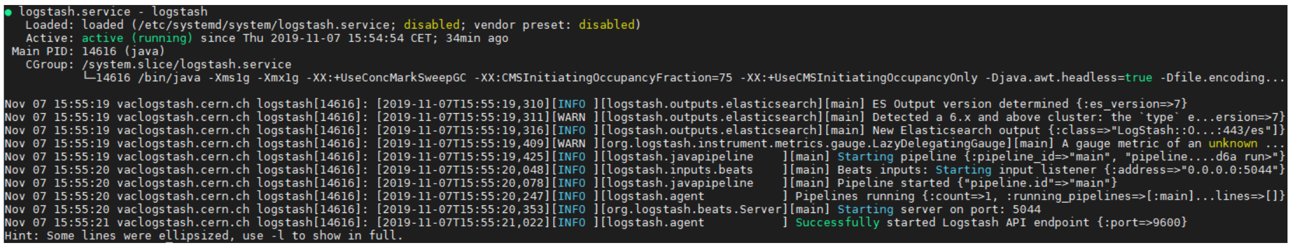
10. Start the service:

➢ **`[sudo] systemctl enable logstash.service`**

➢ **`[sudo] systemctl start logstash.service`**

For checking everything is okey, run and verify that service is "active (running)" as shown in Figure 2:

➢ **`[sudo] systemctl status logstash.service`**



```
● logstash.service - logstash
   Loaded: loaded (/etc/systemd/system/logstash.service; disabled; vendor preset: disabled)
   Active: active (running) since Thu 2019-11-07 15:54:54 CET; 34min ago
 Main PID: 14616 (java)
   CGroup: /system.slice/logstash.service
           └─14616 /bin/java -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyOnly -Djava.awt.headless=true -Dfile.encoding...

Nov 07 15:55:19 vaclogstash.cern.ch logstash[14616]: [2019-11-07T15:55:19,310][INFO ][logstash.outputs.elasticsearch][main] ES Output version determined {:es_version=>7}
Nov 07 15:55:19 vaclogstash.cern.ch logstash[14616]: [2019-11-07T15:55:19,311][WARN ][logstash.outputs.elasticsearch][main] Detected a 6.x and above cluster: the `type` e...ersion=>7}
Nov 07 15:55:19 vaclogstash.cern.ch logstash[14616]: [2019-11-07T15:55:19,316][INFO ][logstash.outputs.elasticsearch][main] New Elasticsearch output {:class=>"LogStash::O...:443/es"]}
Nov 07 15:55:19 vaclogstash.cern.ch logstash[14616]: [2019-11-07T15:55:19,409][WARN ][org.logstash.instrument.metrics.gauge.LazyDelegatingGauge][main] A gauge metric of an unknown ...
Nov 07 15:55:19 vaclogstash.cern.ch logstash[14616]: [2019-11-07T15:55:19,425][INFO ][logstash.javapipeline    ][main] Starting pipeline {:pipeline_id=>"main", "pipeline....d6a run>"}
Nov 07 15:55:20 vaclogstash.cern.ch logstash[14616]: [2019-11-07T15:55:20,048][INFO ][logstash.inputs.beats    ][main] Beats inputs: Starting input listener {:address=>"0.0.0.0:5044"}
Nov 07 15:55:20 vaclogstash.cern.ch logstash[14616]: [2019-11-07T15:55:20,078][INFO ][logstash.javapipeline    ][main] Pipeline started {"pipeline.id"=>"main"}
Nov 07 15:55:20 vaclogstash.cern.ch logstash[14616]: [2019-11-07T15:55:20,247][INFO ][logstash.agent           ] Pipelines running {:count=>1, :running_pipelines=>[:main]...lines=>[]}
Nov 07 15:55:20 vaclogstash.cern.ch logstash[14616]: [2019-11-07T15:55:20,353][INFO ][org.logstash.beats.Server][main] Starting server on port: 5044
Nov 07 15:55:21 vaclogstash.cern.ch logstash[14616]: [2019-11-07T15:55:21,022][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
Hint: Some lines were ellipsized, use -l to show in full.
```

Figure 2 – Logstash service status

If you want to see some additional information about Filebeat execution, you can run:

➢ **`[sudo] journalctl -b | grep filebeat | less`**

➢ **`[sudo] journalctl -b | grep filebeat | tail -n 300`**

If you want to see some additional information about Logstash execution, go to */var/log/logstash* and open file *logstash-plain.log*.

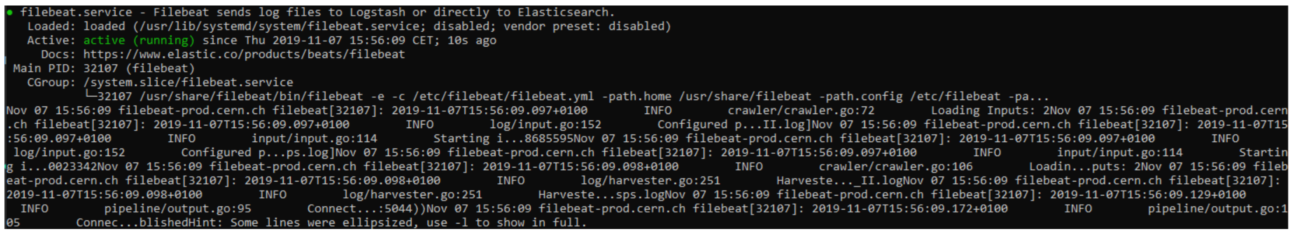
Once reached that point, we have Logstash ready to receive data from Filebeat. But for that happening, we need still one more step to perform: going again to our Filebeat server and run the following (which will start Filebeat as a service):

➢ **`[sudo] systemctl enable filebeat`**

➢ **`[sudo] systemctl start filebeat`**

Verify your Filebeat service is also "active (running)" as shown in Figure 3:

➢ **`[sudo] systemctl status filebeat`**



```
● filebeat.service - Filebeat sends log files to Logstash or directly to Elasticsearch.
   Loaded: loaded (/usr/lib/systemd/system/filebeat.service; disabled; vendor preset: disabled)
   Active: active (running) since Thu 2019-11-07 15:56:09 CET; 10s ago
     Docs: https://www.elastic.co/products/beats/filebeat
 Main PID: 32107 (filebeat)
   CGroup: /system.slice/filebeat.service
           └─32107 /usr/share/filebeat/bin/filebeat -e -c /etc/filebeat/filebeat.yml -path.home /usr/share/filebeat -path.config /etc/filebeat -pa...
Nov 07 15:56:09 filebeat-prod.cern.ch filebeat[32107]: 2019-11-07T15:56:09.097+0100        INFO        crawler/crawler.go:72        Loading Inputs: 2Nov 07 15:56:09 filebeat-prod.cern
.ch filebeat[32107]: 2019-11-07T15:56:09.097+0100        INFO        log/input.go:152        Configured p...II.log]Nov 07 15:56:09 filebeat-prod.cern.ch filebeat[32107]: 2019-11-07T15
:56:09.097+0100        INFO        input/input.go:114        Starting i...8685595Nov 07 15:56:09 filebeat-prod.cern.ch filebeat[32107]: 2019-11-07T15:56:09.097+0100        INFO
 log/input.go:152        Configured p...ps.log]Nov 07 15:56:09 filebeat-prod.cern.ch filebeat[32107]: 2019-11-07T15:56:09.097+0100        INFO        input/input.go:114        Startin
g i...0023342Nov 07 15:56:09 filebeat-prod.cern.ch filebeat[32107]: 2019-11-07T15:56:09.098+0100        INFO        crawler/crawler.go:106        Loadin...puts: 2Nov 07 15:56:09 fileb
eat-prod.cern.ch filebeat[32107]: 2019-11-07T15:56:09.098+0100        INFO        log/harvester.go:251        Harveste..._II.logNov 07 15:56:09 filebeat-prod.cern.ch filebeat[32107]:
2019-11-07T15:56:09.098+0100        INFO        log/harvester.go:251        Harveste...sps.logNov 07 15:56:09 filebeat-prod.cern.ch filebeat[32107]: 2019-11-07T15:56:09.129+0100
  INFO        pipeline/output.go:95        Connect...:5044))Nov 07 15:56:09 filebeat-prod.cern.ch filebeat[32107]: 2019-11-07T15:56:09.172+0100        INFO        pipeline/output.go:1
05        Connec...blishedHint: Some lines were ellipsized, use -l to show in full.
```

Figure 3 – Filebeat service status

In this moment, if all steps have been followed, you should be ready to move forward to the next step in the pipeline: we have just connected the first two items of our data pipeline.

In case you want to stop both executions, you just need to run in each machine:

- **`[sudo] systemctl stop filebeat`**
- **`[sudo] systemctl stop logstash`**

5. Elasticsearch

Any user concerning about log analysis would like to know some way to visualise his data, either statistically or graphically. This can be easily done with Kibana, the Elasticsearch front-end, but we need first to check whether indexes have been correctly created by Elasticsearch. As already mentioned, this tutorial does not cover a log study, but provides a baseline infrastructure for anyone to start afterwards learning all the functionalities that Elasticsearch and Kibana offer, and consequently, being able to carry out certain data study.

For checking the indexes creation, open the following link in the browser and check that specifically, indexes **`vacelastic-logs-scada-v1-vac_vaclhc_1-%{+YYYY.MM.dd}`** and **`vacelastic-logs-scada-v1-vac_vacsps_1-%{+YYYY.MM.dd}`** are listed there apart from the default created ones:

- https://es-vacelastic.cern.ch/es/_cat/indices?v

In case it asks you the credentials when opening that link, type **`'vacelastic'`** as user and the same password that was described in section 4.2, point 7. If you are able to find in that list the name of the index we have just configured, then you are ready to see your logs in Kibana.

6. Kibana

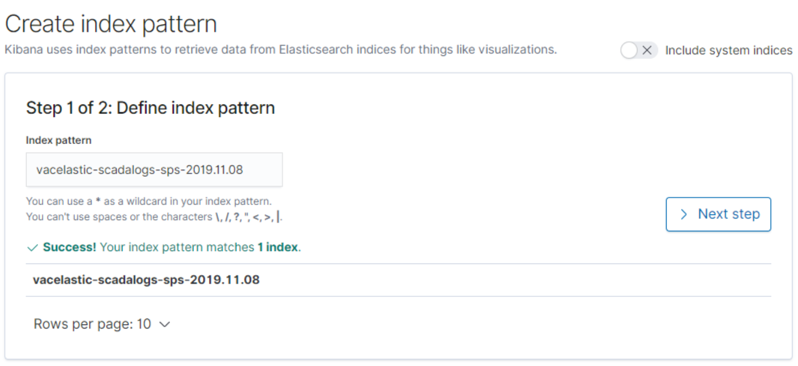
This is the last step in our data pipeline. Once data is already stored in indexes as explained in section 5, we just need to create what is called an "index pattern" in Kibana to finally visualize all logs.

We will provide a link to connect to Kibana service so that users are able to create such index pattern (see steps below):

➢ https://es-vacelastic.cern.ch/kibana_rw/app/kibana#/home?_g=()

Steps to follow for creating an index pattern:

1. Open the link (takes a bit). Main Kibana web page will be loaded
2. Click on "Management" button (last option in the left-side menu)
3. Click on "Index Patterns"
4. Click on "Create index pattern" (blue box)
5. In the "Index pattern" search box, type in the complete index name created in 4.2, point 7 (Figure 4 shows just an example, type yours according to the index name)


Create index pattern
Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Include system indices

Step 1 of 2: Define index pattern

Index pattern: vacelastic-scadalogs-sps-2019.11.08

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

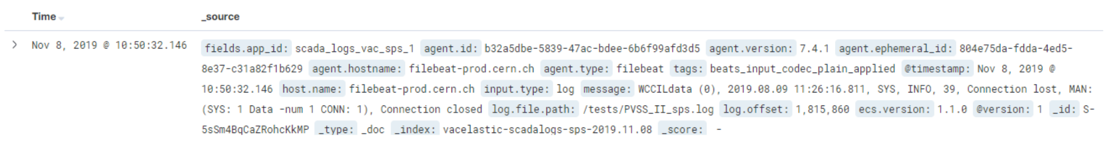
✓ Success! Your index pattern matches 1 index.

vacelastic-scadalogs-sps-2019.11.08

Rows per page: 10

Figure 4 – Creating index pattern in Kibana

6. Click in "Next step"
7. Select *@timestamp* within the "Time Filter field name" menu
8. Click on "Create index pattern"
9. Click on "Discover" button (compass symbol on left-side menu)
10. In the left drop-down menu, select the name of the created index pattern. Just according to Figure 4, this should be **`vacelastic-scadalogs-sps-2019.11.07`**. Just after that, you should be able to see something similar as Figure 5 shows:



```
Time        _source
>  Nov 8, 2019 @ 10:50:32.146  fields.app_id: scada_logs_vac_sps_1 agent.id: b32a5dbe-5839-47ac-bdee-6b6f99afd3d5 agent.version: 7.4.1 agent.ephemeral_id: 804e75da-fdda-4ed5-
8e37-c31a82f1b629 agent.hostname: filebeat-prod.cern.ch agent.type: filebeat tags: beats_input_codec_plain_applied @timestamp: Nov 8, 2019 @
10:50:32.146 host.name: filebeat-prod.cern.ch input.type: log message: WCCILdata (0), 2019.08.09 11:26:16.811, SYS, INFO, 39, Connection lost, MAN:
(SYS: 1 Data -num 1 CONN: 1), Connection closed log.file.path: /tests/PVSS_II_sps.log log.offset: 1,815,860 ecs.version: 1.1.0 @version: 1 _id: S-
5sSm4BqCaZRohcKkMP _type: _doc _index: vacelastic-scadalogs-sps-2019.11.08 _score: -
```

Figure 5 – Sample log sent to Kibana

In case you do not see nothing, it is possible that you should set a correct time window and not the default one. Go then on the right part of the screen and in the search box (see Figure 6) select a wider time range, e.g "This year":

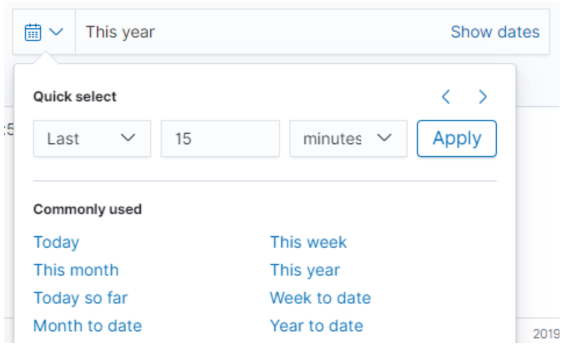


Figure 6 – Changing time range in Kibana

You will be able to see the messages sent from the input file stated in Filebeat as well as information about each log file.

ANNEX

How to re-index data

In this section, we will learn how to move data from an old Elasticsearch index to a new one. This is very useful when we want to create a new index with a different mapping, but we still want to keep old data to visualize it all together (a mapping defines the way the data will be indexed in ES: fields, data types…). This would not be possible unless we re-index the old data (which has some mapping) to the new index (which has a new mapping). Because if we do not re-index and we try to visualize both old and new data in Kibana, the index pattern will be "corrupted" as it is composed of two different mappings, so we will not be able to see any graph.

Therefore, the steps to re-index are:

1. **Define a template (mapping).** An example of it can be the following:

```
{
    "order" : 4,
    "template" : "vacelastic-logs-web-v1-*",
    "settings" : {
      "index.refresh_interval" : "5s"
    },
       "mappings": {
      "properties" : {
        "@timestamp" : {
          "type" : "date"
        },
        "fields" : {
          "properties" : {
            "app_id" : {
              "type" : "keyword"
            }
          }
        },
        "host" : {
          "properties" : {
            "name" : {
              "type" : "keyword"
            }
          }
        },
               "message" : {
                 "type" : "keyword"
```

```
                        }
                    }
                }
            }
```

We can establish an **`order`** depending on the priority of the template. The higher the number, the later the template is applied. That is, if you want to ensure that any other template is overwritten, increase that number.

After the order, we can define the set of indexes to which our template is going to be applied (this is specified in the **`template`** field).

Below this, we can find the **`properties`** section. There is where you manually set the data types you want your fields to have. In our case, we would like them to be of type **`keyword`** so that we can aggregate in Kibana and make some dashboards.

Say that in order to make use of this template for the newly created indexes, it should be pushed to the following folder in the vacelastic repository: https://gitlab.cern.ch/it-elasticsearch-project/endpoint-vacelastic-settings/tree/master/templates. Remember that **only** indexes created **after** pushing the template will consider that template.

2. **Re-index:**

For re-indexing, we need to make some **`curl`** query (**2.b**), but before this, we need to follow some previous steps:

a. Set up destination index (new index):

i. Set a new index name in both machines:

Connect to **`vaclogstash`** and **`vaclogstash2`** machines and in **`/etc/logstash/conf.d`** respectively, open the corresponding configuration file to change the name of the index (web apps or SCADA).

ii. Restart Logstash services in both machines with:

```
sudo systemctl restart logstash
```

iii. Wait and see that new indexes are created:

(go to https://es-vacelastic.cern.ch/es/_cat/indices?v, enter credentials and see if the new index is listed there)

iv. Check the mapping of that new index with:

```
curl -u vacelastic:<password> -X GET "https://es-vacelastic.cern.ch/es/<new_index_here>/_mapping?pretty"
```

Check that such mapping (data types…) corresponds with the one you defined in **1.**.

v. Delete old index pattern in Kibana (if it was created)

b. Make query _reindex:

```
curl -u vacelastic:<password> -X POST -k "https://es-vacelastic.cern.ch:443/es/_reindex?pretty" -H 'Content-Type: application/json' -d '
{
"source": {
"index": "<old_index_here>"
},
"dest": {
"index": "<new_index_here>"
}
}'
```

If everything went well, you will see something like this as output:

```
...
"took" : 6530,
"timed_out" : false,
"total" : 31081,
"updated" : 0,
"created" : 31081,
"deleted" : 0,
"batches" : 32,
"version_conflicts" : 0,
"noops" : 0,
"retries" : {
  "bulk" : 0,
  "search" : 0
},
"throttled_millis" : 0,
"requests_per_second" : -1.0,
"throttled_until_millis" : 0,
"failures" : [ ]
```

Where "`created`" means the number of messages been transferred from the old index to the new one.

If you refresh this previous page now, https://es-vacelastic.cern.ch/es/_cat/indices?v, you will see that your new index contains now more logs than before. If this happens, the re-index has been correctly performed.

3. Delete old indexes with:

```
curl -u vacelastic:<password> -XDELETE https://es-vacelastic.cern.ch/es/<old_index_here>
```