BUILDING EFFECTIVE DATABASE BACKUP AND RECOVERY MONITORING USING ELASTIC STACK

September 2017

AUTHOR:
Yasmine Nasri

SUPERVISOR:
Sebastien Masson

CERN IT-DB Group

CERN openlab Summer Student Report 2017

PROJECT SPECIFICATION

The data produced at CERN is stored into different tremendous Oracle databases and these databases concern mainly administrative, accelerators and experiments data. Due to the importance of this data, CERN implements a full backup and recovery system to protect data against a loss and be able to recover it.

The aim of this project is to find an efficient way to handle all activity of the complex and recovery system at CERN to enable pro-active monitoring. Moreover, the challenge is to publish logs from Oracle Database Backup and Recovery system to the log analysis solution and extract potentially useful patterns or information from them.

ABSTRACT

The data produced at CERN is critical for both physics and administrative purposes. These data must be reachable and available at every moment. To ensure its availability, the CERN IT-DB group implements a full backup and recovery system to protect data against a loss and be able to recover it.

In order to monitor the backup and recovery activities and be able to detect eventual errors, a huge number of logs are generated. Those logs must be analyzed to provide us knowledge about the health of the system.

In this project, we would like to test the Elastic stack solution to analyze the generated logs so as to monitor the backup and recovery system.

In this report, we outline the process we followed to test the Elastic stack solution in the CERN case after defining some theoretical concepts about backup and recovery system and about log analysis.

TABLE OF CONTENTS

- Part I -

INTRODUCTION AND BACKGROUND OF THE STUDY

"If you are unable to understand the cause of a problem it is impossible to solve it."
— Naoto Kan (1946)[1]

.

[1] ***Naoto Kan*** *is a Japanese politician, and former Prime Minister of Japan.*

Chapter 1

INTRODUCTION

1.1 Project context

The volume of data produced at CERN presents a considerable processing challenge. This data is stored into different tremendous Oracle databases and these databases concern mainly administrative, accelerators and experiments data. Due to the importance of this data, CERN implements a full backup and recovery system to protect data against a loss and be able to recover it.

The figure below shows the main sources of data present at CERN.

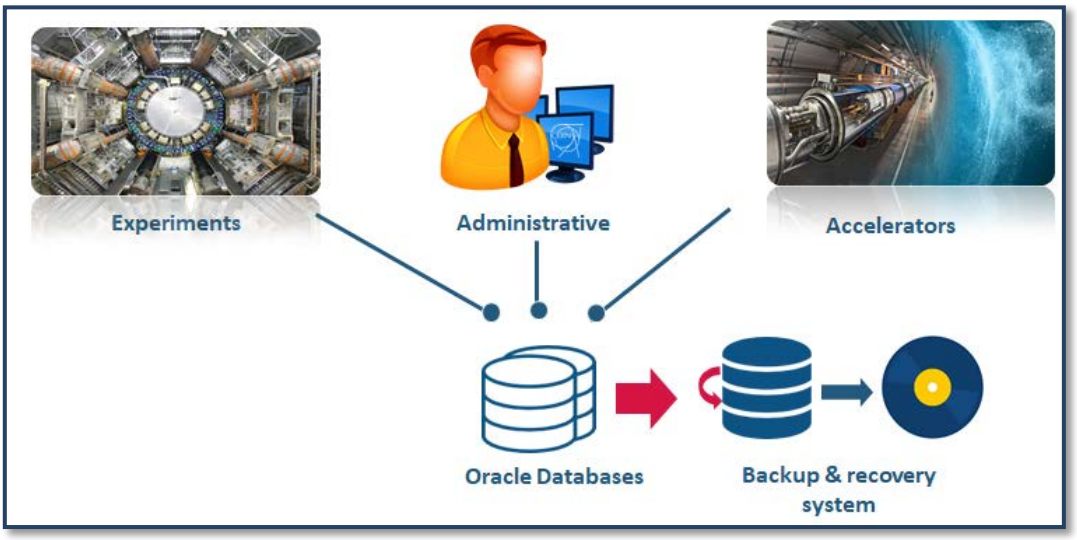

Figure 1 – Data source at CERN

In order to monitor the backup and recovery activities and be able to detect eventual errors, a huge number of logs are generated. Those logs must be analyzed to provide us knowledge about the health of the system.

The current system of backup and recovery at CERN is made of software developed at CERN which stands on Oracle Recovery Manager (RMAN). The logs generated are related to the operations of backup and recovery using RMAN and are sent via emails in case of errors. Those log errors allows the IT-DB members to react immediately and try to solve the errors occurred.

1.2 Problem definition

After analyzing the current backup and recovery system present at CERN, we detected some weaknesses, like:

- Difficulties to analyze a large volume of logs ;
- Loss of time in the analysis of those logs ;
- Loss of information ;
- Too many notifications via email ;
- Monitoring difficulties.

1.3 Objectives

To address the weaknesses of the current system, we fixed the following objectives:

- ✓ Analyze log errors and extract knowledge from it ;
- ✓ Improve proactiveness and then be able to react in real time ;
- ✓ Provide a smart alert system instead of the emails notifications ;
- ✓ Provide statistics of backup and recovery activities.

1.4 Organization of this rapport

We have organized this report in four parts:

Part II: *Review of related literature*, where we give an overview about backup and recovery concepts and log analysis solutions.

Part III: *The selected solution*, we describe the chosen solution.

Part IV: *Technical implementation*, we talk about the setting up and the usage of our solution.

Part V: *Conclusion and future work*, we finish this report with some conclusions and future work.

- Part II -

REVIEW OF RELATED LITERATURE

" He who loves practice without theory is like the sailor who boards ship without a rudder and compass and never knows where he may cast."
*— **Leonardo da Vinci (1452 – 1519)*** [2]

- **Chapter 2:** Backup and Recovery
- **Chapter 3:** Log Analysis.

[2] ***Leonardo da Vinci*** *was a leading artist and intellectual of the Italian renaissance.*

Chapter 2

BACKUP AND RECOVERY

The Backup and recovery concept refers to the theory and practice of protecting a real-life database against data loss and recovering data after a loss. We can lose data either because of a technical problem such a media failure or because of errors made by users. [Kuhn and al., 2007]

In this chapter, we give a first overview of the backup and recovery concepts and then we focus on the backup and recovery system used at CERN.

2.1 Definitions

Database backup *is a way to protect and restore a database. It is performed through database replication and can be done for a database or a database server. Typically, database backup is performed by the RDBMS or similar database management software. Database administrators can use the database backup copy to restore the database to its operational state along with its data and logs. The database backup can be stored locally or on a backup server. [Technopedia, 2017]*

Data recovery *is the process of restoring data that has been lost, accidentally deleted, corrupted or made inaccessible. In enterprise IT, data recovery typically refers to the restoration of data to a desktop, laptop, server or external storage system from a backup. [Rouse, 2012]*

2.2 Causes of data loss

Most data loss is caused by human error, rather than malicious attacks, according to U.K. statistics released in 2016. In fact, human error accounted for almost two-thirds of the incidents reported to the U.K. Information Commissioner's Office. The most common type of breach occurred when someone sent data to the wrong person. [Rouse, 2016]

Other common causes of data loss include power outages, natural disasters, equipment failures or malfunctions, accidental deletion of data, unintentionally formatting a hard drive, damaged hard drive read/write heads, software crashes, logical errors, firmware corruption, continued use of a computer after signs of failure, physical damage to hard drives, laptop theft, and spilling coffee or water on a computer. [Rouse, 2016]

2.3 Data Backup types

There are three main types of backups: Full, incremental and differential. In the following, we present each of them. [Kuhn and al., 2007]

2.3.1 Full backups

A full backup is exactly what the name implies. It is a full copy of your entire data set. Although full backups arguably provide the best protection, most organizations only use them on a periodic basis because they are time consuming, and often require a large number of tapes or disk.

2.3.2 Incremental backups

Because full backups are so time consuming, incremental backups were introduced as a way of decreasing the amount of time that it takes to do a backup. Incremental backups only backup the data that has changed since the previous backup

2.3.2.1 Differential incremental backups

A differential backup is similar to an incremental backup in that it starts with a full backup, and subsequent backups only contain data that has changed. The difference is that while an incremental backup only includes the data that has changed since the previous backup, a differential backup contains all of the data that has changed since the last full backup.

2.3.2.2 Cumulative incremental backups

A Cumulative backup is a backup that is scheduled by the administrator on the master server and backs up files that changed since the last successful full backup. All files are backed up if no prior backup was done.

2.4 Data Recovery types

There are many types of recovery, in this research; we focus on two types that are used at CERN. [Kuhn and al., 2007]

2.4.1 Complete recovery

We perform a complete recovery when we bring a database, a tablespace, or a data file up-to-date with the most current point in time possible. It's important to emphasize that complete recovery isn't synonymous with recovering the complete database. Rather, completeness here alludes to the completeness of the entire database or part of it (tablespace or data file) with reference to the time element. If we update the database tablespace or data file completely by applying all changes from the archived redo logs to the backup files, we're performing a complete backup. [Kuhn and al., 2007]

2.4.2 Point-in-Time recovery

Following a loss of a disk or some other problem, the complete recovery of a database will make the database current by bringing all of its contents up to present. *A point-in-time recovery*, also known as *incomplete recovery*, brings the database to a specified time in the past.

2.5 Backup and recovery system at CERN

In order to protect its data against loss, CERN implements a full system of backup and recovery using oracle Recovery Manager (RMAN).

In the figure below, we can observe the architecture of the current system of backup used at CERN.

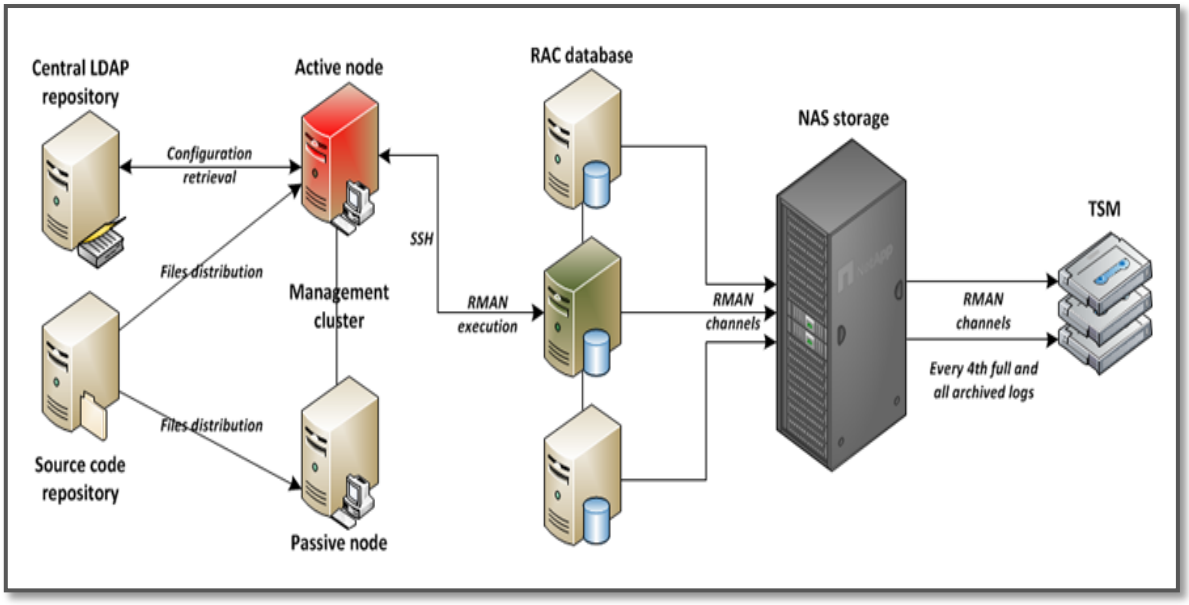

Figure 2 – Architecture of backup used at CERN

The backup system consists of several machines. The main one is the management cluster that triggers all the backup jobs on every database. The scheduler is a simple CRON that launches the execution of RMAN scripts on database hosts through SSH. To be build and executed, those RMAN scripts needs information on the CERN infrastructure that is logically defined in a central LDAP repository. Every backup job creates a set of backup files that are stored on disk or tape storage.

On recovery side, CERN uses four servers to launch automatic recoveries. Those tests are a part of the overall backup strategy. They are performed to test as many backups as possible to be sure that any database could be recovered if needed.

Chapter 3

LOG ANALYSIS

After defining backup and recovery concepts, we are going, in this chapter, to explain the process of log analysis. Moreover, we present some log analysis solutions by comparing them.

3.1 Definition

Log analysis is the process of transforming raw log data into information for solving problems. The market for log analysis software is huge and growing as more business insights are obtained from logs. Stakeholders in this industry need detailed, quantitative data about the log analysis process to identify inefficiencies, streamline workflows, automate tasks, design high-level analysis languages, and spot outstanding challenges rather than qualitative descriptions and experience alone. [Alspaugh and al., 2014]

3.2 Log analysis Process

The steps for the processing of log analysis are described below: [Voldman, 2001]

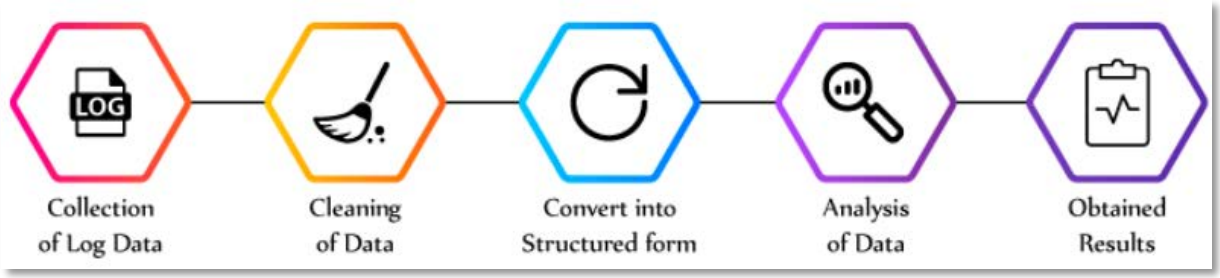

Figure 3 – Log analysis process

- **Collection and cleaning the data**

Firstly, log data is collected from various sources (Databases, logs…). The collected information should be precise and informative as the type of collected data can affect the performance. Therefore, information should be collected from real users. Each type of log contains distinguish the type of information.

Once the log data is arranged in proper manner then, the process of cleaning of data has to be performed. This is because there can be the possibility of the presence of corrupted log data.

- **Structuring the data**

Log data is large as well as complex. Therefore, the presentation of the log data directly affects their ability to correlate with the other data.

An important aspect is that the log data has the ability to directly correlate with the other log data so that deep understanding of the log data can be interpreted by the team members.

- **Analysis of data**

Now, the next step is to analyze the structured form of log data. This can be performed by various methods such as Pattern Recognition, normalization, classification using machine learning, correlation analysis and much more.

3.3 Log analysis tools

Here, we present the most famous log analysis solutions: [Jackson, 2006]

1. **Splunk**

Splunk is a big name in the log and application management space. They have been around since 2003 are no newcomers when it comes to analyzing and monitoring data. They offer great solutions for larger entreprise customers.

2. **Logstash**

Logstash is a free open source tool for managing events and logs. You can use it to collect logs, parse them, and store them for later use. This tool goes hand in hand with both Elasticsearch and Kibana. Using these together can be powerful combination for a log analysis tool.

3. **Sumo logic**

Sumo logic focuses on machine learning for unified logs and metrics to uncover real-time insights into application needs and new customer opportunities. They were founded in 2010 and their cloud-native service analyzes more than 100 petabytes of data per day.

4. **Loggly**

Loggly is a cloud based logging management and analytics service provider founded in 2009. Its main focus is simplicity and ease of use for a devops audience.

5. Graylog

Graylog is an open source log management platform which allows searching, analyzing, and alerting across all log files. Some of their customers include BCBS, eBay, SAP, Cisco, LinkedIn, and Twilio.

6. Comparison between the previous log analysis solutions

The table below compare between the previous log analysis solutions according to some features.

Features	Splunk	Logstash	Sumo Logic	Loggly	Graylog
Plan	Paid	Free	Free and paid	Free and paid	Free
Searching	✓	with elasticsearch	✓	✓	✓
Analysis	✓	with elasticsearch	✓	✓	with integration
Visualization Dashboard	✓	with Kibana	✓	✓	With integration
Monitoring	✓	With X-Pack	✓	✓	With integration
On Premise Setup	✓	✓	X	✓	✓
Plugins & Integration	✓	✓	✓	✓	✓
Input and data type	✓	needs plugins	needs plugins	needs plugins	needs plugins
Customer Support	✓	available; but not proficient	available; but not proficient	available; but not proficient	available; but not proficient
Documentation & Community	✓	✓	X	X	✓

Table 1 – Comparison between log analysis solutions

- Part III -

THE SOLUTION SELECTED

"There is no one giant step that does it. It's a lot of little steps".

*— **Peter A. Cohen (1946)***[3]

- Chapter 4: The Elastic Stack

[3] ***Peter A. Cohen*** *is the former Chairman and Chief Executive Officer of Shearson Lehman Brothers.*

Chapter 4

THE ELASTIC STACK

In this chapter, we are going to talk about the solution we choose for our problem. After comparing between the log analysis solutions in the previous chapter, we finally choose the Elastic stack solution (the one based on logstash). This choice is motivated, in one hand, by the features of Elastic Stack and in the other hand we choose an open source solution. We are going to define each tool of this stack and then we explain the process of log analysis using Elastic Stack.

4.1 Definition

Elastic Stack is a group of open source products from Elastic designed to help users take data from any type of source and in any format and search, analyze, and visualize that data in real time. Elastic Stack can be deployed on premises or made available as Software as a Service (SaaS). [Elastic, 2017]

The Elastic stack contains the following features:

- Distributed restful search ;
- Scale horizontally by adding nodes ;
- Stacked solution with powerful components ;
- Powerful analytics with instant insights ;
- Resistant clusters for security and reliability;
- Full-text search ;
- Document-oriented ;

4.2 Components of the Elastic Stack

Here, we present the five main components of Elastic stack:

- **Beats :** Beats are "data shippers" that are installed on servers as agents used to send different types of operational data to Elasticsearch either directly or through Logstash, where the data might be enhanced or archived.

- **Logstash:** Logstash a data collection engine that unifies data from disparate sources, normalizes it and distributes it. The product was originally optimized for log data but has expanded the scope to take data from all sources.

- **Elasticsearch :** Elasticsearch is a RESTful distributed search engine built on top of Apache Lucene and released under an Apache license. It is Java-based and can search and index document files in diverse formats.

- **Kibana :** Kibana is an open source data visualization and exploration tool from that is specialized for large volumes of streaming and real-time data. The software makes huge and complex data streams more easily and quickly understandable through graphic representation.

- **X-Pack** : X-Pack is an Elastic Stack extension that bundles security, alerting, monitoring, reporting, and graph capabilities into one easy-to-install package.

4.3 Process of log analysis using Elastic stack

Here, we explain the process used by Elastic Stack for log analysis. The figure below shows this process.

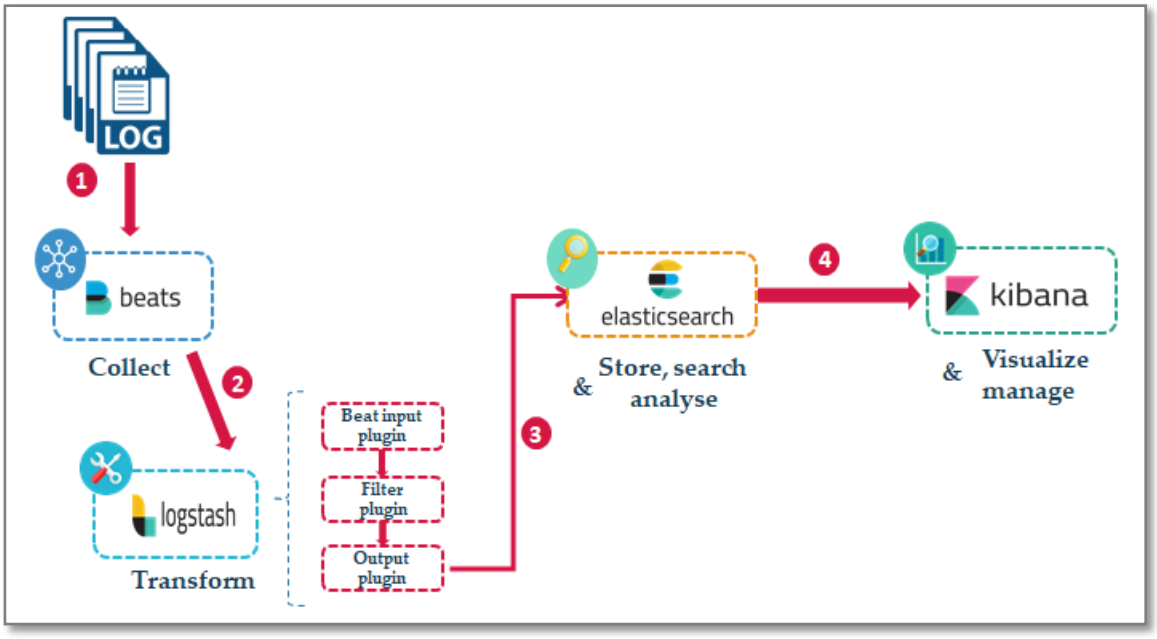

Figure 4 – Log analysis process using Elastic Stack

As we can see in the figure below, we observe first that the logs are collected by Beats. Then, those logs are sent to Logstash which contains three plugins (Input, filter and output). The input plugin get the logs from Beat, transform those logs with the filter plugin and then send it to Elasticsearch through the output plugin. With Elasticsearch, we can store the logs, search it and analyse it. Finally, we can visualize the results of the analysis with Kibana.

- Part IV -

TECHNICAL IMPLEMENTATION

« The best way to escape from a problem is to solve it»

—Brendan Francis (1923 – 1964) [4]

- **Chapter 5:** Setting-up the elastic stack
- **Chapter 6:** Usage and results

[4] ***Brendan Francis*** *was an Irish Republican, poet, short story writer, novelist, and playwright.*

Chapter 5

SETTING-UP THE ELASTIC STACK

In this chapter, we are going to set up the Elastic Stack (Elasticsearch, Kibana, logstash and beats) step by step. We need Java (8 in our case) as a prerequisite.

5.1 Installing Elasticsearch

First, we download elasticsearch 5.2.1 with wget and then install it.

```
$ wget https: //artifacts.elastic.co/downloads//elasticsearch/
  elasticsearch-5.2.1.rpm
  $ rpm -ivh elasticsearch-5.2.1.rpm
```

Elasticsearch is installed. Now, we go to the configuration directory and edit the elasticsearch.yml configuration file.

```
$ cd /etc/elasticsearch/
$ vim elasticsearch.yml
```

We need to enable memory lock for Elasticsearch by removing a comment on line 40. This disables memory swapping for Elasticsearch.

```
bootstrap.memory_lock: true
```

In the « Network » block, we uncomment the network.host and http.port lines.

```
network.host: localhost
http.port: 9200
```

Now, we edit the elasticsearch.service file for the memory lock configuration.

```
$ vim /usr/lib/systemd/system/elasticsearch.service
```

We uncomment LimitMEMLOCK line.

```
LimitMEMLOCK=infinity
```

We edit the sysconfig configuration file for Elasticsearch.

```
$ vim /etc/sysconfig/elasticsearch
```


We uncomment the line 60 and make sure the value is « unlimited ».

```
MAX_LOCKED_MEMORY=unlimited
```

We reload systemd, enable Elasticsearch to start at boot time, and then start the service.

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable elasticsearch
$ sudo systemctl start elasticsearch
```

5.2 Installing Kibana

We download Kibana 5.2.1 with wget and then install it with the rpm command.

```
$ wget https://artifacts.elastic.co/downloads/kibana/kibana-
5.2.1-x86_64.rpm
$ rpm -ivh kibana-5.2.1-x86_64.rpm
```

Now, we edit the Kibana configuration file by uncomment the configuration lines for the server.port, server.host and elasticsearch.url.

```
$ vi /etc/kibana/kibana.yml
server.port: 5601
server.host: "localhost"
elasticsearch.url: "http://localhost:9200"
```

We run Kibana at boot and start it. Kibana will run on port 5601 as node application.

```
$ systemctl enable kibana
$ systemctl start kibana
```

5.3 Installing Logstash

We download Logstash 5.2.1 and then install it with rpm.

```
$ wget https://artifacts.elastic.co/downloads/logstash/
logstash-5.2.1.rpm
$ rpm -ivh logstash-5.2.1.rpm
```

After that, we go to the tls directory and edit the openssl.cnf file.

```
$ cd /etc/pki/tls
$ vim openssl.cnf
```


We add a new line in the '[v3_ca]' section for the server identification.

```
# Server IP Address
subjectAltName = IP:ip_logstash
```

We generate the certificate file with openssl command.

```
$ openssl req -subj '/CN=logstash_server_fqdn/' -x509 -days
365 -batch -nodes -newkey rsa:2048 -keyout
/etc/pki/tls/private/logstash-forwarder.key -out
/etc/pki/tls/certs/logstash-forwarder.crt
```

We create a new 'filebeat-input.conf' file to configure the log sources for filebeat.

```
$ vi /etc/logstash/conf.d/filebeat-input.conf
input {
beats {
 port => 5443
 ssl => true
 ssl_certificate => "/etc/pki/tls/certs/logstash forwarder.crt"
    ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
}}
```

We create also a 'syslog-filter.conf' file for syslog processing.

```
$ vi /etc/logstash/conf.d/syslog-filter.conf
filter {
  if [type] == "syslog" {
    grok {
      match => {"message" =>
"%{SYSLOGTIMESTAMP:syslog_timestamp}
%{SYSLOGHOST:syslog_hostname}
%{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?:
%{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]}
    date {
      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd
HH:mm:ss" ] } }
```

And then, we create the output-elasticsearch.conf' file to define the Elasticsearch output.

```
$ vi /etc/logstash/conf.d/output-elasticsearch.conf
output {
  elasticsearch { hosts => ["localhost:9200"]
    hosts => "localhost:9200"
    manage_template => false
    index =>  "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
    document_type => "%{[@metadata][type]}" }}
```

Finally, we add logstash to start at boot time and start the service.

```
$ systemctl enable logstash
$ systemctl start logstash
```

5.4Installing Filebeat

We dowload Filebeat 5.2.1 and then install it with rpm.

```
$ wget
https://artifacts.elastic.co/downloads/beats/filebeat/filebeat
-5.2.1 x86_64.rpm
$ rpm -vi filebeat-5.2.1-x86_64.rpm
```

Filebeat has been installed; we go to the configuration directory and edit the file 'filebeat.yml'.

```
vi /etc/filebeat/filebeat.yml
paths:    - /var/log/secure
          - /var/log/messages
```

We add a new configuration on line 26 to define the syslog type files.

```
- input_type: log
document_type: syslog
```

Filebeat is using Elasticsearch as the output target by default. We change it to Logshtash. We disable Elasticsearch output by adding comments on the lines 83 and 85.

```
#-------------------Elasticsearch output ------------------
# output.elasticsearch:
```

Now, we add the new logstash output configuration. We uncomment the logstash output configuration and change all value to the configuration that is shown below.

```
#----------------------- Logstash output ---------------------
output.logstash:
  # The Logstash hosts
  hosts: ["ip_ou_fqdn_logstash:5443"]
  bulk_max_size: 1024
  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  ssl.certificate_authorities: ["/etc/ssl/certs/logstash-
forwarder.crt"]
template.name: "filebeat"
template.path: "filebeat.template.json"
template.overwrite: false
```

Finally, we add Filebeat to start at boot time and start it.

```
systemctl enable filebeat
systemctl start filebeat
```

With this last step, we have done the setting up of the Elastic stack, now we can start to use it.

Chapter 6

USAGE AND RESULTS

After setting up the Elastic stack, we are going in this chapter to describe the process we followed for using this stack and the different results we got.

6.1 Structure of logs

Here, we present a sample backup log structure as shown with RMAN.

```
[Thu Jun 22 13:00:28 CEST 2017] level_0F_RESTART_newdisk_noarch accmeas_rac52 itrac5226 ACCMEAS1 ----------------------------------------
[Thu Jun 22 13:00:28 CEST 2017] level_0F_RESTART_newdisk_noarch accmeas_rac52 itrac5226 ACCMEAS1 : rmanGen,v 1.33 2009/08/27 13:23:53 testuser Exp $
[Thu Jun 22 13:00:28 CEST 2017] level_0F_RESTART_newdisk_noarch accmeas_rac52 itrac5226 ACCMEAS1 removing logs : /ORA/dbs01/syscontrol/local/logs/rman/level_EXEC_SNAP.$
tput: No value for $TERM and no -T specified
tput: No value for $TERM and no -T specified
Check for /ORA/dbs01/syscontrol/projects/rman/cmd/rman_templates on oracle@itrac5226
------------------------------------------------------------------------------
secondsBetweenWaitQueue = 1800 sec
secondsBetweenRetries   = 1800 sec
maxWaitQueue            = 3
maxRetries              = 4
parallelConfFile        = /ORA/dbs01/syscontrol/projects/rman/etc/rmanActionsArch01.conf
rmanHost                = itrac5226
tabxmlTsmServer         = TSM521_ORA
remoteTsmServer         = TSM521_ORA
tabxmlTDPONode          = accmeas
remoteTDPONode          = accmeas_ora
[Thu Jun 22 13:00:44 CEST 2017] level_0F_RESTART_newdisk_noarch accmeas_rac52 itrac5226 ACCMEAS1 WARNING: NO pingdb for accmeas_rac52 database.
/ORA/dbs01/syscontrol/bin/rdbms/pingdb rmandb
INFO: Decision making. Who is here: ----------------------
INFO:
-rw-r--r--. 1 0 Jun 22 08:10 /ORA/dbs01/syscontrol/local/logs/rman_status/level_suspended.cmsintr_p52
-rw-r--r--. 1 308 Jun 22 13:00 /ORA/dbs01/syscontrol/local/logs/rman_status/level_0F_RESTART_newdisk_noarch.accmeas_rac52
INFO: I am the only one waiting.
[Thu Jun 22 13:00:50 CEST 2017] level_0F_RESTART_newdisk_noarch accmeas_rac52 itrac5226 ACCMEAS1 Starting Backup
/ORA/dbs01/syscontrol/bin/kssh -kcf -kinit_prefix_fixed /tmp/kssh_backup -kinit_cache_expire 120 -2 -n -x -a -o ConnectionAttempts=1 -o FallBackToRsh=no -o UseRsh=no -$
/ORA/dbs01/syscontrol/bin/kssh -kcf -kinit_prefix_fixed /tmp/kssh_backup -kinit_cache_expire 120 -2 -n -x -a -o ConnectionAttempts=1 -o FallBackToRsh=no -o UseRsh=no -$
Thu Jun 22 13:00:59 CEST 2017 : Main: params passed: $v_params = {
            'BR_TAG' => 'accmeas_rac_20170622T130053_0A',
            'BR_STARTIME' => '2017-06-2213:00:50'
          };

Thu Jun 22 13:00:59 CEST 2017 : Main: BEGIN
Thu Jun 22 13:01:00 CEST 2017 : Main: database_name <ACCMEAS>
Thu Jun 22 13:01:02 CEST 2017 : Main: tsm_server <TSM521_ORA>
Thu Jun 22 13:01:02 CEST 2017 : RunTime.RetrievePasswordForUser: password found for <password_user_cerndb_rman>
Thu Jun 22 13:01:02 CEST 2017 : RunTime.RetrievePasswordForUser: password found for <password_db_cerndb_rman>
Thu Jun 22 13:01:03 CEST 2017 : Main: no rman_channels_connect <-> defined
```

Figure 5 - Structure of a backup log

The complexity in this project is that our logs have a structure that changing several times inside a single log. For example, all logs start with metadata (machine name, volumes

names, TSM server name…) and then we have the RMAN part, with the command blocks and their outputs.

Some of the information we want to get is contained in only one line, whereas RMAN or Oracle errors are split into several lines.

All those characteristics, the changing structure of the logs and the different formats of the messages we want to get have, makes the analysis harder than usual logs files (Java logs,…)

6.2 Collecting logs with beats

As we said before, we collect the logs with beats. Beats send the logs to logstash either by single line or in a block of multiline. In our case, we use the multiline option.

The first approach we tried was to set up several prospectors that mean one prospector per different information. This way we would be able to send logstash very short messages containing exactly the information we are interested in. So logstash would only have to store the message in elasticsearch at the right place.

The thing is that, it's not possible to define several prospectors with the current version of beats. Actually we can set up several prospectors for one filebeat process, but they have to be set on different file locations.

So we need to find another way to get the information we need from our logs.

The only solution we can use, is to set up a multiline pattern that contains several expressions which are separated by OR operators.

The drawbacks of this method are:

- If several events are mixed in the log, we will only raise one single event with filebeat.
- When there is a match, we don't know which of the expressions was recognized.

Moreover, we send with every event, all the information needed to know which log (or backup job) we are talking about. Then logstash will be able to store it in the right index/document.

6.3 Parsing logs with logstash

The logs are sent via beats to logstash in a block of multiline. Logstash parse each log and define the specific fields that will be stored in elasticseach. The figure below shows an example of parsing a single line log into different fields by logstash.

```
grok {
    match => {"message"=>"^\(\[%{WORD:Info}\]\s+)?+
                          (\[%{WORD:Database}\]\s+)?+
                          (\[%{WORD:Entity}\]\s)?
                            %{GREEDYDATA:message}"}
```

Figure 6 – Parsing a single line log with logstash

6.4 Indexing logs into elasticsearch

Once, logstash parse the logs and define the right fields for each information, it indexes it into elasticsearch. In our case, we need two indexes: one for backup and another one for recovery. For example in the backup index, we need to have some metadata and each metadata is stored into specific field as defined by logstash in the previous step. We need also to have RMAN block in specific fields and others errors.

So, each log is stored in a document and split into different fields.

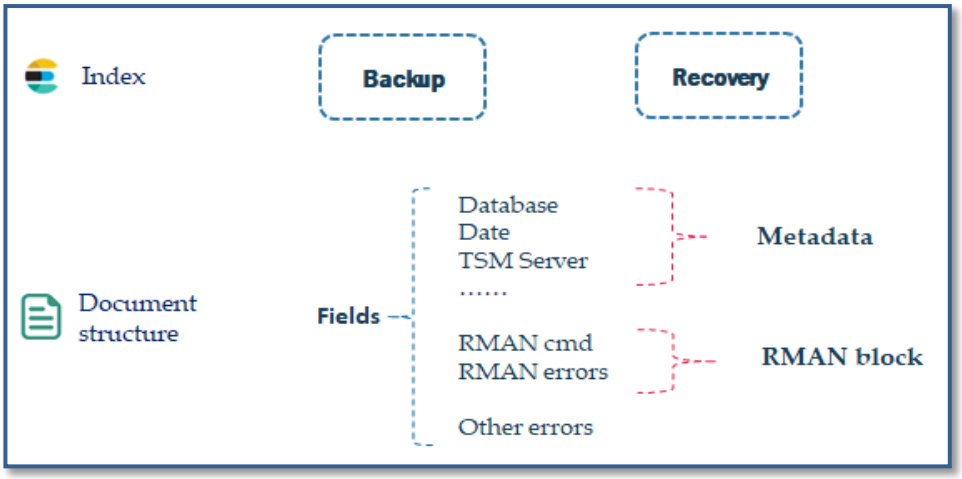

Figure 7 – Indexation structure of logs

6.5 Visualizing logs with kibana

Now, we can visualize our indexes with Kibana. The figure below shows a sample of backup logs as are displayed with kibana.

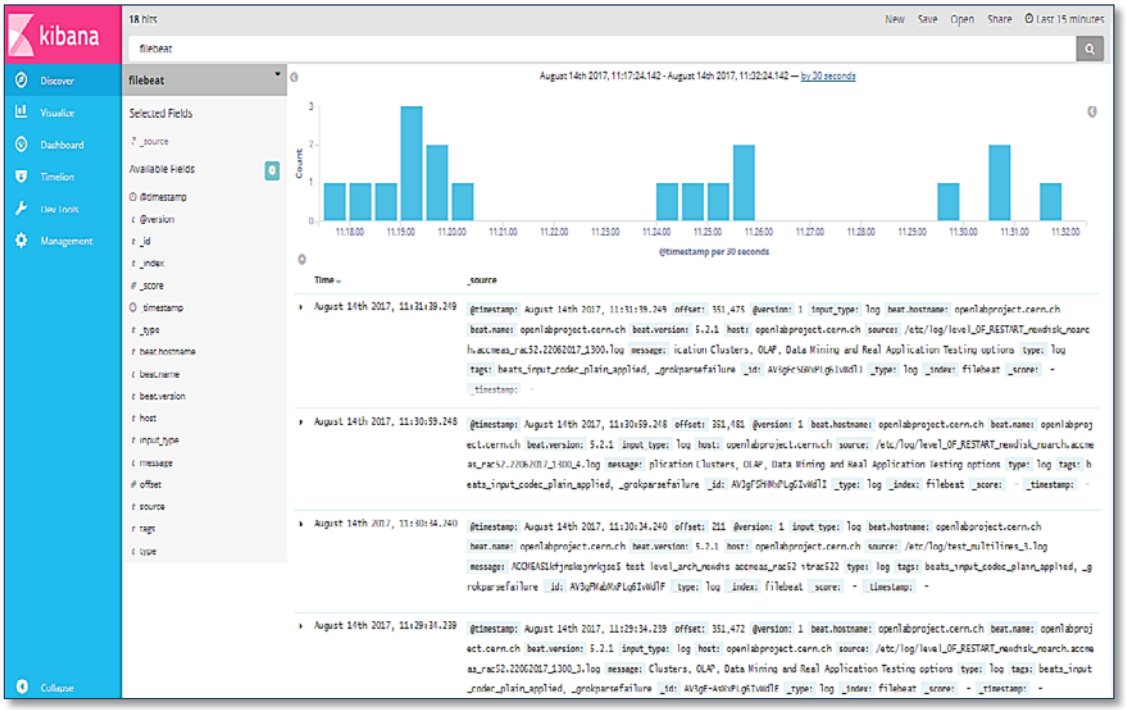

Figure 8 – Visualization of logs with Kibana

- Part V -

CONCLUSION AND FUTURE WORK

"To accomplish great things, we must not only act, but also dream; not only plan, but also believe".
—Anatole France (1844-1924) [5]

[5] ***Anatole France*** *was a French poet, critic, and novelist.*

Chapter 7

CONCLUSION AND FUTURE WORK

The main aim of this project was to test the Elastic stack tools in order to monitor the CERN backup and recovery activities using logs. The typical process begins by collecting the logs with beats, and then parses it with logstash, index it with elasticsearch and finally visualize the logs with kibana.

However, applying this process to the CERN case presents some lacks and this is due to the complex structure of logs generated. So with the current version of the Elastic Stack, it's difficult to get specific information we need from our complex logs.

Through this project, we were able to collect our logs via beats in a multiline way. Then, with logstash, we extracted some metadata information like the timestamp, the name of the database, the name of the entity… Then those logs are indexed into elasticsearch and we could visualize it through kibana.

To complete this project and for future perspectives, we should achieve the following points:

- ✓ Extract the RMAN block and other errors through beats and then send it to logstash to define specific fields for it;
- ✓ Visualize the log analysis results with dashboard;
- ✓ Analyze the logs history and identify some trends;
- ✓ Predict the errors logs with machine learning algorithms.

Finally, this project was really instructive. It allow us, on one hand, to get some knowledge about backup and recovery system used at CERN, the log analysis process and about the Elastic stack tools. And on the other hand, we could work in a great professional environment with database experts.

ACKNOWLEDGMENT

The internship opportunity I had with CERN was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me though this internship period.

First and foremost, I would like to express my sincere gratitude and special thanks to my supervisor "Sebastien Masson" who in spite of being busy with his duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out this project.

I am also thankful to all the IT-DB group members for their warm welcome, their help during this project and for making this experience very pleasant and rewarding.

Lastly, I would like to thank the Openlab team specially Alberto, Kristina, Marion, and Orestis for their excellent organization from the lectures, the trip to Zurich, the CERN's visits to the lighting talks.

.

REFERENCES

BIBLIOGRAPHY

[Alspaugh and al., 2014] Alspaugh, S. Chen, B. Lin, J. Ganapathi, A. Hearst, M. Katz, R. (2014). Analyzing Log Analysis: An Empirical Study of User Log Mining.

[Gormley & Tong, 2015] Gormley, C. Tong, Z. (2015). Elasticsearch: The Definitive Guide.

[Kuhn and al., 2007] Kuhn, D. Alapati, S. Nanda, A. (2007). RMAN Recipes for Oracle Database 11g: A Problem-Solution Approach.

[Paro, 2013] Paro, A. (2013). ElasticSearch Cookbook. [Sharma, 2016] Sharma, C. (2016). Beginning Elastic Stack.

[Turnbull, 2013] Turnbull, J. (2013). The logstash book.

[Voldman, 2001] Voldman, J. (2001). Log File Analysis. PhD thesis, University of West Bohemia in Pilsen.

WEBOGRAPHY

[Elastic, 2017] Elastic. (2017). Learn About the Elastic Stack. Available: www.elastic.co/learn.

[Jackson, 2006] Jackson, B. (2016). Top 10+ Log Analysis Tools – Making Data-Driven Decisions. Available: https://www.keycdn.com/blog/log-analysis-tools/.

[Logz, 2016] Logz. (2016). The Complete Guide to the ELK Stack.
Available: https://logz.io/learn/complete-guide-elk-stack/.

[Oracle, 2017] Oracle. (2017). Getting Started with RMAN.
Available:https://docs.oracle.com/cd/B28359_01/backup.111/b28270/rcmquick.htm#BRADV89346.

[Rouse, 2012] Rouse, M. (2012). Data recovery.
Available: http://searchdisasterrecovery.techtarget.com/definition/data-recovery.

[Technopedia, 2017] Technopedia. (2017). Data backup.
Available: https://www.techopedia.com/definition/29388/database-backup