THE POWER OF PROACTIVE LOG ANALYSIS WITH ELK & ECS

Cristina Alvarez

Denis Jannot
Table of Contents

1. Introduction .................................................................................................................. 3
2. Pre requisites .................................................................................................................. 4
2.1. ELK Stack ................................................................................................................... 4
2.2. Elastic Cloud Storage ............................................................................................... 4
2.3. Network & Load balancer ......................................................................................... 4
3. ECS configuration .......................................................................................................... 5
3.1. Access logs in ECS 3.x ............................................................................................. 5
3.2. Enabling Access Logs in ECS 2.x versions ................................................................ 5
4. Gathering access logs .................................................................................................... 6
5. ELK configuration ......................................................................................................... 7
5.1. Elasticsearch ............................................................................................................. 7
5.2. Kibana ...................................................................................................................... 8
5.3. Logstash .................................................................................................................... 9
5.3.1. Option 1 - Logstash indexing log files locally stored ........................................ 10
5.3.2. Option 2 - Logstash indexing log files gathered using the ecslogs tool ............ 12
5.3.3. Option 3 - Logstash indexing logs received through syslog and storing the data in Elasticsearch ................................................................. 13
5.3.4. Option 4 - Logstash indexing logs received through syslog and storing the data in a S3 bucket 16
5.3.5. Option 5 - Logstash indexing logs from an S3 bucket ........................................ 18
6. Kibana visualisations .................................................................................................... 21
6.1. Accessing Kibana ...................................................................................................... 21
6.2. Kibana Visualizations .............................................................................................. 22
6.2.1. HTTP methods .................................................................................................... 23
6.2.1. HTTP methods over time ................................................................................ 23
6.2.2. Response codes ................................................................................................. 23
6.2.3. Total data transferred per day, per access method ........................................... 24
6.3. Timelion visualisations ............................................................................................ 25
6.3.1. Timelion examples ............................................................................................ 25
6.4. Kibana Dashboards .................................................................................................. 31
6.4.1. Importing/exporting visualizations and dashboards ........................................ 31
6.4.2. Preconfigured dashboard for ECS S3 Access Logs ........................................... 31
7. Conclusion .................................................................................................................... 33
8. References ...................................................................................................................... 34

Disclaimer: The views, processes or methodologies published in this article are those of the authors. They do not necessarily reflect Dell EMC’s views, processes or methodologies.

2018 Dell EMC Proven Professional Knowledge Sharing
1. Introduction

In the Information Technology (IT) world, logs have been traditionally used for troubleshooting. In legacy environments, with a few users and a bunch of servers, having a customer support team going through plain log files and extracting the desired info was something feasible.

But things have changed a lot. Now hundreds or thousands of users and businesses rely on much more complicated apps. These new production environments produce massive files filled with endless lines of text in the form of log files. Spending hours digging through and correlating logs to pinpoint the root cause and gain insight into specifics of what is going on becomes unmanageable. How many man hours are wasted that could be better spent achieving business objectives?

Additionally, the act of manually going through plain log files, grepping all over the place, severely limits the value you can extract from them. Logs can be a lot more powerful than just for troubleshooting. Logs can also be proactively used to measure business metrics (i.e. sales, user behavior, and other product-specific information), to schedule maintenance windows, or to anticipate and prevent failures. Log analysis is becoming a key component in the IT business strategy.

This article explains how to use the Elasticsearch, Logstash and Kibana (ELK) stack for log analysis with Dell EMC Elastic Cloud Storage (ECS). ELK is an open source software that allows collecting logs from a wide variety of systems (different type of logs, in different format) and indexing them into a centralized search engine that provides the ability to quickly run queries and present the data in an interactive dashboard that helps to pinpoint an issue. ELK is a well-known analysis tool that provides high indexing rates, easy scalability, reliability and interoperability with other technologies. That’s why customers could use the ELK deployment not only to analyze ECS logs but also to analyze logs from other systems in their environment.

ECS generates a huge amount of logs. Access logs are especially interesting since they record all the requests processed by the system, including information about the type of request, the response code, user, client, resources involved,… that’s why using ELK with these type of logs is so powerful in terms of troubleshooting, statistics, and proactivity.

This article covers the deployment and use of ELK with ECS, including several scenarios depending on the customer needs. From manually collecting the logs, to automatically doing it with a tool specifically created to enhance the content of the logs, to configuring syslog to collect the ECS data in real-time. It also covers several ways to use the logs once they are collected; they can be directly sent to Elasticsearch for indexing, or they can be stored in an ECS bucket for future analysis. Customers could even use metadata search (unique ECS feature) to index only the logs they want to analyze. And, even better, storing logs in ECS generates a new use case for Dell EMC. The article finishes by explaining how to use Kibana to run high-performance queries on the indexed logs, with examples of valuable visualizations and a pre-designed dashboard that shows the most common statistics.
2. Pre requisites

2.1. ELK Stack

ELK components can be downloaded as Docker containers from http://elastic.co.
This document has been tested for ELK version 5.6.4.

```
# docker pull docker.elastic.co/elasticsearch/elasticsearch:5.6.4
# docker pull docker.elastic.co/kibana/kibana:5.6.4
# docker pull docker.elastic.co/logstash/logstash:5.6.4
```

This document will describe how to deploy these Docker containers. The recommendation is to
deploy each container in a different Virtual Machine (VM), making sure each VM has enough
resources.

Please check the ELK reference [5.6.1] website for more information about supported platforms and
Java (JVM) versions

2.2. Elastic Cloud Storage

Additionally you need Secure Shell (SSH)/Graphical User Interface (GUI) access to an ECS system
to collect the logs from.

This document has been tested with ECS 3.1 version.

2.3. Network & Load balancer

No special network requirements.
Load balancer is not needed.
3. ECS configuration

There is no special ECS configuration to simply collect access logs. However, some of the use cases described in this document interact with ECS to collect logs in real time (rsyslog) or to use ECS as a log repository. How to configure ECS for these purposes is explained in the corresponding 5.x Logstash section.

3.1. Access logs in ECS 3.x

From ECS 3.0, Access Logs are enabled by default in all systems for S3 protocol.

3.2. Enabling Access Logs in ECS 2.x versions

However, prior to the 3.0 version, ECS didn’t generate access logs. They have to be manually enabled.

To check if Access Logs are enabled, run the following command on any ECS node:

```bash
curl -s -H 'Accept:application/json'
http://127.0.0.1:9202/config/com.emc.ecs.objheadsvc.request_log.enabled
```

```
{
  "config" : [ {
    "name" : "com.emc.ecs.objheadsvc.request_log.enabled",
    "description" : "log individual requests",
    "default_value" : "false"
  } ]
}
```

To enable Access Logs in ECS pre-3.0 versions, run the following command on any ECS node:

```bash
curl -v -H 'Accept:application/json' -H 'Content-Type:application/json' -X PUT -d '{"value":true}'
http://127.0.0.1:9202/config/com.emc.ecs.objheadsvc.request_log.enabled
```

You only need to execute this command on one node of each Virtual Data Center (VDC).
4. Gathering access logs

There are several options available to ingest Access Logs in Elasticsearch through Logstash:

- Option 1: Copying the logs manually to the Logstash working directory
- Option 2: Getting all the logs (of one VDC) using the `ecslogs` tool
- Option 3 & 4: Receiving the logs in realtime from the `rsyslog` service running on all the ECS nodes
  - And ingesting them in Elasticsearch [Option 3]
  - Or storing them in an S3 bucket [Option 4]
- Option 5: Getting some logs from the S3 bucket, based on a period of time

Keep in mind that Access Logs are collected in different logs files in ECS, depending on the logs you want to collect:

- S3 current log - `datahead-access.log`
- S3 rotated logs - `dataheadsvc-access.log.<date>.gz`

All of them are located in the same directory in ECS:

```
# <ecs_node>:/opt/emc/caspian/fabric/agent/services/object/main/log/
```

Elasticsearch and Kibana configurations are similar for all the options. However, how to implement Logstash will vary depending on the option selected. This is explained in [section 5.3] of this document.
5. ELK configuration

ELK clusters will be configured in three different phases: 1- *Elasticsearch*, 2- *Kibana*, 3- *Logstash*. These three components provide everything you need to go from raw data to an insightful (and pretty) dashboard in a few easy steps.

Section 5.1 and 5.2 describe how to run Elasticsearch and Kibana.

Section 5.3 describes how to run Logstash depending on the selected log collection method.

5.1. Elasticsearch

Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases. As the heart of the Elastic Stack, it centrally stores your data so you can discover the expected and uncover the unexpected.

You can simply start the corresponding Docker container by executing the command below in one of your VMs:

```
# docker run -d -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node"
docker.elastic.co/elasticsearch/elasticsearch:5.6.4
```

Note that you’ll probably need to change the `max_map_count` kernel value using the following command:

```
# sudo sysctl -w vm.max_map_count=262144
```

Additionally, you’ll probably need to increase the size of the filesystem in the Elasticsearch docker container, to avoid any potential failure caused by the filesystem becoming full. By default, it is a 10GB filesystem; you can increase it to 100GB, for example, modifying the `docker-storage` file in the VM hosting your Elasticsearch container:

```
# vi /etc/sysconfig/docker-storage
DOCKER_STORAGE_OPTIONS=--storage-opt dm.basesize=100G
# service docker stop
# rm -rf /var/lib/docker
# service docker start
```

Note: The previous task will remove your Docker images. Remember that you may need to pull them again.

The default credentials for Elasticsearch are `elastic / changeme`
5.2. Kibana

Kibana lets you visualize your Elasticsearch data and navigate the Elastic Stack, so you can do anything from learning why you’re getting paged at 2:00 a.m. to understanding the impact rain might have on your quarterly numbers.

You can simply start the corresponding Docker container by executing the command below:

```
# docker run -p 5601:5601 -e ELASTICSEARCH_URL=http://xxx.xxx.xxx.xxx:9200 -e XPACK_MONITORING_ENABLED=true docker.elastic.co/kibana/kibana:5.6.4
```

You need to replace xxx.xxx.xxx.xxx by the IP address of the host where you started the Elasticsearch Docker container.

Now, you can connect to [http://yyy.yyy.yyy.yyy:5601](http://yyy.yyy.yyy.yyy:5601) where yyy.yyy.yyy.yyy is the IP address of the host where you started the Kibana Docker container.

You need to use the default Elasticsearch credentials (elastic / changeme).

You should get something similar to what’s displayed in the picture below because no data has been indexed yet.
5.3. Logstash

Logstash is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to Elasticsearch.

Run `setenforce` command in the VM hosting your Logstash Docker container to avoid permission issues:

```
su -c "setenforce 0"
```

There are several options available to ingest Access Logs in Elasticsearch through Logstash:

- **Option 1:** Copying the logs manually to the Logstash working directory
  - Logstash input: local log files
  - Logstash output: `elasticsearch`

- **Option 2:** Getting all the logs (of one VDC) using the `ecslogs` tool
  - Logstash input: local log files collected by `ecstool`
  - Logstash output: `elasticsearch`

- **Option 3:** Receiving the logs in realtime from the `rsyslog` service running on all the ECS nodes and ingesting them in `elasticsearch`
  - Logstash input: ECS `rsyslog` service
  - Logstash output: `elasticsearch`

- **Option 4:** Receiving the logs in realtime from the `rsyslog` service running on all the ECS nodes and storing them in an S3 bucket
  - Logstash input: ECS `rsyslog` service
  - Logstash output: S3 Bucket

- **Option 5:** Getting some logs from the S3 bucket, based on a period of time
  - Logstash input: S3 Bucket
  - Logstash output: `elasticsearch`

For Options 1 & 2, just note that S3 Access Logs are collected in `dataheadsvc-access.log`.

Please note that Option 4 & 5 usually go together, since you store your logs in a repository (Option 4), so that you can select and analyze them later on (Option 5). This repository is an S3 bucket in ECS with MetadataSearch enabled. This is a unique ECS feature that will allow to easily select the logs based on the creation time. Then, you will be able to copy only the logs you want to a analyze to another bucket (Option 5) and run Elasticsearch on it.

Finally, the examples described in this document will use `Grok` as the Logstash filter used to parse unstructured log data into something structured and queryable.
Now, please go to the corresponding section according to the option you want to implement.

5.3.1. Option 1 - Logstash indexing log files locally stored

5.3.1.1. Gathering logs

If logs have been manually retrieved, they have to be copied to the current (working) directory.

For example, from your ELK VM:

```
# scp admin@<ecs_node>:/opt/emc/caspian/fabric/agent/services/object/main/log/dataheadssvc-access.log.20180117-020102.gz .
```

5.3.1.2. Configuring Logstash

Logstash configuration is described in a file called `logstash.conf`, that has to be created in the working directory.

- **logstash.conf template for ECS S3 Access Logs:**

```sh
input {
  file {
    path => "/data/*.log"
    start_position => "beginning"
    ignore_older => 0
  }
}

filter {
  grok {
    match => { "message" => 
    date {
      match => [ "timestamp" , "YYYY-MM-dd'T'HH:mm:ss,SSS" ]
    }
  }
}

output {
  elasticsearch {
    hosts => "http://xxx.xxx.xxx.xxx:9200"
    user => "elastic"
    password => "changeme"
  }
}
```
You need to replace xxx.xxx.xxx.xxx with the IP address of the host where you started the Elasticsearch Docker container.

This file will recognize logs in the current directory, ending in `.log`. If your logs have a different format, please modify this file.

### 5.3.1.3. Starting the Logstash process

Now, you can simply start the corresponding Docker container by executing the command below from your current directory:

```bash
# docker run -it -e xpack.monitoring.enabled=false -v `pwd`:/data
docker.elastic.co/logstash/logstash:5.6.4 bash
```

In the Docker container, run:

```bash
## /opt/logstash/bin/logstash -f /data/logstash.conf
```
5.3.2. Option 2 - Logstash indexing log files gathered using the ecslogs tool

The *ecslogs* tool is a utility developed by Dell EMC that gathers the lines corresponding to the Access Logs from the *dataheadsvc.log* and *dataheadsvc-access.log* files and adds the IP address of the ECS node in each line. This addition allows you to determine if the requests are evenly spread across the ECS nodes.

The *ecslogs* tool is also available as a Docker container. The source code can be found here: https://github.com/djannot/ecslogs

It can also be used in a executable format (the compiled Go binary from Github). The executable can be downloaded from this link.

5.3.2.1. Gathering logs

Here is the syntax you need to use to gather the logs using the *ecslogs* Docker container:

```
echo <password> | docker run -i djannot/ecslogs ./ecslogs <user> <host:port> <pattern> <input log> <# days> <pipe|file> <output file>
```

Or using the *ecslogs* executable:

```
echo <password> | ./ecslogs <user> <host:port> <pattern> <input log> <# days> dynamic pipe | gzip -c <output file>
```

• Ecslogs example for S3 Access Logs

For example, if the password of the SSH admin account is *ChangeMe* and you want to get the access logs from the last 3 days and store them in a local /tmp/dataheadsvc-access.log.gz file:

```
echo "ChangeMe" | ./ecslogs admin xxx.xxx.xxx.xxx:22 . dataheadsvc-access.log 3 dynamic pipe | gzip -c > /tmp/dataheadsvc-access.log.gz
```

You need to replace xxx.xxx.xxx.xxx with the IP address of one ECS node.

Once all logs are gathered, copy them to the working directory.

5.3.2.2. Configuring Logstash

Logstash configuration is described in a file called *logstash.conf*, that has to be created in the working directory.

• *logstash.conf* template for ECS S3 Access Logs:

```
input {
  file {
    path => "/data/*.log"
    start_position => "beginning"
    ignore_older => 0
  }
}
```
filter {
  grok {
    match => { "message" =>
      ["%@IP:ecs_node%@|PROG:log_file|%{TIMESTAMP_ISO8601:timestamp}%{SPACE}%{NOTSPACE:request_id}%{SPACE}%{IP:destination_ip}:%{NUMBER:destination_port}%{SPACE}%{IP:client}:%{NUMBER:source_port}%{SPACE}((?<user>[\^-]*\[^-])*|\^-)%{SPACE}|(?<namespace>[\^-]*\[^-])*|\^-)%{SPACE}|(?<bucket>[\^-]*\[^-])*|\^-)%{SPACE}|(?<key>[\^-]*\[^-])*|\^-)%{SPACE}|(?<query>[\^-]*\[^-])*|\^-)%{SPACE}|HTTP/%{BASE16FLOAT:http_version}%{SPACE}%{NUMBER:response_code}:%{SPACE}%{NUMBER:duration}:%{NUMBER:upload}%{SPACE}|%{NUMBER:download}%{SPACE}"
    }
    date {
      match => [ "timestamp" , "YYYY-MM-dd'T'HH:mm:ss,SSS" ]
    }
  }
}

output {
  elasticsearch {
    hosts => "http://xxx.xxx.xxx.xxx:9200"
    user => "elastic"
    password => "changeme"
  }
}

You need to replace xxx.xxx.xxx.xxx with the IP address of the host where you started the Elasticsearch Docker container.

This file will recognize logs in the current directory, ending in .log. If your logs have a different format, please modify this file.

### 5.3.2.3. Starting the Logstash process

Now, you can simply start the corresponding Docker container by executing the command below from your current directory:

```
# docker run -it -e xpack.monitoring.enabled=false -v `pwd`:/data
docker.elastic.co/logstash/logstash:5.6.4 bash
```

In the Docker container, run:

```
## /opt/logstash/bin/logstash -f /data/logstash.conf
```

### 5.3.3. Option 3 - Logstash indexing logs received through syslog and storing the data in Elasticsearch

Syslog can be configured in ECS to forward alerts, audit messages, and Access Logs in this case, to one or multiple remote syslog servers. This section will describe how to configure it to send the ECS syslog output to Elasticsearch.

#### 5.3.3.1. Gathering logs
Logs will be retrieved directly from *Syslog*, through port 514 (TCP & UCP).

### 5.3.3.2. Configuring Logstash

Logstash configuration is described in a file called *logstash.conf*, that has to be created in the working directory.

To overcome the limitation of non-privileged users (like *logstash*) not able to bind to ports under 1024, the Logstash container will listen at port 514 (ECS *syslog*) and will bind it to port 5000 internally.

- *logstash.conf* template for ECS S3 Access Logs:

```ruby
input {
  tcp {
    port => 5000
    type => syslog
  }
  udp {
    port => 5000
    type => syslog
  }
}

filter {
  grok {
    match => { "message" =>
      "[%{TIMESTAMP_ISO8601:timestamp}%{DATA:request_id}%{IP:ecs_node}:
        %{NUMBER:destination_port}%{IP:client}%{NUMBER:source_port}%{SPACE}((?<user>^[-]([\w\-])[^\s]+)\|\-)%{SPACE}%{WORD:method}%{SPACE}((?<namespace>^[-]([\w\-])[^\s]+)\|\-)%{SPACE}((?<bucket>^[-]([\w\-])[^\s]+)\|\-)%{SPACE}((?<key>^[-]([\w\-])[^\s]+)\|\-)%{SPACE}((?<query>^[-]([\w\-])[^\s]+)\|\-)%{SPACE}((?<http_version>^[\w\-]+)\|\-)%{NUMBER:response_code} %{SPACE}%{NUMBER:duration} %{SPACE}(%{NUMBER:upload} | %{NUMBER:download:in})"
    }
  }
  date {
    match => [ "timestamp", "YYYY-MM-dd'T'HH:mm:ss,SSS" ]
  }
}

output {
  elasticsearch {
    hosts => ["http://xxx.xxx.xxx.xxx:9200"]
    user => "elastic"
    password => "changeme"
  }
}
```

You need to replace `xxx.xxx.xxx.xxx` with the IP address of the host where you started the Elasticsearch Docker container.
### 5.3.3.3. Starting the Logstash process

Now, you can simply start the corresponding Docker container by executing the command below from your current directory:

```
# docker run -it -e xpack.monitoring.enabled=false -v `pwd`:/data
docker.elastic.co/logstash/logstash:5.6.4 bash
```

In the Docker container, run:

```
## /opt/logstash/bin/logstash -f /data/logstash.conf
```

### 5.3.3.4. Configure rsyslog on ECS to send the logs to Logstash

**Note: An RPQ approval is needed to configure ECS to send the logs through rsyslog to the Logstash server in a production environment.**

On each ECS node, you can either modify the default rsyslog configuration located at `/etc/rsyslog.conf`, or create a new file with that configuration in the `/etc/rsyslog.d` directory (all files in that directory are included in the run-time configuration by a directive in the master configuration).

```
# ssh <ecsnodeX>
# cd /etc/rsyslog.d
# sudo vi push-dataheadsvc-log-to-syslog.conf

# cat push-dataheadsvc-log-to-syslog.conf

#$DebugFile /home/admin/rsyslog.debug
#$DebugLevel 2

module(load="imfile" PollingInterval="10") #needs to be done just once

input(type="imfile"
    File="/opt/emc/caspian/fabric/agent/services/object/main/log/datahead-access.log"
    Tag="ecs"
    Severity="info"
    Facility="local7"
    StateFile="ecstosyslog")

action(type="omfwd" Target="xxx.xxx.xxx.xxx" Port="514" Protocol="udp")
```

You need to replace `xxx.xxx.xxx.xxx` with the IP address of the host where you started the Logstash Docker container.

Restart `rsyslog` service in all ECS nodes:

```
# sudo vipreexec "service rsyslog restart"
```
5.3.4. Option 4 - Logstash indexing logs received through syslog and storing the data in a S3 bucket

Syslog can be configured in ECS to forward alerts, audit messages, and Access Logs in this case, to one or multiple remote syslog servers. This section will describe how to configure it to send the ECS syslog output to an S3 bucket in ECS.

5.3.4.1. Gathering logs

Logs will be retrieved directly from Syslog, through port 514 (TCP & UCP).

5.3.4.2. Configuring Logstash

The latest Logstash releases from elastic.co only support AWS buckets in the S3 output plugin list. To overcome that limitation, we will use a modified Logstash container that you can pull from docker.io/djannot/logstash. This way you can use an ECS bucket as the output of Logstash.

# docker pull djannot/logstash

If you want to modify the Docker container used in this section, here is the Dockerfile used to create it:

```dockerfile
FROM ubuntu
RUN apt-get update
RUN apt-get install -y wget openjdk-7-jre unzip
RUN wget https://download.elastic.co/logstash/logstash/logstash-2.3.4.zip -O /tmp/logstash-2.3.4.zip --no-check-certificate
RUN unzip /tmp/logstash-2.3.4.zip -d /opt
RUN rm -rf /tmp/*
ADD run.sh /usr/local/bin/run.sh
RUN chmod +x /usr/local/bin/run.sh
WORKDIR /opt/logstash-2.3.4
RUN bin/logstash-plugin install logstash-codec-gzip_lines
CMD ["/usr/local/bin/run.sh"]
```

Logstash configuration is described in a file called logstash.conf, that has to be created in the working directory.

```json
input {
  tcp {
    port => 5000
    type => syslog
  }
  udp {
    port => 5000
    type => syslog
  }
}
output {
  s3 {
```
A few notes regarding this logstash.conf file:

- To overcome the limitation of non-privileged users (like logstash) not able to bind to ports under 1024, the Logstash container will listen at port 514 (ECS syslog) and will bind it to port 5000 internally.
- The ECS S3 bucket must be created with metadata search enabled for the CreateTime system metadata in order to only index logs for a specific period of time later.
- The time_file parameter corresponds to the number of minutes to wait before uploading the new data to the S3 bucket.
- It's also recommended to specify the endpoint of the load balancer instead of one ECS node (for the proxy_uri option).

### 5.3.4.3. Starting the Logstash process

Now, you can simply start the corresponding Docker container by executing the command below:

```
# docker run -t -i -e xpack.monitoring.enabled=false -v `pwd`:/data djannot/logstash
```

You can use S3 Browser or any other S3 tool to see how the bucket is populated with ECS logs, to verify that everything is working properly.

### 5.3.4.4. Configure rsyslog on ECS to send the logs to Logstash

On each ECS node, you can either modify the default rsyslog configuration located at /etc/rsyslog.conf, or create a new file with that configuration in the /etc/rsyslog.d directory (all files in that directory are included in the run-time configuration by a directive in the master configuration).

```
# ssh <ecsnodeX>
# cd /etc/rsyslog.d
# sudo vi push-dataheadsvc-log-to-syslog.conf
```

```
# cat push-dataheadsvc-log-to-syslog.conf

#$DebugFile /home/admin/rsyslog.debug
#$DebugLevel 2

module(load="imfile" PollingInterval="10") #needs to be done just once

input(type="imfile"
   File="/opt/emc/caspian/fabric/agent/services/object/main/log/datahead-access.log"
   Tag="ecs"

#ssh <ecsnodeX>
# cd /etc/rsyslog.d
# sudo vi push-dataheadsvc-log-to-syslog.conf
```
You need to replace xxx.xxx.xxx.xxx with the IP address of the host where you started the Logstash Docker container.

Restart rsyslog service in all ECS nodes:

```
# sudo viprexec "service rsyslog restart"
```

**5.3.5. Option 5 - Logstash indexing logs from an S3 bucket**

Using an ECS S3 bucket as a repository provides a lot of flexibility to go back to a certain time period and analyze only the logs you are interested in. This can be easily done using metadata search in the ECS S3 bucket used as a repository. By selecting the time period with the `CreateTime` property, only the desired logs are copied to a new bucket that will be used as the input in Elasticsearch.

**5.3.5.1. Create a new bucket containing the logs for the period of time you want to analyze**

`ecss3copy` is a tool developed in Golang to copy objects from one bucket to another one using S3 copy operations. Metadata search queries can also be indicated to select the objects to copy. The `ecss3copy` source code can be found here: [https://github.com/djannot/ecss3copy](https://github.com/djannot/ecss3copy).

Pull the `ecss3copy` Docker container:

```
# docker pull djannot/ecss3copy
```

Create a new bucket in ECS which will be used as the target for the copy, owned by the same object user as the source bucket. Select the timeframe you are interested in, and run the `ecss3copy` copy.

Here is an example to copy the logs created from the 24th and the 26th of January from the `ecslogs1` bucket to the `ecslogs2` bucket:

```
# docker run -it djannot/ecss3copy ./ecss3copy -e http://xxx.xxx.xxx.xxx:9020 -u ACCESSKEY -p SECRETKEY -s ecslogs1 -t ecslogs2 -q "CreateTime>2018-01-24T00:00:00Z and CreateTime<2018-01-27T00:00:00Z"
```

You need to replace xxx.xxx.xxx.xxx with the IP address of one ECS node.

**5.3.5.2. Configuring Logstash**

The latest Logstash releases from [elastic.co](http://elastic.co) only support AWS buckets in the S3 output plugin list. To overcome that limitation, we will use a modified Logstash container that you can pull from `docker.io/djannot/logstash`. This way you can use an ECS bucket as the output of Logstash.

```
# docker pull djannot/logstash
```
To modify the Docker container used in this section, here is the Dockerfile used to create it:

```bash
FROM ubuntu
RUN apt-get update
RUN apt-get install -y wget openjdk-7-jre unzip
RUN wget https://download.elastic.co/logstash/logstash/logstash-2.3.4.zip -O /tmp/logstash-2.3.4.zip --no-check-certificate
RUN unzip /tmp/logstash-2.3.4.zip -d /opt
RUN rm -rf /tmp/*
ADD run.sh /usr/local/bin/run.sh
RUN chmod +x /usr/local/bin/run.sh
WORKDIR /opt/logstash
RUN bin/logstash-plugin install logstash-codec-gzip_lines
CMD ["/usr/local/bin/run.sh"]
```

Logstash configuration is described in a file called `logstash.conf`, that has to be created in the working directory.

- **logstash.conf template for ECS S3 Access Logs:**

```plaintext
input {
  s3 {
    access_key_id => "ACCESS KEY"
    secret_access_key => "SECRET KEY"
    bucket => "BUCKET NAME"
    endpoint => "http://<ECS IP ADDRESS>:9020"
  }
}

filter {
  grok {
  }
  date {
    match => [ "timestamp", "YYYY-MM-dd'T'HH:mm:ss,SSS" ]
  }
}

output {
  elasticsearch {
    hosts => "http://xxx.xxx.xxx.xxx:9200"
    user => "elastic"
    password => "changeme"
  }
}
```
You need to replace xxx.xxx.xxx.xxx with the IP address of the host where you started the Elasticsearch Docker container.

### 5.3.5.3. Starting the Logstash process

Now, you can simply start the corresponding Docker container by executing the command below:

```bash
# docker run -t -i -e xpack.monitoring.enabled=false -v `pwd`:/data djannot/logstash
```

You can use S3 Browser or any other S3 tool to see how the bucket is populated with ECS logs, to verify that everything is working properly.
6. Kibana visualisations

At this stage, logs are indexed and Kibana can be used to interactively analyze and visualize them.

This section will explain how to access Kibana, how to configure it according to the ELK stack and will show examples about visualizations.

6.1. Accessing Kibana

When connecting to Kibana through a web browser, http://yyy.yyy.yyy.yyy:5601, you should get something similar to what’s displayed in the picture below because some data has now been indexed.

![Kibana Interface](image)

Select \texttt{@timestamp} in the Time-field name list and click on \textit{Create}. This \texttt{@timestamp} feature allows keeping the time from the logs in Kibana when filtering by time.

To see the list of variables imported from Logstash, go to \textit{Management -> Index Pattern} tab. These variables, like \texttt{bucket}, \texttt{namespace}, \texttt{user}, \texttt{response_code},... will be used to create Kibana visualizations.

![Index Pattern](image)

A great number of interesting visualisations can now be created using Kibana.

2018 Dell EMC Proven Professional Knowledge Sharing
6.2. Kibana Visualizations

From the Visualize tab you can select the visualization that better fits your requirements.

Select visualization type

<table>
<thead>
<tr>
<th>Basic Charts</th>
<th>Data</th>
<th>Maps</th>
<th>Time Series</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>Heat Map</td>
<td>Horizontal Bar</td>
<td>Line</td>
</tr>
<tr>
<td></td>
<td>Pie</td>
<td></td>
<td>Vertical Bar</td>
</tr>
<tr>
<td>Data Table</td>
<td>Gauge</td>
<td>Goal</td>
<td>Metric</td>
</tr>
<tr>
<td>Coordinate Map</td>
<td>Region Map</td>
<td></td>
<td>Timelion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Visual Builder</td>
</tr>
</tbody>
</table>

This section will go through a few of them.
6.2.1. HTTP methods
Pie visualization, splitting by `method` keyword. It displays a chart with slices representing the amount of requests of each type in the selected timeframe.

6.2.2. HTTP methods over time
Chart visualization of methods over time. Y-axis represents `count`, X-axis represents `time` (@timestamp). Additionally, the graph is split by `method`.

6.2.2. Response codes
Pie visualization, splitting by `response_code` keyword. It displays a chart with slices representing the amount of `response_codes` of each type in the selected timeframe.
6.2.3. Total data transferred per day, per access method

Vertical bar visualization, splitting by method keyword. Y-axis represents count, X-axis represents time (@timestamp). Additionally, the graph is split by method type.

For this one, you need to create a new variable: transferred. Transferred is the sum of uploads and downloads (both variables extracted from the logs using the Grok filter).

To do that, go to [Management – Index Patterns - Scripted field] and click in Add Scripted Field.
• Name: transferred
• Script:

\[
\text{doc['upload'].value > 0 || doc['download'].value > 0 ? doc['upload'].value + doc['download'].value : 0}
\]

Please note that another interesting Scripted field could be the throughput, combining uploads and downloads:
• Name: throughput
• Script:

\[
\text{doc['upload'].value > 0 || doc['download'].value > 0 ? (doc['upload'].value + doc['download'].value) / doc['duration'].value : 0}
\]

6.3. Timelion visualisations
Timelion is a time series composer for Kibana. It allows you to use 2 different Y-axis. It’s very useful to visualize graphs over the time. Go to the Timelion tab in Kibana to run this kind of visualizations.

6.3.1. Timelion examples
• Number of response code 500 vs total number of requests

\[
.es(q='response_code: 500', metric='count:*'), .es(q='*', metric='count:*').yaxis(2)
\]

Note that 500 errors in S3 mean that application is not able to retrieve data from the system (data unavailability).
- Average GET response time according to the average GET download size

\[ \text{es(q='method:GET', metric='avg:duration'), es(q='method:GET', metric='avg:download').yaxis(2)} \]

Sometimes it’s useful to understand that the time needed to retrieve an object is usually proportional to the object size. That explains why response time is higher sometimes in GET requests.

- Average PUT response time according to the average PUT upload size

\[ \text{es(q='method:PUT', metric='avg:duration'), es(q='method:PUT', metric='avg:upload').yaxis(2)} \]
Sometimes it’s useful to understand that the time needed to write an object is usually proportional to the object size. That explains why response time is higher sometimes in PUT requests.

- Average GET + PUT response times according to the average GET download + PUT upload sizes

\[
\text{es(q='method:GET or method:PUT', metric='avg:duration'), es(q='method:GET', metric='avg:download').sum(es(q='method:PUT', metric='avg:upload')).yaxis(2)}
\]

Relationship between the response time and the amount of data involved (PUT / GET).
• Maximum GET download size and PUT upload size

\[\text{Maximum GET download size: } \max \text{ download, method: GET, metric: \ max:download}\], \[\text{Maximum PUT upload size: } \max \text{ upload, method: PUT, metric: \ max:upload}\].

According to this analysis, it is possible to analyze applications patterns. In this case, 500MB is the maximum part size in this application.

• Number of GET request for 500 MB objects:

\[\text{Number of GET request for 500 MB objects: } \\ \text{es(q='method:GET AND download:524288000', metric='count: *')}\]
By knowing the maximum part size on an application, it’s easy to get the multipart download pattern for that application by getting the number of GET request for that part size.

- Average HEAD and DELETE response times vs total number of requests

```
es(q='method:HEAD or method:DELETE', metric='avg:duration'), .es(q='*', metric='count:*').yaxis(2)
```

When customers delete a massive amount of data, they can get a higher response time. A DELETE requests usually involves a HEAD request, to see if the object exists, and a DELETE request to actually delete it. This graph shows the relationship between both requests and response time.

- Average response time vs total number of requests:

```
es(metric='avg:duration'), .es(metric='count: *').yaxis(2)
```
To have a global view of the response time behavior in a system, it’s useful to compare how it is related with the total number of requests. They should be proportional. This could be helpful to identify peaks and time, and more deeply investigate why that peak happened in that period of time.
6.4. Kibana Dashboards

All the described visualizations can be integrated in a dashboard that can be exported and used in other ELK clusters.

6.4.1. Importing/exporting visualizations and dashboards

You can import/export visualizations and dashboards from the Management – Saved Objects page:

Dashboards are usually in Jason format.

6.4.2. Preconfigured dashboard for ECS S3 Access Logs

Here is a dashboard created from some of the visualisations described in the previous sections that can be interesting to analyze the behavior of S3 applications in ECS.

You can get the dashboard from this link.
It includes visualizations for:

- Total number of requests processed with that ELK cluster
- Methods over the time
- Pie with method type percentages
- Total data transferred per day, per access method
- Pie with Response Codes percentages
- Sum GET response time vs sum GET download size
- Sum PUT response time vs sum PUT upload size
- Number of errors vs total number of requests
- Average throughput (B/ms) for GET requests larger than 1 MB
- Average throughput (B/ms) for PUT requests larger than 1 MB

And three important tables on the right:

- Top 100 buckets
- Top 100 users
- Top 100 buckets

These three tables allow the user to filter all the visualizations by bucket, user or client (or other criteria) by just clicking in the selected element. That selection will apply to all the visualizations in the dashboard.

This facilitates troubleshooting and generating personalized reports.

This dashboard can be also modified to add/remove other visualizations by editing it. This also offers a good starting point to analyze ECS Access Logs, understand application patterns and facilitate troubleshooting.
7. Conclusion

This document describes how to configure Elasticsearch, Logstash and Kibana to analyze ECS Access logs.

As the main benefits of this proposal, we highlight that it facilitates the daily work of Dell EMC customer support with faster troubleshooting, doing the heavy lifting when it comes to identifying the root cause of issues occurring in ECS, and interactively generating graphs and statistics that help to reaffirm the message to the customer. Additionally, it also helps Dell EMC identify weaknesses in the customer design or environment, and proactively suggest changes that could improve the solution performance and the relationship with the customer.

In conclusion, ELK analysis on ECS Access Logs provides a powerful tool that Dell EMC can use internally to save resources and time in troubleshooting, but also externally, as a proactive approach to provide feedback, generate trust and improve customer experience.
8. References


[7] – ecslogs tool – Executable - https://publiclinks.object.ecstestdrive.com:443/ecslogs?AWSAccessKeyId=130753079869627056@ecstestdrive.emc.com&Expires=1579949253&Signature=Cz2C6Y1VH1subJSD%2BBBEHa5OPrUM%3D


Dell EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” DELL EMC MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying and distribution of any Dell EMC software described in this publication requires an applicable software license.

Dell, EMC and other trademarks are trademarks of Dell Inc. or its subsidiaries.