

Visualizing SA Meta Data

Using Elasticsearch and Kibana

Helmut Wahrmann

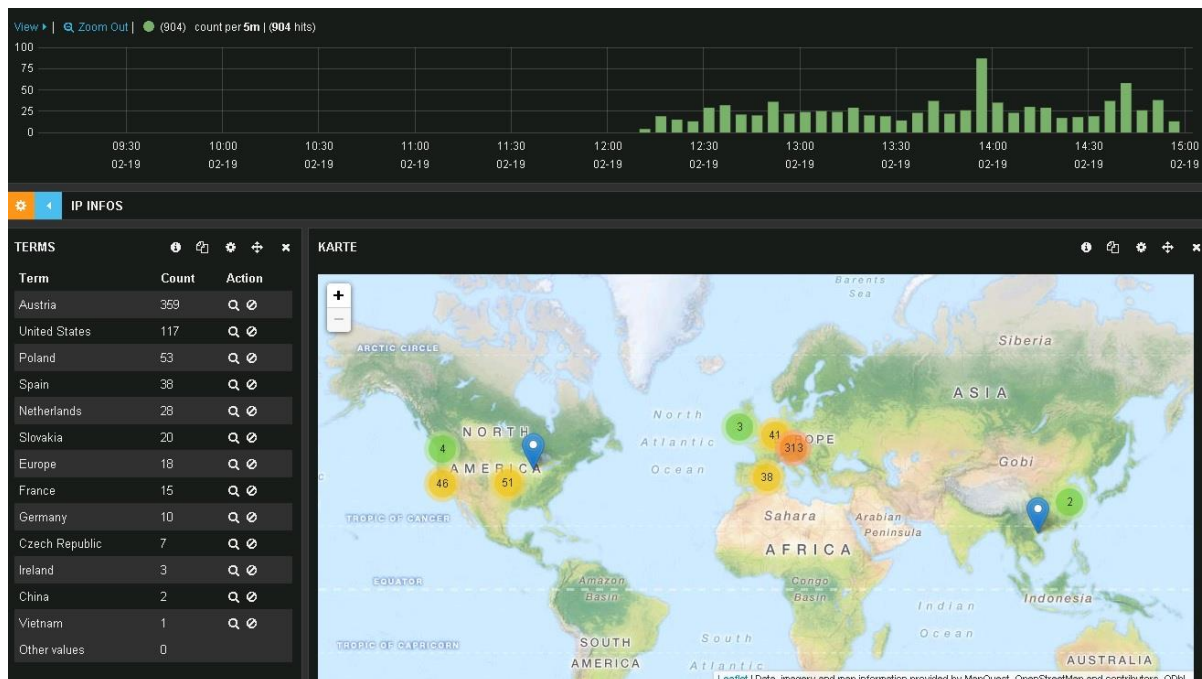


Table of Content

Contents

- Summary 2
- Installation..... 3
 - Install Java 3
 - Install Elasticsearch 3
 - Install Apache Flume 5
 - Modify Java home and Memory..... 5
 - Get additional Java libs..... 6
 - Install Kibana 7
 - Configure Kibana 7
 - Install NFS Server..... 10
- Configuration..... 12
 - Configure SA Warehouse Connector..... 12
 - Add NFS mount point 12
 - Add Warehouse Connector Source and Destination Configuration 12
 - Configure Apache Flume 15
 - Custom Event Deserializer..... 16
 - Starting Flume 16
 - Configure Elasticsearch 17
 - Installing the head plugin 18
 - Flume Directory Watcher 19
- Kibana 20

Summary

Security Analytics has a lot of advantages, but it clearly lacks on Data Visualization. A customer of mine wanted to display SA Information, Packets and Logs, Information from SecOps as well as Information from other sources on SOC walls in his new CIRC.

First they had rotating SA Charts, but were not satisfied, with the information they could display and other limitations that RE charting has. Then they were using the REST API to extract data to an Archer ODA and used Archer for visualization, but still couldn't fulfill all their needs.

Then someone had the idea to use an ELK stack. ELK stand for Elasticsearch, Logstash, and Kibana. The SA Warehouse connector provides already the ability to extract Meta data into AVRO files.

There just needs to be a way to process the AVRO files and insert them into Elasticsearch. Unfortunately the Logstash Avro plugin couldn't be used, because it needs to get the schema information from a URL or a schema file, but we have the schema stored directly in the Avro file.

Apache Flume has the ability to process the Avro files in a way, which we would like to have, so instead of Logstash, we are using Flume to process AVRO files and insert them into Elasticsearch.

Installation

The solution will run on any Linux Server version. I have used CentOS to have the same distribution as we use for SA. I am using CentOS 6, but will list also the commands you would run on a CentOS 7, which introduces SystemD for managing services. It is a bit different compared to CentOS 6, but soon or later SA will move to CentOS 7, and then we need to know those things anyhow.

I used the minimal CentOS installation, so some packets might need to be installed via yum.

Install Java

Elasticsearch, Kibana and Flume are Java based, so we need to install at least Java 1.7.

```
yum install java-1.7.0-openjdk
```

Install Elasticsearch

The extracted meta data from your Log and Packet Decoder(s) ends up in an Elasticsearch cluster, which is accessed by Kibana for visualization.

More information on Elasticsearch is available here: <https://www.elastic.co/>

Elasticsearch is distributed in form of RPM packages or can be installed via their repositories. We will use the later way.

Download and install Public Signing Key:

```
rpm --import https://packages.elastic.co/GPG-KEY-elasticsearch
```

Add the following in your `/etc/yum.repos.d/` directory in a file called e.g. `elasticsearch.repo`:

```
[elasticsearch-1.7]
name=Elasticsearch repository for 1.7.x packages
baseurl=http://packages.elastic.co/elasticsearch/1.7/centos
gpgcheck=1
gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

Note: At the time of this writing version 1.7 is current. Please check here for newer versions.

<https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-repositories.html#setup-repositories>

Install it:

```
yum install elasticsearch
```

For the Elasticsearch instance to start automatically, do the following.

On CentOS 6 with SysV:

```
chkconfig --add elasticsearch
service elasticsearch start
```

On CentOS 7, which uses SystemD:

```
systemctl daemon-reload
systemctl enable elasticsearch.service
service elasticsearch start
```

Test, if Elasticsearch started correctly:

```
curl http://localhost:9200
```

should respond with:

```
{
  "status" : 200,
  "name" : "Salvo",
  "cluster_name" : "elasticsearch",
  "version" : {
    "number" : "1.7.1",
    "build_hash" : "b88f43fc40b0bcd7f173a1f9ee2e97816de80b19",
    "build_timestamp" : "2015-07-29T09:54:16Z",
    "build_snapshot" : false,
    "lucene_version" : "4.10.4"
  },
  "tagline" : "You Know, for Search"
}
```

Install Apache Flume

Apache Flume is on version 1.6.0 at the time of this writing. You will always find the latest version here: <https://flume.apache.org/download.html>

Download the binary distribution apache-flume-*-bin.tar.gz, ftp it to your home folder, extract it to /opt and set a link:

```
cd /opt
tar -xvzf ~/apache-flume-1.6.0-bin.tar.gz
ln -s apache-flume-1.6.0-bin/ flume
```

Now you can access flume in /opt/flume.

Modify Java home and Memory

Edit flume-env.sh to put your JAVA_HOME and a reasonable JAVA_OPTS (this is important, if you have a large streaming data per sec). Especially the "-Xmx" value to set the Java Heap size, needs some tuning.

```
cd /opt/flume/conf
cp flume-env.sh.template flume-env.sh
vi flume-env.sh
```

Add Elasticsearch jar to the class path

```
FLUME_CLASSPATH="/usr/share/elasticsearch/lib/elasticsearch-1.7.1.jar"
```

Note: 1.7.1 was the current version at the time of this writing. You might need to adapt the path.

This is a sample of the flume environment configuration file:

```
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# If this file is placed at FLUME_CONF_DIR/flume-env.sh, it will be sourced
# during Flume startup.

# Enviroment variables can be set here.

export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.85.x86_64
```

```
# Give Flume more memory and pre-allocate, enable remote monitoring via JMX
export JAVA_OPTS="-Xms100m -Xmx2000m -Dcom.sun.management.jmxremote"

# Note that the Flume conf directory is always included in the classpath.
FLUME_CLASSPATH="/usr/share/elasticsearch/lib/elasticsearch-1.7.1.jar"
```

Get additional Java libs

Flume needs the lucene-core libs to use the elastic search sink

```
cp /usr/share/elasticsearch/lib/lucene-core-4.10.4.jar /opt/flume/lib
```

The version of the library might be different, depending on the version of Elasticsearch installed.

Install Kibana

Either download the current Kibana version using your browser or use wget to retrieve it directly from their download site.

```
wget https://download.elastic.co/kibana/kibana/kibana-4.1.1-linux-x64.tar.gz
cd /opt
tar -xvzf ~/kibana-4.1.1-linux-x64.tar.gz
ln -s kibana-4.1.1-linux-x64/ kibana
```

Kibana is now accessible in /opt/kibana

Configure Kibana

```
vi /opt/kibana/config/kibana.yml
```

Change the "host" variable to the ip of the host you installed Kibana and change the "elasticsearch_url" to point to your Elasticsearch cluster.

NOTE: Use the real ip address. Not 127.0.0.1, even if both systems run on same host.

Add a rule to the firewall

For CentOS 6 do:

```
iptables -A INPUT -p tcp --dport 5601 -j ACCEPT
service iptables save
service iptables restart
```

On CentOS 7 do:

```
firewall-cmd --permanent --zone=public --add-port=5601/tcp
firewall-cmd --reload
```

Start Kibana (make sure Elastic search is running)

```
/opt/kibana/bin/kibana
```

Test if everything is ok:

<http://<your kibana host>:5601>

Here's an Init script, which you might use to start Kibana automatically. Copy it into /etc/init.d/kibana and make the script executable. `chmod +x kibana`

```
#!/bin/sh
#
# kibana -- startup script for kibana4
#
# chkconfig: - 85 15
# processname: kibana
# pidfile: /var/run/kibana.pid
```



```

# description: Kibana is a webui to visualize data
#
### BEGIN INIT INFO
# Provides: kibana
# Required-Start: $local_fs $remote_fs $network
# Required-Stop: $local_fs $remote_fs $network
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: start and stop kibana
### END INIT INFO
#
#
. /etc/init.d/functions

PIDFILE="/var/run/kibana.pid"
KIBANA_DIR="/opt/kibana/"

start() {
    echo -n "Starting Kibana Daemon: "
    if [ -f $PIDFILE ]; then
        PID=`cat $PIDFILE`
        echo Kibana already running: $PID
        exit 1;
    else
        cd $KIBANA_DIR
        PID=`./bin/kibana >/dev/null 2>&1 & echo $! > $PIDFILE`
        echo `cat $PIDFILE`
    fi
}

stop() {
    echo -n "Shutting down Kibana Daemon: "
    echo
    killproc kibana
    echo
    rm -f /var/lock/subsys/kibana
    return 0
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    status)
        status kibana
        ;;
    restart)
        stop

```

```
    start
    ;;
*)
    echo "Usage: {start|stop|status|restart}"
    exit 1
    ;;
esac
exit $?
```

On CentOS 6 with SysV:

```
chkconfig --add kibana
service kibana start
```

On CentOS 7, which uses SystemD:

```
systemctl daemon-reload
systemctl enable kibana.service
service kibana start
```

Install NFS Server

The Security Analytics Warehouse Connector uses a NFS mount for storing the AVRO files. We therefor need a NFS Server running on our Elasticsearch host.

```
yum install nfs-utils nfs-utils-lib
```

For CentOS 6 do:

```
chkconfig --levels 235 nfs on  
service nfs start
```

On CentOS 7 do:

```
systemctl enable rpcbind  
systemctl enable nfs-server  
systemctl enable nfs-lock  
systemctl enable nfs-idmap  
systemctl start rpcbind  
systemctl start nfs-server  
systemctl start nfs-lock  
systemctl start nfs-idmap
```

Create the directory, where the Avro files, which are shipped from the Warehouse Connector, should end up:

```
mkdir /var/export  
chmod -R 777 /var/export
```

Note: The 777 is a quick and dirty method not to end up with permission problems. It shouldn't normally be a problem and can be changed to match the security needed for nfs.

Now we share the folder via nfs by editing `/etc/exports`:

```
/var/export 192.168.1.0/255.255.255.0(rw,sync,no_subtree_check)
```

Note: The IP address and folder should match your settings.

NFS requires rpcbind, which dynamically assigns ports for RPC services and can cause problems for configuring firewall rules. To allow clients to access NFS shares behind a firewall, edit the `/etc/sysconfig/nfs` configuration file to control which ports the required RPC services run on.

In the above file uncomment the lines with `MOUNTD_PORT`, `LOCKD_TCP` and `LOCKD_UDP`. Then restart the NFS server and apply the firewall rules.

On CentOS 6 do:

```
service nfs restart  
iptables -I INPUT -m state --state NEW -p tcp -m multiport --dport  
111,892,2049,32803 -j ACCEPT  
iptables -I INPUT -m state --state NEW -p udp -m multiport --dport  
111,892,2049,32769 -j ACCEPT  
service iptables save
```

On CentOS 7 this is:

```
systemctl restart nfs-server  
firewall-cmd --permanent --zone=public --add-service=nfs  
firewall-cmd --reload
```

Configuration

Now that we have installed the various components, we need to configure SA to allow the Warehouse Connector sending data to the Elasticsearch machine and also set up Flume to read this data and insert it into the Elasticsearch instance.

Configure SA Warehouse Connector

Add NFS mount point

We want to ship the Avro files from SA to an nfs mount on the Elasticsearch machine, so we need to setup a mount point. On your Decoder define a mount point. I use “/mnt/elastic” in my sample.

```
mkdir /mnt/elastic
```

and then mount the remote NFS export on this mount point:

```
mount -t nfs -o nolock,tcp,soft,intr 192.168.1.50:/var/export /mnt/elastic
```

to auto mount the nfs volume on decoder restart, you shall add it to /etc/fstab:

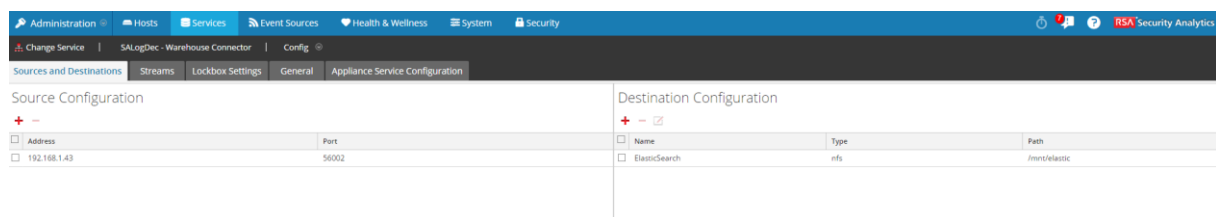
```
tail -n 1 /etc/mtab >> /etc/fstab
```

edit /etc/fstab and add the option “auto” like in this example:

```
192.168.1.50:/var/export /mnt/elastic nfs  
rw,nolock,tcp,soft,intr,vers=4,auto,addr=192.168.1.50,clientaddr=192.168.1.43 0 0
```

Add Warehouse Connector Source and Destination Configuration

In the Config Section of the Warehouse Connector Service, add your Decoder(s) to the Source Configuration and add the NFS Mount, which you just configured above to the Destination Configuration:

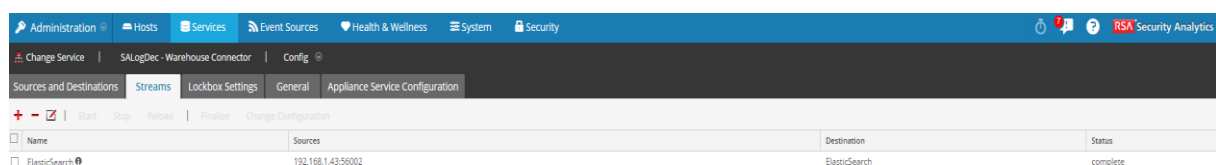


The screenshot shows the configuration page for the Warehouse Connector service. It is divided into two main sections: Source Configuration and Destination Configuration.

Source Configuration	
<input type="checkbox"/>	192.168.1.43

Destination Configuration			
Name	Type	Path	
<input type="checkbox"/>	ElasticSearch	nfs	/mnt/elastic

Now we add a stream, using our Source and destination:

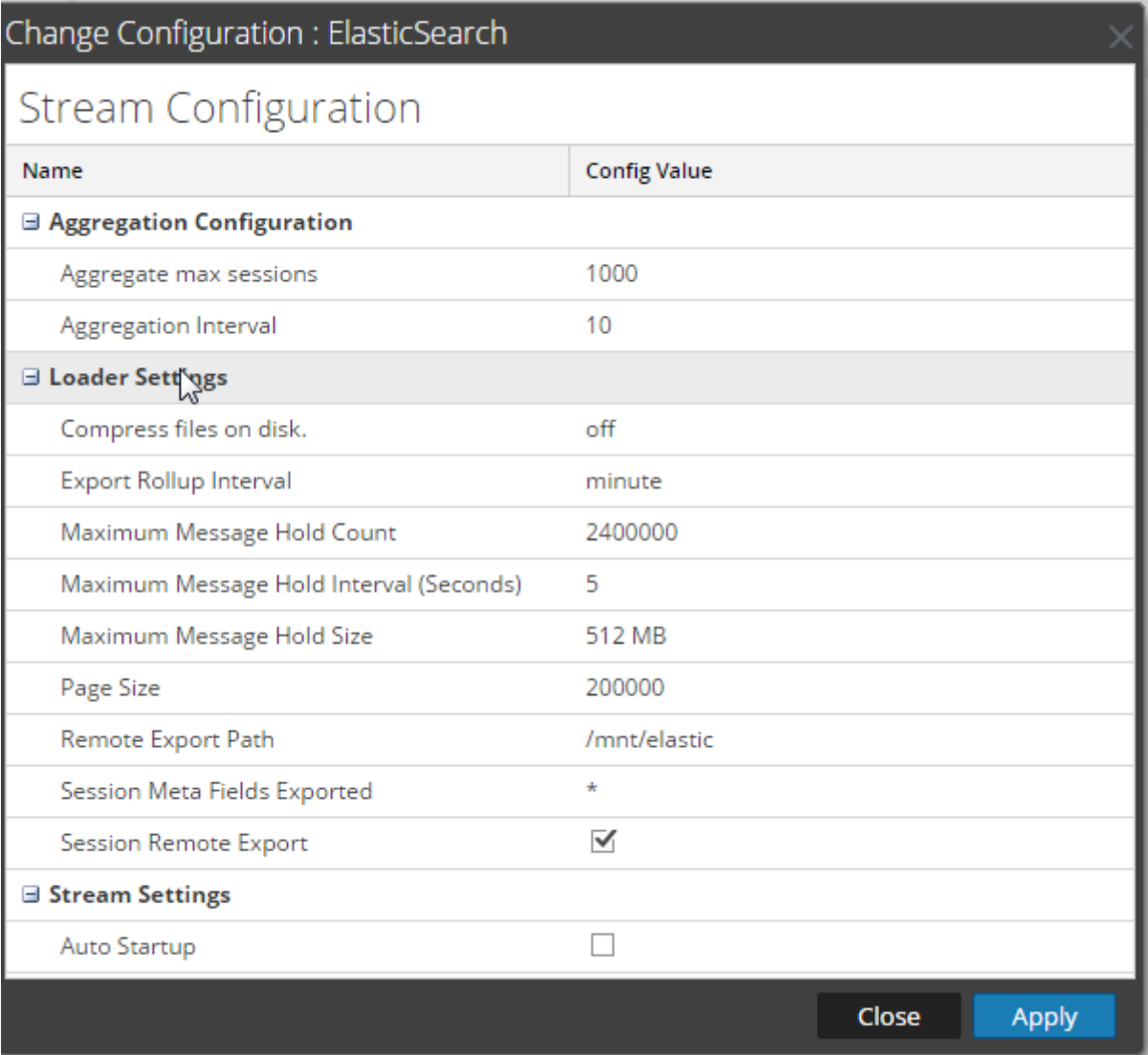


The screenshot shows the configuration page for the Warehouse Connector service, specifically the Streams section. A stream named 'ElasticSearch' has been configured.

Name	Sources	Destination	Status	
<input type="checkbox"/>	ElasticSearch	192.168.1.43:56002	ElasticSearch	complete

As we want to have “Near Real-time data”, the Export Rollup interval should be set to “minute” and I have set the Maximum message hold size to 512 MB.

You might experiment with your own values here and see what best fits your environment.



The screenshot shows a configuration window titled "Change Configuration : ElasticSearch". Inside, there is a section for "Stream Configuration" which contains a table of settings. The table has two columns: "Name" and "Config Value". The settings are grouped into three sections: "Aggregation Configuration", "Loader Settings", and "Stream Settings".

Name	Config Value
Aggregation Configuration	
Aggregate max sessions	1000
Aggregation Interval	10
Loader Settings	
Compress files on disk.	off
Export Rollup Interval	minute
Maximum Message Hold Count	2400000
Maximum Message Hold Interval (Seconds)	5
Maximum Message Hold Size	512 MB
Page Size	200000
Remote Export Path	/mnt/elastic
Session Meta Fields Exported	*
Session Remote Export	<input checked="" type="checkbox"/>
Stream Settings	
Auto Startup	<input type="checkbox"/>

At the bottom right of the dialog, there are two buttons: "Close" and "Apply".

After you saved the above settings and finalized the stream, you should tell the stream not to be “nice”. Otherwise you will notice that you have gaps delivering the data. We saw several times a day periods of 15 minutes, where no data was sent. Setting the value of `aggregate.nice` to `false` will solve that.

The screenshot shows the Administration console for the 'SALogDec - Warehouse Connector'. The left sidebar shows a tree view with 'warehouseconnector' expanded to 'config'. The main panel displays the configuration for the stream at the path '/warehouseconnector/streams/ElasticSearch/config'. The configuration table is as follows:

Property	Value
aggregate.nice	false
aggregate.sessions	1000
aggregation.interval	10
auto.startup	off
destination	/warehouseconnector/destinations/ElasticSearch
free.space	1024 MB
handle.multivalue	false
heartbeat.error	120
heartbeat.interval	10
max.pending	80
rejected.maxusage	30
sources	/warehouseconnector/sources/192.168.1.43:56002
state	stopped
stream.id	1442319994124
stream.startpercent	50
stream.stopdiskusage	0
transform	on

After that you are ready to start the Stream.

Configure Apache Flume

Now that we receive the Avro files on your NFS mount, we need to configure Apache Flume to pick up the data from the Avro files and insert it into Elasticsearch.

For that purpose, we create a configuration file in the `/opt/flume/conf` folder. In our example it is called “elastic.conf”.

```
1 # elastic.conf: A single-node Flume configuration
2 # Reading avro files via spool dir and use the elastic sink
3
4 # Name the components on this agent
5 a1.sources = avrofld
6 a1.sinks = k1
7 a1.channels = c1
8
9 # Describe/configure the folder with the avro files from Warehouse Connector
10 a1.sources.avrofld.type = spooldir
11 a1.sources.avrofld.channels = c1
12 a1.sources.avrofld.spoolDir = /var/export
13 a1.sources.avrofld.fileHeader = true
14 a1.sources.avrofld.deserializer = avro
15
16 # Describe the elasticsearch sink
17 a1.sinks.k1.type = org.apache.flume.sink.elasticsearch.ElasticSearchSink
18 a1.sinks.k1.hostNames = 127.0.0.1:9300
19 a1.sinks.k1.indexName = sa
20 a1.sinks.k1.indexType = data
21 a1.sinks.k1.clusterName = elasticsearch
22 a1.sinks.k1.batchSize = 1000
23 a1.sinks.k1.ttl = 2d
24 a1.sinks.k1.serializer = com.rsa.flume.serialization.FlumeAvroEventDeserializer
25
26 # Use a channel which buffers events in memory
27 a1.channels.c1.type = memory
28 a1.channels.c1.capacity = 1000000
29 a1.channels.c1.transactionCapacity = 100000
30
31 # Bind the source and sink to the channel
32 a1.sources.r1.channels = c1
33 a1.sinks.k1.channel = c1
```

Lines 5 – 7 name the Sources, Sinks and Channels used in this configuration.

Line 10 activates the SpoolDir Source, which monitors a folder (line 12) for new files to appear. It deserializes them using the Avro deserializer (line 14) and sends the so called Events via Channel name c1 (line 11) to the sink.

From Line 16 onwards we define the Elasticsearch sink, which instructs Flume to create an index and insert the events into the index.

Line 18 defines the hostname and port of Elasticsearch. Because Flume and Elasticsearch run on the same machine, we use localhost here.

Line 19 and line 20 define the Index Name and Index Type created on Elasticsearch. We will use that with Kibana. In our example, with Index Name of “sa” we will get daily indices named “sa-YYYY-MM-DD” and the index type used is “data”.

Line 24 uses a special deserializer, written by me, which converts the Flume Events into a format that Logstash would have created. This is file “flume-rsa-1.0.jar”, which needs to reside in /opt/flume/lib. More on the custom deserializer can be found later.

Line 26 – 29 define the channel to be used. We want to have all Events added to Memory and the events should be inserted into Elasticsearch from there. In environments with high transactions the capacity parameter in line 28 might need adjustments.

And finally in lines 32 and 33 we bind the source and sink to the channel

More Information on Sources, Sinks and Agents is available in the Flume User Guide:
<http://flume.apache.org/FlumeUserGuide.html>

Custom Event Deserializer

The custom event deserializer takes all Meta fields, which have been read from the Avro file and puts them to the channel, so that it can be inserted into elastic search.

Furthermore it extracts the Decoder name from the filename and sets it to the source field. Geo IP information (latdec_src, latdec_dst, longdec_src and longdec_dst) are converted to Geo IP information suitable for Kibana. They are stored as fields location_src and location_dst.

A configuration file named “FlumeAvroEventDeserializer.xml” in /opt/flume/conf allows specifying a Time correction to be applied to specific device types and can also specify Meta fields to be excluded. Editing a XML file is easier than having a comma separated values in an ugly Text field in the Warehouse Configurator.

Starting Flume

Having the configuration in place, we can start Flume. It is best to run it in a foreground window first, so that you can see if there are any problems:

```
bin/flume-ng agent --conf conf --conf-file conf/elastic.conf --name a1 -  
Dflume.root.logger=INFO,console
```

The above starts a Flume Agent using our elastic.conf file and using the agent named “a1”.

If you remove the “-Dflume.root.logger=INFO,console” from the command line, Flume will run silent and write into the logs folder as configured in conf/log4j.properties.

Configure Elasticsearch

Elasticsearch uses the Lucene search engine for indexing data. With the default settings it would analyze strings and separate them. This results in country “United States” to appear twice in Kibana as “United” and “States”.

Fortunately we can have Index templates in elastic search which prevents this to happen. We will do a similar thing to numeric fields, so that they are treated as numbers and not as strings.

Templates are stored in `/etc/elasticsearch/templates` and be named after the index they should be applied on. So the template for our index “sa”, which we specified in our flume config file should be called “sa_template.json”.

A sample is provided with this distribution.

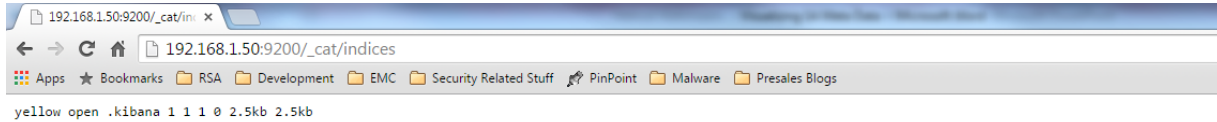
```
1 {
2   "sa_template" : {
3     "template" : "sa-*",
4     "mappings" : {
5       "data" : {
6         "properties" : {
7           "@source" : { "type": "string", "index": "not_analyzed" },
8           "@fields" : {
9             "properties" : {
10              "action" : { "type": "string", "index": "not_analyzed" },
11              "alias_host" : { "type": "string", "index": "not_analyzed" },
12              "bytes" : {"type": "long"},
```

Line 2 should specify the template name. Line 3 refers to the index name in elastic.conf and line 5 refers to the index type in elastic.conf.

Installing the head plugin

You can check the status of Elasticsearch using specific urls, like:

```
http://192.168.1.50:9200/_cat/indices
```



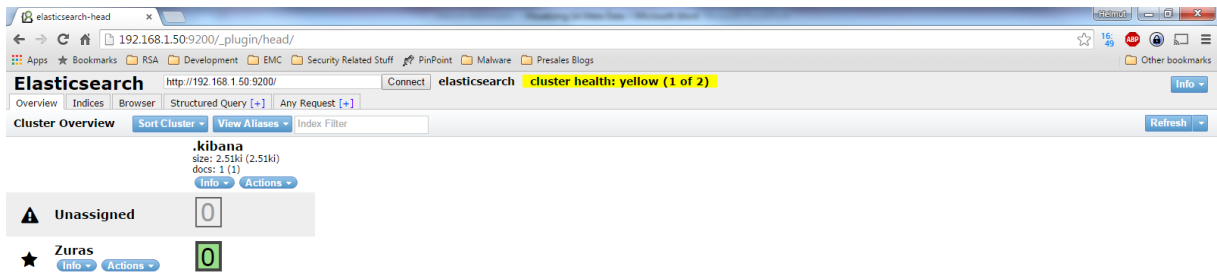
But it is easier to use the head plugin for that purpose. Simply install it with:

```
/usr/share/elasticsearch/bin/plugin -install mobz/elasticsearch-head
```

And then use your browser:

http://192.168.1.50:9200/_plugin/head/

Which gives you a way to manage the indices, issue queries, etc.



Flume Directory Watcher

With our configurations we will have now in /var/export a structure like we have on SAW. But we just want to have the Avro files with Meta data from Log Decoders and Packet Decoders, which are stored in the “sessions” folder. We don’t need the raw logs from the logs folder.

And once an Avro file has been processed it can be removed.

For that purpose I have created a python script, named flumedirwatcher.py, which should be added to cron to run every minute.

It traverses the “/var/export/rsasoc/v1” folder and looks for files starting with “session”. Those are the files containing Meta data. If such a file is found, I am adding the time stamp of the copy to the filename, for trouble shooting and to prevent duplicate filenames, and move the file to the folder which is watched by flume. It is “/var/export”.

Note: The SpoolDir Source is configured not to process subfolders.

If files starting with “logs” are found, we are deleting them, because we don’t need raw logs.

At the same time empty folders are removed, so that they folders, where the session files have been moved from don’t waste disk space.

And finally we are removing the files, which were processed already by flume. Those files are suffixed with “.COMPLETED” in the “/var/export” folder.

Note: We keep them for 1 hour, since Flume might have a delay in processing all the Events in memory and this could cause then problems with the custom Deserializer

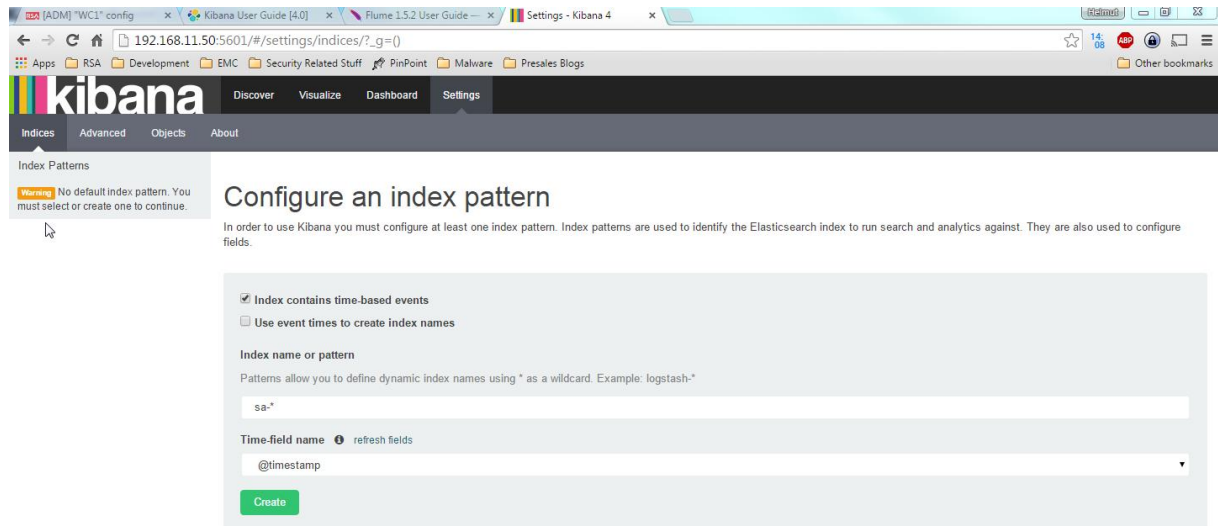
Kibana

Now that we have some data in Kibana, we would like to visualize it.

We navigate to the Kibana host on port 5601.

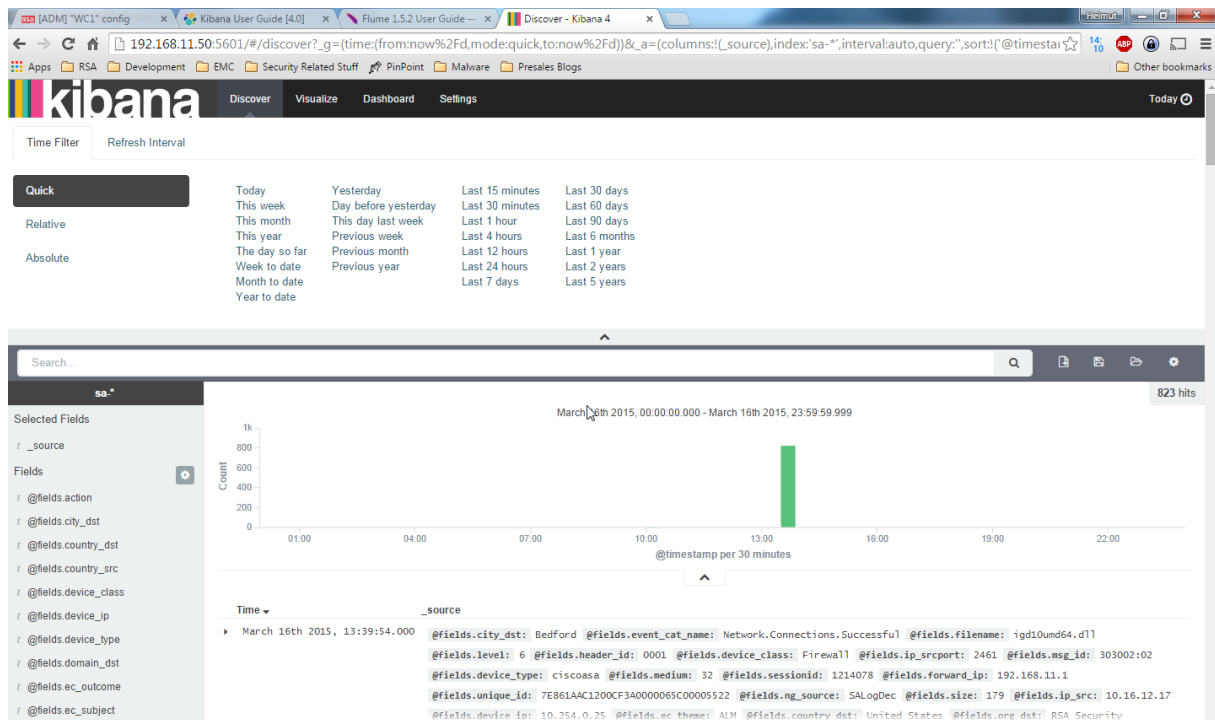
First we need to configure an Index Pattern. Remember our index is called “sa-” and not “logstash-” as shown in the defaults.

And we use the “@timestamp” field.



Set it as default index.

Now you can go to “Discover” and look at your data. Be careful that the default time filter only looks at the last 15 minutes, so you might select last hour or today.



And now you can start creating visualizations and putting them on dashboards.

Have a look at the Kibana User guide: <http://www.elastic.co/guide/en/kibana/current/index.html>