Monitoring Alfresco

With ElasticSearch

Contents

[Introduction 3](#_Toc445726456)

[OpenSource Tools 3](#_Toc445726457)

[Assumptions 3](#_Toc445726458)

[Prereqs 4](#_Toc445726459)

[Installing and Configuring Monitoring VM 4](#_Toc445726460)

[Redis 4](#_Toc445726461)

[Kibana 5](#_Toc445726462)

[Logstash 8](#_Toc445726463)

[ElasticSearch 9](#_Toc445726464)

[Installing and Configuring Logstash agent in Alfresco Servers 9](#_Toc445726465)

[Changes to Alfresco servers 10](#_Toc445726466)

[Changes to Alfresco application 10](#_Toc445726467)

[Changes to logstash agent 10](#_Toc445726468)

[Starting/Stopping processes in Monitoring VM 12](#_Toc445726469)

[Starting/Stopping logstash agent 12](#_Toc445726470)

[Accessing the monitoring Web interface 13](#_Toc445726471)

[Index Maintenance 14](#_Toc445726472)

[Setting Kibana to Read-Only mode 15](#_Toc445726473)

[Appendix A: logstash.conf for VM 16](#_Toc445726474)

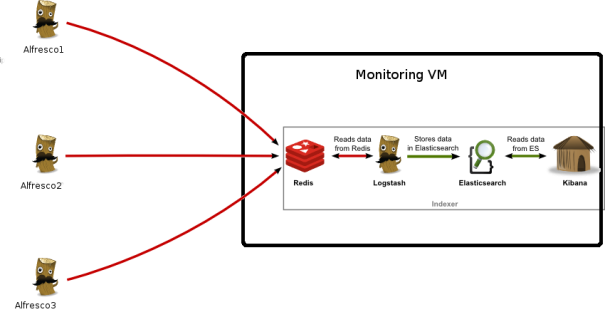
[Appendix B: elasticsearch-remove-old-indices.sh 24](#_Toc445726475)

# Introduction

This document describes how to monitor Alfresco software with OpenSource software such as Logstash, ElasticSearch and Kibana. For further information about these components please refer to: <https://www.elastic.co>

# OpenSource Tools

There are two major components used for monitoring Alfresco, one is the logstash agent that runs on each Alfresco node to collect logs and other additional data and the other one is the Virtual Machine running other components such as ElasticSearch and Kibana to store and display the data.



In the picture above we can see the relation between the components. On the left side we have the logstash “Shippers”, these are the logstash java processes running on each of the Alfresco and Solr nodes collecting the necessary monitoring data. On the right side we have the Monitoring VM with “Redis” receiving the traffic from the “Shippers”, “Logstash” to parse/modifiy the incoming messages, “ElasticSearch” to index and store the messages and “Kibana” to display the content via Web interface.

# Assumptions

The following is assumed:

* Alfresco is running on a Linux server i.e. RHEL
* Alfresco application is running under alfresco user
* logstash agent is installed in /home/alfresco/logstash-elasticsearch
* Alfresco is running under Tomcat web server
* Redis port 6379 needs to be reachable from Alfresco nodes, make sure firewall is open

# Prereqs

* Java JDK needs to be installed on each Alfresco node

**NOTE:**

Any deviations from the assumptions may cause the solution not to work as expected as additional configuration changes may be required.

Elastic software changes versions quite frequently so adjustments may be required for different versions of the software.

At the time of writing this document the solution was tested with the following software versions

* Redis 3.0.0
* ElasticSearch 2.2.0
* Logstash 2.2.2
* Kibana 4.4.1

# Installing and Configuring Monitoring VM

The Monitoring server can be a physical or virtual server. There are 4 software components that need to be installed in the Monitoring VM, these are:

1. Redis
2. Logstash
3. ElasticSearch
4. Kibana

## Redis

Redis software can be obtained from here:

<http://redis.io/download>

Installation instructions are also available on the same page:

<http://redis.io/download#installation>

An easier way may be installing the EPEL repo first as this may not require software compilation. Instructions available here:

<http://www.tecmint.com/how-to-enable-epel-repository-for-rhel-centos-6-5/>

By default redis listens for connections from all the network interfaces available on the server. If this is not the case in your installation then change redis configuration file /etc/redis/6379.conf or /etc/redis/redis.conf and set the bind parameter:

bind 0.0.0.0

Restart redis after the change

# service redis\_6379 restart

or

# service redis-server restart

depending on your redis installation.

## ElasticSearch

ElasticSearch software and installation steps can be found here:

https://www.elastic.co/downloads/elasticsearch

## Logstash

Logstash software and installation steps can be obtained here:

https://www.elastic.co/downloads/logstash

Logstash configuration file is /etc/logstash/conf/logstash.conf. Copy the provided logstash configuration file (Appendix A: logstash.conf for VM) to this location.

**Note:** If monitoring more than one environment i.e. UAT, PROD, etc then it is advisable to create new indexes for each environment. This way it is easier to maintain/keep index data for your environments independently of each other, for example you can keep 30 day of UAT data and 180 days of PROD data. This can be easily configured by using the hostname or any other unique field to determine which index to be used. The following example shows how to achieve this in /etc/logstash/conf.d/logstash.conf file as shown below.

output {

if [host] =~ /alfresco1\_uat.+/ {

if [type] in ["jmx", "AlfrescoCPU", "ServerCPU"] {

#Uncomment for debugging purposes

#stdout { codec => rubydebug }

elasticsearch {

index => "uat\_jmx-%{+YYYY.MM.dd}"

hosts => "localhost:9200"

}

} else if [type] == "TomcatAccessLog" {

#Uncomment for debugging purposes

#stdout { codec => rubydebug }

elasticsearch {

index => "uat\_tomcat-%{+YYYY.MM.dd}"

hosts => "localhost:9200"

}

} else {

#Uncomment for debugging purposes

#stdout { codec => rubydebug }

elasticsearch {

index => "uat\_logstash-%{+YYYY.MM.dd}"

hosts => "localhost:9200"

}

}

}

…

**Note for ubuntu 14.04:**

If using Ubuntu you may have to create a solftlink for the libcrpt.so library as mentioned in this link <https://github.com/elastic/logstash/issues/3127#issuecomment-101068714>

ln -s /lib/x86\_64-linux-gnu/libcrypt.so.1 /usr/lib/x86\_64-linux-gnu/libcrypt.so

## Kibana

Kibana software and installation step can be obtained here:

https://www.elastic.co/downloads/kibana

Installing Kibana as a service. As root user create a service file for Kibana called /etc/init.d/kibana with the contents shown below.

#!/bin/bash

### BEGIN INIT INFO

# Provides: kibana

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Short-Description: Runs kibana daemon

# Description: Runs the kibana daemon as a non-root user

### END INIT INFO

# Process name

NAME=kibana

DESC="Kibana4"

PROG="/etc/init.d/kibana"

# Configure location of Kibana bin

KIBANA\_BIN=/home/kibana\_user/kibana-current/bin

# PID Info

PID\_FOLDER=/var/run/kibana/

PID\_FILE=/var/run/kibana/$NAME.pid

LOCK\_FILE=/var/lock/subsys/$NAME

PATH=/bin:/usr/bin:/sbin:/usr/sbin:$KIBANA\_BIN

DAEMON=$KIBANA\_BIN/$NAME

# Configure User to run daemon process

DAEMON\_USER=kibana\_user

# Configure logging location

KIBANA\_LOG=/var/log/kibana.log

# Begin Script

RETVAL=0

if [ `id -u` -ne 0 ]; then

echo "You need root privileges to run this script"

exit 1

fi

# Function library

. /etc/init.d/functions

start() {

echo -n "Starting $DESC : "

pid=`pidofproc -p $PID\_FILE kibana`

if [ -n "$pid" ] ; then

echo "Already running."

exit 0

else

# Start Daemon

if [ ! -d "$PID\_FOLDER" ] ; then

mkdir $PID\_FOLDER

fi

daemon --user=$DAEMON\_USER --pidfile=$PID\_FILE $DAEMON 1>"$KIBANA\_LOG" 2>&1 &

sleep 2

pidofproc node > $PID\_FILE

RETVAL=$?

[[ $? -eq 0 ]] && success || failure

echo

[ $RETVAL = 0 ] && touch $LOCK\_FILE

return $RETVAL

fi

}

reload()

{

echo "Reload command is not implemented for this service."

return $RETVAL

}

stop() {

echo -n "Stopping $DESC : "

killproc -p $PID\_FILE $DAEMON

RETVAL=$?

echo

[ $RETVAL = 0 ] && rm -f $PID\_FILE $LOCK\_FILE

}

case "$1" in

start)

start

;;

stop)

stop

;;

status)

status -p $PID\_FILE $DAEMON

RETVAL=$?

;;

restart)

stop

start

;;

reload)

reload

;;

\*)

# Invalid Arguments, print the following message.

echo "Usage: $0 {start|stop|status|restart}" >&2

exit 2

;;

esac

To make this script work, you'll need to change the following;

Location of Kibana bin file:

KIBANA\_BIN=/home/kibana\_user/kibana.xxxx

DAEMON\_USER=kibana\_user

Before you can display any data in Kibana we need to get some data into ElasticSearch so Kibana knows about the indexes and fields to be used. So first make sure Alfresco and Logstash are started in the Alfresco nodes and Redis, Logstash and ElasticSearch are running in the Monitoring VM. To verify if the indexes have been created you can run this command from the Monitoring VM command line

[alfresco@alfrescoMonitor ~]$ curl 'localhost:9200/\_cat/indices?v'

health status index pri rep docs.count docs.deleted store.size pri.store.size

yellow open logstash-2016.02.17 5 1 10 0 83.3kb 83.3kb

yellow open .kibana 1 1 239 5 156.5kb 156.5kb

yellow open jmx-2016.02.17 5 1 72 0 214.2kb 214.2kb

yellow open tomcat-2016.02.17 5 1 67066 0 19mb 19mb

As you can see we now have some indexes created (logstash, jmx and tomcat) so we can now upload the rest of the configuration for Kibana.

You can now add the service, enable it and start Kibana with the following commands.

# chkconfig --add kibana

# chkconfig kibana on

# chmod +x /etc/init.d/kibana

# mkdir /var/run/kibana

# service kibana start

You should now be able to reach kibana on the following url: http://<alfrescoMonitor IP>:5601

# Installing and Configuring Logstash agent in Alfresco Servers

In this section we are going to configure the logstash agent running on Alfresco and Solr nodes. Changes are required to the server, Alfresco application and the logstash agent itself.

First copy the zip file “logstash-elasticsearch.zip” to each of the Alfresco nodes and unzip it under /home/alfresco.

## Changes to Alfresco servers

* As root user edit /etc/hosts and add an entry for the monitoring server IP address i.e.

xxx.xxx.xxx.xxx alfrescoMonitor

* As alfresco user edit /home/alfresco/.bashrc file and add the following entries

export RUN\_ELASTICSEARCH=/home/alfresco/logstash-elasticsearch

export JAVA\_HOME=/home/alfresco/jdk1.8.xx

export PATH=$JAVA\_HOME/bin:$PATH

* Execute the .bashrc file to apply the changes

. /home/alfresco/.bashrc

* It is advisable to keep the host short name first in /etc/hosts as this hostname will be used in Kibana dashboards. For example if our Alfresco host is called alfresco1 the shortname should appear first as shown here:

xxx.xxx.xxx.xxx alfresco1 alfresco1.alfresco.com

## Changes to Alfresco application

Edit tomcat/conf/server.xml file and replace the access log valve as shown below

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

prefix="access-" suffix=".log"

pattern='%a %l %u %t "%r" %s %b "%{Referer}i" "%{User-agent}i" %D "%I"'

resolveHosts="false"/>

Add the following entry to JAVA\_OPTS. In Alfresco 5.x that's configured in tomcat/bin/setenv.sh file

-Djava.rmi.server.hostname=<your host name>

In Alfresco 5.1 make sure jmx connector (RMI) is enabled in alfresco-global.properties

### Remote JMX (Default: disabled) ###alfresco.jmx.connector.enabled=true

## Changes to logstash agent

Copy logstash-elasticsearch folder to /home/alfresco on each of the Alfresco nodes. Logstash-elastic search folder can be downloaded from GitHub <https://github.com/miguel-rodriguez/alfresco-monitoring> under Alfresco Nodes/Logstash link.

Edit file /home/alfresco/logstash-elasticsearch/jmx/jmx.conf and adjust the rmi settings.

"host" : "<your host name>",

"port" : 50500,

"username" : "controlRole",

"password": "change\_asap",

The following files need to have executable permission

chmod +x /home/alfresco/logstash-elasticsearch/run\_logstash.sh

chmod +x /home/alfresco/logstash-elasticsearch/bin/logstash

chmod +x /home/alfresco/logstash-elasticsearch/vendor/jruby/bin/jruby

**Note for ubuntu 14.04:**

If using Ubuntu you may have to create a solftlink for the libcrpt.so library as mentioned in this link <https://github.com/elastic/logstash/issues/3127#issuecomment-101068714>

# Starting/Stopping processes in Monitoring VM

There are several processes running in the Monitoring VM:

1. Redis
2. Logstash
3. ElasticSearch
4. Kibana

Redis, Logstash, ElasticSearch and Kibana are running as services and therefore they should be started automatically when the server starts up. These services can also be started/stopped manually as root user:

# service redis stop

# service redis start

# service logstash stop

# service logstash start

# service elasticsearch stop

# service elasticsearch start

# service kibana stop

# service kibana start

# Starting/Stopping logstash agent

The logstash agent is installed on all Alfresco and Solr nodes under *“/home/alfresco/logstash-elasticsearch/”*

The agent is started/stopped as *“alfresco”* user as shown below

# To start the monitoring process

cd /home/alfresco/*logstash-elasticsearch*

./run\_logstash.sh start

# To stop the monitoring process

cd /home/alfresco/*logstash-elasticsearch*

./run\_logstash.sh stop

To verify if the process is running type:

ps -ef | grep logstash

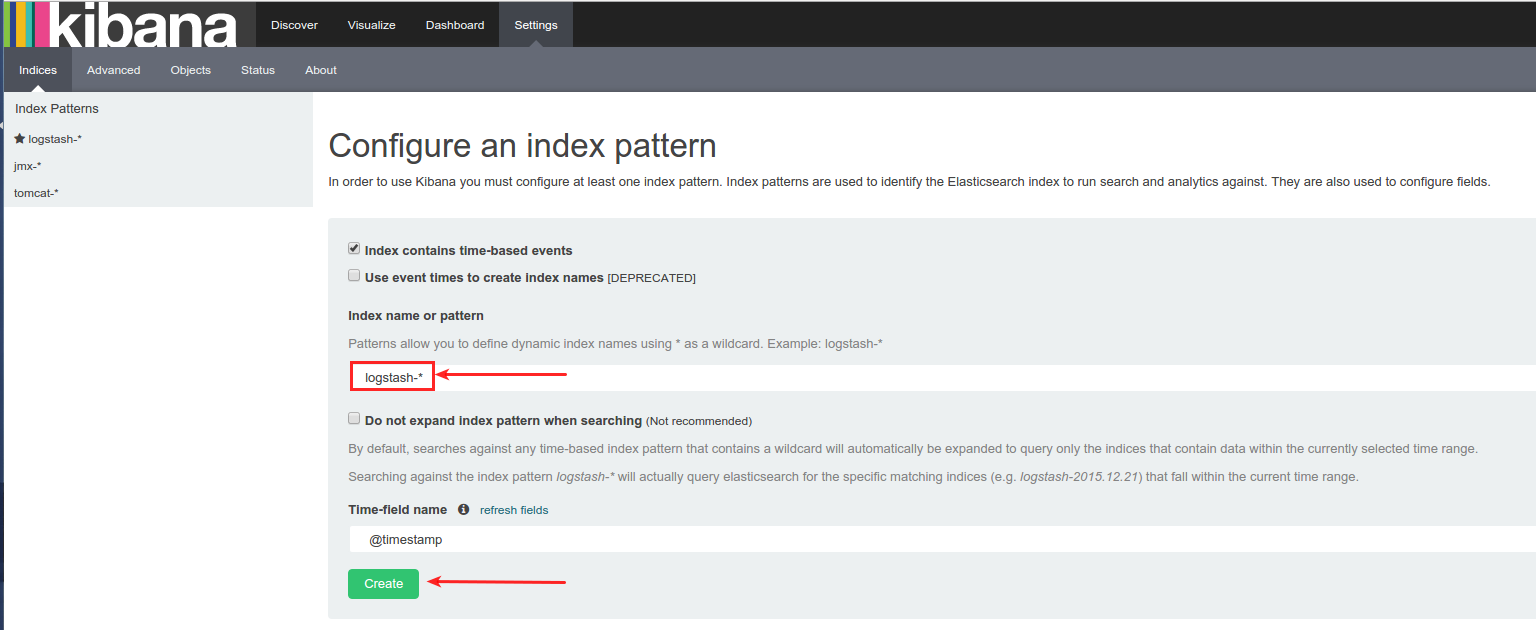
alfresco 2432 1 99 09:46 pts/0 00:00:03 /home/alfresco/jdk1.8.0\_65/bin/java -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -Djava.awt.headless=true -XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyOnly -Xmx500m -Xss2048k -Djffi.boot.library.path=/home/alfresco/logstash-elasticsearch/vendor/jruby/lib/jni -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -Djava.awt.headless=true -XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyOnly -Xbootclasspath/a:/home/alfresco/logstash-elasticsearch/vendor/jruby/lib/jruby.jar -classpath : -Djruby.home=/home/alfresco/logstash-elasticsearch/vendor/jruby -Djruby.lib=/home/alfresco/logstash-elasticsearch/vendor/jruby/lib -Djruby.script=jruby -Djruby.shell=/bin/sh org.jruby.Main --1.9 /home/alfresco/logstash-elasticsearch/lib/bootstrap/environment.rb logstash/runner.rb agent -f /home/alfresco/logstash-elasticsearch/logstash.conf

# Accessing the monitoring Web interface

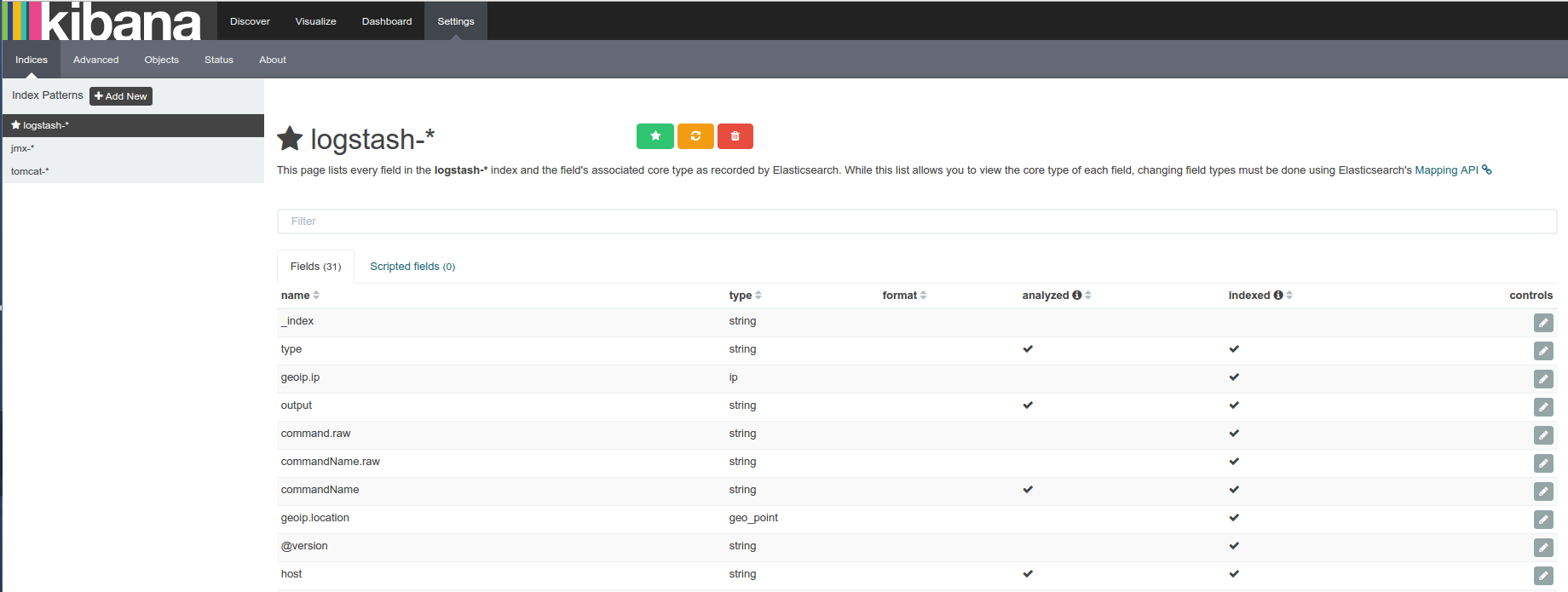
To access Alfresco monitoring web interface open a browser and enter the following url:

<http://alfrescoMonitor:5601/>

We now need to tell Kibana about the indexes to be used. Go to “Settings” and then “Indices”. Under “Index name or parttern” enter “logstash-\*” and click on create. Perform the same operation for indices “jmx-\*” and “logstash-\*”.



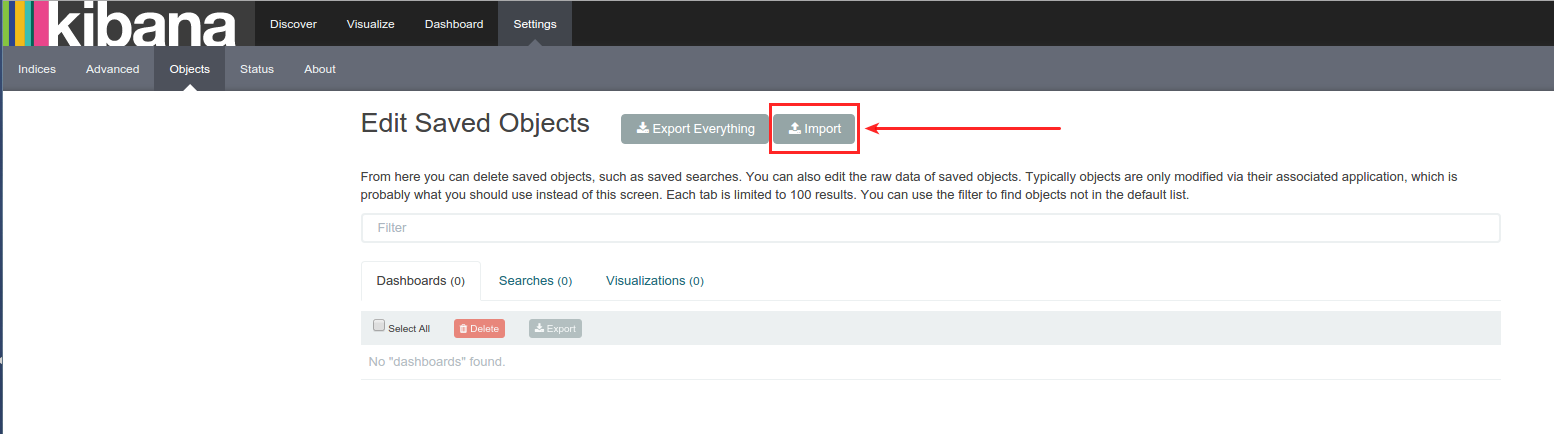
The index information will be generated as shown below



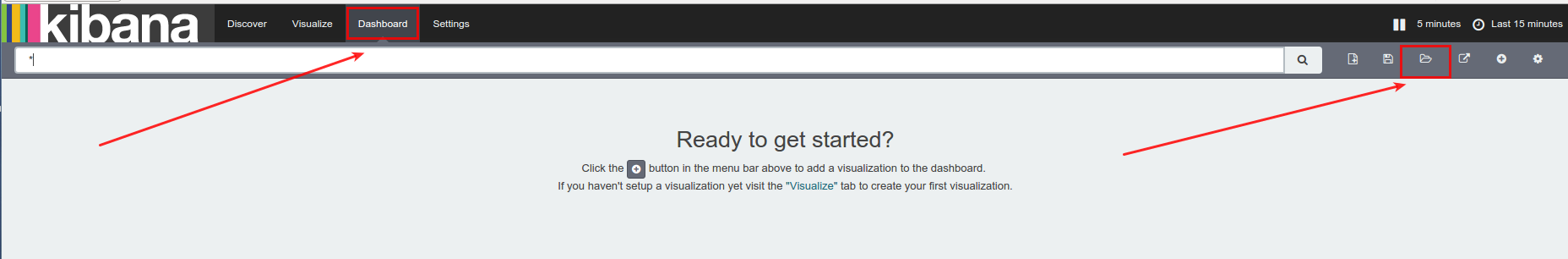
Now Kibana is up and running and the indexes are ready we need to load the sample searches, visualizations and dashboards used to display Alfresco logs and statistical data. These can be found here: <https://github.com/miguel-rodriguez/alfresco-monitoring/tree/master/Kibana>

Download these files and do a search on the files for “NodeX” and replace it with the name of your Alfresco host i.e. “alfresco1”.

Go to “Settings” and then “Objects” and upload the configuration files to Kibana in the following order, first searches.json, then visualizations.json and finally dashboards.json

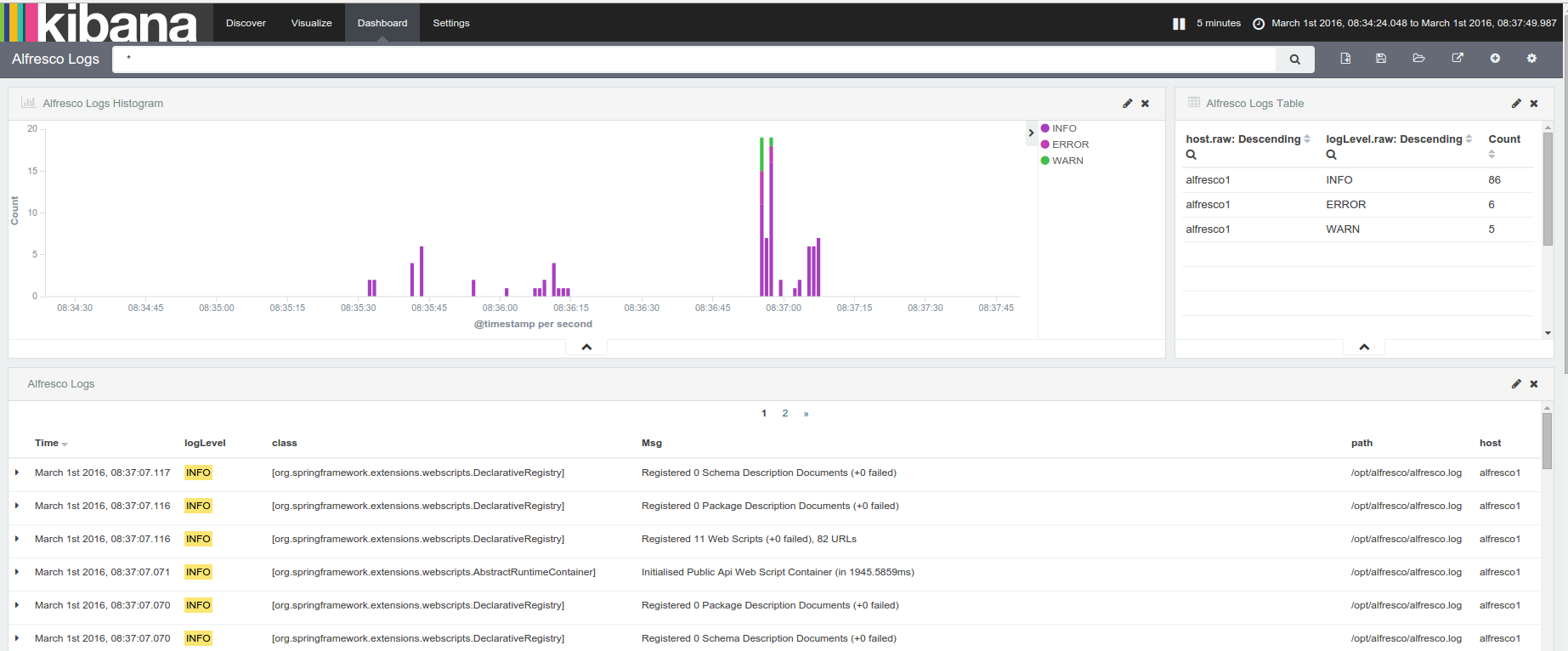


Once the configuration files have been loaded click on “*Dashboard*” menu link. On the next page click on the “Open” dashboard link to see the available dashboards.

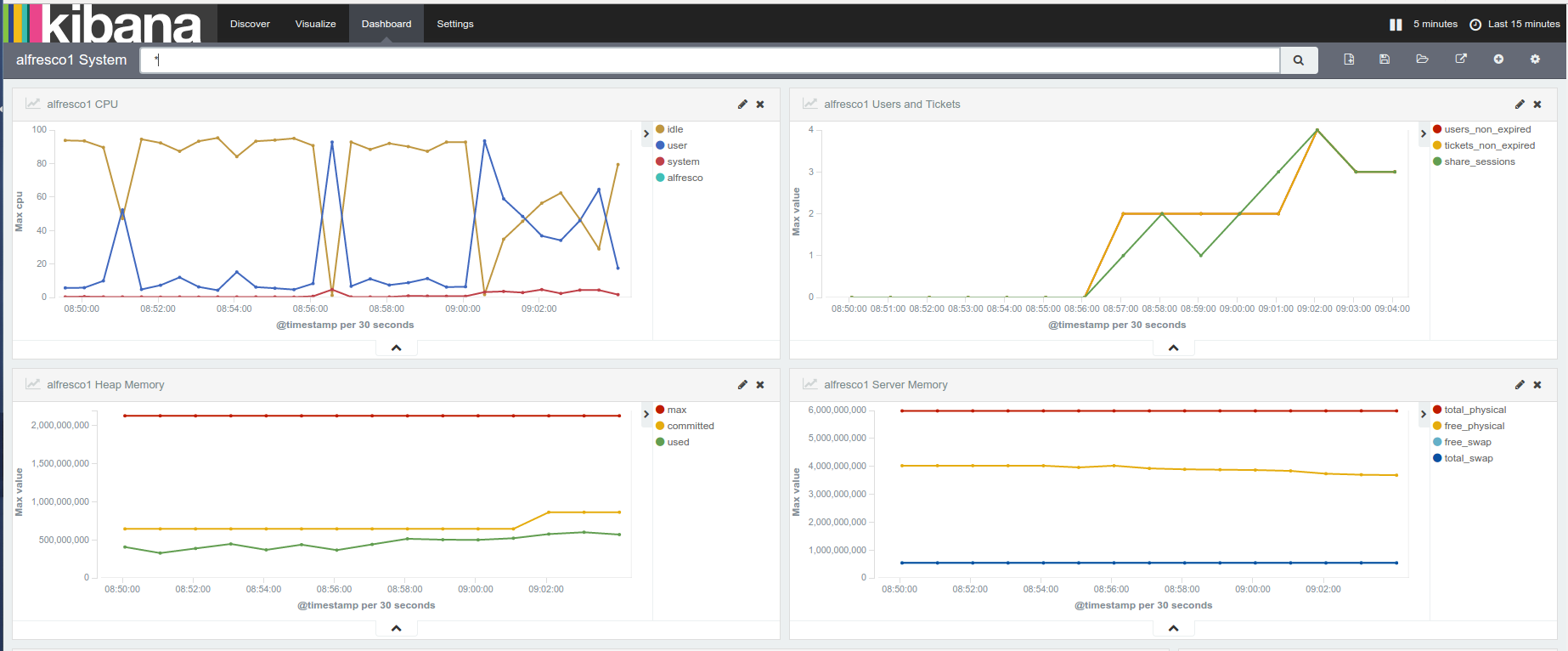


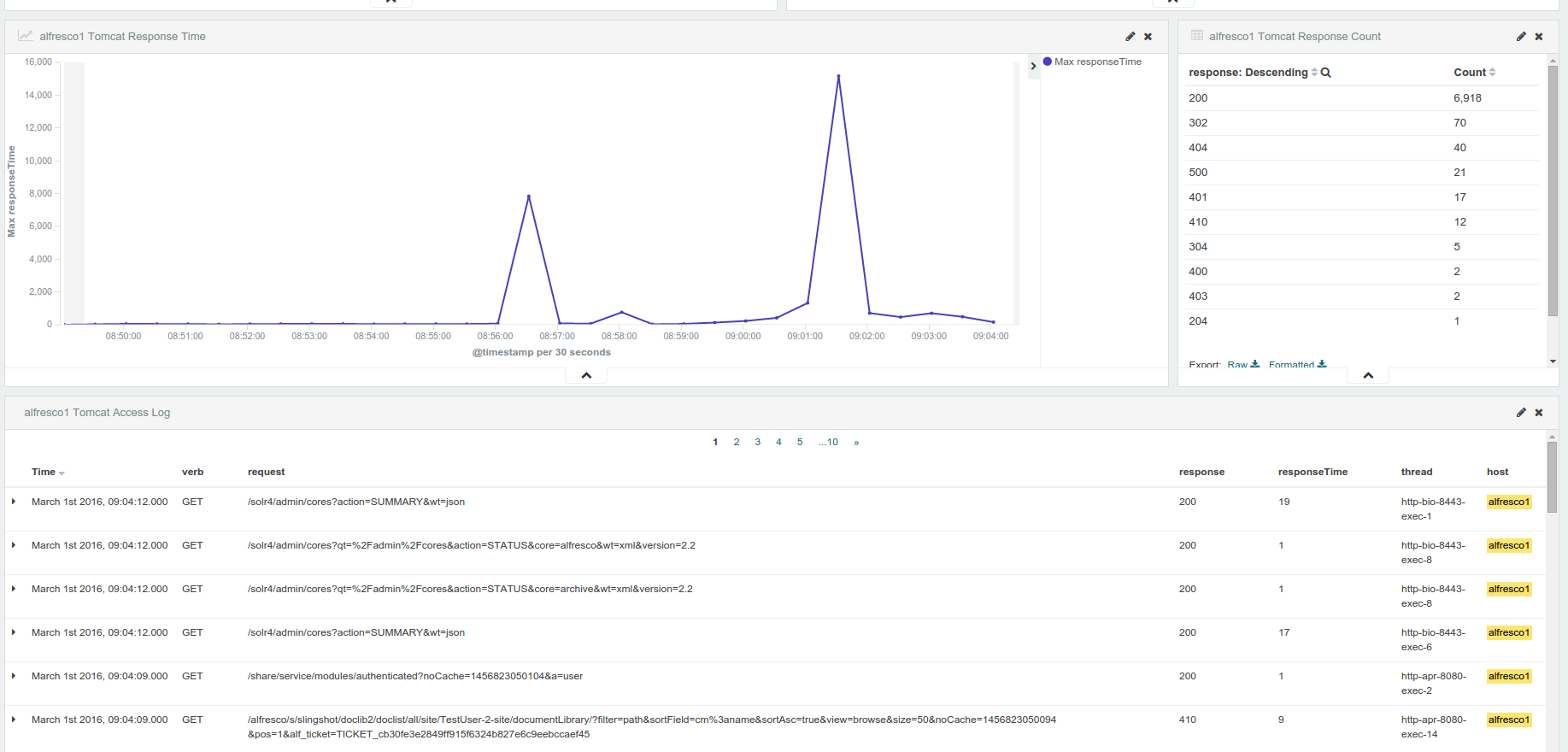
Select the desired dashboard to visualize the data.

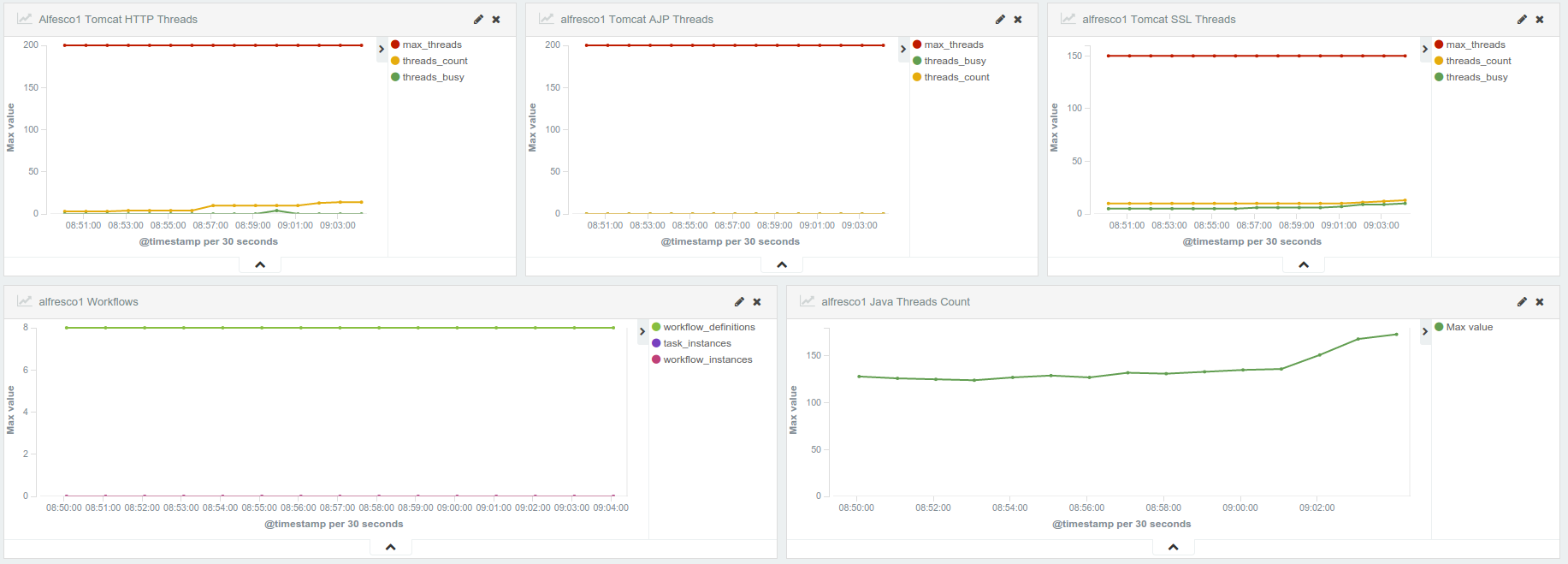
Alfresco Logs Dashboard sample



Alfresco System Dashboard sample







# Index Maintenance

In ElasticSearch a new index is created every day to store the content. To see the existing indexes execute the following command from the monitoring VM:

[alfresco@alfrescoMonitor ~]$ curl 'localhost:9200/\_cat/indices?v'

health status index pri rep docs.count docs.deleted store.size pri.store.size

yellow open logstash-2016.02.17 5 1 10 0 83.3kb 83.3kb

yellow open .kibana 1 1 239 5 156.5kb 156.5kb

yellow open jmx-2016.02.17 5 1 72 0 214.2kb 214.2kb

yellow open tomcat-2016.02.17 5 1 67066 0 19mb 19mb

There is a script that can be used to remove old indexes i.e. just keep 60 days worth of data.

The script is located here in the monitoring VM: */home/alfresco/elasticsearch-remove-old-indices.sh*

The script has the following syntax:

USAGE: ./elasticsearch-remove-old-indices.sh [OPTIONS]

OPTIONS:

-h Show this message

-i Indices to keep (default: 60 days)

-e Elasticsearch URL (default: http://localhost:9200)

-g Consistent index name (default: logstash)

-o Output actions to a specified file

Crontab entries can be added to automatically keep X number of indexes, if the index name has been changed, then make sure it matches the new name. As alfresco user execute the following command.

# crontab –e

Add the following entries to keep 90 days of data and save the file.

\* 0 \* \* \* /home/alfresco/elasticsearch-remove-old-indices.sh –i 90 -g logstash &>/dev/null

\* 0 \* \* \* /home/alfresco/elasticsearch-remove-old-indices.sh –i 90 -g tomcat &>/dev/null

\* 0 \* \* \* /home/alfresco/elasticsearch-remove-old-indices.sh –i 90 -g jmx &>/dev/null

A copy of elasticsearch-remove-old-indices.sh script can be found in Appendix B: elasticsearch-remove-old-indices.sh

## Setting Kibana to Read-Only mode

As an administrator you may not want all users to add new dashboards or change existing ones. A way to prevent changes is to lock dashboards/visualization/searches in Kibana.

This can be achieved by submitting the following request to ElasticSearch from command line on the Monitoring VM.

curl -XPUT 'localhost:9200/.kibana/\_settings' -d '{ "index.blocks.read\_only" : true }'

To allow changes again set the *“index.blocks.read\_only”* option to false.

curl -XPUT 'localhost:9200/.kibana/\_settings' -d '{ "index.blocks.read\_only" : false }'

# Appendix A: logstash.conf for VM

Logstash config file to be used in Monitoring VM /etc/logstash/conf.d/logstash.conf

#######################################

# Input data comes from Redis service #

#######################################

input {

redis {

host => "localhost"

type => "redis-input"

data\_type => "list"

key => "logstash"

}

}

####################################################################################################

# Here we start parsing the incoming messages and create fields we can use for reporting in Kibana #

####################################################################################################

################

# alfresco.log #

################

filter {

if [type] == "alfrescoLog" {

# Define multiline events from and entry that starts with a date to the next one

multiline {

pattern => "(^\d\d\d\d-\d\d-\d\d \d\d:\d\d:\d\d,\d\d\d)"

negate => true

what => "previous"

}

# replace double blank space with single blank space

mutate {

gsub => [

"message", " ", " "

]

}

# Match incoming log entries to fields

grok { match => [ "message", "%{TIMESTAMP\_ISO8601:logdate}\s\*%{LOGLEVEL} %{NOTSPACE:class} Got: %{NUMBER} in %{DATA:responseTime} ms"] }

grok { match => [ "message", "%{TIMESTAMP\_ISO8601:logdate}\s\*%{LOGLEVEL} %{NOTSPACE:class} %{WORD:action} %{WORD} %{NOTSPACE:script} in %{DATA:responseTime}ms"] }

grok { match => [ "message", "%{TIMESTAMP\_ISO8601:logdate}\s\*%{LOGLEVEL} %{NOTSPACE:class} %{NOTSPACE:thread} %{WORD:action} %{WORD} %{NOTSPACE:script} in %{DATA:responseTime}ms"] }

grok { match => [ "message", "%{TIMESTAMP\_ISO8601:logdate}\s\*%{LOGLEVEL:logLevel} %{NOTSPACE:class}\s\*%{GREEDYDATA:Msg}" ] }

mutate {

gsub => [ "duration", ",", "" ]

}

mutate {

convert => [ "duration", "float" ]

}

mutate {

gsub => [ "responseTime", ",", "" ]

}

mutate {

convert => [ "responseTime", "float" ]

}

date {

match => ["logdate" , "yyyy-MM-dd HH:mm:ss,SSS"]

target => "@timestamp"

}

grok {

match => [ "message", "Exception" ]

add\_tag => [ "Exception"]

}

grok {

match => [ "message", "NullPointerException" ]

add\_tag => [ "NullPointerException"]

}

grok {

match => [ "message", "OutOfMemoryError" ]

add\_tag => [ "OutOfMemoryError"]

}

grok {

match => [ "message", "Too many open files" ]

add\_tag => [ "TooManyOpenFiles"]

}

}

}

#############

# share.log #

#############

filter {

if [type] == "shareLog" {

# Define multiline events from and entry that starts with a date to the next one

multiline {

pattern => "(^\d\d\d\d-\d\d-\d\d \d\d:\d\d:\d\d,\d\d\d)"

negate => true

what => "previous"

}

# replace double blank space with single blank space

mutate {

gsub => [

"message", " ", " "

]

}

grok {

match => [ "message", "%{TIMESTAMP\_ISO8601:logdate}\s\*%{LOGLEVEL:logLevel} %{NOTSPACE:class} %{GREEDYDATA:data}" ]

add\_tag => [ "%{logLevel}" ]

add\_field => [ "Msg", "%{class} %{data}" ]

}

date {

match => ["logdate" , "yyyy-MM-dd HH:mm:ss,SSS"]

target => "@timestamp"

}

grok {

match => [ "message", "Exception" ]

add\_tag => [ "Exception"]

}

grok {

match => [ "message", "NullPointerException" ]

add\_tag => [ "NullPointerException"]

}

grok {

match => [ "message", "OutOfMemoryError" ]

add\_tag => [ "OutOfMemoryError"]

}

grok {

match => [ "message", "Too many open files" ]

add\_tag => [ "FileHandles"]

}

}

}

############

# solr.log #

############

filter {

if [type] == "solrLog" {

# Define multiline events from and entry that starts with a date to the next one

multiline {

pattern => "(^\d\d\d\d-\d\d-\d\d \d\d:\d\d:\d\d,\d\d\d)"

negate => true

what => "previous"

}

# replace double blank space with single blank space

mutate {

gsub => [

"message", " ", " "

]

}

grok {

match => [ "message", "%{TIMESTAMP\_ISO8601:logdate}\s\*%{LOGLEVEL:logLevel} %{NOTSPACE:class} %{GREEDYDATA:data}" ]

add\_tag => [ "%{logLevel}" ]

add\_field => [ "Msg", "%{class} %{data}" ]

}

date {

match => ["logdate" , "yyyy-MM-dd HH:mm:ss,SSS"]

target => "@timestamp"

}

grok {

match => [ "message", "Exception" ]

add\_tag => [ "Exception"]

}

grok {

match => [ "message", "NullPointerException" ]

add\_tag => [ "NullPointerException"]

}

grok {

match => [ "message", "OutOfMemoryError" ]

add\_tag => [ "OutOfMemoryError"]

}

grok {

match => [ "message", "Too many open files" ]

add\_tag => [ "FileHandles"]

}

}

}

#########################

# Filter for Server CPU #

#########################

filter {

if [type] == "AlfrescoCPU" {

grok {

match => [ "message", "%{NUMBER:AlfrescoCPU}" ]

#add\_field => [ "cpu", "%{Alfresco}" ]

add\_tag => [ "Alfresco" ]

}

mutate { convert => [ "cpu", "float" ] }

}

}

filter {

if [type] == "ServerCPU" {

grok {

match => [ "message", "us,%{NUMBER:cpu}" ]

add\_tag => [ "User" ]

}

grok {

match => [ "message", "sy,%{NUMBER:cpu}" ]

add\_tag => [ "System" ]

}

grok {

match => [ "message", "id,%{NUMBER:cpu}" ]

add\_tag => [ "Idle" ]

}

mutate { convert => [ "cpu", "float" ] }

}

}

############

# Commands #

############

filter {

if [type] == "command" {

grok { match => [ "message", "%{GREEDYDATA:output}"] }

}

}

#######

# jmx #

#######

# Identify cache entries that we want to convert to a value between 0 and 100

filter {

if [type] == "jmx" {

if [metric\_path] in [

"alfresco.queryResultCache.cumulative\_hitratio",

"alfresco.perSegFilter.cumulative\_hitratio",

"alfresco.filterCache.cumulative\_hitratio",

"alfresco.fieldValueCache.cumulative\_hitratio",

"alfresco.documentCache.cumulative\_hitratio",

"alfresco.alfrescoReaderCache.cumulative\_hitratio",

"alfresco.alfrescoPathCache.cumulative\_hitratio",

"alfresco.alfrescoOwnerCache.cumulative\_hitratio",

"alfresco.alfrescoDeniedCache.cumulative\_hitratio",

"alfresco.alfrescoAuthorityCache.cumulative\_hitratio",

"alfresco.contentDataTransactionalCache.HitMissRatio",

"alfresco.immutableEntityTransactionalCache.HitMissRatio",

"alfresco.aspectsTransactionalCache.HitMissRatio",

"alfresco.childByNameTransactionalCache.HitMissRatio",

"alfresco.nodesTransactionalCache.HitMissRatio",

"alfresco.propertiesTransactionalCache.HitMissRatio",

"alfresco.rootNodesTransactionalCache.HitMissRatio",

"alfresco.propertyClassTransactionalCache.HitMissRatio",

"alfresco.propertyUniqueContextTransactionalCache.HitMissRatio",

"alfresco.propertyValueTransactionalCache.HitMissRatio",

"alfresco.solrFacetNodeRefTransactionalCache.HitMissRatio",

"alfresco.tenantEntityTransactionalCache.HitMissRatio",

"alfresco.loadedResourceBundlesTransactionalCache.HitMissRatio",

"alfresco.messagesTransactionalCache.HitMissRatio",

"alfresco.personTransactionalCache.HitMissRatio",

"alfresco.resourceBundleBaseNamesTransactionalCache.HitMissRatio",

"alfresco.routingContentStoreTransactionalCache.HitMissRatio",

"alfresco.userToAuthorityTransactionalCache.HitMissRatio",

"alfresco.zoneToAuthorityTransactionalCache.HitMissRatio",

"alfresco.Server\_CPU.ProcessCpuLoad",

"alfresco.Server\_CPU.SystemCpuLoad"

] {

if [metric\_value\_number] {

ruby {

code => "event['metric\_value\_number'] = event['metric\_value\_number'] \* 100"

}

}

}

}

if [metric\_path] == "alfresco.Heap\_Memory.HeapMemoryUsage.used" {

mutate {

add\_tag => [ "used" ]

}

} else if [metric\_path] == "alfresco.Heap\_Memory.HeapMemoryUsage.committed" {

mutate {

add\_tag => [ "committed" ]

}

} else if [metric\_path] == "alfresco.Heap\_Memory.HeapMemoryUsage.max" {

mutate {

add\_tag => [ "max" ]

}

} else if [metric\_path] == "alfresco.Repo\_Server\_Mgmt.TicketCountNonExpired" {

mutate {

add\_tag => [ "Tickets\_Non\_Expired" ]

}

} else if [metric\_path] == "alfresco.Repo\_Server\_Mgmt.UserCountNonExpired" {

mutate {

add\_tag => [ "Users\_Non\_Expired" ]

}

} else if [metric\_path] == "alfresco.Share\_Active\_Sessions.activeSessions" {

mutate {

add\_tag => [ "Share\_Sessions" ]

}

} else if [metric\_path] == "alfresco.Operating\_System.TotalSwapSpaceSize" {

mutate {

add\_tag => [ "Total\_Swap" ]

}

} else if [metric\_path] == "alfresco.Operating\_System.FreeSwapSpaceSize" {

mutate {

add\_tag => [ "Free\_Swap" ]

}

} else if [metric\_path] == "alfresco.Operating\_System.FreePhysicalMemorySize" {

mutate {

add\_tag => [ "Free\_Physical" ]

}

} else if [metric\_path] == "alfresco.Operating\_System.TotalPhysicalMemorySize" {

mutate {

add\_tag => [ "Total\_Physical" ]

}

} else if [metric\_path] == "alfresco.Tomcat\_HTTP\_Threads.currentThreadsBusy" {

mutate {

add\_tag => [ "Threads\_Busy" ]

}

} else if [metric\_path] == "alfresco.Tomcat\_HTTP\_Threads.currentThreadCount" {

mutate {

add\_tag => [ "Threads\_Count" ]

}

} else if [metric\_path] == "alfresco.Tomcat\_HTTP\_Threads.MaxThreads" {

mutate {

add\_tag => [ "Max\_Threads" ]

}

} else if [metric\_path] == "alfresco.Tomcat\_AJP\_Threads.currentThreadsBusy" {

mutate {

add\_tag => [ "Threads\_Busy" ]

}

} else if [metric\_path] == "alfresco.Tomcat\_AJP\_Threads.currentThreadCount" {

mutate {

add\_tag => [ "Threads\_Count" ]

}

} else if [metric\_path] == "alfresco.Tomcat\_AJP\_Threads.MaxThreads" {

mutate {

add\_tag => [ "Max\_Threads" ]

}

} else if [metric\_path] == "alfresco.Tomcat\_SSL\_Threads.currentThreadsBusy" {

mutate {

add\_tag => [ "Threads\_Busy" ]

}

} else if [metric\_path] == "alfresco.Tomcat\_SSL\_Threads.currentThreadCount" {

mutate {

add\_tag => [ "Threads\_Count" ]

}

} else if [metric\_path] == "alfresco.Tomcat\_SSL\_Threads.MaxThreads" {

mutate {

add\_tag => [ "Max\_Threads" ]

}

} else if [metric\_path] == "alfresco.Workflow\_Information.NumberOfActivitiWorkflowDefinitionsDeployed" {

mutate {

add\_tag => [ "Workflow\_Definitions" ]

}

} else if [metric\_path] == "alfresco.Workflow\_Information.NumberOfActivitiWorkflowInstances" {

mutate {

add\_tag => [ "Workflow\_Instances" ]

}

} else if [metric\_path] == "alfresco.Workflow\_Information.NumberOfActivitiTaskInstances" {

mutate {

add\_tag => [ "Task\_Instances" ]

}

} else if [metric\_path] == "alfresco.DB\_Connection\_Pool.NumActive" {

mutate {

add\_tag => [ "Active\_DB\_Conns" ]

}

} else if [metric\_path] == "alfresco.DB\_Connection\_Pool.MaxActive" {

mutate {

add\_tag => [ "Max\_DB\_Conns" ]

}

} else if [metric\_path] == "alfresco.Operating\_System.OpenFileDescriptorCount" {

mutate {

add\_tag => [ "Open\_Files" ]

}

} else if [metric\_path] == "alfresco.Disk\_ContentStore.SpaceFree" {

mutate {

add\_tag => [ "Free\_disk\_space" ]

}

} else if [metric\_path] == "alfresco.Disk\_ContentStore.SpaceTotal" {

mutate {

add\_tag => [ "Total\_disk\_space" ]

}

}

# Convert string metric to numeric value

if [metric\_value\_string] {

mutate {

add\_field => [ "value", "%{metric\_value\_string}" ]

convert => [ "metric\_value\_number", "float" ]

remove\_field => [ "metric\_value\_string" ]

}

}

# Renames the 'metric\_value\_number' field to 'value'

mutate {

rename => { "metric\_value\_number" => "value" }

}

}

################################

# Filter for Tomcat Access log #

################################

filter {

if [type] == "TomcatAccessLog" {

grok {

match => [ "message", "%{IPORHOST:clientip} %{USER:ident} %{DATA:auth} \[%{HTTPDATE:timestamp}\] \"(%{WORD:verb} %{NOTSPACE:request} (HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})\" %{NUMBER:response} (?:%{NUMBER:bytes}|-) %{QS:referrer} %{QS:agent} %{NUMBER:responseTime:float} \"%{DATA:thread}\"" ]

}

date {

match => [ "timestamp" , "dd/MMM/YYYY:HH:mm:ss Z" ]

target => "@timestamp"

}

}

}

# Drop unwanted data

filter {

mutate {

remove\_field => [ 'message' ]

remove\_field => [ 'data' ]

remove\_field => [ 'logdate' ]

remove\_tag => [ '\_grokparsefailure' ]

}

}

###########################

# output to elasticsearch #

###########################

output {

if [type] in ["jmx", "AlfrescoCPU", "ServerCPU"] {

#Uncomment for debugging purposes

#stdout { codec => rubydebug }

elasticsearch {

index => "jmx-%{+YYYY.MM.dd}"

hosts => "127.0.0.1:9200"

}

} else if [type] == "postgres" {

#Uncomment for debugging purposes

#stdout { codec => rubydebug }

elasticsearch {

index => "postgres-%{+YYYY.MM.dd}"

hosts => "127.0.0.1:9200"

}

} else if [type] == "TomcatAccessLog" {

#Uncomment for debugging purposes

#stdout { codec => rubydebug }

elasticsearch {

index => "tomcat-%{+YYYY.MM.dd}"

hosts => "127.0.0.1:9200"

}

} else {

#Uncomment for debugging purposes

stdout { codec => rubydebug }

elasticsearch {

hosts => "127.0.0.1:9200"

}

}

}

# 

# Appendix B: elasticsearch-remove-old-indices.sh

Script to remove old indices, to be place in /home/alfresco/ folder.

#!/bin/bash

# elasticsearch-remove-old-indices.sh

#

# Delete logstash format indices from elasticsearch maintaining only a

# specified number.

# http://logstash.net

# http://www.elasticsearch.org

#

# Inspiration:

# http://tech.superhappykittymeow.com/?p=296

#

# Must have access to the specified elasticsearch node.

usage()

{

cat << EOF

elasticsearch-remove-old-indices.sh

Compares the current list of indices to a configured value and deletes any

indices surpassing that value. Sort is lexicographical; the first n of a 'sort

-r' list are kept, all others are deleted.

USAGE: ./elasticsearch-remove-old-indices.sh [OPTIONS]

OPTIONS:

-h Show this message

-i Indices to keep (default: 14)

-e Elasticsearch URL (default: http://localhost:9200)

-g Consistent index name (default: logstash)

-o Output actions to a specified file

EXAMPLES:

./elasticsearch-remove-old-indices.sh

Connect to http://localhost:9200 and get a list of indices matching

'logstash'. Keep the top lexicographical 14 indices, delete any others.

./elasticsearch-remove-old-indices.sh -e "http://es.example.com:9200" \

-i 28 -g my-logs -o /mnt/es/logfile.log

Connect to http://es.example.com:9200 and get a list of indices matching

'my-logs'. Keep the top 28 indices, delete any others. When using a custom

index naming scheme be sure that a 'sort -r' places the indices you want to

keep at the top of the list. Output index deletes to /mnt/es/logfile.log.

EOF

}

# Defaults

ELASTICSEARCH="http://localhost:9200"

KEEP=0

#GREP=".kibana"

GREP="tomcat"

#GREP="logstash"

#GREP="jmx"

# Validate numeric values

RE\_D="^[0-9]+$"

while getopts ":i:e:g:o:h" flag

do

case "$flag" in

h)

usage

exit 0

;;

i)

if [[ $OPTARG =~ $RE\_D ]]; then

KEEP=$OPTARG

else

ERROR="${ERROR}Indexes to keep must be an integer.\n"

fi

;;

e)

ELASTICSEARCH=$OPTARG

;;

g)

GREP=$OPTARG

;;

o)

LOGFILE=$OPTARG

;;

?)

usage

exit 1

;;

esac

done

# If we have errors, show the errors with usage data and exit.

if [ -n "$ERROR" ]; then

echo -e $ERROR

usage

exit 1

fi

# Get the indices from elasticsearch

INDICES\_TEXT=`curl -s "$ELASTICSEARCH/\_cat/indices?pretty=true" | grep $GREP | sort -r | awk {'print $3'}`

if [ -z "$INDICES\_TEXT" ]; then

echo "No indices returned containing '$GREP' from $ELASTICSEARCH."

exit 1

fi

# If we are logging, make sure we have a logfile TODO - handle errors here

if [ -n "$LOGFILE" ] && ! [ -e $LOGFILE ]; then

touch $LOGFILE

fi

# Delete indices

declare -a INDEX=($INDICES\_TEXT)

if [ ${#INDEX[@]} -gt $KEEP ]; then

for index in ${INDEX[@]:$KEEP};do

# We don't want to accidentally delete everything

if [ -n "$index" ]; then

if [ -z "$LOGFILE" ]; then

curl -s -XDELETE "$ELASTICSEARCH/$index/" > /dev/null

else

echo `date "+[%Y-%m-%d %H:%M] "`" Deleting index: $index." >> $LOGFILE

curl -s -XDELETE "$ELASTICSEARCH/$index/" >> $LOGFILE

fi

fi

done

fi

exit 0