Docker Image for

Alfresco Monitoring with ELK

Contents

[Contents 2](#_Toc481129670)

[1. Project Objective 3](#_Toc481129671)

[2. Install Docker on Host machine 3](#_Toc481129672)

[3. Virtual Memory 3](#_Toc481129673)

[4. Download “Docker-ELK-Alfresco-Monitoring” container software 3](#_Toc481129674)

[5. Creating the Docker Image 4](#_Toc481129675)

[6. Creating the Docker Container 5](#_Toc481129676)

[7. Starting the Docker Container 5](#_Toc481129677)

[8. Configuring the logstash-agent 6](#_Toc481129678)

[9. Configuring Alfresco to generate data for monitoring 6](#_Toc481129679)

[10.1 Alfresco Logs 6](#_Toc481129690)

[10.2 Document Transformations 7](#_Toc481129691)

[10.3 Tomcat Access Logs 8](#_Toc481129692)

[10.4 Solr Searches 9](#_Toc481129693)

[10.5 Alfresco auditing 11](#_Toc481129694)

[10.6 Database Monitoring 11](#_Toc481129695)

[P6spy 12](#_Toc481129696)

[10. Starting / Stopping the logstash agent 14](#_Toc481129697)

[11. Accessing the Dashboard 14](#_Toc481129698)

[12. Accessing the container 15](#_Toc481129699)

[13. Stopping the container 16](#_Toc481129700)

[14. Removing the Docker Container 16](#_Toc481129701)

[15. Removing the Docker Image 16](#_Toc481129702)

[Happy Monitoring!!! 16](#_Toc481129703)

# Project Objective

The aim of this Docker image is to provide a full ELK (Elasticsearch, Logstash and Kibana) environment with the necessary configuration and scripts to collect and present data to monitor your Alfresco application.

Elastic tools can ease the processing and manipulation of large amounts of data collected from logs, operating system, network, etc.

Elastic tools can be used to search for data such as errors, exceptions and debug entries and to present statistical information such as throughput and response times in a meaningful way. This information is very useful when monitoring and troubleshooting Alfresco systems.

# Install Docker on Host machine

Install Docker on your host machine (server) as per Docker website. Please note the Docker Community Edition is sufficient to run this project (<https://www.docker.com/community-edition>)

# Virtual Memory

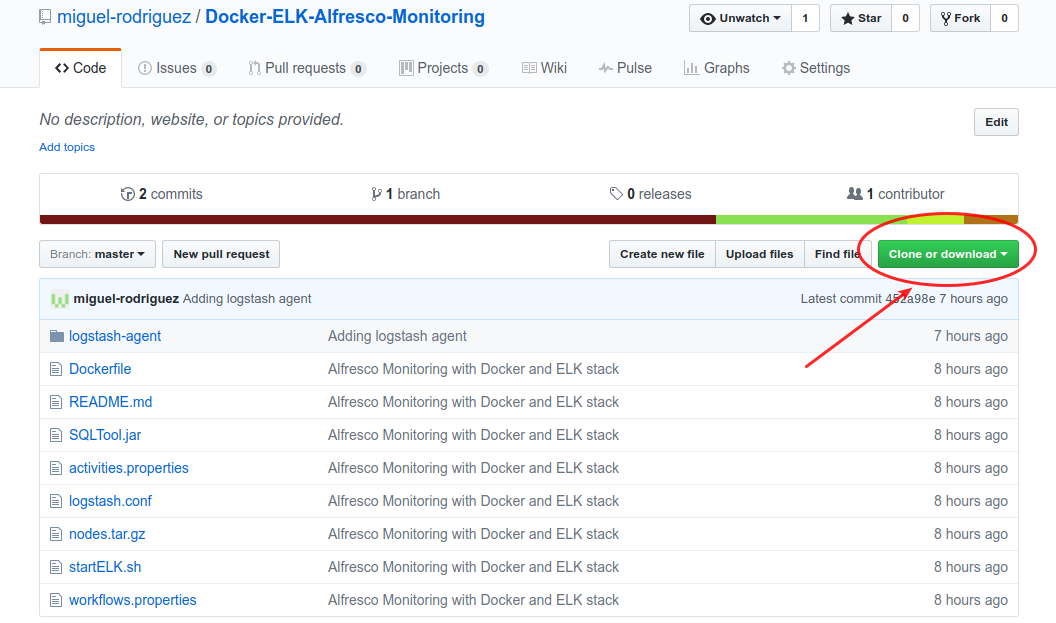
Elasticsearch uses a hybrid mmapfs / niofs directory by default to store its indices. The default operating system limits on mmap counts is likely to be too low, which may result in out of memory exceptions. On Linux, you can increase the limits by running the following command as root on the host machine:

# sysctl -w vm.max\_map\_count=262144

To set this value permanently, update the vm.max\_map\_count setting in /etc/sysctl.conf. To verify the value has been applied run sysctl vm.max\_map\_count.

# Download “Docker-ELK-Alfresco-Monitoring” container software

Download the software to create the Docker container from GitHub: <https://github.com/miguel-rodriguez/Docker-ELK-Alfresco-Monitoring> and extract the files to the file system.



# Creating the Docker Image

Before creating the Docker image we need to configure access to Alfresco’s database from the Docker container. Assuming the files have been extracted to /opt/docker-projects/Docker-ELK-Alfresco-Monitoring-master, edit files activities.properties and workflows.properties and set the access to the DB server as appropriate, for example:

#postgresql settings

db\_type=postgresql

db\_url=jdbc:postgresql://172.17.0.1:5432/alfresco

db\_user=alfresco

db\_password=admin

Please make sure the database server allows for remote connections to Alfresco’s database. A couple of examples how to configure the database are shown here:

1. For MySQL

Access your database server as an administrator and grant the correct permissions i.e.

# mysql -u root -p

* grant all privileges on alfresco.\* to alfresco@'%' identified by 'admin';

The grant command is granting access to all tables in ‘alfresco’ database to ‘alfresco’ user from any host using ‘admin’ password.

1. For PostgreSQL

Change the file ‘postgresql.conf’ to listen on all interfaces

listen\_addresses = '\*'

then add an entry in file ‘pg\_hba.conf’ to allow connections from any host

host all all 0.0.0.0/0 trust

Restart PostgreSQL database server to pick up the changes.

We have installed a small java application inside the container in /opt/activities folder that executes calls against the database configured in /opt/activities/activities.properties file.

For example to connect to PostgreSQL we have the following settings:

db\_type=postgresql

db\_url=jdbc:postgresql://172.17.0.1:5432/alfresco

db\_user=alfresco

db\_password=admin

We also need to set the timezone in the container, this can be done by editing the following entry in the startELK.sh script

export TZ=GB

From the command line execute the following command to create the Docker image:

# docker build --tag=alfresco-elk /opt/docker-projects/Docker-ELK-Alfresco-Monitoring-master/

Sending build context to Docker daemon 188.9MB

Step 1/33 : FROM sebp/elk:530

530: Pulling from sebp/elk

…..

# Creating the Docker Container

Once the Docker image has been created we can create the container from it by executing the following command:

# docker create -it -p 5601:5601 --name alfresco-elk alfresco-elk:latest

# Starting the Docker Container

Once the Docker container has been created it can be started with the following command:

# docker start alfresco-elk

Verify the ELK stack is running by accessing Kibana on [http://localhost:5601](http://localhost:5601/) on the host machine.

At this point Elasticsearch and Kibana do not have any data…so we need to get Alfresco’s logstash agent up and running to feed some data to Elasticsearch.

# Configuring the logstash-agent

The logstash agent consists of logstash and some other scripts to capture entries from Alfresco log files, JVM stats using jstatbeat (<https://github.com/cero-t/jstatbeat>), entries from Alfresco audit tables, DB slow queries, etc.

Copy the logstash-agent folder to a directory on all the servers running Alfresco or Solr applications.

Assuming you have copied logstash-agent folder to /opt/logstash-agent, edit the file /opt/alfresco-agent/run\_logstash.sh and set the following properties according to your own settings

export tomcatLogs=/opt/alfresco/tomcat/logs

export logstashAgentDir=/opt/logstash-agent

export logstashAgentLogs=${logstashAgentDir}/logs

export alfrescoELKServer=172.17.0.2

# Configuring Alfresco to generate data for monitoring

Alfresco needs some additional configuration to produce data to be sent to the monitoring Docker container.



## Alfresco Logs

Alfresco logs i.e. alfresco.log, share.log, solr.log or the equivalent catalina.out can be parsed to provide information such as number of errors or exceptions over a period of time. We can also search these logs for specific data.

The first thing is to make sure the logs are displaying the full date time format at the beginning of each line. This is important so we can display the entries in the correct order.

Make sure in your log4j properties files (there is more than one) the file layout pattern is as follows:

log4j.appender.File.layout.ConversionPattern=**%d{yyyy-MM-dd} %d{ABSOLUTE}** %-5p [%c] [%t] %m%n

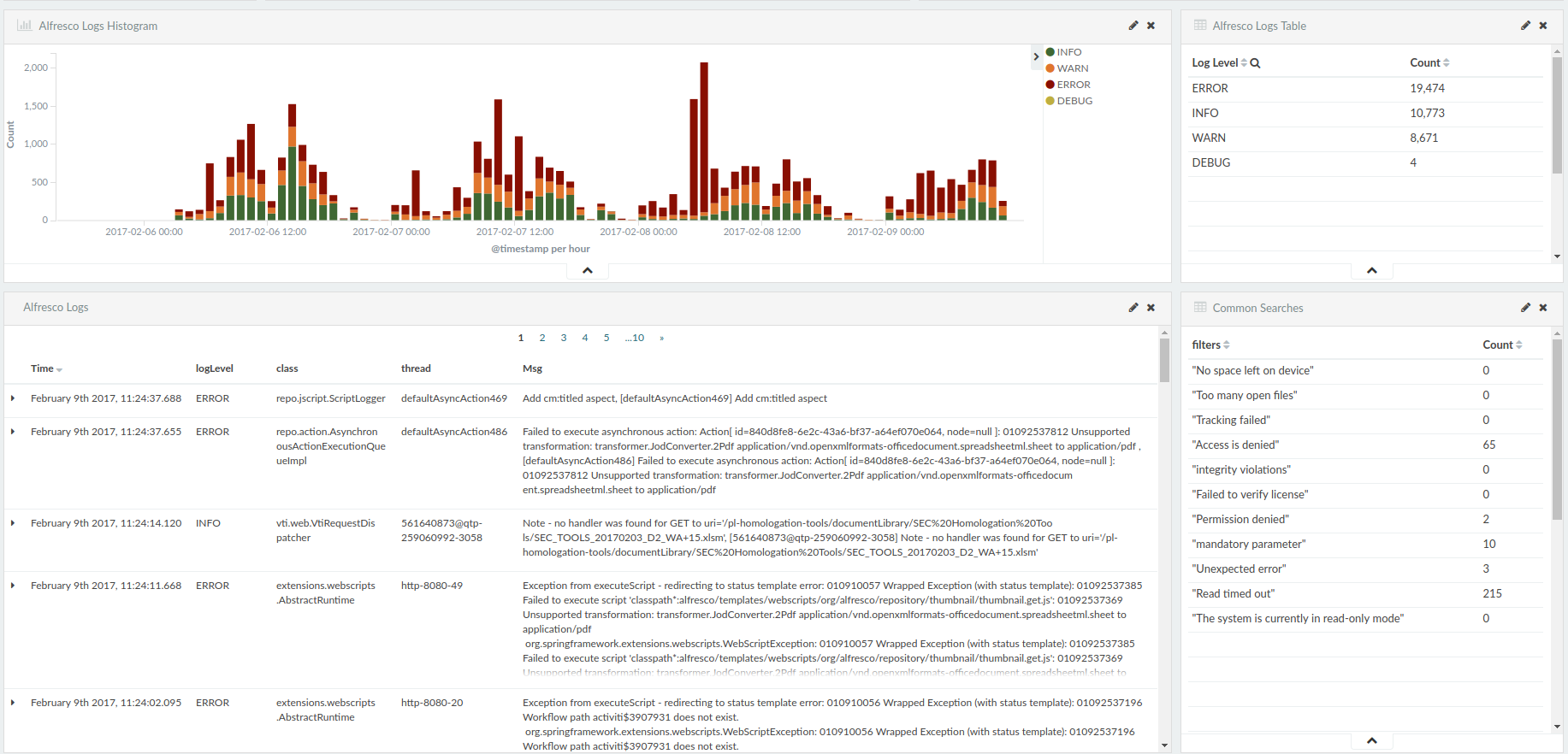
This will produce log entries with the date at the beginning of the line as this one:

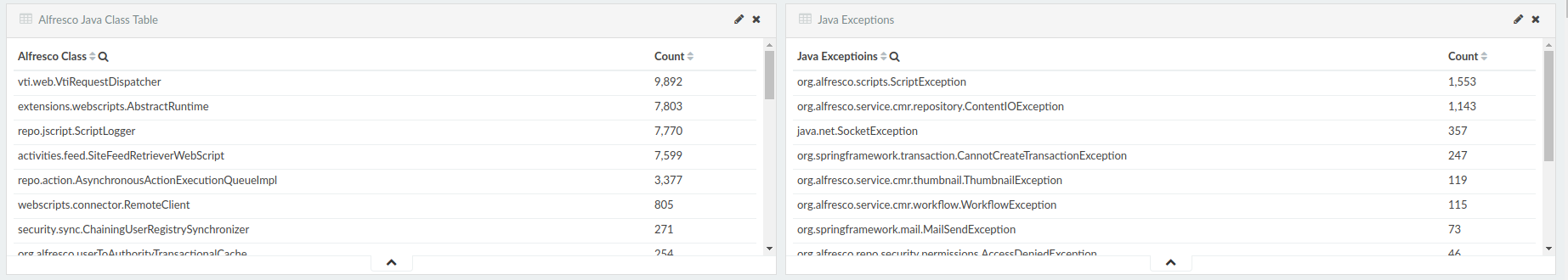
**2016-09-12 12:16:28,460** INFO [org.alfresco.repo.admin] [localhost-startStop-1] Connected to database PostgreSQL version 9.3.6

**Important Note:** If you upload catalina files then don’t upload alfresco (alfresco, share, solr) log files for the same time period since they contain the same entries and you will end up with duplicate entries in the Log Analyser tool.

Once the logs are processed the resulting data is shown:

* Number errors, warnings, debug, fatal messages, etc over time
* Total number of errors, warnings, debug, fatal messages, etc
* Common messages that may reflect issues with the application
* Number of entries grouped by java class
* Number of exceptions logged
* All log files are searchable using ES (Elasticsearch) search syntax





## Document Transformations

Alfresco performs document transformations for document previews, thumbnails, indexing content, etc. To monitor document transformations enable logging for class “*TransformerLog”* by adding the following line to *tomcat/shared/classes/alfresco/extension/custom-log4j.properties* on all alfresco nodes:

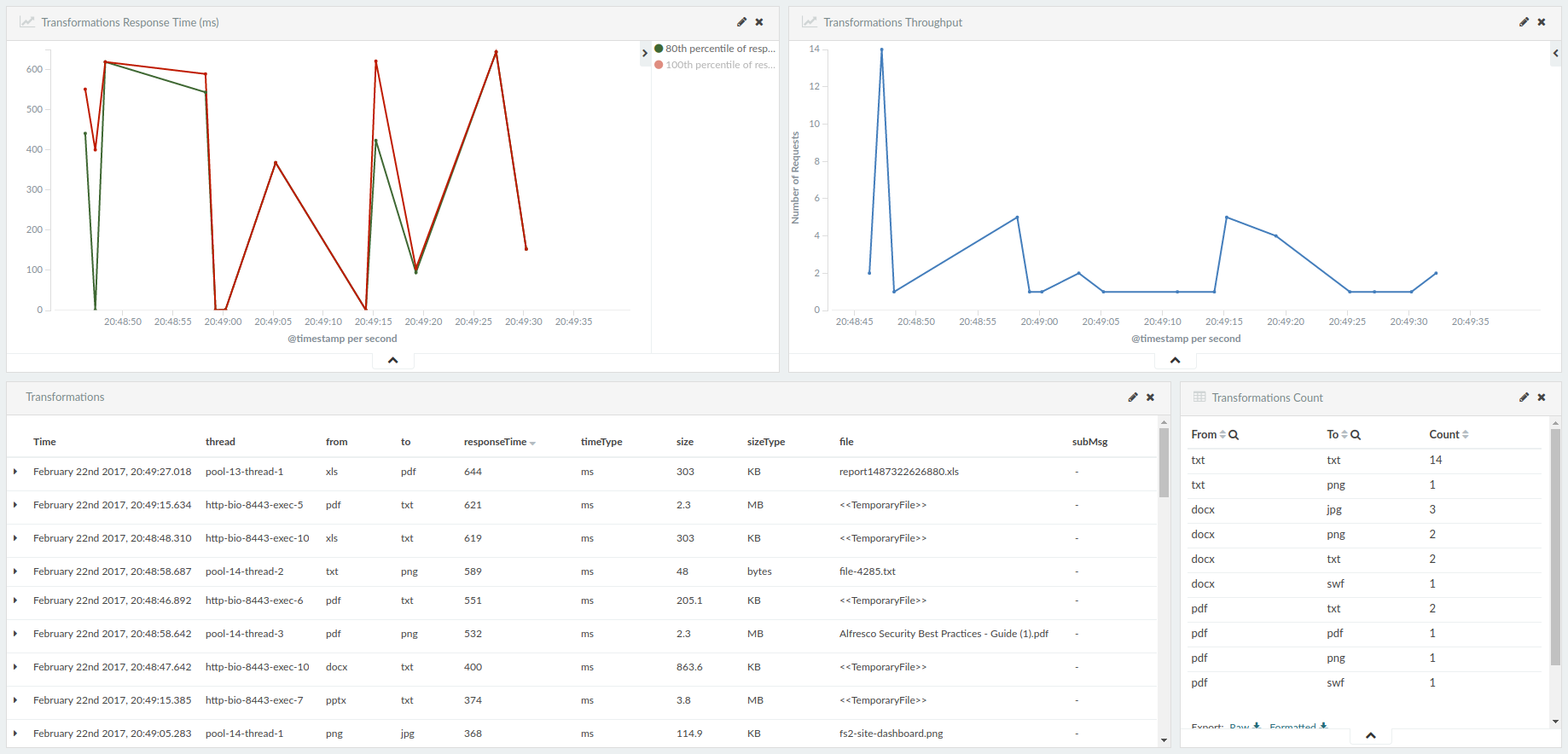
|  |
| --- |
| log4j.logger.org.alfresco.repo.content.transform.TransformerLog=debug |

The following is a sample output from alfresco.log file showing document transformation times, document extensions, transformer used, etc.

|  |
| --- |
| 2016-07-14 18:24:56,003 DEBUG [content.transform.TransformerLog] [pool-14-thread-1] 0 **xlsx png** INFO **Calculate\_Memory\_Solr Beta 0.2.xlsx** 200.6 KB **897 ms** complex.JodConverter.Image<<Complex>> |

Once Alfresco logs are processed the following data is shown for transformations:

* Response time of transformation requests over time
* Transformation throughput
* Total count of transformations grouped by file type
* Document size, transformation time, transformer used, etc



## Tomcat Access Logs

Tomcat access logs can be used to monitor HTTP requests, throughput and response times. In order to get the right data format in the logs we need to add/replace the “Valve” entry in tomcat/conf/server.xml file, normally located at the end of the file, with this one 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"

/>

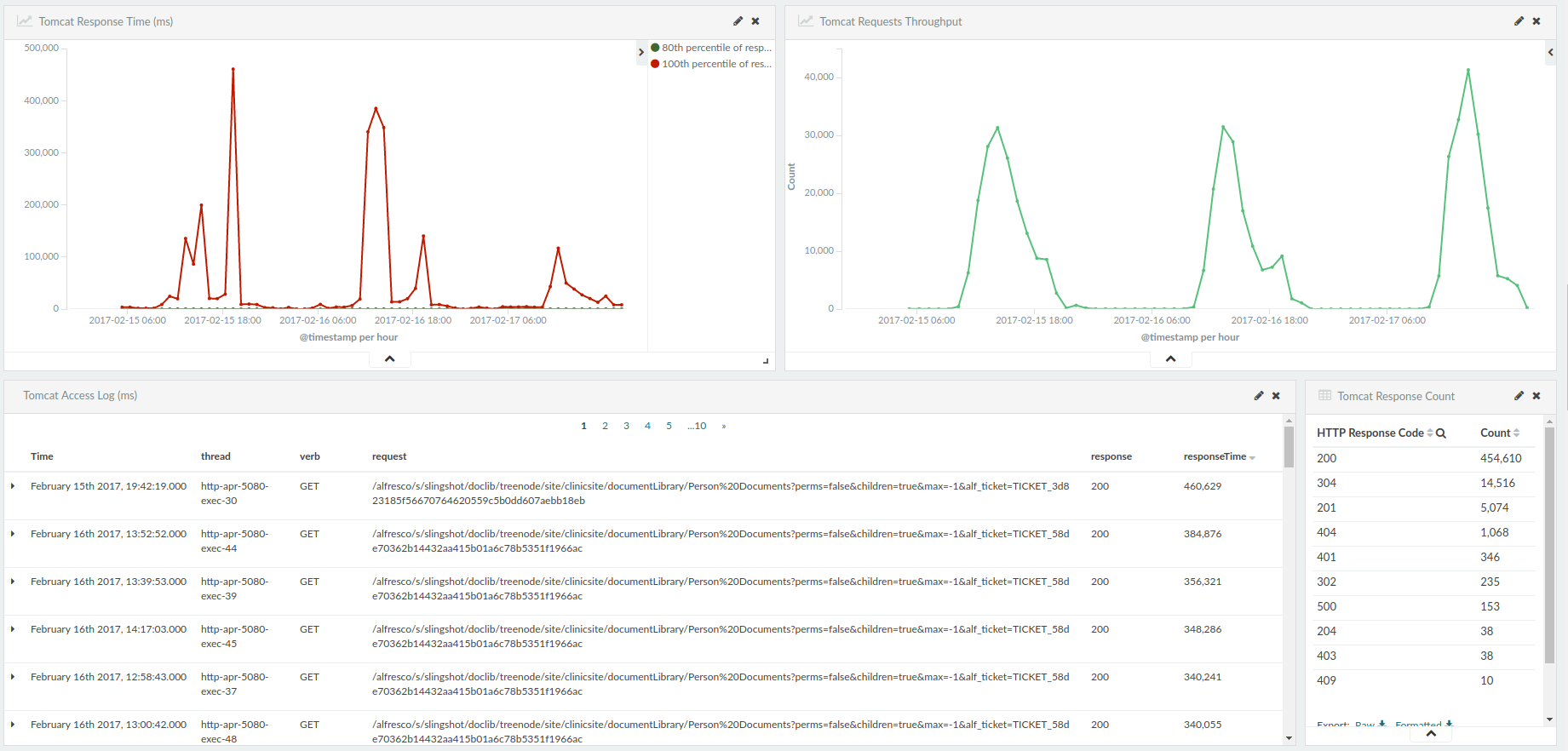
For further clarification on the log pattern refer to: [https://tomcat.apache.org/tomcat-7.0-doc/config/valve.html#Access\_Logging](https://tomcat.apache.org/tomcat-7.0-doc/config/valve.html" \l "Access_Logging)

Sample output from tomcat access log under tomcat/logs directory. The important fields here are the HTTP resquest, the HTTP response status i.e. 200 and the time taken to process the request i.e. 33 milliseconds

|  |
| --- |
| 127.0.0.1 - CN=Alfresco Repository Client, OU=Unknown, O=Alfresco Software Ltd., L=Maidenhead, ST=UK, C=GB [14/Jul/2016:18:49:45 +0100] "POST **/alfresco/service/api/solr/modelsdiff** HTTP/1.1" **200** 37 "-" "Spring Surf via Apache HttpClient/3.1" **33** "http-bio-8443-exec-10" |

Once the Tomcat access logs are processed the following data is shown:

* Response time of HTTP requests over time
* HTTP traffic throughput
* Total count of responses grouped by HTTP response code
* Tomcat access logs files are searchable using ES (Elasticsearch) search syntax



## Solr Searches

We can monitor Solr queries and response times by enabling debug for class *SolrQueryHTTPClient* by adding the following entry to *tomcat/shared/classes/alfresco/extension/custom-log4j.properties* on all Alfresco (front end) nodes:

|  |
| --- |
| log4j.logger.org.alfresco.repo.search.impl.solr.SolrQueryHTTPClient=debug |

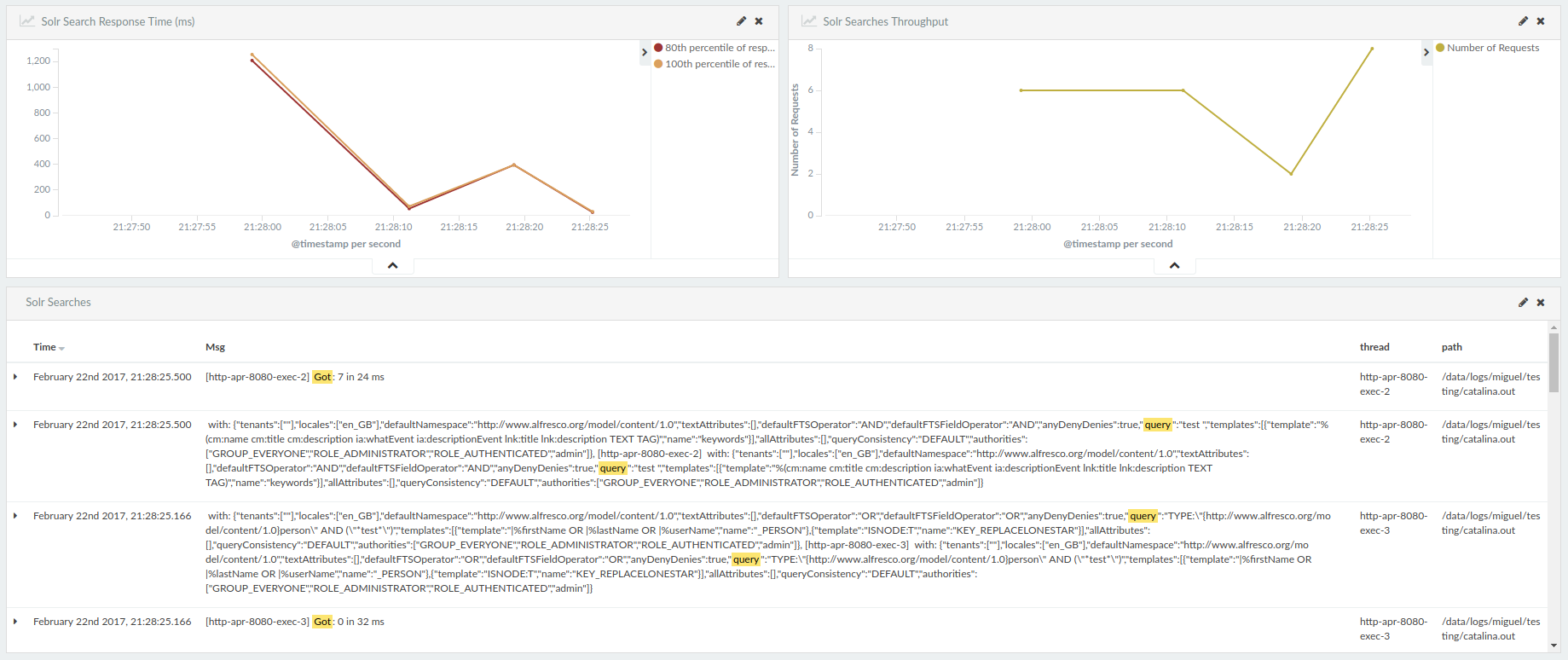
Sample output from alfresco.log file showing Solr searches response times:

|  |
| --- |
| DEBUG [impl.solr.SolrQueryHTTPClient] [**http-apr-8080-exec-6**] with: {"queryConsistency":"DEFAULT","textAttributes":[],"allAttributes":[],"templates":[{"template":"%(cm:name cm:title cm:description ia:whatEvent ia:descriptionEvent lnk:title lnk:description TEXT TAG)","name":"keywords"}],"authorities":["GROUP\_EVERYONE","ROLE\_ADMINISTRATOR","ROLE\_AUTHENTICATED","admin"],"tenants":[""],"**query**":"((test.txt AND (+TYPE:\"cm:content\" +TYPE:\"cm:folder\")) AND -TYPE:\"cm:thumbnail\" AND -TYPE:\"cm:failedThumbnail\" AND -TYPE:\"cm:rating\") AND NOT ASPECT:\"sys:hidden\"","locales":["en"],"defaultNamespace":"http://www.alfresco.org/model/content/1.0","defaultFTSFieldOperator":"OR","defaultFTSOperator":"OR"}  2016-03-19 19:55:54,106  DEBUG [impl.solr.SolrQueryHTTPClient] [**http-apr-8080-exec-6**] **Got: 1 in 21 ms** |

**Note:** There is no specific transaction id to correlate the Solr search to the corresponding response. The best way to do this is to look at the time when the search and response were logged together with the java thread name, this should give you a match for the query and its response.

Once Alfresco logs are processed the following data is shown for Solr searches:

* Response time for Solr searches over time
* Solr searches throughput
* Solr queries, number of results found and individual response times



## Alfresco auditing

If you want to audit Alfresco access you can enable auditing by adding the following entries to alfresco-global.properties file:

# Enable auditing

audit.enabled=true

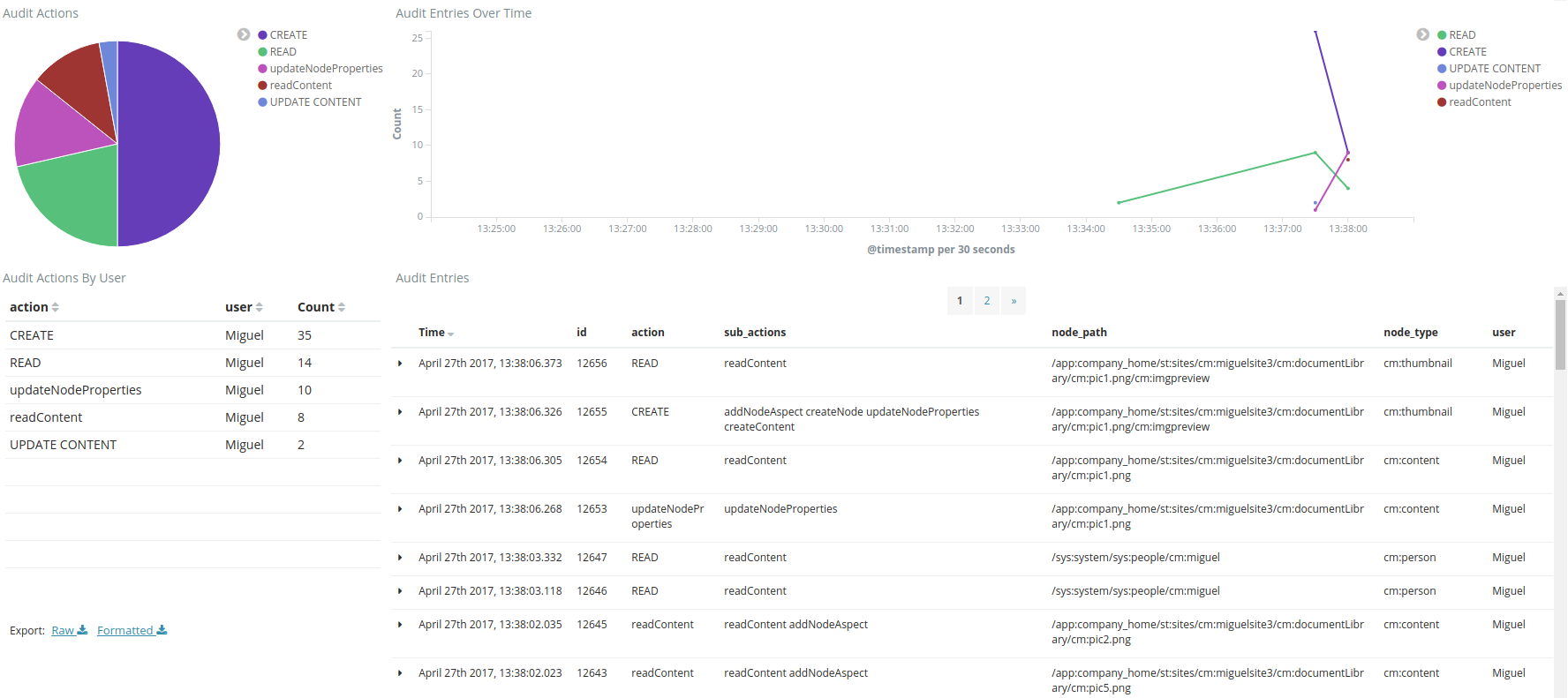
audit.alfresco-access.enabled=true

audit.tagging.enabled=true

audit.alfresco-access.sub-actions.enabled=true

audit.cmischangelog.enabled=true

Now you can monitor audit access entries.



**Note**: Only one of the logstash agents should collect Alfresco's audit data since the script gathers data for the whole cluster/solution. So edit the file logstash\_agent/run\_logstash.sh in one of the other Alfresco nodes and set the variable collectAuditData to "yes" as indicated below:

collectAuditData="yes"

## Database Monitoring

Database performance can be monitored with two different tools: p6spy and packetbeats. The main difference between these tools is that p6spy acts as a proxy jdbc driver and packetbeat is a network traffic sniffer. Also packetbeat can only sniff traffic for MySQL and PostgreSQL databases while p6spy can also do Oracle among others.

### P6spy

P6spy software is delivered as a jar file that needs to be placed in the application class path i.e. tomcat/lib/ folder. There are 3 steps to get p6spy configured and running.

1. Place p6spy jar file in tomcat/lib/ folder
2. Create spy.properties file also in tomcat/lib/folder with the following configuration

modulelist=com.p6spy.engine.spy.P6SpyFactory,com.p6spy.engine.logging.P6LogFactory,com.p6spy.engine.outage.P6OutageFactory

appender=com.p6spy.engine.spy.appender.FileLogger

deregisterdrivers=true

dateformat=MM-dd-yy HH:mm:ss:SS

appender=com.p6spy.engine.spy.appender.FileLogger

autoflush=true

append=true

useprefix=true

# Update driver list correct driver i.e.

# driverlist=oracle.jdbc.OracleDriver

# driverlist=org.mariadb.jdbc.Driver

# driverlist=org.postgresql.Driver

driverlist=org.postgresql.Driver

# Location where spy.log file will be created

logfile=/opt/logstash-agent/logs/spy.log

# Set the execution threshold to log queries taking longer than 1000 milliseconds (slow queries only)

executionThreshold=1000

**Note**: if there are no queries taking longer than the value in executionThreshod (in milliseconds) then the file will not be created.

**Note**: set the “logfile” variable to the logs folder inside the logstash-agent path as shown above.

1. Add entry to tomcat/conf/Catalina/localhost/alfresco.xml file

Example for PostgreSQL

<Resource

defaultTransactionIsolation="-1"

defaultAutoCommit="false"

maxActive="275"

initialSize="10"

password="admin"

username="alfresco"

url="jdbc:p6spy:postgresql://localhost/p6spy:alfresco"

driverClassName="com.p6spy.engine.spy.P6SpyDriver"

type="javax.sql.DataSource"

auth="Container"

name="jdbc/dataSource"

/>

Example for Oracle

<Resource

defaultTransactionIsolation="-1"

defaultAutoCommit="false"

maxActive="275"

initialSize="10"

password="admin"

username="alfresco"

url="jdbc:p6spy:oracle:thin:@192.168.56.101:1521:XE"

driverClassName="com.p6spy.engine.spy.P6SpyDriver"

type="javax.sql.DataSource"

auth="Container"

name="jdbc/dataSource"

/>

Example for MariaDB

<Resource

defaultTransactionIsolation="-1"

defaultAutoCommit="false"

maxActive="275"

initialSize="10"

password="admin"

username="alfresco"

url="jdbc:p6spy:postgresql://localhost/p6spy:alfresco"

driverClassName="com.p6spy.engine.spy.P6SpyDriver"

type="javax.sql.DataSource"

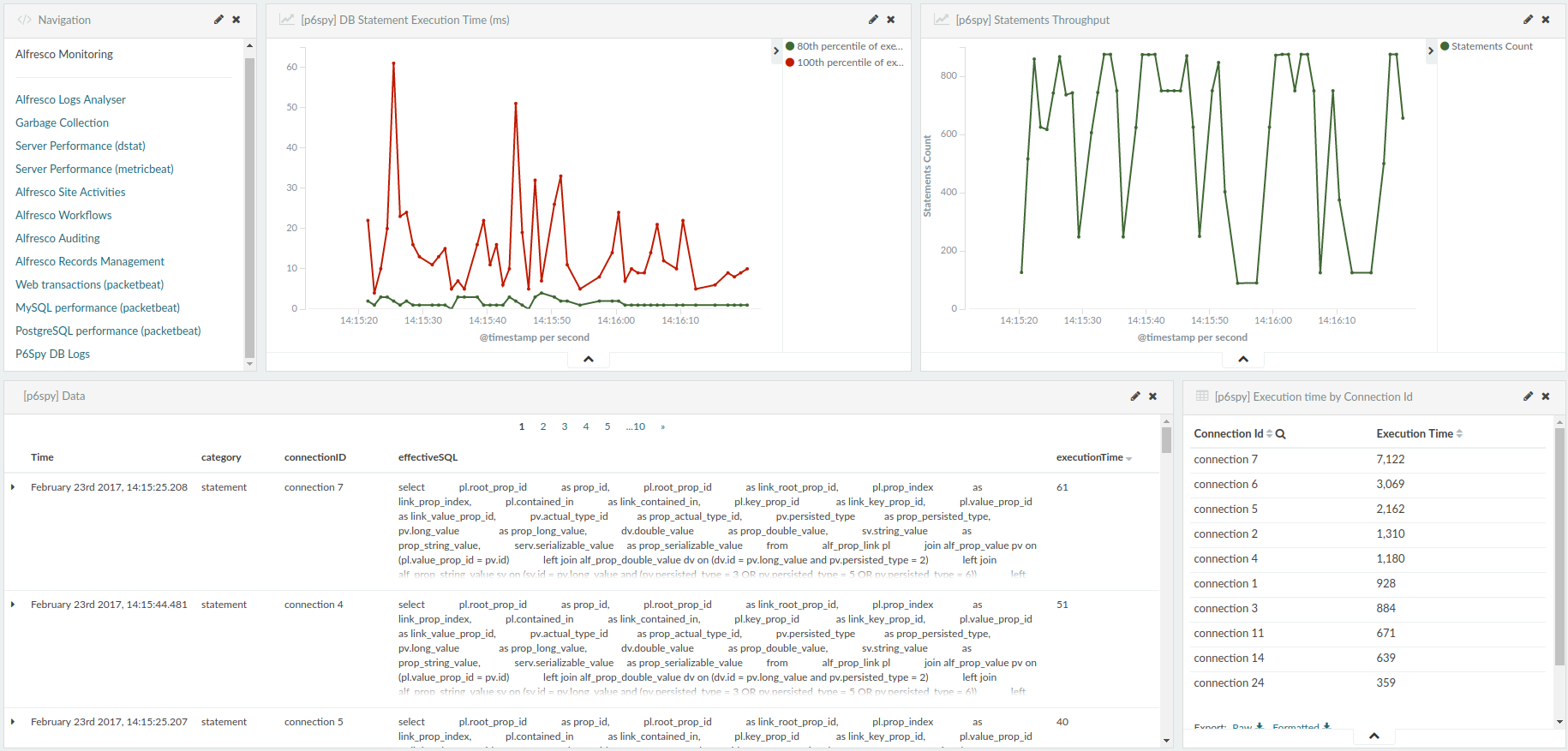
auth="Container"

name="jdbc/dataSource"

/>

Once the spy.log file has been processed the following information is show:

* DB Statements execution time over time
* DB Statements throughput over time
* Table showing individual DB statements and execution times
* DB execution times by connection id



# Starting / Stopping the logstash agent

The logstash agent script can be started from the command line with "./run\_logstash.sh start" as shown below:

./run\_logstash.sh start

Starting logstash

Starting jstatbeat

Starting dstat

Staring audit access script

and can be stopped with the command "./run\_logstash.sh stop" as shown below:

./run\_logstash.sh stop

Stopping logstash

Stopping jstatbeat

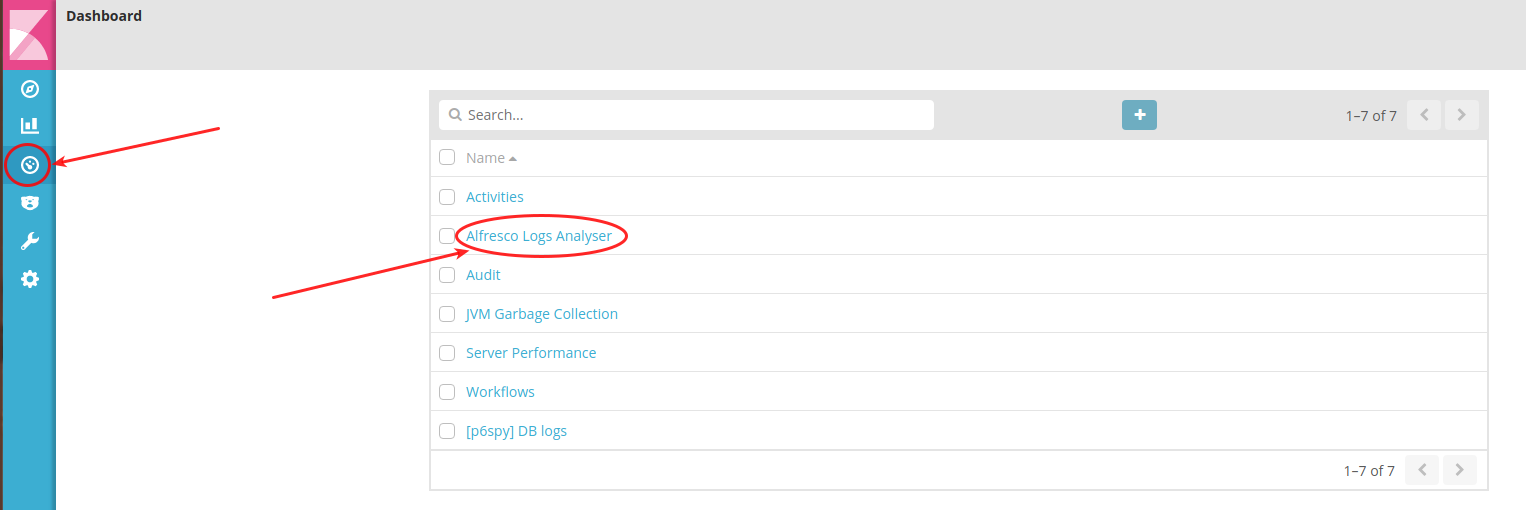
Stopping dstat

Stopping audit access script

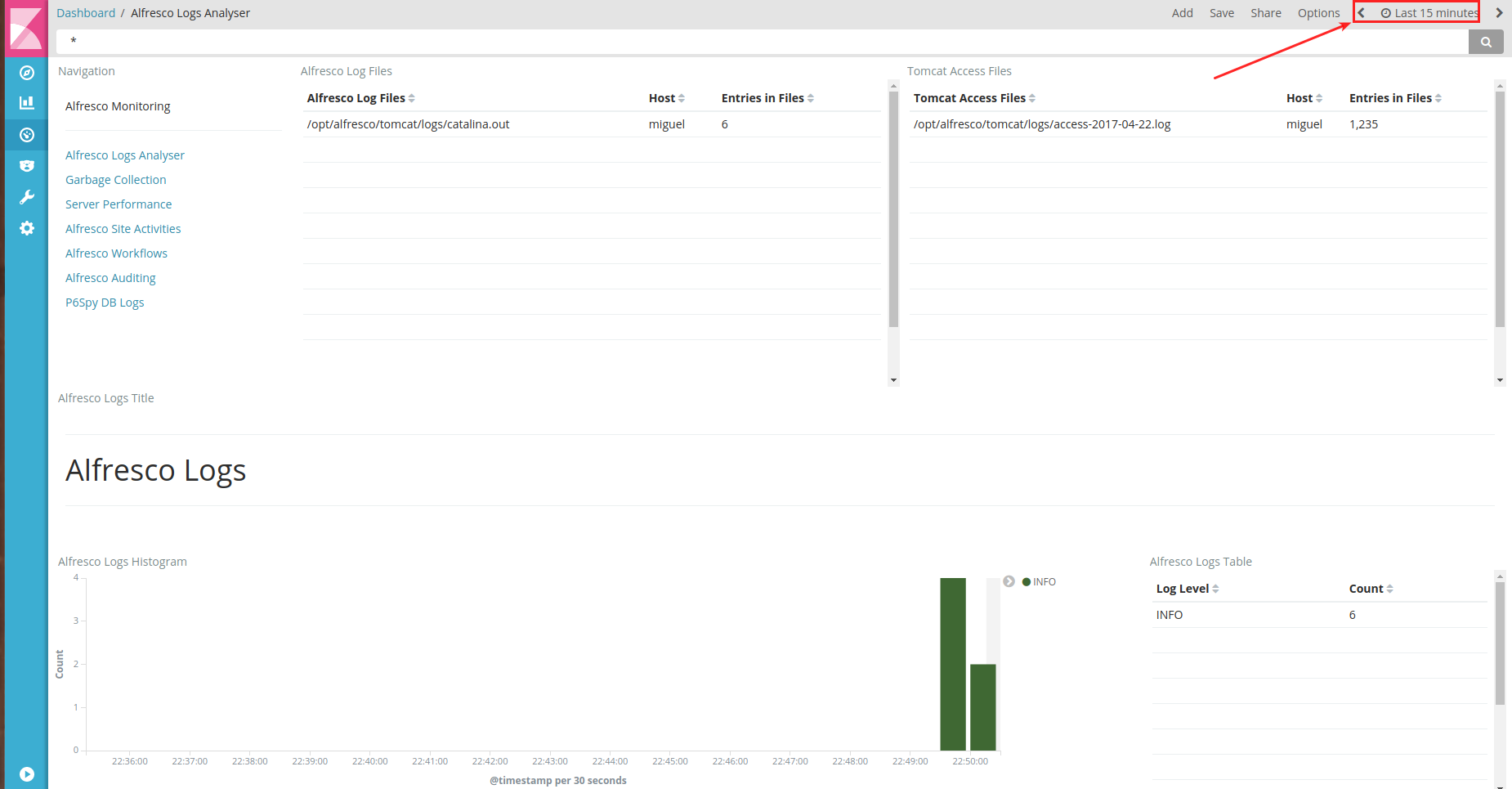
**Note**: If Alfresco is restarted then restart logstash-agent. This is needed since there is a script collecting GC data for a specific process id...restarting Alfresco changes the process id of the java process and therefore we need to restart the GC script.

# Accessing the Dashboard

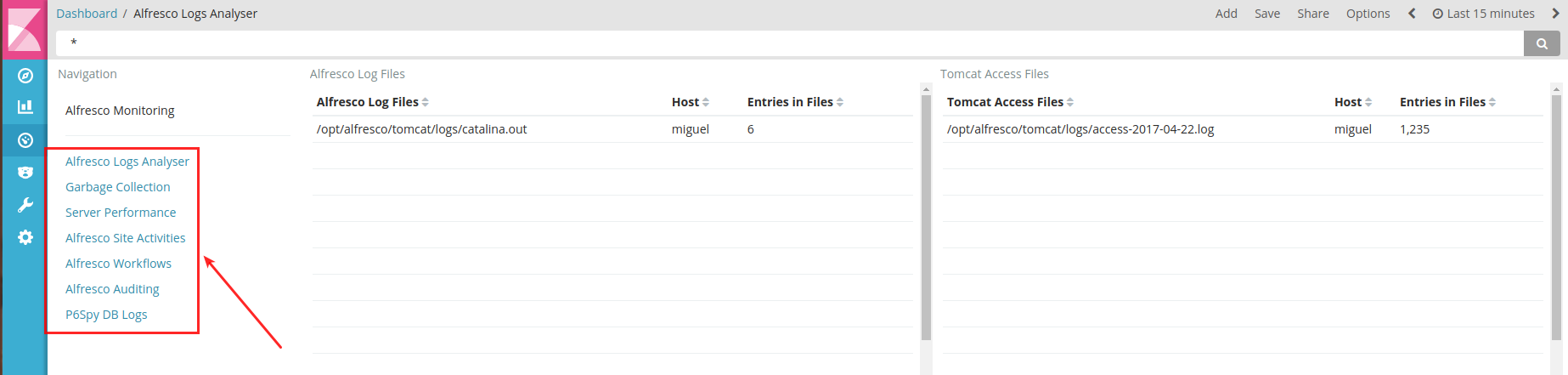
Finally access the dashboard by going to this URL http://<docker container IP>:5601 (use the IP of the server where you installed the Docker container) and clicking on the “Dashboard” link on the left panel and then click on the “Activities” link.



The data should be available for the selected time period.



Navigate to the other dashboards by clicking on the appropriate link



# Accessing the container

To enter the running container use the following command:

# docker exec -i -t alfresco-elk bash

And to exit the container just type “exit” and you will find yourself back on the host machine.

# Stopping the container

To stop the container from running type the following command on the host machine:

# docker stop alfresco-elk

# Removing the Docker Container

To delete the container you first need to stop the container and then run the following command:

# docker rm alfresco-elk

# Removing the Docker Image

To delete the container you first need to stop the container and then run the following command:

# docker rmi alfresco-elk:latest

# Happy Monitoring!!!