Alfresco® Simple + Smart™

Guide to Alfresco Monitoring

alfresco.com

San Mateo

Atlanta

London

Maidenhead

Sydney

Tokyo

UK Headquarters
Bridge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600

alfresco.com
San Mateo
Atlanta
London
Maidenhead
Sydney
Tokyo

Contents

Introduction 3
What do we need to monitor? 3
Hardware resources 4
- CPU 4
- Memory 5
- Disk Space 6
- Disk I/O and Network I/O 6

Applications 7
- Alfresco 7
- Solr 10
- Database Server 13
- Access Logs 13

Monitoring Applications 14
- Screenshots 17

UK Headquarters
Bridge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600

alfresco.com
San Mateo
Atlanta
London
Maidenhead
Sydney
Tokyo

Introduction

The objective of this article is to describe what to monitor in and around Alfresco in order to have a good understanding of how the applications are performing and to be aware of potential issues.

What needs to be monitored will change depending on business requirements and application usage, for example if we were to only use CMIS protocol to interact with the repository then there would be no need to monitor other protocols such as WebDav. This article is as generic as possible to cover most areas.

There is a huge number of software applications and tools that can be used to monitor Alfresco. In most cases you will need more than one application but it is advisable to keep the number of tools as low as possible. It is really up to the administrators (or company policy) to choose the tools that will meet their needs. We will define a set of requirements or features these tools should be able to perform in order to make the monitoring experience as simple and efficient as possible.

The data to be monitored should be collected by the monitoring tools on a regular basis from logs, application API, commands, etc. so that it can be presented to the administrators on demand.

Monitoring is not about collecting data but about being able to interpret the information provided.

Monitoring is not the same as profiling! Profiling is about measuring the frequency and duration of function calls in order to optimize an application or to analyze deep performance issues.

What do we need to monitor?

In order to have a good view of what the application is doing the following areas need to be monitored:

1. *Hardware resources*. This includes CPU, Memory, Disk space, Disk I/O and Network I/O.

2. *Applications*. This includes Alfresco, Solr, Web Server, Database server and any other applications used.
Note: depending on your installation some of these applications may not be used.

Let's take a look at these two groups in more detail.

UK Headquarters
Bridge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600

Hardware resources

CPU

CPU usage is one of the key resources to be monitored. We need to be able to tell how much CPU time user applications and the OS are using, not only for Alfresco but for all applications that are running under the Operating System. This is because the performance of Alfresco can be hindered by other applications taking too much CPU and we need to be able to tell which one may be causing problems.
Top command is the most widely used command to gather this information on Linux and Mac OS Operating Systems.

A lot of useful information can be obtained for the output of the top command (this is a Linux and Mac OS command, other Operating Systems will have different commands to obtain similar information).

[Top command output sample]

```
Tasks: 293 total,   1 running, 292 sleeping,   0 stopped,   0 zombie
%Cpu(s):  6.0 us,  2.5 sy,  0.0 ni, 87.2 id,  4.3 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:  16336792 total,  9076268 used,  7260524 free,    95564 buffers
KiB Swap: 16678908 total,    70764 used, 16608144 free.  4013320 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
11707 alfresco  20   0 3879972  1.512g 13828 S  38.2  0.2   0:00.19 java
  853 root      20   0       0      0      0 S   6.0  0.0   0:01.12 kworker/7:2
  848 root     -51   0       0      0      0 S   2.0  0.0   1:34.16 irq/49-iwlwifi
 1882 root      20   0  682432 134528 103508 S   2.0  0.8   3:55.76 Xorg
 2727 alfresco  20   0 1665592 276892  77612 S   2.0  1.7   3:30.07 compiz
 2737 alfresco  20   0  603312  45284  25624 S   2.0  0.3   0:32.40 unity-panel-ser
 3152 alfresco  20   0 1389668 257468 121596 S   2.0  1.6   3:50.70 chromium-browse
 9046 alfresco  20   0  670092  34880  27100 S   2.0  0.2   0:00.67 gnome-terminal
10823 root      20   0       0      0      0 S   2.0  0.0   0:00.35 kworker/3:2
11693 alfresco  20   0   29308   3308   2696 R   2.0  0.0   0:00.02 top
    1 root      20   0   29520   4140   2472 S   0.0  0.0   0:01.30 init
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.01 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S   0.0  0.0   0:25.42 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:01.20 rcuos/0
    9 root      20   0       0      0      0 S   0.0  0.0   0:05.17 rcuos/1
   10 root      20   0       0      0      0 S   0.0  0.0   0:05.54 rcuos/2
   ...
```

From this example we can see “user” applications (including Alfresco) are consuming 6% of the CPU time, “system” applications are consuming 2.5% and the CPUs are 87.2% idle.

We can also see Alfresco's java process is using 38.2% of CPU time. This is a percentage value over the total number of CPUs. In this case the server has 8 CPUs so the java process is using 38.2% over 800, that is 4.7% of the total CPU time.

This information is giving us a good overall view of how the server and applications are performing. CPU monitoring should be done on all servers including Database server and Transformation server.

alfresco.com
San Mateo
Atlanta
London
Maidenhead
Sydney
Tokyo

In general it is hard to tell when an application is using too much CPU. As a rule of thumb a short spike of CPU usage is not a major issue but if Alfresco is using high CPU for long periods of time then this could be a potential problem as it may be affecting user experience or other processes.

When experiencing long periods of high CPU try finding out what Alfresco is doing at that point in time, as sometime there could be a good reason for having high CPU i.e. doing a bulk import. A good way to know what Alfresco is doing is by capturing several thread dumps using the "jstack" command over a short period of time i.e. 6 thread dumps over a minute. With this information we can see what classes and methods are being executed and if a particular thread is running for long periods.

For example:

```
jstack <process id>
```

Repeat the same command a few times and save the output to be analysed. Analysis of jstack output is outside the scope of this document. This output should be analysed by an Alfresco support engineer if you are not familiar with java stack TRACEs.

Memory

Like CPU memory usage is one of the key resources to be monitored. In terms of memory we need to know how much physical and swap memory is available on the server and how much memory is used by the individual applications.

From the output of the top command shown above we can see the server has 16Gb of physical memory, there are 7.2Gb of free memory and there is no swapping at this point in time. Alfresco is also using 1.5Gb of memory.

Lack of physical memory is not the only possible cause for a memory error. Alfresco is a java application running under the JVM (Java Virtual Machine). When running Alfresco we use java options to indicate how much memory can be allocated to Alfresco and the JVM will do necessary memory management i.e. garbage collection, to keep the application within the specified limits, unless the demand for memory is too high and a "OutOfMemoryError" might occur.

The java option -Xmx indicates the maximum size of the memory to be allocated to the application.

It is advisable to check memory and garbage collection statistics for the java process on a regular basis. This can be done with the "jstat" command, for example:

```
jstat -gc -t <process id> 1000 60
```

This command will collect memory and garbage collection statistics every second for one minute. It is also highly advisable to generate a memory dump when Alfresco throws an out of memory error. This can be achieve by adding the following options to the java command:

UK Headquarters
Bridge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600

alfresco.com
San Mateo
Atlanta
London
Maidenhead
Sydney
Tokyo

```
-XX:+HeapDumpOnOutOfMemoryError  -XX:HeapDumpPath=<path to heap dump file>
```

To generate a memory dump manually (without waiting for an out of memory error) you can use the following command:

```
jmap -dump:format=b,file=/tmp/heapDump.out <java pid>
```

It is always good practise to take a few stack TRACEs before taking a heap dump, that way we can check what Alfresco is doing when the memory usage is high.

Make sure memory statistics are also collected from all servers, including the Database server.

Disk Space

As we know Alfresco content is stored on disk and therefore we need to make sure there is enough disk space to store content, transformations (thumbnails, previews, text, etc), multiple versions if using versioning, etc.

Solr/Lucene indexes are also stored on disk and they will grow as new content is added to the repository. Also during index merging we will require around twice the size of the indexes to perform the merge operation. We also need to make sure there is enough space for the index backups. If you are using Solr the default number of index backups to keep is three.

As disk space changes fairly quickly we need to monitor how much disk space is left and make additional disk space available when needed.

Disk I/O and Network I/O

Disk and Network I/O ratio should not change on a regular basis. Nevertheless as disk and network I/O speed is an important factor for the overall performance of the solution, we should take samples to check against when performance degradation happens.

You should use as fast as possible disks for the indexes as they have high I/O ratio. Solid State Drives (SSD) are recommended when possible.

UK Headquarters
Bridge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600

Applications

Alfresco

Alfresco is a large and complex application. It provides many subsystems (Activities Feed, Audit, Authentication...), File Servers (CIFS, FTP, NFS), Protocols (WebDAV, CMS, SharePoint...) and it works together with other applications/tools (Solr, Transformation server, LibreOffice...) Therefore monitoring every aspect of Alfresco is not an easy task.

Out of the box when running Alfresco we don't get statistics for all these different areas, so we will start with the most common/useful aspects that can be monitored.

1- Alfresco logs. Alfresco makes use of log4j to produce application logs in a consistent format. The specific format is declared using a log4j appender i.e:

```
###### File appender definition #######
log4j.appender.File=org.apache.log4j.DailyRollingFileAppender
log4j.appender.File.File=alfresco.log
log4j.appender.File.Append=true
log4j.appender.File.DatePattern='.'yyyy-MM-dd
log4j.appender.File.layout=org.apache.log4j.PatternLayout
log4j.appender.File.layout.ConversionPattern=%d{ABSOLUTE} %-5p [%c] %m%n
```

As an administrator these log files are the first and most important source of information as application errors will be reported here. Therefore we need to monitor the logs for any errors being reported, these are usually reported using the “ERROR” or “FATAL” logger level. For example here is an error when a database disconnection occurs:

```
15:58:08,976 ERROR [org.hibernate.transaction.JDBCTransaction] JDBC rollback failed
org.postgresql.util.PSQLException: This connection has been closed.
        at org.postgresql.jdbc2.AbstractJdbc2Connection.checkClosed(AbstractJdbc2Connection.java:714)
        at org.postgresql.jdbc2.AbstractJdbc2Connection.rollback(AbstractJdbc2Connection.java:731)
        at org.apache.commons.dbcp.DelegatingConnection.rollback(DelegatingConnection.java:368)
        at org.apache.commons.dbcp.DelegatingConnection.rollback(DelegatingConnection.java:368)
        at org.apache.commons.dbcp.PoolingDataSource$PoolGuardConnectionWrapper.rollback(PoolingDataSource.java:323)
        at org.hibernate.transaction.JDBCTransaction.rollbackAndResetAutoCommit(JDBCTransaction.java:183)
        at org.hibernate.transaction.JDBCTransaction.rollback(JDBCTransaction.java:162)
        at org.springframework.orm.hibernate3.HibernateTransactionManager.doRollback(HibernateTransactionManager.java:676)
        at
org.springframework.transaction.support.AbstractPlatformTransactionManager.processRollback(AbstractPlatformTransactionManager.java:845)
        at org.springframework.transaction.support.AbstractPlatformTransactionManager.rollback(AbstractPlatformTransactionManager.java:822)
        at org.springframework.transaction.support.TransactionTemplate.rollbackOnException(TransactionTemplate.java:161)
        at org.springframework.transaction.support.TransactionTemplate.execute(TransactionTemplate.java:134)
        at org.activiti.spring.SpringTransactionInterceptor.execute(SpringTransactionInterceptor.java:40)
        at org.activiti.engine.impl.interceptor.LogInterceptor.execute(LogInterceptor.java:33)
        at org.activiti.engine.impl.jobexecutor.JobAcquisitionThread.run(JobAcquisitionThread.java:63)
```

We can see here that after the logger level we log the class reporting the error. If a stack TRACE has been thrown it will also be logged. This information is key to understand where the problem is coming from.

If we want to monitor a specific area of the application i.e. permissions checking, we can enable debugging for the classes that report information about permissions. For example by adding the following entries to log4.properties file.

alfresco.com
San Mateo
Atlanta
London
Maidenhead
Sydney
Tokyo

```
log4j.logger.net.sf.acegisecurity.intercept.AbstractSecurityInterceptor=debug
log4j.logger.org.alfresco.repo.security.permissions.impl.acegi.ACLEntryVoter=debug
```

We can get debug information in the log file such as:

```
2013-06-17 12:55:01,875  DEBUG [acegisecurity.intercept.AbstractSecurityInterceptor] [http-8080-5]
Authenticated: net.sf.acegisecurity.providers.UsernamePasswordAuthenticationToken@7c10616a: Username:
net.sf.acegisecurity.providers.dao.User@176bb5c8: Username: alfresco; Password: [PROTECTED]; Enabled: true;
AccountNonExpired: true; credentialsNonExpired: true; AccountNonLocked: true; Granted Authorities:
ROLE_AUTHENTICATED; Password: [PROTECTED]; Authenticated: true; Details:
net.sf.acegisecurity.providers.dao.User@176bb5c8: Username: alfresco; Password: [PROTECTED]; Enabled: true;
AccountNonExpired: true; credentialsNonExpired: true; AccountNonLocked: true; Granted Authorities:
ROLE_AUTHENTICATED; Granted Authorities: ROLE_AUTHENTICATED
 2013-06-17 12:55:01,875  DEBUG [impl.acegi.ACLEntryVoter] [http-8080-5] Method: public abstract void
org.alfresco.service.cmr.repository.NodeService.deleteNode(org.alfresco.service.cmr.repository.NodeRef)
throws org.alfresco.service.cmr.repository.InvalidNodeRefException
 2013-06-17 12:55:01,876  DEBUG [impl.acegi.ACLEntryVoter] [http-8080-5]         Permission test on node
/{http://www.alfresco.org/model/application/1.0}company_home/{http://www.alfresco.org/model/site/1.0}sites/{
http://www.alfresco.org/model/content/1.0}privatesite/{http://www.alfresco.org/model/content/1.0}documentLib
rary/{http://www.alfresco.org/model/content/1.0}_x0033_1st-May.txt
 2013-06-17 12:55:01,876  DEBUG [impl.acegi.ACLEntryVoter] [http-8080-5]                 Node ref is not
null
 2013-06-17 12:55:01,886  DEBUG [impl.acegi.ACLEntryVoter] [http-8080-5]                 Permission is
denied
```

There are many classes that can be enabled for debugging for specific areas. Here are some examples that may help to get started.

```
# Authentication
log4j.logger.org.alfresco.repo.importer.ImporterJob=DEBUG
log4j.logger.org.alfresco.repo.importer.ExportSourceImporter=DEBUG
log4j.logger.org.alfresco.repo.security=DEBUG
log4j.logger.org.alfresco.repo.security.authentication.ldap=DEBUG
log4j.logger.org.alfresco.repo.security.authentication.AuthenticationUtil=INFO
log4j.logger.org.alfresco.repo.security.permissions=INFO
log4j.logger.org.alfresco.repo.webdav.auth=DEBUG
log4j.logger.com.sun.security=DEBUG
log4j.logger.org.alfresco.web.app.servlet=DEBUG

# Permissions
log4j.logger.net.sf.acegisecurity.intercept.AbstractSecurityInterceptor=DEBUG
log4j.logger.org.alfresco.repo.security.permissions.impl.acegi.ACLEntryVoter=DEBUG

# Activities
log4j.logger.org.alfresco.repo.activities=DEBUG

# Rules
log4j.logger.org.alfresco.repo.rule=DEBUG
log4j.logger.org.alfresco.repo.OnPropertyUpdateRuleTrigger=INFO

# Webscripts/javascript
log4j.logger.org.springframework.extensions.webscripts.AbstractWebScript=DEBUG
log4j.logger.org.springframework.extensions.webscripts.AbstractRuntime=DEBUG
log4j.logger.org.springframework.extensions.webscripts.servlet.WebScriptServlet=DEBUG
log4j.logger.org.springframework.extensions.webscripts.ScriptLogger=DEBUG
log4j.logger.org.alfresco.repo.jscript.ScriptLogger=DEBUG

# Synchronization
log4j.logger.org.alfresco.repo.security.sync=DEBUG

# CIFS
log4j.logger.org.smb.protocol=DEBUG
log4j.logger.org.alfresco.fileserver=DEBUG
log4j.logger.org.alfresco.filesys=DEBUG
```

UK Headquarters
idge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600

alfresco.com
San Mateo
Atlanta
London
Maidenhead
Sydney
Tokyo

```
# NFS
log4j.logger.org.alfresco.nfs.server=DEBUG
log4j.logger.org.alfresco.fileserver=DEBUG
log4j.logger.org.alfresco.nfs.protocol=DEBUG
log4j.logger.org.alfresco.filesys=DEBUG

# FTP
log4j.logger.org.alfresco.ftp.protocol=DEBUG
log4j.logger.org.alfresco.ftp.server=DEBUG
log4j.logger.org.alfresco.filesys=DEBUG

# Document transformations
log4j.logger.org.alfresco.repo.content.transform.TransformerDEBUG=DEBUG
log4j.logger.org.alfresco.repo.content.transform.ContentTransformerRegistry=DEBUG
log4j.logger.org.alfresco.repo.content.metadata.MetadataExtracterRegistry=DEBUG

# CMIS
log4j.logger.org.alfresco.opencmis.AlfrescoCmisServiceInterceptor=TRACE
log4j.logger.org.alfresco.cmis=TRACE
log4j.logger.org.alfresco.cmis.dictionary=TRACE
log4j.logger.org.apache.chemistry.opencmis=TRACE
log4j.logger.audit.cmischangelog.enabled=DEBUG

# Sharepoint protocol
log4j.logger.org.alfresco.module.vti=DEBUG

# SQL
log4j.logger.java.sql=DEBUG

# Lucene index tracking
log4j.logger.org.alfresco.repo.node.index.AbstractReindexComponent=DEBUG
log4j.logger.org.alfresco.repo.node.index.FullIndexRecoveryComponent=DEBUG
log4j.logger.org.alfresco.repo.node.index.IndexTransactionTracker=DEBUG

# Lucene searches
log4j.logger.org.alfresco.repo.jscript.ScriptLogger=DEBUG
log4j.logger.org.alfresco.repo.search.impl.lucene.LuceneQueryParser=DEBUG
log4j.logger.org.alfresco.repo.search.impl.lucene.analysis=DEBUG

# Solr index tracking
log4j.logger.org.alfresco.solr.tracker.CoreTracker=DEBUG

# SOLR searches
log4j.logger.org.alfresco.repo.search.impl.solr.SolrQueryHTTPClient=DEBUG
```

2 – Audit logging. Alfresco has the capability to audit most operations happening on the application.

This can help us to find information such as what happened on a particular period of time or who deleted a specific document. We can also use auditing to monitor the number of operations i.e. CREATE, UPDATE, DELETE, READ content, this will help us to understand how the application is being used over time.

The basic audit logging can be configured by adding the following entries to alfresco-global.properties (application restart needed):

```
# 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
```

UK Headquarters
idge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600

alfresco.com
San Mateo
Atlanta
London
Maidenhead
Sydney
Tokyo

Audit entries are recorded in the database, not the logs. We can make use of the REST calls to get information out of the system. For example we can execute the following query in the browser to get 10 audit entries starting from audit Id 10000

```
http://localhost:8080/alfresco/service/api/audit/query/alfresco-access?fromId=10000&verbose=true&limit=10
```

For further information on how to configure or use auditing please read the Alfresco documentation.

3- Transformations. Transformations are an important operation in Alfresco as we transform documents to generate thumbnails, previews, metadata extraction and more importantly we convert them to text to index their content.

Therefore it is important to monitor transformations and their response times. This can be done by adding debugging the following class in Alfresco (application restart needed).

```
log4j.logger.org.alfresco.repo.content.transform.TransformerLog=debug
```

Then in alfresco.log file we can see the transformation entries being recorded including the time it took to perform the transformation.

```
DEBUG [org.alfresco.repo.content.transform.TransformerLog] 3973 pdf  png  INFO Alfresco_Monitoring (4).pdf 665.1 KB 577 ms ImageMagick<<Proxy>>
```

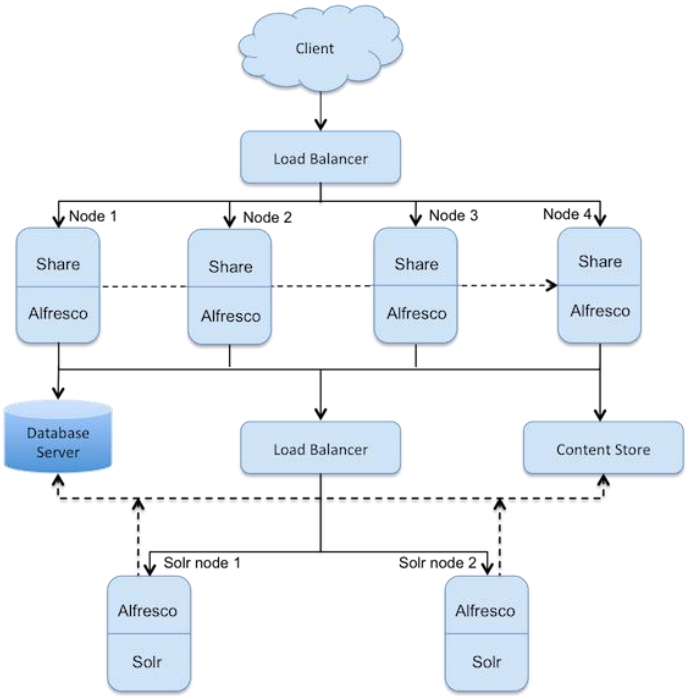
Solr

Solr and Lucene are search engines used for indexing content, metadata, ACLs, etc. We can then use query languages to search and retrieve this data. We are going to concentrate on Solr as this is the preferred search engine for Alfresco deployments.

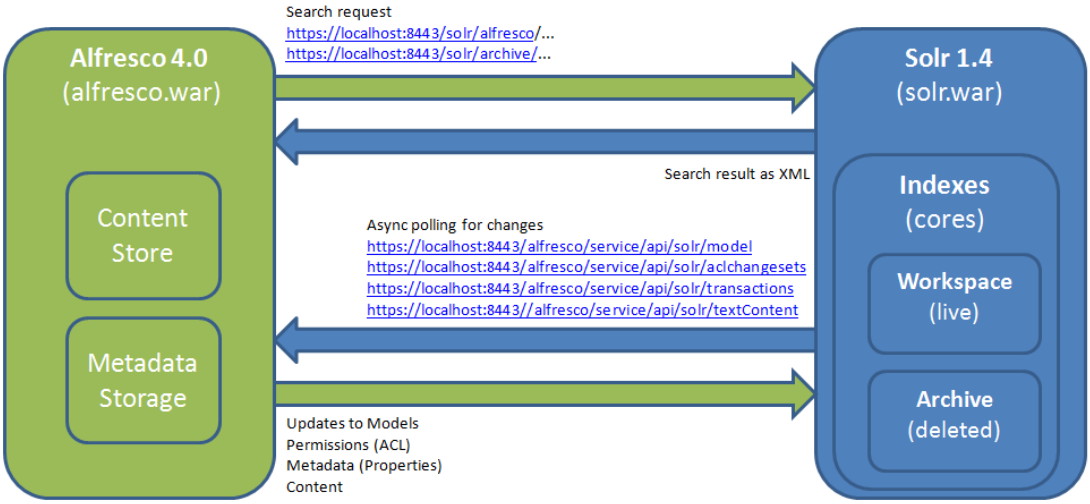
On a production system it is highly recommended to run Alfresco and Solr co-located on the same server. In this case Alfresco is only used for serving tracking requests coming from Solr. This Alfresco node should not be part of the Alfresco cluster and business users should not be connecting to this Alfresco. The following diagram is a good example of a production architecture. At the bottom of the diagram we can see the server running Alfresco and Solr together.

UK Headquarters
Bridge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600



To get an idea of how tracking works we can take a look at the following diagram:



In Alfresco we can monitor both index tracking and the searches requests.

1- Indexing. On the Solr side add the following entry to alf_data/solr/log4j-solr.properties

```
log4j.logger.org.alfresco.solr.tracker.CoreTracker=DEBUG
```

Then issue the following URL request to reload the log4j file.

```
http://localhost:8080/solr/admin/cores?action=LOG4J
```

In solr.log file we should see nodes and transactions being processed by Solr i.e.

```
DEBUG [solr.tracker.CoreTracker] [SolrTrackerScheduler_Worker-1] NodeInfo [id=2857, txnId=593, status=UPDATED]
DEBUG [solr.tracker.CoreTracker] [SolrTrackerScheduler_Worker-1] NodeInfo [id=2861, txnId=593, status=UPDATED]
```

2- Searches. To monitor searches and solr response times we can add the following debug to Alfresco's log4j.properties file (application restart needed).

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

In alfresco.log we can see the queries and 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","defa
ultFTSFieldOperator":"OR","defaultFTSOperator":"OR"}

DEBUG [impl.solr.SolrQueryHTTPClient] [http-apr-8080-exec-6] Got: 1 in 21 ms
```

There are two Solr reports that can be used to monitor the state of the indexes:

```
https://localhost:8443/solr/admin/cores?action=SUMMARY
https://localhost:8443/solr/admin/cores?action=REPORT
```

These report pages provide a large amount of information that will help to understand if the Solr indexes are in sync with Alfresco. For detailed explanation of each field in the report please refer to the Alfresco documentation or the following wiki page:

```
https://wiki.alfresco.com/wiki/Alfresco_And_SOLR#Checking_the_status_of_the_Solr_index
```

3- Solr uses caches to improve query performance. We can monitor Solr caches effectiveness by monitoring cache values shown in the Solr reports:

```
https://localhost:8443/solr/alfresco/admin/stats.jsp#cache
https://localhost:8443/solr/archive/admin/stats.jsp#cache
```

Each set of cache statistics has a number of different metrics. To determine the effectiveness of a cache, the most interesting figures are:

The cumulative hit ratio (cumulative_hitratio) - The percentage of queries that were satisfied by the cache (a number between 0 and 1, where 1 is ideal).

The cumulative number of inserts (cumulative_inserts) - The number of entries added to the cache over its lifetime.

The cumulative number of evictions (cumulative_evictions) - The number of entries removed from the cache over its lifetime.

The ultimate measure of a cache's performance is its hit ratio.

alfresco.com
San Mateo
Atlanta
London
Maidenhead
Sydney
Tokyo

If you see a high number of evictions relative to inserts, try increasing the size of that cache and monitor the effect on its hit ratio. It might be that entries are being evicted too quickly for your levels of search activity.

If a cache has a high hit ratio but very few evictions, it might be too large. Try reducing the cache size and see if there's any corresponding change in the hit ratio.

Don't be discouraged if your hit ratio remains low for certain caches. If your queries are generally non-repetitive then no amount of cache sizing is going to get that number up, and you might as well opt for a small cache size.

Database Server

Alfresco can use different database servers to store information such as Oracle, PostgreSQL, MySQL, MS SQL, etc.

It is recommended to run the Database Server on a separate node and not to use the same Database Server for other applications as this may have a performance impact on Alfresco.

The main things to monitor on a Database Server are:

1- CPU and Memory usage

2- Number of concurrent database connections

3- Slow Queries and database locking

4- Database Server logs

5- Network round-trip

Collecting and monitoring this information will help us to understand if the Database Server is performing well. Tuning the Database Server should be done by a qualified database administrator. It is important to make sure database statistics are refreshed on a regular basis as this may have a high impact on response times.

Access Logs

Access logs can be used to monitor HTTP requests made to the application and their response time. Response times from access logs provide a good guidance of application overall performance.

Web Servers and Proxy Servers such as Apache, Nginx, Tomcat, etc can generate access logs. Please refer to the product documentation to find out how to set up these logs.

For example to enable access logs in Tomcat web server we can add the following configuration to the `conf/server.xml` file (application restart needed):

```
<!-- Access logs -->
```

UK Headquarters
idge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600

```
<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" %T "%I"'
resolveHosts="false"/>
```

The extract below is from Tomcat access logs (under tomcat/logs directory) showing information such as the host making the request, the type of request, the URL and parameters in the request, the response type, the response time, etc.

```
127.0.0.1 - - [19/Nov/2014:15:54:02 +0000] "GET
/share/proxy/alfresco/api/node/workspace/SpacesStore/150398b3-7f82-4cf6-af63-
c450ef6c5eb8/content/thumbnails/doclib?c=queue&ph=true&lastModified=2011-02-24T16:16:37.300Z HTTP/1.1" 200
4774 "http://localhost:8080/share/page/search?t=*a&s=&a=true&r=false" "Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/38.0.2125.111 Chrome/38.0.2125.111 Safari/537.36"
0.028 "http-apr-8080-exec-6"
```

Monitoring Applications

There are a large number of monitoring applications both commercial and OpenSource. From a monitoring perspective these are the set of features they should support:

- Provide “near” real time updates
- Monitoring of large systems (multiple servers and applications)
- Reporting capabilities
- Global and Custom search mechanism
- Configurable and extensible

Some of the most popular tools are:

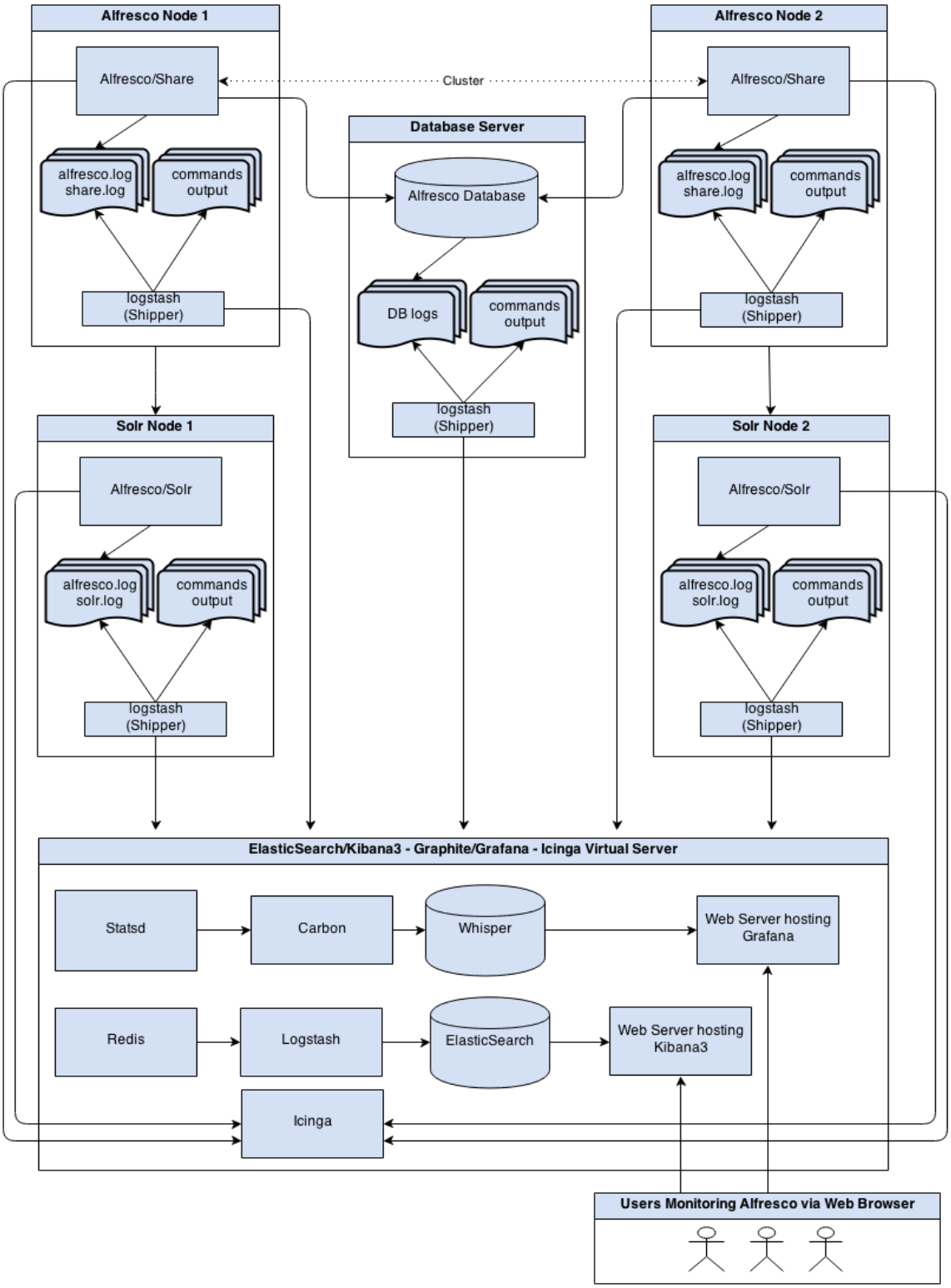
Commercial	OpenSource
AppDynamics	ElasticSearch, Logstash, Kibana
NewRelic	Nagios/Icinga
Splunk	JavaMelody
	Jconsole/JvisualVM, Jstack, Jstat

In this section we are going to look at some of the OpenSource applications and how together they can provide a solid and feature rich monitoring solution.

As a proof of concept we have used:

- Logstash, ElasticSearch and Kibana3 to monitor log files and output from custom commands
- Logstash, Graphite and Grafana to collect and display “numeric” data
- Icinga for sending alerts

The following diagram shows an example of an Alfresco solution consisting of a two Alfresco clustered nodes and two Solr nodes running the monitoring applications/tools.



UK Headquarters
Bridge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600

alfresco.com
San Mateo
Atlanta
London
Maidenhead
Sydney
Tokyo

In the diagram we can see we have added a logstash “agent” to each of the Aflresco nodes and to the Database Server.

We also have created a Virtual Machine to host the server side of the monitor tools. We don't want to run these alongside Alfresco as we don't want to take the resource away from Alfresco.

The main components in the monitoring tools are:

- Logstash (http://www.elasticsearch.org/overview/logstash/) is a tool for managing events and logs. You can use it to collect logs, parse them, and store them for later use. In our platform we use logstash to “tail” all logs files such as alfresco.log, share.log, solr.log, database logs, http logs, etc. The data from these logs is pushed by logstash to the monitoring server (towards the Virtual Machine).

Logstash is also used to execute commands to collect information that it is not stored in the logs such as CPU and Memory usage, statistics provided via JMX interface, etc. Again this data is pushed directly by logstash to the monitoring server.

- Elasticsearch (http://www.elasticsearch.org/overview/elasticsearch/) is a flexible and powerful open source, distributed, real-time search and analytics engine. Elasticsearch is used to store “text” data that can be searched for at a later time.

- Kibana (http://www.elasticsearch.org/overview/kibana/) is Elasticsearch’s data visualization engine, allowing you to natively interact with all your data in Elasticsearch via custom dashboards. Kibana’s dynamic dashboard panels are savable, shareable and exportable, displaying changes to queries into Elasticsearch in real-time.

- Graphite (http://graphite.wikidot.com/) is a highly scalable real-time graphing system. As a user, you write an application that collects numeric time-series data that you are interested in graphing, and send it to Graphite's processing backend,carbon, which stores the data in Graphite's specialized database. The data can then be visualized through graphite's web interfaces.

- Grafana (http://grafana.org/) is An open source, feature rich metrics dashboard and graph editor for Graphite.

- Icinga (https://www.icinga.org/) Icinga is a monitoring system checking hosts and services you specify and notifying you when things go wrong and when they recover. The systems to be monitored can be nearly anything connected to a network.

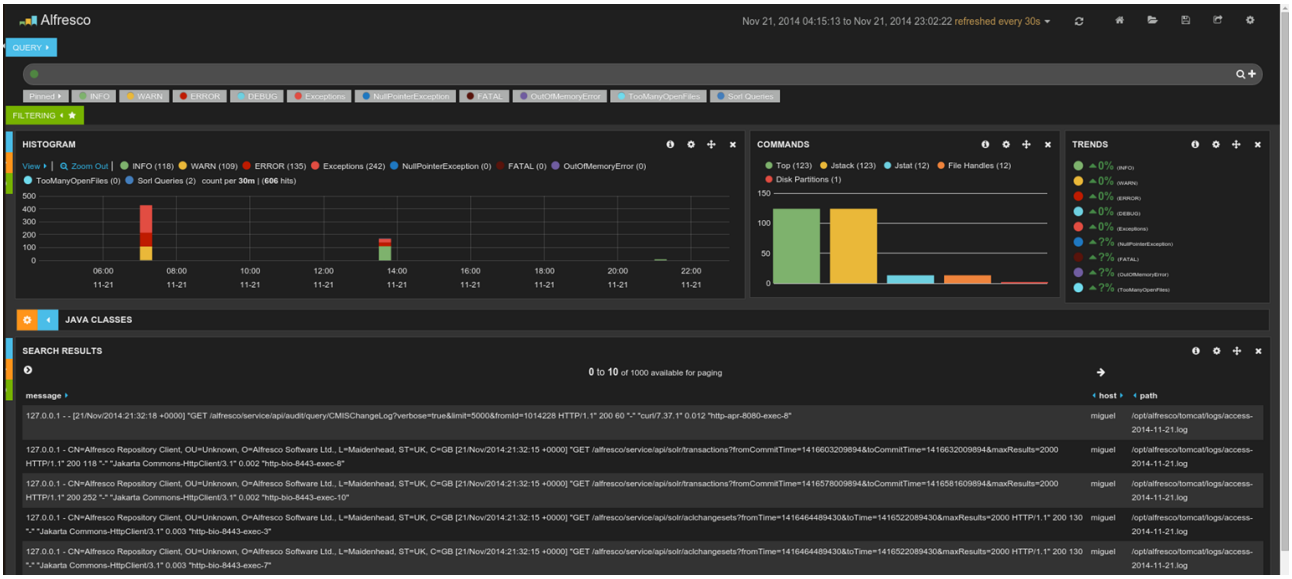
UK Headquarters
Bridge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600

alfresco.com
San Mateo
Atlanta
London
Maidenhead
Sydney
Tokyo

Screenshots

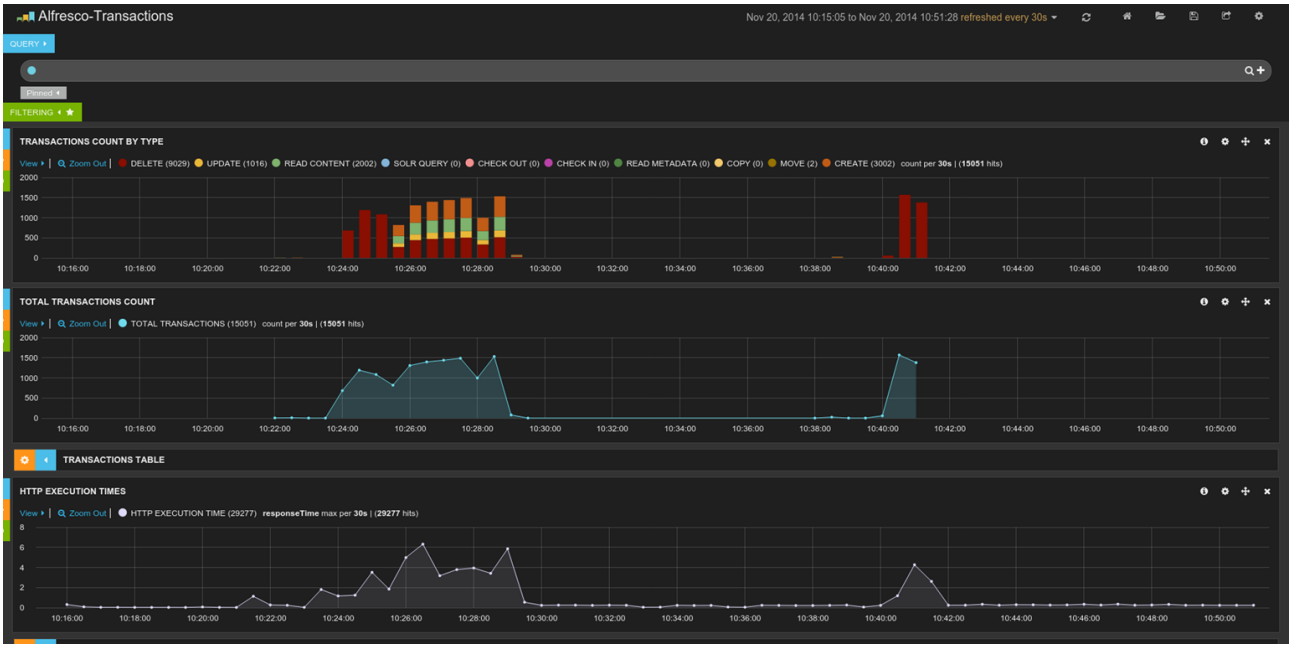
Here we can see several Kibana screenshots.



The screenshot above if of Kibana interface showing the data from Alfresco logs and the output from custom commands executed on the nodes such as jstak. On the to section we have a search bar to perform global searches for data across all nodes and all log files. We can navigate back and forth to show data corresponding to a specific point in time. We can also show data from all nodes or specific nodes using the filtering features in Kibana. The interface is very flexible and all configuration is done via the Web browser.

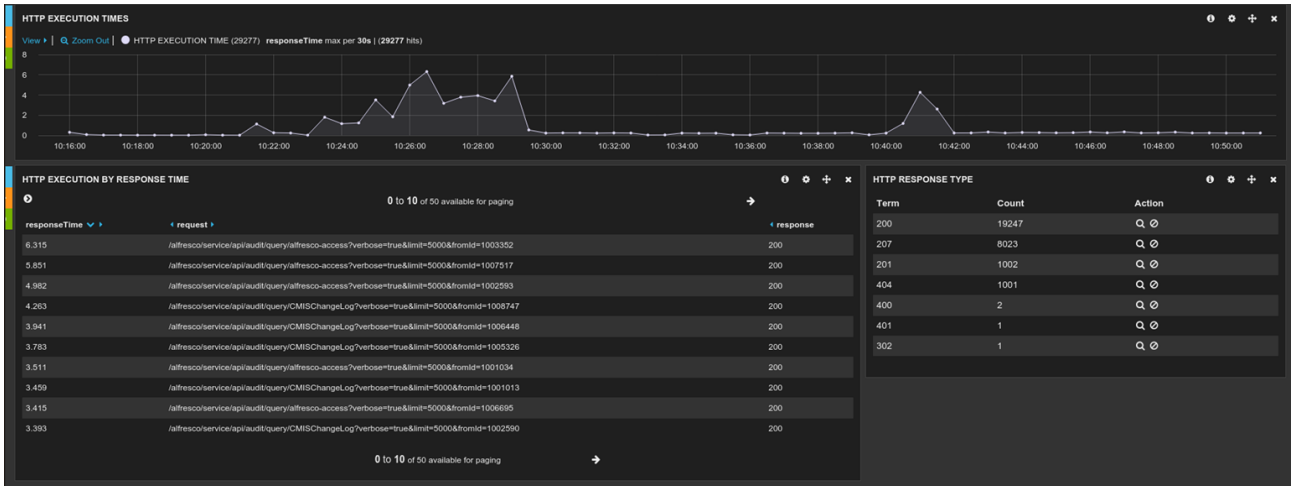
UK Headquarters
Bridge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600



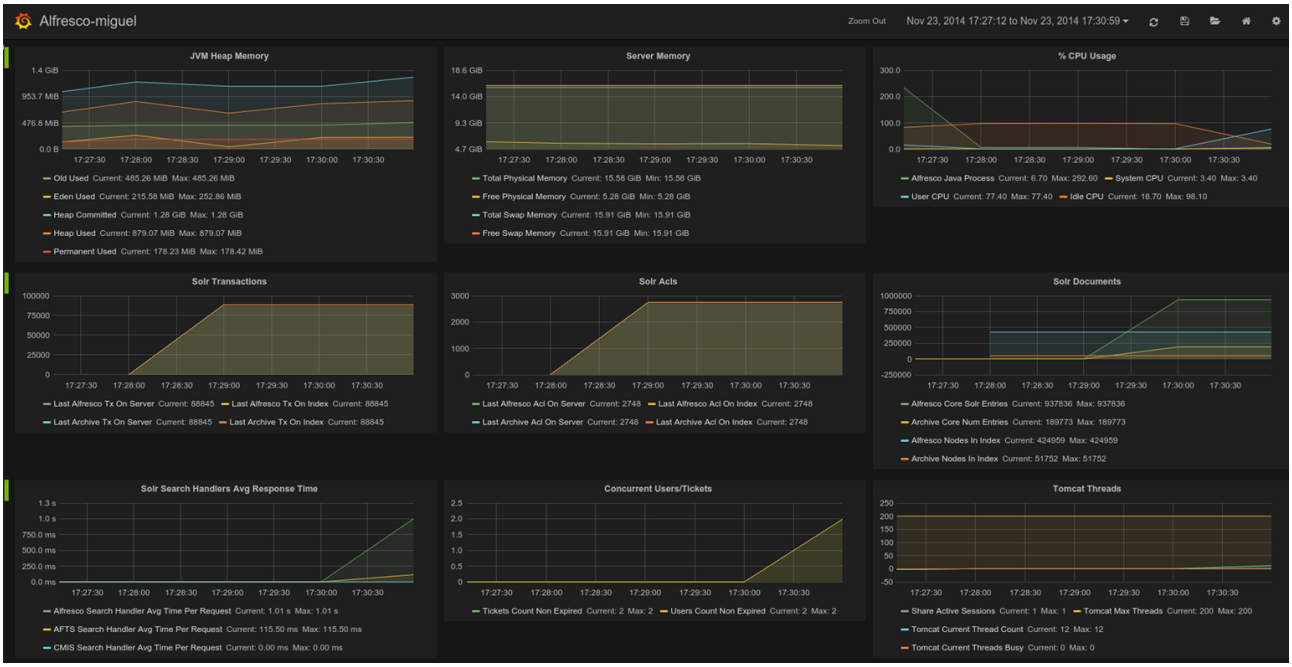
The screenshot above is also from Kibana. In this case we are using data from Alfresco audit tables, HTTP requests from Tomcat access logs, slow queries from Postgres database, Searches from Solr, Transformations requests from Alfresco logs, etc

We can display this data in a way that will help us to understand what the nodes are doing. We can display data for the whole solution or individual nodes.



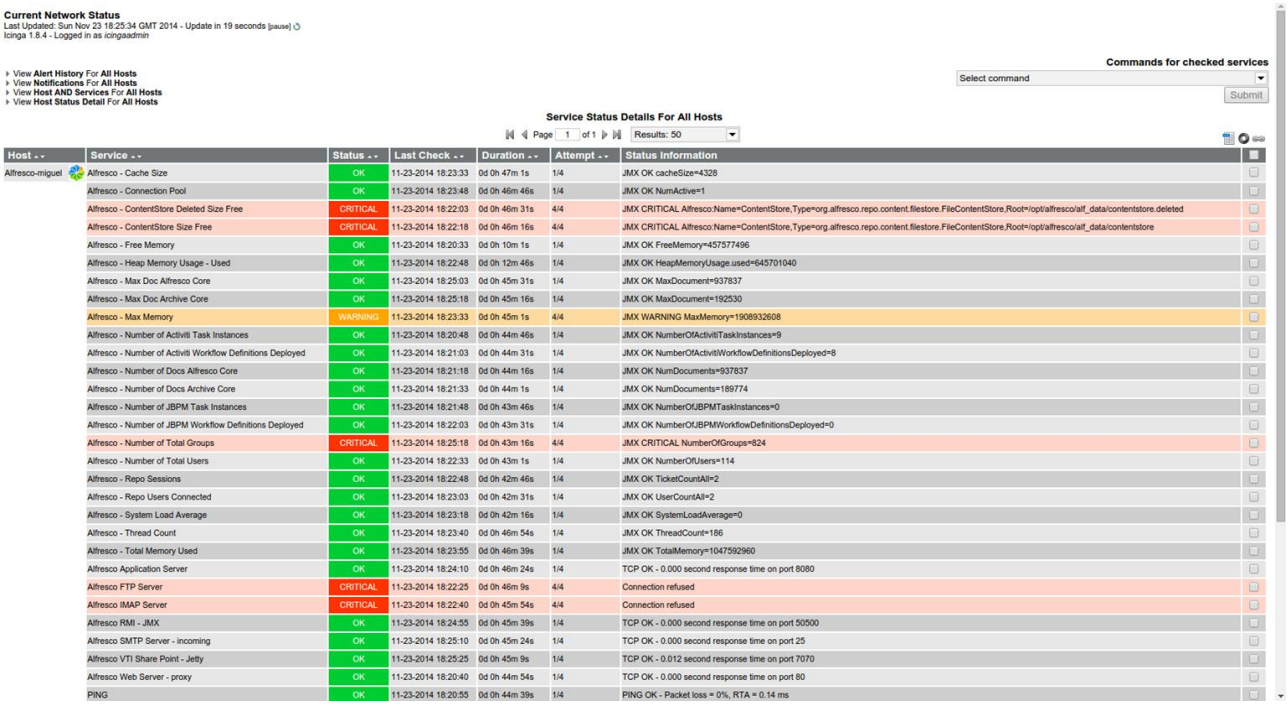
We can also display additional data in tables to show us slow performance from these graphs i.e. by sorting data by slow response time or group requests/responses by type, user, etc.

Again this gives us a good insight into performance issues and activities that are happening on the systems.



The graph above show the Grafana dashboard with some of the dashlets for metrics such as CPU, Memory, Sor document, concurrent users, Tomcat threads, etc.

Other data that can be displayed by Grafana is Database connections, Disk and Network IO, File Handles usage, Solr caches, etc.



Host	Service	Status	Last Check	Duration	Attempt	Status Information
Alfresco-miguel	Alfresco - Cache Size	OK	11-23-2014 18:23:33	0d 0h 47m 1s	1/4	JMX OK cacheSize=4328
	Alfresco - Connection Pool	OK	11-23-2014 18:23:48	0d 0h 46m 46s	1/4	JMX OK NumActive=1
	Alfresco - ContentStore Deleted Size Free	CRITICAL	11-23-2014 18:22:03	0d 0h 46m 31s	4/4	JMX CRITICAL Alfresco:Name=ContentStore,Type=org.alfresco.repo.content.filestore.FileContentStore,Root=/opt/alfresco/alf_data/contentstore.deleted
	Alfresco - ContentStore Size Free	CRITICAL	11-23-2014 18:22:18	0d 0h 46m 16s	4/4	JMX CRITICAL Alfresco:Name=ContentStore,Type=org.alfresco.repo.content.filestore.FileContentStore,Root=/opt/alfresco/alf_data/contentstore
	Alfresco - Free Memory	OK	11-23-2014 18:20:33	0d 0h 10m 1s	1/4	JMX OK FreeMemory=457577496
	Alfresco - Heap Memory Usage - Used	OK	11-23-2014 18:22:48	0d 0h 12m 46s	1/4	JMX OK HeapMemoryUsage.used=645701040
	Alfresco - Max Doc Alfresco Core	OK	11-23-2014 18:25:03	0d 0h 45m 31s	1/4	JMX OK MaxDocument=937837
	Alfresco - Max Doc Archive Core	OK	11-23-2014 18:25:18	0d 0h 45m 16s	1/4	JMX OK MaxDocument=192530
	Alfresco - Max Memory	WARNING	11-23-2014 18:23:33	0d 0h 45m 1s	4/4	JMX WARNING MaxMemory=1908932608
	Alfresco - Number of Activiti Task Instances	OK	11-23-2014 18:20:48	0d 0h 44m 46s	1/4	JMX OK NumberOfActivitiTaskInstances=9
	Alfresco - Number of Activiti Workflow Definitions Deployed	OK	11-23-2014 18:21:03	0d 0h 44m 31s	1/4	JMX OK NumberOfActivitiWorkflowDefinitionsDeployed=8
	Alfresco - Number of Docs Alfresco Core	OK	11-23-2014 18:21:18	0d 0h 44m 16s	1/4	JMX OK NumDocuments=937837
	Alfresco - Number of Docs Archive Core	OK	11-23-2014 18:21:33	0d 0h 44m 1s	1/4	JMX OK NumDocuments=189774
	Alfresco - Number of JBPM Task Instances	OK	11-23-2014 18:21:48	0d 0h 43m 46s	1/4	JMX OK NumberOfJBPMTaskInstances=0
	Alfresco - Number of JBPM Workflow Definitions Deployed	OK	11-23-2014 18:22:03	0d 0h 43m 31s	1/4	JMX OK NumberOfJBPMWorkflowDefinitionsDeployed=0
	Alfresco - Number of Total Groups	CRITICAL	11-23-2014 18:25:18	0d 0h 43m 16s	4/4	JMX CRITICAL NumberOfGroups=824
	Alfresco - Number of Total Users	OK	11-23-2014 18:22:33	0d 0h 43m 1s	1/4	JMX OK NumberOfUsers=114
	Alfresco - Repo Sessions	OK	11-23-2014 18:22:48	0d 0h 42m 46s	1/4	JMX OK TicketCountAll=2
	Alfresco - Repo Users Connected	OK	11-23-2014 18:23:03	0d 0h 42m 31s	1/4	JMX OK UserCountAll=2
	Alfresco - System Load Average	OK	11-23-2014 18:23:18	0d 0h 42m 16s	1/4	JMX OK SystemLoadAverage=0
	Alfresco - Thread Count	OK	11-23-2014 18:23:40	0d 0h 46m 54s	1/4	JMX OK ThreadCount=186
	Alfresco - Total Memory Used	OK	11-23-2014 18:23:55	0d 0h 46m 39s	1/4	JMX OK TotalMemory=1047592960
	Alfresco Application Server	OK	11-23-2014 18:24:10	0d 0h 46m 24s	1/4	TCP OK - 0.000 second response time on port 8080
	Alfresco FTP Server	CRITICAL	11-23-2014 18:22:25	0d 0h 46m 9s	4/4	Connection refused
	Alfresco IMAP Server	CRITICAL	11-23-2014 18:22:40	0d 0h 45m 54s	4/4	Connection refused
	Alfresco RMI - JMX	OK	11-23-2014 18:24:55	0d 0h 45m 39s	1/4	TCP OK - 0.000 second response time on port 50500
	Alfresco SMTP Server - incoming	OK	11-23-2014 18:25:10	0d 0h 45m 24s	1/4	TCP OK - 0.000 second response time on port 25
	Alfresco VTI Share Point - Jetty	OK	11-23-2014 18:25:25	0d 0h 45m 9s	1/4	TCP OK - 0.012 second response time on port 7070
	Alfresco Web Server - proxy	OK	11-23-2014 18:20:40	0d 0h 44m 54s	1/4	TCP OK - 0.000 second response time on port 80
	PING	OK	11-23-2014 18:20:55	0d 0h 44m 39s	1/4	PING OK - Packet loss = 0%, RTA = 0.14 ms

Finally the screenshot above is from Icinga application. We use an Alfresco plugin to monitor Alfresco and its services. We can configure thresholds for Warning and Critical

alfresco.com
San Mateo
Atlanta
London
Maidenhead
Sydney
Tokyo

events and which users need to be notified when these thresholds have been reached.

There is an OpenSource project in Github to create the Virtual Machine running the monitoring server and to provide all the client tools and configuration to get your own monitoring server running in a quick and easy way. Please visit https://github.com/miguel-rodriguez/alfresco-monitoring for further information.

UK Headquarters
Bridge Ave,The Place
Maidenhead, SL6 1AF
United Kingdom

+44 (0)1628 876 600