INSIGHTS INTO ECS DATA UTILIZATION USING OPEN SOURCE TOOLS

Analyzing ECS Access Logs with Elasticsearch, Logstash, and Kibana (ELK)

ABSTRACT

This white paper explains how administrators can better understand how data is being accessed on their ECS environment so they can better manage capacity, workload, and traffic. Analyzing access logs using ELK allows users visualize ECS data access.

February 2017
The information in this publication is provided "as is." EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

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

EMC®, EMC, the EMC logo, <insert other applicable product trademarks in alphabetical order—DELETE if not adding products> are registered trademarks or trademarks of EMC Corporation in the United States and other countries. All other trademarks used herein are the property of their respective owners. © Copyright 2016 EMC Corporation. All rights reserved. Published in the USA 03/17, white paper, H15961

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

EMC is now part of the Dell group of companies.
TABLE OF CONTENTS

EXECUTIVE SUMMARY ...........................................................................................................5

PRE REQUISITES .....................................................................................................................5
  Software ............................................................................................................................................ 5
  Network ............................................................................................................................................. 5
  Load balancer .................................................................................................................................... 5
  Docker Host ...................................................................................................................................... 5

ECS CONFIGURATION .............................................................................................................5

GATHERING ACCESS LOGS ...................................................................................................6

ELK CONFIGURATION .............................................................................................................6
  Elasticsearch ..................................................................................................................................... 6
  Kibana ............................................................................................................................................... 7
  Logstash ............................................................................................................................................ 8
    Option 1 - Logstash indexing log files locally stored ................................................................. 8
    Configuring Logstash .................................................................................................................... 8
    Starting the logstash process ....................................................................................................... 9
    Option 2 - Logstash indexing log files gathered using the ecslogs tool .................................... 9
    Gathering logs .............................................................................................................................. 9
    Configuring logstash ................................................................................................................... 10
    Starting the logstash process .................................................................................................... 10
    Option 3 - Logstash indexing logs received through syslog and storing the data in elasticsearch ... 10
    Configuring logstash ................................................................................................................... 11
    Starting the logstash process .................................................................................................... 11
    Configure rsyslog on ECS to send the logs to logstash ............................................................ 11
    Option 4 - Logstash indexing logs received through syslog and storing the data in a S3 bucket .... 12
    Configuring logstash ................................................................................................................... 12
    Starting the logstash process .................................................................................................... 12
    Configure rsyslog on ECS to send the logs to logstash ............................................................ 13
    Option 5 - Logstash indexing logs from an S3 bucket ............................................................... 13
    Create a new bucket containing the logs for the period of time you want to analyze ............... 13
    Configuring logstash ................................................................................................................... 14
    Starting the logstash process .................................................................................................... 15
EXECUTIVE SUMMARY

Dell EMC’s Elastic Cloud Storage (ECS) provides exascale object storage scalability and supports a variety of workloads. Customers taking advantage of its versatile data services employ multiple workflows and data types to support their modern data center application infrastructure. In order to balance access of resources provided by ECS, analyzing resource access and utilization is critical.

This document illustrates how to utilize Elasticsearch, Logstash, and Kibana (ELK) to analyze ECS access logs in order to provide insight into data utilization in ECS. The combined toolset enhances the type of information that can be gathered about the functioning status of an ECS cluster. Datacenter and operational administrators can employ these log analysis tools in a very simple virtual environment, utilizing open source, web-based toolsets to drill down on protocol based object read and write patterns and trends. Utilizing the graphing capability of Kibana, customers see how the workloads are balanced across the individual nodes of an ECS cluster, discover the actual read/write ratios of user traffic, look at the success rate of data lookups, peak access times, and even determine the cache-hit success rate.

The ELK stack in this example is run within a Docker container utilizing some pre-configured elements available on GitHub. This example details an assortment of ECS log gathering options including manual, batch and streaming which demonstrates the flexibility of both ECS and this data analysis toolset.

PRE REQUISITES

SOFTWARE

ELK components can be downloaded on http://elastic.co. We have packaged them in Docker images in order to simplify deployment.

NETWORK

There are no special networking requirements to run this. However, if this ELK deployment is intended to be persistent and used repeatedly over time, a static IP address is recommended in order to provide a stable target.

LOAD BALANCER

Not required.

DOCKER HOST

We recommend a separate physical or virtual host capable of being used as a Docker virtual app platform. At least three Docker containers will be running on this host.

ECS CONFIGURATION

By default, prior to the 3.0 version, ECS doesn’t generate access logs.

To check if access logs are enabled, run the following command on any ECS node:

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

```

```

To enable access logs, run the following command on any ECS node:

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

```
You only need to execute this command on a single node of each Virtual Data Center (VDC).

GATHERING ACCESS LOGS
There are several options available to use Elasticsearch to ingest ECS access logs through Logstash:

- Copy the logs manually from ECS to the Logstash host
- Get all the logs (of one VDC) using the ecslogs tool
- Receive the logs in realtime from the rsyslog service running on all the ECS nodes then
  - Ingest them into Elasticsearch
  - Store them in an S3 bucket
- Get some logs from the S3 bucket, based on a period of time

The Logstash section will explain how to implement these options.

ELK CONFIGURATION

ELASTICSEARCH
The Elasticsearch application container must be running before the Kibana web application will function correctly. Simply start the corresponding Docker container by executing the following command.

docker run -d -p 9200:9200 -p 9300:9300 djannot/elasticsearch:5.1.1

Note that you'll probably need to change the max_map_count kernel value using the following command on the Linux Docker host, not the container.

sudo sysctl -w vm.max_map_count=262144

If you want to modify the container, here is the Dockerfile used to create it:

```bash
FROM java
RUN \
    cd /tmp && \
    wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.1.1.tar.gz --no-check-certificate && \
    tar xvzf elasticsearch-5.1.1.tar.gz && \
    rm -f elasticsearch-5.1.1.tar.gz && \
    mv /tmp/elasticsearch-5.1.1 /elasticsearch
RUN useradd elasticsearch
RUN /elasticsearch/bin/elasticsearch-plugin install x-pack
RUN chmod -R 777 /elasticsearch
USER elasticsearch
ADD run.sh /elasticsearch/run.sh
CMD ["/elasticsearch/run.sh"]
EXPOSE 9200
EXPOSE 9300
```

The run.sh file content is:
#!/bin/sh
echo network.host: 0.0.0.0 > /elasticsearch/config/elasticsearch.yml
echo script.inline: true >> /elasticsearch/config/elasticsearch.yml
echo script.stored: true >> /elasticsearch/config/elasticsearch.yml
/elasticsearch/bin/elasticsearch

The default credentials for Elasticsearch are elastic/changeme.

KIBANA
Kibana provides the frontend web service through which reports and charts are generated. Since it is a web server, it is tied to the IP address of your Docker host on which the Kibana container resides. You can simply start the corresponding Docker container by executing the command below:

docker run -d -e ES_HOST=xxx.xxx.xxx.xxx -e ES_PORT=9200 -p 5601:5601 djannot/kibana:5.1.1

Replace xxx.xxx.xxx.xxx with the IP address of the host where you started the Elasticsearch Docker container.

If you want to modify the container, here is the Dockerfile used to create it:

FROM java
RUN apt-get -y update
RUN apt-get install -y wget libfreetype6 fontconfig
RUN (cd /tmp && wget https://artifacts.elastic.co/downloads/kibana/kibana-5.1.1-linux-x86_64.tar.gz --no-check-certificate -O pkg.tar.gz && tar zxf pkg.tar.gz && cd kibana-* && mkdir /kibana && cp -rf ./* /kibana)
ADD run.sh /kibana/run.sh
RUN chmod +x /kibana/run.sh
RUN /kibana/bin/kibana-plugin install x-pack
EXPOSE 5601
CMD ["/kibana/run.sh"]

The run.sh file content is:

#!/bin/sh
echo server.host: 0.0.0.0 > /kibana/config/kibana.yml
echo elasticsearch.requestTimeout: 300000 >> /kibana/config/kibana.yml
echo elasticsearch.url: "http://${ES_HOST}:${ES_PORT}" >> /kibana/config/kibana.yml
/kibana/bin/kibana

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

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

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

The Logstash container is treated somewhat differently. It may be started and stopped often as your data ingest needs change. This container is sensitive to the present working directory 'PWD' environmental variable of your shell as you work on the Docker host. If you want to modify the Docker container used in this section, here is the Dockerfile used to create it:

```
FROM java
RUN apt-get update
RUN apt-get install -y wget openjdk-7-jre unzip git
RUN wget https://artifacts.elastic.co/downloads/logstash/logstash-5.1.1.zip -O /tmp/logstash.zip --no-check-certificate && \
    unzip /tmp/logstash.zip -d /opt && \
    mv /opt/logstash* /opt/logstash && \
    rm -rf /tmp/*
WORKDIR /opt/logstash
RUN git clone https://github.com/djannot/logstash-input-s3.git
RUN sed -i 's/gem "logstash-input-s3"/gem "logstash-input-s3", :path => "/opt/logstash/logstash-input-s3"/g' Gemfile
RUN bin/logstash-plugin install --no-verify
ADD run.sh /usr/local/bin/run.sh
RUN chmod +x /usr/local/bin/run.sh
WORKDIR /opt/logstash
CMD ["/usr/local/bin/run.sh"]
```

The run.sh file content is:

```
#!/bin/bash
cp /data/logstash.conf /opt/logstash.conf
grep -i "hosts => docker" /data/logstash.conf | sed 's/hosts => docker/hosts => "$ES_HOST:$ES_PORT"/g' | grep -v "!logstash.conf" > /tmp/logstash.conf
sed -i 's/hosts => docker/hosts => "$ES_HOST:$ES_PORT"/g' /opt/logstash/bin/logstash -f /opt/logstash.conf
```

OPTION 1 - LOGSTASH INDEXING LOG FILES LOCALLY STORED

Configuring Logstash

This option involves manually copying relevant access log files from ECS nodes to the Docker host where the Logstash container will be started. Using an SFTP tool, copy the log files to the current directory (or appropriate folder) on the Docker host AND create a file called `logstash.conf` in the same directory as the transferred access logs with the following content:

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

filter {
  grok {
```

```
match => { "message" =>
  
  
  
  
}

date {
  match => [ "timestamp" , "YYYY-MM-dd HH:mm:ss, SSS" ]
}

output {
  elasticsearch {
    hosts => docker
    user => "elastic"
    password => "changeme"
  }
}

Starting the logstash process

Be sure to change directory into the folder where the access logs and the logstash.conf files are stored. This will be your working directory. Now, you can simply start the corresponding Docker container by executing the command below from your working directory:

```
docker run -t -i -e ES_HOST=xxx.xxx.xxx.xxx -e ES_PORT=9200 -v `pwd`:/data djannot/logstash:5.1.1
```

Depending on your choice of shell, this command may also use the $PWD variable instead of the `pwd` to indicate the working directory.

Replace xxx.xxx.xxx.xxx with the IP address of the host where you started the Elasticsearch Docker container.

OPTION 2 - LOGSTASH INDEXING LOG FILES GATHERED USING THE ECSLOGS TOOL

The ecslogs tool is also available as a Docker container. The source code can be found here:

[https://github.com/djannot/ecslogs](https://github.com/djannot/ecslogs)

The ecslogs tool adds the IP address of the ECS node to the corresponding lines from the access logs it retrieves from dataheadsvc.log

This allows you to determine if the requests are evenly spread across the ECS nodes.

Gathering logs

Here is the syntax you need to use to gather the logs:

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

For example, if the password of the SSH admin account is ChangeMe and you want to get the access logs from the last 3 days and store them in a local /tmp/access.log file:
Replace xxx.xxx.xxx.xxx with the IP address of an ECS node. This gathers log data from all of the nodes in the ECS cluster and stores the resulting 'access.log' into the present working directory where the ecslogs container was started. This is an interactively attached container, so it will run only as long as the log gather process is running. Upon completion, the container exits.

Configuring logstash

In the working directory. created the logstash.conf with the following content. Then copy the access.log file gathered in the above step to this same directory.

```plaintext
input {
  file {
    path => "/data/*.log"
    start_position => "beginning"
    ignore_older => 0
  }
}
filter {
  grok {
    match => { "message" => 
    }
  }
  date {
    match => [ "timestamp" , "YYYY-MM-dd HH:mm:ss,SSS" ]
  }
}
output {
  elasticsearch {
    hosts => docker
    user => "elastic"
    password => "changeme"
  }
}
```

Starting the logstash process

Change directory to the location where your access.log and logstash.conf file reside, you can start the corresponding Docker container by executing the command below from your current working directory:

```plaintext
docker run -t -i -e ES_HOST=xxx.xxx.xxx.xxx -e ES_PORT=9200 -v `pwd`:data djannt/logstash:5.1.1
```

Replace xxx.xxx.xxx.xxx with the IP address of the host where you started the Elasticsearch Docker container.

OPTION 3 - LOGSTASH INDEXING LOGS RECEIVED THROUGH SYSLOG AND STORING THE DATA IN ELASTICSEARCH
Configuring logstash

This Logstash option utilizes the syslog utility that is native to most *nix environments including your Linux Docker host. When configured correctly, each node of the ECS cluster will send access log entries via rsyslog to the Docker host where the logstash container is running. The hosts native rsyslog service will receive these continual updates and append them to a log file which will then be ingested by logstash.

The configuration file is different from the other options. Begin by creating a file called `logstash.conf` with the following content:

```{bash}
input {
    tcp {
        port => 514
        type => syslog
    }
    udp {
        port => 514
        type => syslog
    }
}

filter {
    grok {
        }
    }
    date {
        match => [ "timestamp" , "YYYY-MM-dd HH:mm:ss,SSS" ]
    }
}

output {
    elasticsearch {
        hosts => docker
        user => "elastic"
        password => "changeme"
    }
}
```

Starting the logstash process

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

```{bash}

docker run -t -i -e ES_HOST=xxx.xxx.xxx.xxx -e ES_PORT=9200 -p 514:514 -p 514:514/udp -v `pwd`:/data djannot/logstash:5.1.1
```

Replace `xxx.xxx.xxx.xxx` with the IP address of the host where you started the Elasticsearch Docker container.

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

On each ECS node, add the content below at the end of the `/etc/rsyslog.conf` file.

```plaintext
$ModLoad imfile
$InputFileName /opt/emc/caspian/fabric/agent/services/object/main/log/dataheadsvc.log
$InputFileTag ecs:
$InputFileStateFile ecs
$InputFileSeverity info
$InputFileFacility local7
$InputRunFileMonitor
:msg, contains, "RequestLog.java" @xxx.xxx.xxx.xxx:514
```

Replace `xxx.xxx.xxx.xxx` with the IP address of the host where you started the Logstash Docker container. It is necessary to restart the rsyslog service on each ECS node. This can be accomplished on ECS 3.0 by using the following command:

```
admin@nile-1422 :~> sudo systemctl restart rsyslog
```

**OPTION 4 - LOGSTASH INDEXING LOGS RECEIVED THROUGH SYSLOG AND STORING THE DATA IN A S3 BUCKET**

**Configuring logstash**

Create a file called `logstash.conf` with the following content:

```plaintext
input {
    tcp {
        port => 514
        type => syslog
    }
    udp {
        port => 514
        type => syslog
    }
}

output {
    s3 {
        access_key_id => "ACCESS KEY"
        secret_access_key => "SECRET KEY"
        bucket => "BUCKET NAME"
        proxy_uri => "http://<ECS IP ADDRESS>:9020"
        use_ssl => false
        time_file => 1
    }
}
```

The `time_file` parameter corresponds to the number of minutes to wait before uploading the new data to the S3 bucket.

It's also recommended to specify the endpoint of the load balancer instead of one ECS node (for the `proxy_uri` option).

The S3 bucket must be created with metadata search enabled for the `CreateTime` system metadata in order to only index logs for a specific period of time later.

**Starting the logstash process**

Start the corresponding Docker container using the following command:
docker run -t -i -e ES_HOST=xxx.xxx.xxx.xxx -e ES_PORT=9200 -p 514:514 -p 514:514/udp -v "pwd":/data/djannot/logstash:5.1.1

Replace xxx.xxx.xxx.xxx with the IP address of the host where you started the Elasticsearch Docker container.

This procedure started a logstash container that is configured to receive inbound syslog activity from the ECS nodes. It stores those access logs back onto the ECS cluster in an S3 bucket. No analysis is performed at this stage.

Configure rsyslog on ECS to send the logs to logstash

On each ECS node, add the content below at the end of the /etc/rsyslog.conf file. Before doing so, it is a good idea to make a backup copy of the rsyslog.conf file by doing:

```bash
#> sudo cp rsyslog.conf rsyslog.conf.bak
```

Then the configuration file can be edited by:

```bash
#> sudo vi rsyslog.conf
```

$ModLoad imfile
$InputFileName /opt/emc/caspian/fabric/agent/services/object/main/log/dataheadsvc.log
$InputFileTag ecs:
$InputFileStateFile ecs
$InputFileSeverity info
$InputFileFacility local7
$InputRunFileMonitor :msg, contains, "RequestLog.java" @0xxx.xxx.xxx.xxx:514

Replace xxx.xxx.xxx.xxx with the IP address of the host where you started the Logstash Docker container.

Once edits are complete, the rsyslog service must be restarted to insure that the new config file has been read.

```
admin@nile-1422-d46985b-663:/etc> sudo service rsyslog restart
admin@nile-1422-d46985b-663:/etc> sudo service rsyslog status
rsyslog.service - System Logging Service
 Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled)
 Active: active (running) since Mon 2017-01-30 18:40:27 UTC; 10s ago
 Process: 80403 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCESS)
 Process: 77606 ExecStartPre=/usr/sbin/rsyslog-service-prepare (code=exited, status=0/SUCCESS)
 Main PID: 77619 (rsyslogd)
 CGroup: /system.slice/rsyslog.service
     `-77619 /usr/sbin/rsyslogd -n

```

OPTION 5 - LOGSTASH INDEXING LOGS FROM AN S3 BUCKET

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

Use the ecss3copy tool to create a new bucket containing logs for a specific period of time later. A second (target) S3 bucket must be created first, before the performing the next steps.

The ecss3copy tool is available as a Docker container. The source code can be found here:


The following example copies the logs created from the 24th and the 26th of November from the ecslogs1 bucket to the ecslogs2 bucket:

```bash
ecss3copy -e http://xxx.xxx.xxx.xxx:9020 -u ACCESSKEY -p SECRETKEY -s ecslogs1 -t ecslogs2 -q "CreateTime>2016-11-24T00:00:00Z AND CreateTime<2016-11-27T00:00:00Z"
```
Replace xxx.xxx.xxx.xxx with the IP address of one ECS node. In this example, the bucket with the source logs is called ELKlogs and the bucket that we are copying the subset selection to is called ELKlogs2. Here is the full command to run it in a Docker instance.

```
docker run -it djannot/ecss3copy ./ecss3copy -e http://10.246.150.192:9020 -u ecsuser-repl -p "nshZG7qbtXnVvYCdEUVG5GEx7NpnP2AxxO1+dha3H" -s ELKlogs -t ELKlogs2 -q "CreateTime>2017-01-19T00:00:00Z and CreateTime<2017-01-30T20:00:00Z"
```

NOTE: Ensure the logical operator ‘and’ in this command is lower case.

Output should look something like this.

```
2017/01/31 00:06:34 100 operations executed in 0.357303 s
2017/01/31 00:06:34 279.874261 operations per second
2017/01/31 00:06:34 100 operations succeeded
2017/01/31 00:06:34 0 operations failed
The container will exit when complete.
```

Configuring logstash

It might be a good idea to create a second working folder on the Docker host to contain this logstash.conf file as it pertains to a separate instance of the logstash container. You need to create a file called `logstash.conf` with the following content:

```yaml
input {
  s3 {
    access_key_id => "ACCESS KEY"
    secret_access_key => "SECRET KEY"
    bucket => "BUCKET NAME"
    endpoint => http://<ECS_IP_ADDRESS>:9020
  }
}
filter {
  grok {
  date {
    match => [ "timestamp", "YYYY-MM-dd'T'HH:mm:ss,SSS" ]
  }
}
output {
  elasticsearch {
    hosts => docker
    user => "elastic"
    password => "changeme"
  }
}
The `time_file` parameter corresponds to the number of minutes to wait before uploading the new data to the S3 bucket.

**Starting the logstash process**

Start the corresponding Docker container by executing the command below:

```
docker run -t -i -e ES_HOST=xxx.xxx.xxx.xxx -e ES_PORT=9200 -v `pwd`:/data djannot/logstash:5.1.1
```

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

There should now be four docker containers running as shown is this example:

```
elasticsearch:~ # docker ps
CONTAINER ID        IMAGE                         COMMAND                  STATUS              NAMES
125a22c9bba5        djannot/logstash:5.1.1        "/usr/local/bin/run.s"   Up 41 seconds ago      dreamy_wescoff
218340a516af        djannot/logstash:5.1.1        "/usr/local/bin/run.s"   Up 21 minutes ago      agitated_nobel
21 minutes          0.0.0.0:514->514/tcp, 0.0.0.0:514->514/udp       agitated_nobel
1262e62cbb7a        djannot/kibana:5.1.1          "/kibana/run.sh"         Up 6 hours ago         hopeful_yalow
262bcbff24758        djannot/elasticsearch:5.1.1  "/elasticsearch/run.s"   Up 6 hours ago         romantic_banach
```

There should now be four docker containers running as shown is this example:
KIBANA VIZUALISATIONS

ACCESSING KIBANA

If you connect again to [http://yyv.yyy.yyy.yyy:5601](http://yyv.yyy.yyy.yyy:5601), you should get something similar to what's displayed in the picture below because some data has now been indexed.

Select @timestamp in the Time-field name list and click on Create.

A wide variety of interesting visualizations can now be created using Kibana.

HTTP METHODS
DATA TRANSFERRED PER DAY

For this one, you need to create a **Scripted field** that sums the uploads and downloads.

```
doc['upload'].value > 0 || doc['download'].value > 0 ? doc['upload'].value + doc['download'].value : 0
```

**Timelion Visualizations**

Timelion is a time series composer for Kibana. It allows you to use 2 different Y-axis.

**TIMELION EXAMPLES**

- Number of response code 500 vs total number of requests

```
.es(q='response_code: 500', metric='count:*'), .es(q='*', metric='count:*').yaxis(2)
```
- Average GET response time according to the average GET download size

```
.es(q='method:GET', metric='avg:duration'), .es(q='method:GET', metric='avg:download').yaxis(2)
```

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

```
.es(q='method:PUT', metric='avg:duration'), .es(q='method:PUT', metric='avg:upload').yaxis(2)
```
- Average GET + PUT response times according to the average GET download + PUT upload sizes

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

- Maximum GET download size and PUT upload size

\[
\text{es(q='method:GET', metric='max:download'), es(q='method:PUT', metric='max:upload').yaxis(2)}
\]
According to this analysis, it is possible to analyze applications patterns. In this case, 500MB is the maximum part size in this application.

- Number of GET request for 500 MB objects:

  ```
  .es(q='method:GET AND download:524288000', metric='count:*')
  ```

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

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

You can save visualizations and create dashboards from them.

Here is a dashboard created from some of the visualizations described in the previous sections:
IMPORTING/EXPORTING VISUALIZATIONS AND DASHBOARDS

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