Introduction

This chapter will be about introduction to Elasticsearch-Logstash-Kibana (ELK) stack and our underlying technologies. We will begin by explaining some background on the ELK we use, ATLAS grid system and dCache billing log files and more, then move on to how to get Docker running on your system and finally how to get it set up to start working with. After reading this introcuntion and the following chapters, we expect that you will easily start running your own ELK instance on Docker.

Distributed grid systems and dCache storage system used by them

In a ATLAS distributed grid site, its transfers and job efficiencies are monitored by some central dashboard systems such as the Rucio and Big PanDA. Each grid site has not been checked only by the latest status in the dashboards, also by periodic checks such as nagios, availability tests (e.g. HammerCloud) and so on. Furthermore, many of these systems are dealt with by many experts, central shifters and the national computing facilities/groups with having several remote meetings/communications they can work with. So, one grid site can collaborate with different groups of people in different monitoring ways simultaneously within the same experiment, in our case, ATLAS. This allows us to distribute several types of monitoring and administration tasks that are possible in both centralized systems and hierarchical models [8]. As for a storage system of German ATLAS computing Tier1/Tier2 sites, dCache storage system developed by DESY is widely used. The dCache can use hierarchical storage management (e.g. over many RAID disks and tapes), provides mechanisms to automatically increase performance and balance loads, increase resilience and availability [9]. It also supplies advanced control systems and various grid protocols to manage data as well as data workflow in each site.

About Elasticsearch-Logstash-Kibana (ELK)

What is the ELK stack, and why should we care? ELK stack deliver actionable insights in real time for almost any type of structured and unstructured data source, at any scale. From centralizing infrastructure logs, traces, and metrics to deliver administrators. ELK is a combined system that parse the records, collect and display information of different log files or messages so that one can recall and display specific records [4]. In our practical use-case, final outcomes given by the Kibana dashbaord has been displayed by internal/external dCache data workflow and information recorded by its billing log files so far. Many organizations worldwide are using the ELK Stack for various mission-critical use cases in ATLAS and some large Tier1s as well.

Docker

Docker has become the de facto standard that developers and system administrators use for packaging, deploying, and running distributed applications. It provides tools for simplifying mission-critial operations and administrations by enabling developers to create templates called images that can be used to create lightweight virtual machines called containers, which include their applications and all of their applications' dependencies. These lightweight virtual machines can be promoted through testing and production environments where system administrators deploy and run them.

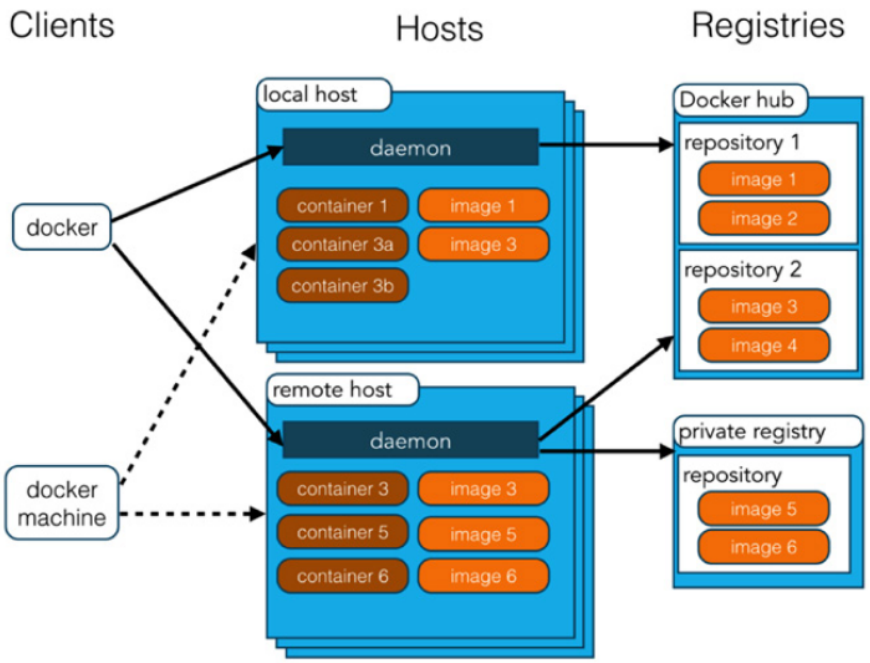


Figure 1: How docker works.

Similar to the popular version control software Git, Docker has a social aspect, in that developers and administrators are able to share their images via Docker Hub (Figure 1-1). Docker is an open-source solution that runs natively on Linux but also works on Windows and Mac using a lightweight Linux distribution and VirtualBox. Many tools have also grown up around Docker to make it easier to manage and orchestrate complex distributed applications such as grid monitoring tools and the ELK stack.

Docker compose and Gantry command

Docker compose is a tool for defining and running multi-container Docker applications. With docker-compose command, one uses a YAML file to configure the application's services. Then, with a single docker-compose command, the one create and start all the services from its configuration.

The docker-compose command works in all environments we need: production, staging, development, testing, as well as Continuous Integration (CI) workflows. We can learn more about each case in our cookbook chapter. In our settings, we simply implemented a wrapper command, called mad-gantry, which is a wrapper of docker-compose, making easier to generate docker template and docker instance template for different monitoring tools we have developed such HappyFace and now the ELK stack.

The concept of our 'Gantry' is following

- Simple design (pre-defined YAML files and its settings are given, and easily re-generated)
- A wrapper of docker-compose
- Easily distributed or separated by one config file
- Standardisable and adaptable for many sites, monitoring tools and different local settings
- Checking if services are running
- Syncronising the tools and configurations with the code in the Git repository

Settings

Our settings running with test data of GoeGrid Tier2 site are

- CPU: Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz x 40 Cores
- Memory: 65 GB
- HDD and SSD: 1 TB
- Operating system: Scientific Linux 7.5
- Docker: 1.12.6
- Docker Compose: 1.9.0
- Size of dCache billing log (10 years): 442 GB

Size of billing (GB)	Years	Index size	Elapsed time (hours)
72	1	156.2	13.5
129	2	296.8	27.3

Size of billing (GB)	Years	Index size	Elapsed time (hours)
365	4	512.6	131.9
430	8	X	Y
442	10	X	Y

Logstash

Logstash is a tool to collect, process, and forward events and log messages. Collection is accomplished via configurable input plugins including raw socket/packet communication, file tailing, and several message bus clients. There are many built-in patterns and plugins that are supported out-of-the-box by Logstash for filtering items such as words, numbers, and dates. If we cannot find the pattern we need, we can write own custom pattern. In our dCache billing log use-case so far, we put our pattern based on a standard dCache logstash template [4][5].

Logstash configuraiton for dCache billing log

The pattern and pipeline files are defined by 'customs/*/logstash/pattern' and 'customs/*/logstash/pipeline'.

- Some examples of dCache log pattern

The raw information about all dCache activities can be found in /var/lib/dcache/billing/YYYY/MM/billing-YYYY.MM.DD. A typical line looks like

```
05.31 22:35:16 [pool:pool-name:transfer] [000100000000000000001320,24675] \
myStore:STRING@osm 24675 474 true {GFtp-1.0 client-host-fqn 37592} {0:""}
```

The first bracket contains the pool name, the second the pnfs ID and the size of the file which is transferred. Then follows the storage class, the actual amount of bytes transferred, and the number of milliseconds the transfer took. The next entry is true if the transfer was a wrote data to the pool. The first braces contain the protocol, client FQN, and the client host data transfer listen port. The final bracket contains the return status and a possible error message. The billing info keeps such general accountings about the cell, pool, door and actions, such as removal, move and so on, for listing the details of their requests. The output of those requests often contains useful information for describing details of the transfers and solving problems. For instance, the

following patterns describe a general (classical) transfer request and a store request. Each accounting line should have appropriate fields such as datetime, PNFS ID, file size, transfered time, protocol, pool or door names, error and so on. The fields are already defined by 'customs/*/logstash/pattern' and its pattern file is called by 'customs/*/logstash/pipeline'.

```
TRANSFER_CLASSIC %{BILLING_TIME:billing_time} %{CELL_AND_TYPE} %{PNFSID_SIZE} %{PATH} \
%{SUNIT} %{TRANSFER_SIZE} %{TRANSFER_TIME} %{IS_WRITE} %{PROTOCOL} %{DOOR} %{ERROR}

STORE_CLASSIC %{BILLING_TIME:billing_time} %{CELL_AND_TYPE} %{PNFSID_TSIZE} %{PATH} \
%{SUNIT} %{TRANSFER_TIME} %{QUEUE_TIME} %{ERROR}
```

Elasticsearch

Elasticsearch is a real-time distributed search and analytics engine [4][6]. It allows one to explore data at a speed and at a scale never before possible. It is used for full-text search, structured search, analytics, and all in combination. Elasticsearch is an open-source search engine built on top of Apache Lucene, a fulltext search-engine library. However, Elasticsearch is much more than just Lucene and much more than just full-text search. It can also be described as follows:

- A distributed real-time document store where every field is indexed and searchable
- A distributed search engine with real-time analytics
- Capable of scaling to hundreds of servers and petabytes of structured and unstructured data

And it packages up all this functionality into a standalone server that applications can talk to via a simple RESTful API, using a web client from our favorite programming language, or even from the command line.

For instance, after starting our service up, our simplest example is directly sending a query to the Elasticsearch instance (port 9200 in this case) and get number of indices stored.

```
$ curl 'localhost:9200/_cat/indices?v'

health status index                    uuid                   pri rep docs.count
yellow open   dcache-billing-2017.10 9ocm5mfoQX255Z8Z4NWwrA   5   1   33336024
yellow open   dcache-billing-2017.09 bz1ZBIa4SmqetFGBpqruWA   5   1   17306766
yellow open   dcache-billing-2018.02 iFuFB03PR_SsNNpf6sXpxw   5   1   25118420
yellow open   dcache-billing-2018.01 HitUUihaRuGBnd47Ma98Bg   5   1   24001272
```

```
yellow open   dcache-billing-2017.05 rnU_rxhcTt6Y-D6AeWxy8A   5   1   23641828
yellow open   dcache-billing-2017.11 8s3hp4lBRzatjMzXDaNbUQ   5   1   49539540
yellow open   dcache-billing-2018.04 vRUN0b08T4iZrvyNpxOcgA   5   1   26639090
yellow open   dcache-billing-2018.12 GFhoMTqfSWKO80lp6iurfg   5   1   63447534
yellow open   dcache-billing-2017.07 cleYoAc-QM2DgcpfiJnj6Q   5   1   31308502
yellow open   dcache-billing-2018.11 QJxVUAqESSOGumiG5lk1cA   5   1   48383944
yellow open   dcache-billing-2018.07 gfCszQ_4RUulIjn42guNUw   5   1   16616834
green  open   .kibana                E-J_GSkdRj22BN7DQHMN_Q   1   0         27
yellow open   dcache-billing-2019.04 A9Ic3--tQjaML_Zg_daFXA   5   1   11335536
yellow open   dcache-billing-2018.09 MkD363YHRUqsIAZvTMCsCA   5   1   29330778
```

Kibana

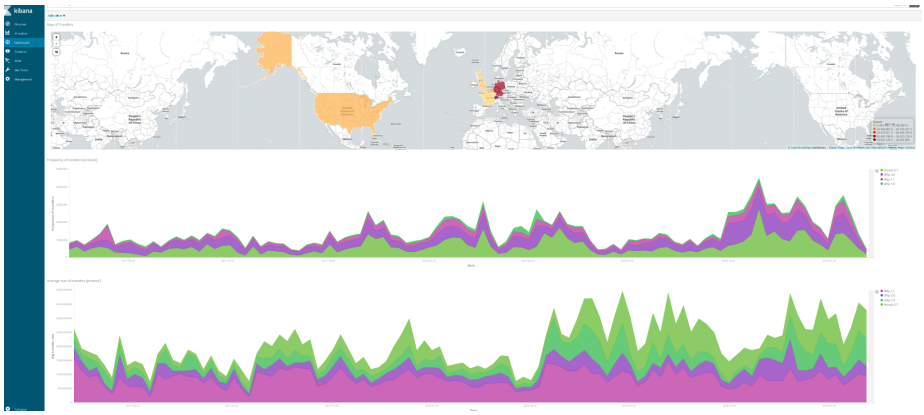


Figure 2: Main Dashboard

Kibana is an open source analytics and visualization platform designed to work with Elasticsearch [4][6]. We use Kibana to search, view, and interact with data stored in Elasticsearch indices. We can easily perform the data analysis and visualize our data in a variety of charts, tables, and maps. Kibana makes a large local site easy to understand large volumes of data, for instance, dCache billing log. Its simple, browser-based interface enables you to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time. As default, our 'docker-welt' command with a 'tier2_ELK' option provide an Kibana instance and a port mapping of 20261.

Grafana

Grafana is a multi-platform open source analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected

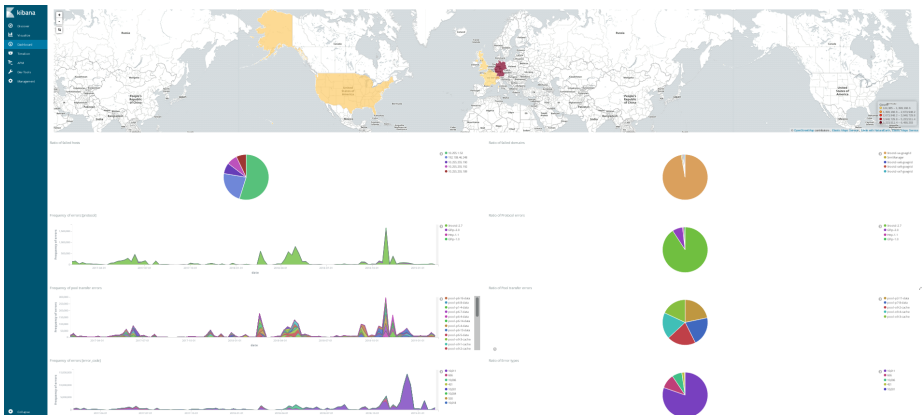


Figure 3: Main Error Dashboard

to supported data sources. It is expandable through a plug-in system. End users can create complex monitoring dashboards using interactive query builders. As a visualization tool, Grafana is a popular component in monitoring stacks often used in combination with time series databases such as InfluxDB, Prometheus and Graphite; monitoring platforms such as Sensu, Icinga, Zabbix, Netdata and PRTG; SIEMs such as Elasticsearch and Splunk; and other data sources. Our 'docker-welt' command with a 'tier2_ELK' option provide an Kibana instance and a port mapping of 20261. The data source is from Elasticsearch.

Figure 4: Main Grafana Dashboard 1

Figure 5: Main Grafana Dashboard 2

Cookbook

Running ELK stack in a ATLAS Tier2 site

- Cloning the main command and generating ELK templates

  $ git clone https://github.com/GenKawamura/docker-welt.git

  $ cd docker-welt

  $ ./docker-welt -C -r tier2_ELK

  $ ls workarea/tier2_ELK/GoeGridELK

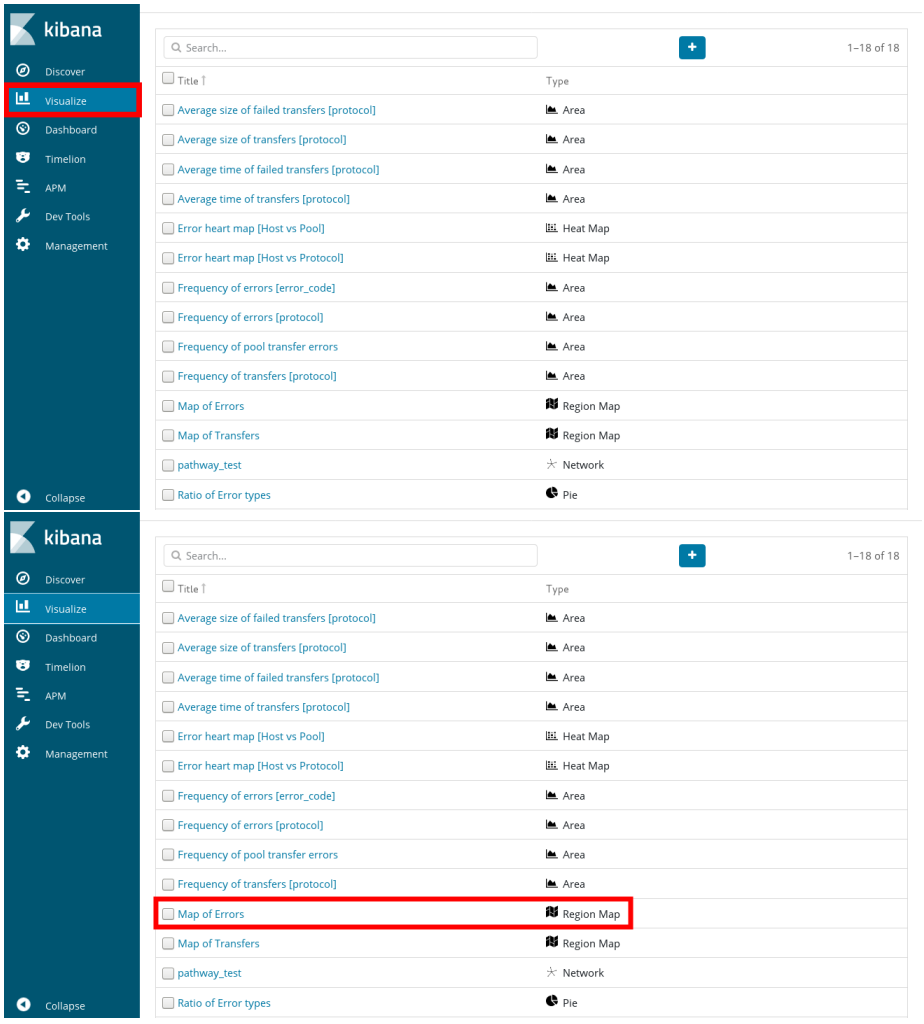
  arcce_log dcache_billing elasticsearch_index_data grafana htcondor_eventlog pbs_log

- Copy dCache/ARC/HTCondor/PBS logs into the billing directory (Note: indices will be kept in the 'elasticsearch_index_data' directory)

- Logstash instance send the log information to Elasticsearch instance. A port 20261 is used for Kibana and a port 20262 is used for Grafana.

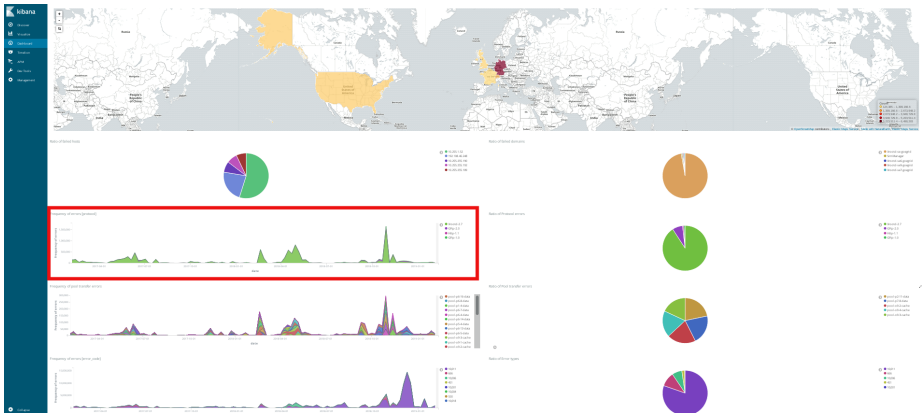
  $ docker ps

Manipulating Kibana Visualisation



The Kibana web interface contains under the menu point “Visualize” tab listing from all plots we created. The visualisation plots can also be redirected.

Manipulating Kibana Dashboard



Each visualisation plot is gathered and sorted by the main dashboard, so that one changes the visualisation plot it will affect the dashboard view.

Saving Kibana dashboard

In our case, we implemented our own darshboard (ID: WeXmuoICywmhE8FvCht). So, exporting the JSON output, and re-using it when the service is up again.

```
$ id=WeXmuoICywmhE8FvCht
$ export_url="http://localhost:20261/api/kibana/dashboards/export?dashboard=$id"
$ GET "$export_url" > export.json

$ import_url="http://localhost:20261/api/kibana/dashboards/import"
$ curl -u elastic:changeme -k -XPOST "$import_url" \
    -H 'Content-Type: application/json' -H "kbn-xsrf: true" -d @export.json
```

Command snippets for Elasticsearch

There are many client tools for Elasticsearch. These can most easily be communicated through RESTful web service in Elasticsearch engine.

- List indexes

  $ curl -XGET http://localhost:9200/_cat/indices?v

- Simple search in PBS log (show json structures)

  $ curl -XGET 'localhost:9200/pbs/_search?&pretty=true&size=1000'

- Simple search in dCache billing log

  $ curl -XGET 'localhost:9200/dcache-billing-2020.05/_search?pretty=true&size=1000'

- Filter if 'status' in PBS is 'E'

  $ curl -XGET 'localhost:9200/pbs/_search?q=status:E&pretty=true&size=1000'

- Filter if 'error_code' in dCache log is not '0'

  $ curl -XGET 'localhost:9200/dcache-billing-2020.05/*search?q=!(error*code:0)&pretty=true&size=1000'

- Show mapping structure (A definition of the PBS index)

  $ curl localhost:9200/pbs/_mapping?pretty

- Simple matches using pool_name in 'billing log'

  $ curl -XPOST 'localhost:9200/_search?pretty=true' -d '{"query": { "match_all": {} } }'

  $ curl -XPOST 'localhost:9200/_search?pretty=true' -d '{"query": { "match": { "pool_name": "pool-p1-1-data" } } }'

- Aggregations

  $ curl -XPOST 'localhost:9200/*search' -d '{"aggs": { "all_interests": { "terms": {"field": "size"} } } }' $ curl -XPOST 'localhost:9200/*search' -d '{"aggs": { "queries": { "terms": {"field": "size"} } } }'

  $ curl -XPOST 'localhost:9200/*search' -d '{"query": { "match": { "pool_name": "pool-p1-1-data" } }, "aggs": { "all_interests": { "terms": {"field": "size"} } } }' $ curl -XPOST 'localhost:9200/*search' -d '{"query": { "match": { "pool_name": "pool-p1-1-data" } }, "aggs": { "queries": { "terms": {"field": "size"} } } }'

Reference

[1] Docker, https://www.docker.com/

[2] Docker Hub, https://hub.docker.com/

[3] Docker Compose, https://docs.docker.com/compose/

[4] ELK Stack, https://www.elastic.co/elk-stack

[5] Logstash, https://www.elastic.co/products/logstash

[6] Elasticsearch, https://www.elastic.co/products/elasticsearch

[7] Kibana, https://www.elastic.co/products/kibana

[8] Campana, S., & Atlas Collaboration. (2014). Evolution of the ATLAS distributed computing system during the LHC long shutdown. In Journal of Physics: Conference Series (Vol. 513, No. 3, p. 032016). IOP Publishing.

[9] dCache, https://www.dcache.org/