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. 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. 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. 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.
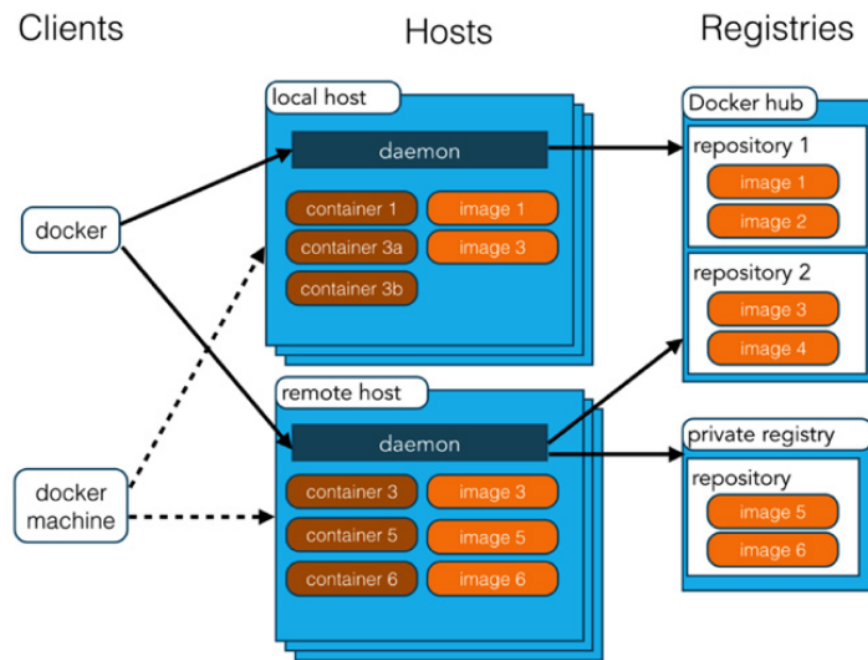

Figure 1: Figure 1-1. How docker works.

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: XXX
- Memory: XXX
- HDD and SSD: XXX
- Operating system: Scientific Linux 7.5
- Docker: X.X.X
- Docker Compose: X.X.X
- Size of dCache billing log (10 years): XXX

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 (Ref X).

Logstash configuraiton for dCache billing log

Elasticsearch

Elasticsearch is a real-time distributed search and analytics engine. 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'
```

Kibana

Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. 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.
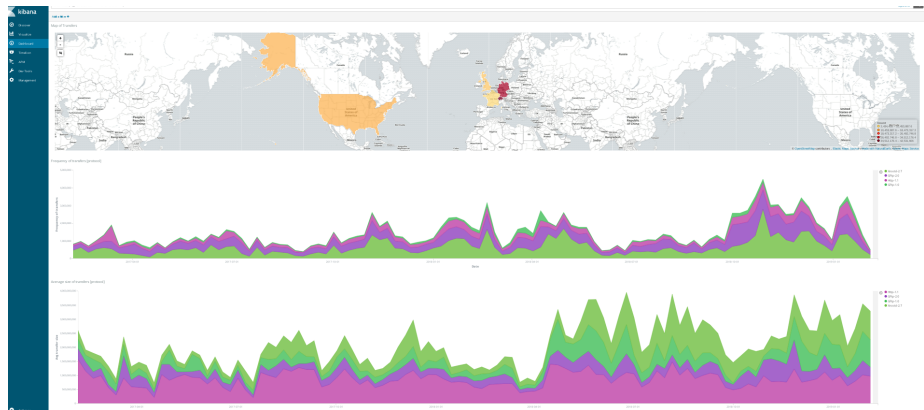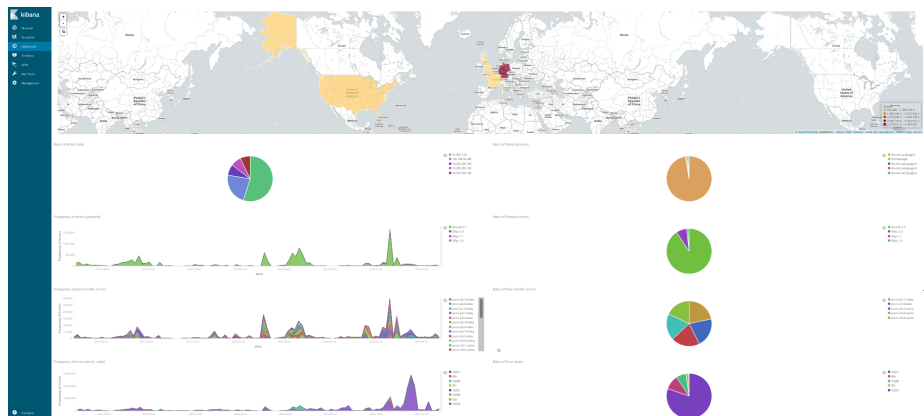
Figure 2: Figure 5-1. Main dashboard

Figure 3: Figure 5-1. Main error dashboard

Cookbook

Running ELK stack in a ATLAS Tier2 site

- Cloning the main command and generating ELK templates

  $ git clone https://github.com/HappyFaceGoettingen/mad-gantry.git

  $ cd mad-gantry

  $ ./mad-gantry -b -t templates/docker-elk

  $ ./mad-gantry -D -a setup

  $ ls payloads/data/GoeGridELK

  billing elasticsearch_index_data

- Copy billing logs into the billing directory

- Turn GoeGridELK instance up

  $ ./mad-gantry -s GoeGridELK -a up

  $ docker ps

Manipulating Kibana Visualisation

Manipulating Kibana Dashboard

Saving Kibana dashboard

In our case, we implemented our own darshboard (ID: WeXmuoICy-wmhE8FvCht). So, exporting the JSON output, and re-using it when the service is up again.

```
$ GET http://localhost:20261/api/kibana/dashboards/export?dashboard=AWeXmuoICywmhE8FvCht >
$ curl -u elastic:changeme -k -XPOST 'http://localhost:20261/api/kibana/dashboards/import'
```

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 'localhost:9200/_cat/indices?v'

- Filter if error_code in billing log is '0'

  $ curl -XPOST 'localhost:9200/_search' -d '{"query": { "bool": { "filter": { "term": {"error_code": "0"} } } } }'

- 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