Journal of Physics: Conference Series **513** (2014) 062001 doi:10.1088/1742-6596/513/6/062001

Agile Infrastructure Monitoring

P. Andrade[1], J. Ascenso[1], I. Fedorko[1], B. Fiorini[1], M. Paladin[1], L. Pigueiras[1], M. Santos[1]

[1] European Organisation for Nuclear Research CERN, CH-1211 Geneva, Switzerland

Abstract. At the present time, data centres are facing a massive rise in virtualisation and cloud computing. The Agile Infrastructure (AI) project is working to deliver new solutions to ease the management of CERN data centres. Part of the solution consists in a new "shared monitoring architecture" which collects and manages monitoring data from all data centre resources. In this article, we present the building blocks of this new monitoring architecture, the different open source technologies selected for each architecture layer, and how we are building a community around this common effort.

1. Introduction

Similar to other areas of the AI project, infrastructure and service monitoring is a core activity in IT which can be greatly improved with the adoption of new tools and processes. To deliver successful monitoring solutions it is important to continuously monitor the status of all resources (network equipment, physical machines, virtual machines, operating systems, applications services, etc.), to efficiently process all collected data, to promptly deliver monitoring results (notifications, alarms, reports, etc.) to the appropriate target, and to have the capability of executing complex queries across distinct monitoring data sets.

Until recently, multiple monitoring applications were deployed at CERN, often tailored for monitoring the status of specific resources. These were independent tools based on different tool chains. Despite their heterogeneity, they all shared a similar architecture and faced the same limitations, leading to unnecessary duplication of effort and increased difficulties in sharing monitoring data. A detailed understanding of the infrastructure performance is also becoming more important, requiring better solutions to combine data and execute complex analysis. In addition, the shift to a cloud infrastructure brings new requirements on monitoring related to more dynamic deployment model of hosts and services.

To improve this situation and deliver a successful monitoring solution we defined and implemented a new "shared monitoring architecture" based on a common set of technologies, delivered under a common collaborative effort.

2. Architecture

The agreed "shared monitoring architecture" is based on well identified building blocks as presented in Figure 1. This architecture is composed of distinct producers which collect data from all data centre resources (these includes generic infrastructure monitoring producers and service specific producers), a common data transport layer, a central archive to store all collected data, an analytics layer to extract knowledge out of the data, a visualisation layer for dashboards and reports, and a separate notifications layer for automatic processing of monitoring data.

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

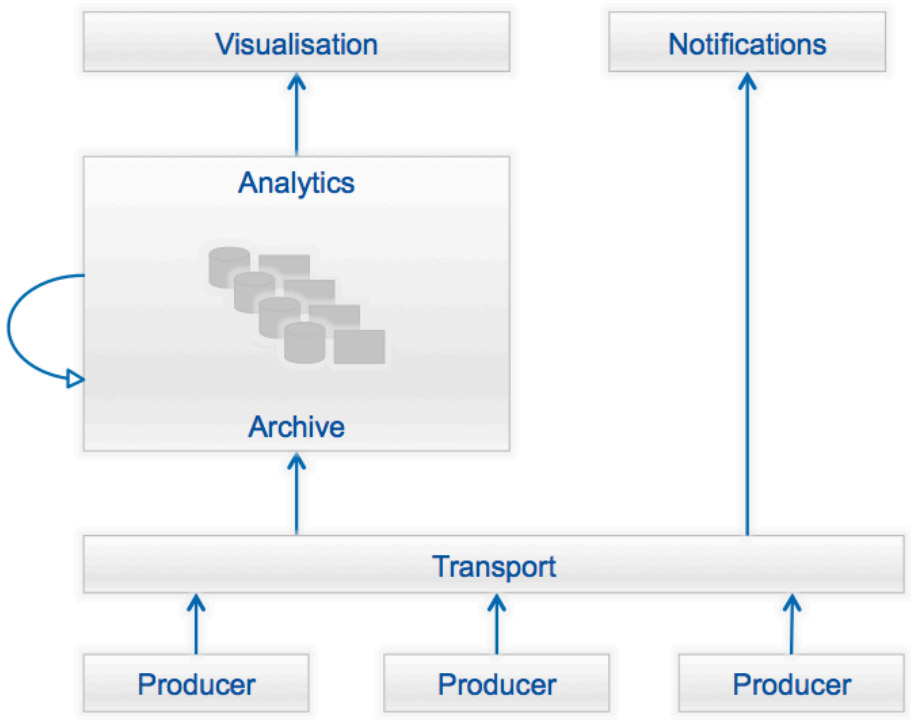

Figure 1. "Shared monitoring architecture"

The scope of each architecture block can be summarised as follows:

- Producers
  - Reuse existing monitoring producers whenever possible
  - Easy tool-chain of new and legacy producers
- Transport
  - Scalable transport to collect all monitoring data
  - Easy integration with different providers and consumers
- Archive
  - Archival of monitoring data
  - Allow future data replay to other tools
  - Offline processing of collected data
- Analytics
  - Real-time queries based on clear APIs
  - Limited data retention
  - Easy to deploy and horizontally scalable
- Visualisation
  - Dynamic and user friendly dashboards
  - Tailored for global and application specific dashboards
- Notifications
  - Quick and reliable delivery of alarms
  - Delivery of notifications via multiple channels

3. Technology

To deliver concrete solutions for a "shared monitoring architecture" we have based our work on existing open source tools. We have selected different tools that fit our requirements, that have large adoption outside CERN and a strong community support. These tools should be easy to adopt, to test, and to deliver results. They should also be modular to allow their replacement in case new (better) solutions are found. Based on these considerations the following technologies were tested and integrated in a common monitoring workflow.

3.1. Transport

For the transport layer we used Flume [1]. Flume is a distributed service for collecting large amounts of data. It is robust and fault tolerant and can horizontally scale due to its multi-tier deployment. It provides many ready-to-use input and output plugins, such as Avro, Thrift, JMS, Syslog, HTTP, ES, HDFS, and allows the easy implementation of custom ones. We have one Flume agent running in each node producing monitoring records (lemon metrics and syslog) that are sent to a second tier of aggregation agents (10 nodes). Finally, a third tier of agents (5+5 nodes) get the aggregated data and write to HDFS and to ElasticSearch.

3.2. Archive

For the archive layer we used Hadoop HDFS. Hadoop [2] is a distributed framework for large data sets processing. HDFS [3] is a distributed filesystem designed to run in commodity hardware. These tools are suitable for applications based on large data sets. We are using the Hadoop cluster provided by another team offering approximately 500 TB of storage. The monitoring data is organised per cluster and a total of 1.8 TB was already archived since mid July 2013. After being stored the data is aggregated by month.

3.3. Analytics

For the analytics layer we used ElasticSearch [4]. ElasticSearch is a distributed RESTful search and analytics engine. It provides real time acquisition capabilities and all data is indexed in real time. ElasticSearch offers built-in features for automatic sharding and replication. It is schema-free and document-oriented (JSON). ElasticSearch is based on the Lucene [5] full text search library. Data can be queried using the provided RESTful API. We have in production 1 master node, 1 search node, and 8 data nodes. Indexes are created per producer and per day (e.g. flume-lemon-YYYY-MM-DD, flume-syslog-YYYY-MM-DD). These indexes have a TTL of 30 days and are divided in 10 shards with 2 replicas per shard.

3.4. Visualisation

For the visualisation layer we used Kibana [6]. Kibana allows to visualize time-stamped data from ElasticSearch. It is designed to query and analyse logs but can also work with time-series data. It allows the dynamic creation of dashboards with simple point and click, no coding is required. Kibana is extremely powerful and is built on AngularJS [7]. We are deploying it using the ElasticSearch Kibana plugin.

3.5. Notifications

For the notification layer we have developed a General Notification Infrastructure (GNI) that is composed of several components responsible for dispatching alarms triggered in each data centre node to multiple operations tools. Different from the other tools, GNI relies on a dedicated transport layer, based on messaging brokers. Different producers publish notifications to the messaging infrastructure, which are then taken by two consumers: one consumer creates tickets for non-masked nodes in the CERN central ticketing system (ServiceNow [8]), while the second

consumer populates a web application showing current notifications. One final component, the no contact processor, is responsible for processing monitoring heartbeat metrics (taken from ElasticSearch) and generate new notifications when nodes become unavailable.

4. Community

The technologies presented above were tested not only for infrastructure monitoring of CERN data centres, but also by other CERN IT teams, creating a community of interest around the "shared monitoring architecture". Some examples of teams building service-specific monitoring solutions using the same technologies:

Cloud Infrastructure
Collection of OpenStack logs, transport via Flume, archive in HDFS, dashboards in Kibana

Batch LSF Accounting
Collection of LSF metrics, transport via Flume, archive in HDFS, dashboards in Kibana

Data Centre Security
Collection of security data, transport via Flume, archive in HDFS, analysis in ElasticSearch

5. Next Steps

We plan to improve the current architecture by exploring two different paths. From one side extend the architecture with a solution for real-time analytics. We plan to integrate a distributed real-time computation engine, such as Spark [9] or Storm [10]. This will allow to solve many use cases which can profit from automatic real-time analytics, such as aggregation of notifications, and to look at other use cases related to online machine learning or extract, transform, and load (ETL). The second improvement relates to the way the architecture is provided. We plan to investigate how to deliver our monitoring services as Platform as a Service (PaaS) and make them easy to instantiate by others. We will look at existing projects, such as OpenStack Heat [11], to understand how they can integrate in our architecture.

6. Summary

A new "shared monitoring architecture" has been established to enhance the monitoring of CERN data centres. We experimented and integrated several open source technologies to build such architecture. The selected tools allowed to deliver complete workflows for different monitoring problems. Several monitoring producers verified these solutions and we will continue to work with these teams to establish a common monitoring effort at CERN.

7. References

[1] http://flume.apache.org/
[2] http://hadoop.apache.org/
[3] http://hadoop.apache.org/docs/stable1/hdfsuserguide.html
[4] http://www.elasticsearch.org/
[5] http://lucene.apache.org/
[6] http://www.elasticsearch.org/overview/kibana/
[7] http://angularjs.org/
[8] http://www.servicenow.com/
[9] http://spark.incubator.apache.org/
[10] http://storm-project.net/
[11] https://wiki.openstack.org/wiki/Heat