Monitoring of IaaS and scientific applications on the Cloud using the Elasticsearch ecosystem

S Bagnasco[1], D Berzano[2], A Guarise[1], S Lusso[1], M Masera[1,3], S Vallero[1,3]

[1]Istituto Nazionale di Fisica Nucleare, Via Pietro Giuria 1, 10125 Torino, IT
[2] CERN - European Organization for Nuclear Research, CH-1211 Geneva 23, CH
[3] Dipartimento di Fisica, Università degli Studi di Torino, Via Pietro Giuria 1, 10125 Torino, IT

E-mail: svallero@to.infn.it

Abstract. The private Cloud at the Torino INFN computing centre offers IaaS services to different scientific computing applications. The infrastructure is managed with the OpenNebula cloud controller. The main stakeholders of the facility are a grid Tier-2 site for the ALICE collaboration at LHC, an interactive analysis facility for the same experiment and a grid Tier-2 site for the BES-III collaboration, plus an increasing number of other small tenants. Besides keeping track of the usage, the automation of dynamic allocation of resources to tenants requires detailed monitoring and accounting of the resource usage. As a first investigation towards this, we set up a monitoring system to inspect the site activities both in terms of IaaS and applications running on the hosted virtual instances. For this purpose we used the Elasticsearch, Logstash and Kibana stack. In the current implementation, the heterogeneous accounting information is fed to different MySQL databases and sent to Elasticsearch via a custom Logstash plugin. For the IaaS metering, we developed sensors for the OpenNebula API. The IaaS level information gathered through the API is sent to the MySQL database through an ad-hoc developed RESTful web service, which is also used for other accounting purposes. Concerning the application level, we used the Root plugin TProofMonSenderSQL to collect accounting data from the interactive analysis facility. The BES-III virtual instances used to be monitored with Zabbix, as a proof of concept we also retrieve the information contained in the Zabbix database. Each of these three cases is indexed separately in Elasticsearch. We are now starting to consider dismissing the intermediate level provided by the SQL database and evaluating a NoSQL option as a unique central database for all the monitoring information. We setup a set of Kibana dashboards with pre-defined queries in order to monitor the relevant information in each case. In this way we have achieved a uniform monitoring interface for both the IaaS and the scientific applications, mostly leveraging off-the-shelf tools.

1. Introduction

The INFN-Torino computing centre hosts a private Cloud infrastructure, which provides IaaS services to different scientific computing applications [1]. The infrastructure is managed with the OpenNebula cloud controller [2]. The main stakeholders of the facility are a grid Tier-2 site for the ALICE collaboration at the CERN LHC, an interactive analysis facility for the same experiment and a separate grid site for the BES-III collaboration. Moreover, the centre offers virtualised batch farms on-demand to a number of smaller local tenants, such as a theory group

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

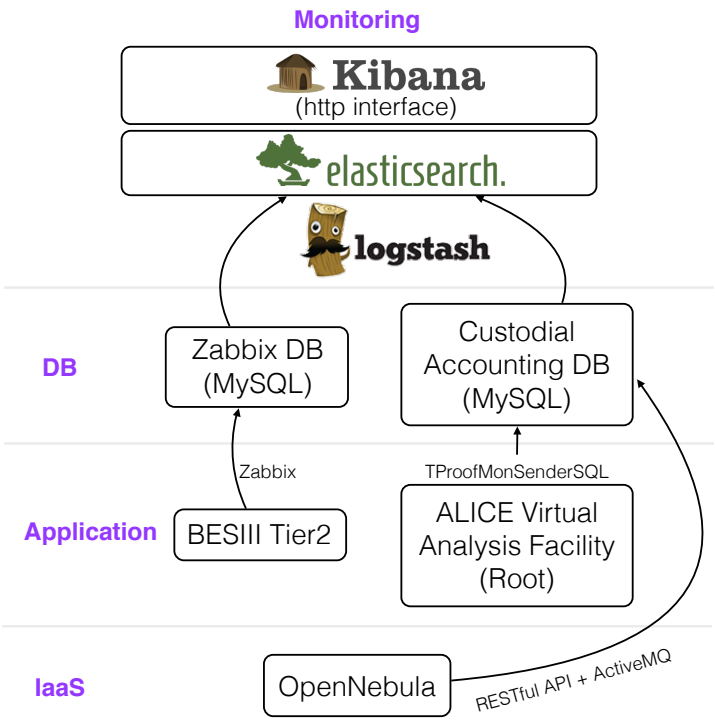

Figure 1. Set-up of the monitoring framework for IaaS and applications at the INFN-Torino private Cloud.

and the Compass collaboration, and single virtual machines tailored to the needs of specific use-cases (e.g. nuclear plant simulations). We are working to achieve the complete automation of resources allocation, which is already fulfilled in the case of the ALICE Virtual Analysis Facility (VAF) [3].

Within this scenario, a detailed monitoring and accounting of resource usage is both mandatory, not least for billing purposes, and challenging because of the heterogeneous data sources. For instance, we need to gather information at the IaaS level (from the cloud controller) and at the application level (e.g. Root [4]) and possibly exploit the data coming from other tools already in use to monitor some application on our Cloud (i.e. Zabbix [5]). Our goal is to achieve a uniform and user-friendly monitoring interface as a single entry point to these miscellaneous data. This paper illustrates how we set up a monitoring framework based on the Elasticsearch [6], Logstash [7] and Kibana [8] stack.

2. Implementation

In the current implementation, the heterogeneous accounting data is fed to different MySQL databases and sent to Elasticsearch via Logstash. We configured a set of Kibana dashboards with pre-defined queries in order to display the relevant information in each case. The set-up is illustrated in Figure 1 and explained in more detail in the following sub-sections.

2.1. The Elasticsearch ecosystem

The Elasticsearch ecosystem is composed by Elasticsearch (ES), Logstash and Kibana and it is generally referred to as the 'ELK stack'.

ES is a search and analytics engine built on top of the Apache Lucene information retrieval

library (Apache open source license). It is document-driven: entries are stored as JSON documents and all fields can be indexed and used in a single query. In our set-up, each application is indexed separately. ES also allows for full-text search on unstructured data, though in our specific case this feature is not fully exploited. Moreover, ES is API driven and can be interfaced with any RESTful API using JSON over http.

Logstash is an open source tool used to collect and parse events and logs to a central service. It can be easily customised via plugins for input, output and data-filters. In order to fit Logstash into our set-up we have developed only a simple plugin to retrieve input data from a MySQL database and a set of configuration files to customise data indexing (one for each application we wished to monitor).

Kibana is the official GUI to display and search ES data. The software could be exposed with any web server, we chose Apache, and no installation is required. Since Kibana does not provide any authentication/authorisation mechanism, these should be implemented at the level of the web server. Kibana has an easy search syntax to query ES. The creation of custom interactive dashboards can be achieved in few mouse clicks, without any prior GUI programming knowledge. This feature makes it a good candidate for provisioning monitoring-as-a-service to the Cloud tenants. Moreover, Kibana provides a set of useful pre-defined plot types like pies, histograms or trends.

The ELK stack was our first choice for implementing the monitoring infrastructure because it is open source, it provides most of the needed features out of the box and it is relatively easy to configure and use. So far, this solution has proven to be suited for our purposes.

2.2. Data preservation: the SQL backend

Monitoring data are stored on a high-availability MySQL server prior to be inserted in the ES engine. This might be seen as a redundant step since data are partially duplicated between the database and ES. Indeed, ES itself can be configured as a fully-fledged (NoSQL) database cluster using its built-in high-availability features. However, in order to ensure flexibility and modularity to our monitoring system, we chose to use ES as a pure search engine with a simple single-node configuration and to delegate data preservation to the MySQL database. With the use of a standard and widely used back-end solution we will be able to move seamlessly away from the ELK stack without losing historical data, should it prove itself not to be the optimal tool. Moreover, the main application hosted at our Cloud besides the Tier-2 sites (i.e. the VAF) relies on Proof [4], which comes with a built-in plugin to send monitoring data to a SQL database at the end of each query. We preferred to avoid writing a custom plugin fitting explicitly our specific monitoring solution.

2.3. Sending data to the database

For the IaaS metering, we developed sensors for the OpenNebula xml-rpc API. The IaaS level information gathered through the API is sent to the MySQL database through an ad-hoc developed RESTful web service, which is also used for accounting purposes. The service relies on the ActiveMQ messaging server to asynchronously and efficiently feed metering records to the database. A similar application is also being developed for billing. The relevant quantities that we feed to the database are the resource usage per tenant/group in terms of number of virtual machines, number of cores, memory and ephemeral disk space.

Concerning the application level, we use the Proof plugin *TProofMonSenderSQL* to collect accounting data from the interactive analysis facility. At the end of each user query, data are gathered from the worker-nodes to the Proof master and sent to the database with a standard MySQL client/server protocol. In this case, we also monitor some additional observables such as e.g. the number of workers, the datasets analysed or the number of events processed. Moreover, since the BES-III Tier-2 site used to be monitored with Zabbix, as a proof of concept we also

Figure 2. Partial view of the Kibana dashboard for the ALICE Analysis Facility at the INFN-Torino computing centre.

use the data from the Zabbix database. In this case we can get even more refined information such as resource consumption for specific processes, e.g. the BES-III Offline Software System (BOSS).

3. Custom dashboards

At present, the monitoring framework described in this paper is used to display data coming from the cloud controller (both from the accounting service and the test version of the billing service), the VAF and the BES-III Tier-2. Each use-case has its own Kibana dashboard.

As an example, in Figure 2 we show a snapshot of part of the VAF dashboard. The top-most

plot is a *terms* panel, which displays the results of an ES facet as a bar chart. By mouse-clicking on a user's bar, only entries for that user are displayed in all other panels.
The second top-most plots are *histogram* panels, showing the number of workers requested by each user and the average CPU efficiency as a function of the query time (similar plots for the memory usage on the master/workers are hidden in the figure). The plots' time-range can be easily selected by dragging with the mouse on the corresponding axis. This operation also sets the desired time-range on all other plots in the dashboard. In order to draw a separate line for each user in the plots, we have saved a set of pre-defined ES queries, one for each user, and configured Kibana in order to show them all in the same panel.
In the bottom-most row one can see some information concerning the datasets analysed: the LHC data-taking period flag, the run number and the type of file analysed. By selecting e.g. one data-taking period in the bottom-left panel, one automatically gets a list of users who analysed it in the top-most panel.

Similar dynamic dashboards can be set-up in few minutes without any programming, only a good knowledge of the back-end database schema is required.

4. Summary and outlook

At the INFN-Torino computing centre we have set-up a uniform monitoring interface for both the IaaS and the scientific applications, mostly leveraging off-the-shelf tools. The infrastructure relies on a SQL database back-end for data preservation and to ensure flexibility to choose a different monitoring solution if needed. We are now starting to consider dismissing the intermediate level provided by the SQL database and evaluating a NoSQL option as a unique central database for all the monitoring information.
For the IaaS accounting we have developed a custom RESTful web service to manage and query the database, which relies on a message queue to transport usage records.

We are currently working on a billing system for our private Cloud, which also relies on the ELK stack in its graphical interface.
The next step will be to define a model for monitoring-as-a-service, based on the tools described in this paper, which the Cloud tenants could easily configure to suit their needs.

Acknowledgements

The present work is partially funded under contract 20108T4XTM of Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale (PRIN), Italy.

References

[1] Bagnsco S, Berzano D, Brunetti R, Lusso S and Vallero S 2014 *Journal of Physics: Conference Series* **513** 032100
[2] Moreno-Vozmediano R, Montero R S and Llorente I M 2012 *IEEE Computer* **45** 65-72
[3] Berzano D, Blomer J, Buncic P, Charalampidis I, Ganis G, Lestaris G and Meusel R 2014 *Journal of Physics: Conference Series* **513** 032007
[4] Brun R and Rademakers F 1997 *Nuclear Instruments and Methods in Physics Research* A **389** 81-86
[5] Zabbix [`http://www.zabbix.com/`]
[6] Elasticsearch [`http://www.elasticsearch.org`]
[7] Logstash [`http://logstash.net`]
[8] Kibana [`http://www.elasticsearch.org/overview/kibana`]