Monitoring CMS experiment data and infrastructure for LHC Run3 and beyond

Ceyhun Uzunoglu[1], Felipe Gomez[2], Brij Kishor Jashal[3], Valentin Y Kuznetsov[4], Federica Legger[5], Benedikt Maier[1], Oscar Fernando Garzon Miguez[6], Garyfallia Paspalaki[7], on behalf of the CMS Collaboration

[1] CERN, Geneva, Switzerland
[2] Universidad de los Andes, Colombia
[3] Tata Inst. of Fundamental Research, Mumbai, India
[4] Cornell University, New York, U.S.A.
[5] Istituto Nazionale di Fisica Nucleare, via Pietro Giuria 1, 10125 Torino, Italy
[6] Fermi National Accelerator Lab., IL, U.S.A.
[7] Purdue University, IN, U.S.A.

Abstract. The CMS experiment at CERN is a scientific endeavor that requires the coordinated efforts of thousands of researchers and engineers from around the world. To process and store the petabytes of data produced by CMS, the experiment relies on a distributed computing infrastructure. The CMS Offline and Computing group is responsible for workflow and data management, to ensure that critical datasets are processed and made available for the physics community in a timely manner. Scalable and reliable monitoring services are essential to ensure the efficient usage and performance of the computing infrastructure. This note presents the CMS Monitoring infrastructure, services, and applications, along with the improvements made in data management monitoring over the past few years. By adopting open-source technologies such as Kubernetes and Prometheus, and by relying on the CERN IT and MONIT services, the CMS Monitoring group has established a reliable and scalable monitoring infrastructure and services which are essential for computing operations of the CMS collaboration.

1. Introduction

Computational tasks for the CMS experiment [1] at the Large Hadron Collider (LHC) at CERN are performed in a distributed infrastructure consisting of more than one hundred Worldwide LHC Computing Grid (WLCG) sites [2]. Physics data are stored, processed, and transferred in these computing centers worldwide. Computing operations in CMS focus on the efficient exploitation of the pledged resources dedicated to CMS in the worldwide computing grid. Services involve workload and workflow management [3, 4, 5], data storage and transfer management [6, 7, 8]. To help manage and analyze the vast amounts of data produced by the CMS computing services, the CMS Monitoring and Analytics group was established [9].

Over the past few years, monitoring of CMS computing services has evolved and consolidated around the CERN-IT and Monit infrastructures [10] and open-source technologies. Along with typical device metrics such as availability, audit, system, and event logs, many different services produce application logs and other important metadata which need to be regularly monitored.

Rather than using custom monitoring tools, many different services have adopted the CMS Monitoring infrastructure for their needs. CERN-IT provides services to store, process and access these muti-temporal monitoring data-sets. This infrastructure includes services like Hadoop [11], Spark [11], OpenSearch (formerly ElasticSearch [12]), Grafana [13], OpenStack, messaging queues [11], and more. In recent years, CMS Monitoring has built its infrastructure around these services and has greatly benefited from them. In addition to the services provided by CERN-IT, CMS Monitoring has also added its own monitoring infrastructure to cover also the collaboration's more advanced monitoring needs. Most of these monitoring services are deployed in Kubernetes [14] (see Fig. 1).

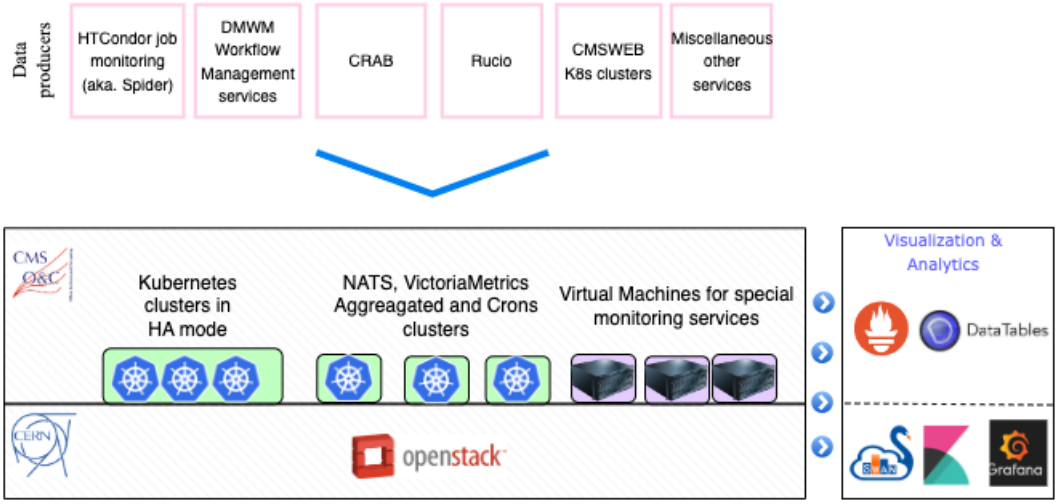

Figure 1. Overview of the CMS Monitoring Infrastructure. We show the data producers, the computing resources on which the monitoring services run, and the visualization frameworks used to access and display the monitoring metrics.

The monitoring needs of the CMS computing infrastructure vary widely based on data type, structure, and size. Therefore, each monitoring data is processed, transferred through, and stored in different mediums. These mediums can be categorized as follows:

1.1. Time-series metrics and pull-based monitoring

Many service metrics are essential to monitor service availability and usage statistics. These metrics are mostly in time-series format and are stored in the CMS monitoring Prometheus [15]/VictoriaMetrics [16] services. Typically these metrics stem from different services in different domains and infrastructures, which force pull-based access only. Therefore, they are accessed with tailor-made tools and exposed to Prometheus. For this purpose, CMS Monitoring has leveraged special exporters mostly written in GoLang and Python, running as a service in Kubernetes.

1.2. Logs with different formats and push-based monitoring

Application logs may come in very different formats or not formatted at all. Therefore, they need to be parsed in the correct format and stored depending on their retention policy, in HDFS, OpenSearch, InfluxDB [17] services (respectively long, medium or short term). For this

push-based monitoring, CMS Monitoring hosts LogStash [12] services to parse unstructured data and transfer it to proper monitoring storage. Additionally, our users are not forced to use Logstash to parse and format their logs. Custom producers can send data directly to ElasticSearch/OpenSearch, HDFS, or InfluxDB through message brokers provided by CERN-IT. To make this kind of data publishing more robust and standardized, we provide Python library support for message broker communication.

1.3. Aggregation and analysis support

In recent years, the monitoring needs that require processing big data in CMS have increased, and we have standardized our big data aggregations using Spark and data transfer tools in Kubernetes environments. This is explained further in Section 2.

1.4. Kubernetes monitoring

There are many Kubernetes clusters used for the computing operations in the CMS collaboration. Our Prometheus/VictoriaMetrics services are made available as a back-end metric storage for these clusters. They have also helped in centralizing the monitoring access of all Kubernetes cluster metrics.

1.5. Data access and front-end services

Front-end tools are an essential part of monitoring, and CMS Monitoring uses CERN-IT Grafana, which is the main entry-point for the monitoring stack, and OpenSearch Kibana [12] as central front-ends for all back-end monitoring services. We also use custom monitoring tools developed for special purposes such as JQuery DataTable pages and Plotly dashboards.

Overall, CMS Monitoring handles unstructured, structured, time-series and static data, and stores or transfers them to NoSQL (CERN-IT: OpenSearch, InfluxDB; in-house: Prometheus, VictoriaMetrics) and long-term storage (HDFS) using open-source technologies.

2. Kubernetes infrastructure

CMS Monitoring has transitioned to Kubernetes as a service infrastructure and increased its number of Kubernetes services over the years. Kubernetes is not only useful for application servers but also for all kinds of monitoring suites. Because of its core features such as scalability, fast deployment processes, low operation costs, and flexibility, it allows for the implementation of new monitoring services and their maintenance in a scalable and sustainable way.

Currently, CMS Monitoring manages and operates services deployed in eight Kubernetes clusters. Three of them (called MAIN, HA-1, HA-2) work together to provide critical services in high availability mode, while the other five (CRONS, NATS, VM-AGG, DM-MONIT, TEST) host special services for different purposes. See Fig. 2 for an architectural diagram.

2.1. Highly Available Prometheus, VictoriaMetrics (VM) and their service suites

Prometheus is a Cloud Native Computing Foundation (CNCF) supported open-source time-series metric storage with a rich ecosystem of features and tools for various operations. In our infrastructure, Prometheus is deployed in high availability mode with Prometheus instances deployed in three different Kubernetes clusters in different CERN compute availability zones. Along with Prometheus instances, VictoriaMetrics services are also deployed in each cluster that hosts Prometheus. VictoriaMetrics serves as both a long-term metric storage and remote-write storage to our own Prometheus instances, as well as for more than ten external Prometheus instances hosted by other CMS groups. All metrics in Prometheus and VictoriaMetrics can be

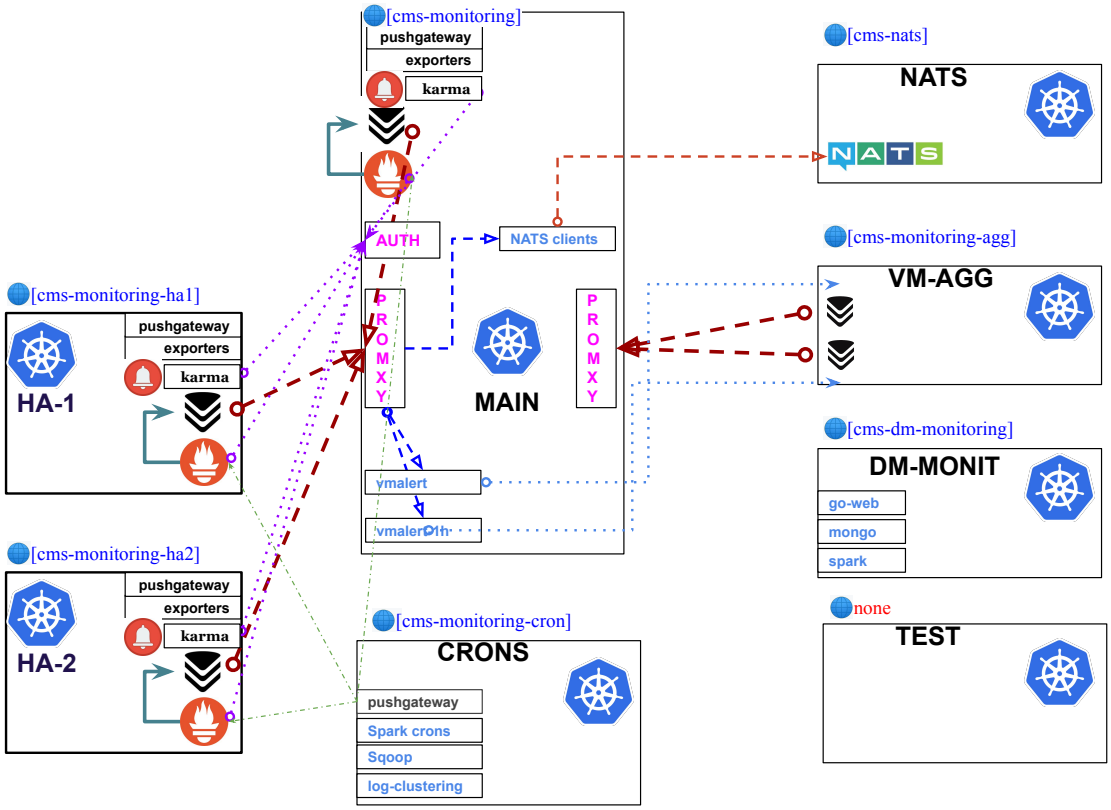

Figure 2. The CMS Monitoring Kubernetes infrastructure. The Kubernetes clusters MAIN, HA-1, HA-2, CRONS, NATS, VM-AGG, DM-MONIT, and TEST are shown. A detailed description of the services running in each cluster can be found in the text.

accessed through a single Grafana data source using the Promxy service, which is connected to all Prometheus and VictoriaMetrics instances in these three clusters.

CMS Monitoring's Prometheus service scrapes metrics from over 150 targets, which include external node exporters of machines, along with custom exporters mentioned in the pull-based monitoring section. Some of these targets are special-purpose exporters that collect data from different sources, convert them to Prometheus time-series metric format, and expose them. All these exporters are also deployed in each high availability cluster. For batch job metrics, CMS Monitoring provides the PushGateway service as another monitoring solution.

Similar to the aforementioned services, AlertManager (AM) [15] instances are also distributed in these three high availability Kubernetes clusters. The computing operation teams may create their own alert rules for their metrics in the Prometheus service, and AlertManager manages the transmission of notifications via various communication channels (email, Slack, custom webhook).

The Prometheus-VictoriaMetrics-AlertManager-Exporters service stack in three clusters has proven its abilities over the years, with zero downtime even during mandatory migrations and IT service incidents.

2.2. Other Kubernetes clusters and services

Other Kubernetes clusters in CMS Monitoring were created for specific use cases and host many important services. Following the Kubernetes architectural design principles, each service stack is deployed in a dedicated cluster. These are listed below:

- The CRONS cluster hosts services for Spark jobs, Sqoop [11] jobs, and other batch processing services in the Kubernetes environment. Each Spark job, running hourly, daily or monthly, is a critical data producer for specific monitoring workflows. Due to the heavy requirements of CPU and memory usage for the execution of these jobs, it is necessary to orchestrate their execution to optimise the usage of the cluster.
- The CMS Monitoring VM-aggregated (VM-AGG) cluster hosts two VictoriaMetrics instances and their services to serve as long-term storage for Rucio service metrics. These metrics reach the VictoriaMetrics backend after some aggregations are performed in the vmagent services of the MAIN cluster.
- NATS [19] is a messaging-oriented middleware service deployed in our NATS cluster, providing another alternative solution for message broker needs.
- The Data Management (DM-MONIT) monitoring cluster is dedicated to a set of recently created services, described in the next section.

3. Data Management Monitoring Improvements

To help with the transition [20] from PhEDEx to Rucio for data transfers and management, CMS Monitoring has developed a critical service to monitor dataset usage and storage statistics in the Rucio Storage Elements (RSE) on the WLCG sites. Rucio has its own set of monitoring services to keep detailed metrics about the current placement of dataset replicas in RSEs. However, Rucio internal monitoring tools were not integrated with the CMS Data Bookkeeping Service (DBS) [21], which stores metadata of all datasets. Rucio tables provide metrics such as size, last access, last creation, event count, replica file count, accessed file count, locks, lock rules, and account information of file replicas, blocks, and datasets and containers in the RSEs. On the other hand, DBS tables provide metadata of the file, block, and dataset, including data-tier, campaign, acquisition era, and size. While Rucio provides the actual location of dataset replicas and the reason (lock rules) in the computing grid storage, DBS is the main source of their definition.

To enable this integration, CMS Monitoring has developed two data producers that aggregate Rucio and DBS database tables using Spark jobs and send the data to MONIT OpenSearch for time-series historical monitoring, along with a stand-alone MongoDB instance that serves as a storage to a web service backed by Go and JQuery DataTables stack (see Fig. 3).

Figure 3. The data pipeline for data management monitoring based on Sqoop dumps of the Rucio and DBS database tables. The monitoring metrics are aggregated with Spark jobs and stored in MongoDB. Metrics are visualised using a web service written in GoLang and based on DataTables [22]

3.1. Data pipeline

The data pipeline for data management monitoring consists of Sqoop jobs, Spark jobs, and an import tool. Sqoop jobs move data from the DBS and Rucio databases to HDFS storage. Spark jobs are designed to join two different information systems with custom, domain-specific knowledge. They involve more than twenty join operations between Spark datasets and tens of aggregations to reach the final data schema, which can be defined as all the required specifications of a dataset including all its metadata and locations. The data pipeline ends with transporting the final aggregated results to OpenSearch through an AMQ broker and to MongoDB with its import tools.

3.2. Data access

Historical aggregated dataset monitoring results are stored in a NoSQL storage (OpenSearch) with longer retention and visualised with a dedicated Grafana dashboard that uses OpenSearch as data source. The dashboard provides fully capable visualizations that offer a time-series monitoring overview. Secondly, a web service of a single page JQuery DataTables, written in GoLang and backed by MongoDB, provides fast access to dataset monitoring results for operational needs, including advanced search capabilities over more than 600 thousand dataset entries and close to 6 million RSE replica entries.

4. Conclusions and Future Work

CMS Monitoring has evolved with its service stack by developing a multitude of monitoring services and meeting diverse requirements of different infrastructures, data types, and services over the years, with the help of supporting infrastructure from CERN IT.

CMS Monitoring will continue to improve its infrastructure and add new services to its toolbox by adopting more open-search technologies. Rucio and DBS dataset monitoring services have significantly contributed to the data management operations, and will be further improved with additional features in case new operational needs arise.

References

[1] CMS Collaboration, JINST 3 S08004 (2008)
[2] I. Bird et al., CERN-LHCC-2014-014, LCG-TDR-002 (2014)
[3] I. Sfiligoi et al., proceedings of the WRI World Congress on Computer Science and Information Engineering, Vol. 2, 2428-432 (2009)
[4] D. Thain, T. Tannenbaum, and M. Livny, Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, 323-356 (2005)
[5] T. Ivanov et al., EPJ Web Conf, 03006 (2019)
[6] M. Giffels, Y. Guo, V. Kuznetsov, N. Magini and T. Wildish, J. Phys.: Conf. Ser., Vol. 513, Issue 4 (2014)
[7] J. Rehn, et al. Proc. CHEP06, Computing in High Energy Physics, Mumbai, India (2006).
[8] M. Barisits, T. Beermann, F. Berghaus, et al. Comput Softw Big Sci (2019) 3: 11
[9] C. Ariza-Porras, V. Kuznetsov, F. Legger, Comput Softw Big Sci (2021) 5:5
[10] A. Aimar, et al., J. Phys.: Conf. Ser. 898 (2017) 092033
[11] *Apache Projects*, `https://projects.apache.org/projects.html` (2023), accessed: 2023-02-18
[12] *Elastic Stack*, `https://www.elastic.co/elastic-stack/` (2023), accessed: 2023-02-18
[13] *Grafana*, `http://grafana.org` (2023), accessed: 2023-02-18
[14] *Kubernetes*, `https://kubernetes.io/` (2023), accessed: 2023-02-18
[15] *Prometheus, AM*, `https://prometheus.io/docs/introduction/overview/` (2023), accessed: 2023-02-18
[16] *VictoriaMetrics*, `https://victoriametrics.com/` (2023), accessed: 2023-02-18
[17] *InfluxDB*, `https://www.influxdata.com/time-series-platform/influxdb/` (2023), accessed: 2023-02-18
[18] *VictoriaMetrics at CMS* `https://docs.victoriametrics.com/CaseStudies.html#cern` (2020)
[19] *NATS* `https://nats.io/` (2023), accessed: 2023-02-18
[20] E. Vaandering et al., EPJ Web of Conferences 245, 04033 (2020)
[21] V. Kuznetsov et al. J. Phys.: Conf. Ser. 219 042043 (2010)
[22] *Rucio Dataset Monitoring Web Service*, `https://github.com/dmwm/CMSMonitoring/tree/master/rucio-dataset-monitoring` (2022)