PROCEEDINGS OF SCIENCE

A Big Data Platform for Scalable Monitoring and Analytics at INFN-CNAF

Marco Lorusso[a,*] **and Diego Michelotto**[a]

[a] *CNAF Division, Istituto Nazionale di Fisica Nucleare, Viale Berti Pichat 6/2, Bologna, Italy*

E-mail: marco.lorusso@cnaf.infn.it

A heterogeneous data management and analytics platform has been deployed over the years at INFN-CNAF in Bologna, with the goal to support both system administrators and user-support teams in monitoring, diagnostics, and operational decision-making across the data center, that serves multiple particle and astroparticle experiments (and beyond) in Italy and internationally. The platform is built around a general-purpose message-handling backbone based on Apache Kafka, enabling scalable ingestion of high-volume, high-velocity data streams. On top of this backbone, a log-analysis stack - comprising Logstash, OpenSearch, and OpenSearch Dashboards, the open-source fork of the ELK suite maintained by AWS - provides powerful indexing, search, and visualization capabilities, including advanced authentication and authorization plugins. This contribution presents the architecture of the platform and the design choices that guided its implementation, complemented by a use case on farming services that demonstrates the platform's applicability in real-world scenarios.

International Symposium on Grids and Clouds (ISGC2026)
15-20, March, 2026
Academia Sinica, Taipei, Taiwan

*Speaker

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0) All rights for text and data mining, AI training, and similar technologies for commercial purposes, are reserved. ISSN 1824-8039 . Published by SISSA Medialab.

PoS(ISGC2026)012

1. Introduction

The management and reporting of modern data centers has become increasingly challenging due to the rapid growth in scale and complexity, i.e. establishment of heterogeneous computing. Facilities supporting scientific computing must handle thousands of interconnected components, ranging from compute nodes and storage systems to complex networking infrastructures, especially when tasked to provide cloud-like services and large-scale and distributed data processing. In such environments, even minor inefficiencies or undetected anomalies can propagate quickly, leading to performance degradation or service disruption.

In this context, the Computing division of the Italian National Institute for Nuclear and Subnuclear Physics (INFN), called CNAF [1], has to keep up with this trend employing new technologies like AI and predictive maintenance. Indeed, CNAF, and the data centre it manages in Bologna (Italy), plays a central role in supporting large-scale scientific computing, most notably as a Tier-1 facility of the CERN Worldwide LHC Computing Grid (WLCG). This is the reason why it is required to manage substantial volumes of data and provide reliable, high-throughput processing capabilities for distributed workloads. At the same time, CNAF is involved in a broader ecosystem that includes technology transfer activities and the coordination of distributed cloud infrastructures, further increasing the diversity of services and operational requirements.

It is useful to distinguish two parts of the CNAF monitoring environment. The first is the established production monitoring service based on Sensu [2], InfluxDB [3], and Grafana [4]. This stack is currently used for numerical infrastructure metrics such as CPU load, memory usage, and I/O activity. The second is the Big Data Platform (BDP), which extends the monitoring environment with a log-processing path based on Filebeat [5], Apache Kafka [6], Logstash [7], OpenSearch [8], and OpenSearch Dashboards [9]. At the time of writing, the OpenSearch/OpenSearch Dashboards component is in pre-production for the OpenStack environment rather than being the production system for all CNAF monitoring data.

The BDP builds on previous CNAF activities. [10] presented an initial ELK-based implementation for collecting, parsing, and displaying heterogeneous system logs, while [11] investigated predictive-maintenance techniques using structured monitoring data, with a focus on storage services. [12] subsequently described the broader BDP architecture while it was still under development, including Kafka-based decoupling between producers and consumers and multiple storage/analysis back ends. The present work reports the subsequent evolution of that architecture toward an OpenSearch-based log-analysis service, and concentrates on its current deployment for CNAF computing services.

Given the increasing complexity of interdependencies among services operating at the data centre, together with the anticipated growth in resource demand, CNAF is undertaking efforts to deploy a more robust and flexible monitoring system. The proposed platform is intended to unify the heterogeneous infrastructures currently in place, enabling the exploration of solutions for the development of predictive maintenance models aimed at early detection and prevention of failures. Specifically, the BDP will provide a framework capable of correlating logs and metrics originating from diverse sources and devices through the application of Big Data analytics techniques.

PoS(ISGC2026)012

2. The Big Data Platform: architecture and current deployment status

The BDP follows the data path shown in Figure 1. Filebeat agents collect log records from service hosts and forward them to Kafka. Kafka acts as the message broker and buffering layer, decoupling producers from downstream processing. Logstash consumes events, applies service-specific parsing and normalization, and forwards structured documents to OpenSearch. OpenSearch provides indexed storage and search, while OpenSearch Dashboards provides the interactive user interface.

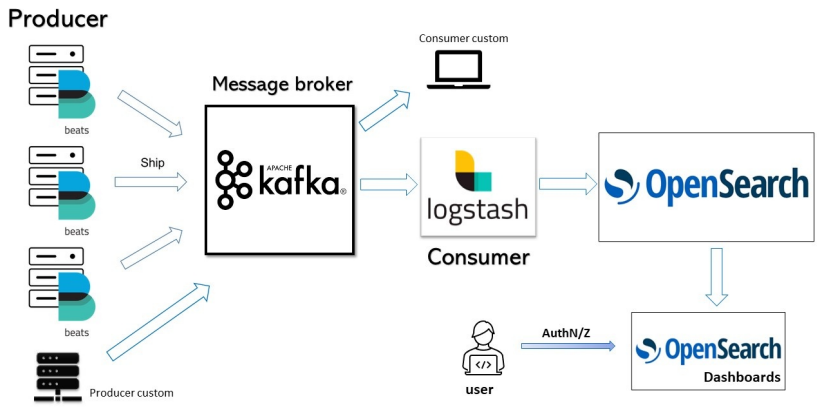

Figure 1: CNAF Big Data Platform architecture. Filebeat collects logs, Kafka provides the message-broker and buffering layer, Logstash parses and normalizes events, and OpenSearch indexes them for exploration through OpenSearch Dashboards. Original schematic produced by the authors.

Rather than treating the individual software packages as the contribution of this work, Table 1 summarizes their role in the CNAF deployment. The architectural value of the pipeline is the separation of ingestion, buffering, transformation, indexing, and visualization, so that each stage can be evolved independently.

Table 1: Role of the main BDP components in the CNAF deployment.

Component	Role at CNAF
Filebeat	Lightweight collection of log files on source hosts and forwarding of events to Kafka.
Kafka	Decoupling and buffering between producers and downstream consumers, with topic-based separation of data streams.
Logstash	Service-specific parsing, normalization, field enrichment, and routing toward the appropriate index namespace.
OpenSearch	Distributed indexing and search of structured log events.
OpenSearch Dashboards	Tenant-aware interactive search, filtering, visualization, and dashboard creation.

Kafka is used as the common event-streaming layer between ingestion and consumers. This

PoS(ISGC2026)012

avoids coupling the availability and processing rate of Filebeat producers directly to Logstash and makes it possible to attach additional consumers when needed. The BDP configuration, reported in [12], sees three Kafka brokers, three partitions per topic, two replicas, and a three-day retention window. With this design, topic separation also limits interference between independent producers and use cases. In the present configuration, OpenStack logs used in the Farming use case described in Section 4 are carried by the `farm` Kafka topic.

2.1 OpenSearch and multi-tenancy

OpenSearch is a fully open-source search and analytics suite based on Apache Lucene. The project originated from Elasticsearch 7.10.2 and OpenSearch Dashboards from Kibana 7.10.2 [13]. Since September 2024, the OpenSearch project has been hosted under the OpenSearch Software Foundation, a vendor-neutral project of the Linux Foundation; the Foundation provides organizational support, while technical governance is defined separately by the project's technical charter [14]. This governance model is different from describing OpenSearch as an ELK fork maintained by a single vendor.

At CNAF, OpenSearch is currently deployed as a two-node cluster. Each primary shard is configured with two replicas, allowing copies of the shard to be distributed across the nodes. Operational domains are separated using OpenSearch tenants; examples include the Farming and Storage domains. This provides logical multi-tenancy inside the shared cluster, keeping tenant-specific data exploration and dashboards separate while still sharing the same underlying OpenSearch resources. It should therefore be understood as logical rather than physical resource isolation.

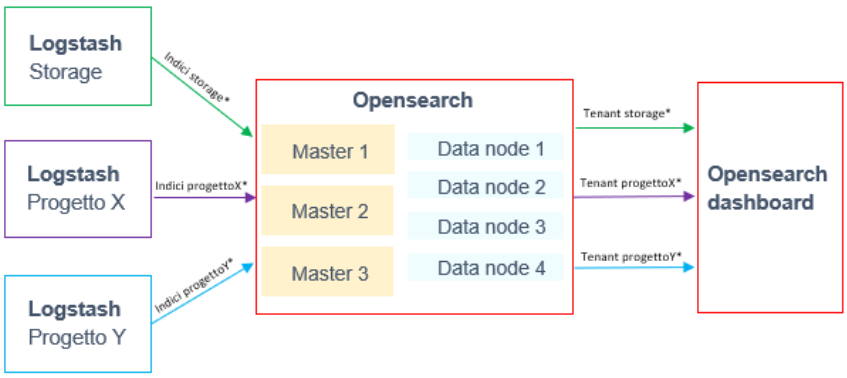

Figure 2: Author-generated schematic of the shared OpenSearch cluster and logical tenant separation. Logstash pipelines route data associated with different operational domains to the common cluster, while OpenSearch Dashboards exposes tenant-specific views.

3. Deployment scale and data lifecycle

The current pre-production OpenStack deployment receives logs from 17 hosts. During a representative 15-day observation interval, from 14 to 28 September, the pipeline ingested 27,818,309 log events. This corresponds to an average of approximately 1.85×10^6 events per day, or about 21.5

PoS(ISGC2026)012

events/s when averaged over the complete interval. These are observed workload figures rather than a benchmark of the maximum sustainable throughput; instantaneous rates at individual pipeline stages can be substantially higher than the long-term average.

3.1 Observed Kafka workload and broker resource use

Operational monitoring is available for the Kafka layer through Grafana. Figure 3 shows a representative 15-day view, from 14 to 28 September 2026, for one Kafka broker. Because the dashboard covers a single broker, the values below are used only to characterize the observed workload on that node and must not be interpreted as aggregate throughput or capacity of the Kafka cluster.

The Kafka topic relevant to the OpenStack use case presented later in this paper is `farm`. On the monitored broker, this topic shows a mean input rate of 13.7 messages/s and a maximum of 39.6 messages/s, with a mean incoming data rate of 1.89 kB/s and a maximum of 4.45 kB/s. Other application topics are also visible in the dashboard: `farm-test` averages 59.7 messages/s and peaks at 104 messages/s, `log-storage` averages 20.0 messages/s with a maximum of 50.0 messages/s, and `servnaz` averages 0.872 messages/s. These additional streams are reported only to characterize the workload observed on the broker and are not the source of the OpenStack logs discussed in Section 4. The dashboard also reports the Kafka internal `__consumer_offsets` topic, at 4 messages/s and 284 B/s on average. Summing the application topics shown gives approximately 94 messages/s on this broker during the interval, but this number mixes several independent data streams and is therefore not directly comparable with the OpenSearch indexed-event average reported above.

The same dashboard indicates low broker-side resource consumption for this workload. CPU use remains close to 0.15 CPU cores, JVM memory use fluctuates roughly between 0.4 and 0.95 GiB, and time spent in garbage collection remains below approximately 0.07% for the displayed period. These values are descriptive observations from one broker rather than the result of a controlled performance benchmark.

OpenSearch indices are organized on a daily basis. A new index is opened for each day of data, providing a simple time-based unit for searching and lifecycle management. After one month, older indices are closed. Closing an index removes it from normal search activity while retaining the indexed data so that it can be reopened if required; it should therefore not be confused with permanent deletion. This policy reduces the number of actively searchable indices while retaining older data according to the current operational practice.

Table 2 summarizes the current deployment parameters that are available at the time of writing. Because the OpenSearch component is not yet in full production, a systematic query-latency benchmark has not yet been performed; consequently, no quantitative latency claim is made in this paper.

These figures are intended to characterize the present deployment rather than to establish a saturation point. The Kafka dashboard provides node-level operational evidence for one broker, but OpenSearch monitoring is not currently available with sufficient reliability to report resource-consumption or query-latency statistics. A dedicated end-to-end performance study, including aggregate peak event rate, indexing throughput, disk growth, and percentile query latency, is therefore left for future characterization.

PoS(ISGC2026)012

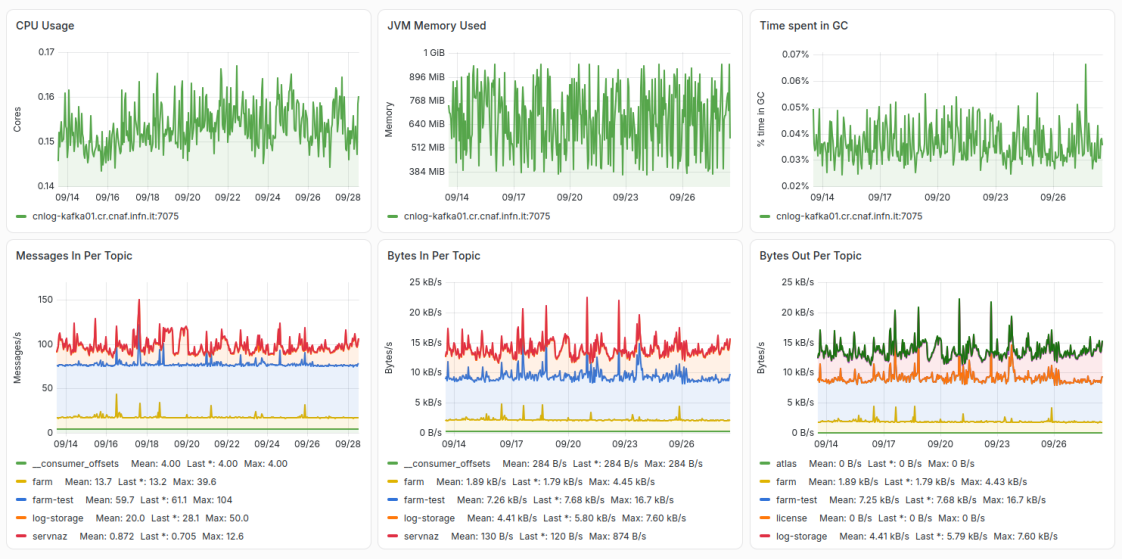

Figure 3: Representative CNAF Kafka monitoring dashboard for one broker over a 15-day interval (14 – 28 September). The panels show broker CPU use, JVM memory, garbage-collection time, and per-topic message and byte rates. The measurements refer to a single Kafka node and are not aggregate Kafka-cluster throughput. Screenshot from the CNAF operational Grafana monitoring service.

Table 2: Representative parameters of the current OpenStack log-analysis deployment.

Parameter	**Current value/status**
Metrics monitoring	Sensu + InfluxDB + Grafana; production
Log analysis	Kafka + Logstash + OpenSearch + OpenSearch Dashboards; pre-production for OpenStack
OpenStack source hosts	17
Observed ingested events	27,818,309 over 15 days (14–28 September)
Average ingestion	About 1.85 million events/day (about 21.5 events/s averaged over 15 days)
Single-broker Kafka observation	`farm` (OpenStack) averages 13.7 messages/s and peaks at 39.6 messages/s; all application topics shown total about 94 messages/s mean
OpenSearch cluster	3 nodes
Replica setting	2 replicas per primary shard
Index lifecycle	One new index per day; indices closed after one month
Tenant separation	Logical OpenSearch tenants, e.g. Farming and Storage
Typical query latency	Not yet systematically benchmarked

PoS(ISGC2026)012

4. The BDP for Computing Services

The computing cluster managed by CNAF is capable of supporting local, Grid and Cloud resource requests on both local and remote environments. The available computing capacity is distributed across both on-site and external resources: a significant fraction of the processing power is provided by leased nodes hosted at the CINECA facility [15], interconnected with CNAF through a high-bandwidth 400 Gbit/s link. The federated architecture enables CNAF to deliver scalable and flexible computing services to multiple experiments.

The operation and management of the core computing services (including Worker Nodes, user access interfaces, Grid services, and authentication infrastructure) are overseen by the Farming group within the Tier-1 data centre unit.

To complement its batch and Grid computing capabilities, CNAF provides Cloud services based on OpenStack [16], an open-source platform for the management and orchestration of virtualized resources. It allows to control large pools of compute, storage, and networking resources throughout a datacenter, all managed and provisioned through APIs with common authentication mechanisms.

Through a modular architecture, OpenStack manages key aspects of the Cloud stack, including compute resources, networking, and storage. A non-exhaustive list of modules includes:

Nova : provisions and manages virtual machine instances and other compute resource;

Cinder : provides block storage volumes for virtual machines, including volume creation, attachment, and snapshots;

Neutron : manages networks, subnets, ports, IP addressing, and other virtual network functions;

Octavia : provides load balancing as a service, distributing traffic across application instances to improve availability and scalability;

Keystone : service responsible for authentication, authorization, and the service catalogue across OpenStack;

Glance : service that stores, discovers, and retrieves VM images and related metadata;

Given the variety of modules orchestrated by OpenStack, it provides an ideal use case for evaluating the BDP infrastructure, particularly due to the complexity of handling and correlating logs generated in heterogeneous formats and with diverse contents.

4.1 Managing OpenStack Logs

The correct configuration of Logstash filters is a critical step in ensuring effective log parsing and data normalization within the processing pipeline. Since logs originate from heterogeneous systems and are often unstructured or semi-structured, accurately extracting meaningful information requires carefully designed parsing rules. A large amount of these rules were implemented using Regex-based grok filters, allowing specific portions of log entries to be identified, extracted using predefined or custom patterns, and mapped to named, searchable and query-able fields.

In Listing 1 there is an example of filter configuration: the filter processes logs tagged with `farm`, applying both field normalization and conditional parsing based on the log source. Initially, a mutate

PoS(ISGC2026)012

```
filter {
   if "farm" in [tags]{
     mutate {
       add_field => {"index_prefix" => "farming-cloud" }
       remove_field => ["[event][original]"]
       remove_field => ["agent"]
       remove_field => ["[log][offset]"]
       remove_field => ["[input][type]"]
     }
  }
  [...]
   else if "messages" in [log][file][path]{
     grok {
       patterns_dir => ["/etc/logstash/patterns"]
       match => { "message" => "%{MSGSERVICE:service}(?:\[%{NUMBER:[log][pid]}\])?: %{GREEDYDATA:[log][msg]}" }
     }
     if "nova-compute" in [service][1]{
       grok {
          match => {
            "[log][msg]" => [
              "%{TIMESTAMP_ISO8601:nova_timestamp} %{NUMBER:nova_pid} %{LOGLEVEL:loglevel} %{DATA:[log][module]}
              ↪  \[%{DATA:[log][context]}\] %{GREEDYDATA:tempmsg}"
            ]
          }
       }
       mutate {
         remove_field =>["nova_timestamp"]
         remove_field =>["loglevel"]
         remove_field =>["nova_pid"]
         rename => {"tempmsg" => "[log][msg]"}
       }
     }
   }
   [...]
}
```

Listing 1: Part of a Logstash filter to select and parse logs coming from the Nova service inside the OpenStack suite.

filter enriches the events by adding an `index_prefix` field (set to `farming-cloud`) and removes unnecessary metadata fields to reduce noise and storage overhead. For logs coming from messages files, the filter uses a custom pattern directory to identify the service that generated the event, an optional process ID, and the remaining log content. If the detected service is `nova-compute`, a second grok filter is applied to further parse the OpenStack Nova log structure, extracting fields such as timestamp, process ID, log level, module, context, and message body. Temporary or redundant fields are then removed or renamed, producing cleaner and more structured events for indexing and analysis.

```
Mar 13 17:46:58 tb-cloud-ctrl01 glance-api[1417995]: 2026-03-13 17:46:58.666 1417995 INFO eventlet.wsgi.server
[None req-49200678-a4d7-4afa-a03f-1317f7f4a987 a0f49fc452aa425880c30cdd0776f6c7 48b7db29a41f4c99b5b75c055c64abda
- - default default] 192.168.160.13,127.0.0.1 - - [13/Mar/2026 17:46:58] "OPTIONS / HTTP/1.1" 200 99
0.001266#033[00m
```

Listing 2: Example of log line produced by the Glance service of the OpenStack suite.

Using this kind of filter configurations it is possible to go from a raw log line like the one in Listing 2, to a completely parsed log displayed in OpenSearch, as shown in Figure 4: all important fields and information contained in the unprocessed text are condensed into pairs of field name and values. This allows to use the advanced search and query features of OpenSearch Dashboards, increasing the efficiency of tracking faults or anomalous behaviours.

PoS(ISGC2026)012

@timestamp	Mar 13, 2026 @ 17:46:58.788
@version	1
_id	D8cY6JwBtkMZTF5rISUX
_index	farming-cloud-2026.03.13
_score	-
_type	-
ecs.version	8.0.0
host.name	tb-cloud-ctrl02.cr.cnaf.infn.it
index_prefix	farming-cloud
log.bytes	99
log.client_ip	192.168.160.13
log.domain	default
log.file.path	/var/log/messages
log.global_request_id	None
log.http_version	1.1
log.level	INFO
log.local_request_id	req-4c7e35ee-f12f-489f-ba87-5ebe1c009454
log.loopback_ip	127.0.0.1
log.method	OPTIONS
log.module	eventlet.wsgi.server
log.msg	192.168.160.13,127.0.0.1 - - [13/Mar/2026 17:46:58] "OPTIONS / HTTP/1.1" 200 99 0.001226#033[00m
log.pid	4008554
log.project_domain	default
log.project_id	a6c92fb8fa2942a39a88c29f5c52ddef
log.projectname	-
log.request	/
log.status	200
log.t_response	0.001226
log.user_id	a0f49fc452aa425880c30cdd0776f6c7
log.username	-
message	Mar 13 17:46:58 tb-cloud-ctrl02 glance-api 9fc452aa425880c30cdd0776f6c7 a6c92fb8fa294 0.001226#033[00m
service	messages, glance-api
tags	farm-cloud

Figure 4: Completely parsed log line coming from the Glance service of the OpenStack suite. Every relevant element of the log is stored in an appropriately named field to improve searchability and open up the possibility of creating custom dashboards.

Building on this structured representation, the availability of well-defined fields enables more advanced data exploration and visualization capabilities. Once logs are parsed and indexed in this form, they can be easily aggregated, filtered, and correlated across different services and time ranges. This structured approach is fundamental for exploiting the full potential of OpenSearch, as it allows users to move from simple log inspection to higher-level analysis.

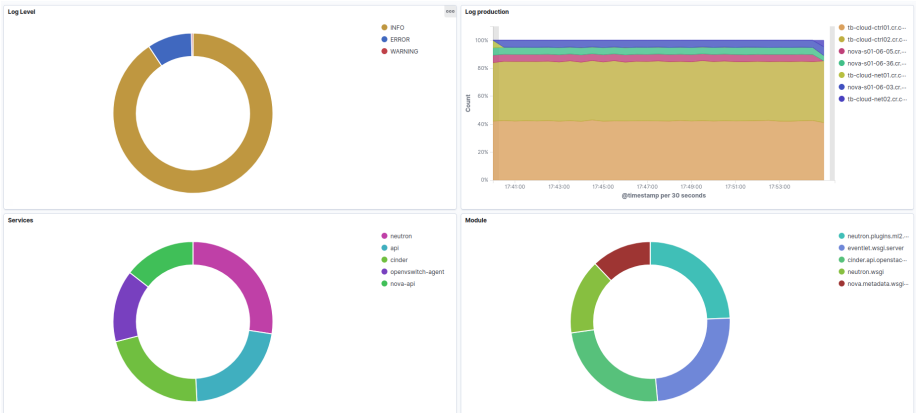

Figure 5: Simple dashboard available using OpenSearch Dashboards. The different kinds of graphs and plots are possible only with parsed logs.

In particular, the extracted fields can be used to construct interactive dashboards that provide a comprehensive view of the system behaviour. Metrics such as request rates, error distributions, or service-specific activities can be visualized through charts and graphs, facilitating rapid identification of anomalies and performance trends. An example of such visualization is shown in Figure 5, where parsed log data are used to generate meaningful insights through the OpenSearch Dashboards interface.

PoS(ISGC2026)012

4.2 Grok operational considerations

Grok provides the flexibility required by OpenStack services whose textual formats are not completely uniform, but this flexibility has an operational cost. Patterns must track changes in upstream log formats, and a format change can cause an otherwise valid event to stop matching the expected expression. For this reason, custom patterns should be kept service-specific and tested against representative samples whenever service versions or logging configurations change. Parse-failure tags can also be monitored so that format regressions are detected rather than silently producing partially structured events.

Regular-expression parsing is also more CPU-intensive than delimiter-based tokenization, especially when patterns become complex or contain very permissive expressions. The present deployment has not yet been subjected to a dedicated parser CPU benchmark, so no quantitative overhead is claimed here. Where a log format has a stable delimiter structure, the Logstash Dissect filter is a simpler and faster alternative because it does not use regular expressions [17]. A hybrid pipeline is also possible, using Dissect for the stable portion of a record and Grok only for the variable portion. Finally, when a service can emit structured JSON directly, structured logging is preferable because it reduces both parser complexity and maintenance effort. Grok remains necessary for legacy and heterogeneous text logs that cannot be changed at the source.

5. Conclusion

CNAF currently operates two complementary monitoring paths with different maturity levels. Sensu, InfluxDB, and Grafana form the established production service for numerical metrics. The BDP log-analysis path extends observability with Kafka-based buffering, Logstash parsing, and OpenSearch/OpenSearch Dashboards, but the OpenSearch component is still in pre-production for the OpenStack environment.

The Big Data Platform (BDP) has been under development at CNAF for several years, and in this pre-production state has demonstrated its effectiveness in collecting and processing logs from a large and heterogeneous set of services and servers. Its modular architecture, based on a distributed pipeline of open-source components, has proven to be flexible and scalable, allowing seamless integration of new data sources and services. In particular, the adoption of Logstash for log parsing enabled the transformation of complex and unstructured OpenStack logs into a structured format suitable for indexing and analysis within OpenSearch, significantly improving the observability of the Cloud infrastructure.

The results presented highlight how the combination of centralized log collection, structured parsing, and advanced visualization tools can enhance monitoring capabilities and facilitate the identification of faults and anomalous behaviours in a complex computing environment. The ability to correlate data across different services represents a key advantage for large-scale data centres such as CNAF. At the same time, the OpenStack case highlights an important maintenance trade-off: Grok is flexible for irregular formats but requires pattern maintenance and can add CPU cost, whereas Dissect or structured logging are preferable when the source format permits them.

Future developments will focus on extending the analytical capabilities of the platform by integrating Machine Learning techniques aimed at automated anomaly detection and failure prediction.

PoS(ISGC2026)012

In particular, approaches based on Autoencoders will be explored for the analysis of numerical metrics, while Large Language Models (LLMs) will be investigated for extracting insights from textual log data. Additional efforts will be devoted to increasing the number of systems feeding data into the BDP, further improving coverage and observability, as well as enhancing the underlying hardware infrastructure to support growing data volumes and computational demands.

References

[1] CNAF Website. https://www.cnaf.infn.it/en.

[2] Sensu Website. https://www.sensu.io.

[3] Influxdb Website. https://www.influxdata.com/.

[4] Grafana Website. https://grafana.com/.

[5] Filebeat Website. https://www.elastic.co/beats/filebeat.

[6] Kafka Website. https://kafka.apache.org.

[7] Logstash Website. https://www.elastic.co/logstash.

[8] OpenSearch Website. https://opensearch.org/.

[9] OpenSearch Dashboards Website. https://opensearch.org/platform/opensearch-dashboards.

[10] T. Diotalevi, D. Bonacorsi, A. Falabella, L. Giommi, B. Martelli, D. Michelotto et al., *Collection and harmonization of system logs and prototypal Analytics services with the Elastic (ELK) suite at the INFN-CNAF computing centre*, *PoS* **ISGC2019** (2019) 027.

[11] L. Giommi, D. Bonacorsi, T. Diotalevi, L. Rinaldi, L. Morganti, A. Falabella et al., *Towards Predictive Maintenance with Machine Learning at the INFN-CNAF computing centre*, *PoS* **ISGC2019** (2019) 003.

[12] S. Rossi Tisbeni, D. Cesini, B. Martelli, A. Carbone, C. Cavallaro, D.C. Duma et al., *A Big Data Platform for heterogeneous data collection and analysis in large-scale data centres*, *PoS* **ISGC2021** (2021) 008.

[13] OpenSearch Project, "Opensearch faq." https://opensearch.org/faq/.

[14] OpenSearch Software Foundation, "Opensearch software foundation." https://opensearch.org/foundation/.

[15] Leonardo Supercomputer Website. https://leonardo-supercomputer.cineca.eu.

[16] OpenStack Website. https://www.openstack.org.

[17] Elastic, "Logstash dissect filter plugin." https://www.elastic.co/docs/reference/logstash/plugins/plugins-filters-dissect.

PoS(ISGC2026)012