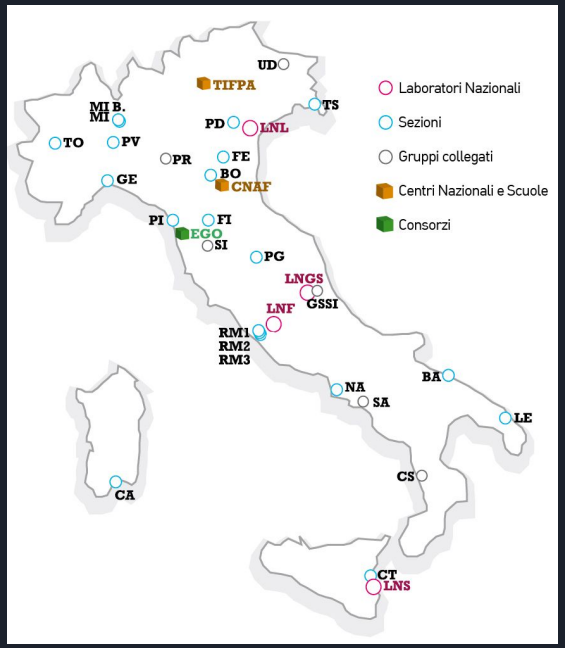
A Big Data Platform for Scalable Monitoring and Analytics at INFN-CNAF

Dr. Marco Lorusso, PhD - INFN, CNAF

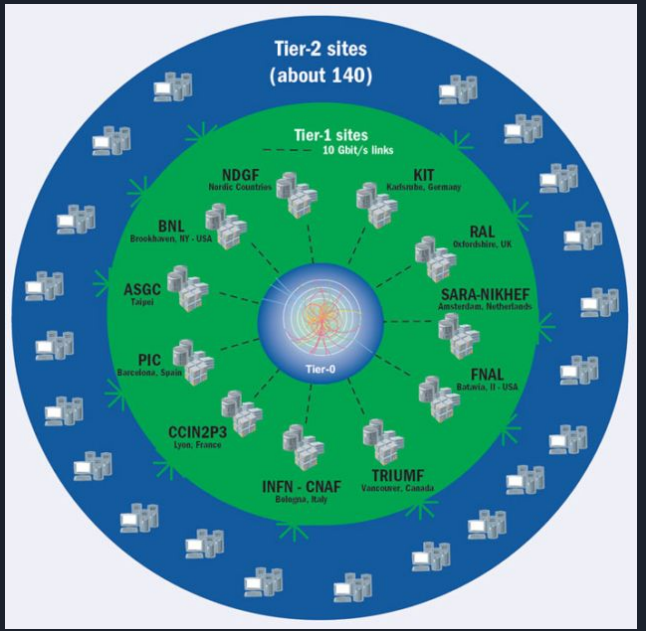
INFN-CNAF

- CNAF is the national center of INFN (Italian Institute for Nuclear Physics) dedicated to Research and Development on Information and Communication Technologies;
- It is involved in the management and evolution of the most important information and data transmission services in Italy;
- Since 2003, CNAF has hosted the Italian Tier-1 data center for the HEP experiments at the LHC in Geneva, providing the resources, support and services needed for
    - data storage and distribution
    - data processing and analysis
    - Monte Carlo production.



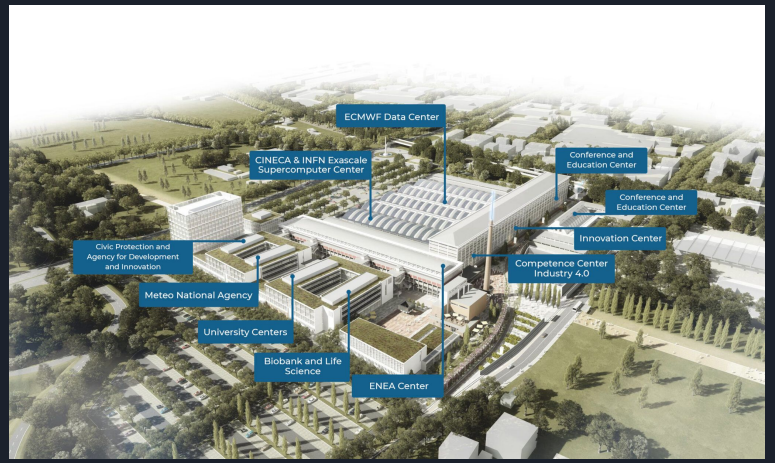
The Worldwide LHC Computing Grid @ CNAF

- The WLCG is a global computing infrastructure providing computing resources to store, distribute and analyse the data generated by the LHC;
- Tier structure with the CERN data centre as Tier-0:
  - INFN-CNAF hosts one of the Tier-1s
- Key challenge:
  - modernization of the center to be able to cope with the increasing flux of data expected in the near future e.g. with the new phase of operations of the High-Luminosity LHC



The new Datacenter @ Tecnopolo

- The hub of the Emilia-Romagna "**Data Valley**" in Italy
    - complex innovation ecosystem, based on research, businesses and expertise
- It houses **Leonardo**, the **10th SuperComputer** in the top 500 (Nov. 2025), with 241.20 PFlop/s!
- But also the **new and upgraded CNAF Datacenter**.

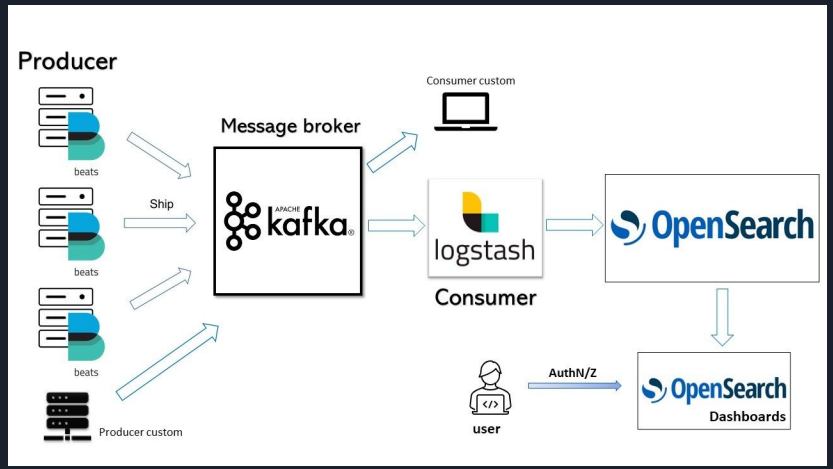


Why a Big Data Platform

- At CNAF scale, the operational data flow is itself something that must be collected and analyzed;
- The Tier-1 has a direct connection to WLCG Tier-0 with a guaranteed minimum bandwidth of 10 Gbps;
  - A dedicated system is needed to collect, store, and monitor this data flow
- A centralized log platform simplifies monitoring and maintenance;
  - Supports researchers needing access to job reports redirected to CNAF resources.

BDP architecture

- The Big Data Platform is deployed across different machines as a modular pipeline, with separated components, from producers to consumers:
- Data producers: Filebeat
- Message broker: Apache Kafka
- Data consumer / transformation: Logstash
- Storage and analytics: OpenSearch
- Visualization: OpenSearch Dashboards



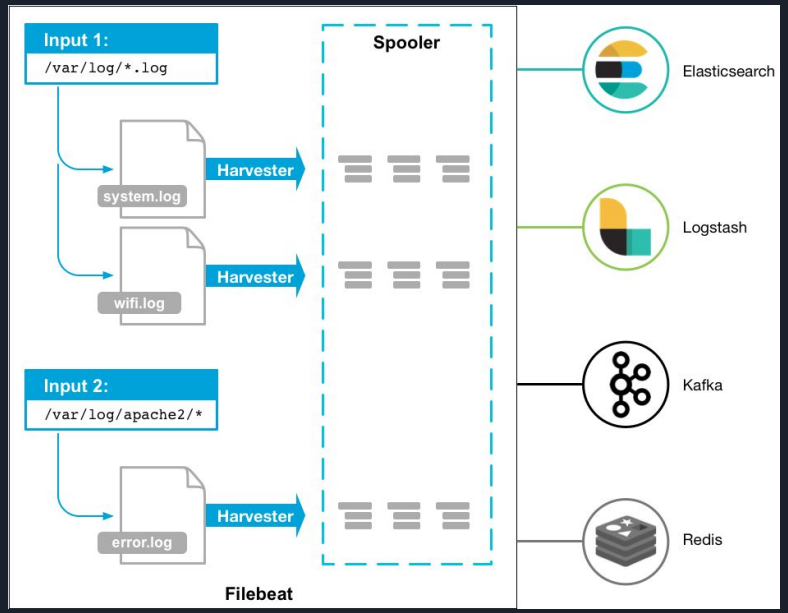
Filebeat

Filebeat is a lightweight shipper for forwarding and centralizing log data:

- Installed as an agent on servers;
- Monitors the log files or locations specified;
- Collects log events, and forwards them either to Elasticsearch or Logstash for indexing.

How it works:

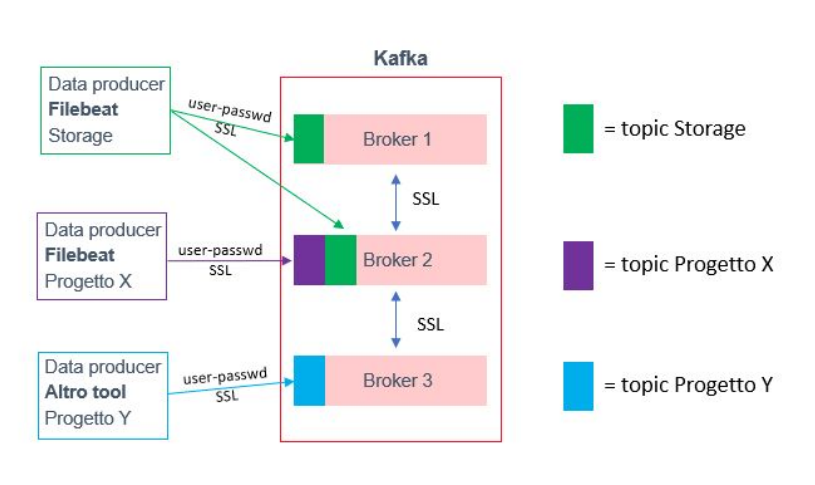
- It starts one or more inputs that look in the locations specified for log data;
- For each log found, it starts a harvester which reads a single log for new content and sends the new log data to libbeat
  - which aggregates the events;
  - sends the aggregated data to the configured output



Apache Kafka

Apache Kafka is a distributed event streaming platform used to build real-time data pipelines and event-driven applications.

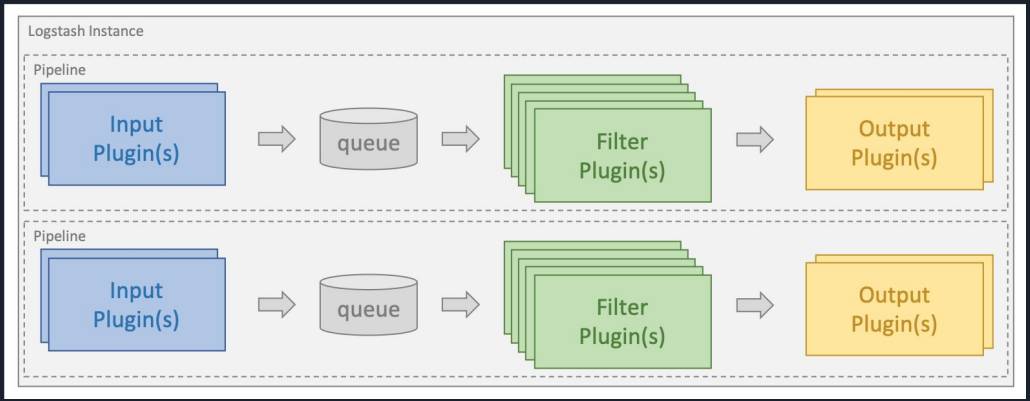
- Enables systems to publish, store, and process streams of data with high throughput and low latency.
- Uses a publish–subscribe model with producers and consumers;
- Data is organized into topics across a cluster of brokers;
- Provides scalability, fault tolerance, and durable storage;



Logstash

Logstash is an open source data collection engine that collects, transforms, and sends data to various destinations

- Can dynamically unify data from disparate sources;
- Processes and transforms data using filters and plugins;
- Sends data to outputs such as databases or message queues;
- Supports real-time data processing



Opensearch

OpenSearch is an open-source search and analytics engine used to store, search, and analyze large volumes of data in real time.

- Distributed architecture for scalability and high availability
- Supports full-text search and real-time analytics

- Integrates with data pipelines
- Includes OpenSearch Dashboards for visualization and monitoring

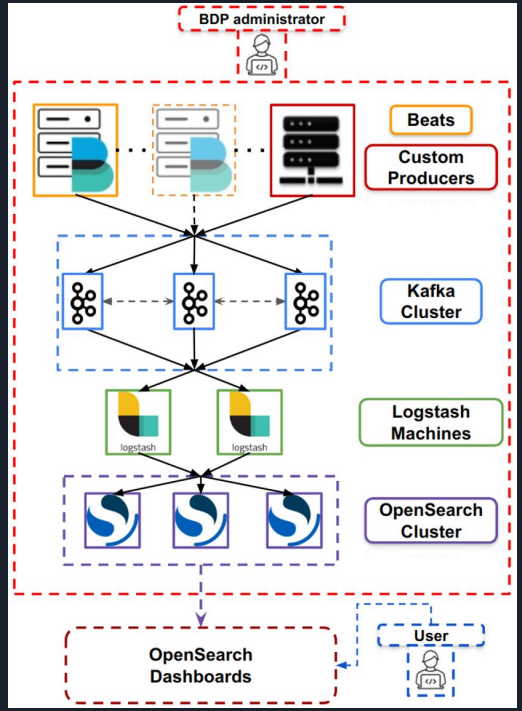
Opensearch Dashboard

OpenSearch Dashboards is a web-based visualization and analytics interface for OpenSearch. It allows users to explore, analyze, and visualize data stored in OpenSearch through interactive dashboards and charts.

- Provides data visualization through charts, graphs, and maps
- Enables interactive dashboards for monitoring and analysis
- Supports searching and filtering data in real time
- Used for log analysis, observability, and security monitoring

How data move through the pipeline

1. Filebeat forwards logs from selected locations on admin VMs;
2. Kafka acts as a buffer, implemented as a three-machine cluster, to avoid congestion;
3. Logstash consumes the queued data and reshapes them into compact JSON for indexing and later analysis;
4. OpenSearch Dashboards exposes the indexed data through interactive web dashboards.



Custom Producer and/or Consumer allowed if needed

Security and operating model

- Access to dashboards is limited to authorized users;
- The web interface is available only through the local CNAF network;
- Inter-host communication is mediated by X509 security certificates;
- The health of the platform is continuously monitored by CNAF administrator teams;
- Only administrators can directly access the underlying machines hosting the pipeline components.

BDP for Computing Services

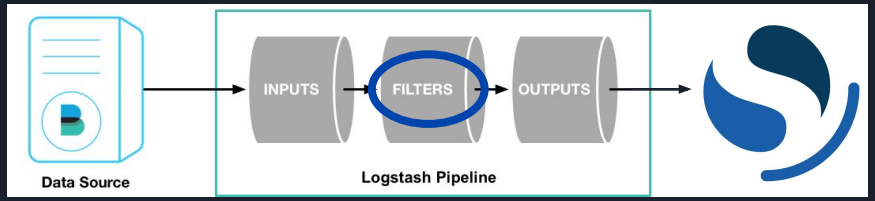
Mainly OpenStack core services:

- 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 catalog across OpenStack;
- Glance: service that stores, discovers, and retrieves VM images and related metadata;
- And many others… (e.g. Barbican, Placement, Horizon)

Parse Logs using Logstash

Choosing filters for correct log parsing is crucial.

A large amount of them implemented using grok filters:



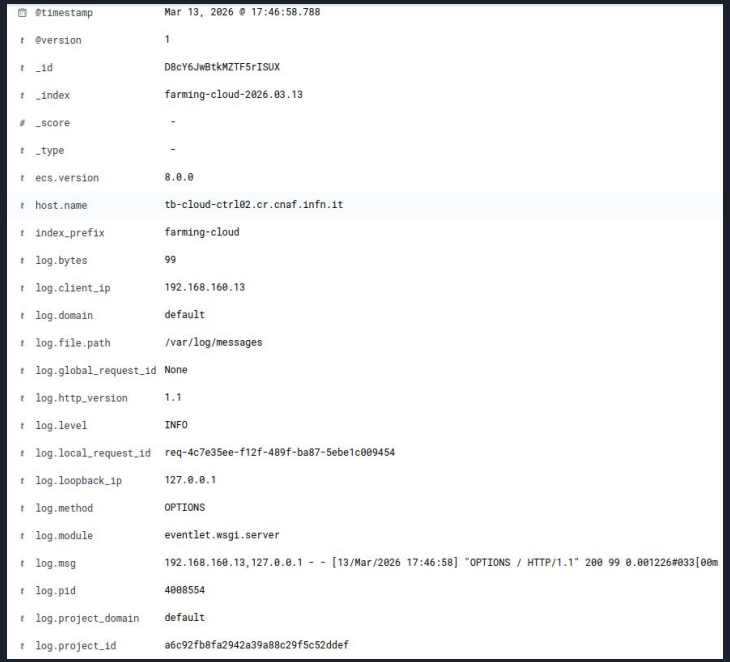
- Based on Regular Expression (Regex)
  - Specific portions of log entries are selected using predefined or custom patterns

```
      grok {
        match => {
          "tempmsg" => "%{IP:[log][client_ip]},%{IP:[log][loopback_ip]} - - \[.*?\] \"%{WORD:[log][method]} %{URIPATHPARAM:[log][request]} HTTP/%{NUMBER:[log][http_version]}\" %{NUMBER:[
log][status]} %{NUMBER:[log][bytes]} %{DATA:[log][t_response]}#"
        }
      }
```

- Other filter plugins available, e.g. *Ruby* scripts, *Date* (specific for timestamps), *KV*, *Split*, …

Example of parsed logs

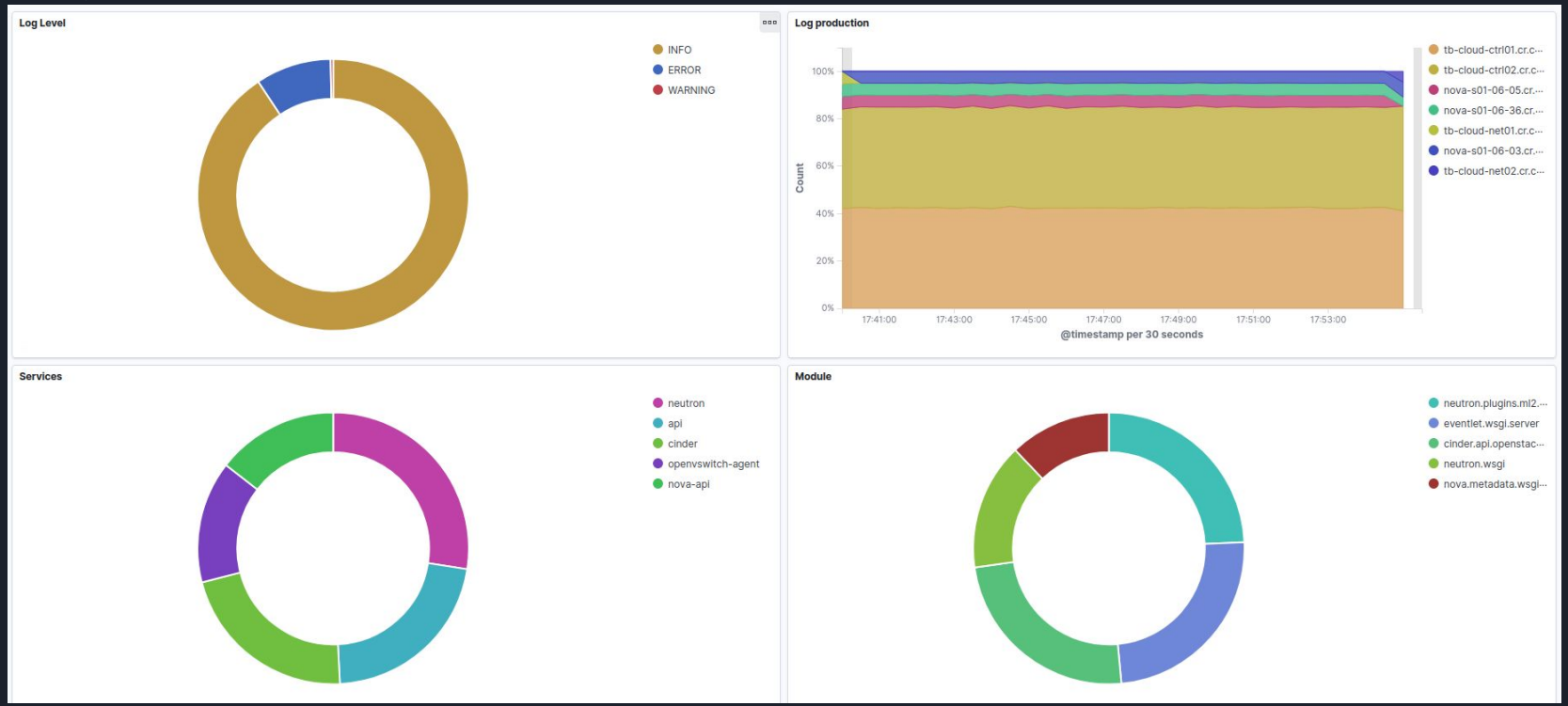
```
Mar 13 17:46:58 tb-cloud-ctrl01 glance-api[1417995]: 2026-03-13 17:46:58.666 1417995 INFO eventlet.wsgi.server
[None req-49200678-a4d7-4afa-a03f-1317f7f4a987 a0f49fc452aa425880c30cdd0776f6c7 48b7db29a41f4c99b5b75c055c64abda
- - default default] 192.168.160.13,127.0.0.1 - - [13/Mar/2026 17:46:58]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



@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

Example of Dashboard



Conclusion

- The BDP infrastructure has been active for several years, collecting logs from numerous servers managed by CNAF;
- Deployed across different machines, consists in a modular pipeline of different components;
- Through the usage of Logstash, logs related to OpenStack were parsed and indexed effectively.

Future Developments:

- Implementation of ML to investigate failed jobs and seeking anomalies (see my other talk), e.g.:
  - Autoencoders for numerical metrics;
  - LLM for textual logs;
- Increase machines injecting logs into BDP;
- Improve hardware foundation of the BDP.

Thank you!