eXtreme monitoring: CERN video conference system and audio-visual IoT device infrastructure

Ruben Gaspar Aparicio[1], and *Theo* Soulie[1]

[1]European Organization for Nuclear Research (CERN), CH-1211, Genève 23, Switzerland

E-mail: ruben.gaspar.aparicio@cern.ch, theo.soulie@cern.ch

Abstract. Two different use cases for monitoring are analysed in this paper: the CERN video conference system – a complex ecosystem, which is being used by most HEP institutes, together with Swiss Universities through SWITCH; and the CERN Audio-Visual and Conferencing (AVC) environment – a vast Internet of Things (IoT), which includes a great variety of devices accessible via IP. Despite the differences between both use cases, a common set of techniques underpinned by IT services is discussed in order to tackle each situation.

1 Introduction

Video conference and Audio-Visual Conferencing services are part of the CERN IT department's Collaboration, Devices & Applications (CDA) group.
The Video conference service has more than 10.000 active users scattered around the globe and a server infrastructure that is also spread across the CERN data centre in Switzerland, and various other data centers on different continents (see Figure 1). The video conference infrastructure provides on average more than 900 video meetings per day, lasting on average more than 1 hour with peaks of attendees in a single meeting, reaching slightly more of 500 users using desktop or a mobile device, or even on phone call via Public Switched Telephone Network (PSTN) or Voice over IP (VoIP). The maximum number of simultaneous users ever reached 1125. The software is based on Vidyo [1], a video conference solution.

The CERN AVC IoT is made of hundreds of devices e.g.: projectors, screens, clocks, controllers, microphones, encoders, etc. all of those bringing a variety of models and different vendors, despite attempts to keep variety as functional and minimal as possible. Due to legacy and evolution of the products it is nearly impossible to avoid a wide ecosystem of devices.

The monitoring techniques used for these two use cases treat each system as data abstraction. By doing so the actual system is quite irrelevant; what matters is the model of data used and how to get the data into the database, either nosql or time series in our setup. A clear understanding of what needs to be monitored, e.g. the abstraction model should be known or come out after an initial data collection phase. Sometimes this is not trivial due to the lack of a proper interface or API. In general AVC IoT has been controlled by proprietary protocols, noticing a shift to IP and standards in the last years. This is especially the case for the transmission of audio and video, AV/IP (AV over IP). Many AV vendors are now moving away from traditional AV setups using protocols, such as HDMI, SDI, HDBaseT in favour of AV over IP protocols like JPEG-2000 or H.264. By doing so, they achieve a more open and flexible infrastructure, as IP technologies bring the flexibility of IP switches in order to extend/connect more devices and the possibility for longer distances at a lower cost than when using traditional AV cabling, where extensions and long distances become rapidly unaffordable.

IP technologies are also based on standards; therefore interoperability is greatly improved. The major concern for moving away from circuit-based traditional AV systems is security. Although this paper does not address this concern, issues such as encryption, control access on applications and protocols together with segmented or access protected networks, should be part of the design of an AV over IP system.

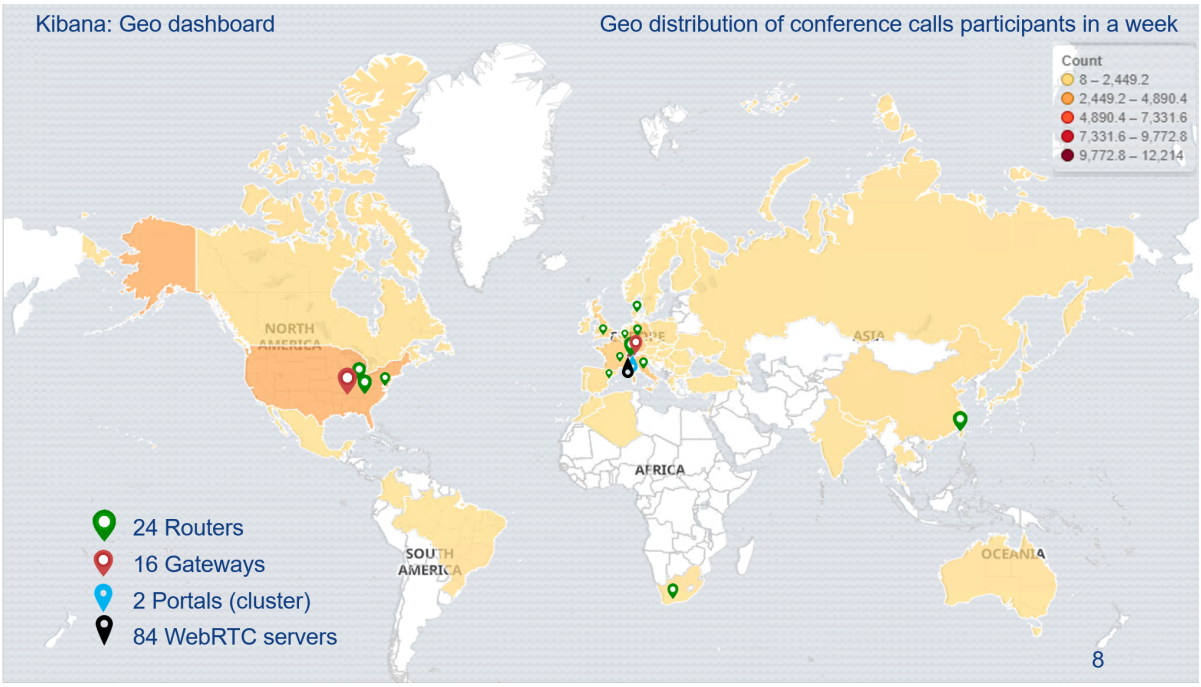

Figure 1. On premise infrastructure worldwide and geolocation of attendees using Vidyo during a week time span, mostly coming from Switzerland.

In the following paragraphs a general trend of the technical choice will be explained, exploring examples that apply to both use cases.

2 General description of the chosen technologies: containers, nosql and time series

Kubernetes 10 or Openshift 11 are used to run the programs that collect data. Both orchestration engines offered the same level of automation for the application deployment, scaling and management. Both are also provided by the CERN IT central services. Generally speaking, both monitoring engines run on Kubernetes, however Openshift also provides some add-ons, like a HAproxy for load balancing or a domain SSL certificate for web portals.

The arrival of nosql databases has made it easy to ingest non-normalized data and still manage to index it and extract information from it. Among all possible variants of nosql databases, Elasticsearch 2 was chosen for two reasons. First, it offers a full stack of functionalities from data collection to visualization and it is document oriented; second, there is a central CERN IT service that provides instances of Elasticsearch.

Working with document-based databases/storage systems like Elasticsearch requires in advanced preparation of all data that should be contained in the document. In case of the Vidyo Call Detail Records (CDR) database, running on MySQL, data is gathered using normal standard Structured Query Language (SQL), where joins are performed and the result, after further formatting, is ingested in Elasticsearch. Formatting can be done programmatically or following the Elasticsearch stack, using both a beat like Filebeat 3 or one step further at Logstash 4 (see Table 1).

Table 1. Using the extraction power of SQL to create NoSQL documents with all information needed.
SQL: select joining two tables from Vidyo CDR database

```
select a.callID, b.Name as RouterName, ..
       from portal2.ConferenceCall2 a INNER JOIN portal2.components b
            ON a.RouterID = b.COMP_ID
       where EndpointType <> 'L' and a.CallState='COMPLETED'…
```

NoSQL: JSON document collecting data of both tables

```
{
     "_index": "collaborativeapps6-vidyocon-2020.01.22",
     "_type": "collaborativeapps6-vidyocon",
     "_id": "6950320",
     "@timestamp": "2020-01-22T17:04:33.000Z",
     "callID": "6950320",
     "RouterName": "RouterKIT",
     "geoip": {
         "longitude": 34.75,
         "latitude": 31.5,
      ...
}
```

This will result in more data and, therefore, possibly more space consumption on the backend. It is important to note that in Elasticsearch, there are also ways to implement this type of relationship, however, it is recommended to de-normalize data, as its implementation and performance while querying is costly.

Last but not least, the Elasticsearch stack comes with a visualization component called Kibana 5. Visualizations can be collected in dashboards, and adding new ones from third open-source parties is possible.

InfluxDB stack 6 is used for metrics data. InfluxDB is a specialised database for time series. Computer metrics data e.g. CPU, memory, disk space, etc. are well suited for this type of database. InfluxDB also offers a stack to easily ingest data via Telegraf, an agent deployed on servers 7. Data is stored in an optimised way almost automatically. It was only required to set the desired retention policies for the data, which in our case is one year. Visualization dashboards are implemented using Grafana 8, an open-source tool to visualize time series of different sources, including InfluxDB and Elasticsearch. Grafana offers a great feature, the possibility to generate alarms from a given time series statistic and to connect those alerts to collaboration tools like email or Mattermost 9, an open-source Slack-like software.

3. CERN video conference use case: Vidyo

Vidyo on-premise monitoring tools for its conference system are not sufficient. For the time being, running the version 19.1 on the portal, only the up/down status on behalf of connectivity to the Vidyo Manager or availability of the server is displayed. When a server component appears online (green arrow up), it does not necessarily mean there are no issues with it. This situation was qualified as critical because it did not allow having an understanding of the status of the video conference system or its utilization.

The following criteria were applied when designing a monitoring system for the Vidyo platform. 1) It should be non-intrusive and should not affect the platform's operation. 2) It should be as easy as possible for service managers to maintain it, therefore we rely as much as possible on the IT central services, compliance with the European Union General Data Protection Regulation (EU GDPR) 12, since sensitive data, such as IP and user names, may be logged. It is based on getting essential information, not necessarily beautiful but effective, and it should provide the right audience with the right information.

The right audience consists of two groups: second-level supporters who require information of online meetings, and IT management that needs to have an overview of the system in big numbers and statistics over an extended period of time. The collector architecture can be seen in Figure 2.

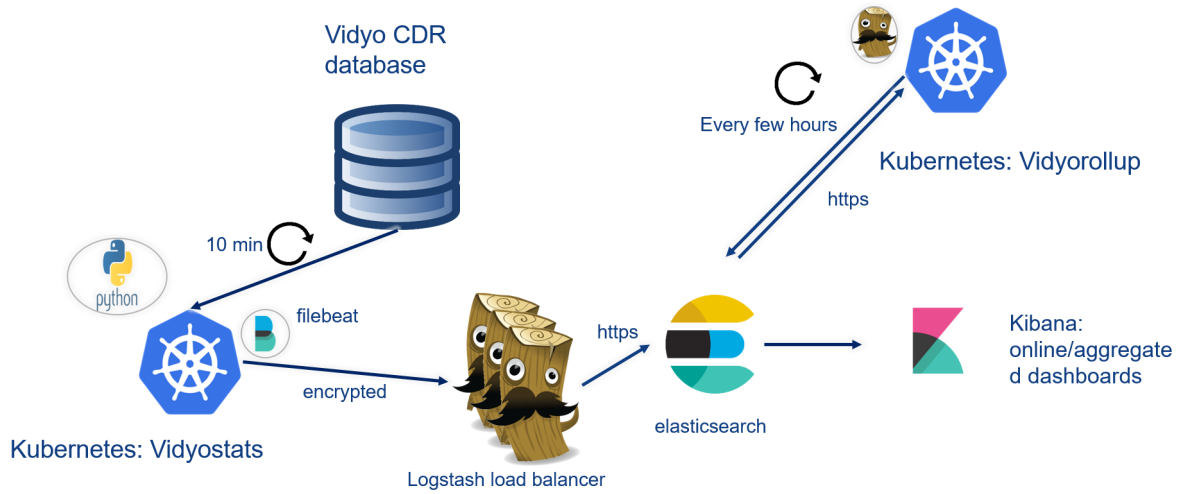

Figure 2. Vidyo stats gathering architecture.

The first Kubernetes cluster is called Vidyostats [13]; it queries Vidyo CDR database every 10 minutes and provides online information about the video conference, the users connected and how different server components are used. It is a python program deployed into a Kubernetes cluster together with a Filebeat container. The second Kubernetes cluster is called Vidyorollup [14], which periodically runs queries on the system to aggregate data. Those are Logstash and a Filebeat containers. Vidyorollup will log in different Elasticsearch indexes, much smaller in size, that are intended to be kept for longer periods. Vidyostats indexes will not last longer than three months; they are much more bulky as they contain raw data.

The Logstash component is built up by several Logstash servers under a load balancer alias. This component is also reused for the order service of the IT CDA group. Logstash has a function of doing basic calculations like matching a log off/disconnection from a particular user, in order to calculate the duration of its participation on a video conference (see Table 2). Data manipulation is done on Logstash to spare CPU and memory on the Vidyo servers.

Table 2. Sample of logstash configuration.

```
if ([CallState] == "COMPLETED") {
 …
   ruby {
     id => "vidycon_ruby_session_duration_min"
     code => "
       require 'date'
       login_time = DateTime.strptime(event.get('JoinTime'), '%Y-%m-%d
%H:%M:%S')
       end_time = DateTime.strptime(event.get('LeaveTime'), '%Y-%m-%d %H:
%M:%S')
       if login_time && end_time
         event.set('sess_duration_min', ((end_time - login_time) * 24 *
60).to_i
       else
         event.set('sess_duration_min', 0)
       end
       "
   }
…
}
```

Vidyostats will lead to online data. Dashboards are focused to detect possible problems on a Vidyo server component or Vidyo video conference. In cooperation with the IT Elasticsearch central service, our Elasticsearch cluster was extended with a network visualization plugin 15 that allows having a visual representation on how the platform is being used (see Figure 3). Data is prepared at ingestion as indicated in the Network visualization plugin 15 by means of several joins (see Table 1), so that the visualization can be effective.

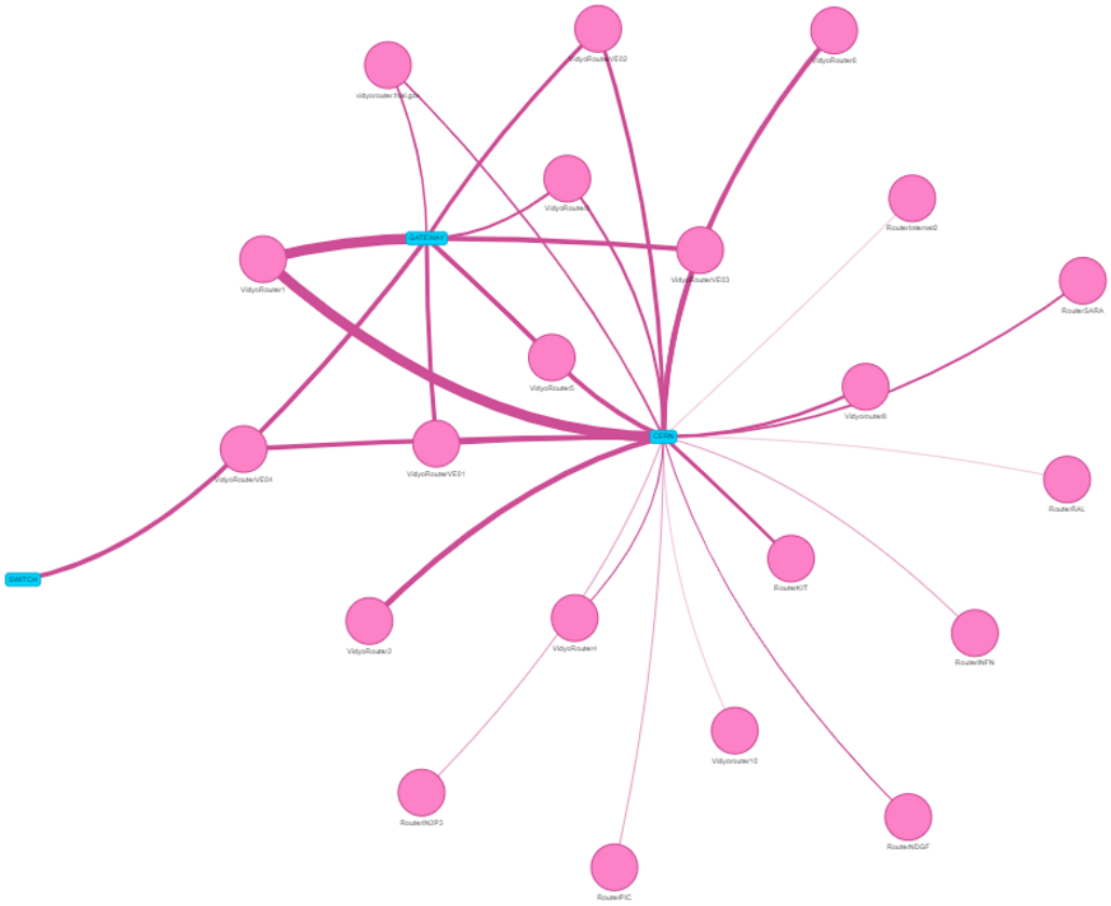

Figure 3. Online visualization of Vidyo server components utilisation: routers and gateways.

The second pillar of Vidyo monitoring is Metrics, that is collecting, displaying and alerting on specific metrics of Vidyo servers. Vidyo images are deployed on Ubuntu operating system. Any major release deployment will usually wipe out any customization done on the servers. It was unthinkable to always require a new manual redeployment of CERN customizations. Due to the type of image used by Vidyo and the distributed architecture, outside of CERN data centres, it was not possible to use the CERN configuration management infrastructure based on Puppet. Ansible 16 was chosen instead, as it allows managing all distributed servers with no need to deploy any agent on those, only via ssh connections. A view of the configuration management architecture can be seen in Figure 4.

The Ansible control node runs on the CERN data centre on a Centos virtual machine. It deploys software or configuration on all CERN Vidyo servers. A set of roles and playbooks have been developed in order to manage Vidyo servers. Nodes outside the CERN network need a proxy in order to access our instance of InfluxDB not

exposed to the Internet. This proxy has been implemented on Openshift, so that all external monitoring traffic gets conveyed to CERN via https.

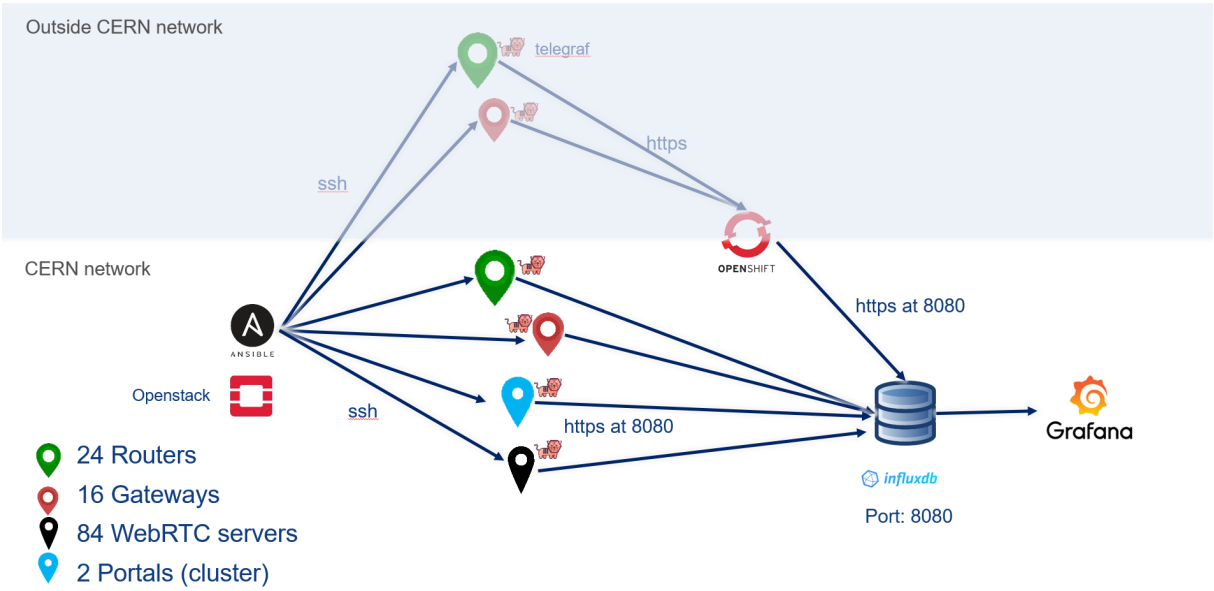

Figure 4. Deployment of software and configuration with Ansible.

Visualisation and alerting are done by Grafana, which is a quite standard implementation of the InfluxDB stack. Monitoring metrics, despite being a basic pillar in monitoring, have been extremely helpful in solving problems on remote servers, for example, an issue that was detected on US gateways in June 2019[1]. Monitoring and dashboards have been adapted to show also statistics from Vidyo processes e.g. vr router process.

4. Audio Visual IoT use case

Audio Visual and Conferencing (AVC) infrastructure is a complex ecosystem. With the arrival of IP in the AV world, new rules to extract information and to use the same IT techniques developed for different domains can apply. Ultimately, it's just a soup of data coming from very different devices, however, the same analysis to clean-up and make sense of the data can be used. Our architecture to monitor AV IoT can be seen in Figure 5.

[1] https://cern.service-now.com/service-portal/view-outage.do?n=OTG0050828

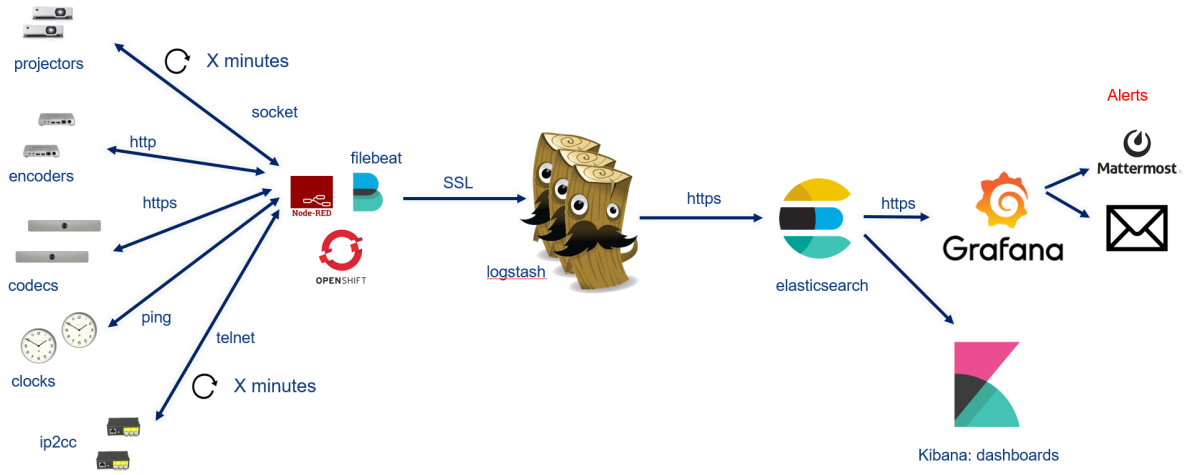

Figure 5. IoT architecture.

Basically, once the data has been collected from devices, it follows the same path as per the video conference monitoring use case. One of the key components of the architecture is the IBM OSS project Node-Red 17. By means of flows, a great number of devices are monitored. The system should run as autonomously as possible once it has been configured. Collecting new devices, from the types already configured, for example a new projector, is done automatically from our network database (see Figure 6).

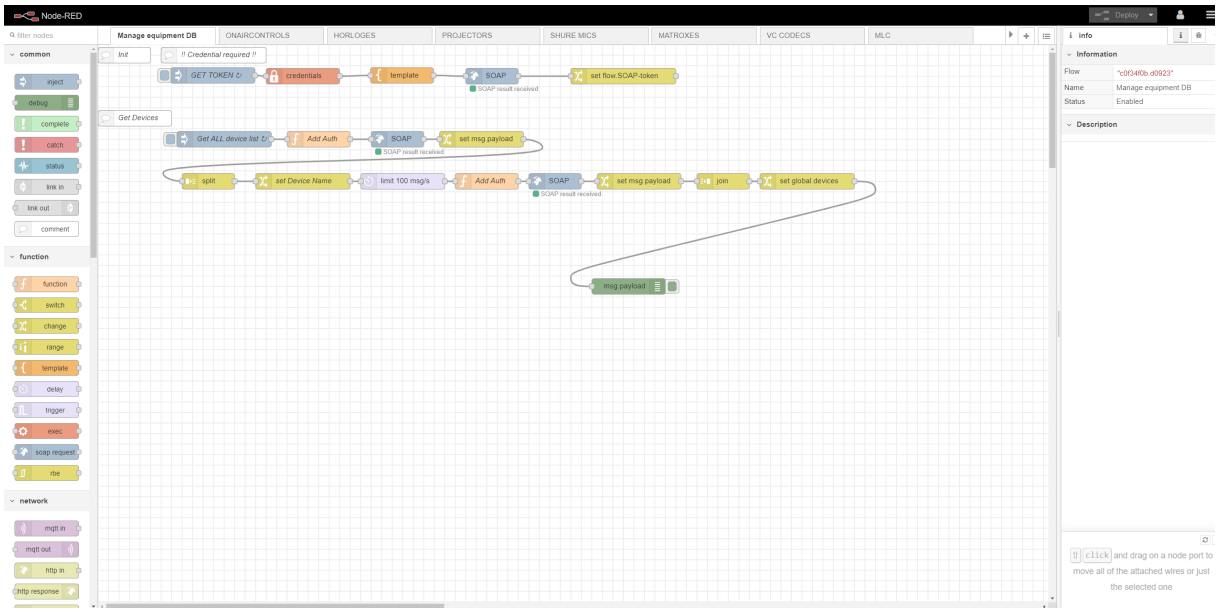

Figure 6. Node-red flow panel showing a flow to retrieve devices from our network database.

Data follows a set of nodes that perform different actions, such as providing authentication, connecting to the device, parsing information and, finally, logging that information on a file that will be sent to Elasticsearch. Messages following this structure inspired on-brokers messaging systems:

```
{"val":true,"ts":1580379416494,"lc":1580379416494,"topic":"0503/2/0007/get/
micreceiver/MICRECEIVER004/ping"}
{"val":[99,255],"ts":1580379416878,"lc":1580379416878,"topic":"0503/2/0007/
get/micreceiver/MICRECEIVER004/batteries"}
```

Parts of the message are sufficiently clear. `Val` can be a single value or an array, or just text. Further manipulation of those values can be done at the Logstash level. About the `topic`, it shows location of the device, i.e. building/floor/office, type of operation, type of device, the concrete instance of it, and finally, what is retrieved, the API/protocol call. Ultimately, Kibana and Grafana are used to display data as a time series or some stats derived from it, together with alerting, for example, projector lamps' lifespan coming to an end.

5. Conclusion

Two complex use cases from the perspective of monitoring have been discussed. Despite being quite different from one another, a set of servers distributed all over the world, running proprietary images for the video conference system, and a set of devices distributed all around CERN, its Geneva and France sites, the techniques shown here consider the systems as a soup of data, in which analysis and visualization need to be performed. Technologies used together with an approach on how to extract and manipulate data have been presented.

More work is being done to develop further the IoT monitoring infrastructure and to enrich it with data from other systems, like the CERN Indico [18], so that more information can be extracted from it than from a standalone system. The aim is to gain preventive knowledge of AVC constellation of devices, as well as to improve response to failure and usability for the CERN user community.

6. Acknowledgments

We thank all CERN IT services that make it possible for us to create new projects upon the Database on Demand, Elasticsearch service, Openshift & Openstack, and all the people running them. Special thanks go to the Serco colleagues for all their useful input.

References

1 Vidyo, https://www.vidyo.com/ bought on 15th May 2019 by Enghouse Systems Limited.
2 Elasticsearch stack, https://www.elastic.co/es/what-is/elk-stack
3 Elasticsearch, Filebeat, https://www.elastic.co/products/beats/filebeat
4 Elasticsearch, Logstash, https://www.elastic.co/products/logstash
5 Elasticseach, Kibana, https://www.elastic.co/es/products/kibana
6 InfluxDB, https://www.influxdata.com/time-series-platform/
7 InfluxDB, Telegraf, https://www.influxdata.com/time-series-platform/telegraf/
8 Grafana, https://grafana.com/
9 Mattermost, https://mattermost.com/
10 Kubernetes, https://kubernetes.io/
11 Red Hat Openshift, https://www.openshift.com/
12 GDPR European Union site, https://gdpr.eu/
13 Vidyostats project, https://github.com/CERNCDAIC/resthttpck
14 Vidyorollup project, https://github.com/CERNCDAIC/aggsvidyo
15 Network visualization plugin, https://github.com/dlumbrer/kbn_network
16 Ansible, https://www.ansible.com/
17 IBM Node-red, https://nodered.org/
18 Indico, open source tool for event organisation, archival and collaboration, https://getindico.io/