August 2021

Grafana for WinCC-OA based SCADA systems at CERN

AUTHOR(S):
Yash Kataria

SUPERVISOR(S):
Rafal Kulaga
Anthony Hennessey

CERN openlab

ABSTRACT

WinCC-Open Architecture is a toolkit for creating Supervisory Control and Data Acquisition (SCADA) applications. There are about 650 instances of WinCC OA deployed across the accelerator control systems and experiments at CERN. By default WinCC OA is only aware of the current state of a system; WinCC OA Archiving is used to keep a historical record of changes that occur to the state of the system.

The WinCC OA archiving solutions deployed at CERN, the legacy 'RDB Archiver' and its replacement the 'Next Generation Archiver' (NGA), archive to Oracle databases, and for the most part the design and configuration of these databases is consistent between archivers and across CERN. Monitoring and visualising the data archived to these databases is a vital task.

Grafana is a multi-platform open source analytics and interactive visualization web application. The goal of this project is to develop a Grafana Datasource that abstracts away the CERN specific Oracle database schema design and hence allows a Grafana user to intuitively retrieve WinCC OA archived data. Any proposed datasource solution must also meet performance and scalability requirements, and meet CERN security standards.

This report details the extensions and modifications to a prototype Grafana datasource to achieve the goals described above. We detail the progression of the work and the technologies and technical solutions used. Additionally we detail several deployment scenarios that address performance and security requirements as they relate to CERN infrastructure, and give details of work completed to automate this deployment.

TABLE OF CONTENTS

1. INTRODUCTION

WinCC Open Architecture is a toolkit for creating Supervisory Control and Data Acquisition (SCADA) applications. There are about 650 instances of WinCC OA deployed across the accelerator control systems and experiments at CERN. WinCC OA is only aware of the current state of the systems it is configured against; the addition of 'archiving' allows us to keep a historical record of changes that occur in these systems over time.

At CERN Oracle databases are used for archiving control data from both of the current WinCC OA archiving solutions, the legacy 'RDB Archiver' and its replacement the 'Next Generation Archiver' (NGA). Visualising the archived data is very important for tasks such as the analysis of historical events and monitoring system trends, and Grafana is a visualization tool that offers a rich set of functionality for presenting, in particular, this type of time series data.

The Oracle database schema used for archiving by WinCC OA contains a lot of complexity, and the existing Grafana datasource for querying Oracle schemas would expose this complexity to the user. By developing a custom Grafana datasource we are able to both leverage the functionality offered by Grafana, and to abstract away the complexity of the WinCC OA Oracle schema.

Figure 1 is a high level overview showing where the datasource fits into the overall architecture. Here, the Grafana server sends requests to a datasource backend which in turn returns relevant responses and data regarding alarms and events from the Oracle database.

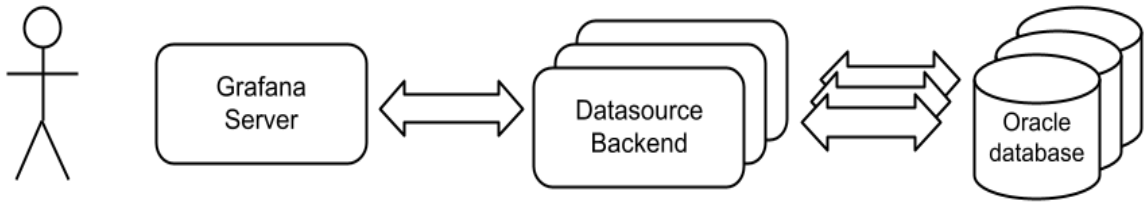

Fig. 1 Simplified Architecture for Oracle Datasource

The implementation of the architecture described above uses various technologies available to projects at CERN. The following is a brief description of each of these technologies:

- **Grafana** is an interactive visualization tool, here Grafana provides the interface to the user and as such is used to visualise the archived events and alarms data.
- **Flask** is a micro web framework and is used to develop the datasource in the form of a web application facilitating communication to the Grafana server.

- **simpod-json-datasource** is a third party Grafana plugin that is used to connect our custom datasource to the Grafana server.
- **Docker** is a set of operating system level virtualization tools that we use to build containerized versions of our backends.
- **OpenShift** is a family of containerization software products which will be used to deploy both Grafana server and our custom datasource at CERN. Openshift allows us to easily deploy our Datasource and scale it in an efficient and secure manner.
- **Gitlab CI** is used to automate the deployment process so that incremental changes can be easily tested and deployed.
- **SQLite** is an embeddable lightweight relational database management system which we use for in-memory caching.

2. ORACLE DATASOURCE FOR GRAFANA

This section describes the four main areas of contribution made to the 'Oracle datasource for Grafana' as part of this project. For the following description use the concept of 'datapoint elements' or DPEs. The DPEs represent the control signals, and a signal is an individual measurement for which changes are being archived; an example of a signal would be the current measured at a particular power junction.

a. CONFIGURING GRAFANA AND THE DATASOURCE

The open source version of Grafana does not support the Oracle database. In addition there is a lot of complexity in the Oracle schema used by WinCC OA archiving that we would like to abstract such that it is transparent to the user. Therefore, we implement a datasource designed to interact with the Oracle schema that is specific to WinCC OA archiving.

We utilise the third party plugin *simpod-json-datasource* that allows an arbitrary backend to connect to a Grafana server. However, the backend needs to implement some required endpoints. These endpoints are basically URLs which provide access to our backend. We use endpoints to return the data associated with particular DPEs and also to return metadata associated with the DPEs. Users can query events data or alarms data by choosing either time-series or table data format from Grafana for the intended DPEs.

The other problem was to abstract the schema definition from the user as we run SQL queries on backend and not from the Grafana side which is usually the case. For this purpose, we use Grafana variables. We have implemented three Grafana variables:

- `schema`: User can provide the schema to retrieve DPEs.
- `filter`: As the number of DPEs can be very high and may result in timeout, this variable allows access to a subset of DPEs from the specified schema.
- `metric`: This variable returns a list of DPEs based on schema and filter.

By using variables, users can also pick multiple DPEs and display the data for all the chosen DPEs at once. As more DPEs are queried, the queries take up more resources and hence we restrict the number of DPEs that can be queried simultaneously.

Fig. 2 Grafana Dashboard using our Datasource

b. IN-MEMORY DATABASE CACHING

The queries against the Oracle database are sometimes slow, especially with the large volumes of data experienced at CERN. Therefore any mechanism that we can implement to improve the performance of queries, or reduce the impact of slow queries, can significantly improve the user experience.

One set of queries we need to perform relates to fetching metadata. The metadata contains a description of the DPEs that are being archived. Metadata changes are infrequent and so these queries are a good candidate for caching.

We use a SQLite database as an in-memory database to offload requests which return DPEs. We achieve this by caching all of the metadata from Oracle database across all schemas. The SQLite database is updated by running cron jobs at a frequency dependent on use case.

In cases where metadata in the Oracle database is being updated frequently the SQLite database would not get updated with the latest meta-data in a timely manner and may return stale data. In these cases the system can be configured not to use metadata caching.

c. CI/CD PIPELINE FOR AUTOMATIC DEPLOYMENT

A continuous integration / continuous deployment (CI/CD) pipeline is a method for automating the series of steps that must be performed in order to build, test and deploy a new version of software. The proposed deployment scenarios (see section below) for this project involve multiple deployment steps related to containerization and the specifics of the CERN infrastructure; these steps are well suited to being automated using a CI/CD pipeline.

In brief: we build a docker container for our Datasource and register the image to gitlab's internal container registry. Openshift can pull this image and build the container and run it. We automated this process using Gitlab CI as illustrated in Figure 2. We use Openshift as it can run multiple instances of our Datasource, monitor them and regulate access. Hence Openshift allows us to address requirements around scalability and security.

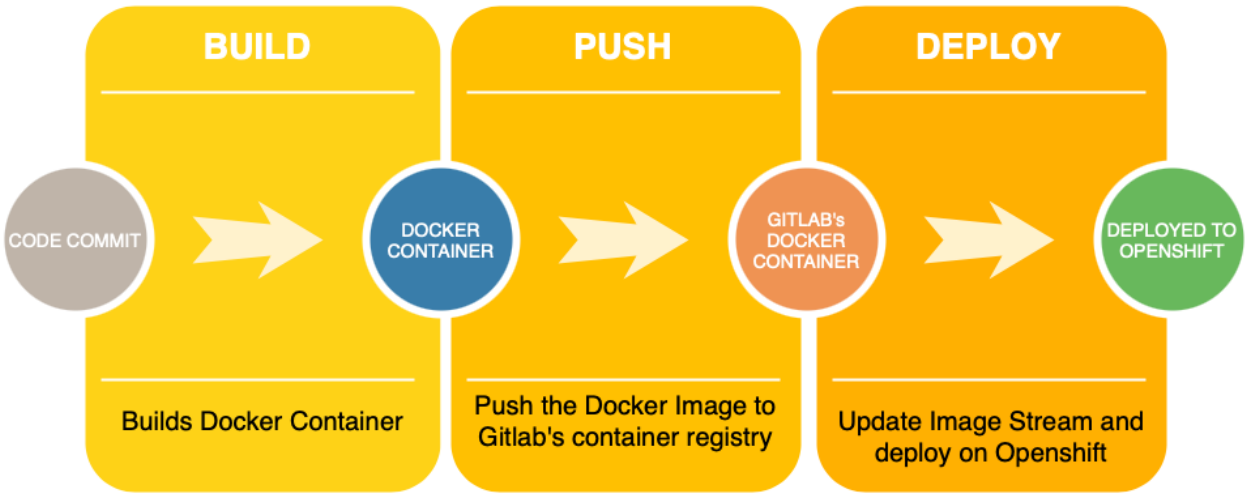

Fig. 2 CI/CD Pipeline for automatic redeployment

d. DEPLOYMENT SCENARIOS

There are many instances of WinCC OA systems deployed throughout CERN on a wide variety of infrastructure architectures, hence there cannot be a '*one size fits all*' solution for deployment. As part of this project we investigated multiple potential deployment scenarios that take into account scalability and security requirements, two of these deployment scenarios are described here; one using Openshift and other using Openstack.

In Figure 3, we deploy our Datasource backend and Grafana instance in Openshift pods allowing us to run multiple instances to achieve scalability. The Openshift pods are running within the General Purpose Network (GPN). The GPN in the internal CERN network that is used for most day-to-day technical activities by users at CERN. As the

Oracle databases are in the more secure Technical Network (TN), we use a 'database link' from an Oracle instance running in the GPN to connect to it using an external database. This setup gives us a single entry point into the more secure TN that gives us a well defined scenario against which to apply our security policy. To control access to our datasource we ensure that it can only accept/receive connections from Grafana Pod and external Database in GPN using Network Policy provided by Openshift.

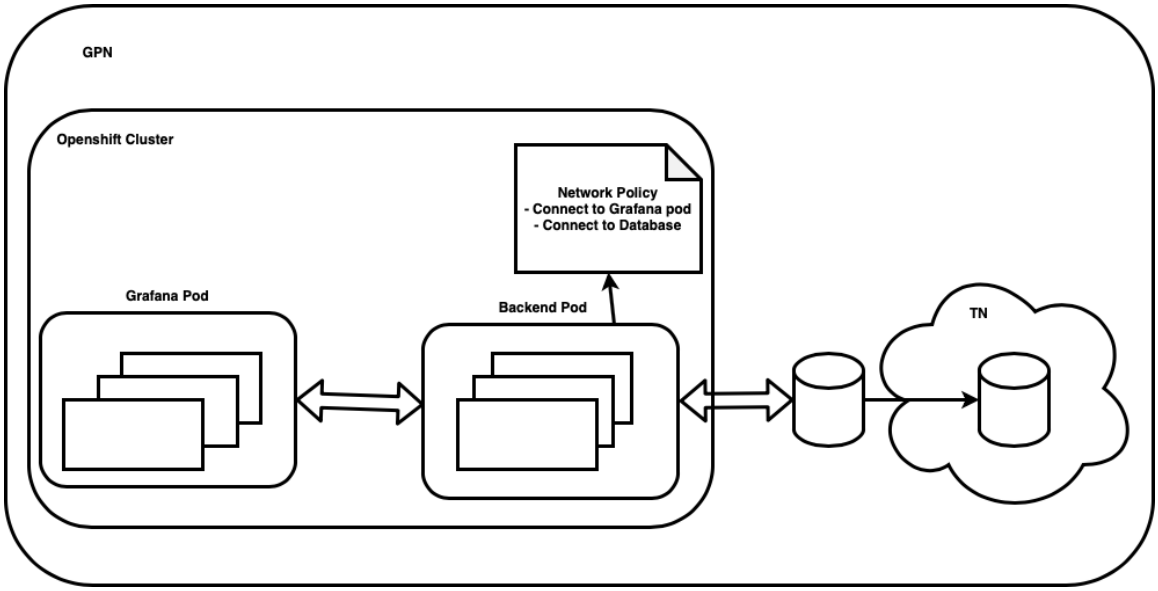

Fig. 3 Deployment using Openshift

In Figure 4 we describe an alternative deployment scenario. In this scenario the Datasource backend is deployed on Openstack. NGINX is used as a reverse proxy to allow communication with the Datasource backend for different services. To ensure security, the firewall is configured in NGINX to allow connections from Grafana pod and the Oracle database only. Other details are similar to those described in the previous scenario.

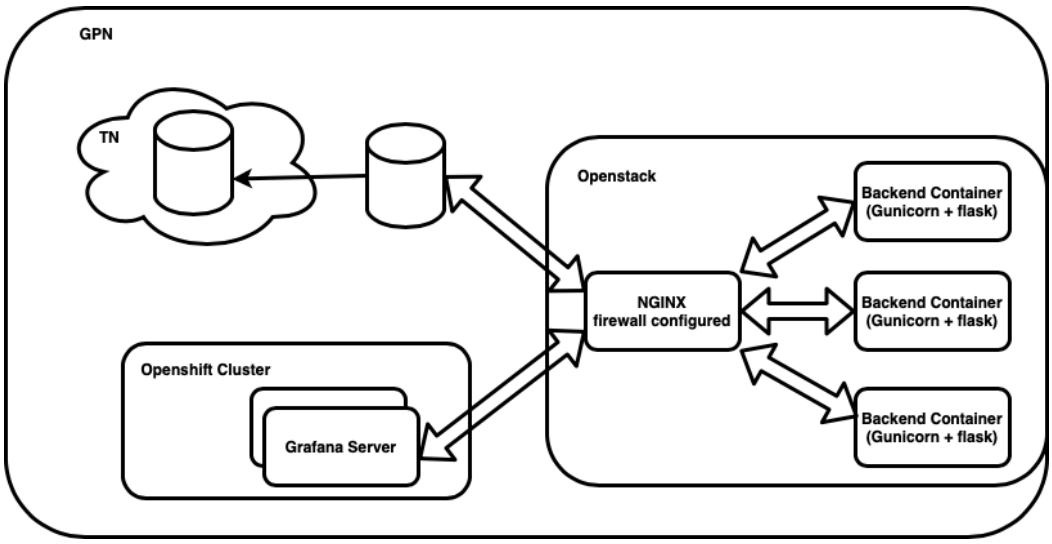

Fig. 4 Deployment using Openstack

3. CONCLUSION

The proposed solutions described in this project, to access WinCC OA archived data using a Grafana based solution within the CERN infrastructure, offer users a powerful and flexible monitoring and visualisation tool. In addition the abstraction of the complexity of the Oracle schema by the datasource removes any requirement for specialist knowledge from the user of the archiving schema.

This project has successfully extended an existing prototype to a level ready to enter initial user testing. We plan to iteratively make changes based on user feedback.

There are still some challenges to be tackled, as there are different systems and experiments at CERN, the deployment would have to be done on a case-to-case basis.