DevOps in cloud hosted database for HL-LHC FBCM detector development

Applicant: Kulakov Danylo (kulakov_d1@knu.ua).
Mentors:
Olena Karacheban (olena.karacheban@cern.ch);
Arkady Lokhovitskiy (Arkady.Lokhovitskiy@cern.ch);
Mihailo Obradovic (mihailo.obradovic@cern.ch).

Abstract: For High-Lumi LHC in the CMS experiment the standalone luminometer FBCM is being designed. It is a silicon pad-based detector with dedicated fast ASIC. Various parts of the detector are at the final design stage and will be produced in 2025. To keeping track of the produced components, test results, and overall progress of the detector construction the database is under development. It is a Django framework with an accompanying Python frontend. The student will contribute to the design and optimization of the database, and will take part in the ongoing development and subsequent deployment of this application. They will gain knowledge and experience in Python scripting, databases, and containerized deployment (OKD).

Project description and deliverables. The Fast Beam Conditions Monitor (FBCM) is the luminometer, which is being designed for HL-LHC. It is built from two rings around the beam pipe: one on the $+Z$ and the other one on the $-Z$ end. Detector is modular and is based on silicon sensors. Each segment consists of 3 front-end (FE) modules, 1 service board for powering (with DC-DC 12 V $\rightarrow$ 1.25 V converter) and one portcard for electrical to optical signal conversion. Each front-end module is composed of 1 or 2 ASICs and 6 silicon pads.

At the prototyping stage of the detector there are multiple versions of the test boards – pre-production versions of the FE modules.

The BRIL group [1] performs different tests to collect and compare data from different versions of test boards. Some of them are equipped with 1, some of them with 2 ASICs. ASICs characteristics are individual and must be saved per channel in an organized manner. This is the primary motivation for the database. In addition, silicon sensors must be characterized in the range of voltage before installation of the test board.

The sensor tests can vary, ranging from IV and CV measurements to studies on the impact of radiation. These tests can require significant time, and the resulting data can come in various formats. Storing these data locally can be risky, as the failure of a local machine could lead to the loss of collected data.

This is the other motivation for the secured database with a simple and reliable user interface.

The Django framework was chosen for this purpose, as it is one of the most powerful Python-based tools and is already widely used in CERN IT services. For hosting database and web application, Kubernetes [2] was selected. Kubernetes is a way of automating application deployment and management, ensuring high availability and reliability.

Work plan:

- **Week 1.** Onboarding: reading literature and papers regarding the CMS inner structure and the signal processing pipeline. Setting up a CERN computer account, gaining access to the data, and running code to open and plot the available data.

- **Week 2−5.** Starting work on the current GitHub project: downloading and developing locally a visualization for data upload for ASIC zip files, which do not need data plotting. Next to be added is the "2-pad" sensors data. Searching for an appropriate design. Uploading data samples that our group needs to store for future analysis, adding plotting where required. Implementation of a high-availability PostgreSQL database to collect and store the data. "6-pad" sensor data will be similar, but with 6 files instead of 2 per sensor.

- **Week 6 − 8.** Test the frontend in isolation, decoupled from the back-end database. Connecting all parts of the project: using Kubernetes cluster to host the frontend, exploring different deployment configurations; testing deployed frontend with high-availability database connected. Testing the system in online mode with different data formats and sizes, and working on optimization. Test deployed database with several users and manage access privileges.

- **Week 8 − 12.** Developing tools for fast data interpretation, such as automatic plot generation and calculation of average values for selected boards. Adding new pages for other board types with different data formats to be saved in the online database.

References:

1. CERN CMS Collaboration, "BRIL Group",
   Available at: https://cms.cern/tags/bril.

2. Kubernetes Documentation, Available at: https://kubernetes.io.