CLARUS MLOps: Boosting Collaboration in Manufacturing Data Value Chains

Aitor Agirre[1], Ana Adell[1], Blanca Kremer[1] and Francisco Fraile[2]

[1] *Ikerlan S.Coop., Arizmendiarrieta 2, Mondragón, Spain*
[2] *Universitat Politècnica de València, Valencia, Spain*

Abstract[1]
This paper presents the MLOps framework adopted in the EU-funded CLARUS Project. This framework combines the Industrial Data Spaces (IDS) framework, a secure, sovereign system of data sharing, with state-of-the-art Machine Learning (MLOps) services aimed at deploying and maintaining machine learning models in production. This innovative stack allows stakeholders (for instance manufacturers and external data scientists) to conform to effective value chains to leverage data in an efficient way.

Keywords
MLOps, IDS, Data Space, Clarus

1. Introduction

MLOps (Machine Learning Operations) is a set of practices, principles and methods aimed at enhancing the lifecycle management of machine learning (ML) models in production environments, streamlining their training, tracking, deployment, monitoring, and maintenance in operation. It can be seen as an extension of DevOps (Development and Operations) principles applied specifically to the field of machine learning.

MLOps is based on automation technologies to ensure the use of the right model in the right place at the right time. MLOps abstract the developer from manual, repetitive and error prone *plumbing* work that consumes much time and does not provide added value, helping the developer in focusing on the development of the AI models themselves.

When developing a solution based on machine learning technologies, there are two main stages to be considered: model training and model execution (inference). Whilst model training usually needs more computing resources (even specialized hardware) and thus is likely to be performed in the cloud, the model inference is not so heavy process and can benefit from the advantages of edge computing. Thus, the edge-to-cloud computing continuum space becomes a natural solution to develop, store, deploy, and execute the AI models.

But this raises a big challenge that must be addressed to enable an effective deployment of AI based industrial solutions on the Edge: *data openness*. On the one hand, the MLOps architecture must access the data that is generated on the Edge (floor plant). On the other hand, the models running on the Edge require updates when new model versions are available, or when concept drift is detected. Thus, a reliable resource (i.e. data and models) exchange mechanism needs to be integrated between the Edge, where the data is produced and the models are executed, and the Cloud, where the models are trained and stored.

Traditionally, this interchange of resources and artifacts between the different stakeholders of the value chain has been made manually, leading to costly and inefficient processes that lack the

[1] Proceedings Acronym: I-ESA 2024 12th International Conference on Interoperability for Enterprise Systems and Applications, April 10–12th, 2024, Crete, Greece
aagirre@ikerlan.es (A. Agirre); aadell@ikerlan.es (A. Adell); bkremer@ikerlan.es (B. Kremer); ffraile@cigip.upv.es (F. Fraile)
0000-0003-4377-8835 (A. Agirre); 0000-0003-4266-182X (A. Adell); 0000-0003-2323-614X (B. Kremer); 0000-0002-3275-7740 (F. Fraile)

advantages that a real automated workflow can provide. In this sense, IDSA (International Data Spaces Association) has positioned [1] to provide a trustable dataspace that considers security, privacy and data sovereignty as first class citizens, following the trend stated in the European Strategy for Data [2].

This work proposes a cutting edge MLOps framework for the edge to cloud continuum, integrated with IDS, which enables a contract-based resource exchange that guarantees data privacy, security, and sovereignty. This way, an effective and trustable data space is stablished between the different stakeholders, e.g. the factories producing the data and the ML model developers. Thus, the factories can expose the data to be consumed by the ML model providers, and, vice versa, the AI providers may make the models available to the end users, in a cost effective and trustable way.

2. CLARUS MLOps Framework

The CLARUS MLOps framework [3] comprises an Edge2Cloud solution that includes both (1) a workflow that covers all the most usual stages that the MLOps concept defines and (2) the toolkit support based on the integration of several opensource tools and new components developed in the CLARUS EU Project. Figure 1 shows the architecture of the CLARUS MLOps Framework.

The left part details the cloud services of the framework, whilst the right part depicts the services and components on the Edge. The workflow comprises several stages over the Edge2Cloud continuum. The stages related to the training, tracking, storage, and registration of the model are executed in the Cloud training platform, whilst the stages related to the model update and model inference are executed on the Edge.

Briefly explained, the workflow enabled by the framework is as follows: first, the Edge exposes the data obtained from the source (e.g. an OPC-UA server in the plant floor), through the IDS connector of the Edge, registering it as a resource. Then, when the *Airflow* pipeline is triggered, either manually through the *Airflow web UI* or automatically through its API, the different steps implemented in the Airflow pipeline are executed.

The initial stage *Read Data* accesses the IDS connector in the cloud, which in turn communicates with the IDS connector on the Edge to get the previously registered resource (the source data). After that, a function that preprocesses this dataset (e.g. a .csv file of raw data) is executed prior to the training stage, to perform the data preparation. Using this processed data, the training stage could train one or more models in parallel, as far as the CLARUS MLOps framework executes over a Kubernetes cluster. These models are tracked by *mlflow* [4] and finally, once the best model is selected according to the considered metrics, the model is registered in the cloud IDS connector, thus making it ready to download to the Edge.

The second part of the story involves the Edge. The models published in the cloud can be downloaded to the edge using the *clarus-agent* microservice. It downloads the models using the IDS connector of the edge, which is connected to the IDS connector in the cloud. Once downloaded and persisted, the updated model is available forexecution by the *Clarus inference engine*.

2.1.Cloud services

The left part of Figure 1 depicts the cloud services of the MLOps framework. They are provided by the integration of several open-source tools and other artifacts provided by CLARUS, which facilitates the integration of the user project GIT repositories into the MLOps framework.

From a user perspective, the usage of the platform is quite straightforward. If the repository is public, the only two artifacts to include in the user repo is (1) a file that implements the Airflow DAG (Directed Acyclic Graph) [5] template provided by CLARUS and (2) a file to enable the synchronization of the user repo with the central GIT repo hosted by the MLOps platform. If the user repo is private, additionally, the user must generate an access token to give access rights to the MLOps platform.

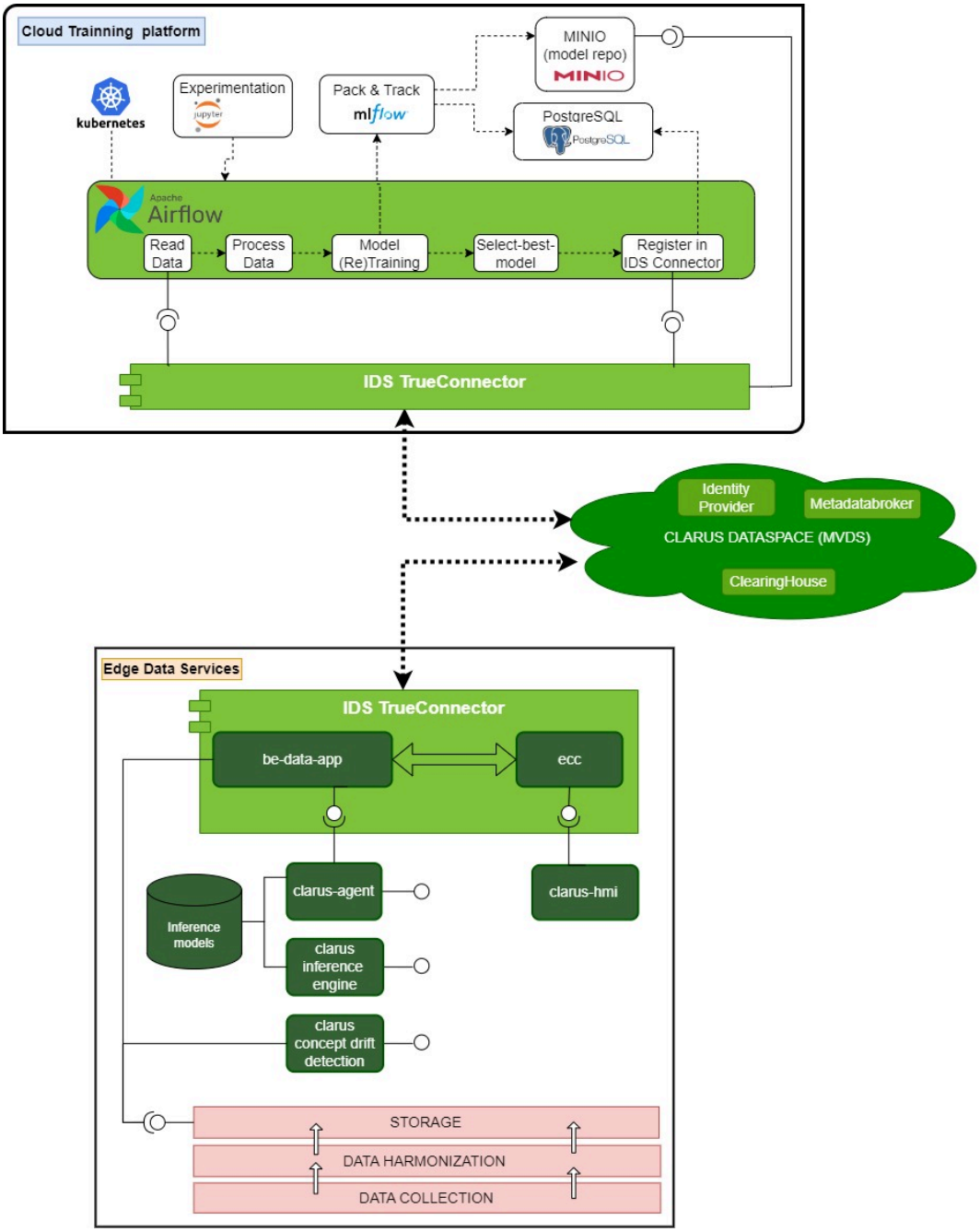

Figure 1: CLARUS MLOps Framework architecture

From an admin perspective, a utility standalone installer is provided to ease the seamless installation of the framework. It includes the installation of Kubernetes and the rest of the components that follow:

Airflow: Airflow [6] is the pipeline orchestrator. It coordinates the execution of the pipeline, which is defined in a user´s DAG file. The source code of the AI project can be hosted in a private user repository, independent from other users. Moreover, the repository can be either public or private, as the MLOps framework supports token based mutual authentication.

mlflow: the framework uses *mlflow* to track experiments, store model artifacts, and provide a user interface for monitoring and managing models. *Mlflow* uses PostgreSQL and MinIO as backend services.

PostgreSQL: a relational database used for storing MLflow tracking data and other information used by the framework, as e.g. the id of the best model selected and stored after parallel training.

MinIO: an object storage server compatible with Amazon S3. It is used to store the models resulting of the training, altough the access to them from the ENG´s IDS *TrueConnector* [7] is performed through the *mlflow* API.

Redis: Redis is an open source, high-performance in-memory data storage system. It is used in the DAG template to exchange data frames between airflow tasks, as far as they are dockerized and could be executed in different hardware nodes of the Kubernetes cluster. This is the recommended (optional) data exchange mechanism. Compared to the previous version [8], now there is no size limit for the data passed between tasks.

Cloud IDS connector: the IDS connector in the cloud serves two main purposes. On the one hand, it is used by the *Read Data* task to access the data exposed by the edge connector through the IDS ecosystem. This way, the data can be securely exchanged, following a contract that specifies the data usage terms and ensures the required levels of privacy and sovereignty. On the other hand, the cloud connector exposes as a resource the trained models and makes them available for downloading.

Note: *The Jupyter Notebooks* [9] *shown in Figure 1 are not installed by the last version v3.0.0 of the CLARUS MLOps Framework setup script, as it is an optional component that is not used for automation but for experimentation when developing the AI models. Nevertheless, it can be manually added to the toolkit, following the readme in the version 2.0.0 of the CLARUS MLOps framework* [8].

Apart from these services, the framework provides the user with a *template* for implementing a CLARUS compatible DAG. In this template DAG the user can configure the execution of each AI task that conforms the pipeline. More specifically, the user can (optionally) specify, for each task, both the docker image and the hardware node of the Kubernetes cluster where the task will be executed. This way, specialized hardware (e.g. GPU-s) that is available in the cluster can be used for the tasks requiring it, and *ad-hoc* docker images can be used for different task needs, such as e.g. a specific Python version or library.

2.2. Edge services

The right part of Figure 1 depicts the edge services of the MLOps framework. The access to the raw data source is represented by the components at the bottom of the figure and is out of the scope of the framework. As an example, they could access e.g. an OPC-UA server or a CAN bus, harmonize the data in .csv format and store the dataset locally to be further consumed by the cloud training platform.

At this point, the ***clarus-hmi service*** comes into play. It provides a REST API that allows the registration of the dataset resource in the edge IDS connector. Thereafter, when the airflow pipeline is triggered, the *Read Data* task could retrive this dataset from the edge, using the IDS connector.

The other way back, the ***clarus-agent*** provides a REST API to enable the downloading of the trained models that are provided by the cloud IDS connector. At its current version, the framework supports the downloading of AI models in *pickle* format [10].

Once a model is downloaded, the *clarus-agent* stores the model in a local database and uses the REST API provided by the ***clarus-inference-engine*** to update the active model for local edge inference. This inference can also be triggered through this API.

The ***clarus-concept-drift-detection*** service (still in development) uses updated raw data provided by the data harmonization component to eventually detect the *concept drift* of the AI model. At its current state, it uses the *evidently* library [11] to detect only *data drift*. When a drift in the source data is detected, the service may eventually trigger a retraining of the model in the cloud.

Finally, an ***edge IDS connector*** has been developed to achieve two objectives. First, it enables the registration and makes available the data coming from the plant floor. These data can be stored in any local database on the edge, although currently the edge IDS connector limits to *MinIO* and *http*

servers as storage mediums for the data files. Second, the edge IDS connector is used by the *clarus-agent* to download trained models, retrieving them from the *cloud IDS connector*.

3. Conclusions

The paper presents the CLARUS MLOps framework, which provides innovative MLOps functionalities over a trustable dataspace for the edge to cloud continuum. To achieve this goal, a novel approach that integrates an Edge2Cloud MLOps workflow over IDS is proposed. As the framework integrates with IDS, it natively considers relevant aspects as security, privacy, and data sovereignty. This enables a trustable playground where all the involved stakeholders can contribute to effectively industrialize a real Edge AI.

The framework leverages a set of well-known and widely used opensource tools such as Airflow or MLFlow and provides additional services to enable a seamless integration of the MLOps workflow over IDS. Moreover, the framework supports useful features as e.g., parallel training over a Kubernetes cluster or unlimited sized data frame passing between tasks and can be installed either on premise (to take advantage of owned specialized hardware) or on the public cloud.

Now of writing, the framework is still under development, although version 3 is already available at [3]. Preliminary results show that the proposed MLOps platform can be easily used by data scientists that are not necessarily software engineers, and can help in managing, organizing, and optimizing access to the hardware infrastructure on premises. By using the platform, the AI model developers can focus on data engineering and model training and validation, obviating the industrialization aspects of the AI models.

Nevertheless, several aspects can be enhanced and will be addressed soon. (1) Regarding the infrastructure, the installation of the framework over hybrid on premise/public clouds will provide more flexibility and scalability when on premise hardware resources are not enough. (2) The concept drift service will also be integrated in the platform as part of the workflow, to eventually trigger the model retraining automatically. (3) To enhance the user experience, a graphical tool to generate the Airflow DAG is being developed. By using this zero-code tool, the AI model developer will be able to easily define the pipeline of the AI related tasks, such as data gathering, data processing or model training. For each task, only the library dependencies (e.g. PyTorch, TensorFlow or Scikit-learn), Python version and required hardware shall be specified, and the platform will manage this information to provision the required docker images and machines. (4) Also, from the library dependencies specification, the platform will automatically build the docker image to be deployed on the edge for model inferencing.

Acknowledgements

This work has been supported by the project “CLARUS”, which has received funding from the European Union’s Horizon Europe research and innovation program under grant agreement No. 101070076.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] International Data Spaces Association (IDSA), "Implementing the European Strategy of Data (position paper)," April 2020. [Online]. Available: https://internationaldataspaces.org/wp-content/uploads/IDSA-Position-Paper-implementing-european-data-strategy-role-of-IDS.pdf.

[2] European Commission, "A European strategy for data," 2020. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52020DC0066.

[3] Ikerlan, "CLARUS MLOps framework v3.0.0," 2024. [Online]. Available: https://github.com/CLARUS-Project/ai-toolkit/tree/v3.0.0-main.

[4] Linux Foundation, "mlflow," [Online]. Available: https://mlflow.org/.
[5] Apache, “DAGs - Airflow Documentation - Apache Airflow,” [Online]. Available: https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dags.html.
[6] Apache, “Airflow,” [Online]. Available: https://airflow.apache.org/.
[7] ENGineering, "True Connector," [Online]. Available: https://github.com/Engineering-Research-and-Development/true-connector.
[8] Ikerlan, "CLARUS MLOps framework v2.0.0," 2023. [Online]. Available: https://github.com/CLARUS-HE-Project/ai-toolkit/tree/v2.0.0-main/Cloud.
[9] Jupyter, "Jupyter Notebooks," [Online]. Available: https://jupyter.org/.
[10] Python.org, "Pickle format," [Online]. Available: https://docs.python.org/3/library/pickle.html.
[11] Evidently AI, "Evidently library," [Online]. Available: https://www.evidentlyai.com/.