Integration and deployment of Model Serving Framework to serve machine learning models at production scale in easy way

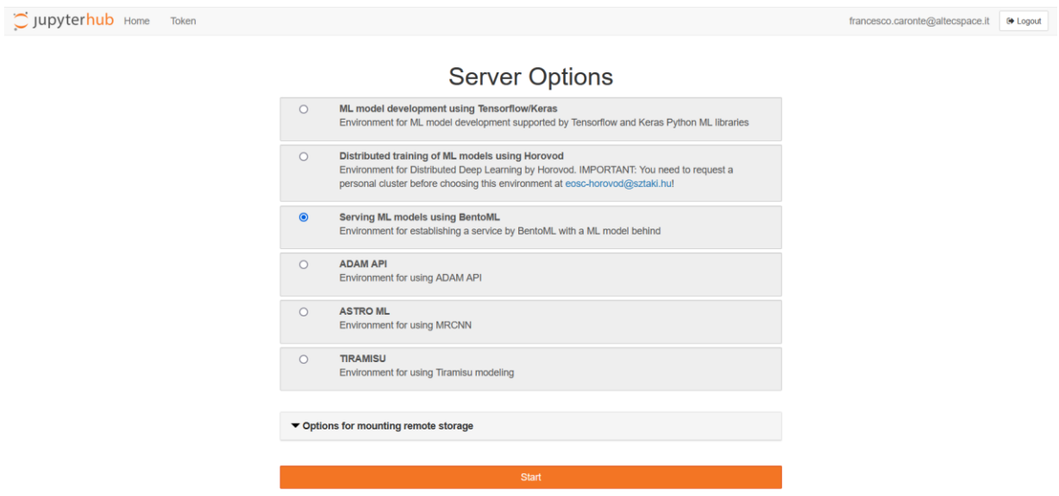
F. Caronte[1], R. Messineo[1], J. Kovacs[2], E. Sciacca[3]

francesco.caronte@altecspace.it, rosario.messineo@altecspace.it, jozsef.kovacs@sztaki.hu, eva.sciacca@inaf.it

ABSTRACT With the help of machine learning systems, we can examine data, learn from that data, and make decisions. Now machine learning projects have become more relevant to various use cases, but to manage them, especially if numerous, is a challenging task. For this reason, several **Machine Learning Operations (MLOps)** tools have been developed. These tools are the main platforms, hosting the full machine learning process lifecycle, starting with data management, and ending with model versioning and deployment.

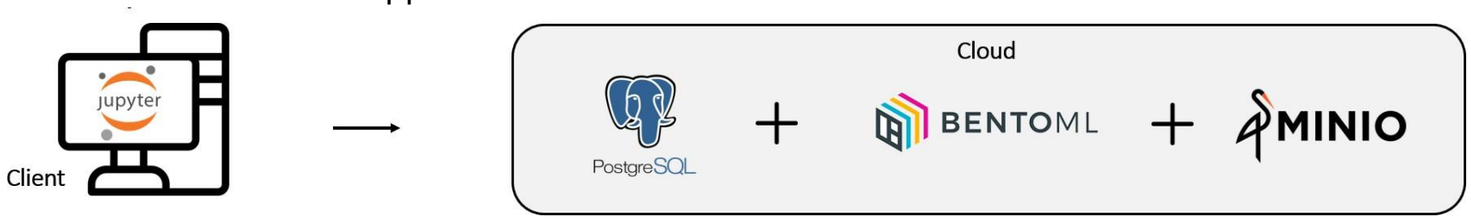
NEANIAS (Novel EOSC Services for Emerging Atmosphere, Underwater & Space Challenges) is an ambitious project that comprehensively addresses the challenges set out in the 'Roadmap for EOSC' foreseen actions. NEANIAS drives the co-design and delivery of innovative thematic services, derived from state-of-the-art research assets and practices in three major sectors: Underwater research, Atmospheric research and Space research.
The **Machine learning core services** are composed by a JupyterHub instance (C3.1 named AI-Gateway) that enable different profile servers to:
(C3.2) Serve machine learning model in production enviroment.
(C3.3) Enable a deep learning training framework using a Horovod cluster.
(C3.4) Perform distributed calculations using Apache Spark.

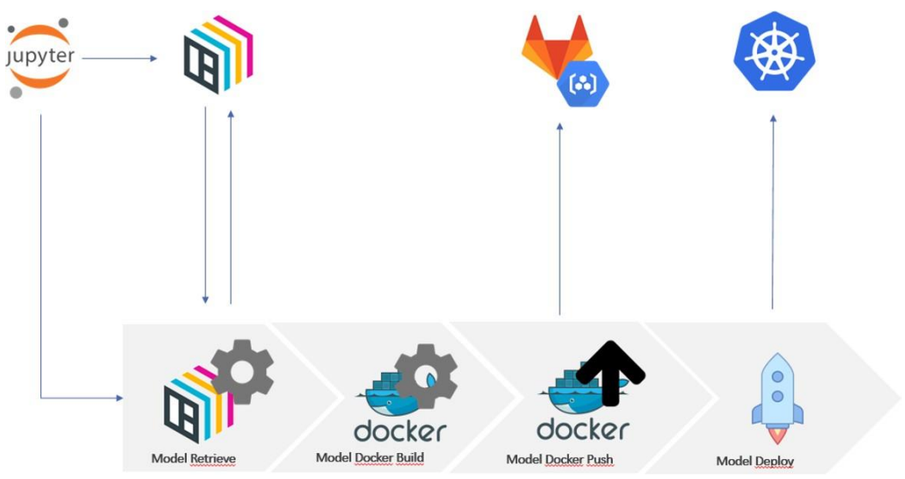
All these services, combined and integrated, are used for astrophysics use cases. In particular, two environments were integrated in C3.1 to classify radio astronomical sources, galaxies and image artefacts. These are based on two deep learning models tailored to **object detection** (based on maskRCNN) and **semantic segmentation** (based on Tiramisu) and trained on radio dataset from the Square Kilometre Array (SKA) precursors.
Kubernetes is the chosen container orchestration platform to run these set of services. The Kubernetes platform has been deployed on top of the OpenStack cloud infrastructure and these two platforms are provided and maintained by GARR.



C3.1 AI-Gateway JupyterHub instance for Neanias

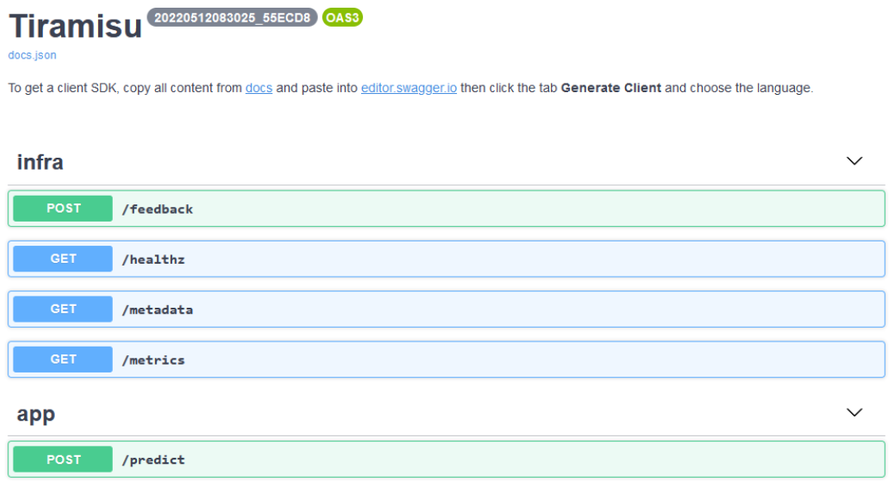
MODEL SERVING is a platform that simplifies ML model deployment and enables to serve models at production scale in minutes. Model serving provides different libraries to **package** the model, **serving** it offline in the development environment or online using a Kubernetes cluster. The model is served using flask and supports micro-batching mechanisms to perform prediction, and it automatically generates a deployment endpoint signed with authorized SSL. Model serving is implemented with the following technology stack:
BentoML: an open platform that simplifies ML model deployment and enables to serve models at production scale in minutes
MinIO: a High Performance Object Storage used to store BentoML artifacts
Yatai Server: the BentoML backend
gRCP Proxy: a proxy between C3.1 (JupyterHub) and Yatai service that permits to authorize and authenticate client requests
GitLab CI/CD: used to deploy or destroy the machine learning model on the Kubernetes platform
Kubernetes: an open-source system for automating deployment, scaling and management of the aforementioned containerized applications





GitLab CI for deploy and destroy the model on Kubernetes platform

With the Model Serving solution the Data Scientist can create different instances of the same model and deploy all models on the Kubernetes cluster and run the prediction using the **model API**.
The API are defined by the Data Scientist so can use different type of data structures as DataFrame or a simple path that contains the data to use for prediction. WeDav is used as webservice, and, currently NextCloud is the web storageservice and the directories can be mount using the WebDav protocol.



Model Served using Model Serving

The model serving have correctly integrated the **Tiramisu** model. Tiramisu introduces a variation in the U-Net architecture, by employing a sequence of DenseNet blocks, rather than standard convolutional blocks in both downsampling and upsampling paths. The Tiramisu network consists of a downsampling path for feature extraction and an upsampling path for output generation, with skip connections. This contributes to making the network require less parameters, as it reuses feature maps from earlier layers, freeing deeper layers from the need to learn redundant features. The model deployed give in input the path directory that contains radio astronomical images of our galaxy and the model save the prediction in an output directory returned by the prediction API.

CONCLUSIONS The Model Serving can provide an easy way to turn the ML model into production-ready API endpoint and permits to standardize model packaging and ML service definition to streamline deployment, deploy and operate ML service workload at scale on Kubernetes using an automatic GitLab CI.

Acknowledgements This work has been fully supported by NEANIAS, funded by the European Union's Horizon 2020 research and innovation program, under grant agreement No 863448.

(1) ALTEC *(2)* SZTAKI *(3)* INAF ISTITUTO NAZIONALE DI ASTROFISICA OSSERVATORIO ASTROFISICO DI CATANIA

NEANIAS