RE4DY

MANUFACTURING DATA NETWORKS

RE4DY TOOLKIT

Name of the Tool	ALIDA
Tool Owner	Industry Commons Foundation
Version	1.0
Date	Nov 2025
Version	V1.0

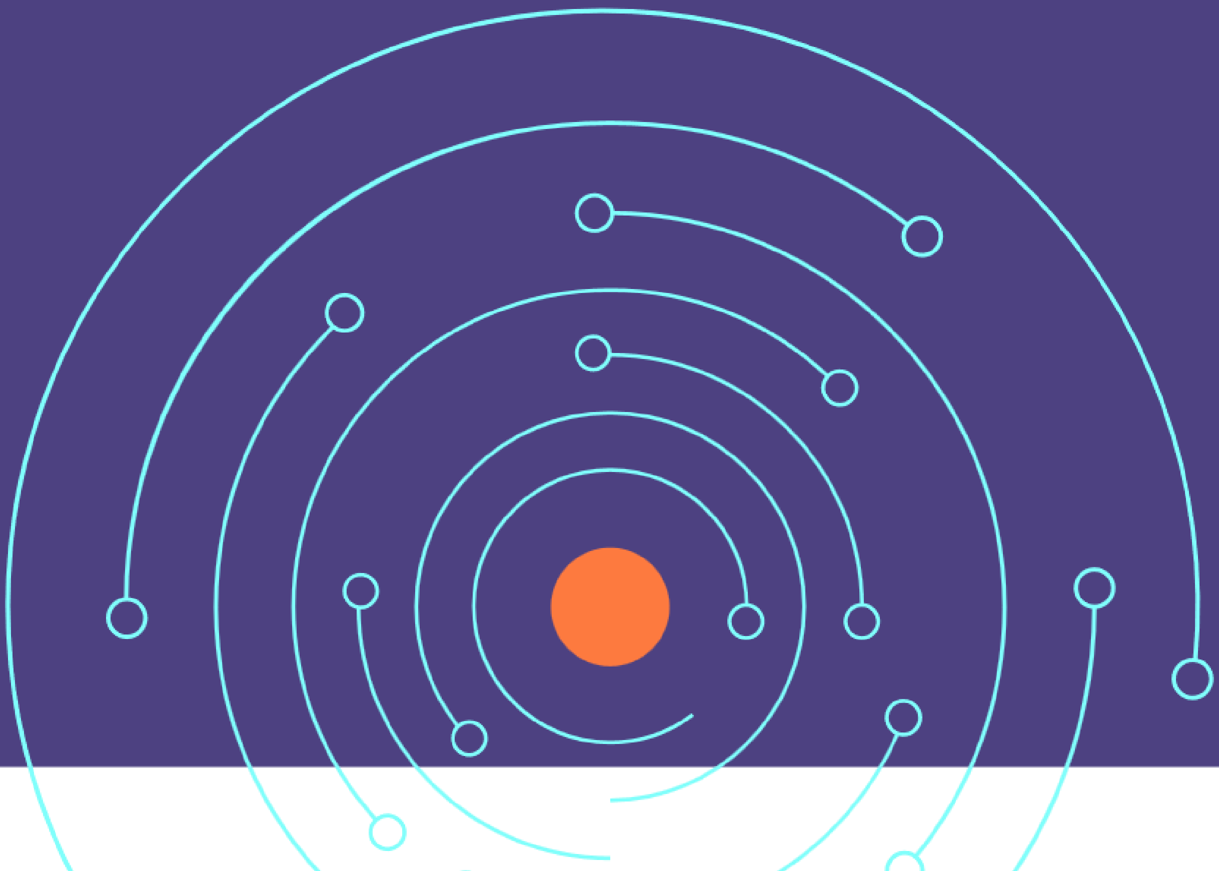

Table of contents

1. Component Description

ALIDA provides a micro-service-oriented platform for the deployment and execution of Big Data Analytics (BDA) application compositions in several domains and scenarios. Within RE4DY project this platform will be enriched with Federating Learning capabilities in order to assure a privacy-preserving federated learning environment with which to build data-driven AI models to be assessed by RE4DY trials.

Therefore, this component enables users to create AI/ML pipelines by dragging and dropping reusable blocks from its own catalogue through a graphical visual designer. The pipelines can either run on the ALIDA cloud or be exported and executed locally. If the needed blocks are not available in the catalogue, the user can code them starting from the provided templates.

2. Input

- Graphical specification of pipelines: that users enter through the graphical designer integrated in the ALIDA platform.
- *Datasets:* that users can upload and use as starting blocks in their pipelines. After the upload, they become ("canvas draggable") ALIDA entities.
- Generic Local datasets/data sources (e.g., local filesystem folder or similar): rather than being ALIDA entities, these are completely external and independent data sources which can be accessed from the exported pipelines.
- Custom *BDA Services:* that users can develop by following the provided templates, upload and use whenever needed in order to assemble pipelines.
- External Streaming Data Sources: that can be accessed through the ALIDA *Streaming Data Ingestion Services - a type of BDA Service - already available in the catalogue or that can be developed by the user.*

3. Output

- Trained ML models: resulting from the interaction between pipelines of Participants (that train the model locally) and Aggregators (that aggregate the partial models returned by the Participants).
- Participant pipelines: that can be exported from ALIDA and executed externally on both edge and cloud sides.

4. Information Flow

Note: the following diagrams are only examples.

Aggregator and Participants Pipelines Preparation

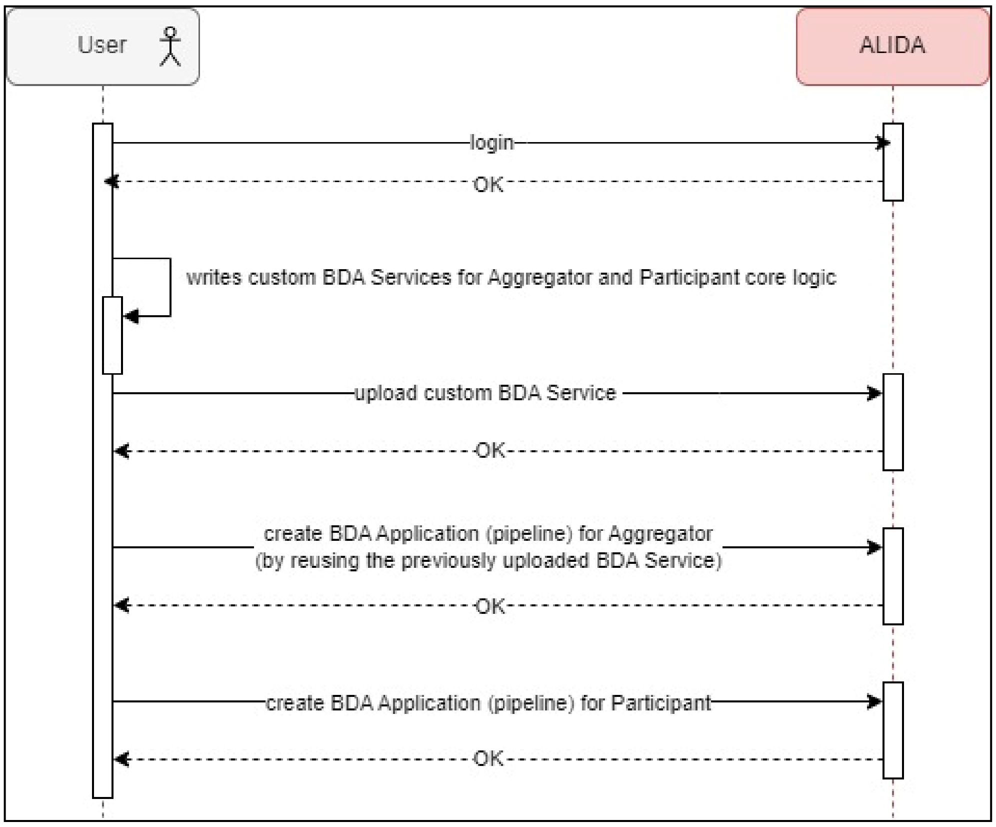
User
ALIDA
login
OK
writes custom BDA Services for Aggregator and Participant core logic
upload custom BDA Service
OK
create BDA Application (pipeline) for Aggregator
(by reusing the previously uploaded BDA Service)
OK
create BDA Application (pipeline) for Participant
OK

Figure 1: ALIDA FML application design steps example

As shown by the sequence diagram above, the design of a FML application with ALIDA platform is summarized through the following main steps.

1. The user log-in to ALIDA platform.
2. If not already available from catalogue, the user defines and develops the BDA services accordingly to his/her needs for the Aggregator and Participants roles.
3. The user upload custom BDA services to ALIDA.
4. At this point the user can design the pipeline for the Aggregator BDA application by simply drag & drop the BDA services needed, included those previously developed and uploaded.
5. Likewise, the user can design the pipeline for the Participant BDA application.

Federated Training

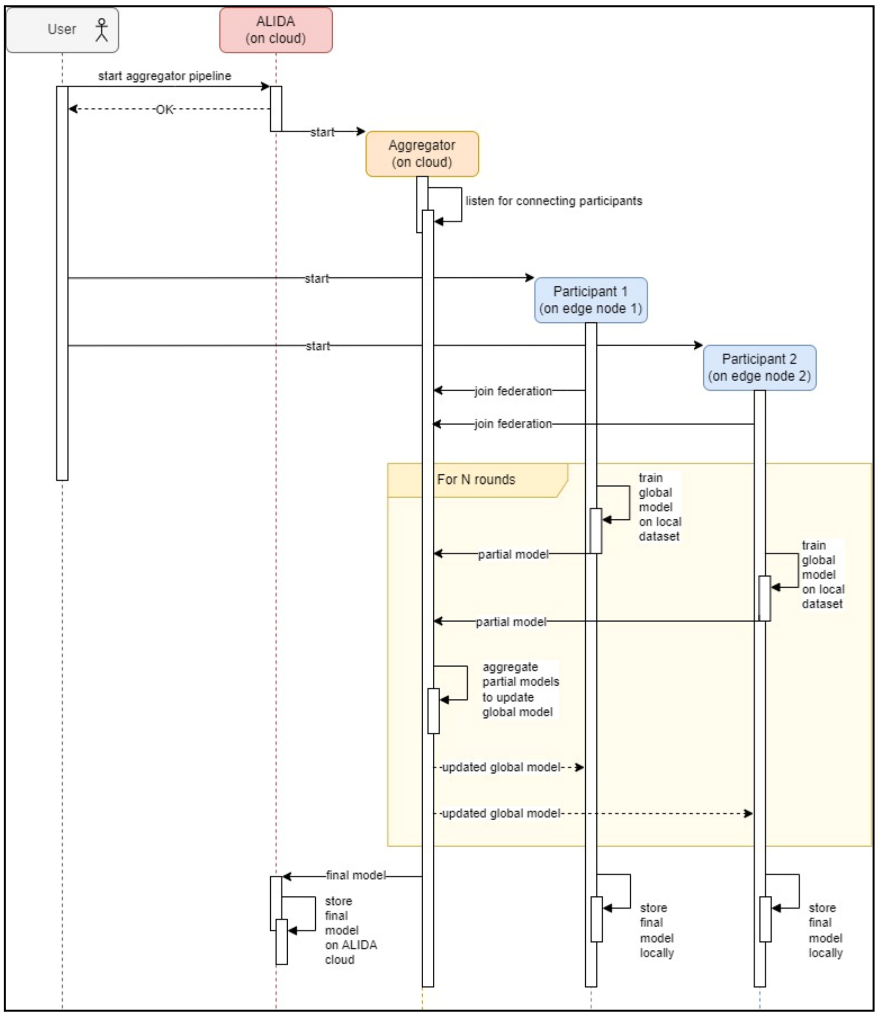
User
ALIDA (on cloud)
start aggregator pipeline
OK
start
Aggregator (on cloud)
listen for connecting participants
start
Participant 1 (on edge node 1)
start
Participant 2 (on edge node 2)
join federation
join federation
For N rounds
train global model on local dataset
partial model
train global model on local dataset
partial model
aggregate partial models to update global model
updated global model
updated global model
final model
store final model on ALIDA cloud
store final model locally
store final model locally

Figure 2: FML training example with ALIDA platform

According to the sequence diagram above, the typical process of Federating Training of models with ALIDA involves the following steps.

1. The User starts the Aggregator pipeline on ALIDA platform.
2. ALIDA starts the execution of the Aggregator pipeline.
3. The Aggregator starts to listen for connecting pipeline Participants.
4. The User starts - on two different edge nodes - Participant 1 and 2 pipelines.
5. The Participants contact the Aggregator to join the training federation.

6. Participants and Aggregator interact[1] with each other for N rounds to carry out the training. The number of rounds can be configured by the developer.

7. At the end of the training, a copy of the final model is stored by Participants locally and by the Aggregator on the ALIDA cloud.

5. Internal Architecture

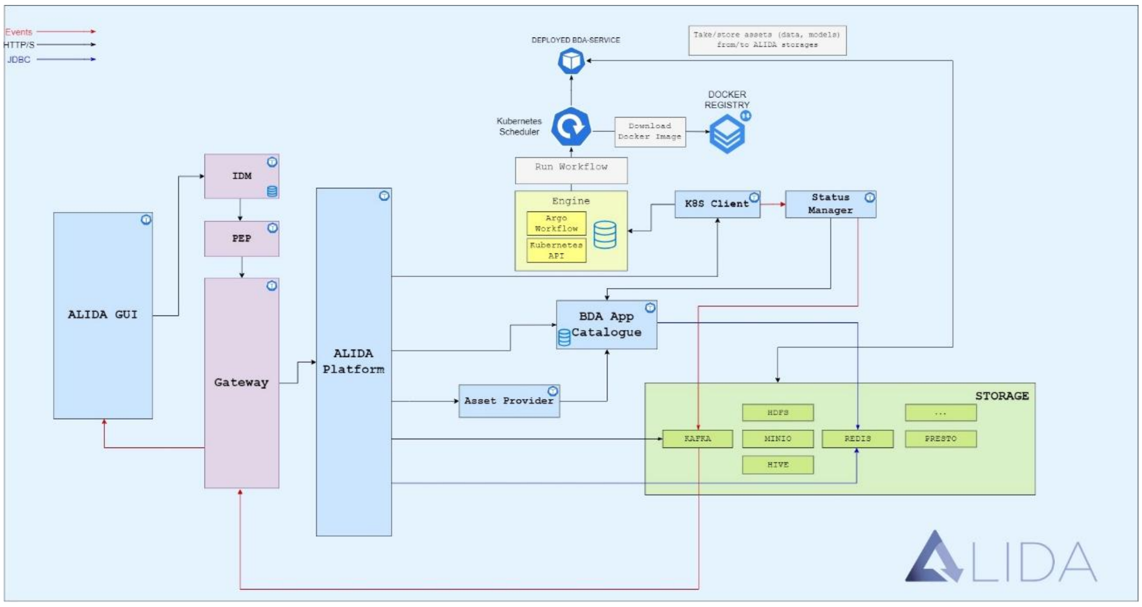

Figure 3: ALIDA internal architecture

- **ALIDA GUI**: the graphical user interface of ALIDA, based on React JavaScript UI library.

- **ALIDA Platform**: the main ALIDA back-end service, responsible for managing all requests coming from the GUI and other components.

- **BDA Application Catalogue**: contains all operations (creation, reading, updating, deletion) related to the objects managed in ALIDA: datasets and models metadata, BDA services, BDA applications, and more.

- **Asset Provider**: component of ALIDA responsible for parsing the BDA application pipeline generated through the ALIDA graphical designer and providing assets such as exporting the BDA application and the generated model.

- **K8S Client**: component for executing/stopping BDA applications workflows and sending BDA application events that are executed in ALIDA, interacting with K8S and Argo Workflow APIs.

[1] *The interaction between Aggregator and Participants during the training is entrusted to the Flower library mechanisms and it is transparent to the end users.*

- **Status Manager**: based on incoming status events, manages, and updates the states of the BDA applications by interfacing with the BDA Catalogue.

6. API

Notes:

- The Flower open-source framework (https://flower.dev/) has been used to enable Federated Machine Learning (FML) applications. The framework allows to focus on the coding of the training part placed at each participant and how the generated model weights can be subsequently aggregated by the aggregator server, including defining multiple strategies. Data exchange between a participant and an aggregator server is done - transparently to the user by the Flower internal mechanisms - using the gRPC (Remote Procedure Call) protocol, a protocol originally optimized by Google more than RPC, which uses protocol buffers and HTTP 2 for data transfer. Because it is a machine-to-machine transfer, gRPC is faster than REST call invocations for both writing and reading. In addition, gRPC supports the use of authentication mechanisms via SSL/TLS to authenticate the server and encrypt all data exchanged between clients and the (model aggregator) server.

- ALIDA REST API description is not publicly available.

REST API

BDA Application Controller

Endpoint	**Summary**
POST /api/v2/bundle/bdas	Register a new BDA application
GET /api/v2/bundle/bdas/**	Get all the registered BDA applications the logged user can get
DELETE /api/v2/bundle/bdas/{id}	Delete a BDA application by ID
PUT /api/v2/bundle/bdas/{id}	Updates an existing BDA application
POST /api/v2/bundle/start/bdas/{bdaId}/workflows/{workflowId}	Start a BDA application by ID

POST /api/v2/bundle/stop/bdas/{bdaId}/workflows/{workflowId}	Stop a BDA application by ID

Models Controller

Endpoint	**Summary**
GET /api/object/models	Get all the machine learning models metadata
POST /gateway/platform-alida/api/proxy/ap/export/bdas/{modelId}	Download a machine learning / deep learning model produced by a specific BDA application by the "modelId"

Registration Controller

Endpoint	**Summary**
GET /api/platform/services	Get metadata about all the registered BDA- services in the ALIDA catalogue
POST /api/platform/services	Register a new BDA-service
DELETE /api/platform/services/{id}	Unregister a BDA- service
GET /api/platform/services/{id}	Get metadata about a specific BDA-service

7. Implementation Technology

Programming Language:

- Python: micro-services for the workflow parser, and for communicating with the Kubernetes and Argo Workflows APIs, were built in Python programming language to execute workflows designed through the ALIDA GUI (graphical user interface).
- Java: most of the micro-services of the ALIDA platform were developed in Java, especially the management of the ALIDA catalogue, and the workflow status manager, was developed using Java libraries.

- JavaScript: was used for the realization of the graphical user interface of ALIDA, along with the use of JavaScript libraries such as React.

Platform:

- Kubernetes: the micro-service based ALIDA platform is run and managed within a Kubernetes cluster (K8s). This enables resource scaling and support of other native Kubernetes frameworks for orchestration and execution of micro-services workflows.

- Docker: the designed workflows (or pipelines) can be exported as docker compose files referencing docker images published on a docker registry. Therefore, they can be run on any node supporting docker and able to access the registry to download the docker images.

Libraries/Frameworks

- Flower - used as a framework to enable Federated Machine Learning in BDA services.

- Argo Workflows - used for the creation and execution of workflows of BDA services within a K8s cluster.

- Apache Spark - big data analytics framework used in BDA services for processing and training models on large amounts of data.

- Apache Kafka - message bus framework used as a means of communication between streaming BDAs services.

- MinIO - object storage used for historicizing and storing data and trained ML/DL models.

- Redis - an in-memory data structure store used to store metadata, useful for the ALIDA platform.

- HDFS - distributed file system for storing and historicizing data and trained ML/DL models.

- Hive - database to store tabular datasets.

- FIWARE KeyRock - used by ALIDA as identity management.

- Spring Boot - framework used to develop the main components of the ALIDA backend.

- React - library used to develop the front-end component of the ALIDA platform.

- React Flow - library used for the development of the workflows' builder.

- ALIDA allows users to write custom pipeline blocks, BDA (Big Data Analytics) services by using Python starting from the provided templates

(https://gitlab.alidalab.it/external/templates-bda-services). The user can include the Python the own or imported/installed machine learning / deep learning / other libraries as:

- TensorFlow
- Keras
- ScikitLearn
- Pandas

8. Comments

None.