D5.5 – i4Q Manufacturing Line Reconfiguration Toolkit

WP5 – BUILD: Rapid Manufacturing Line Qualification and Reconfiguration

i4Q has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958205.

Document Information

GRANT AGREEMENT NUMBER	958205	ACRONYM	i4Q	
FULL TITLE	Industrial Data Services for Quality Control in Smart Manufacturing			
START DATE	01-01-2021	DURATION	36 months	
PROJECT URL	https://www.i4q-project.eu/			
DELIVERABLE	D5.5 – Manufacturing Line Reconfiguration Toolkit			
WORK PACKAGE	WP5 – BUILD: Rapid Manufacturing Line Qualification and Reconfiguration			
DATE OF DELIVERY	CONTRACTUAL	June 2022	ACTUAL	June 2022
NATURE	Report	DISSEMINATION LEVEL	Public	
LEAD BENEFICIARY	UPV			
RESPONSIBLE AUTHOR	Raul Poler, Miguel A. Mateo-Casali, Francisco Fraile, Faustino Alarcón, Beatriz Andrés, Manuel Díaz-Madroñero, Josefa Mula, Raquel Sanchis, Angel Ortiz (UPV)			
CONTRIBUTIONS FROM				
TARGET AUDIENCE	1) i4Q Project partners; 2) industrial community; 3) other H2020 funded projects; 4) scientific community			
DELIVERABLE CONTEXT/ DEPENDENCIES	This document has no preceding documents. The second version will be released at M32 (D5.11) to deliver the final i4Q Rapid Manufacturing Line Qualification and Reconfiguration. This document presents a technical overview of the Manufacturing Liner Reconfiguration Toolkit solution (i4Q LRT).			
EXTERNAL ANNEXES/ SUPPORTING DOCUMENTS	None			
READING NOTES	None			
ABSTRACT	This document is a Technical Specification document about the development of the i4Q Manufacturing Line Reconfiguring Toolkit ($i4Q^{LRT}$). This document which provides a thorough description and analysis of the functionalities, features, and the current implementation status. It provides an in-depth technical overview of the principal functional sub-components (i.e., features) of the Solution.			

Document History

VERSION	ISSUE DATE	STAGE	DESCRIPTION	CONTRIBUTOR
0.1	12-May-2022	ToC	ToC created and sent for review	UPV
0.2	10-Jun-2022	Working Version	1st input to all sections	UPV
0.3	17-Jun-2022	1st Draft	First draft sent for internal review	UPV
0.4	20-Jun-2022	Internal review	Internal review	IKERLAN, FIDIA
0.5	24-Jun-2022	2nd Draft	Addressing the comments from the internal review. Updated draft sent to the coordinator.	UPV
1.0	30-Jun-2022	Final doc	Final quality check and issue of final document	CERTH

Disclaimer

Copyright message

TABLE OF CONTENTS

Executive summary **5**

Document structure **6**

1. General Description **7**

1.1 Overview 7

1.2 Features 7

2. Technical Specifications **8**

2.1 Overview 8

2.2 Architecture Diagram 8

3. Implementation Status **10**

3.1 Current implementation 10

3.1.1 Solution features analysed and mapping with user requirements 10

3.2 Next developments 11

3.3 History 12

4. Conclusions **13**

Appendix I **14**

LIST OF FIGURES

Figure 1. i4Q^{LRT} Solution Architecture 9

LIST OF TABLES

Table 1. i4Q^{LRT} Version history 12

ABBREVIATIONS/ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
CI/CD	Continuous integration and delivery
CVE	Common vulnerabilities and Exposures
DIT	Data Integration and Transformation
DR	Data Repository
i4Q	Industrial data services for Quality Control in Smart Manufacturing
LRT	Manufacturing Line Reconfiguration Toolkit
OEE	Overall Equipment Effectiveness

Executive summary

This document presents an executive explanation of the **i4Q Manufacturing Line Reconfiguration Toolkit** (i4Q^{LRT}) Solution providing the general description, the technical specifications and the implementation status. The deliverable **D5.5** is the Source Code of the i4Q^{LRT} Solution that is in a private repository of Gitlab: https://gitlab.com/i4q.

The documentation associated to the i4Q^{LRT} Solution is deployed on the website http://i4q.upv.es. This website contains the information of all the i4Q Solutions developed in the project "Industrial Data Services for Quality Control in Smart Manufacturing" (i4Q). The direct link to the i4Q^{LRT} Solution documentation is http://i4q.upv.es/21_i4Q_LRT/index.html.

Such documentation is structured according to:

- General description
- Features
- Images
- Authors
- Licensing
- Pricing
- Installation requirements
- Installation Instructions
- Technical specifications of the solution
- User manual

Document structure

Section 1: Contains a general description of the **i4Q Manufacturing Line Reconfiguration Toolkit**, (i4Q^{LRT}) providing an overview and the list of features. It is addressed to final users of the i4Q Solution.

Section 2: Contains the technical specifications of the **i4Q Manufacturing Line Reconfiguration Toolkit** (i4Q^{LRT}), providing an overview and its architecture diagram. It is addressed to software developers.

Section 3: Details the implementation status of the **i4Q Manufacturing Line Reconfiguration Toolkit** (i4Q^{LRT}), explaining the status, next steps and summarizing the implementation history.

Section 4: Provides the conclusions.

APPENDIX I: Provides the PDF version of the **i4Q Manufacturing Line Reconfiguration Toolkit** (i4Q^{LRT}) web documentation, which can be accessed online at:
http://i4q.upv.es/21_i4Q_LRT/index.html

1. General Description

1.1 Overview

The **i4Q Manufacturing Line Reconfiguration Toolkit** (i4Q^{LRT}) is a collection of optimisation micro-services that use optimisation algorithms and simulation to evaluate different possible scenarios and propose changes in the configuration parameters of the manufacturing line to achieve the quality targets. i4Q^{LRT} AI learning algorithms develop strategies for machine parameters calibration, line setup and line reconfiguration.

The objective of the i4Q Manufacturing Line Reconfiguration Toolkit is to increase productivity and reduce the efforts for manufacturing line reconfiguration through AI. This tool consists of a set of analytical components (e.g., optimisation algorithms, machine learning models) to solve known optimisation problems in the manufacturing process quality domain, finding the optimal configuration for the modules and parameters of the manufacturing line. Some examples of the problems that the i4Q^{LRT} can solve for manufacturing companies are the fine-tuning of the configuration parameters of the machines along the line to improve quality standards or improving the manufacturing line set-up time.

1.2 Features

I4Q$^{LRT:}$

- Provides the possibility to deploy algorithms as micro-services at the edge of the network, via distributable files.
- Provides information necessary to understand the use, characteristics and needs of the algorithm.
- Offers the ability to instantiate optimization algorithms as standalone (micro-)services that can be deployed and managed at Edge.
- Allows algorithms to be instantiated and run at the edge of the network, obtaining information directly from the production line, by configuring the data source. This way, the algorithms can optimise the production line by providing a correct configuration.

2. Technical Specifications

2.1 Overview

The **i4Q Manufacturing Line Reconfiguration Toolkit** (i4Q^{LRT}) is mapped to the Simulation and Optimisation sub-component of the Enterprise tier. To facilitate the interconnection between processes and to scale the deployment of solutions across different industries and sectors, it is important that the algorithms are semantically interoperable and that all use the i4Q digital models and ontologies. Also, to ease integration and interoperability with other i4Q solutions, they all must use the same services to access data at rest and data in motion for model instantiation and execution. To address this, the strategy is to wrap algorithms into an interoperability layer that will implement the different cross-cutting functions (clients to data services, security and compliance services, libraries for digital models and vocabularies). The interoperability layer will also provide endpoints so that other solutions can instantiate and run the algorithms in complex workflows.

Some of these workflows may require that models are executed at the edge tier, close to the data source, to minimise latency. Therefore, the interoperability layer must ensure that the algorithms are deployable at the edge, through the Distributed Computing sub-component, and manageable by the Orchestration Management Sub-component.

Finally, to enable the digital twinning of models and algorithms with the underlying physical architecture, they should be regarded as (virtual) assets. For instance, an algorithm to optimise (i.e., fine tune) the configuration parameters of a machine should be regarded as a feature of its digital twin, and in that sense, the services provided by the interoperability layer should be mappable as features of the digital twin.

2.2 Architecture Diagram

i4Q^{LRT} includes different processes, functions and AI models which are mapped to the Enterprise Tier and to the Edge Tier of the i4Q Reference Architecture. Considering that it is a solution found at various levels of the architecture:

- **Enterprise Tier:** The i4Q^{LRT} mapping to "Simulation and Optimization" sub-component enables the effective monitoring of multiple analytical solutions from Enterprise Tier. This ensures the comprehensive analysis of possible simulations.
- **Platform Tier:** One of the sub-components not foreseen at the beginning, which is currently one of the most important, is "Data Brokering and storage". Thanks to the solutions hosted in this sub-component, such as i4Q^{DR} or i4Q^{DIT}, it will be able to work directly with different clients located in the Edge Tier from the application. Either to test or validate, or simply run some algorithms to see their result using the stored data to train them.

- **Edge Tier:** The i4Q[LRT] mapping to "Distributed Computing" subcomponent. This allows to deploy and run workloads of the different algorithms deployed on the edge. This enables efficient analysis and fast action when reconfiguring the production line.

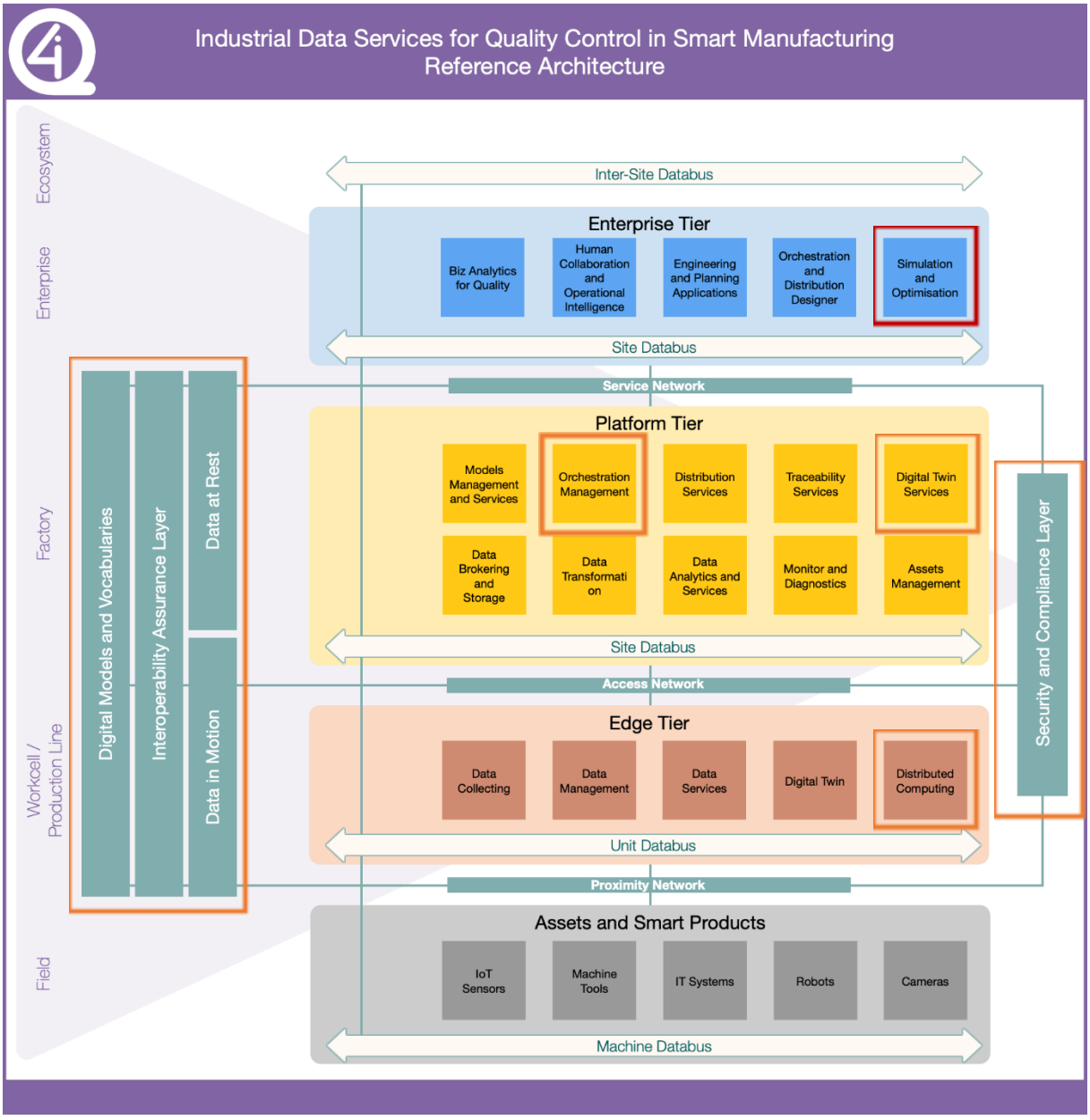


Figure 1. i4Q[LRT] Solution Architecture

3. Implementation Status

3.1 Current implementation

The **i4Q Manufacturing Line Reconfiguration Toolkit** (i4Q^{LRT}) adopts a microservice approach where each optimization algorithm is a standalone (micro-)service that can be deployed and managed at Edge. The deployment strategy relies on containerization technology, specifically using Docker Layers (that allow create a distributable object specifically to instantiate and run an algorithm).

The current implementation of the i4Q^{LRT} Solution is:

- A dockerised file will be provided to implement the interface of the different algorithms. This allows the algorithms to be instantiated and executed as a service. On the other hand, the Mongodb, Mqtt and http service clients have been installed with guicorn.
- An API has been defined to analyse and launch the algorithms.
- A template has been created so that developers can implement an algorithm as a service using this solution.
- Continuous integration and delivery (CI/CD) files have been implemented to automatically analyse software quality, security requirements and to generate a distributable object that can be deployed at runtime.
- The possibility to obtain information about the algorithm deployed through the *metadata.json* file has been implemented.
- The possibility to solve the algorithms by passing the values with an http call has been developed.
- Two optimisation algorithms have been implemented.
- The possibility to Interpret .CSV files has been implemented.
- A specific machine learning model has been developed for Pilot 4 "FACTOR".
- The whole package of files necessary to be able to deploy the algorithms in a Kubernetes environment has been prepared through Helm Charts.
- The possibility of connecting to a message broker to take data continuously from a Kafka client has been implemented.

3.1.1 Solution features analysed and mapping with user requirements

A set of features has already been developed for i4Q^{LRT}, based on the set of user requirements referring to i4Q^{LRT} (Deliverable 1.9) and in line with the functional viewpoints (Deliverable 2.6). Similar requirements have been assigned into common categories of tasks based on an extensive technical study conducted on user requirements, available datasets, etc., introduced to ensure the generalization abilities of the i4Q^{LRT} solution.

- PC1r5.4 “Capability to propose alternative processing parameters to remove chatter presence during the execution of the process” is covered by the feature of being able to get the result of the algorithm. This function runs the algorithm with the input configuration and the defined data taken in the request.
- PC1r6.3 Capability to propose alternative processing parameters to reduce specific equipment components degradation rate” is covered by the feature of being able to get the result of the algorithm. This functionality is responsible for executing the algorithm. All input data shall be considered for the configuration with the different data clients.
- PC2r11 “For each algorithm define in which way the data must be recorded, in terms of sampling frequency, resolution” is covered by the feature of being able to provide metadata about the algorithm. This function provides information about the algorithm, including a description of inputs and outputs.
- PC4r3.4.1 “Acting before a quality deviation happens” is covered by the feature of being able to execute the algorithm. In addition, because sensors will be installed to collect information directly, the action may be taken before the failure occurs.
- PC4r3.4.2 “Improve the process”. The improvement of the process is implicit in the functionality when implementing the solution. It will be possible to access the data, train the models and obtain the best possible configurations.
- PC6r4.1 “Optimization procedure should be converted to autonomous to improve injection machine performance and OEE” is covered by the feature of being able to get the result of the algorithm. This functionality is part of the objective of the algorithm execution. Thanks to the input data, the best decision will be made to improve the process.
- PC6r4.2 “Standardize the solution independently of the equipment that is going to be used” is covered since the solution is developed to be interoperable and to be implemented in any environment.

3.2 Next developments

The solution has now been developed to be deployed as a service on the Edge. In the coming months the following points will be implemented:

- The possibility to instantiate an algorithm by calling the i4Q^{LRT}.
- A frontend for easy handling of the i4Q^{LRT}.
- A guideline for installing the i4Q^{LRT}.
- Further optimisation algorithms in the solution to offer to end-users.
- Covering use cases of different pilots.

3.3 History

Version	Release date	New features
V0.0.1	30/11/2021	Backend OpenApi
V0.0.2	20/12/2021	Generated continuous integration files for the automatic build of docker images.
V0.0.3	21/12/2021	Documentation
V0.0.4	24/12/2021	Tools for automatic quality and safety analysis have been implemented to automatically analyse software quality, safety, and security requirements.
V0.0.5	31/01/2022	Update backend base image and add first algorithm as a micro-service (manpower scheduling)
V0.0.6	01/02/2022	Update Backend Imagen Security – CVE-2018-0269
V0.0.7	07/02/2022	Add mjanaive algorithm
V0.0.8	14/02/2022	Add Kubernetes environment with Helm Charts
V0.0.9	01/04/2022	Refactoring and add Update backend Image. A core with multiple clients has been implemented so that the solution can be used individually in Kubernetes.
V0.0.10	16/04/2022	Add a specific machine learning model
V0.0.11	13/05/2022	Add connection with broker message (kafka consumer)

Table 1. i4Q[LRT] Version history

4. Conclusions

Deliverable D5.5 Manufacturing Reconfiguration Toolkit is a technical specification document, providing an in-depth technical overview of the i4Q^{LRT} solution. It describes in detail the role, the functionalities, and the conceptual architecture of i4Q^{LRT}. It presents a study detailing the main features of the solution to clarify the key functionalities and objectives of the i4Q^{LRT} solution, describing its architecture diagram with respect to i4Q Reference Architecture.

The current implementation status of i4Q^{LRT} is detailed thoroughly, presenting the significant progress of this overall development. This document presents these approaches which include 1) the pilots requirements analysis and engineering to clarify the technical specifications, 2) the technical studies conducted to define the solution's abstract level architecture 3) the input and output definition of the solution, 4) the objectives of i4Q^{LRT} .

Appendix I

The PDF version of the **i4Q Manufacturing Line Reconfiguration Toolkit** (i4Q^{LRT}) web documentation can be accessed online at: **http://i4q.upv.es/21_i4Q_LRT/index.html**

i4Q Manufacturing Line Reconfiguration Toolkit

General Description

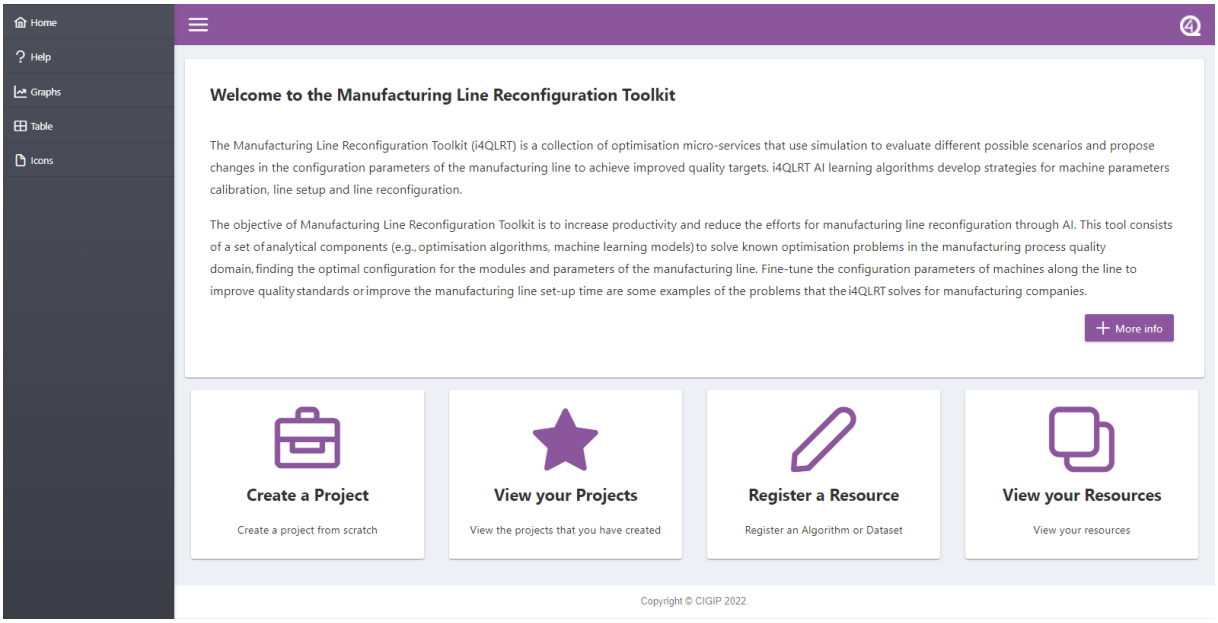
i4Q^{LRT} is a collection of optimisation micro-services that use simulation to evaluate different possible scenarios and propose changes in the configuration parameters of the manufacturing line to achieve improved quality targets. i4Q^{LRT} AI learning algorithms develop strategies for machine parameters calibration, line setup and line reconfiguration.

The objective of Manufacturing Line Reconfiguration Toolkit is to increase productivity and reduce the efforts for manufacturing line reconfiguration through AI. This tool consists of a set of analytical components (e.g., optimisation algorithms, machine learning models) to solve known optimisation problems in the manufacturing process quality domain, finding the optimal configuration for the modules and parameters of the manufacturing line. Fine-tune the configuration parameters of machines along the line to improve quality standards or improve the manufacturing line set-up time are some examples of the problems that the i4Q^{LRT} solves for manufacturing companies.

Features

1. It provides the possibility to deploy algorithms at the edge of the network, via distributable files.
2. Provides informations necessary to understand the use, characteristics and needs of the algorithm.
3. Offers the ability to instantiate an algorithm.
4. Allows algorithms to be run and solved at the edge of the network, obtaining information directly from the production line. In this way, the algorithms can optimise the production line automatically.

ScreenShots



Commercial Information

Authors

Partner	Role	Website	Logo
UPV	Leader	https://www.upv.es	UNIVERSITAT POLITÈCNICA DE VALÈNCIA
IKERLAN	Vice-Leader	https://www.ikerlan.es/en/	ikerlan MEMBER OF BASQUE RESEARCH & TECHNOLOGY ALLIANCE
EXOS	Participant	https://exos-solutions.com/	EXOS Operational Consulting

License

The licence is per subscription of 150€ per month.

Pricing

Subject	Value
Installation	60€ (5-20 hours)
Customisation	60€ (10-50 hours)
Training	60€ (8-24 hours)

Associated i4Q Solutions

Required

- Can operate without the need for another i4Q solution

Optional

- i4Q Data Repository
- i4Q Data Integration and Transformation Services
- i4Q Edge Workloads Placement and Deployment
- i4Q Digital Twin simulation services

System Requirements

docker requirements:

- 4 GB Ram
- 64-bit operating system
- Hardware virtualisation support

API Specification

Resource	POST	GET	PUT	DELETE
/metadata	Not Supported	Provides a description of the Algorithm, including a description of inputs and outputs	Not Supported	Not Supported
/result	Run the algorithm with the current input and output configuration	Not Supported	Not Supported	Not Supported
/subscription	Supported	Supported	Supported	Supported

Resource	POST	GET	PUT	DELETE
/instanciate	Supported	Supported	Supported	Supported

Installation Guidelines

Resource	Location
Last release (v.1.0.0)	Link
Related Datasets	Link
API Sec	Link
Video	Link

Installation

This component is designed to be used by both docker and kubernetes.

- Docker environment:

  For docker-based installation it can be achieved by running a Docker-compose command. For each data processing algorithm, a separate Docker-compose file is provided. For example, the following command starts an instance of the component

```
cd orchestration
docker-compose -f docker-compose.mjanaive.yaml up --build --remove-orphans
```

- Kubernetes environment:
  To deploy it on Kubernetes, helm files are prepared, which can be launched from commands. For example, the following command starts an instance of the Machine-Job Assignment Naive Algorithm.

```
cd charts
cd mjanaive
helm install mjanaive ./ --namespace mjanaive --set
defaultSettings.registrySecret=i4q-lrt-
mjanaive,privateRegistry.registryUrl=registry.gitlab.com,privateRegistry.re
gistryUser=user.gitlab,privateRegistry.registryPasswd=my-password-or-token
```

User Manual

How to use

The API functionality can easily be tested using the Swagger UI, which is currently deployed with the Flask application.

```
{
    {
        "inputs": [{
            "id": "string",
            "format": "string",
            "*format*": {
                //DEPENDING FORMAT STRUCTURE
            }
        }],
        "module_parameters": [{
            "id": "string",
            "format": "string",
        }],
        "outputs": [{
            "id": "string",
            "format": "string",
            "*format*": {
                //DEPENDING FORMAT STRUCTURE
            }
        }]
    }
}
```

The JSON body of the POST request to /result consists of three keys ('inputs', 'outputs' and 'module_parameters') which are lists of JSON objects with the following structure:

- **id**: Identifier of the input or output parameter, specified in the metadata of the component
    - For input/module_parameters: The name of the argument of the "compute" function (of computeUnit) to which the input should be passed
    - For output: The computeUnit returns the results as a python dictionary. The "id" refers to the key of the returned dictionary
- **format**: Specifies the format of the input or output data parameter. The values of this key can be "filename", "mongo-collection", "collection", "mqtt"
- **depending** on the value of key "format", (in the schema above), the key <format> should be replaced with one of the following keys:
    - **filename**: (Optional) For file/csv inputs, specifies the file name of the file containing the input data. Currently the files must be available in a Docker-volume made available using Docker-compose. Later the CSV files will be in the Storage component
    - **mongo-collection**: (Optional) For database/mongo inputs. This config is a JSON that specifies the 'mongo_uri' and the 'query' details:

        - For input:
            - **mongo_uri**: Used to connect to the mongoDB
            - (Optional) **operation**: Either 'find' or 'aggregate', defines which operation is used to retrieve the data. Default is 'find' if not present
            - **collection_name**: The name of the mongo collection on which the operation is applied
        - For output:
            - **mongo_uri**: Used to connect to the mongoDB
            - **collection_name**: The name of the mongo collection under which the data is stored. Note: The data is stored using mongoDBs 'insert_many'
            - **collection**: (Optional) For collection data, contains the collection (data) in json format
- **mqtt**: (Optional) For message bus data. This config is a JSON that contains the message bus topic to subscribe/publish to as well as an optional ring buffer length:
    - For input:
        - **topic**: The name of message bus topic
        - (Optional) **historyLength**: for ring buffer (which stores the stream data) length. Default is 1
    - For output:
        - topic: The name of message bus topic
- **kafka**: (Optional) For broker message kafka. This config is a JSON that contains the message broker topic, the server, group identification and histry limit.
    - For input:
        - **topic**: The name of message broker topic.
        - **limit**: for ring buffer length.
        - **server**: The address of kafka's server.
        - **group**: The group maintains its offset per topic partition.
    - For output:
        - **topic**: The name of the message broker topic where you want to dump the new information.
        - **server**: The address of kafka's server.
        - **group**: The group maintains its offset per topic partition.