ReMAP

Real-time Condition-based Maintenance for Adaptive Aircraft Maintenance Planning

Deliverable D8.3

Integrated ReMAP IFHM framework

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769288

Document History

Revision Nr	Description	Author	Review	Date
V0.1	Initial draft - Report structure, initial description	Miguel Ángel Esbrí (ATOS)	--	13-09-2021
V0.2	Integration and deployment	Iván Carrillo (ATOS)	-	20-09-2021
V0.3	Review, figures, section 5	Floris Freeman (KLM)		27-09-2021
V0.5	Updated health indicator model subsection	Luis Basora (ONERA)		28-09-2021
V1.0	Final QA	Bruno Santos (TUD)		29-09-2021

Index

1. Introduction

1.1. Project Summary

ReMAP "Real-time Condition-based Maintenance for adaptive Aircraft Maintenance Planning" (hereinafter also referred as "ReMAP" or "the project"), is a European project started on the 1st of June 2018 and has a duration of four years. The project addresses the specific challenge to take a step forward into the adoption of Condition-Based Maintenance in the aviation sector. In order to achieve this, a data-driven approach will be implemented, based on hybrid machine learning & physics-based algorithms for systems and data-driven probabilistic algorithms for systems and structures. A similar approach will be followed to develop a maintenance management optimisation solution capable of adapting to the real-time health conditions of the aircraft fleet. These algorithms will run on an open-source IT platform for adaptive fleet maintenance management. The proposed Condition-Based Maintenance solution will be evaluated according to a safety risk assessment, ensuring its reliable implementation and promoting an informed discussion on regulatory challenges and concrete actions towards the certification of Condition-Based Maintenance.

1.2. Scope of this deliverable

This document is the deliverable D8.3 of the ReMAP project and describes the work performed in the WP8 task "T8.2 – Technology integration and deployment", which addresses the integration of the individual technologies developed in WP5 and WP6 into the IT platform in preparation for demonstration tests in IVV scenarios 1 and 2.

1.3. Goals

This document aims to describe how the integration of the different technologies used in the project into the ReMAP IT platform follows the requirements (Task 2.1), architecture design (Task 2.2) and the API protocols and SDK (Task 2.3) defined in WP2, and enables the seamless deployment, integration and execution of WP5 (health prognosis tasks) and WP6 (maintenance & scheduling tasks) models by interacting with ReMAP IT platform via the User Interface frontend and the SDK library.

The focus of the work carried out in this task it is centred on the interoperability between the technologies, including the implementation of the APIs for data extraction and sharing, normalisation of data into an interchangeable format, and communication with the maintenance decision support tool interface (Task 6.4).

This work paves the way for the upcoming planned demonstration activities in WP8 in the next months (as devised in the DoA), which includes the deployment of the ReMAP IT platform (the node module) with the technologies and services that compose the ReMAP IFHM solution, within KLM's IT infrastructure.

1.4. Document structure

The overview of the ReMAP architecture functional blocks (as designed in WP2, deliverable "D2.3 ReMAP System Architecture") is presented in **Section 2**. **Section 3** presents the actual integration of the WP5 and WP6 technologies and models into the ReMAP IT platform. **Section 4** presents the planned (and ongoing) deployment task of ReMAP core modules into KLM's IT infrastructure in preparation for the WP8 Demonstration activities in the next months. Validation exercises will be described in **Section 5**. Finally, conclusions and lessons learned are discussed in **Section 6**.

2. ReMAP IT Architecture

2.1. Overview

The image below (Figure 1) depicts the high-level view of the ReMAP components. As shown, there are two main high-level components: Node and Core, which are composed of internal modules covering concrete functionalities.

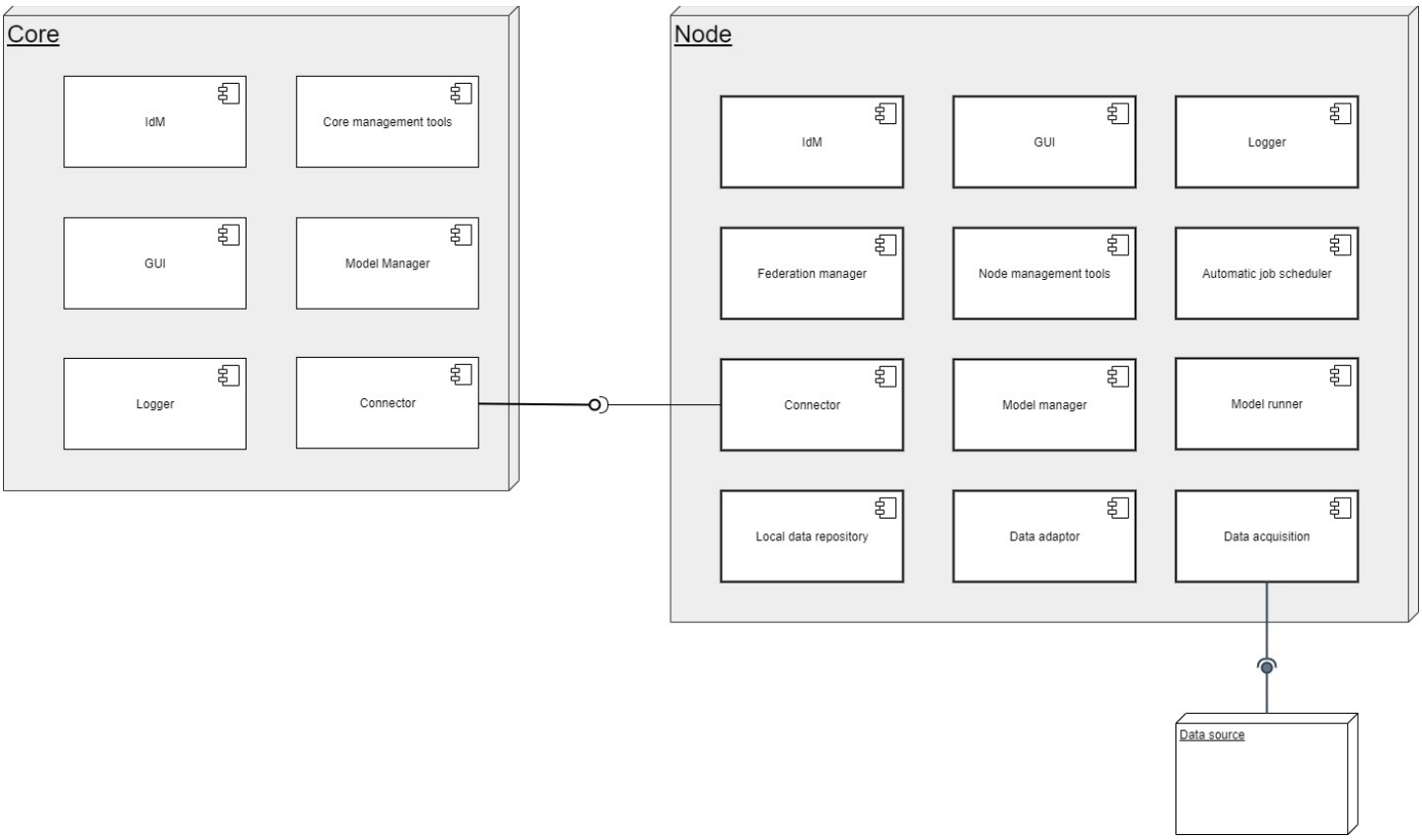

Figure 1. ReMAP high-level architecture

2.2. Core module

The Core module is the central point of the ReMAP IT Platform. It configures the platform, manages the airline's Nodes belonging to ReMAP, and centralises the generation of the models. It also distributes models between the nodes to train and test them with the airline data. It also standardises the data within the platform through a centralised catalogue of aircraft, components, parameters, and tasks.

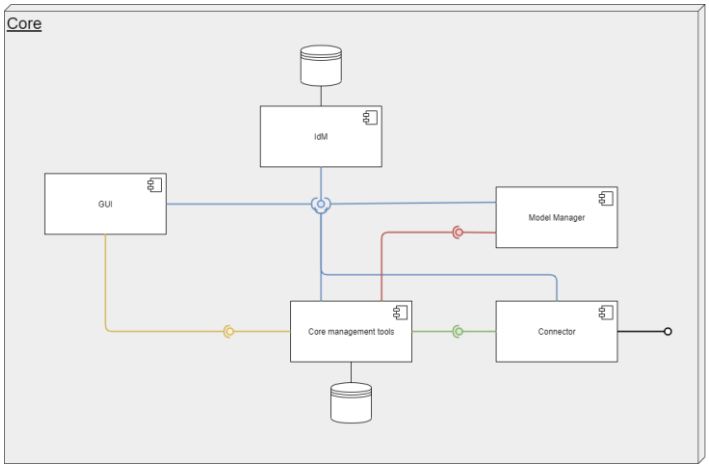

Figure 2. ReMAP core modules interfaces

2.3. Node module

The ReMAP Node is the client infrastructure of the ReMAP distributed architecture, the spoke of the hub&spoke model. It will be deployed inside the airline IT infrastructure to protect and avoid exposing airline data. The node is just a module, but each airline will create an instance of it, creating a network in which all Nodes are connected through the Core.

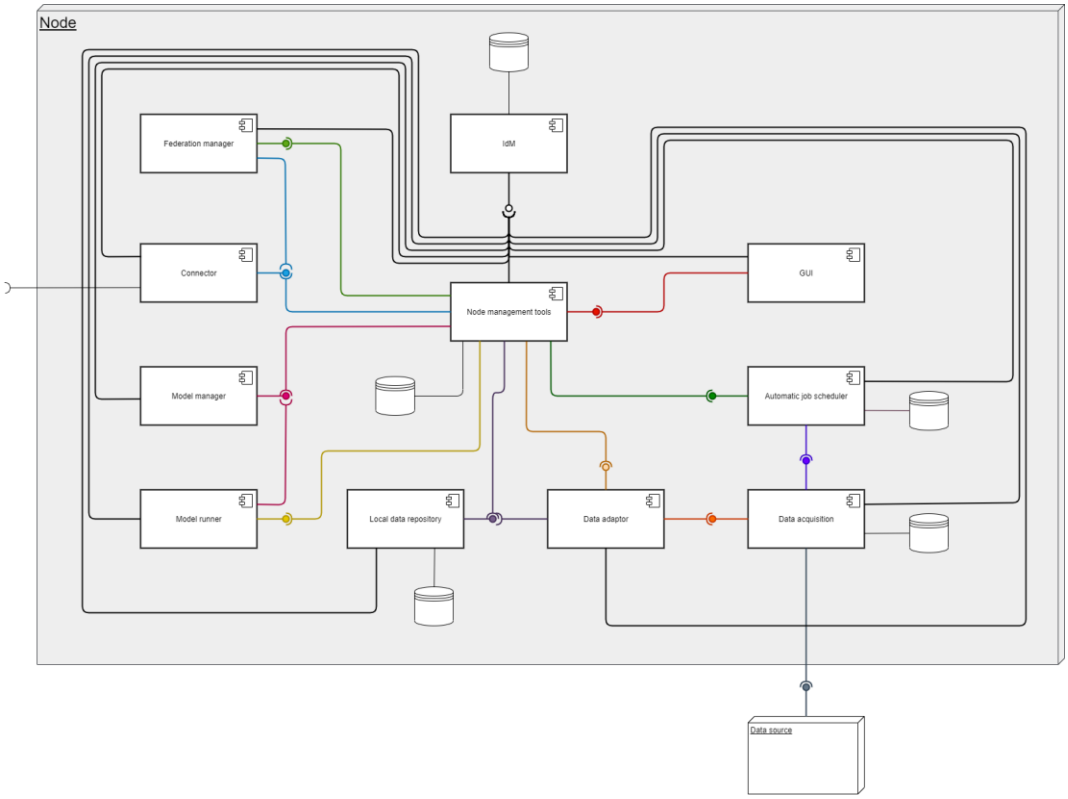

Figure 3. ReMAP Airline Node modules interfaces

3. Integration of technologies and models

3.1. ReMAP IT platform technological approach

In this section, the main modules involved in the integration of external systems and models are described.

3.1.1. Connector

As described in the previous section, the platform is split into "core" and "node" following a hub & spoke architecture. All connections between the nodes and the Core are centralised in a connector module, which handles authorisation and redirects the dataflows to the appropriate internal ReMAP modules.

3.1.2. Models catalogue & PyPi Server

A model catalogue is included in the Core, so that model builders can upload their models as a wheel ('.whl') python package via the web GUI defined in ReMAP. An internal, private PyPi server is included in the Core in order to store, distribute, and manage the models within the platform. While this private repository is not accessible by model builders directly, the internal ReMAP core modules do have access to this service to store, update or delete models as instructed by model builders from the GUI.

On the airline's node side, airline users can browse the model catalogue from their node's web GUI and request the download of any models that they plan to execute on their fleet. The models downloaded from the Core's central model catalogue are stored in a parallel private PyPi server and deployed with each node. Airline staff can then schedule the execution of any models previously downloaded.

3.1.3. SFTP Server

It is important to understand how airline data is managed in ReMAP to understand the limitations and constraints in place for models to process this data.

The central idea behind data management in ReMAP is to distribute models instead of data and have models executed locally on each node so that airline data never leaves the node. Therefore, model builders need to define the required inputs to their model (according to the ReMAP catalogue) when they upload it to the Core.

By looking at this list of required inputs for a given model, airlines can configure the appropriate datasets by selecting the appropriate parameters for which the platform will export data once execution is requested. But how does the platform acquire such data from the airline's internal systems in the first place?

In order to connect input flows from the airline systems, an SFTP connector has been defined in the node, which periodically downloads any new input files supplied by the airline and converts them to the ReMAP standard model for storage in the local node database. The sFTP server may be located on a secure and isolated environment within the airline's infrastructure for security

reasons. The airline staff needs to define the connection parameters (url, credentials, folder) as well as the metadata (mapping between each column in the input CSVs and the appropriate parameter according to the ReMAP parameter).

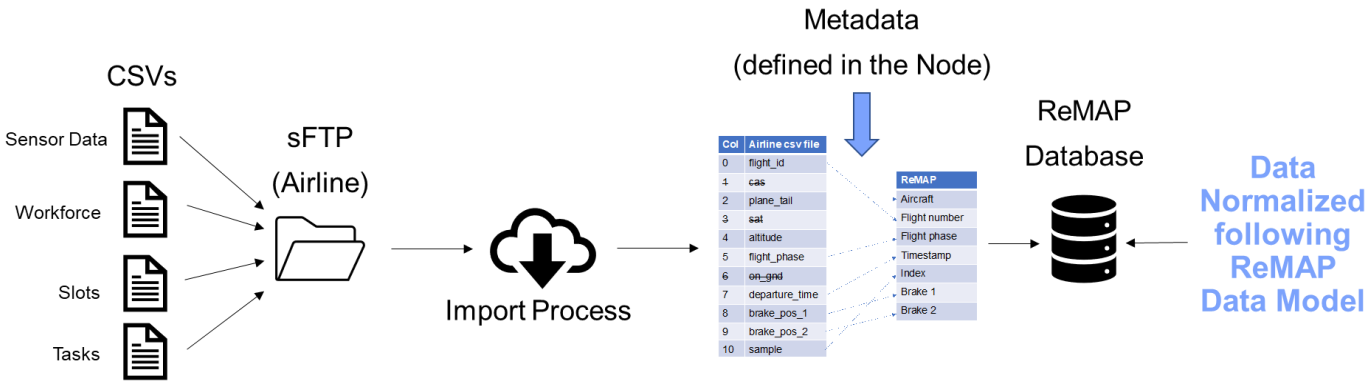

Figure 4. Data management in the ReMAP platform

3.1.4. Model execution

In order to execute a given model with the appropriate input as specified by the airline, the platform generates an export of the dataset and spawns a Docker container that will run the model.

Starting from a python-based Docker image, the platform adds the dataset, the model, as well as some configuration data and starts a container that will execute the model in this controlled environment. Using the ReMAP SDK, models can access the datasets provided and export their output back to the platform in a standardised way.

Models' output is then stored in the airline's node and can be used to operate the platform (e.g., define prognosis tasks from RUL estimation), or even exported in CSV format for later analysis with external tools, such as visualisation tool being developed in WP6 for planner algorithms output.

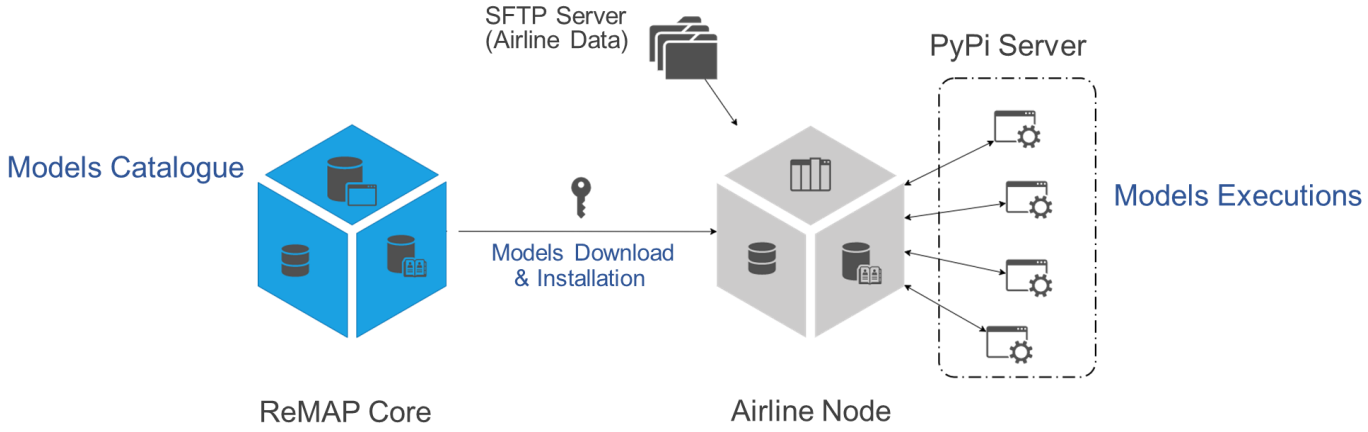

Figure 5. External integration overview

3.2. Models supported in the ReMAP platform

Models integrated into the ReMAP platform are split into 3 categories: "RUL Estimation" (WP5), "Health Indicator" (WP5) and "Planners" (WP6). Whereas the former two categories aim to assess and predict an aircraft component's health status, the latter category aims to plan the maintenance schedule in an optimal way.

3.2.1. RUL Estimation

3.2.1.1. Input

RUL Estimation models receive as input a dataset that contains aircraft sensor-data (defined by the airline staff following the required input listed by the model builder) in CSV format, along with some additional metadata which they can access via the ReMAP SDK, in order to compute a remaining useful life estimation for a given component.

3.2.1.2. Output

- RULValue: The remaining useful life as estimated by the algorithm
- RULUnit: The unit for the RUL estimated, either cycles or flight hours
- confidenceInterval: The upper and lower bound of the confidence interval for this estimation
- distributionType and distributionParams: These variables describe the statistical distribution for the RUL
- failureMode: mode of failure estimated

3.2.2. Health Indicator

3.2.2.1. Input

This model receives the same input as RUL Estimation models.

3.2.2.2. Output

This type of model returns a health indicator score for each flight in the dataset:

- flightNo: The flight number/id for this score
- departure_time: The departure time of the flight
- score: Health Indicator score, e.g. it can be the anomaly score for the models using anomaly detection techniques
- status: The status associated with this score, e.g. faulty/healthy when score is higher/lower than a predefined threshold

3.2.3. Planners

The main goal of the planning algorithms is to create a feasible maintenance schedule that maximises (minimises) one or more KPIs such as 1) ground-time usage, 2) cost, 3) robustness, etc. To do so, the planning algorithms assign tasks to maintenance slots.

3.2.3.1. Input

Planner algorithms, as opposed to RUL and health indicator models, have no configurable input. Instead, these kinds of models will always receive the most recent data concerning the maintenance schedule of the airline:

- Non-prognostic tasks: latest task-list as imported from the airline's sftp
- Slots: latest slots-list as imported from the airline's sftp
- Workforce: latest workforce availability list as imported from the airline's sftp
- Prognostic tasks: list of prognostic tasks exported from the node
- RUL estimations/Health indicators: list of RUL and health indicator outputs computed in the node

3.2.3.2. Output

Planner algorithms will process the inputs provided and export an updated version of the tasks, slots, and workforce files as an output. This execution will be registered in the platform so that airline staff can review the models' proposed schedule.

4. Integration and deployment into KLM's IT infrastructure

While this document is being redacted, the consortium is finalising the last steps towards the deployment and configuration of a ReMAP node in KLM's infrastructure, to be federated with a core hosted and managed by ATOS, as illustrated by the figure below:

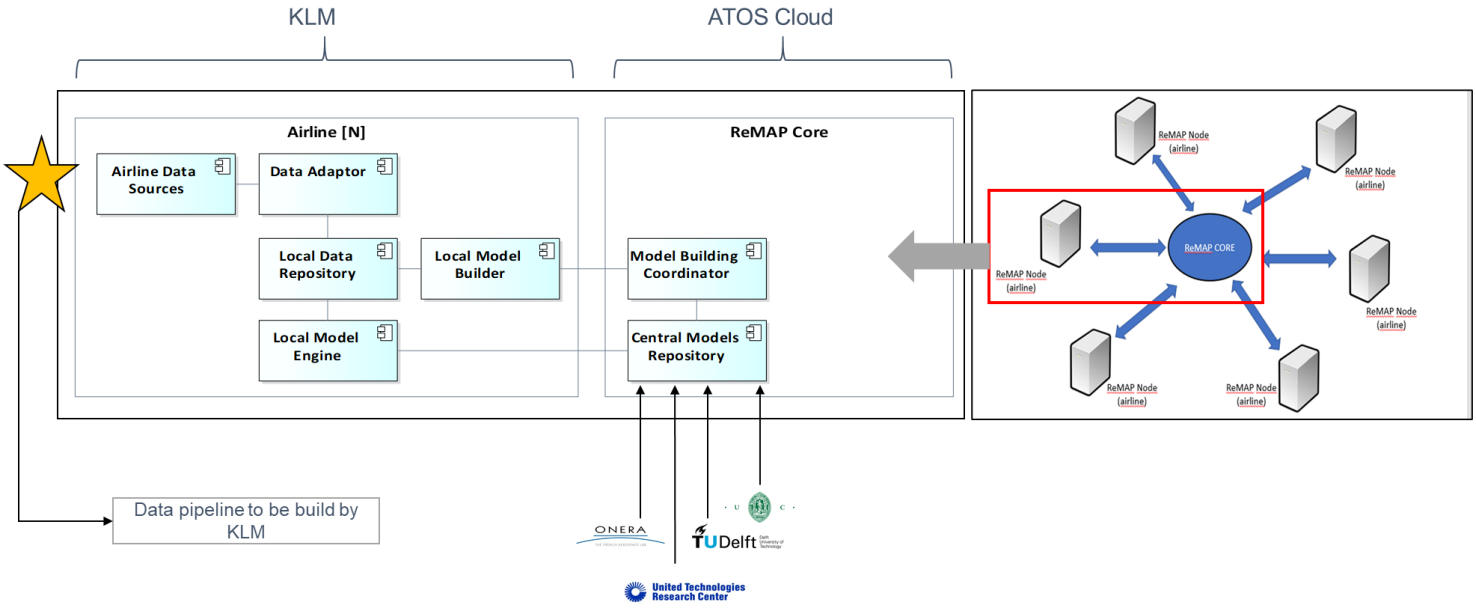

Figure 6. KLM-ATOS deployment of ReMAP

While it might not be obvious at first glance, the deployment of the ReMAP node within KLM's infrastructure raises issues concerning the security and confidentiality of the airline's systems and data.

On the one hand, the deployment of the ReMAP node itself has been kept as simplified as possible:

- Docker and docker-compose are the only software requirements for the machine hosting the node
- All configuration is done by standard docker practices, .env files for parametrising the configuration (domain, protocol, certificates…) and docker secrets for assigning credentials to the different services
- Nodes synchronise the catalogue automatically from the Core they are federated to
- Datasources defined as SFTP connections

On the other hand, on the airline side, the appropriate internal systems must be connected to an SFTP server to export the data that the node will periodically import to its database. Since this implies that the node will be at some level sharing an intranet with the airline systems, there's been a big focus on security when defining the different dataflows in the ReMAP architecture, from the one-way communication flow from node to Core (and not the other way around), to the aggregation of all these data flows in a single, secured connector module, to the way python models are executed in a "sandbox-like" docker container environment.

In this sense, the main issues where the consortium has focused the integration efforts while fine-tuning the developed assets for this deployment can be summed up as:

- Adaptation of KLM internal system dataflows to provide flight and maintenance data required for ReMAP
- Integration of models with the ReMAP SDK, data model and packaging constraints
- In-depth review of ReMAP's dataflows and access control to accommodate airline's security constraints

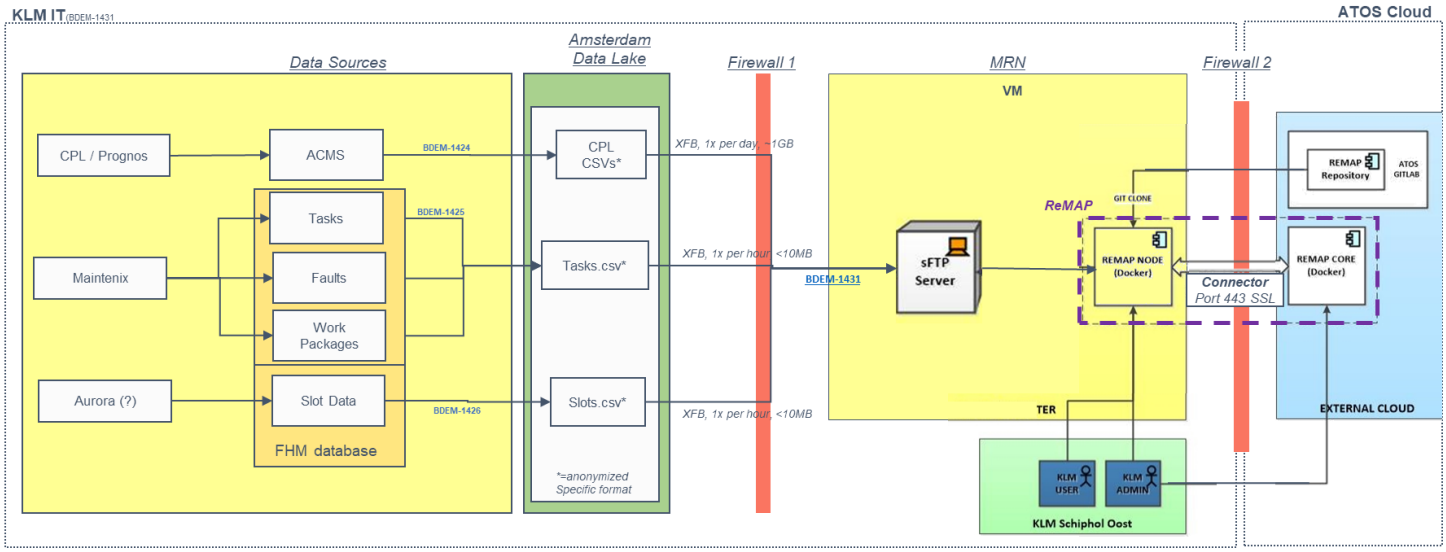

Figure 7. KLM systems - node integration. Detailed view

5. Vallidation & Verification

In M30, the *Laboratory validation and integration test* proved that model builders could package their algorithms and upload it to the central Core for later execution on a node. By means of synthetic data, it was shown that these models were capable of running with (near) real-time data in an early version of the IT Platform. Milestone 7 was achieved by means of this validation. The final validation of the algorithm will take place in a real-life operational environment with real-time data during the *Technology full demonstration and validation*. The scope of the validation has been presented in D8.1, and a benchmarking exercise was completed by means of D8.2

6. Conclusion

This report has given a very high-level view of how the ReMAP architecture has been defined for new airlines deploying their node, model builders adapting to ReMAP SDK and packaging workflow to distribute their models, and the technologies behind the platform enabling such integration. In the last section, we've discussed the upcoming KLM node deployment and the main challenges that the consortium has had to overcome to make this possible while preparing the ground for future airlines that will choose to follow KLM's steps and join as a ReMAP federated node.

In the next months, deliverables D2.4 and D2.5 will greatly complement the technical descriptions of the platform provided here. At the same time, D8.4 will describe the pilot deployment carried out with the setup of KLM's node.

7. References

[1] D2.3 ReMAP System Architecture.