WIKI LAB

Kepler.AI

Intelligent Risk and Compliance Management Platform

AI-Powered Solution: Faster, Smarter, More Reliable

✔ Real-time fraud detection

✔ Next-generation risk assessment

✔ Automated regulatory compliance

Version 07 / July 2025

Onglet 1 : Présentation générale **4**
GENERALITES 5
I. Architecture Globale du LAB 5
II. Contributeurs 6
III. Datasets 7
IV. Scénario 7
CRÉATION DES CAS D'USAGE 9
I. Accès au module Administration 9
1. Gestion des Use Cases 9
2. Gestion des Utilisateurs : 10
3. Paramètres 10
4. Journaux (Logs) 11
II. Configuration d'un Nouveau Use Case 11
GESTION DES DATASETS 14
I. Ingestion des données 14
II. Script d'ingestion (Job) 17
CRÉATION DE SCÉNARIO 20
I. Étapes de création de scénario: 20
II. Utilisation d'un Scénario IA 23
1. Interface de Monitoring de l'Instance 23
2. Création de Jobs Automatisés 26
3. Importation des Datasets 27
Onglet 2 : Ingestion des données **31**
DATA INGESTION 32
Introduction 32
I. Ingestion manuelle via l'interface Kepler 32
A. Ingestion depuis un fichier 33
B. Ingestion depuis une base de données 34
II. Ingestion automatique via Script (Job planifié) 35
III. Cycle de Vie du Script 37
IV. Avantages d'une ingestion centralisée dans Kepler 38
V. Extraction des Données 39
1. Mode append (incrémental) 39
2. Mode overwrite (écrasement complet) 39
Conclusion 40
Onglet 3 : Préparation des données **41**
DATA PREPARATION 42
Introduction 42
I. Étapes de préparation dans Jupyter 42
1. Accès à l'environnement 42
2. Développement des notebooks 44
3. Visualisation et vérification 44

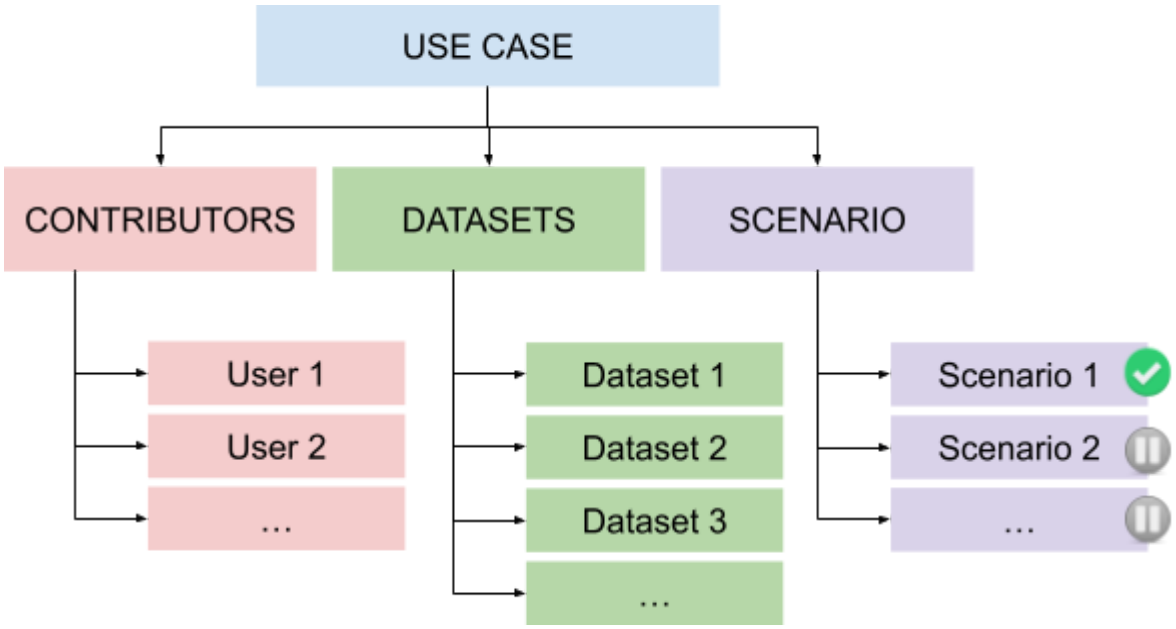
II. Automatisation avec Cron Job 44
1. Transformation du notebook en script exécutable 44
2. Création du Cron Job 45
3. Paramètres à définir 45
4. Enregistrement 46
Conclusion 46
Onglet 4: Entraînement des données 47
DATA TRAINING 48
Introduction 48
I. Phase manuelle dans Jupyter 48
1. Accès à l'environnement Jupyter via KEPLER : 48
2. Travail dans le répertoire training 49
II. Création d'un Job d'entraînement automatisé 50
1. Création du Job dans KEPLER 50
III. Objectifs de l'étape de Training 51
IV. Visualisation et ajustements 52
Conclusion 52
Onglet 5: Scorage des données 53
DATA SCORING 54
Introduction 54
I. Phase manuelle dans Jupyter 54
II. Automatisation avec un Cron Job 55
III. Objectifs de l'étape de Scoring 56
IV. Enregistrement et automatisation 59
Conclusion 59

Par tanitlab AI

Onglet 1 : Overview

GENERAL OVERVIEW

I. Global Architecture of the LAB



Each Use Case in **KeplerAI** acts as an autonomous segment, encompassing access rights, data, and algorithms. A Use Case is composed of three main components:

1. **Contributors**
    - A list of users associated with the Use Case (e.g., User1: Data Scientist, User2: AML Analyst).
    - Permissions are managed via the *Administration* module.
2. **Datasets**
    - Collections of data specific to each Use Case (e.g., Dataset1: Transactions_2023, Dataset2: Client_Profiles).
    - Supported formats: CSV, JSON, Parquet (maximum 5GB per file).
3. **Scenarios**
    - Workflows d'analyse ou de modélisation (ex: *Scenario1: Détection d'Anomalies, Scenario2: Scoring Clients*).
    - Each scenario includes :
        - One or more dedicated Jupyter notebooks.
        - Configurable Spark resources (CPU/RAM).

- **Only one active** scenario is allowed per Use Case.

II. Contributors

The Contributors module enables efficient management of user access and permissions within **KeplerAI**.

Key Features :

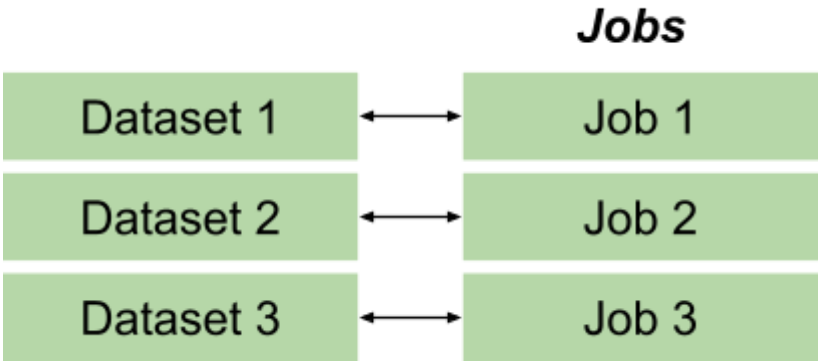
- Visual and Intuitive Role Assignment

	Admin	Devops	Manager	Standard
User 1	X	X		
User 2			X	
…				X

- Four access levels available:
  - **Admin** – Full access to all settings
  - **DevOps** – Technical deployment and monitoring
  - **Manager** – Project validation and tracking
  - **Standard** – Basic feature usage

III. Datasets

In KeplerAI, datasets are structured data collections specific to each use case. Each dataset is processed through dedicated jobs to ensure optimal transformation.



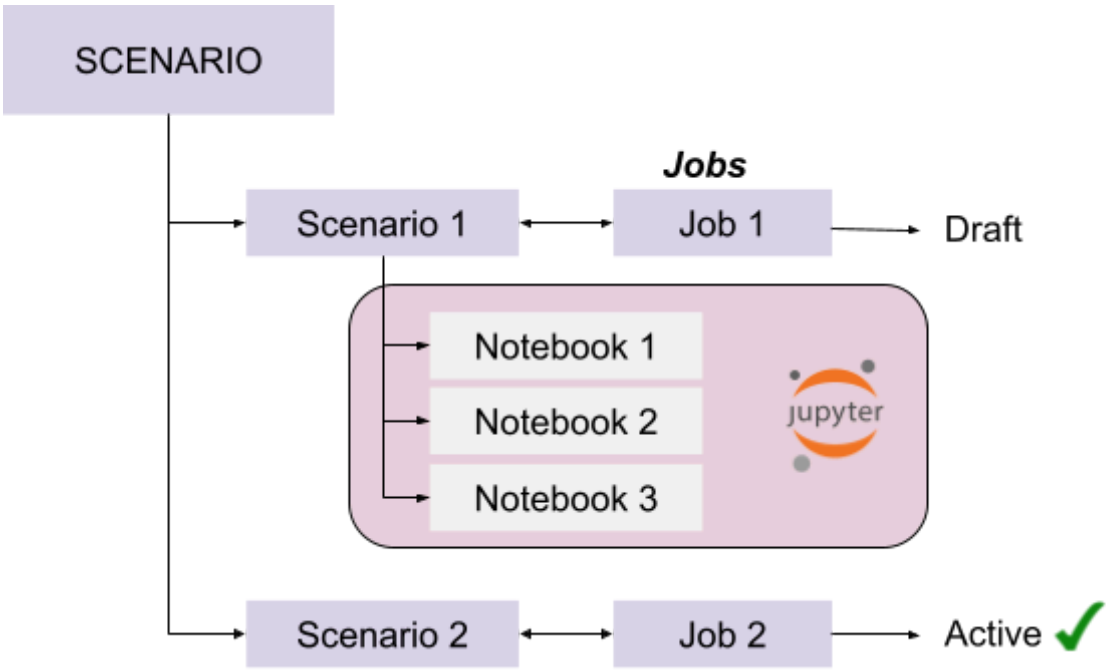
These jobs can also be used to regularly update the source data.

IV. Scenario

This refers to a test and development scenario. The platform allows testing of new models without impacting the one currently active in production.

Only one scenario can be active per use case.

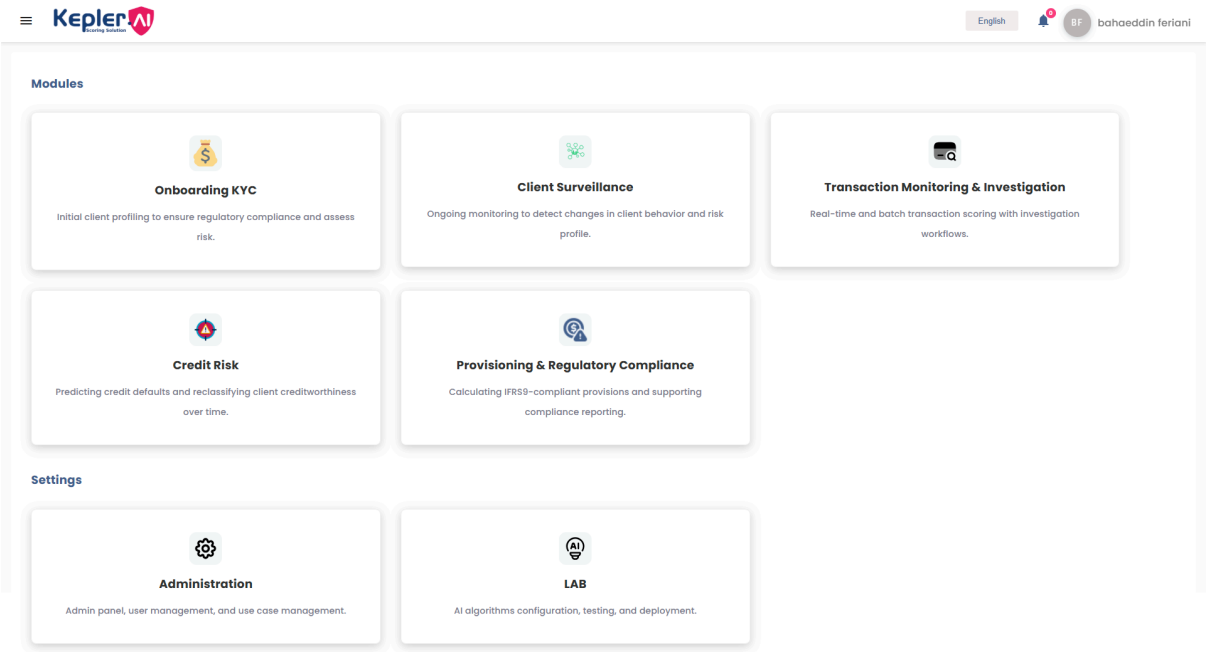
A scenario combines multiple Jupyter notebooks into a single job, generating either a **Draft** (testable version) or **Active** (deployed in production) output.



Par tanitlab AI

USE CASE CREATION

Before any data modeling or exploration, the initial step in **KeplerAI** is to create a **Use Case** via the **Administration** module. This step structures future work and manages access rights.



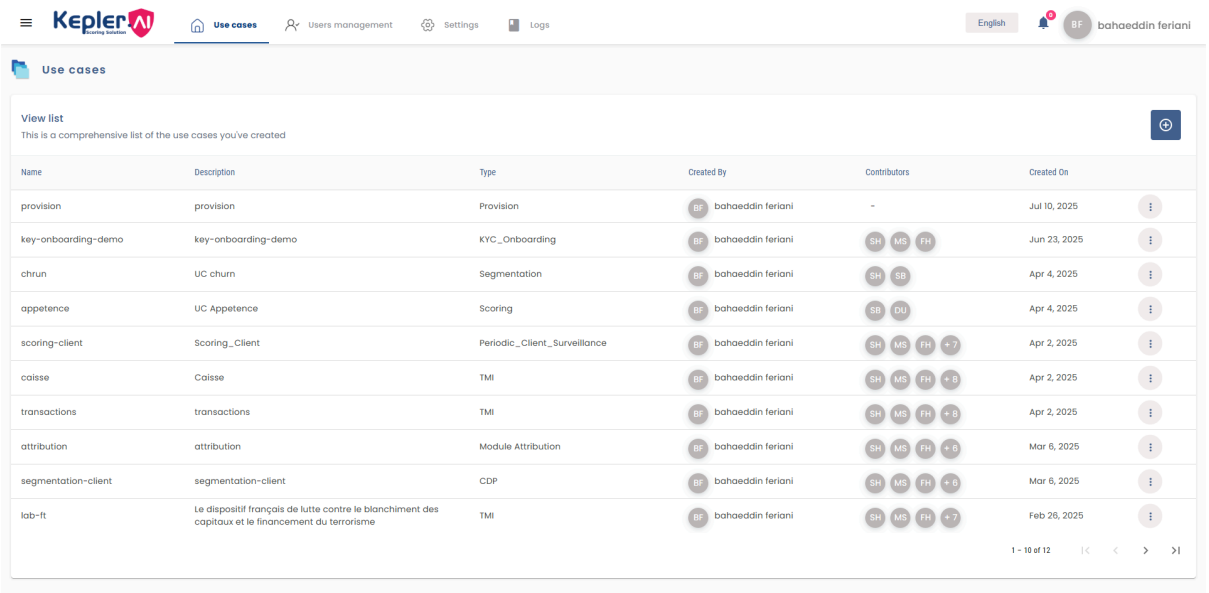
I. Access to the Administration Module

The administrator can access the **Administration** module from the main menu. This module centralizes the following key features:

1. Use Case Management

- **Complete List**: Displays all created use cases.
- **Create Button**: Allows the addition of a new Use Case.

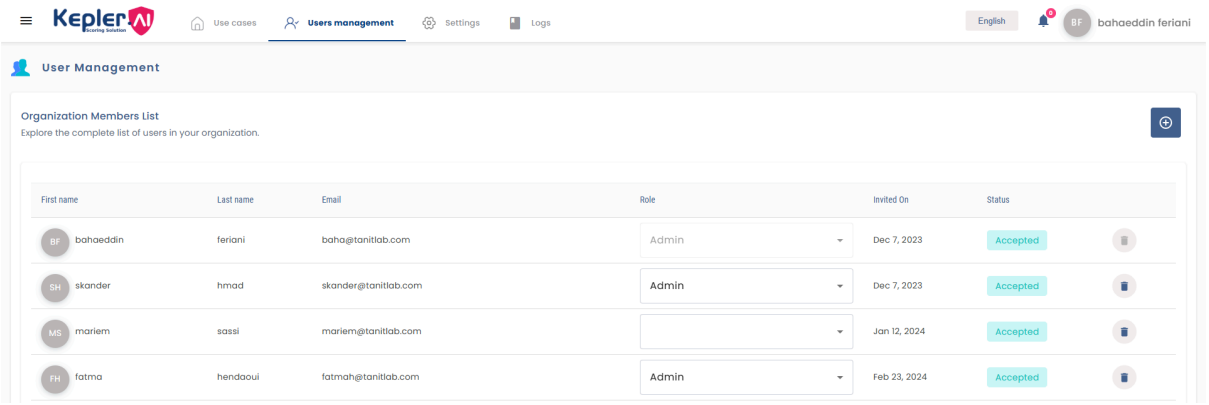
Administration>Use cases



2. User Management

- Add, modify, or delete users.
- Assign or update roles (e.g., Admin, AML Auditor, etc.).

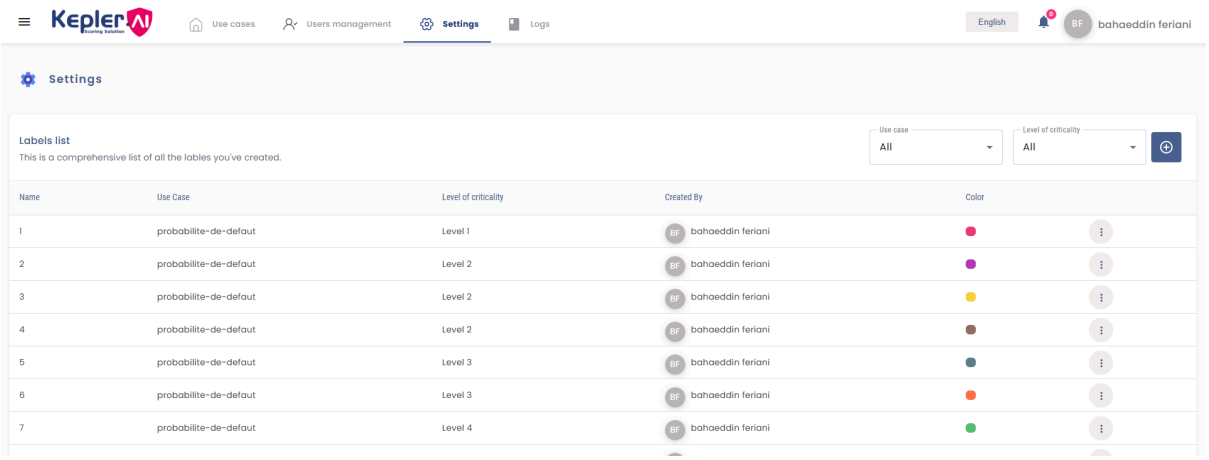
Administration>Users management



3. Settings

- **Custom Labels**: Create specific labels tailored to Use Cases and criticality levels.

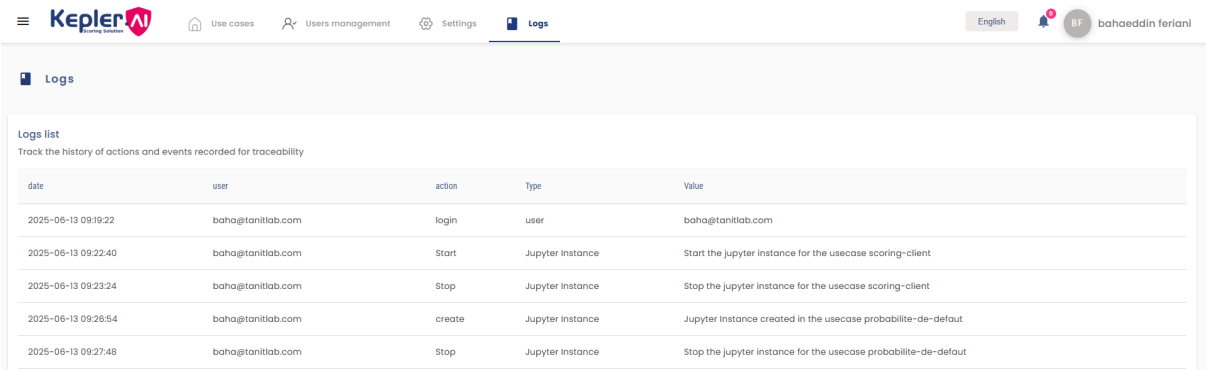
Administration>Settings



4. Logs

- Track historical actions and events for auditability (e.g., login attempts, modifications, etc.).

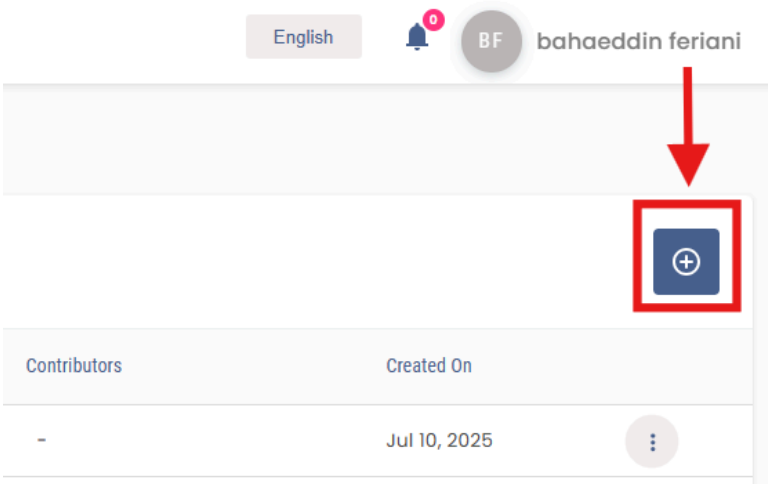
Administration>Logs



II. Configuring a New Use Case

From the **Administration** menu, the administrator clicks the "➕" button (located at the top left of the screen) to initiate the creation of a new Use Case.

Administration>Use cases



A configuration window opens, allowing the administrator to fill in the following fields:

- **Use Case Name**: A clear and unique title to identify the use case.
- **Description**: A brief explanation of the use case (e.g., Detection of Atypical Transactions).
- **Use Case Type**: The administrator selects the relevant category from:
    - Onboarding KYC (Know Your Customer)
    - Client Surveillance
    - Transaction Monitoring & Investigation
    - Credit Risk
    - Provisioning & Regulatory Compliance

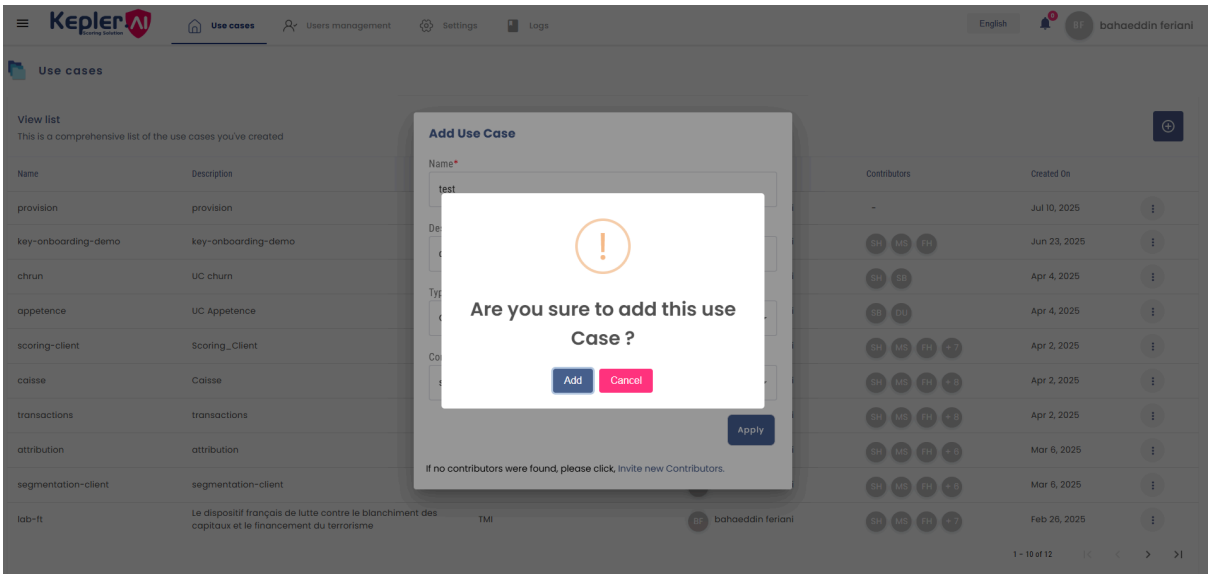
- **Contributors**: il s'agit des utilisateurs (Data Scientists, Analysts…) qui auront accès à ce Use Case. Ils pourront visualiser les données, créer des jobs de traitement, lancer des modèles, et suivre les alertes dans la plateforme de monitoring.

Administration>Use Cases>Add Use Case

🔔 **Note :** Only the added contributors will be able to view this Use Case in their workspace.

Once all required fields are completed, the administrator confirms the creation. The Use Case becomes active and is visible to the designated contributors.

Administration>Use Cases>Add Use Case >Apply



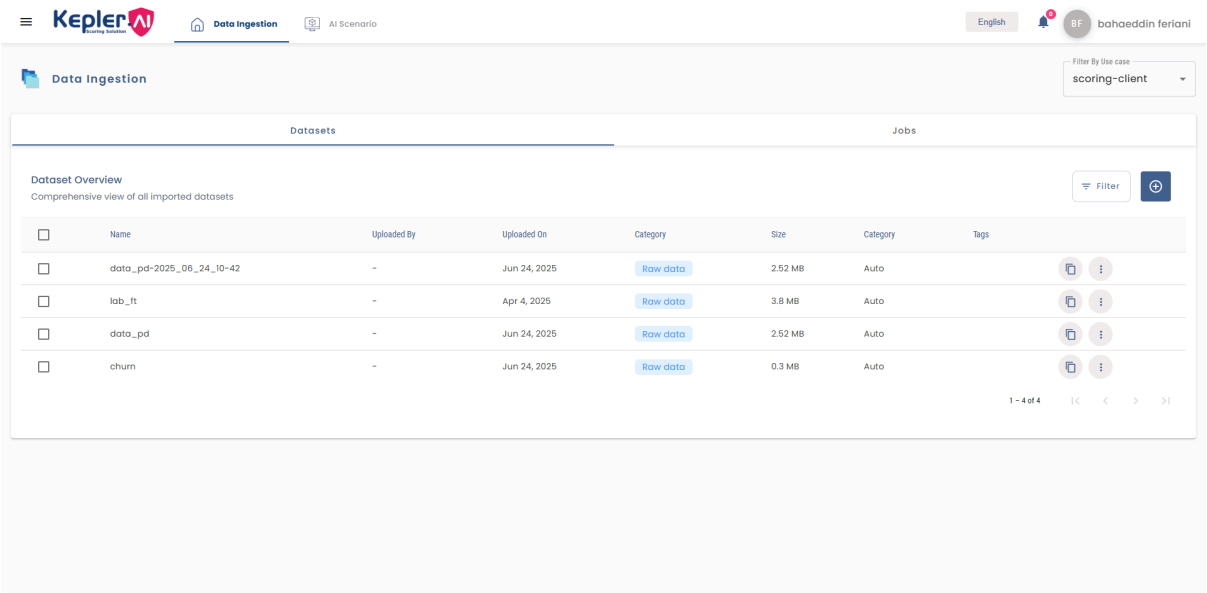
Par tanitlab AI

DATASET MANAGEMENT

I. Data Ingestion

Data ingestion is done through the ***LAB*** **module** in **KeplerAI**. It is the first critical step to bring in data from various sources for processing and analysis.

LAB> Data Ingestion > Datasets



The data can originate from **structured files** (CSV, XLSX, JSON, Parquet) or from relational **databases** such as **MySQL, PostgreSQL, or SQL Server.**

📍 **Steps**:

1. **Selecting the Relevant Use Case**

   From the **LAB** module, the user must first select the Use Case for which they want to ingest data. This allows the datasets to be contextualized and linked to a specific scenario.

2. **Adding a Dataset**

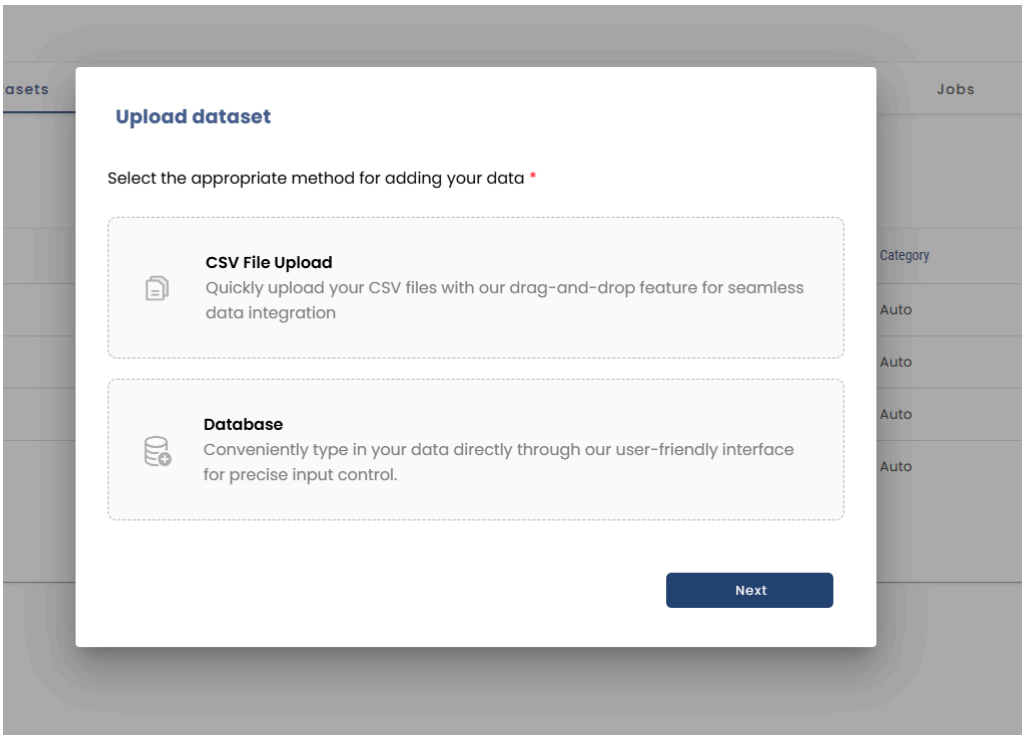
   Click the ➕ « Add a dataset » button.

   LAB> Data Ingestion > Datasets

💡**The datasets added here will be reusable during the scenario creation phase, particularly when ingesting data for a Use Case.**

3. **Selecting the Ingestion Mode Based on the Data Source**

LAB> Data Ingestion > Datasets>+



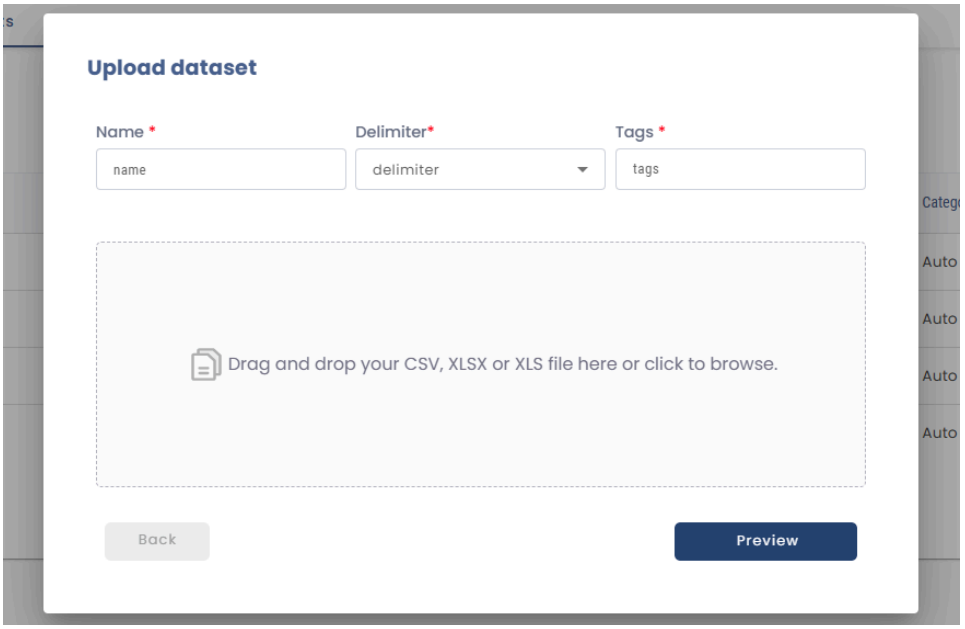
Deux cas sont possibles selon le type d'insertion des données :

➔ 📁 **Ingestion manuel depuis un fichier** :

- Download the dataset
- Fill in the following information: Dataset **Name** (name), used **Delimiter** (delimiter), and associated **Tags** (tags).

- Drag and drop the CSV, XLSX, or JSON file, or click to select it.

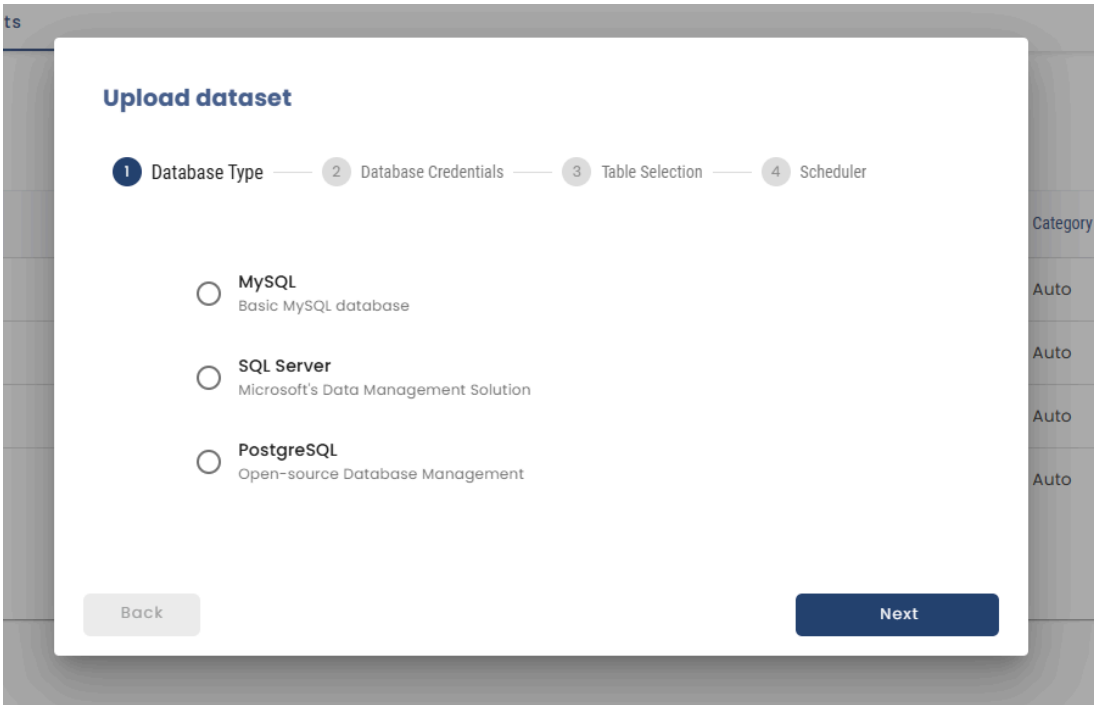
LAB> Data Ingestion > Datasets>+>CSV File Upload



➔ 🗄️ **Ingestion from a database:**

- **Step 1**: Select the **database type** (MySQL, SQL Server, PostgreSQL).
- **Step 2**: Provide **connection credentials** (host, port, username, password).
- **Step 3**: Select the **table** to import.
- **Step 4**: Configure a **scheduler** to automate ingestion (Optional) .

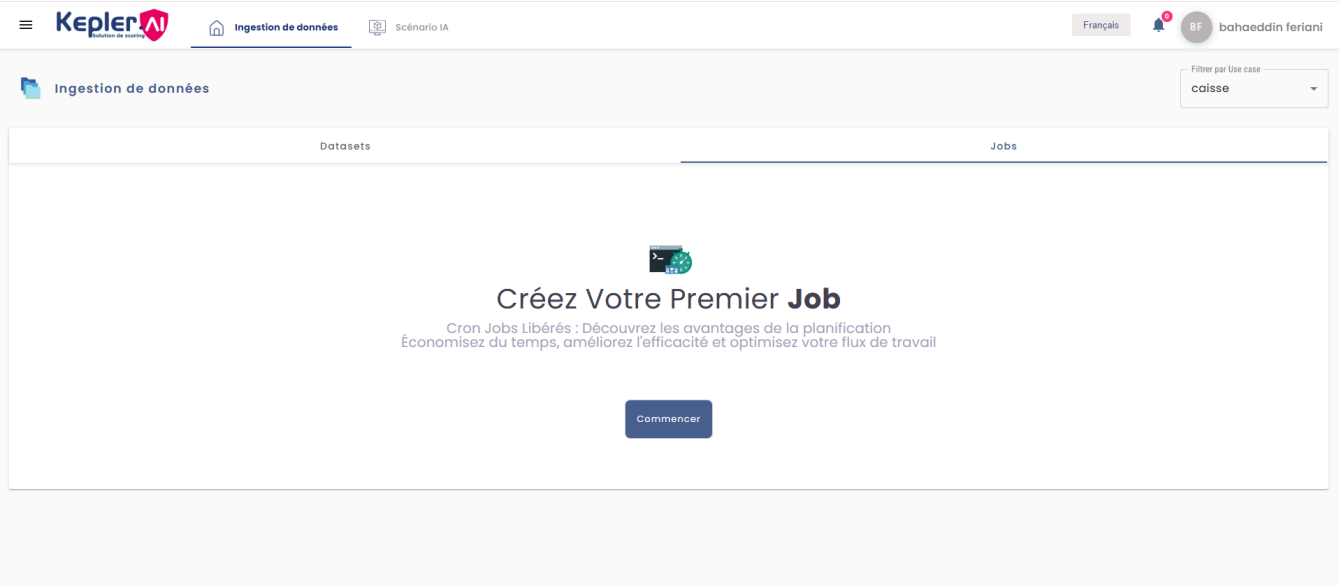
LAB> Data Ingestion > Datasets>+>Database



II. Ingestion Script (Job)

Automated data ingestion in **KeplerAI** is done through scheduled jobs, known as **Cron Jobs**, available in the **LAB** module.

LAB> Data Ingestion>Jobs



These jobs allow scripts to be triggered at regular intervals (daily, weekly, monthly…) to process or update data autonomously, without manual intervention.

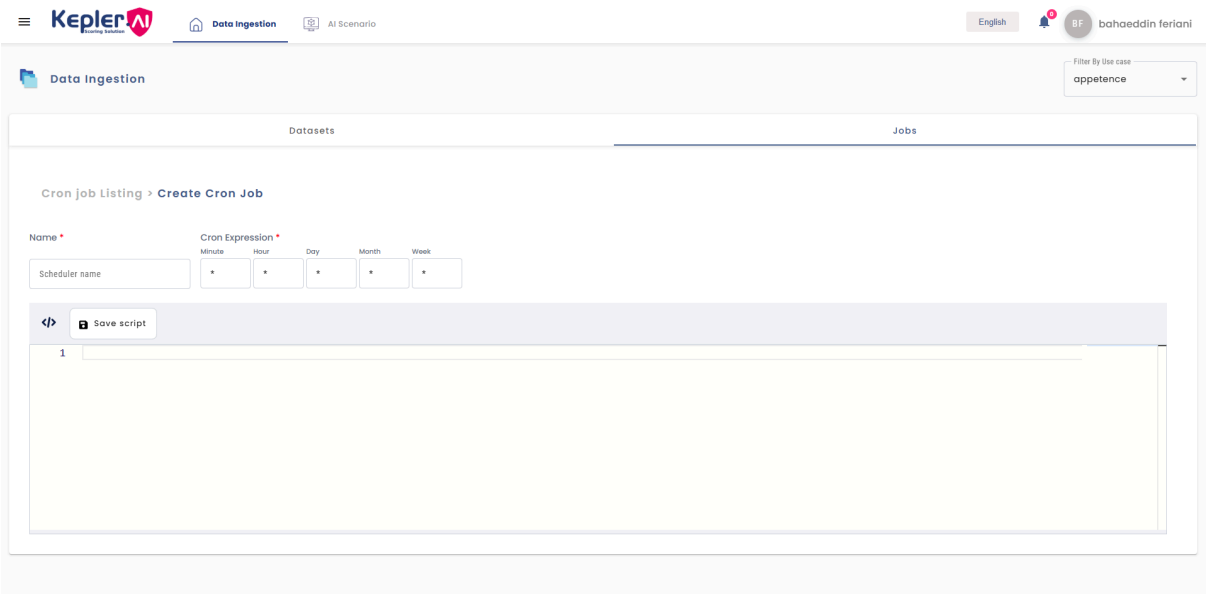
📍 **Steps**:

1. Select the relevant Use Case
2. Click on the "+" button or "Get Started"
3. A **Cron Job** creation interface appears, structured around the following elements :

Field	Description
Job Name	Enter a clear name to identify the automated ingestion task
Cron Expression	Define the execution frequency using a CRON Expression(Minute, Hour, Day, Month, Week).

Script input area	A text editor allows the user to write or paste a Python or Spark ingestion script to be executed periodically

LAB> Data Ingestion>Jobs>Get Started



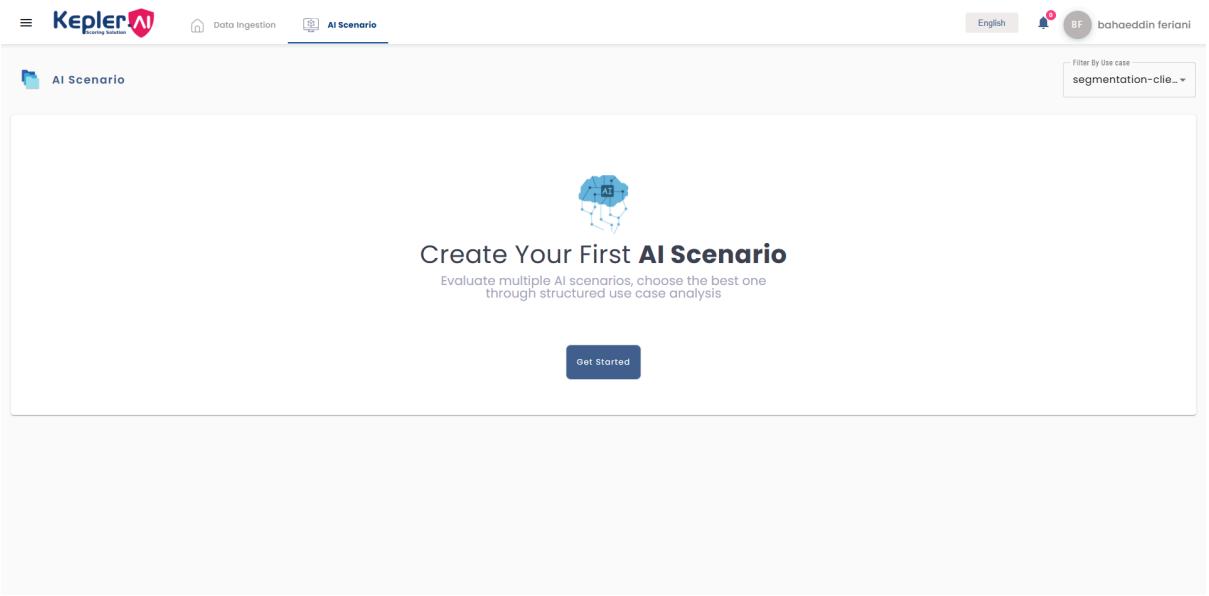
SCENARIO CREATION

Once the Use Case is created and accessible, the **Data Scientist** can begin building an **AI Scenario** through the **LAB** module, which includes all features related to data engineering and model training.

I. Steps to create a scenario:

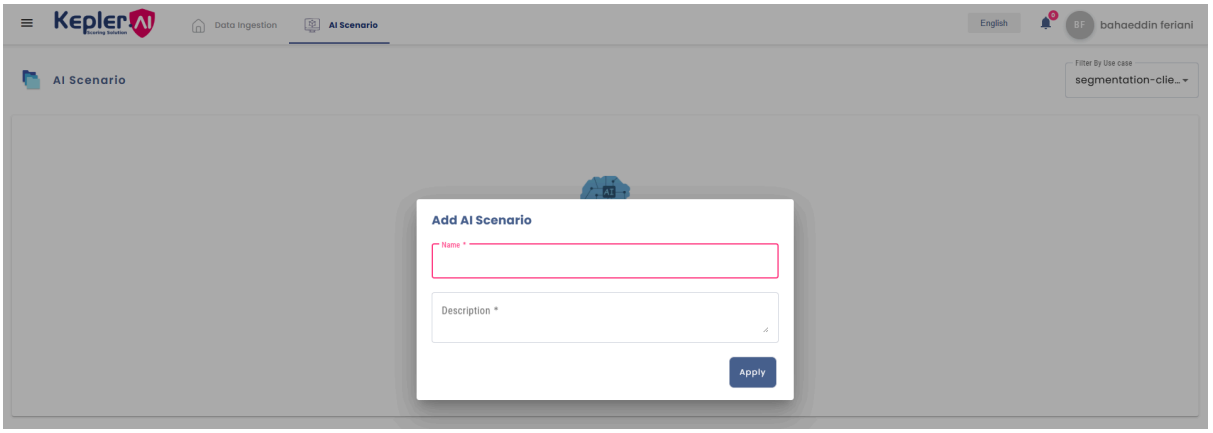
1. From the **LAB** module, within the **AI Scenario** section, select the previously created **Use Case** using the dropdown menu.

LAB>AI Scenario



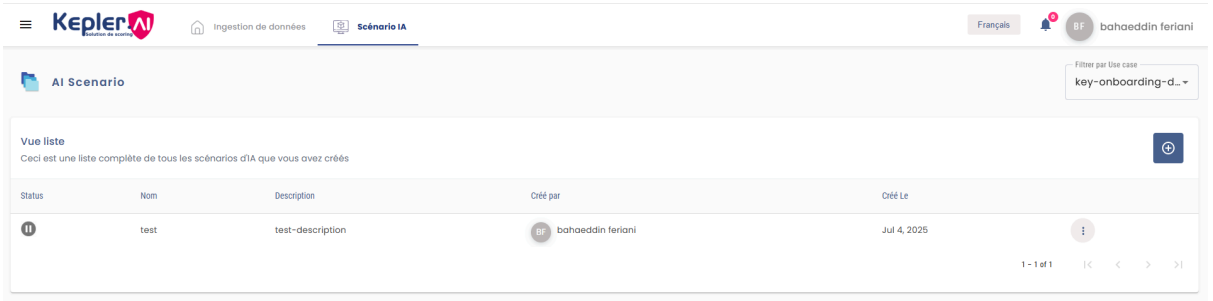
2. Click the "+" or "**Get Started**" button, then provide the following information:
    a. **Scenario Name**
    b. **Description**

LAB>AI Scenario>Get Started



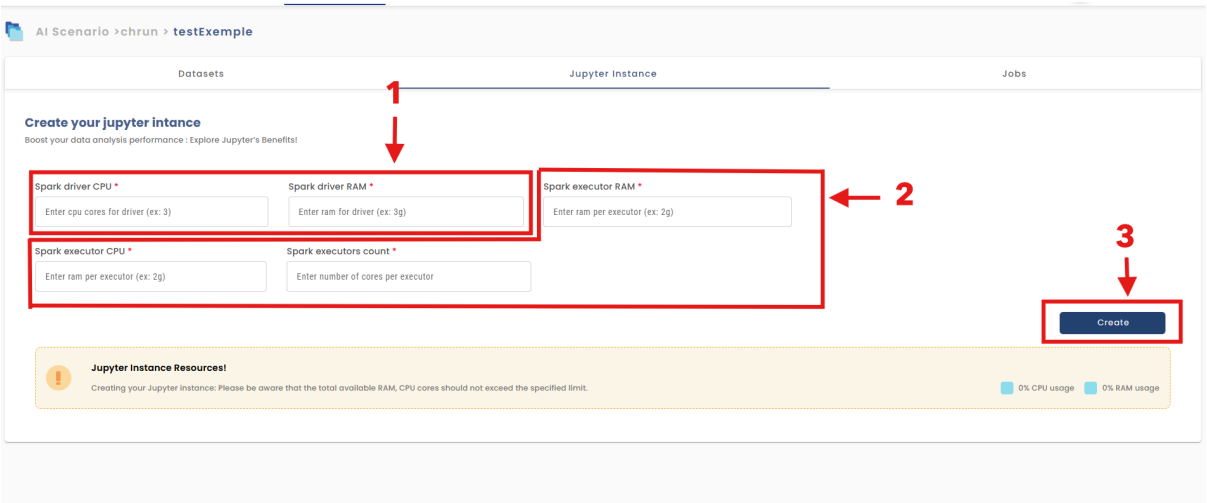
3. Once the scenario is confirmed, it will appear in the list of scenarios associated with the Use Case.

LAB>AI Scenario



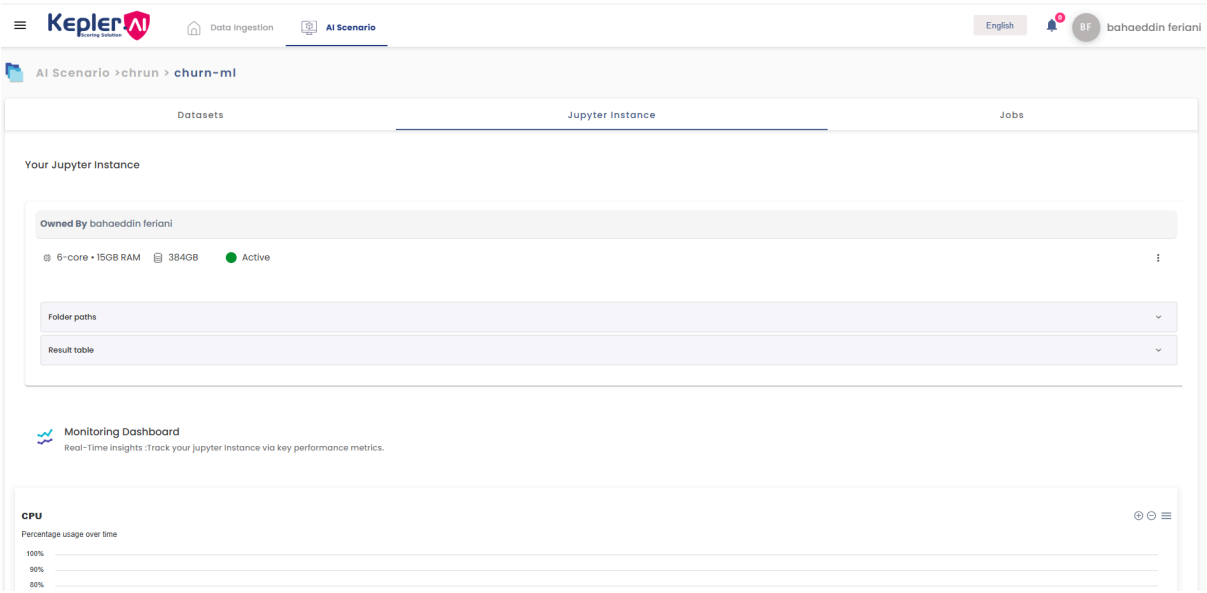
4. Click on the scenario to proceed to the **Jupyter instance** setup:
    1. Configure the Spark Driver
        a. **CPU**: Specify the number of cores (e.g., 3)
        b. **RAM** : Enter the memory in GB (e.g., 30)

LAB> AI Scenario > Example Scenario > Jupyter Instance



2. Configure the Spark Executors
   a. **CPU per executor**: Define the processing power (e.g., 4)
   b. **RAM per executor**: Specify the memory in GB (e.g., 20)
   c. **Number of executors**: Set the required number (e.g., 5)
3. Click on the **"Create"** button to launch the instance with the defined parameters.
4. Jupyter Instance Interface (Post-Creation):

LAB> AI Scenario > Example Scenario > Jupyter Instance



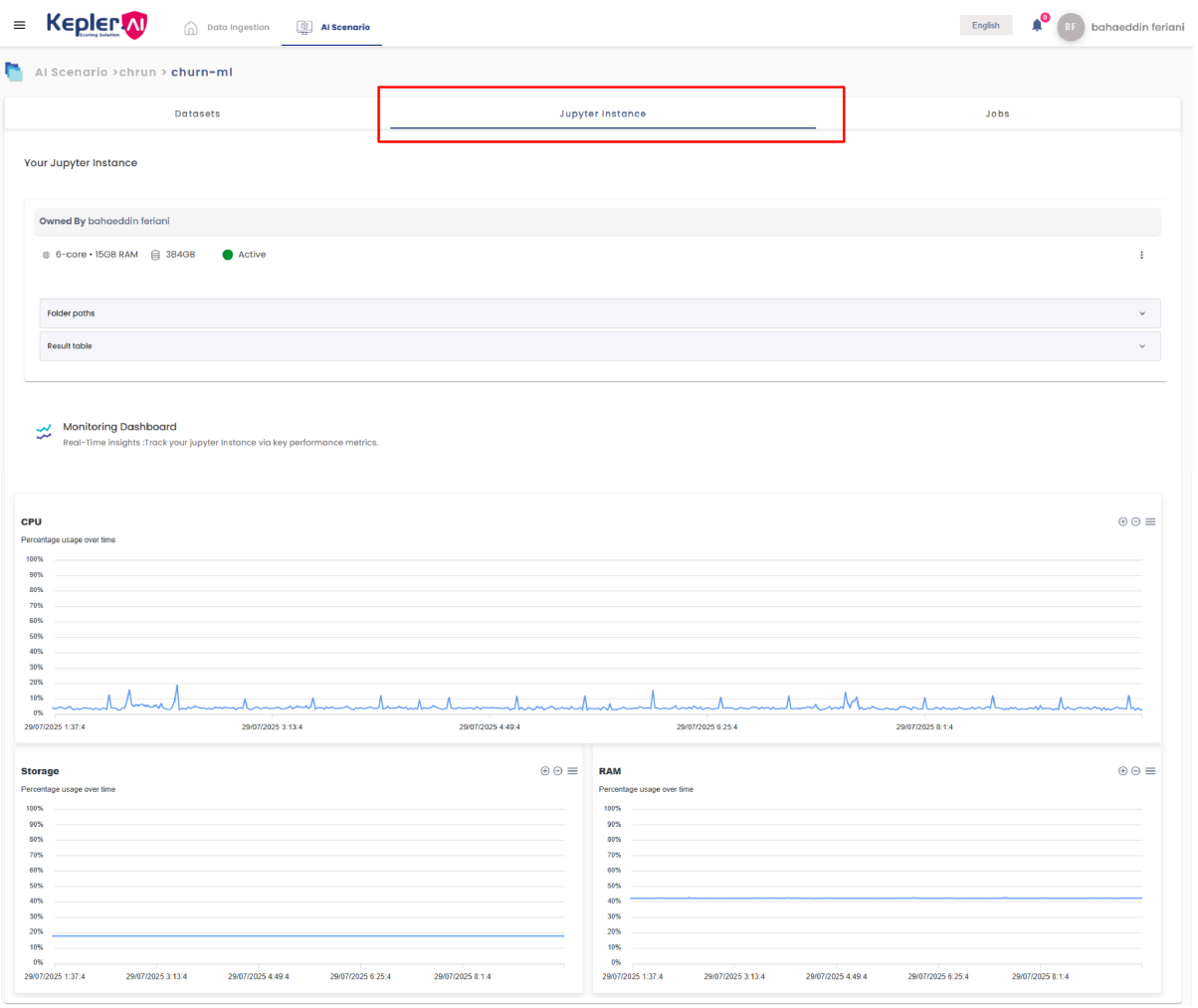
Once these parameters are validated, the Jupyter instance is automatically deployed, allowing the Data Scientist to write and execute their analysis, transformation, and modeling code directly within the **KeplerAI** environment.

II. Using an AI Scenario in KeplerAI

KeplerAI provides a unified platform designed to orchestrate the entire lifecycle of artificial intelligence (AI) projects. With its centralized interface, data teams can efficiently manage their experimentation pipelines while ensuring full traceability and reproducibility across all stages of the workflow.

1. Instance Monitoring Interface

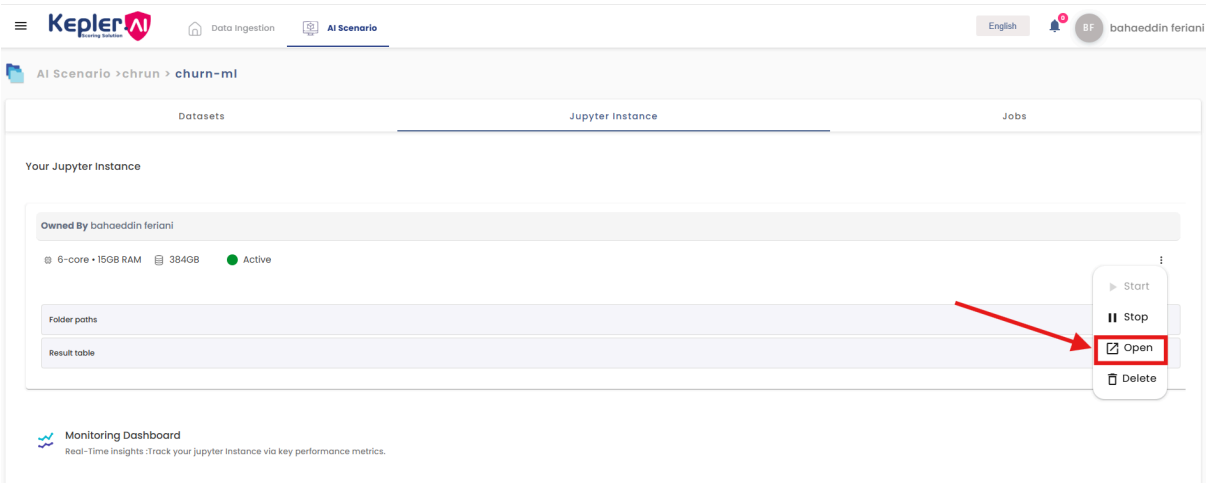
LAB> AI Scenario > Example Scenario > Jupyter Instance



Once the Jupyter instance is launched within **KeplerAI**, the Data Scientist is granted access to a centralized monitoring interface that enables them to:

- **Monitor the real-time** status of the instance through **key performance metrics (CPU usage, RAM consumption, executor status, etc.).**
- **Pause, delete, or restart the instance at any time.**
- Terminating or stopping the instance releases infrastructure resources for other use cases or initiatives.
- Click on **"Open"** to directly access the integrated Jupyter Notebook environment within **KeplerAI**.

LAB> AI Scenario > Example Scenario > Jupyter Instance



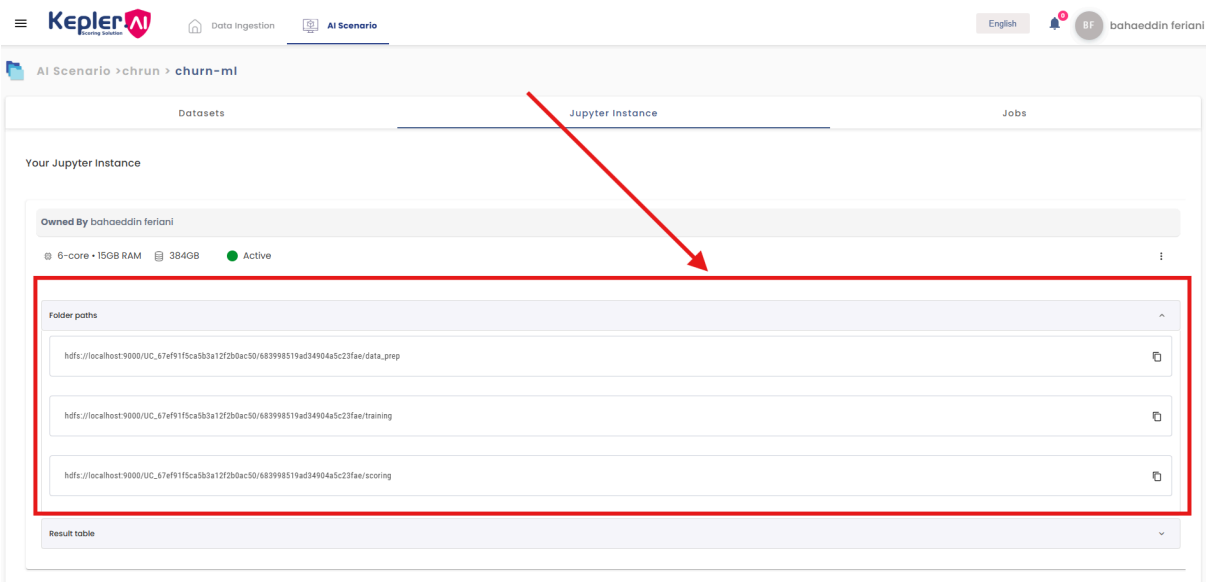
KeplerAI also simplifies navigation to the results generated by AI processes. From the monitoring interface or directly within the Jupyter environment :

1. Click on the desired AI scenario to view the full list of available AI scenarios.
2. Select a scenario such as *KYC (Know Your Customer)* or create a new one.
3. Launch a Jupyter instance by filling in the required parameters and clicking the **Create** button.
4. A new interface will open, displaying all components of your Jupyter instance (user, charts, etc.).
5. When you click on the **Folder Paths** section, a list of HDFS paths will appear. These paths can be copied and reused in other scripts or workflows.

   Modules available in the instance include:

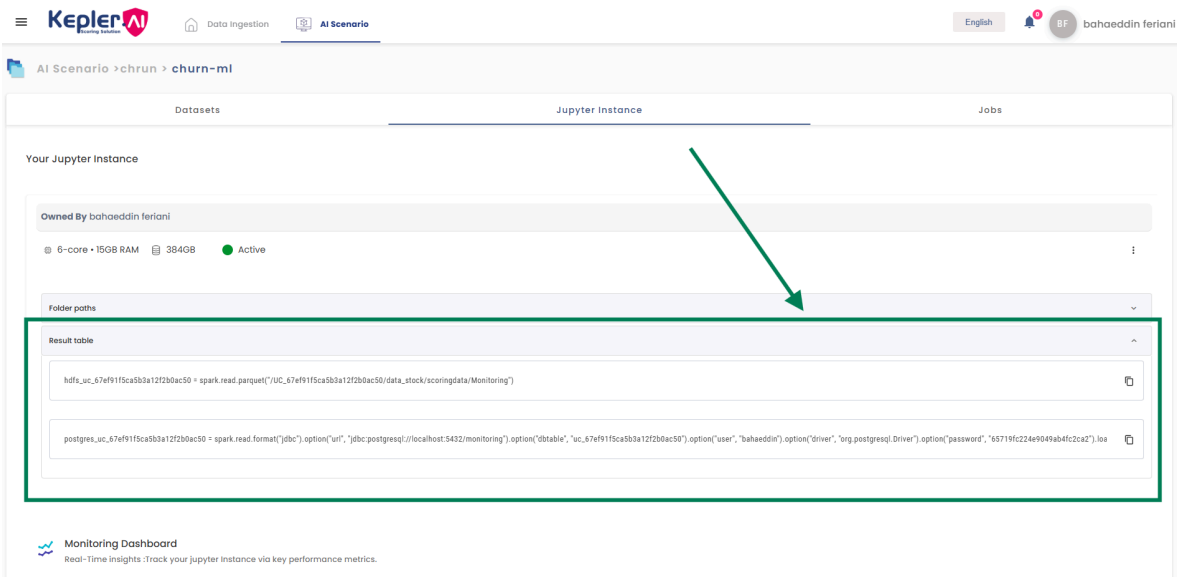
- **Data Prep**: for preparing and transforming raw data into a clean, usable dataset.
- **Training**: for implementing training and testing algorithms.
- **Scoring**: for applying scoring algorithms to new instances—typically those used in production environments.

LAB> AI Scenario > Example Scenario > Jupyter Instance>Folder paths



- Links to results stored in HDFS or PostgreSQL databases are readily available and can be copied with a single click for consultation or sharing.To access them, follow these steps:
  1. **Click on the relevant AI** scenario to display the full list of available scenarios.
  2. **Select an existing scenario or create a new one based on your needs.**
  3. **Launch a Jupyter instance** by completing the required fields and clicking the **Create** button.
  4. A **new interface** will open, displaying all components of your Jupyter instance (user, charts, etc.).
  5. By clicking on the **Results Table** section, a **list of links to the output data** (HDFS or PostgreSQL) will be displayed. These **links can be copied instantly** to streamline consultation or sharing.

LAB> AI Scenario > Example Scenario > Jupyter Instance>Result table

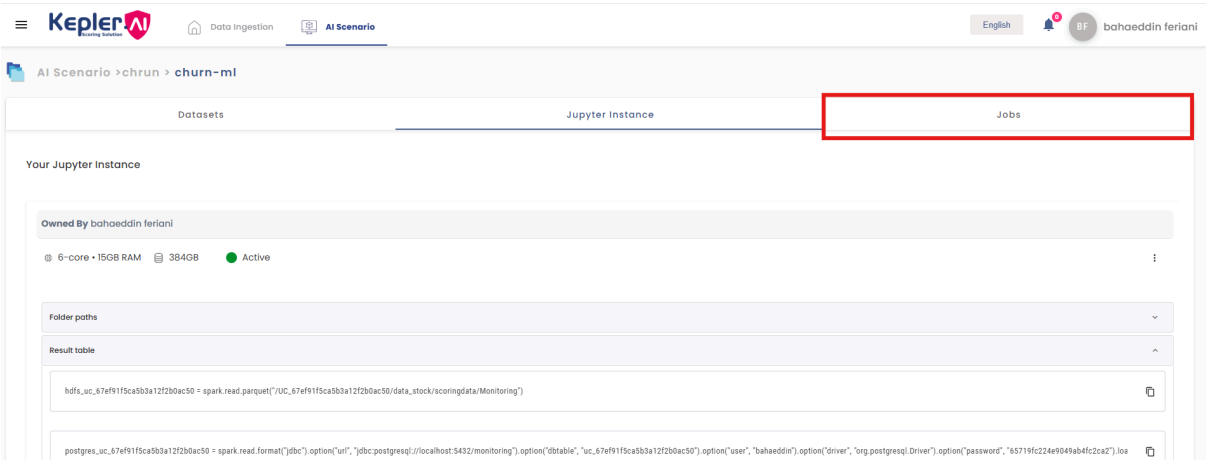


💡 **These features significantly improve efficiency in results analysis, data traceability, and the industrialization of AI workflows.**

2. Automated Job Creation

Kepler provides a job scheduling system via the **Jobs** tab.

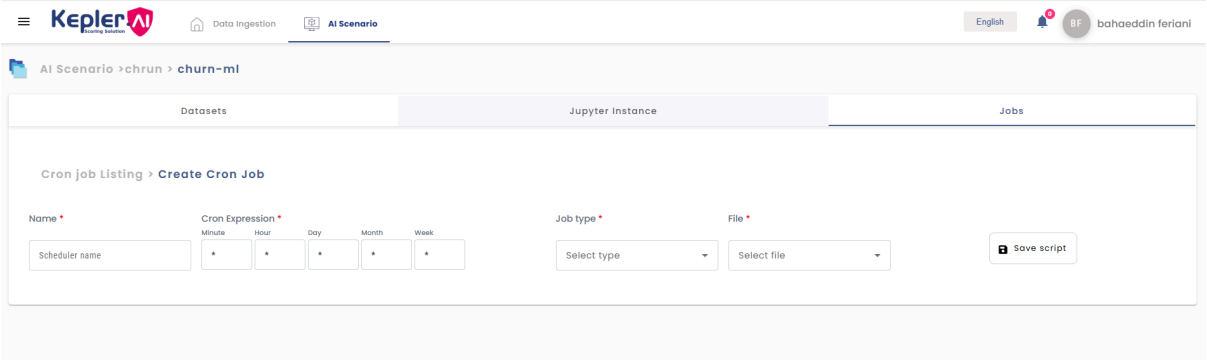
LAB> AI Scenario > Example Scenario >Jobs



📍 **Steps:**

1. From the **Jobs** window, click the **"+"** button or **"Get Started"** to begin.
2. In the **Job creation interface**, the Data Scientist fills in the following fields:

LAB> AI Scenario > Example Scenario >Jobs>Get Started



- **Job Name**
- **CRON Expression***: to define the execution time or frequency of the script
- **Job Type***:
  a. Scenario Training folder
  b. Scenario Scoring folder
  c. Scenario Data Prep folder
- **File Used**
- The **Job is automatically linked to the active Use Case**, identified by its unique ID in the format `UC_<ID>`.

💡 **The created Job enables the automation of critical tasks such as model training, periodic scoring, and data preparation, while ensuring full traceability and reproducibility.**

La section suivante détaille les étapes clés du cycle de vie d'un scénario IA, à savoir : **Data Preparation, Data Training** , and result **scoring and utilization**.**(Data Scoring).** These steps are essential to ensure the quality, performance, and robustness of the deployed solutions.

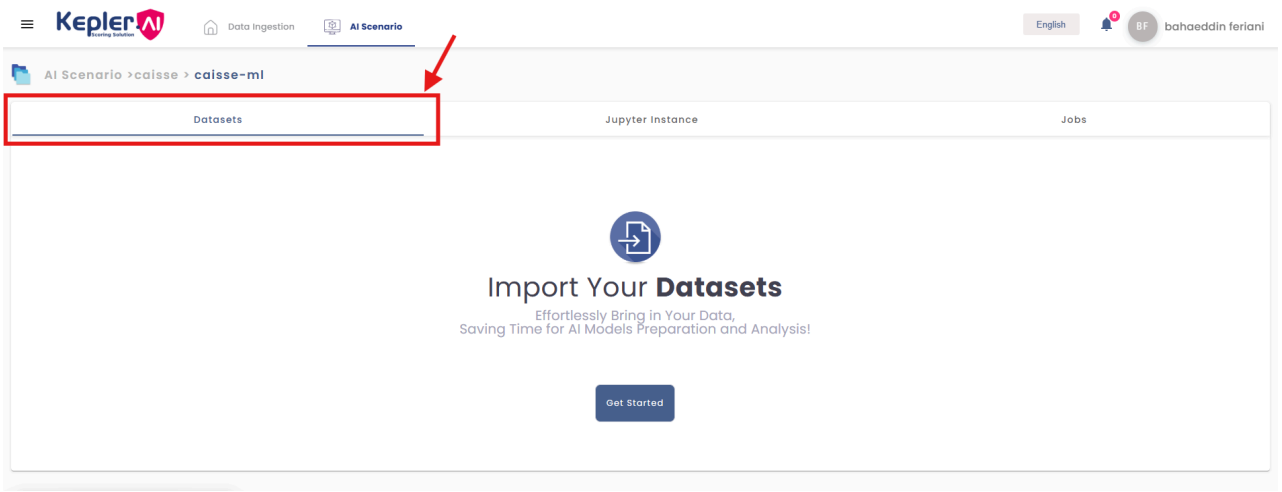
3. Dataset Importation

The datasets used in **KeplerAI**'s AI scenarios are preloaded through the data ingestion interface, based on the selected use case. In this step, the goal is simply to associate these existing datasets with the AI scenario to enable the data preparation, training, and scoring phases.

📍 **Steps:**

1. **Access the Import Module:**

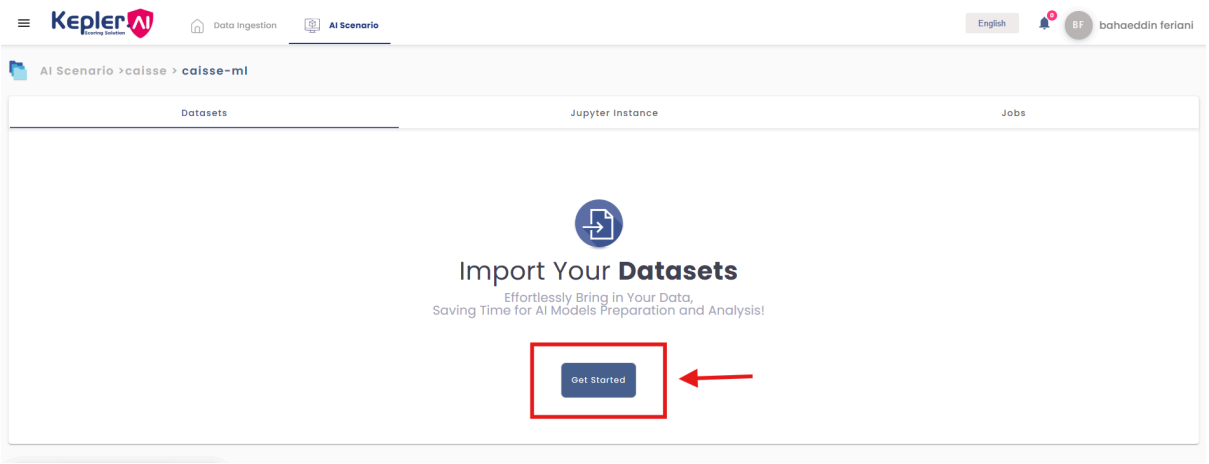
   LAB> AI Scenario > Example Scenario>Datasets



2. **Start the Import Process:**

   Click on the **"+"** button or **"Get Started"** to initiate the import.

LAB> AI Scenario > Example Scenario>Datasets



3. **Select the Use Case:**

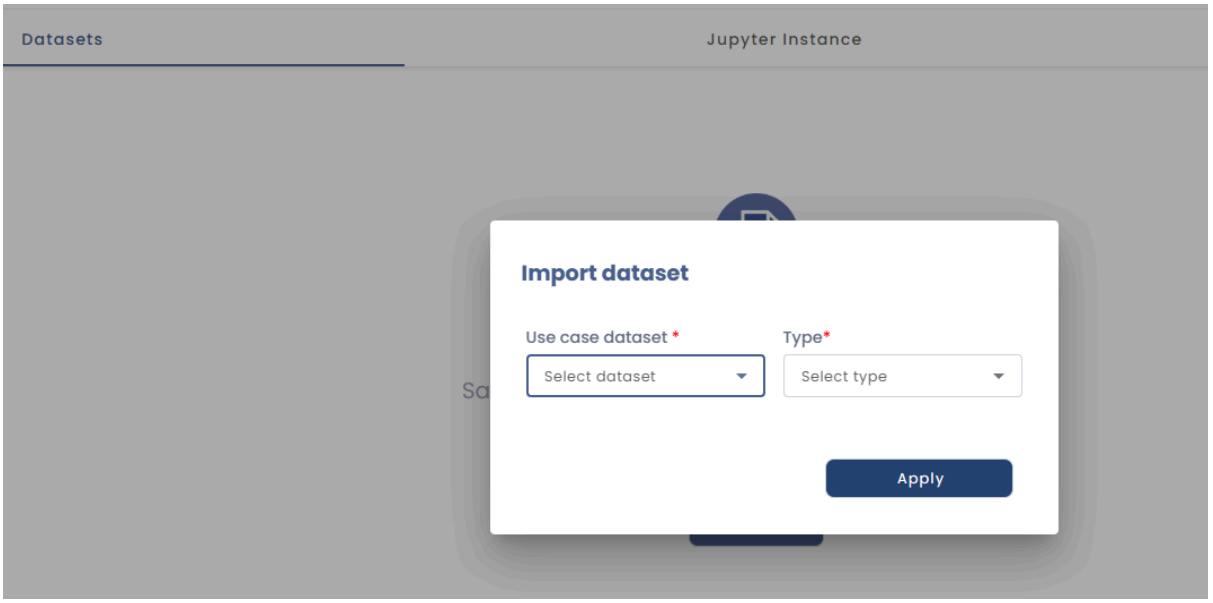
   In the **"Use Case Dataset"** field, select the appropriate use case to associate with the dataset.

4. **Choix du type de données :**

   In the "Type" field, select the data format or ingestion method (e.g., CSV file, database, etc.).

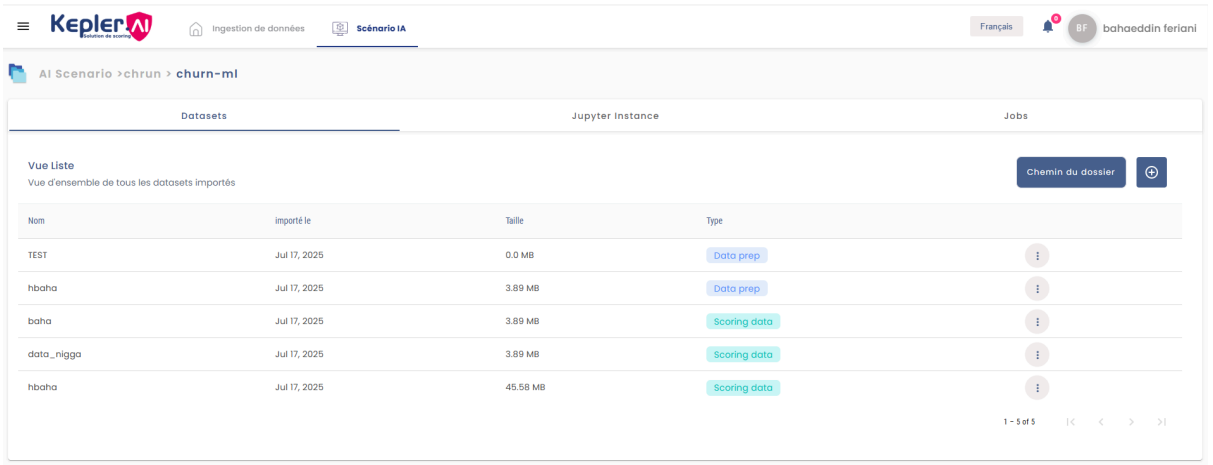
Par tanitlab

LAB> AI Scenario > Example Scenario>Dataset>Get Started



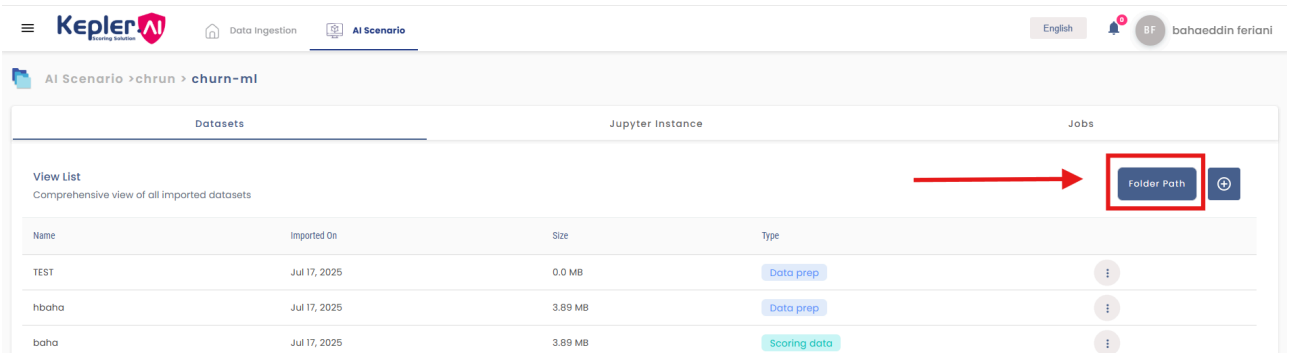
Once successfully imported, the dataset will automatically appear in the list view, displaying all datasets associated with the selected use case.

LAB> AI Scenario > Example Scenario>Dataset



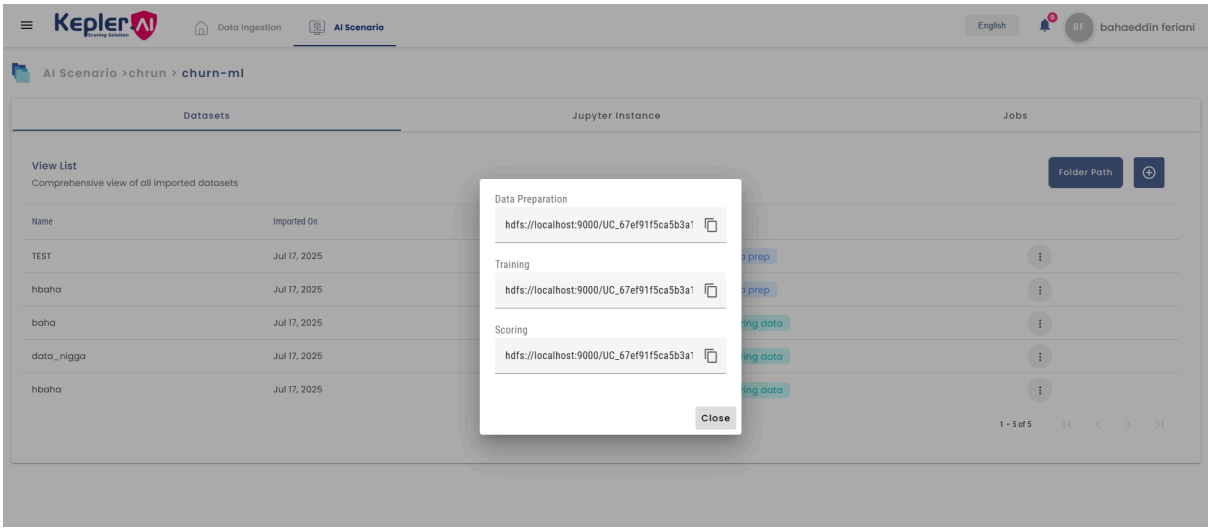
From this view, you can:

LAB> AI Scenario > Example Scenario>Dataset



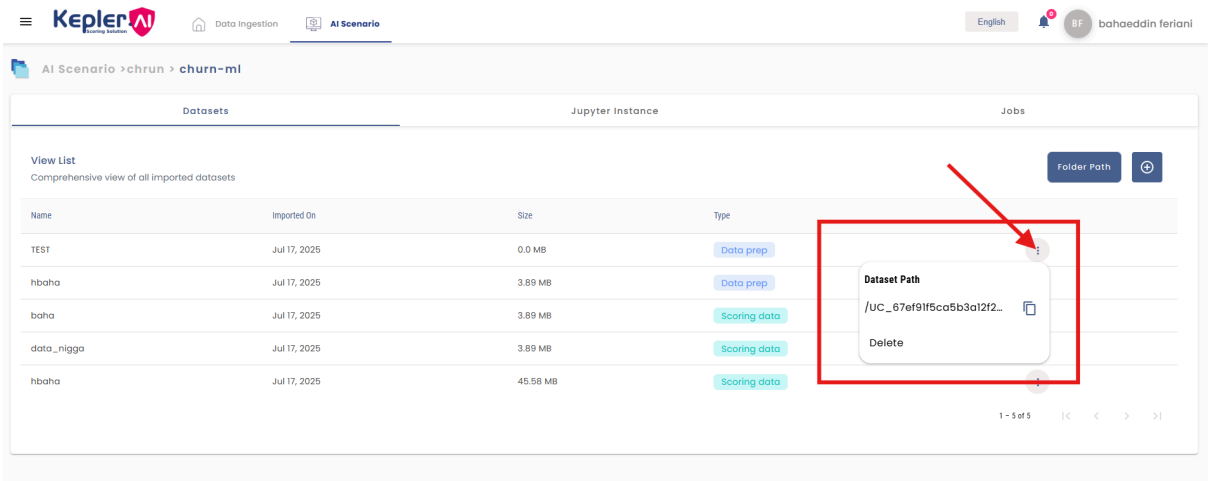
- **Copy the dataset path** to use it across different modules:
    - Data Preparation
    - Training
    - Scoring

LAB> AI Scenario > Example Scenario>Dataset>Folder Path



- **Delete a dataset**, if needed.

LAB> AI Scenario > Example Scenario>Dataset>⋮



💡 **Use the copied dataset paths to quickly integrate your data into Jupyter notebooks or automated ingestion scripts.**

Par tanitlab AI

Onglet 2 :
Data Ingestion

DATA INGESTION

Introduction

The **Data Ingestion** phase is a critical starting point for any data processing or analytics project. It involves importing raw data from various sources (files or databases) into the distributed **HDFS** system, where the data is centralized, organized, and prepared for further use.

In the **KeplerAI** platform, data ingestion can be performed in two modes:

- **Manually** via the graphical user interface
- **Automatically** via a scheduled script (Cron Job)

Both methods support a **DevOps-driven approach** focused on scalability, repeatability, and automation of data pipelines.

I. Manual Ingestion via the Kepler Interface

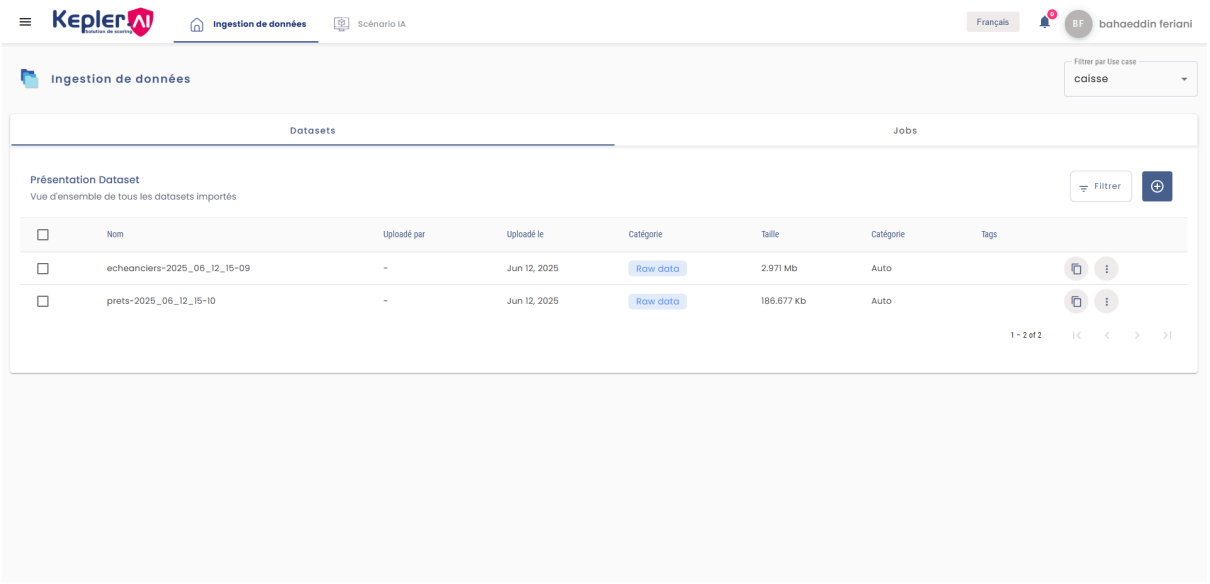
This method is particularly useful for quickly uploading ad hoc datasets or testing sample data before setting up an automated pipeline.

📍 **Steps to Follow :**

1. Access the Platform
   - Open **KeplerAI**

- Navigate to:

  LAB> Data Ingestion > Datasets



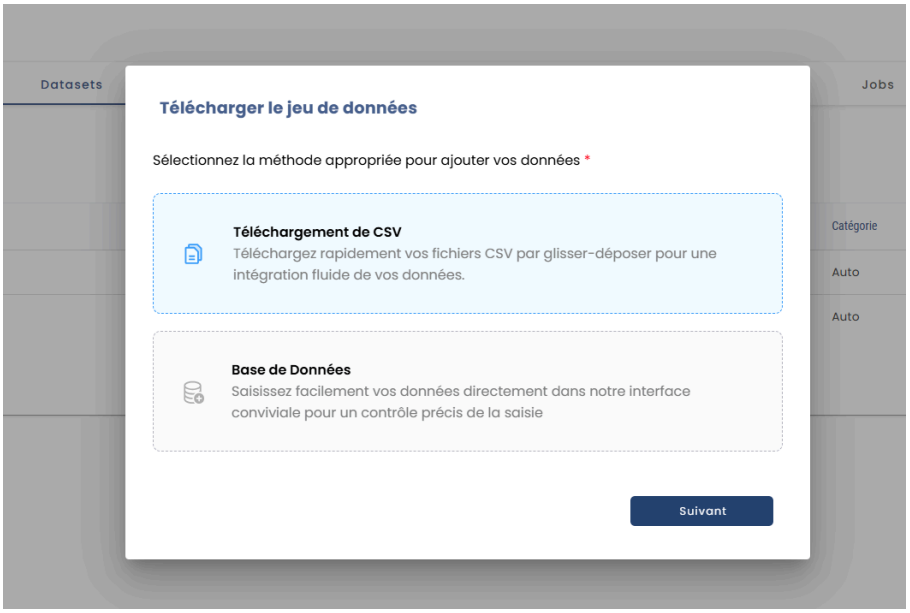
2. Select the Relevant Use Case

- Choose the **Use Case** in which the dataset will be used
- This step is essential for **contextualizing** the data and linking it **to business-specific scenarios.**

3. Add a New Dataset

- Click on ➕ **Add Dataset**

  LAB> Data Ingestion > Datasets>+



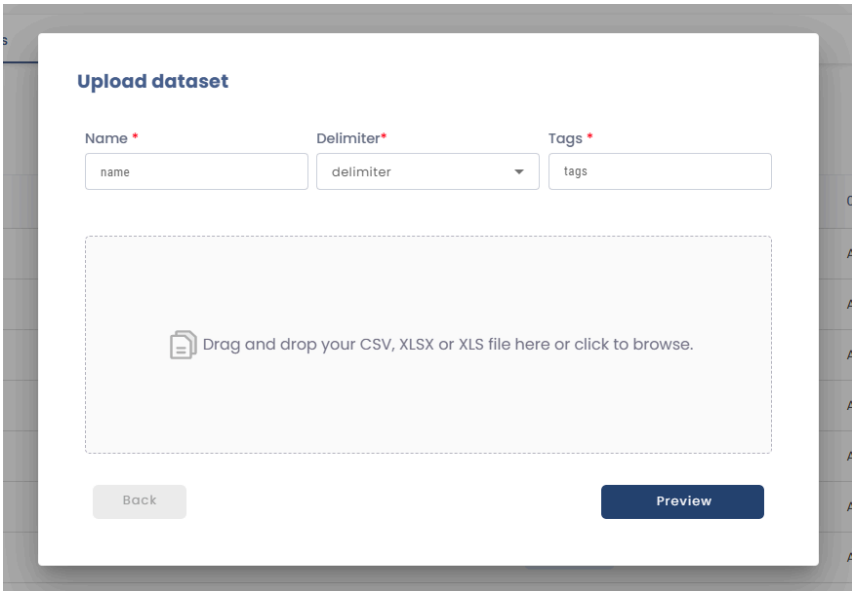
- Choose the Ingestion Mode:
  - 📁 **Local File (CSV, XLSX, JSON)**
  - 🗄️ **Relational Database (MySQL, PostgreSQL, SQL Server, etc.)**

A. Ingestion from a File

📍**Steps:**

1. Fill in the required metadata:
   - **Dataset Name**
   - **Delimiter Used** (e.g., `,`, `;`, tab, etc.)
   - **Tags** for organizing and filtering datasets
2. **Drag and drop the file into the interface or select it manually.**
3. Click **Validate** to upload the dataset

LAB> Data Ingestion > Datasets>+>CSV File Upload



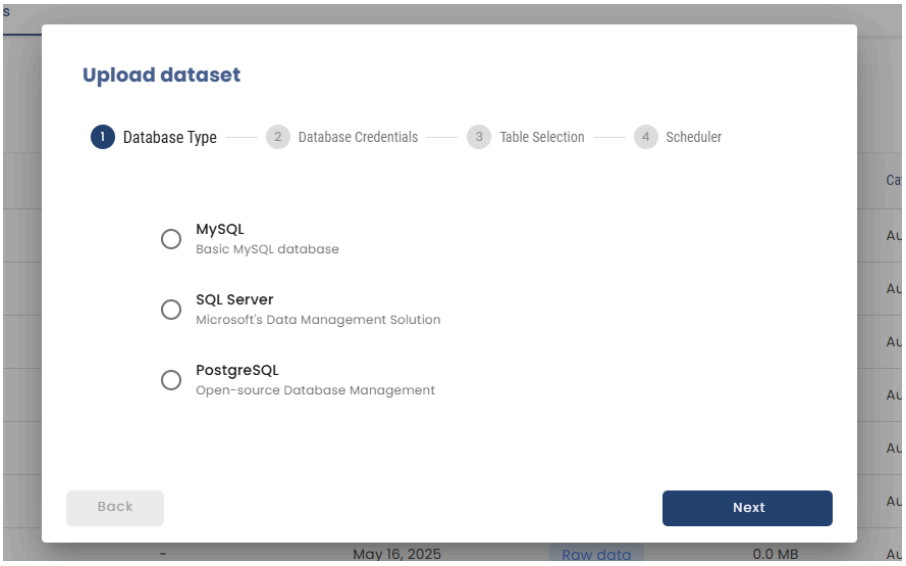
💡Les Imported files are saved in the Use Case's **RAW directory**, allowing future use for data preparation, visualization, or modeling.

B. Ingestion from a Database

📍**Steps:**

1. Choose the database type: MySQL, PostgreSQL, SQL Server, etc.

LAB> Data Ingestion > Datasets>+>Database



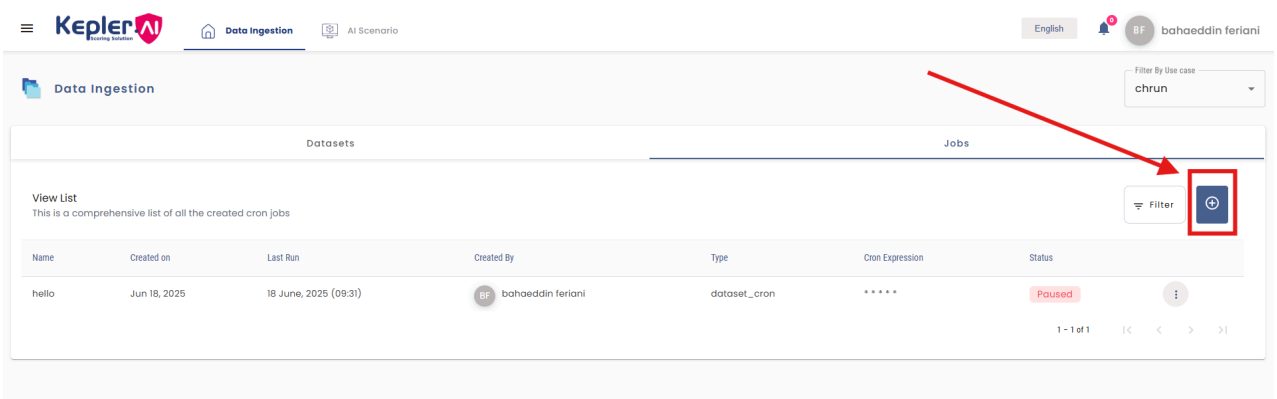
2. Provide connection details:
    - IP address or hostname
    - Port
    - Database name
    - Access credentials (username/password)
3. Select the **source table** to import
4. (Optional) Configure a **scheduler** to define an automatic ingestion frequency

II. Automated Data Ingestion via Script (Scheduled Job)

To automate data ingestion, **KeplerAI** allows the creation of **scheduled jobs (Cron Jobs)**. These jobs periodically execute a **PySpark** or **Python** script, enabling regular data updates without manual intervention.

📍 Steps to Create a Job :

1. Access the Jobs Section
    - Navigate to *: **LAB >Data Ingestion> Jobs***



- Cliquer sur **"Commencer"** ou sur le bouton ➕ **Ajouter un Job**

2. Select the Relevant Use Case

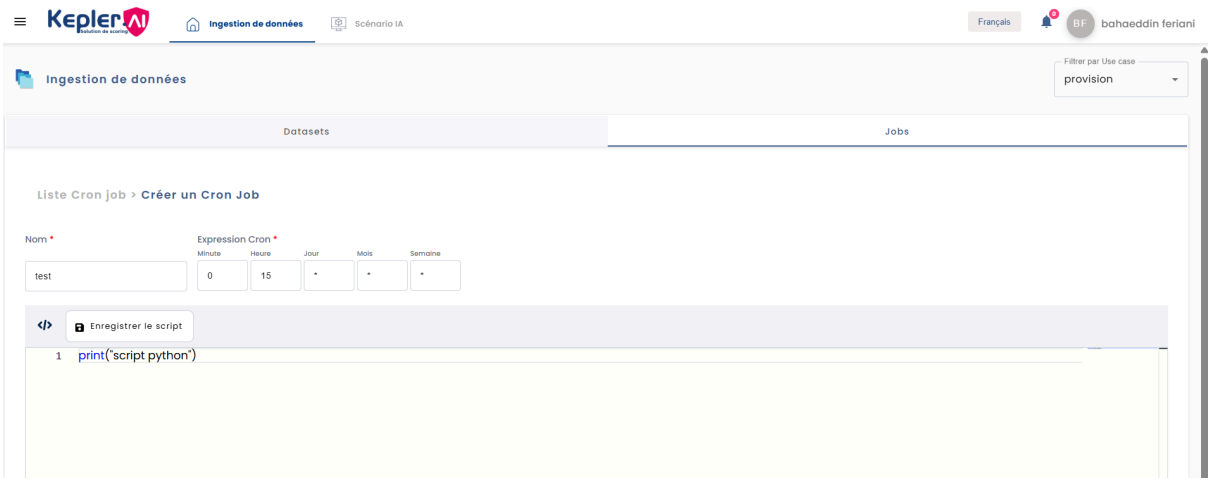
- Associate the job with the appropriate **Use Case**, just as with datasets.

3. Fill in Job Fields

Field	Description
Job Name	Enter a clear and descriptive name to identify the ingestion task.
CRON Expression	Define the execution frequency using a CRON syntax (Minute, Hour, Day, etc.).
Script Editor	Use the embedded editor to write or paste the Python or PySpark ingestion script.

💡 Ensure your script is **validated and tested** within a **Jupyter** environment before deploying it as a job.

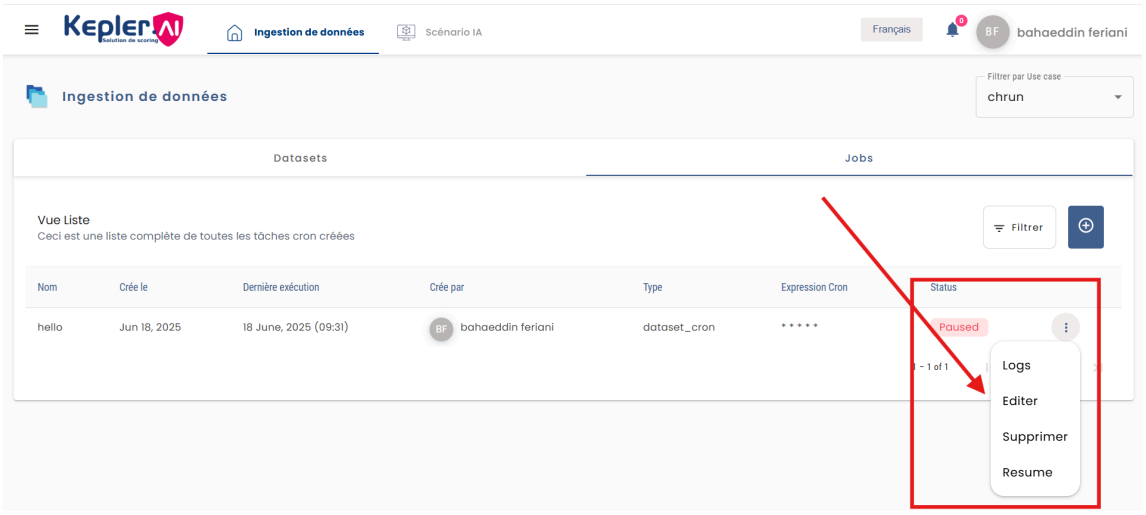
LAB >Data Ingestion>Jobs>Get Started



III. Script Lifecycle Management

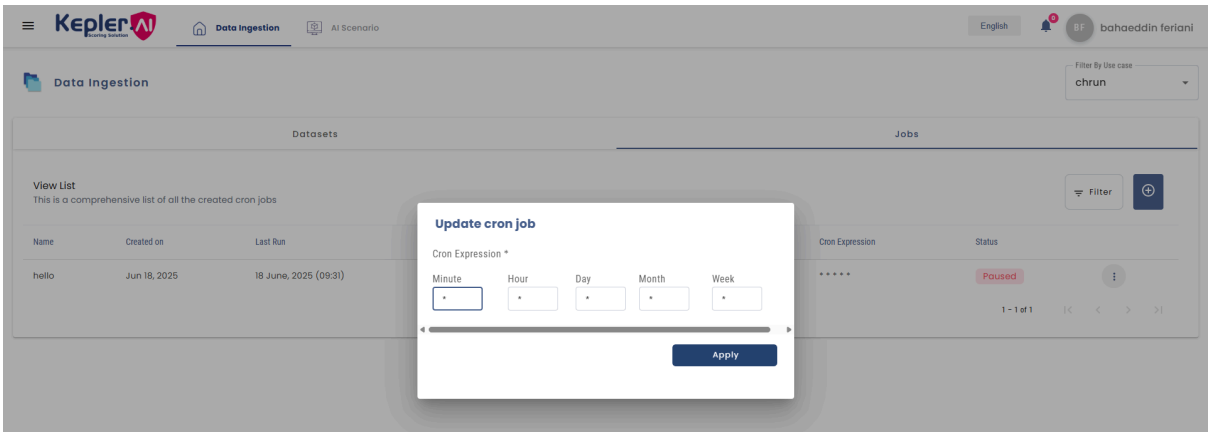
KeplerAI provides a structured lifecycle for managing automated ingestion scripts:

LAB >Data Ingestion>Jobs



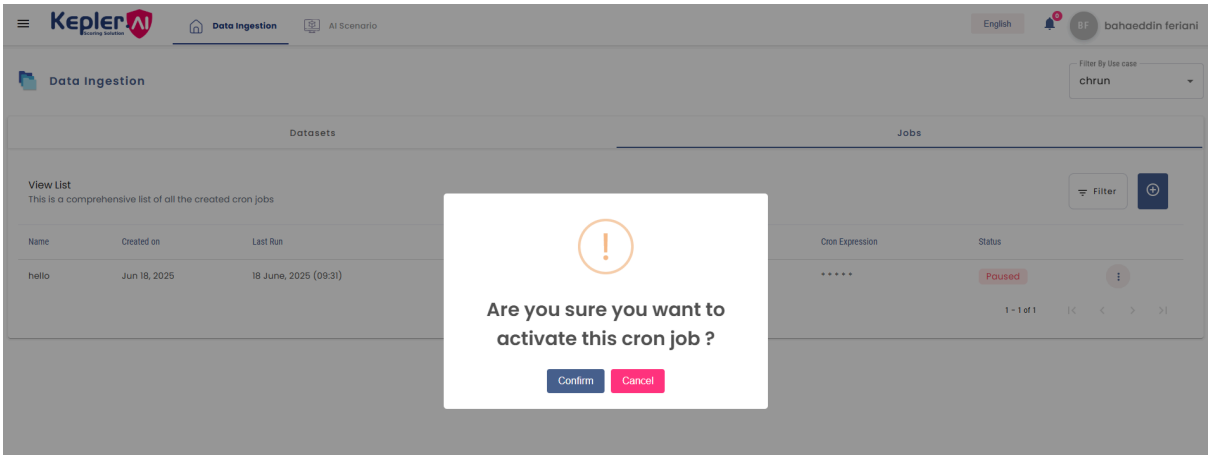
- **Edit**: Update the CRON schedule

LAB >Data Ingestion>Jobs>Edit

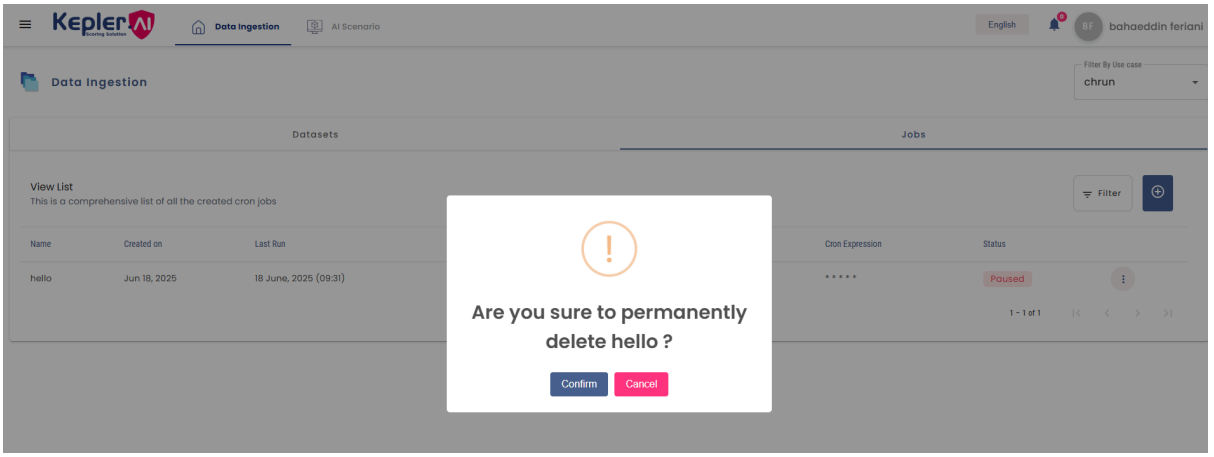


- **Active**: The script is validated and running in production

LAB >Data Ingestion>Jobs>Resume



- **Paused**: The script is temporarily disabled
- **Deleted**: The script is archived or removed



This structure ensures **full traceability** and **smooth pipeline governance**.

IV. Key Advantages of Centralized Ingestion in Kepler

- **Time-saving** through automation
- **Dataset reusability** across multiple scenarios
- **Secured** access with managed credentials
- **Native integration** with analytics and AI processes

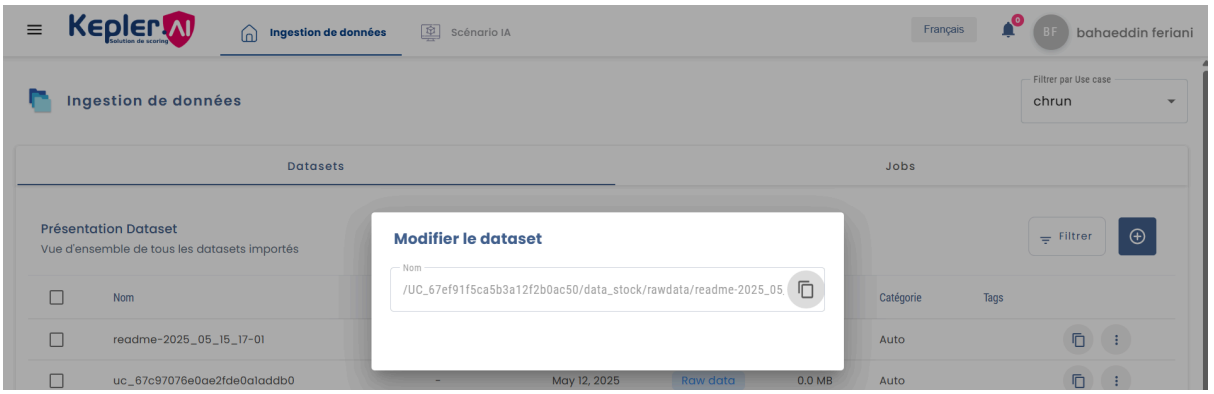
🗂️**Structured** Data Organization in Clear HDFS Directories:

bash

CopierModifier

/UC_<ID>/data_stock/raw_data/table_<NomTable>

- ***LAB >Data Ingestion>Datasets> Cut Icon***



This structure **centralizes raw data** for each use case while maintaining a clean and accessible layout for the next stages: preparation, training, and scoring.

V. Data Extraction

Once data is ingested into HDFS, the **extraction phase** loads it as a **Spark DataFrame**, enabling scalable, resilient processing.

Two extraction modes are available based on business needs:

1. Append Mode (Incremental)

- Automatically filters records from the previous day (D-1) using a timestamp or date column
- Only new data is added, avoiding duplication
- Ideal for **daily data streams**

2. Overwrite Mode (Full Refresh)

- Rewrites all extracted data
- Deletes existing files in the HDFS target directory
- Recommended when a **complete rebuild** is needed to ensure data consistency

Conclusion

The **Data Ingestion** module in **KeplerAI** provides a **comprehensive, scalable, and flexible solution** for importing data from various sources—both manually and automatically. It is part of a **modern DevOps approach**, ensuring continuous data availability ready for analytics, modeling, or scoring workflows..

Onglet 3 :
Data Preparation

DATA PREPARATION

Introduction

Once the **Use Case** and corresponding **AI Scenario** are created in **KEPLER**, the Data Scientist can initiate the **data preparation** phase—an essential step to ensure the quality and relevance of the information used by AI algorithms.

This phase consists of two main stages:

- **Development of the data preparation logic** within the **Jupyter environment**
- **Automation** of the workflow through **Cron Jobs** in the **LAB** interface

I. Preparation Steps in Jupyter

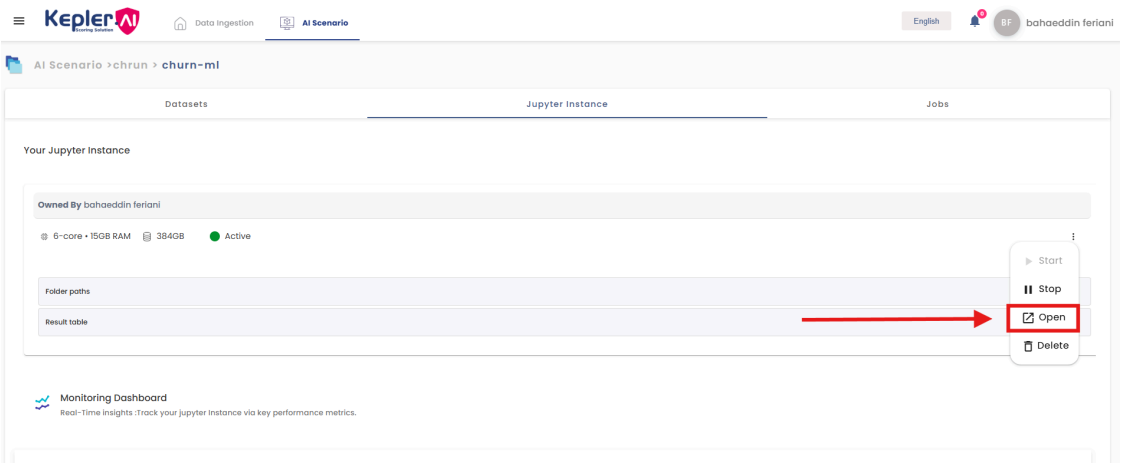
1. Accessing the Environment

The Data Scientist accesses the **Jupyter environment** through the KEPLER interface by following these steps:

1. From the **Home** page, go to the **LAB** module
2. Click on **AI Scenario** in the sidebar
3. Select an existing scenario from the AI Scenarios list
4. In the **Jupyter Instance** section:
    - If the instance is shown as inactive, click **Activate**
    - Once active, click **Open** to launch the Jupyter environment

Par tanitlab

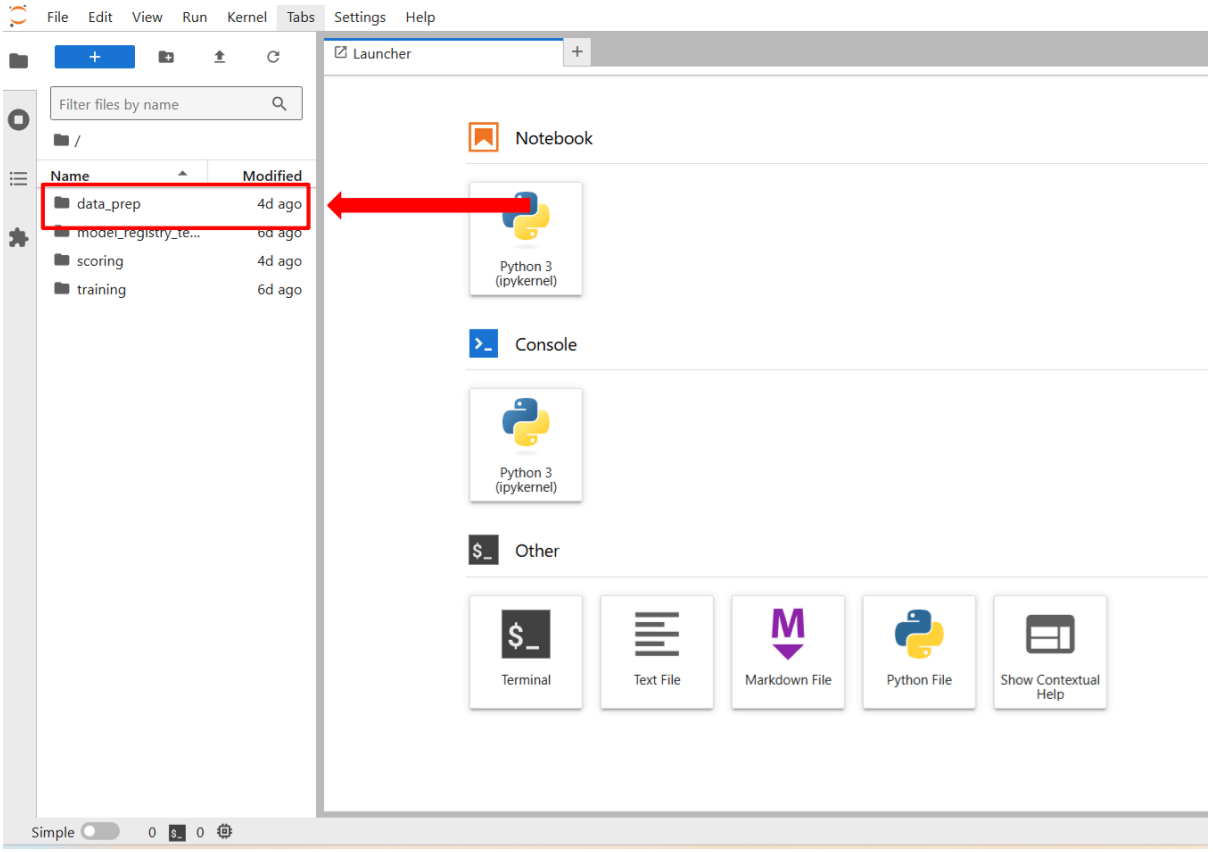
LAB> AI Scenario > Example Scenario>Jupyter Instance



Once inside the Jupyter instance, the user will see a folder structure with three main directories:

- dataprep
- training
- scoring

📁Main folder for this phase: ***Jupiter Lab>dataprep***



These directories are empty by default and are designed to organize the scripts and files required for the AI pipeline.

2. Developing the Notebooks

The Data Scientist begins the data preparation process using `.ipynb` files (**Jupyter Notebooks**). These notebooks act as working drafts used to design, test, and refine the data transformations.

📍 **Typical steps carried out in the notebooks:**

- Loading raw datasets
- Initial cleaning: removing duplicates, filtering anomalies
- Merging multiple data sources
- Applying mappings or enrichment from external reference tables
- Selecting relevant columns
- Generating a **first historical table,** which becomes visible in the **Monitoring view**

3. Visualization and Validation

The resulting datasets are **manually validated** within the Jupyter interface.

KEPLER automatically provides **HDFS dataset paths**, simplifying access, reading, and extraction of the data.

✅ This phase ensures that the transformations are **reliable** before being automated.

II. Automation with Cron Job

1. Transforming the Notebook into an Executable Script

Once data preparation steps have been validated, the Data Scientist :

- Consolidates the key processing logic into a **.py file.**
- Places this file into the appropriate directory (dataprep, scoring, etc.)
- Prepares the script for automated execution using a **Cron Job**

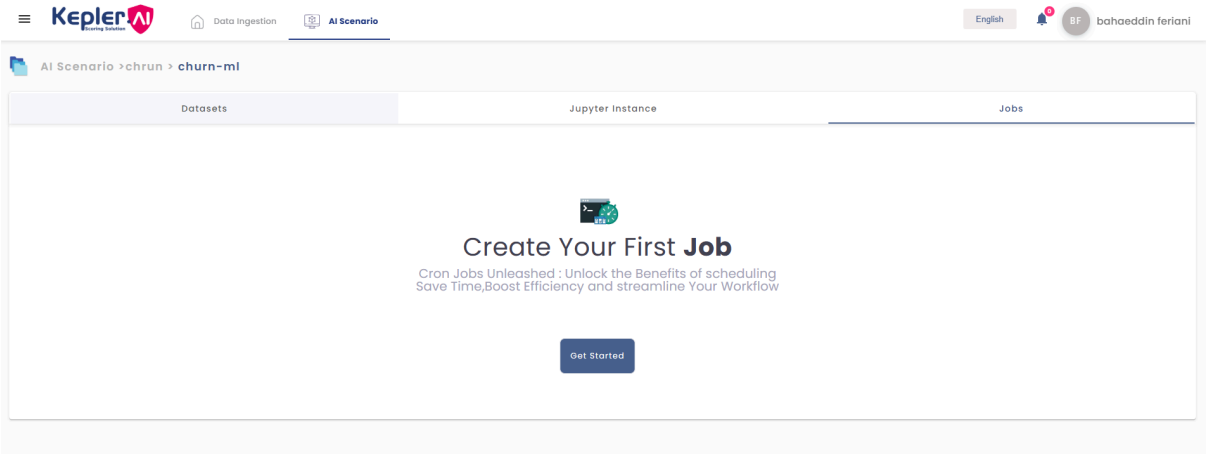
⚠️ Only Python scripts (`.py`) are supported by Cron Jobs—Jupyter notebooks (`.ipynb`) are not compatible.

2. Creating a Cron Job

Kepler provides a **built-in job scheduling system**:

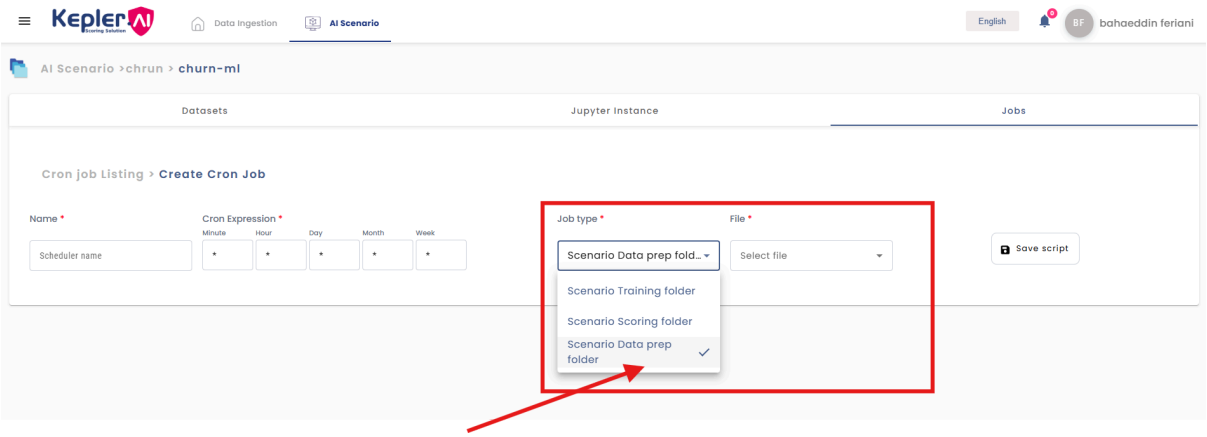
1. From the **Jobs** window, click on **“+”** or **“Get Started”**
2. In the **Job creation interface**, the Data Scientist fills out the required parameters:

LAB> AI Scenario > Example Scenario> Jobs



3. Job Configuration Parameters

LAB> AI Scenario > Example Scenario>Jobs



- **Job Name**
- **CRON Expression***: Defines the script execution frequency
- **Job Type*** :
    a. Scenario Training folder
    b. Scenario Scoring folder
    c. Scenario Data Prep folder
- **Script to Execute (**.py**)**: script final à exécuter automatiquement
- The job is **automatically linked** to the active Use Case, identified by a unique ID in the format `UC_<ID>`.

4. Registration and Execution

Once saved, the job will be executed automatically according to the defined CRON schedule.

The cleaned datasets will be:

- Stored in a new HDFS path, defined within the script
- Ready to be consumed by subsequent steps: training, scoring, or visualization

✅ This automation enables continuous updates of datasets without manual intervention.

Conclusion

The **data preparation** phase in **KeplerAI** is based on a **rigorous and structured approach**, combining:

- A **manual exploratory phase** using Jupyter Notebooks to develop and test data transformation steps

- An **automated execution phase** via Cron Jobs to ensure regularity and sustainability of the AI pipeline

This workflow ensures that data is **reliable, traceable**, and **continuously updated** — all of which are critical to the robust performance of machine learning algorithms.

Par tanitlab AI

Onglet 4:
Data Training

DATA TRAINING

Introduction

Once the data has been **prepared and cleaned** via the **Data Preparation** Job, the **Data Scientist** proceeds to the next stage: **training the artificial intelligence model.**

This phase involves using the transformed data to build a predictive or classification model using machine learning algorithms.

I. Manual Phase in Jupyter

Before any automation is put in place, the Data Scientist conducts experiments and model development within the Jupyter Lab environment—similar to the data preparation phase.

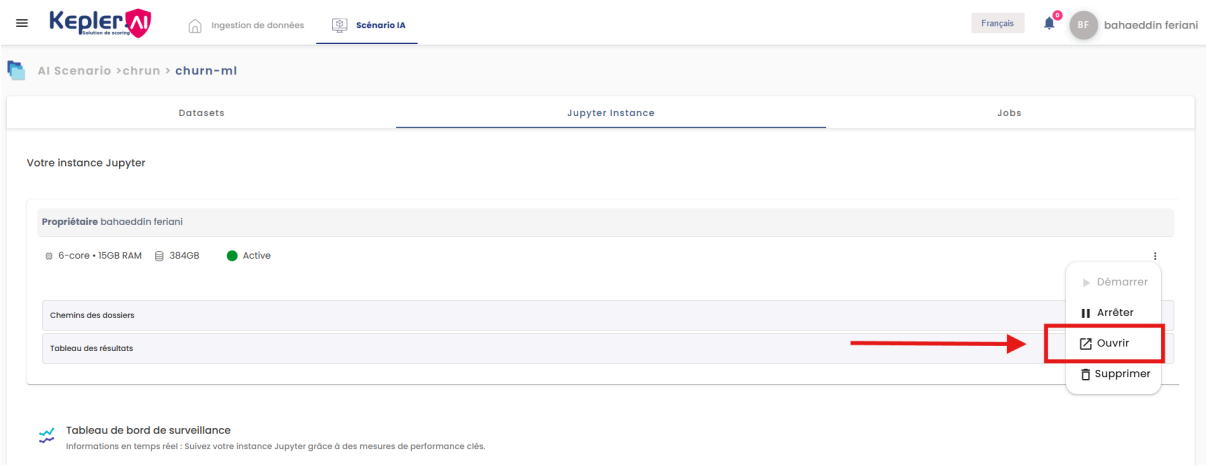
This phase is entirely manual, without the use of Cron Jobs, and allows for the design, testing, and validation of the notebooks used for model training.

1. Accessing Jupyter Environment via KEPLER

The Data Scientist can access the Jupyter environment through the KEPLER interface by following these steps:

1. From the **Home** page, navigate to the **LAB** module;
2. In the sidebar menu, click on **AI Scenario**;
3. Select an existing scenario from the list;
4. In the **Jupyter Instance** section:
    - If the instance is **inactive**, click **Activate**;
    - Once activated, click **Open** to launch Jupyter Lab.

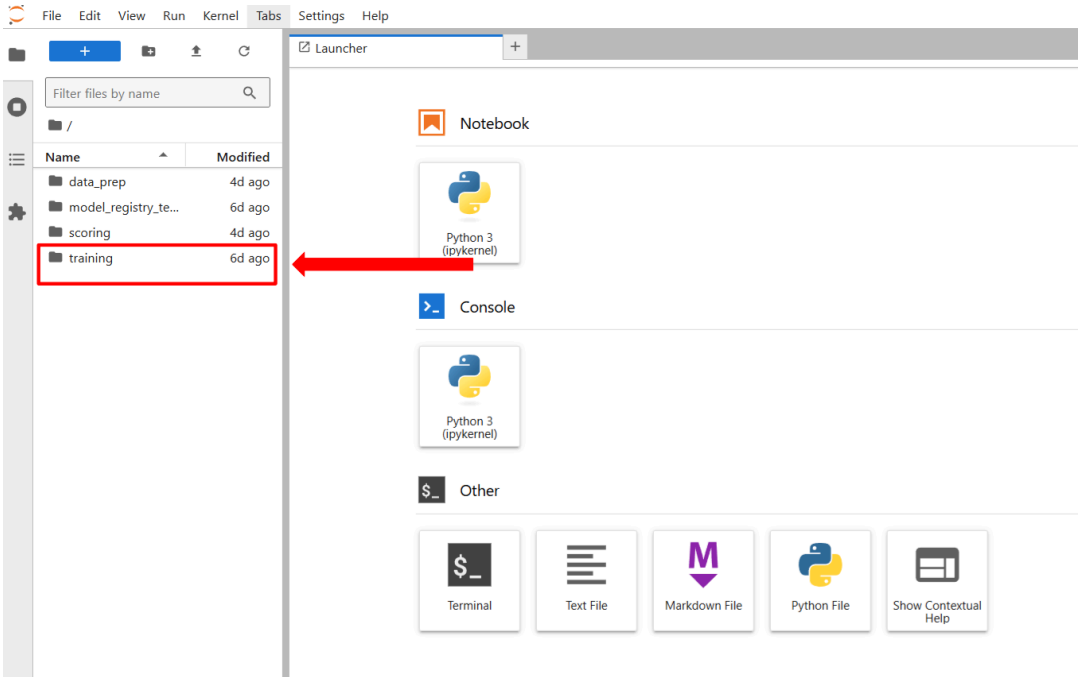
LAB> AI Scenario > Example Scenario>Jupyter Instance



2. Working in the Training Directory

Once in Jupyter, the Data Scientist navigates to the training folder within the scenario workspace. There, they create one or more .ipynb notebooks for experimentation and model training.

📁 Main folder for this phase: ***Jupiter Lab>training***



📍 Typical Steps in a Training Notebook:

- Load the cleaned dataset from the **HDFS path** generated during the previous dataprep phase.

- Perform **feature engineering**: selection, transformation, encoding, or normalization of variables
- **Split the dataset** into **training and test** sets, or apply **cross-validation** techniques
- Train the model (e.g., XGBoost, Random Forest, SVM, etc.)
- **Evaluate the model** using metrics such as **accuracy, recall, AUC, RMSE**, …
- Save the trained model to an HDFS directory (/model or /scenario_training_output)

📌 At this stage, **training is entirely manual** and not part of any automated Cron Job. It is executed **only once unless the model** needs to be retrained.

II. Automated Training Job Creation

In some situations—particularly when the dataset is updated frequently—it may be appropriate to automate the training process using a **Cron Job**.

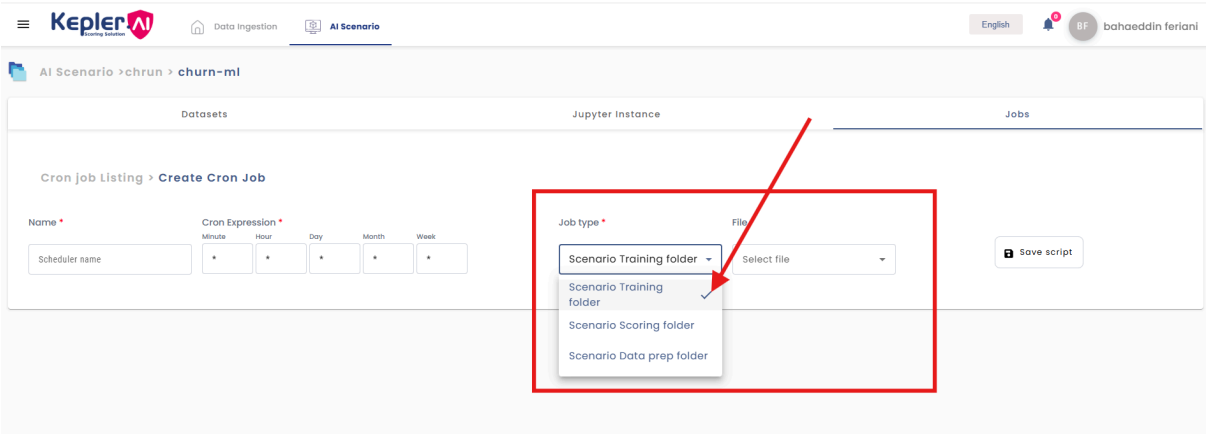
⚠️ Model training is generally **not intended** to be scheduled as frequently as Data Preparation or Scoring. In most cases, the training process is executed **only once**, or **manually** upon request.

1. Creating the Training Job in KEPLER

Once the training script has been exported from the .ipynb notebook to a .py Python file, the job can be created directly from the interface.

Navigation path:

LAB> AI Scenario > Example Scenario>Jobs



🛠️ Training Job Configuration Parameters:

- **Job Name:** e.g., Training_Model_XGB_V1
- **Cron Expression (optional):** only if periodic retraining is required
- **Job Type:** Scenario Training Folder
- **.py File:** the final Python training script

📌 The script must first be authored and tested within **Jupyter**, and then **manually selected** when configuring the job in the KEPLER interface.

III. Objectives of the Training Phase

The objective of this step is to produce an efficient AI model from the processed data.

The main actions include:

- **Utilizing high-quality data** output from the Data Preparation phase
- Building a reliable and robust model
- **Evaluating its performance** on test datasets
- **Saving the trained model** in **HDFS** for use in the next step (Scoring)
- **Analyzing results** and logs from the monitoring interface

IV. Visualization and Adjustments

Once the Job has been executed, the Data Scientist can:

- Review **execution logs** directly in **KEPLER**
- Monitor the **model's performance** (e.g., plots, evaluation metrics)
- Return to **Jupyter** to **fine-tune the code** if needed
- Reschedule a retraining if the results are unsatisfactory

Conclusion

Model training is a **critical stage in the AI pipeline**, where structured data is transformed into usable **predictive models.**

Unlike Data Preparation or Scoring, the **Training Job is rarely automated**, as it typically involves significant manual development and iterative adjustments in Jupyter.

📌 The trained model will later be used in the **Scoring phase**, which *can* be automated for daily or periodic evaluation tasks.

Par tanitlab AI

Onglet 5:
Data Scoring

DATA SCORING

Introduction

Once the model has been trained and validated in the previous phase, the Data Scientist proceeds to the final step of the AI pipeline: **scoring**.

This step involves applying the saved model to new data in order to generate actionable predictions (scores, classes, probabilities, etc.).

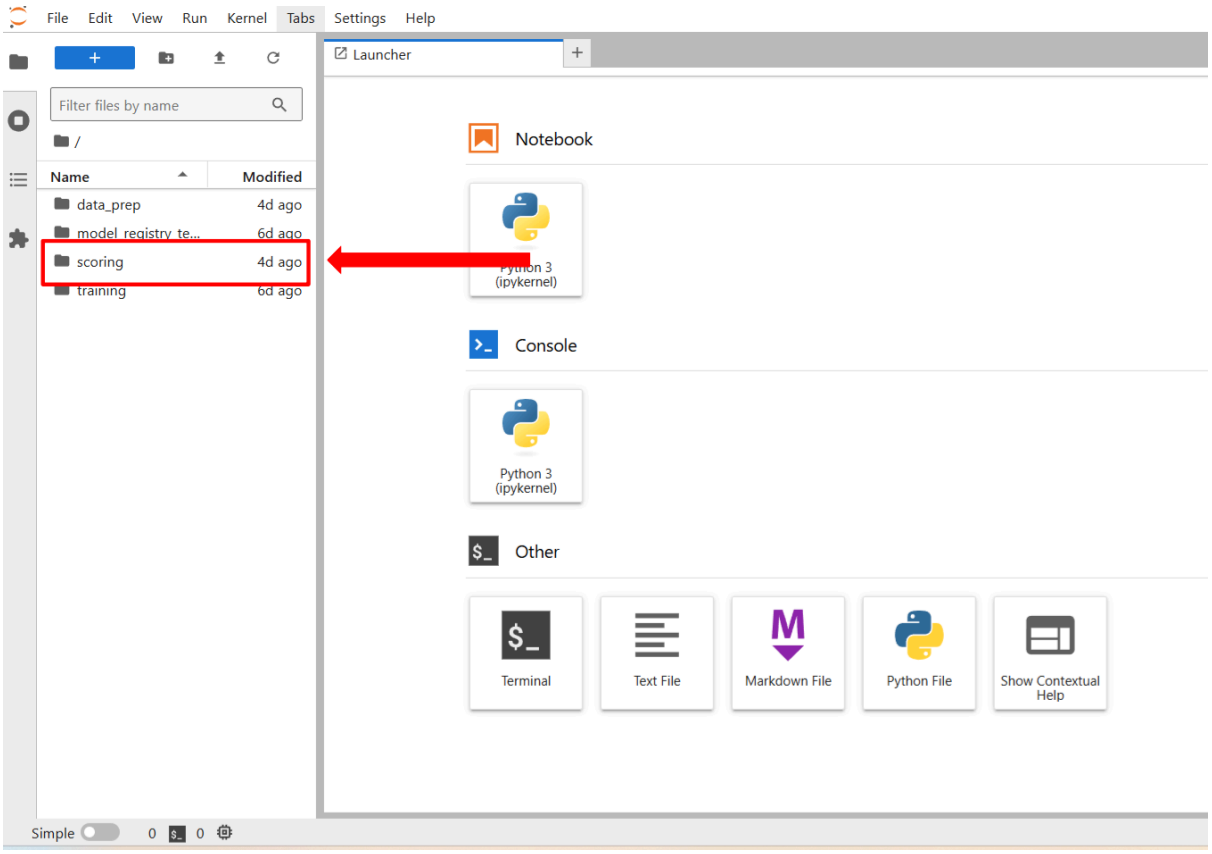
It is typically performed in two phases:

- First, a manual development phase in **Jupyter** to test and validate the process.
- Then, an automated execution via a **Cron Job** within the LAB interface for regular production runs.

I. Manual Phase in Jupyter

As with data preparation and training, the Data Scientist begins in **Jupyter Lab** to develop and test the scoring logic.

📁Main folder for this phase: ***Jupyter Lab > scoring***



📍Typical steps in the scoring notebook:

- Load the trained model (from /model or scenario_training_output folder)
- Load the new data to be scored (from an HDFS path or business table)
- Apply the model to the data
- Perform any necessary post-processing: score normalization, business tag enrichment, etc.
- Save the output results to an HDFS directory (typically /scenario_scoring_output)

📌 **This manual phase** ensures that the scoring logic works as expected before proceeding to automation.

II. Automation with a Cron Job

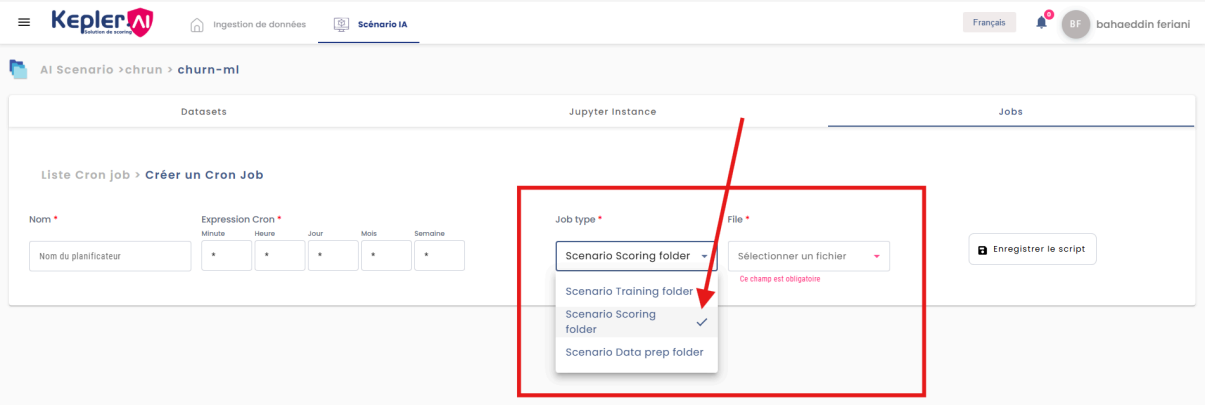
Once the scoring notebook has been validated, it is converted into a `.py` script, with the following requirements:

- The script must be written in **Python** (⚠️PySpark is not supported in Cron Jobs)
- The script must be saved in the scoring directory within the project structure

The Data Scientist can then automate the execution via the **Jobs** interface in KEPLER.

Navigation path:

LAB> AI Scenario > Example Scenario>Jobs



🛠️**Scoring Job Parameters**:

- **Job Name**: e.g., Scoring_V1_Risk_Client
- **Job Type**: Scenario Scoring Folder
- **Script Type**: Python (mandatory)
- **Cron Expression**: defines the execution frequency (e.g., 0 18 * * * → every day at 6 PM)
- **.py File**: the validated scoring script developed in Jupyter

III. Objectifs de l'étape de Scoring

The scoring phase marks the **final stage of the AI pipeline**. Its primary purpose is to **leverage the trained model** to generate actionable predictions on updated data in a regular and automated manner.

Beyond merely applying the model, this step aims to **structure and deploy the outputs** within an appropriate technical architecture, ensuring their seamless integration into monitoring, decision-making, and analytical tools.

⚙️ Expected Outputs and Associated Actions:

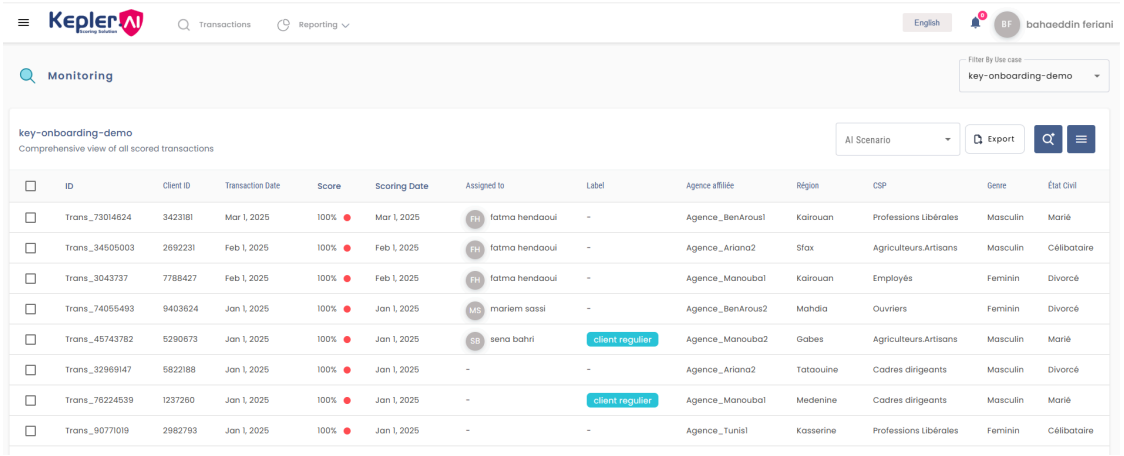
- **Automatic loading** of the validated model (previously saved in the scenario_training_output directory)
- **Application of the model** to new datasets originating from HDFS or business sources.
- **Prediction generation**: scores, probabilities, or class labels depending on the use case.
- **Business-specific post-processing**: normalization, semantic enrichment, filtering, tagging, or categorization.
- **Structured storage of results** in two complementary locations:
    - An **HDFS table** containing all calculated columns and analytical metadata.
    - A **PostgreSQL table**, configured with customized columns for optimized visualization in **KeplerAI**'s Monitoring module dashboards.

🔗 Integration into KeplerAI Monitoring

Within the **KeplerAI** Monitoring interface :

- The **main dashboard** is populated from the PostgreSQL table, chosen for its lightweight structure and visual clarity.

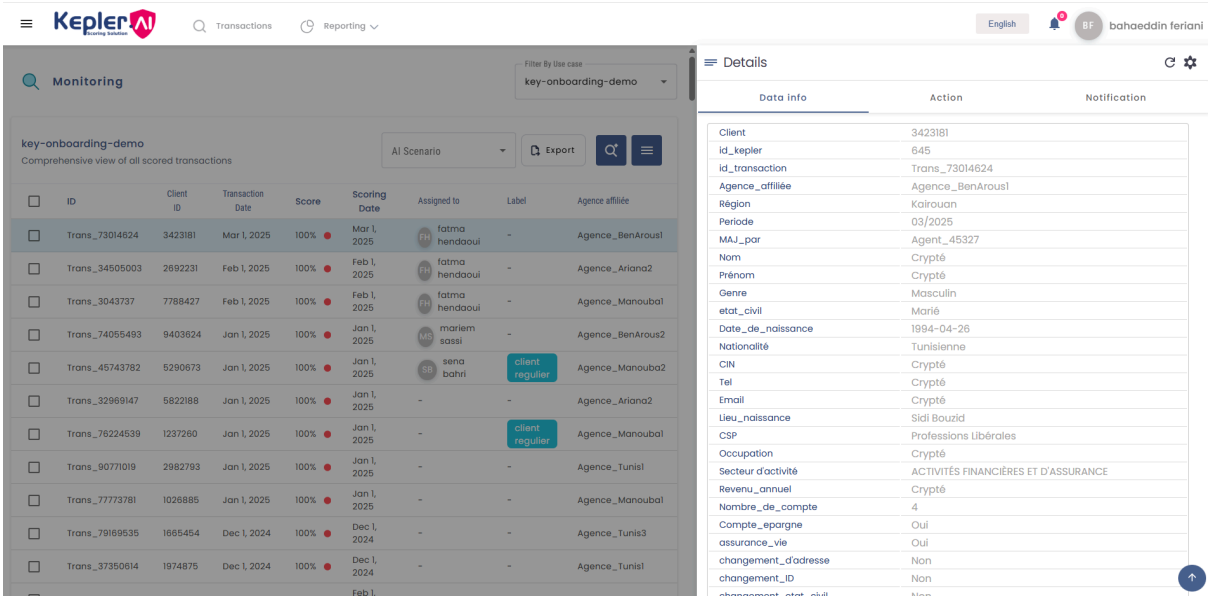
Modules exemple "onboarding">Monitoring



- Each row corresponds to a prediction or transaction.

- When a row is selected, a **contextual popup** opens dynamically. This popup performs a live join using the id_kepler identifier to query the HDFS table and display detailed information (features, scores, IDs, etc.).

Modules exemple "onboarding">Monitoring>Details



🚀 Final Deployment

Scoring deployment goes beyond merely executing a Python script. It entails a full orchestration of the following deliverables:

- Generation of the **complete HDFS table** from scoring outputs.
- Creation of a **lightweight PostgreSQL table** tailored for UI display.
- Setup of **automatic linkage** between both tables via the id_kepler field.
- Verification of the Monitoring interface's behavior (table + pop-up functionality).
- Integration into **business analytics and decision-making workflows**.

✅ This approach ensures full traceability, smooth consultation of predictions, and robust integration into existing business tools.

IV. Execution and Automation

Once configured, the Job runs automatically according to the defined schedule. It becomes a key component of the AI production chain, enabling real-time or periodic monitoring of key phenomena (risks, anomalies, customer behavior, etc.).

📌 All automated Jobs are visible and traceable via the KEPLER interface. Execution logs and potential errors are accessible for easier debugging and maintenance.

Conclusion

Scoring represents the final, operational stage of the AI pipeline. It applies trained models to new data to generate business value.

Unlike the training phase, scoring is automated by default, as it relies on a stable, reproducible process designed to run on a recurring basis.

📌 Only Python scripts are supported by Cron Jobs. Therefore, the scoring process must be tested, converted to .py, and manually selected during Job creation.