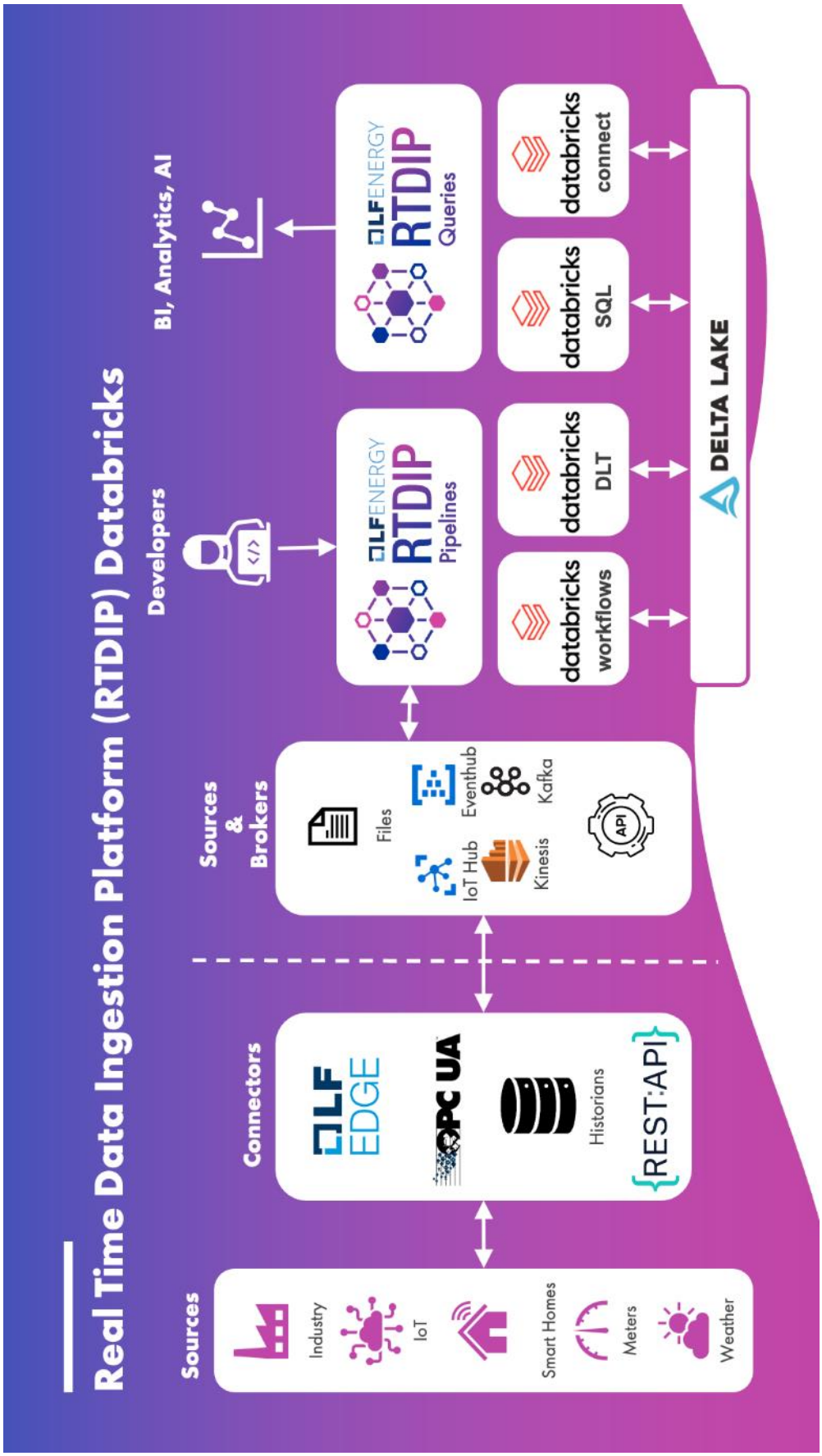
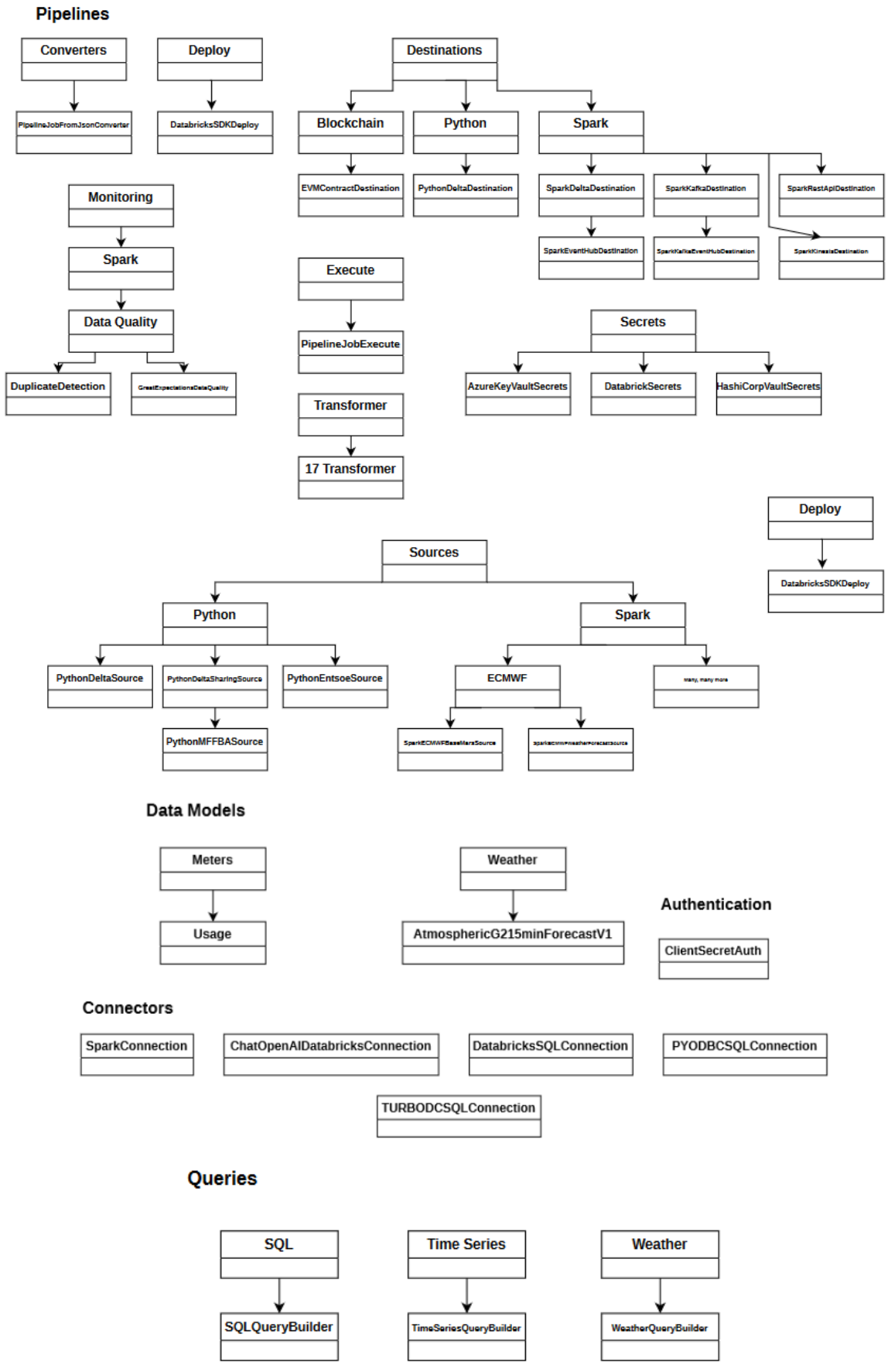
Runtime Components



Code Components



Summary of Technology Stack

The pipeline SDK of RTDIP is implemented in *Python*. Since we continue an already existing project, there are roughly 80 direct dependencies. Notable mentions among these are the well-known python packages *pandas* and *NumPy*, which are used to process the incurring sensor data. For using the cloud computing services *AWS* or *Apache Spark* RTDIP is dependant on packages like *boto3* or *pyspark*, which act as an interface to the respective services. The cloud computing services *Databricks* and *Azure* are supported as well. The package *delta-spark* is interfacing Apache Spark with Delta Lake, an open-source storage layer for data lakes. The web templating engine *jinja2* is used for the Query Builder. For testing data quality of incoming, the package *Great Expectations* is used. As we will extend the Data Analysis tools provided by RTDIP, it will be likely that additional libraries like *scitkitlearn* or *SciPy* will be included for gathering new insight from the collected data.

The quality of the code contributions is continously tested with the usage of the code formatter *black* and the testing tool *pytest*. The tools are automatically run on each *Github* commit with a pipeline provided by *Github Actions*. Further analysis of the code for vulnerabilities is provivided by the online cloud service *Sonarqube cloud*. The documentation is written in markdown and is generated by *mkdocs*.

Explanations of Diagrams

1. Pipeline Framework

This diagram shows the data pipeline, which time series data travels through from a source until to a storage location, and how RTDIP is involved in it. RTDIP has the capabilities to adapt to a great variety of data format, data sources & brokers. The RTDIP Pipeline SDK is responsible for processing incoming sensor data with a streaming or batch ingestion pipeline and saving it into data storage. RTDIP Query on the other hand reads the saved data and allows for discovering insights, which can be used further in business decision processes or the training of artificial intelligence.

2. Functions

This diagram shows the existing code files sorted by their storage location in repository. Each node is a .py-file.

The already existing codebase is split up into the two main modules of the *Pipeline SDK* and the *Query API*. These are supported the three additional modules for *Authentification* functions, application-specific *data models* and *Connectors* for different services. Of note here is the Pipeline SDK module. A pipeline is here created by chaining of components of five different types of components:

- *Destination*, where data is stored
- *Monitoring, where data is analyzed without modification*
- *Secrets*, for accessing sensitive protected information
- *Sources*, where sensor data is imported into the pipeline
- *Transformers*, which can modify the data for e. g. preprocessing

The goal of the current AMOS project is to add additional Transformers & Monitoring to extend the Data Analytics capabilities of the project.