Innehåll

[1. Revision History 2](#_Toc203331090)

[2. Purpose 2](#_Toc203331091)

[3. Cloud architecture design 3](#_Toc203331092)

[3.1 Input data 3](#_Toc203331093)

[3.2 Data engineering layer 4](#_Toc203331094)

[3.3 Data science layer 4](#_Toc203331095)

[3.4 Service layer 4](#_Toc203331096)

[3.5 Internal insight API 5](#_Toc203331097)

[3.6 Important for all components 5](#_Toc203331098)

[4. ETL script 5](#_Toc203331099)

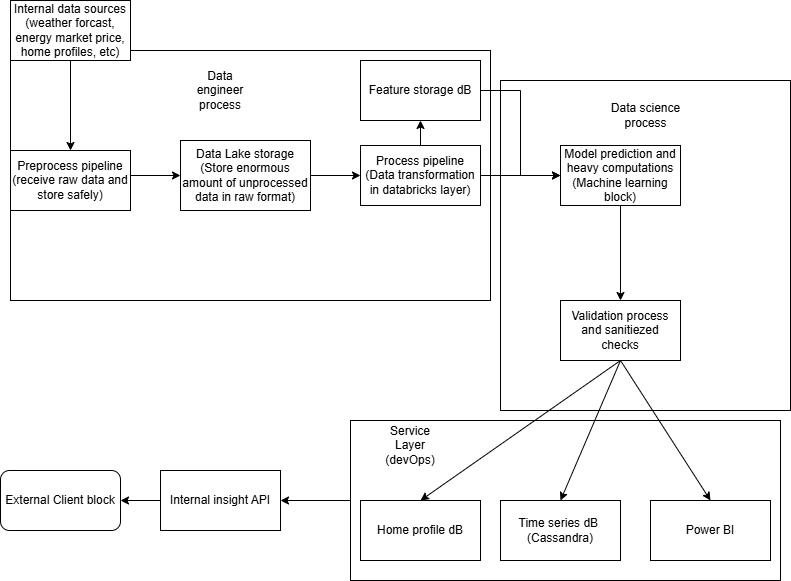
# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| REV | DATE | SIGN | COMMENT |
| PA1 | 20250713 | RS | Created |

# Purpose

The purpose of this document is to present a proposed solution for the home assignment provided by Eliq. The objective is to give a high level description of a design for a scalable and modular platform and provide a suitable cloud architecture. A practical ETL script that transform time series energy consumption data into a format more suitable for analysis and usage.

# Cloud architecture design



Figur 1 Cloud architecture

# Input data

The image above is a proposed solution for a design for a scalable and modular platform. The first step is to review what kind of input data we have to the system. In this case we need to consider weather forecasts, energy market prices, home profiles and meters and other type of sensors installed. These sources may arrive at varying frequencies, for example monthly to 15 minute intervals. Even different protocols may be used depending on the sources here such as REST APIs or MQTT brokers.

# Data engineering layer

The purpose of this block is to safely extract relevant data, store data safely, clean and sanitize data for the data science block. The receiving of data from the sources can be done with a REST API to consider traffic on HTTPS or MQTT broker depending on how the data looks. Considering that the data can have a frequence of monthly to 15 minutes interval an MQTT is highly recommended. This is why the blocks are divided into 3 separate blocks. The first block (preprocess pipeline) is a REST API hosted with azure app function and monitoring settings with authentication and TLS recommended. The purpose is to constantly receive data and send it to the data lake to store the raw format safely, in case something goes wrong in the entire flow the raw input is stored and can be reprocessed. The data lake is storing all raw values in a structured way. Allowing scalable and schema consistent ingestion for all input sources. The process pipeline requests data from the data lake which is also its own azure app function with monitoring. The purpose is to extract the data for necessary features requested by the data science block. The features are stored in a separate Azure SQL database for same purpose.

# Data science layer

The Data science block purpose is to receive data in requested format. In this case the input to the model is forecast data, usage categorization. Mostly depended on how the ML model is built. Both raw format and feature format is very useful. The Data science block contains the high computational process where ML models are trained and hyper tuned. The ML is validated before sending result to the services block to ensure predictions meet accuracy and performance thresholds.

# Service layer

The service block purpose is to provide data which is expected to be used by the client. This block can also be used for collecting result and output meaningful insights about the data using power BI. The service block separates the home profile outputs in its own azure SQL database. Other useful time series result is stored in a separate database like Casandra for example forecasted usage.

# Internal insight API

The internal insight API is a gateway for external clients to access energy insights. This is the communication link the client has to the platform. The important job is to only display what the agreement is with the clients. The purpose is to validate outputs. For example, forecasted consumption or solar disaggregation. Any data quality, formatting or security issues are handled in the layers above to ensure stable and consistent interface.

# Important for all components

All the blocks have a high expectation on self monitoring all their own components with azure log analytics and Azure monitor and cloud costs. It is highly recommended that there all vulnerability is tested and proven for end-to-end reliability. Meaning unittest, integrationtest and systemtest is highly important. The communication between each block can be whitelisted IP addresses and/or role based access control for who can read, write and access. Each component has high standard of input validation and encryption. Each component should have versioning handling and must go through tests before deploying. The CI/CD can be done in github with YAML script. The components should be dockerized and be stored in azure container registry in the cloud.

# ETL script

Script can be found in github. Follow Readme to run the script.