|  |  |
| --- | --- |
| Streaming Predictive Maintenance | ARCHITECTURE EXPLAINED |

Version/revision: 1.1

Date: 31/07/2017

Authors: Pablo Doval, Jose Vizoso, Francisco Martínez, Daniel Escribano

Change Control

| DATE | VER./REV. | DESCRIPTION |
| --- | --- | --- |
| 31/07/2017 | **1.1** | Added batch details, updated architecture diagrams. |
| 25/07/2017 | **1.0** | Initial Version. |

Contents

[1. Introduction 4](#_Toc489273412)

[2. Proposed system 4](#_Toc489273413)

[2.1. Data Ingestion 5](#_Toc489273414)

[2.2. Streaming Data Processing 6](#_Toc489273415)

[2.3. Batch Data Processing 8](#_Toc489273416)

[2.4. Monitoring 8](#_Toc489273417)

# Introduction

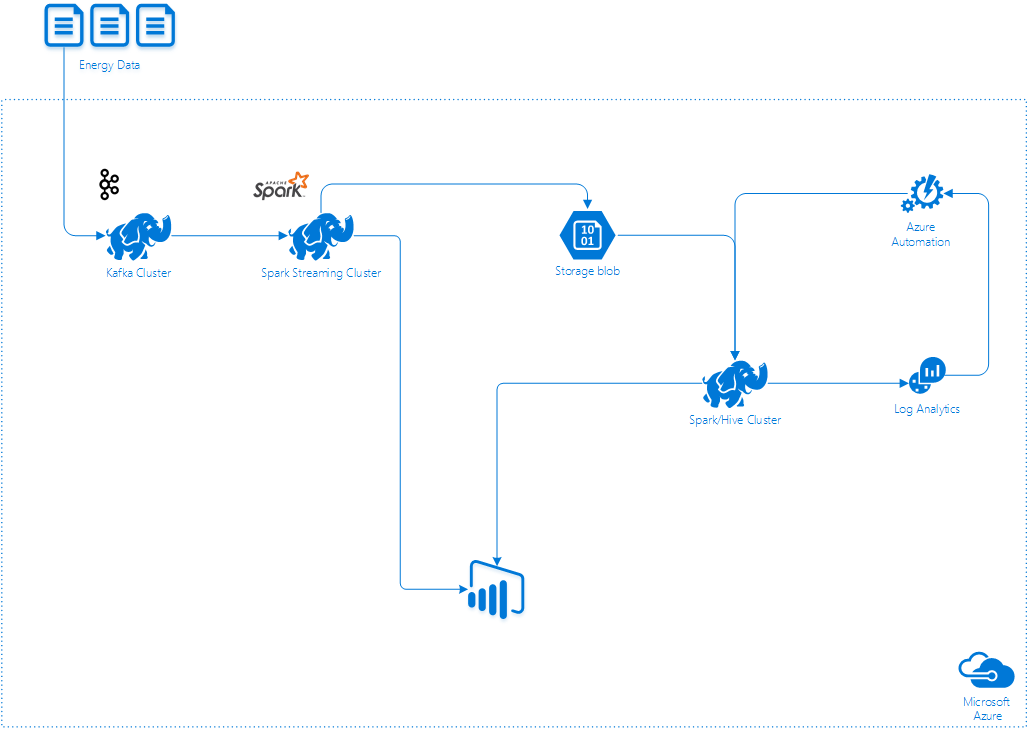
This document will walk through the architecture of the Energy demo streaming, providing details about the proposed scenario and the different components that are part of it.

The idea behind the project is to recreate a scenario that enables fast and easy analysis of the current situation of the transformers in the energy grid of an energy company. Incoming data from the transformer sensors will be analyzed and pushed to a real-time dashboard, where it can be consulted to see how each transformer is performing, and whether human intervention is required.

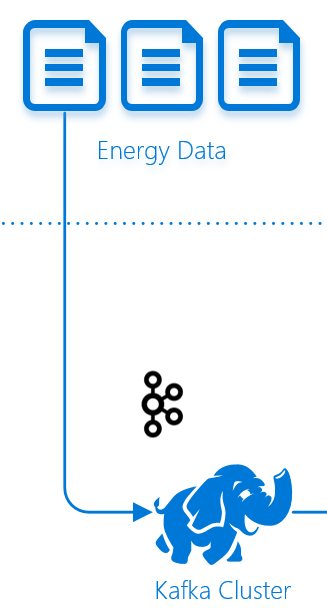
On top of that, a machine learning experiment will analyze the streaming data received from the transformer sensors, and look for anomalies on the time series patterns. That will also be presented as part of the real-time dashboard, helping drive important maintenance decisions.

# Proposed system

An overview of the proposed architecture for this demo scenario can be seen in the diagram below:



* An HDInsight Kafka cluster is the entry point of the system. In a real-world scenario, substations will send data from their transformer sensors to the cluster. For the purposes of this demo, a C# application will simulate real time ingestion from live sensors in the field.
* Data will be processed in real time by a Spark HDInsight cluster, using Spark Streaming. Data pulled from the Kafka queue every few seconds will be processed and analyzed to detect anomalies using a machine learning model. The results of this analysis and the raw data will then be pushed to a real-time PowerBI dashboard, and to Azure Blob Storage to enable further processing.
* Data stored in Azure Blob Storage will be processed by the Spark cluster using Spark to perform the re-training of the machine learning model. The re-trained model will be stored in blob storage so it can be used by the streaming process.
* OMS Log Analytics will be used to gather information about the cluster, including usage levels. If (when) those levels move beyond a set of thresholds, an alert will be triggered and, through Azure Automation, the cluster will be upscaled or downscaled as required.
  1. Data Ingestion



**eSmart API**

To simulate real time ingestion, a C# application has been developed to push data continuously to the Kafka cluster. Synthetic data has been generated and stored in a SQL Server database. The C# application will pull data from there and modify the timestamp to simulate the real-time behavior.

The generated data consists of time series generated by different sensors and transformers, that are contained within substations. These time series contain the following values, with a frequency of one second:

* Load
* Voltage
* Temperature

These variables are related, so the anomaly detection model works will all three series simultaneously to detect anomalies in the combined values.

**Kafka cluster**

Apache Kafka is an open-source distributed streaming platform that can be used to build real-time streaming data pipelines and applications. Kafka also provides message broker functionality similar to a message queue, where you can publish and subscribe to named data streams.

Azure HDInisight allows the creation of Kafka clusters as a managed, highly scalable and highly available service in the cloud.

It is not the purpose of this document to explain in depth how Apache Kafka works, but its components consist of topics, producers and consumers.

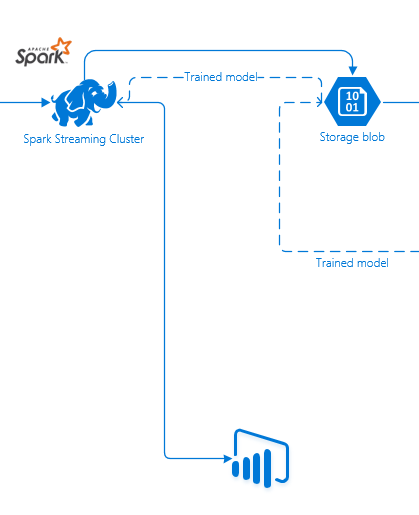
* A topic is a category or feed name to which records are published.
* A producer publish data to the topics of their choice.
* Consumers label themselves with a consumer group name, and each record published to a topic is delivered to one consumer instance within each subscribing consumer group.

The proposed system will have a topic where the time series will be published.

The producer would be a substation (grouping of transformers equipped with sensors). To simulate this behavior in the demo scenario, the C# application mentioned previously will be used to publish time series values to the Kafka topic.

And, lastly, the consumer in our system would be the Spark Streaming job running in the Spark cluster, which will process the data published to the topic.

* 1. Streaming Data Processing



**Spark cluster**

The data stored in Kafka will be processed by an application running in the Spark cluster. This application will be created in Python, and will use the Spark Streaming API to process data in real time.

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Data can be ingested from many sources like (Kafka is among those sources) and can be processed be pushed out to filesystems, databases, and live dashboards later.

Basically, what the application will do is create a stream using Kafka as the source. Every few seconds, the application will process a window of values retrieved from Kafka and will evaluate them against a model previously trained to identify anomalies in the data.

**Power BI dashboard**

Processed data will be outputted to a Power BI dashboard in which the user will be able to see the data sent by the existent substations, as well as quickly detect if there is an anomaly.

Power BI provides streaming features to show real time data as part of a dashboard. There are two special types of datasets that has been considered for the development of this scenario:

* + Push dataset
  + Streaming dataset

The main difference is that when a push dataset is created, Power BI creates a database to store the data, meanwhile with streaming datasets there is no database. In this last one, the data is stored in a temporary cache that quickly expires.

The visuals created using a push dataset can be pinned in the dashboard and they will be automatically refreshed when new data is received. However, it is not possible to pin an entire report in the dashboard and having its visuals automatically updated.

On the other side, streaming datasets are useful when it is preferred to minimize the latency between the data being pushed and the visual being updated. However, they cannot be used to build reports, since there is no underlying database. This results in the impossibility to use report functionality like filtering, custom visuals and other functions.

Since the idea is to create more sophisticated visuals that can be filtered by several parameters and can contain additional data that provides more context to the user, a push dataset is used.

**Blob storage**

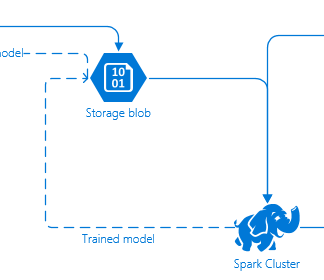
Besides outputting data to Power BI, the Spark cluster will also storage the raw data and the streaming analytics results in an Azure Blob storage account, so it can be used by other applications. Within the boundaries of this demo, it will also be used to enable the re-training of the model, as explained in the next section.

Azure Blob Storage is a specialized storage solution for storing unstructured data offering great durability, availability, scalability, and performance. It offers two different access tiers:

* **Hot:** for frequently accessed data
* **Cool:** for less frequently accessed data, but with a considerable cost reduction in comparison with the hot storage.

Azure Hot Blob Storage will be used, since it provides better performance and availability.

* 1. Batch Data Processing



As stated before, raw data will be also outputted to a Blob storage account.

This data will be used during the re-training of the model, allowing the system to evaluate if there are anomalies with better precision.

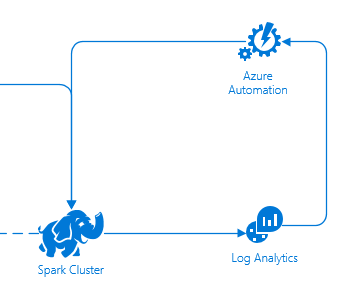
In a real scenario, every day, the system would pick the five latest files of data available in storage. Each file corresponds to one day of data. All this information is taken only from a single device, since it is assumed that all the data arriving to the system comes from devices of the same type as the one used to train the model.

In our scenario, the user will be in charge of executing the script that re-trains the model when he is told, so he can see by himself how the process is done. He will be provided a few files that simulates data stored from the last five days.

After training, the model will be serialized and stored in Azure Storage.

The Spark Streaming job will check periodically if there is a new version of model available to evaluate data against. In that case, it will be downloaded and the job will start using it to determine if there are anomalies.

* 1. Monitoring



**Log Analytics**

Log Analytics is a service in Operations Management Suite (OMS) that monitors cloud and on-premises environments to maintain their availability and performance. It collects data generated by resources in the environments and from other monitoring tools to provide analysis across multiple sources.

To maintain a good performance of the Spark cluster, the system will use Log Analytics so it can monitor the CPU use percentage of the nodes. When the average of CPU usage over a period crosses some defined thresholds, it will create an alarm that will trigger a runbook in Azure Automation.

**Azure Automation**

Microsoft Azure Automation provides a way to automate manual and frequently repeated tasks that are commonly performed in a cloud and enterprise environment. It saves time and increases the reliability of regular administrative tasks and even schedules them to be automatically performed at regular intervals. Runbooks can be used to achieve this automation. A runbook is a set of tasks that perform some automated process.

As described previously, the Log Analytics alarm will trigger a runbook, that will either add or remove nodes from the Spark cluster, depending on the crossed threshold.