IoT y Analítica – Conectando TO y TI

Juan David Chaverra Cuartas

Informe de practica académica para optar al título de Ingeniero Electrónico

Asesor
David Stephen Fernandez Mc Cann, PhD

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería Electrónica
Medellín
2023

Cita	(Chaverra Cuartas, J. D. (2023))
Referencia **Estilo APA 7 (2020)**	Chaverra Cuartas, J. D. (2023). IoT y Analítica – Conectando TO y TI [Semestre de industria]. Universidad de Antioquia, Medellín

Asesor de prácticas: David Stephen Fernández Mc Cann

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia

Sistema de Bibliotecas

Centro de Documentación Ingeniería (CENDOI)

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda Cespedes.

Decano/Director: Jesús Francisco Vargas Bonilla.

Jefe departamento: Augusto Enrique Salazar Jiménez.

Contenido

1. Resumen

Recientemente el concepto de IoT ha tomado mucho poder en la industria, especialmente donde se monitorean procesos críticos, con el objetivo de obtener información en tiempo real del estado de equipos para prevenir fallas o alertar sobre procesos que no están cumpliendo con la función que se espera. En el presente artículo se describe una solución sobre el monitoreo de sistemas eléctricos de una planta de energía, utilizando herramientas de nube con la plataforma Azure y servicios de IoT, que permiten crear una telemetría para leer los datos en un tiempo cercano al real (menos de un segundo de latencia), para alertar sobre posibles fallos de los generadores, con el fin de garantizar el funcionamiento de los mismos y realizar mantenimiento preventivos o intervenciones antes de una sobre carga del generador. Se describe la arquitectura utilizada y la configuración de cada uno de los servicios utilizados para cumplir con tal objetivo, que finalmente se presenta una visualización sobre los datos recopilados para analizar el estado del generador de energía.

Palabras clave: Azure, IoT, Tiempo Real, datos, generador, visualización, telemetría, SCADA

2. Introducción

El presente artículo busca generar una fuente de conocimiento general del funcionamiento de una solución de IoT que permite extraer los eventos de generación eléctrica, con soluciones IoT y Analítica de Azure, por medio de la transmisión de telemetría desde sistemas SCADA.

Este proceso se realizó, como una iniciativa para estandarizar y centralizar las variables que interfieren en los diferentes procesos, especialmente los relacionados con el sistema eléctrico. Y de esta manera mejorar el sistema actual que cuentan en la compañía donde analizan los datos cada 10 minutos por que el proceso funciona en Batch, es decir, se ejecuta en un tiempo fijo; la solución realizada para solventar la latencia tan grande que se tenía para analizar los datos recopilados, se base en el uso de servicio que funcionan en streaming, que en otras palabras quiere decir que son servicios que extraen información en un tiempo cercano al real, donde la latencia es menor a un segundo; el tiempo puede variar un poco debido a la volumetría de la data a la hora de extraer los datos.

Adicional a la creación de los servicios de streaming para extraer los datos en un tiempo cercano al real, se debe crear un traductor OPC DA a OPC UA, con el fin de establecer una comunicación entre el servidor donde se encuentra el Scada y el módulo de Azure Edge, que se resumen en la comunicación de los datos en local con la nube. Este protocolo debe ser creado en el servidor del Scada

El resultado final del artículo se presenta las visualizaciones creadas para el análisis de los datos, en un tiempo cercano al real, donde se puede observar la telemetría de la información y como se está comportando el generador eléctrico conectado con el scada

A continuación, se expondrá de forma detallada cada una de las actividades que se llevaron a cabo durante el desarrollo del proyecto.

3. Objetivo General

Desarrollar e implementar una arquitectura en el ambiente de desarrollo Azure[1] que permita correlacionar eventos de generación eléctrica y su afectación en la operación, a partir de datos obtenidos mediante sistemas de Telemetría, con el fin de monitorear en tiempo real la operación de un sistema eléctrico.

3.1. Objetivos específicos

- Crear módulo traductor de protocolo de OPC DA a OPC UA, para tener la comunicación entre los componentes de Azure Edge y el servidor de SCADAS, el cual se crea directamente en el servidor SCADA

- Desarrollar arquitectura en la plataforma Azure, con servicios PaaS para él envió de la telemetría y visualizaciones, cercanas al tiempo real (latencia 1 segundo)

- Implementar alertas mediante los servicios de IoT de Azure para garantizar la calidad de los datos; además de utilizar los mismos servicios para la integración de los datos con otras fuentes de información para centralizar el Core de negocio para visualizaciones y análisis.

- Asegurar el envío seguro de la información, con latencia menor a un segundo, mediante los servicios de IoT de Azure diseñados para transportar grandes volúmenes de datos en poco tiempo, realizando el proceso de telemetría

[1] Plataforma de nube de Microsoft https://azure.microsoft.com/

4. Marco teórico

TO hace referencia a la tecnología Operativa, que se define como el uso de hardware y software para monitorear y controlar los procesos físicos, los dispositivos y la infraestructura. La tecnología de la información (TI) es el proceso de creación, almacenamiento, transmisión y percepción de la información y los métodos de aplicación de dichos procesos**.** Ambos conceptos son los que se quiere unir en el proyecto para fusionar las tecnologías que cuenta la empresa y sacar mayor provecho a los datos que ya se están recopilando.

La plataforma utilizada para unir las dos tecnologías descritas anteriormente es Azure, la cual es creada por Microsoft y que cuenta con servicios Iaas y Paas. Para el proyecto utilizamos solo servicios PaaS, que lo que implica es la combinación de servidores, almacenamiento e infraestructura de red con el software que necesita para desplegar aplicaciones, con PaaS, no necesita intervenir hardware local in preocuparte por configurar un entorno virtual para gestionar las aplicaciones.

Además de servicios Paas, se utilizarán servicios orientados a IoT, o internet de las cosas que es un sistema de dispositivos informáticos interrelacionados, máquinas mecánicas y digitales, objetos, animales o personas que cuentan con identificadores únicos y la capacidad de transferir datos a través de una red sin necesidad de una persona a otra.

Un ejemplo de IoT en la vida real puede ser una persona con un implante de monitor cardiaco, un animal de granja con un transpondedor de biochip, un automóvil que tiene sensores incorporados para alertar al conductor cuando la presión de los neumáticos es baja o cualquier otra cosa natural o artificial. Estos objetos o sensores se le pueden asignar una dirección de protocolo de internet (IP) y puede transferir datos a través de una red.

Un ecosistema de IoT consta de dispositivos inteligentes habilitados para la web que utilizan sistemas integrados, como procesadores, sensores y hardware de comunicación, para recopilar, enviar y actuar sobre los datos que adquieren de sus entornos. Los dispositivos de IoT coparten los datos de los sensores que recopilan al conectarse a una puerta de enlace de IoT u otro dispositivo de borde donde los datos se envían a la nube para analizarlos o analizarlos localmente

Internet de las cosas (IoT) de Azure es un conjunto de servicios en la nube administrados por Microsoft que permiten conectar, supervisar y controlar miles de millones de recursos de IoT. En términos más sencillos, una solución de IoT se compone de uno o varios dispositivos IoT y uno o varios servicios de back-end que se ejecutan en la nube y se comunican entre **sí**.

La palabra telemetría se refiere a sistemas de transmisión de información. Su desarrollo y evolución a través de los años ha facilitado las laboras en muchas áreas de trabajo. Por ejemplo, la aeronáutica, la agricultura, la biología e incluso la medicina moderna hace uso de ella. La telemetría es un sistema automatizado de comunicación (alámbrico o inalámbrico)

que permite recopilar datos en lugares remotos. Se encarga de recoger información, procesarla y transmitirla hasta el lugar donde se monitores el sistema.

El equipo de telemetría se compone de uno o varios sensores que miden ciertas magnitudes físicas o químicas. Luego transforman esta información en señales análogas o inalámbricas para su envió y procesamiento.

Uno de los principales beneficios de estos sistemas es que permiten recoger datos de lugares remotos. Además, con ellos se evita tener que desplazarse al sitio para recoger la información. De igual modo, resultan útiles porque ofrecen información constante y en tiempo real.

Cualquier sistema que utilice telemetría puede ser operado remotamente. Esta función facilita la operación de equipos, además permite corregir errores y enviar información. Funciones que son muy apreciadas en muchas áreas. La telemetría capta información por sensores que la transmiten. Resultado útil para tomar decisiones y prevenir accidentes.

La telemetría funciona como un sistema que capta información, la procesa y la envía a los receptores. Pero desde el momento en que los datos son captados por los sensores, la información para por diferentes etapas antes de llegar al desino definido

Aveva es la única solución adaptable y escalable del mundo para aplicaciones de supervisión, que contextualiza los procesos de operaciones en toda la organización; esto proporciona una base colaborativa basada en estándares que unifica personas, procesos y activos en todas las instalaciones para un mejora operativa continua y soporte de decisiones en tiempo real

La plataforma **Azure** de Microsoft está compuesta por más de 200 productos y servicios en la nube diseñados para brindar soluciones que permitan resolver las dificultades actuales, al crear, ejecutar y administrar aplicaciones en varias nubes, en el entorno local y sus alrededores, con las herramientas y los marcos que seprefiera.[6] Para el desarrollo de este proyecto se hará uso de los siguientes servicios,que se trabajan bajo las suscripciones de la "Empresa".

Azure IoT Hub: es un servicio administrado, hospedado en la nube, que actúa como centro de mensajes para la comunicación entre una aplicación de IoT y los dispositivos conectados. Puede conectar millones de dispositivos y sus soluciones de back-end con confianza y de forma segura. La mayoría de los dispositivos se pueden conectar a un centro de IoT.

Azure Time Series Insights: es un servicio de análisis de IoT de un extremo a otro abierto y escalable que incluye las mejores experiencias de usuario y API enriquecidas para integrar sus eficaces capacidades con su flujo de trabajo o aplicación existentes

Azure Stream Analytics: es un motor de procesamiento de flujos totalmente administrado diseñado para analizar y procesar grandes volúmenes de datos de streaming con latencias de submilisegundos. Se pueden identificar patrones y relaciones en los datos que se originan a partir de una variedad de orígenes de entrada, incluidas las aplicaciones, los dispositivos, los sensores, los flujos de clics y los orígenes de los medios sociales. Estos patrones se pueden usar para desencadenar acciones e iniciar flujos de trabajo, como la creación de alertas, la provisión de información a una herramienta de generación de informes o el almacenamiento de datos transformados para usarlos posteriormente. Stream Analytics también está disponible en el entorno de ejecución de Azure IoT Edge, lo que permite procesar los datos directamente en los dispositivos IoT.

5. Metodología

La metodología se divide en tres etapas para cumplir con el objetivo general.

Etapa 1

Se realiza la comunicación entre los diferentes sistemas que proporcionan la información de los dispositivos IoT instalados en las centrales industriales para el monitoreo de cada uno de los equipos de la empresa.

Esta etapa se divide en los siguientes pasos:

Ingesta de datos:

Se implementará Azure IoT Edge en el entorno local para separar y consolidar los datos operativos en la nube de Azure, y de esta manera lograr que se ejecuten directamente en los dispositivos IoT hacia la nube.

En IoT Edge funcionará como Gateway. Los datos procesados a la nube por medio de dispositivos virtuales definidos en el IoT Hub

Almacenamiento

Para la homologación en estructuras y fuente de datos se propone la creación de una Data Lake (lago de datos) con Azure Data Lake storage, el cual nos va a permitir tener la centralización de todas las fuentes de información, debido a que simplifica la integración de diferentes orígenes, la ejecución de consultas a gran escala con alto rendimiento constante y estable y un gobierno de datos basado en estándares y escalabilidad. Se propone la utilización de una base de datos Azure Cosmos DB, que

permita almacenar los datos más recientes (Ej: de 15 días atrás) y ya transformados para su consumo y análisis

Etapa2

Los datos almacenados deben ser procesados para su correcta lectura y posterior entendimiento, para de este modo lograr identificar las alarmas provenientes de la información recolectada.

Esta etapa se divide en los siguientes pasos:

Procesamiento

Se propone Azure DataBricks para transformar, limpiar y modelar los datos, por su facilidad en la configuración y porque es un servicio de análisis rápido, sencillo y de trabajo interactivo para la colaboración entre científicos de datos, ingenieros de datos y analistar de negocios. Adicionalmente, nos permite trabajar con lenguajes como Python, Scala, R y Sql, con facilidad en el control de versiones de GitHub y Azure DevOps

Análisis de los datos

Con el servicio Azure Stream Analytics se realizarán análisis en tiempo real. Ete servicio permite crear pipelines con datos IoT, usando SQL, además se puede ampliar fácilmente con código personalizado y funcionalidad de aprendizaje automático integrada para escenarios más avanzados.

Etapa3

Para esta etapa se espera lograr validar los datos desde una visualización amigable al usuario donde se pueda identificar por medio de gráficas, el estado de cada uno de los equipos de la empresa

Esta etapa se divide en los pasos:

Report storage

Azure Synapse Analytics es un servicio de análisis ilimitado que reúne la integración de datos, el almacenamiento de datos empresariales y el análisis de macrodatos

Visualización

Usando Power Bi Embebed se proporcionan paneles, análisis e informes. Con visualizaciones optimizadas para el escritorio y los dispositivos móviles, los usuarios

pueden tomar decisiones fácilmente donde quiera que se encuentren cercano al tiempo real. Así mismo, se hará uso de Azure Data Explorer, para preparar dashboard visualizando ciertos KPI para su visualización y análisis.

6. Arquitectura de la solución.

El siguiente esquema representa la arquitectura implementada para el desarrollo del proyecto, ver Figura 1, el flujo de datos ocurre de izquierda a derecha, a continuación, se describen cada uno de los servicios que hacen parte de la solución.

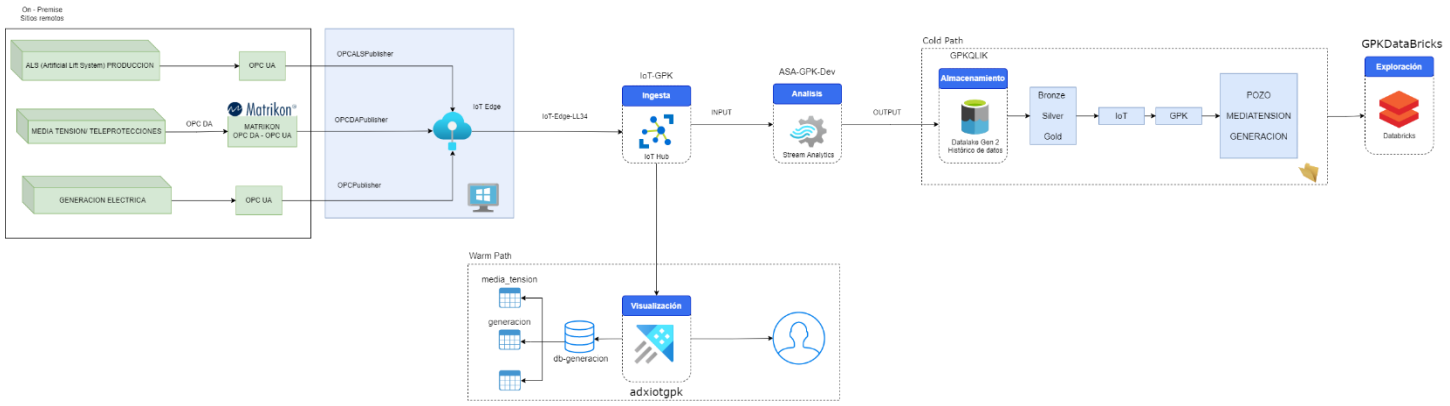

Figura 1 Arquitectura solución IoT

6.1. Inventario de fuentes de datos

Para el piloto se contaron con las siguientes fuentes de datos:

- **SCADAs**

Fuente	**EndPoint**	**Protocolo**
SRVSCADTIGO-05	opc.tcp://xx.57.xxx.13:xxxxx	OPC UA
SRVSCADTIGO1-04	Axon.Builder. Runtime.OPCServer.1	OPC DA
SRVSCADTIGO1-03	opc.tcp://xx.57.xxx.36:xxxxx	OPC UA

6.2. Servicios/Recursos desplegados

Los recursos o servicios desplegados para este proyecto se encuentran en el **grupo recurso de Azure RG-IoT-GPK**. En adelante el usuario tendrá una breve descripción de la función de los recursos.

IoT Edge

Azure IoT Edge acerca la capacidad de proceso al origen de los datos, es decir, a los dispositivos perimetrales. IoT Edge proporciona la capacidad de trasladar las cargas de trabajo de la nube al perímetro. Al hacerlo, IoT Edge supera los problemas de latencia y tiempos de respuesta. Este servicio no se despliega en la suscripción de Azure del usuario sino de manera local en una máquina.

Para monitorear la máquina de IoT Edge que se tiene onpremise se cuenta con modulo adicional en la nube de Azure: *IoTMetricsCollector*, encargado de recolectar la información en tiempo real del rendimiento de dicho componente, los módulos conectados a la nube y la máquina virtual que los soporta. Esta información se encuentra ubicada en la sección de workbooks del servicio de IoT Hub explicado en el siguiente numeral. A continuación, se presentan un ejemplo de las métricas que recolecta:

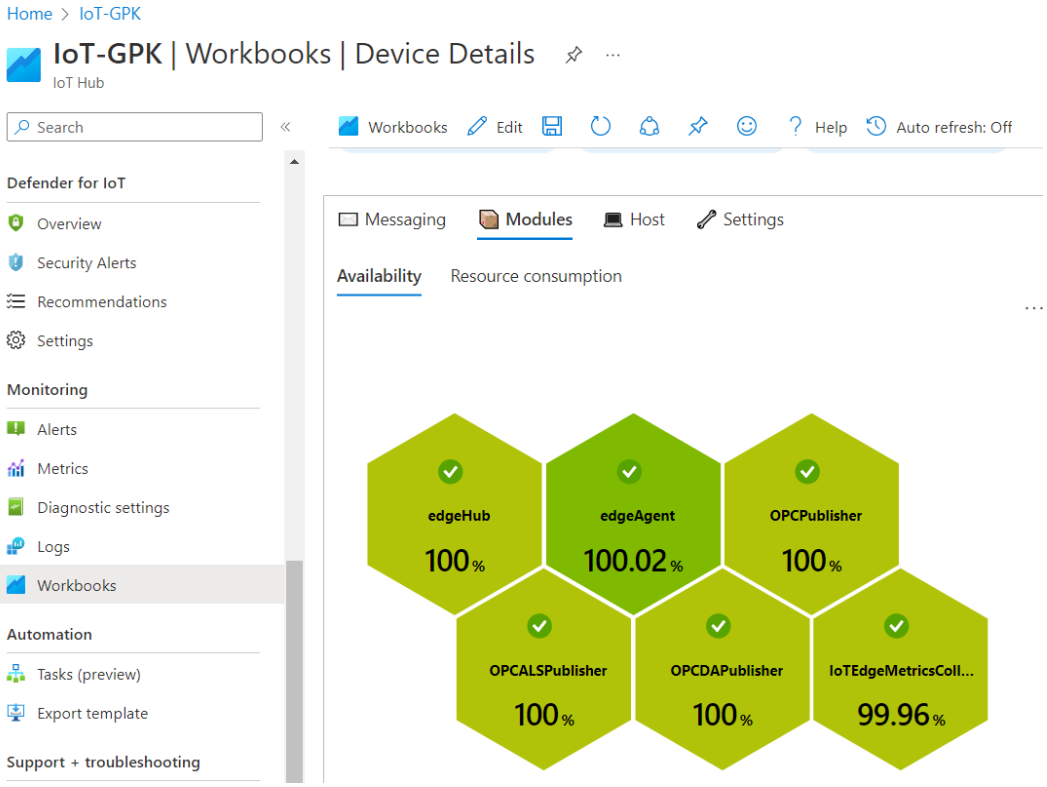

Ilustración 1. Metrics Collector

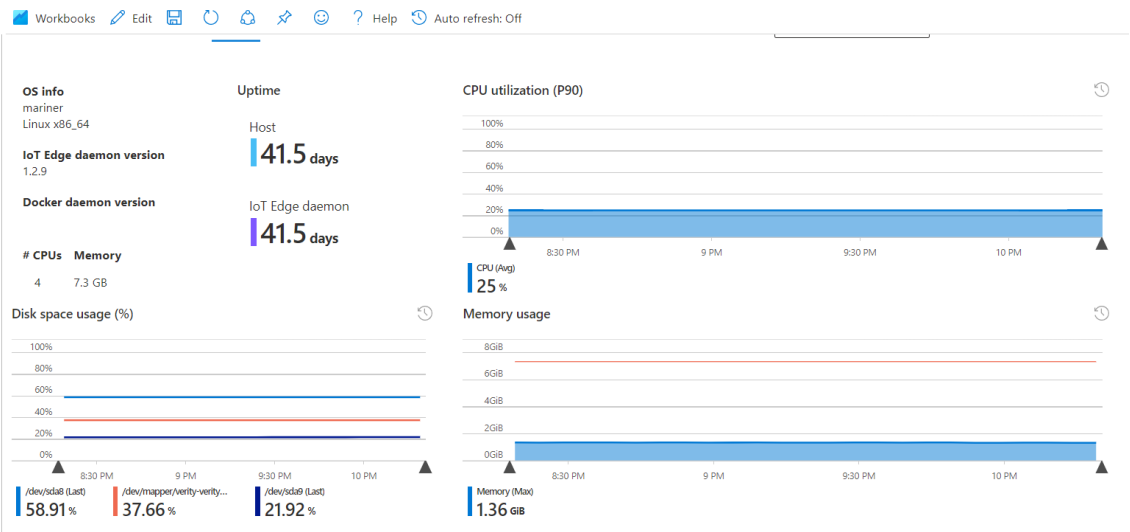

Ilustración 2. Metrics Collector

Posibles fallos IoT Edge: El agente de IoT Edge es uno de los dos módulos que componen el tiempo de ejecución de Azure IoT Edge. Este agente es el responsable de crear los módulos, garantizar que continúen ejecutándose e informar el estado de los módulos a IoT Hub. El agente de IoT Edge envía una respuesta en tiempo de real a IoT Hub. Aquí hay una lista de posibles fallas y lo que significan:

400: la configuración de implementación tiene un formato incorrecto o no es válida, se debe revisar, si la cadena de conexión del IoT Hub es la correcta.

417: el dispositivo no tiene una configuración de implementación establecida, se debe configurar la conexión entre el IoT Hub de la nube y el IoT Edge On-premises.

412: la versión del esquema en la configuración de implementación no es válida, revisar el endpoint de configuración.

406: el dispositivo IoT Edge está desconectado o no envía informes de estado.

500: se produjo un error en el tiempo de ejecución de IoT Edge.

IoT Hub

Es un servicio que permite gestionar, controlar y monitorizar dispositivos o unidades mediante el uso de mensajes. Cada dispositivo puede tener conectados multitud de sensores, los cuales a su vez son objeto de monitorización. Este servicio por lo tanto centraliza toda esa información, y permite su posterior tratamiento o gestión.

El servicio desplegado cuenta con la capacidad de recibir 1200.000 (un millón doscientos mil) mensajes al día, de estos actualmente se usan 137.000 (ciento treinta y siete mil)

mensajes diarios, en promedio. Lo cual indica que se cuenta con capacidad suficiente para agregar, al menos, 25 Módulos o dispositivos adicionales.

Nombre del recurso: IoT-GPK

Azure Data Explorer.

Es una plataforma que facilita el análisis de grandes volúmenes de datos casi en tiempo real. Azure Data Explorer proporciona una solución para la ingesta, consulta, visualización y administración de datos. Mediante el análisis de datos estructurados, semiestructurados y no estructurados en series temporales, y por medio de Machine Learning, Azure Data Explorer facilita la extracción de conclusiones clave, la detección de patrones y tendencias, y la creación de modelos de previsión.

Nombre del recurso: adxiotgpk

Stream Analytics.

Es un motor de procesamiento de flujos de datos totalmente administrado diseñado para analizar y procesar grandes volúmenes de datos de streaming con baja latencia. Se pueden identificar patrones y relaciones en los datos que se originan a partir de una variedad de orígenes de entrada, incluidas las aplicaciones, los dispositivos, los sensores, los flujos de clics y los orígenes de los medios sociales.

Nombre del recurso: ASA-GPK-Dev

Log Analytics Workspace.

Es una herramienta de Azure que recolecta y organiza data de diferentes servicios de monitoreo de Azure, permitiendo realizar análisis de desempeño y monitoreo, generando visualizaciones tipo dashboard con métricas.

Nombre del recurso: LA-IoT

Storage Account.

Una cuenta de Azure Storage contiene todos los objetos de datos de Azure Storage: blobs, archivos, colas, tablas y discos.La cuenta de almacenamiento proporciona un espacio de nombres único para los datos de Azure Storage que es accesible desde cualquier lugar del mundo a través de HTTP o HTTPS.

Nombre del recurso: gpkqlik

6.3. Pre-requisitos máquina virtual IoT Edge.

La máquina en donde se realiza el despliegue del IoT Edge debe cumplir con algunos requerimientos técnicos. Si el sistema operativo es Windows los requerimientos son:

- Maquina (2-4 cpu / 4-8GB ram / 60GB Disco)
- Windows 10 versión 1809 o posterior

- Windows Server 2019 Compilación 17763 o posterior
- Ediciones Pro, Enterprise o Server
- Compatibilidad con la virtualización (Hyper-V).
- Permisos de administrador y conexión RDP.

Si el sistema operativo es Linux los requerimientos son:

- Maquina (2 cpu / 4GB ram / 60GB Disco)
- Linux ubuntu x64 18.04 o superiores y similares.
- Permisos de root y conexión SSH.

Adicionalmente la maquina debe tener habilitados en el Firewall los siguientes puertos:

- HTTP - 443
- AMQP - 5671
- MQTT – 8883

Mediante estos puertos se realizará él envió de la telemetría a los diferentes servicios de la nube, se debe asegurar no solamente que estén habilitados, sino que permita el tráfico cifrado (SSL), esto lo puede consultar con su proveedor de Firewall.

7. Extracción e ingesta de datos.

7.1. Preparación para despliegue de Edge en VM.

Como se mencionó anteriormente el IoT Edge es un servicio de Azure que se instala de manera local, que permite la recolección, centralización y envió de telemetría de diversas fuentes. Para el desarrollo de este proyecto se realizó la instalación del IoT Edge en una máquina virtual con sistema operativo Windows. El proceso de instalación del Edge consiste en instalar un kernel de Linux denominado EFLOW, que dockeriza los recursos del sistema operativo que necesita aprovisionar. Ver Figura 2.

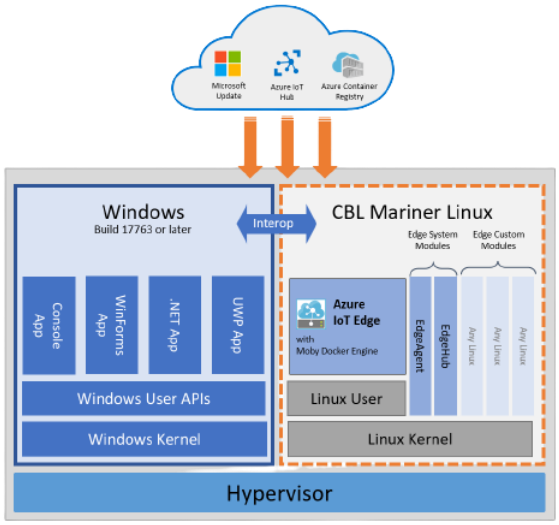

Figura 2. Arquitectura EFLOW.

Para empezar con su instalación se deben cumplir los requisitos mencionados en la sección 3.3. Además, se debe tener el servicio de IoT Hub desplegado en Azure.

7.2. Registro del Dispositivo en IoT Hub

El proceso de despliegue de Azure IoT Edge para Linux en Windows inicia con la creación del dispositivo en el IoT Hub. Para esto se ingresa al servicio de IoT Hub, en el panel a la derecha se selecciona IoT Edge, en la ventana se selecciona Add IoT Edge Device, en Device ID se ingresa el nombre, se mantiene Symetrics Keys y se guarda.

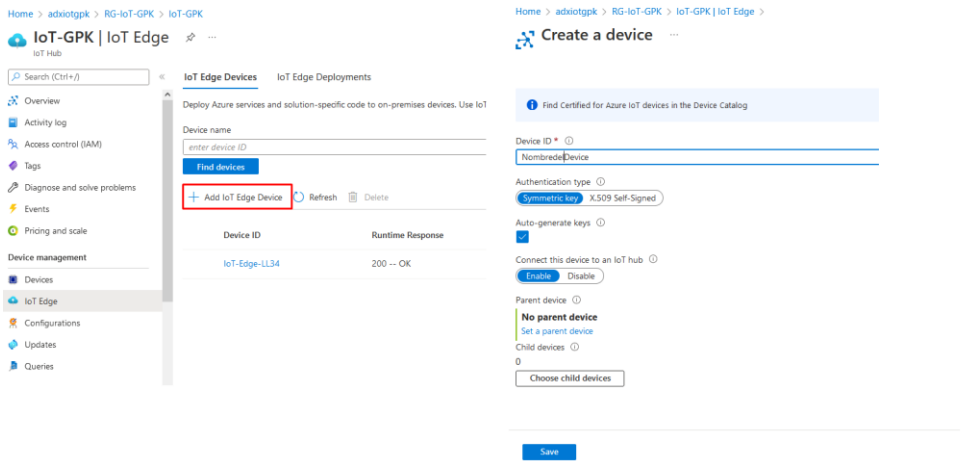

Figura 3. Despliegue IoT Edge.

Una vez desplegado el Device se puede observar un overview con la información de este.

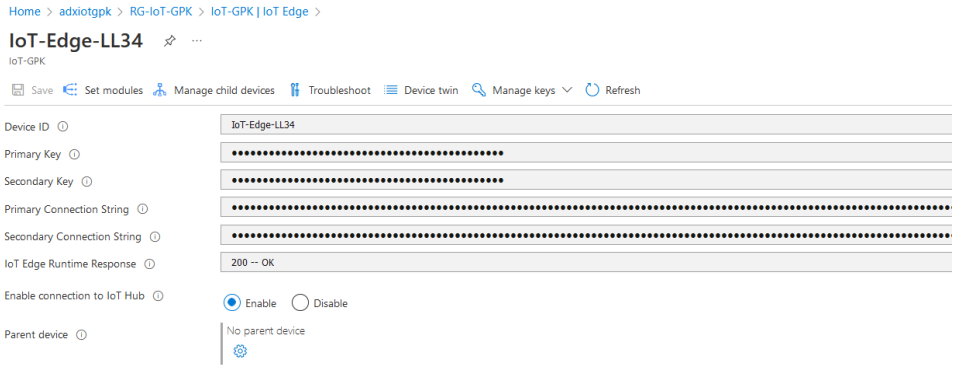

Figura 4. OverView IoT Edge.

A continuación, se realiza la instalación del Edge en la VM.

7.3. Instalación del Edge para Linux en VM Windows

Para la instalación del Edge se verifica que las políticas de ejecución en la máquina virtual estén habilitadas, ejecutando en una Power Shell en modo administrador el comando **Get-ExecutionPolicy -List**. Si el parámetro en LocalMachine no está firmado, se ejecuta el siguiente comando **Set-ExecutionPolicy -ExecutionPolicy AllSigned -Force**.

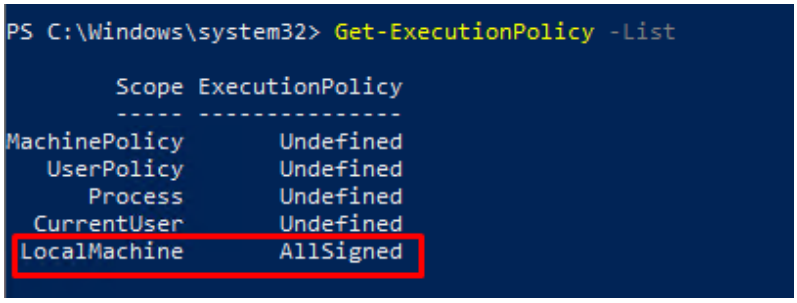

Figura 5. Verificaciones políticas IoT Edge.

A continuación, en la misma Power Shell en modo administrador se ejecutan los siguientes comandos, que se encarga de descargar el IoT Edge Linux for Windows.

$msiPath = $([io.Path]::Combine($env:TEMP, 'AzureIoTEdge.msi'))
$ProgressPreference = 'SilentlyContinue'
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
Invoke-WebRequest "https://aka.ms/AzEFLOWMSI-CR-X64" -OutFile $msiPath

Una vez descargado se procede a realizar la instalación mediante el siguiente comando.

Start-Process -Wait msiexec -ArgumentList "/i","$([io.Path]::Combine($env:TEMP, 'AzureIoTEdge.msi'))","/qn"

Se continua con el despliegue del IoT Edge y el aprovisionamiento de la máquina virtual Eflow mediante el siguiente comando.

Deploy-Eflow

En el despliegue aparecen parámetros a los que se les indica 'Y' para aceptar los términos y licencias, y 'O' o 'R' para activar o desactivar los datos de diagnóstico opcionales, según sus preferencias. En el despliegue de la maquina se puede observar en la consola como se le asignan los diferentes recursos, como dirección IP y MAC. Si el despliegue es exitoso aparece un mensaje con Deyployment Succesful.

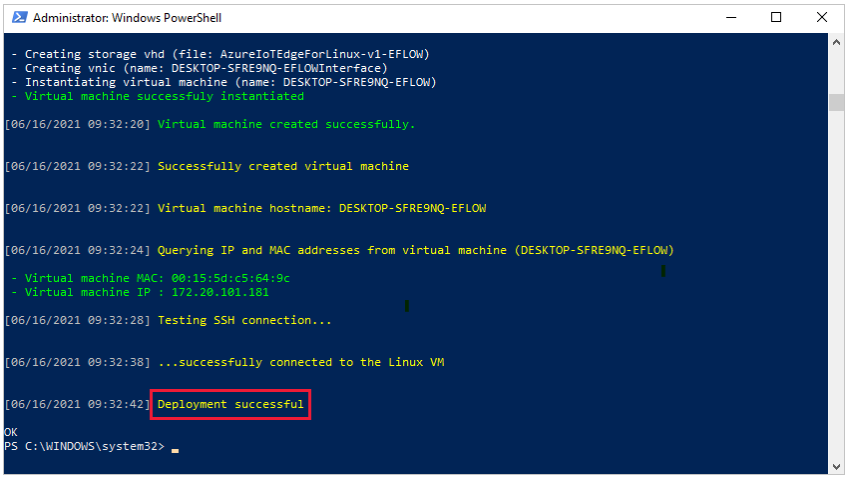

Figura 6. Finalización instalación IoT Edge.

Una vez desplegada la maquina se procede a conectarla con el servicio de IoT Hub en la nube mediante el siguiente comando, en el que se deben agregar las llaves que se encuentran el overview, ver figura 4.

Provision-EflowVm -provisioningType ManualConnectionString -devConnString "PASTE_DEVICE_CONNECTION_STRING_HERE"

En este punto se encuentra desplegado el IoT Edge y conectado al IoT Hub en la nube. Para acceder a la máquina del Edge (Eflow) se realiza mediante el comando:

Connect-EflowVm

Una vez ingresado en la maquina el usuario se encuentra en un entorno Linux, se pueden usar los siguientes comandos para hacer validaciones.

- sudo iotedge list: Lista los dispositivos desplegados.
- sudo iotedge system: Verifica los logs del sistema.
- sudo iotedge check: Verifica la configuración de la máquina.

7.4. Parametrización para envío de telemetría entre IoT Edge y IoT Hub.

Para este punto el IoT Edge se encuentra conectado como dispositivo en el IoT Hub, se debe configurar el archivo publishnode.json, en el que se indica los tags o variables que se van cargar y el endpoint del servidor de donde se toma la telemetría.

Se ingresa a la maquina EFLOW y se crea el archivo mediante el siguiente comando.

```
echo "" >
cat << EOF > publishednodes.json
[{hola mundo}]
EOF
```

Mediante el comando **ls**, se verifica que el archivo se creó.

Nuevamente se modifica el archivo, se borra la información que contenga y se ingresa el siguiente contenido.

```
[
{
"EndpointUrl": <"Punta OPC Server">
"UseSecurity": false,
"OpcNodes": [
{
"Id": "ns=3;i=1001"
},
{
"Id": "ns=3;i=1006"
}
]
}
]
```

El contenido del archivo se clasifica así:

- EndpointUrl: URL del servidor OPC de donde se extraen los datos.
- UseSecurity: Indica si requiere parámetros de inicio de sesión para ingresar al servidor del Endpoint. Puede ser false o true, en caso de ser true se incluyen los siguientes parámetros, después del UseSecurity.

  “OpcAuthenticationMode”: “UsernamePassword”,

  “OpcAuthenticationUsername”: “USER”,

  “OpcAuthenticationPassword”: “Password”

- OpcNodes: Se compone del name space(ns) y del indicador del tag, estos valores son propios del sistema de donde se extraen los datos.

Una vez configurado el archivo, se realiza la configuración en Azure para la recepción de la telemetría.

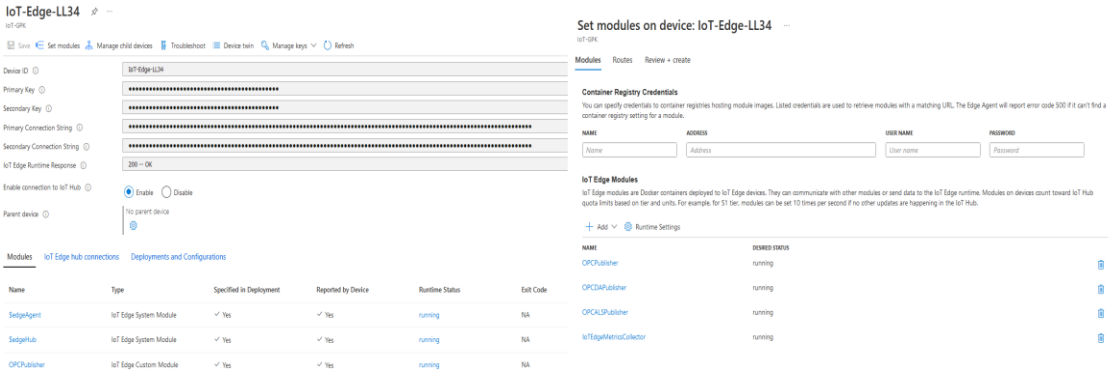

Figura 7. Configuración en Azure del Device.

Se ingresa a la sección de IoT Edge, desde el IoT Hub y se selecciona el módulo a configurar, a continuación, se selecciona la opción de Set modules en donde se visualizan los módulos configurados, seleccionamos el módulo correspondiente.

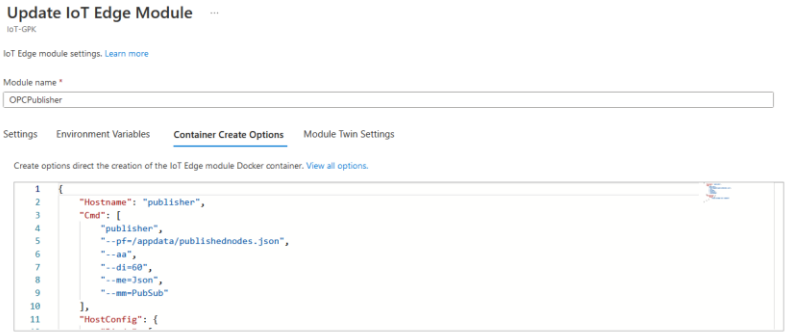

Figura 8. Configuración Azure Device Container.

Al ingresar al módulo se observan diferentes configuraciones, se selecciona Container Create Options y se le indican los siguientes parámetros.

En este punto ya se encuentra configurado el módulo IoT Hub y IoT Edge para el envío y recepción de telemetría.

A continuación, se explica cada uno de los módulos creados en el IoT Hub y su configuración.

Name	Type	Specified in Deployment	Reported by Device	Runtime Status	Exit Code
$edgeAgent	IoT Edge System Module	✓ Yes	✓ Yes	running	NA
$edgeHub	IoT Edge System Module	✓ Yes	✓ Yes	running	NA
OPCPublisher	IoT Edge Custom Module	✓ Yes	✓ Yes	running	NA
OPCDAPublisher	IoT Edge Custom Module	✓ Yes	✓ Yes	running	NA
OPCALSPublisher	IoT Edge Custom Module	✓ Yes	✓ Yes	running	NA

Figura 9. Módulos del IoT Hub.

OPCPublisher

Este módulo corresponde al SCADA cimplicity o de generación eléctrica, la comunicación entre el servidor y el módulo Edge se realiza mediante protocolo OPC UA, en el IoT Hub se identifica como OPCPublisher, el esquema del envío de la información se observa en la Figura 10.

Figura 10. Configuración archivo JSON OPC Publisher.

OPCDAPublisher

Este módulo corresponde al SCADA de media tensión / teleprotecciones. La comunicación entre el servidor y el módulo Edge se realiza mediante protocolo OPC UA, sin embargo, la salida directa del SCADA es OPC DA por lo que se implementó un módulo traductor de OPC DA a OPC UA directamente en el servidor SCADA, asegurando la comunicación con el módulo. El esquema para el envío de la telemetría se observa en la imagen.

Figura 11. Configuración archivo JSON OPCDAPublisher.

OPCALSPublisher

Este módulo corresponde al SCADA ALS o de producción. La comunicación entre el servidor y el módulo Edge se realiza mediante protocolo OPC UA, en el IoT Hub se identifica como OPCALSPublisher, el esquema del envío de la información se observa en la imagen.

```
[
{
"EndpointUrl": "opc.tcp://10.57.182.36:48031",
"UseSecurity": false,
"OpcNodes": [
{"Id": "ns=3;s=1978:Jacana_37.Freq_Min.RawMax"},
{"Id": "ns=3;s=1978:Jacana_37.Freq_Min.RawMin"},
{"Id": "ns=3;s=1978:Jacana_37.Pip_Low_Setting"},
{"Id": "ns=3;s=1978:Jacana_37.Presion_Intake"}
]
}
]
```


Figura 12. Configuración archivo JSON OPCALSPublisher.

8. Transformación y almacenamiento de datos.

Una vez los datos se extraen e ingesta por medio del IoT Hub, se procede a organizar la telemetría y el formato json que será almacenado en el datalake. Para esto se hace uso del servicio de Azure Stream Analytics, este recurso será el encargado de leer la telemetría, analizarla, modificar el formato/contenido y enviarla hacia su destino. Para logarlo se deben definir los elementos de Entrada y Salida en el ASA.

8.1. Definición de entradas y salidas.

Para crear una entrada en el Stream Analytics (Input Stream), se selecciona en el panel Inputs, y se adiciona una entrada del tipo stream, en este caso se le indica el nombre del IoT Hub y se comparten las credenciales de conexión Hub.

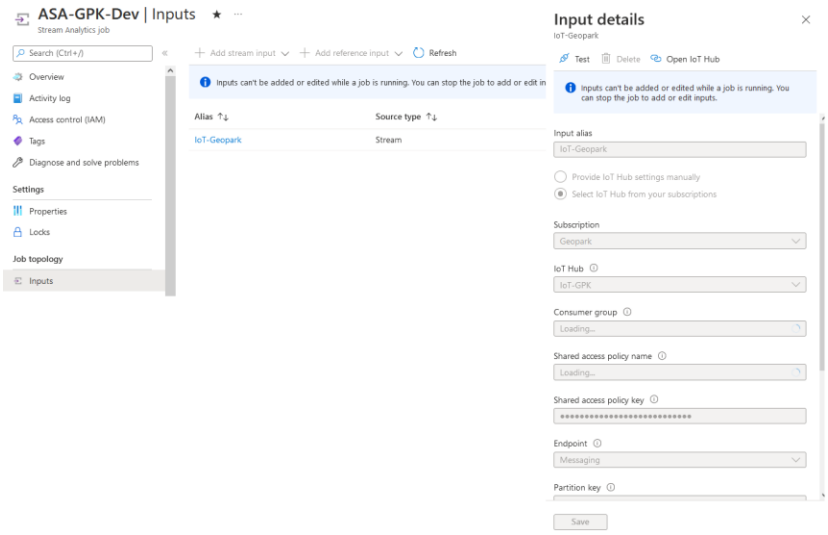

Figura 13. Creación Input Azure Stream Analytics.

Una vez definida la entrada, se configura la salida seleccionando Outputs desde el panel, se indica el tipo de almacenamiento en este caso Blob storage, el tipo de autenticación

Connection String, esta requiere la llave de conexión del storage account y la ruta de almacenamiento en el Storage, que en este caso es IoT/GPK/{Ruta}/{date}. Los parámetros Ruta y date son dinámicos y varían según la configuración del Query que más adelante se explicara.

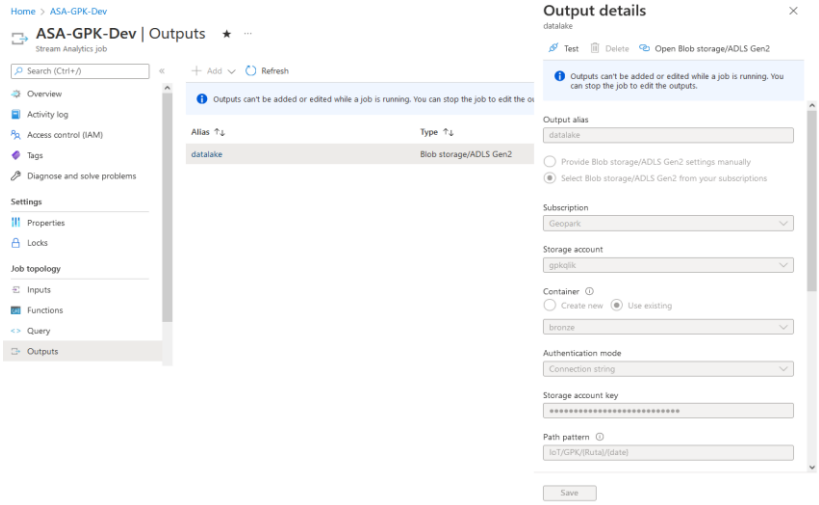

Figura 14. Creación Input Azure Stream Analytics.

8.2. Creación Query para almacenamiento.

Una vez configurado los elementos de entrada (Inputs) y salida (Outputs), se crea el Query que permitirá consultar los datos que llegan en tiempo real desde el IoT Hub. Se selecciona la opción de Query en el panel, se observa los elementos que se han definido como Inputs y Outputs, además de algunas funciones que se crearon para la implementación de la consulta.

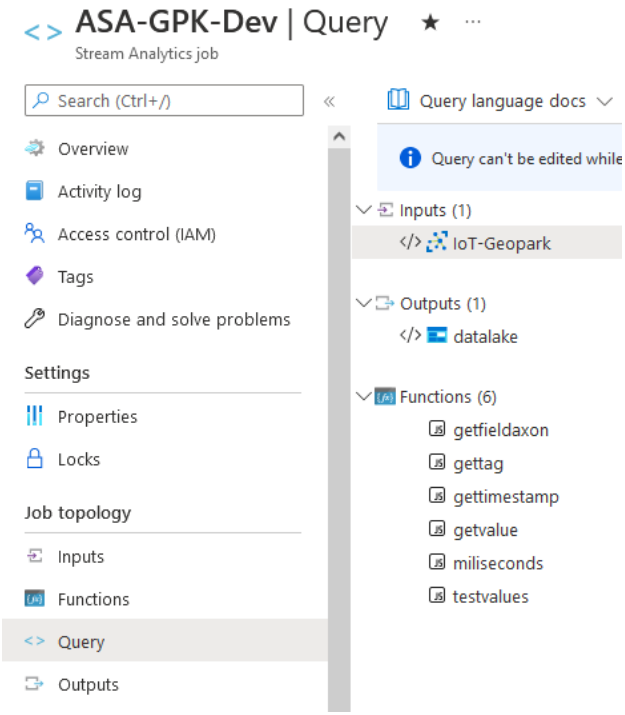

Figura 15. Entorno creación Query.

Para la definición de la consulta, primero se analiza la estructura con la que llegan los datos al stream analytics.


```
{
  "MessageId": "846678",
  "MessageType": "ua-data",
  "PublisherId": "Standalone_IoT-Edge-LL34_OPCPublisher",
  "DataSetClassId": null,
  "Messages": [
    {
      "DataSetWriterId": "opc.tcp://10.57.172.13:51800_af7367c42d8611fadc267f05230ffa1b42a04135",
      "MetaDataVersion": {
        "MajorVersion": 1,
        "MinorVersion": 0
      },
      "Payload": {
        "http://ge.com/ua/CIMPLICITY/TIGANA-JACANA#s=JAC_POZOS_AGC_RED1_I1.Value": {
          "Value": 325,
          "SourceTimestamp": "2022-09-06T19:56:43.0700241Z"
        },
        "http://ge.com/ua/CIMPLICITY/TIGANA-JACANA#s=JAC_POZOS_AGC_RED1_BB_HZ.Value": {
          "Value": 60,
          "SourceTimestamp": "2022-09-06T19:56:42.9730237Z"
        },
        "http://ge.com/ua/CIMPLICITY/TIGANA-JACANA#s=JAC_POZOS_AGC_RED1_U_BB_L2L3.Value": {
          "Value": 486,
          "SourceTimestamp": "2022-09-06T19:56:42.9730237Z"
        },
        "http://ge.com/ua/CIMPLICITY/TIGANA-JACANA#s=JAC_B_AGC_ACO2_BB_A_HZ.Value": {
          "Value": 59.99,
          "SourceTimestamp": "2022-09-06T19:56:42.7230278Z"
```


Figura 16. Estructura del mensaje.

En la Figura 16 se observa que el mensaje recibido en una estructura tipo JSON y que contiene los campos:

- MessageId: Consecutivo que identifica el número del mensaje enviado.
- MessageType: Indica el tipo de mensaje recibido.

- PublisherId: Indica la fuente de información, en este caso se tienen tres fuentes de información.
    - OPCPublisher: SCADA Generación.
    - OPCDAPublisher: SCADA Media tensión.
    - OPCALSPublisher: SCADA Pozo.
- DataSetClassId: No aplica
- Messages: Contiene la meta data enviada desde el IoT Hub, internamente posee otra estructura.

Para empezar la construcción del Query se hace referencia a los datos de entrada mediante el comando FROM apuntando a la entrada definida previamente como IoT, indicándo que será identificada como "i". Los campos que se generan con la consulta son:

- NodeID: Nombre del tag o variable.
- Value: Medición numérica de la variable.
- TimeStamp: Medida de tiempo con la que se captura el dato.
- Time: Tiempo en milisegundos.
- Field: Campo físico donde ocurren los eventos.
- Scada: Sistema SCADA de donde se toman los datos.
- Ruta: Esquema de almacenamiento.

El contenido del Query principal se define en el sub-query telemetry, que posteriormente se invoca.

```
WITH telemetry AS (

SELECT
    udf.gettag(i.Messages) as NodeID,
    udf.getvalue(udf.testvalues(i.Messages)) as Value,
    udf.gettimestamp(udf.testvalues(i.Messages)) as TimeStamp,
    udf.miliseconds(udf.gettimestamp(udf.testvalues(i.Messages))) as Time,
    CASE WHEN udf.gettag(i.Messages) LIKE '%TIG%' OR
              udf.gettag(i.Messages) LIKE '%Tig%' THEN 'TIGANA'
    WHEN udf.gettag(i.Messages) LIKE '%JAC%' OR
        udf.gettag(i.Messages) LIKE '%Jac%'THEN 'JACANA'
    WHEN udf.getfieldaxon(i.Messages) LIKE 'Axon' THEN 'AXON'
    ELSE '' END as Field,
    CASE WHEN i.PublisherId LIKE '%OPCPublisher%' THEN 'GENERACION'
    WHEN i.PublisherId  LIKE '%OPCDAPublisher%' THEN 'MEDIATENSION'
    WHEN i.PublisherId  LIKE '%OPCALSPublisher%' THEN 'POZO'
    ELSE '' END as Scada
FROM
    [IoT-Geopark] i
)
```

Figura 17. Sub-query Telemetry.

NodeID

Este campo indica el nombre del tag del cual se esta transmitiendo información, puede hacer referencia a la variable de un equipo o proceso, este dato se encuentra en el campo Messages de la estructura principal. En la Figura 18 se observa el campo identificado como NodeID para el SCADA CIMPLICITY.

```
[
  {

"DataSetWriterId":"opc.tcp://10.57.172.13:51800_2bf1bcd9971ef73a9e446a61fdc26cd176ea677f",
"MetaDataVersion":
{"MajorVersion":1,
"MinorVersion":0
},
"Payload":{
  "http://ge.com/ua/CIMPLICITY/TIGANA-JACANA#s=JAC_B_AGC_ACO2_BB_A_UL1L2.Value":{
    "Value":3325,
    "SourceTimestamp":"2022-06-23T18:53:29.5951541Z"
    },
    "http://ge.com/ua/CIMPLICITY/TIGANA-JACANA#s=JAC_B_AGC_ACO2_BB_A_UL1L3.Value":{
      "Value":3312,
      "SourceTimestamp":"2022-06-23T18:53:29.5951541Z"
      },
    }
  }
]
```


Figura 18. Telemetría SCADA CIMPLICITY.

Como se observa en la Figura 17, el primer paso para extraer el campo es hacer referencia al campo Messages que lo contiene por medio i.Messages y este campo se pasa como parámetro a la función udf.gettag, que se observa en la Figura 19.

```
function main(nodeid) {
    const str = JSON.stringify(nodeid);

    if (str.includes("CIMPLICITY")) {
        var valores = str.split('=')
        var tag = valores[1].split('.')
        var name = tag[0]
    }

     if (str.includes("Axon")) {
        var valores = str.split('%')
        var tag = valores[1].split('"')
        var name = tag[0].replace("3a","")
    }

    if (str.includes("aveva")) {
        var valores = str.split('%')
        var tag = valores[1].split('"')
        var name = tag[0].replace("3a","")
    }


     return name
}
```

Figura 19. Función gettag.

La función udf.gettag convierte en string el parámetro que recibe como entrada e identifica cual es el tipo de SCADA, según esto ejecuta una función única para cada uno, ya que la extracción del NodeID es diferente para cada caso, según el identificado ejecuta operaciones sobre el string y retorna el valor.

Value

El siguiente campo por extraer es value, para este nuevamente se parte del contenido de Messages proporcionado por “i” que es la fuente de datos, este campo se pasa como parámetro a la función udf.testvalues y esta a su vez a la función udf.getvalue.

En la Figura 20 se observa que el campo value se encuentra dentro del parámetro Payload, para acceder al dato se suministra el Messages como variable de entrada en la función y se nombra como nodeid, indicando que se quiere acceder a la posición cero del payload.

```
function main(nodeid) {
    var str = nodeid[0].Payload
    //console.log(str)
    return str
}
```

Figura 20. Función testvalues.

El valor que retorna esta función se pasa a la función udf.getvalue, que se encarga de convertir en string, en contenido y navegar la variable str hasta obtener el valor y retornarlo.

```
function main(nodeid) {
    const str = JSON.stringify(nodeid);
    var valores = str.split(':')
    //console.log(str)
    var aux = valores[3].split(',')
    return aux[0]
}
```

Figura 21. Función getvalue.

Timestap

Para obtener la estampa de tiempo, se hace uso de la función udf.testvalue nuevamente para obtener el contenido del mensaje en el Payload y se pasa como parámetro a la función udf.gettimestamp.

```
function main(nodeid) {
    const str = JSON.stringify(nodeid);
    var valores = str.split(',')
    //console.log(str)
    var aux = valores[1].split('"')
    return aux[3]
}
```

Figura 22. Función gettimestamp.

Esta función convierte en string el valor que recibe de entrada y se encarga de obtener el valor de tiempo, navegando por la cadena. Finalmente retorna el valor.

Time

El campo time es una medida de tiempo en milisegundos que indica, el tiempo que ha pasado desde el 00:00:00 del 1 de enero de 1970, esta es una medida de referencia que se usa en tecnologías IoT. Para su obtención se hace uso de la función udf.miliseconds que retorna el valor calculado.

Este campo se obtiene pasando a la función udf.milisenconds las funciones que intervienen para obtener el campo TimeStamp, ya que este campo obtiene fecha y hora del envio de la telemetria y a partir de este se realiza el cálculo de los milisegundos, mediante la función de java script parse().

```
function main(actualtiempo){

    const dt = Date.parse(actualtiempo);
    return String(dt);

}
```

Figura 23. Función miliseconds.

Field

Este campo indica el campo físico de donde proviene la información para este caso puede ser Tigana o Jacana.

```
CASE WHEN udf.gettag(i.Messages) LIKE '%TIG%' OR
          udf.gettag(i.Messages) LIKE '%Tig%' THEN 'TIGANA'
WHEN udf.gettag(i.Messages) LIKE '%JAC%' OR
    udf.gettag(i.Messages) LIKE '%Jac%'THEN 'JACANA'
WHEN udf.getfieldaxon(i.Messages) LIKE 'Axon' THEN 'AXON'
ELSE '' END as Field,
```

Figura 24. Definición del campo field.

Para obtenerlo se hace uso de la función udf.gettag que retorna el contenido del Payload en formato string, de este contenido se identifican mediante unos CASE WHEN los patrones que hacen referencia al campo Tigana y al campo Jacana, así mismo cuando no se encuentre un patrón retorna como vacío el campo.

SCADA

Mediante este campo se indica el sistema SCADA de cual proviene la telemetría, para esto se toma el campo PublisherId que contiene el patrón para identificar cada sistema según el caso. Por ejemplo, el SCADA de media tensión se identifica mediante el patrón OPCDAPublisher.

```
CASE WHEN i.PublisherId LIKE '%OPCPublisher%' THEN 'GENERACION'
WHEN i.PublisherId  LIKE '%OPCDAPublisher%' THEN 'MEDIATENSION'
WHEN i.PublisherId  LIKE '%OPCALSPublisher%' THEN 'POZO'
ELSE '' END as Scada
```

Figura 25. Definición del campo Scada.

Ruta

Mediante este campo se indica el esquema de almacenamiento de los datos, concatenando los campos de SCADA y FIELD.

En la figura 26 se observa los campos obtenidos en la consulta.

```
SELECT *, CONCAT (Scada,'/',Field) as Ruta
INTO
    [datalake]
FROM [telemetry]
```

Figura 26. Definición del campo Ruta y almacenamiento.

Para el almacenamiento de los datos se indica mediante el comando INTO el destino ver Figura 26, en este caso se identifica como datalake, que hace referencia a la salida configurada previamente.

En la Figura 27 se observa la estructura y los campos de los datos, que son almacenados en formato JSON.

Figura 27. Datos para almacenamiento.

8.3. Almacenamiento en el data lake.

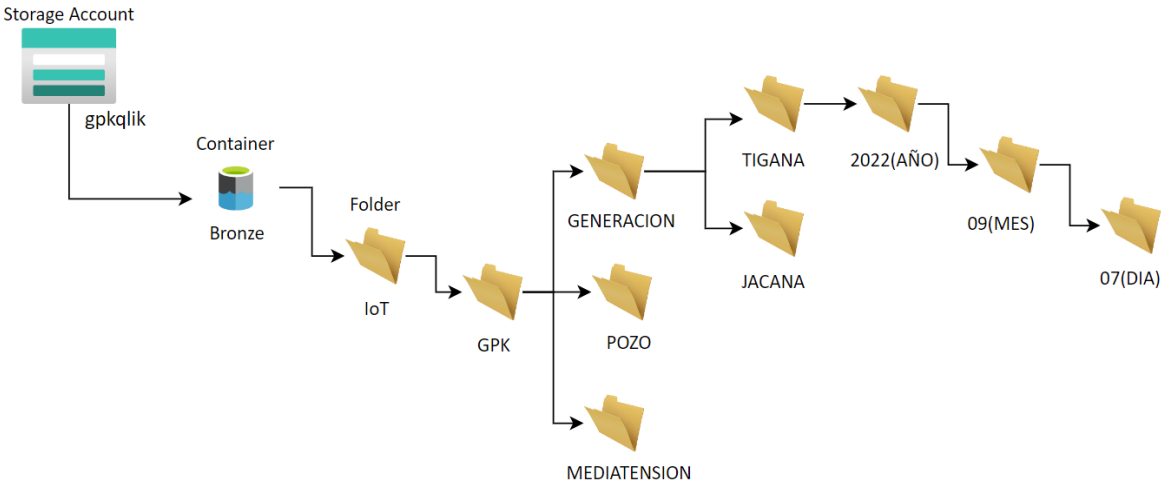

Figura 28. Estructura de almacenamiento Data Lake.

La estructura para el almacenamiento de los datos en el Data lake, se puede observar el Fig. 28. El storage Account seleccionado es el gpkqlik, este data lake contiene diferente container, en este caso se selecciona el Bronze, este parámetro se define en la creación del Output del Stream Analytics ver Figura 13. Dentro del container se encuentran los folders de almacenamiento, de igual forma este campo se define mediante el output del Stream, con el path definido es el siguiente IoT/GPK/{Ruta}/{date}.

El primer folder al que se accede es de IoT, seguido se ubica el folder GPK, dentro de este folder se encuentran 3 subcarpetas:

- GENERACION
- POZO
- MEDIATENSION

El almacenamiento de los datos en cada una dependerá del parámetro dinámico Ruta declarado en el path, ejemplo: "GENERACION/JACANA". Este parámetro se construye en el Query que utiliza el Stream Analytics, dependiendo de que SCADA reciba los datos así mismo apuntara a un folder. En la Figura 28, se utiliza el SCADA de Generación. Una vez identificado el folder del SCADA al que apuntan los datos, se define el campo al que pertenecen los datos, puede ser TIGANA o JACANA.

El siguiente parámetro que se define el path es date, que indica la fecha con la que se almacenan los datos, el formato de almacenamiento es YYYY/MM/DD, en donde cada campo representa un folder de almacenamiento. Por ejemplo, en la Figura 28 se observa la carpeta 2022 correspondiente al año, la carpeta 09 que hace referencia al mes y la carpeta 07 correspondiente al día. Finalmente, en esta ultima la carpeta correspondiente al día se encuentran los archivos en formato JSON que contiene la telemetría almacenada.

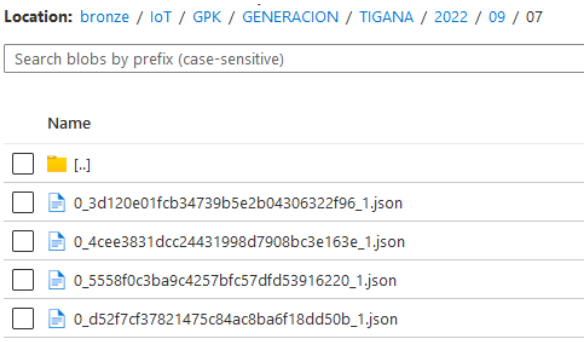

Figura 29. Datos almacenados en Datalake.

9. Visualización.

Un componente fundamental de la implementación es poder contar con una herramienta que permita consultar, analizar y visualizar la telemetría obtenida de los diferentes sistemas. El servicio desplegado para este propósito fue el ***Azure Data Explorer***, un servicio con una base de datos optimizada especialmente para series de tiempo. A continuación, se describe todo el proceso desde la creación de la base de datos, el almacenamiento de la telemetría y la creación de las diferentes visualizaciones.

9.1. Creación de bases de datos

El primer paso consiste en crear una base de datos que almacene los datos en un periodo de 365 días.

En la barra lateral, en el apartado ***Data*** se ingresa a ***Databases***, una vez allí se procede a crear la nueva base de datos con el botón ***+ Add Database***. En la barra lateral derecha emergente, se procede a definir el nombre de la base de datos y el periodo de retención. En la Figura 30 se puede apreciar la sección del portal utilizada, el nombre de la base de datos creada y el periodo de retención.

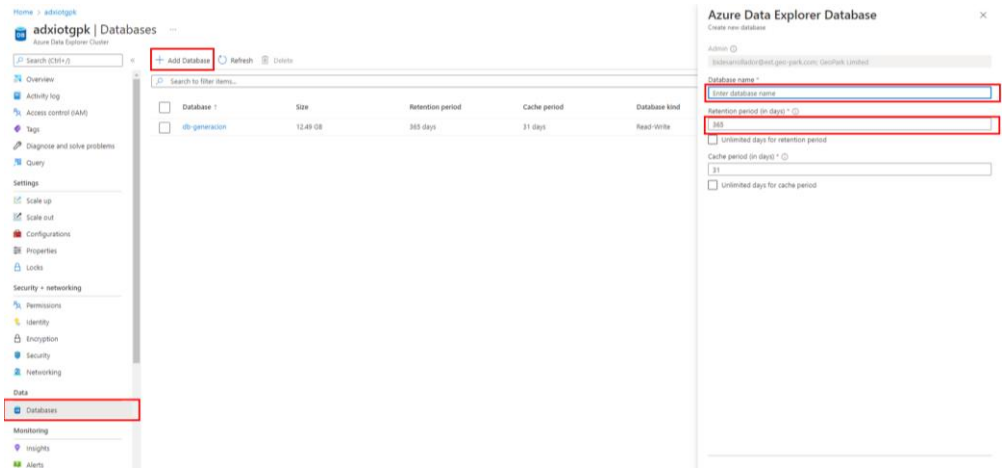

Figura 30. Creación de base de datos.

9.2. Conexión con IoT Hub

Para realizar la conexión con el ***IoT Hub***, se selecciona la base de datos creada y se procede a añadir la conexión como se muestra en la Figura 31, en "***Data connections*** > ***+Add Data Connections*** > ***IoT Hub***".

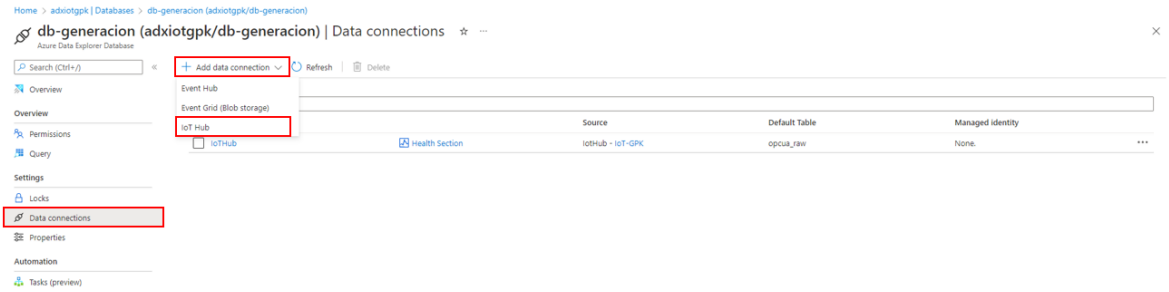

Figura 31. Apartado para conexión con IoT Hub.

Como se observa en la Figura 32, se continúa editando los diferentes campos de la conexión entre los cuales se encuentran el nombre de la conexión, selección de la subscripción, selección del IoT Hub y grupo de consumidores.

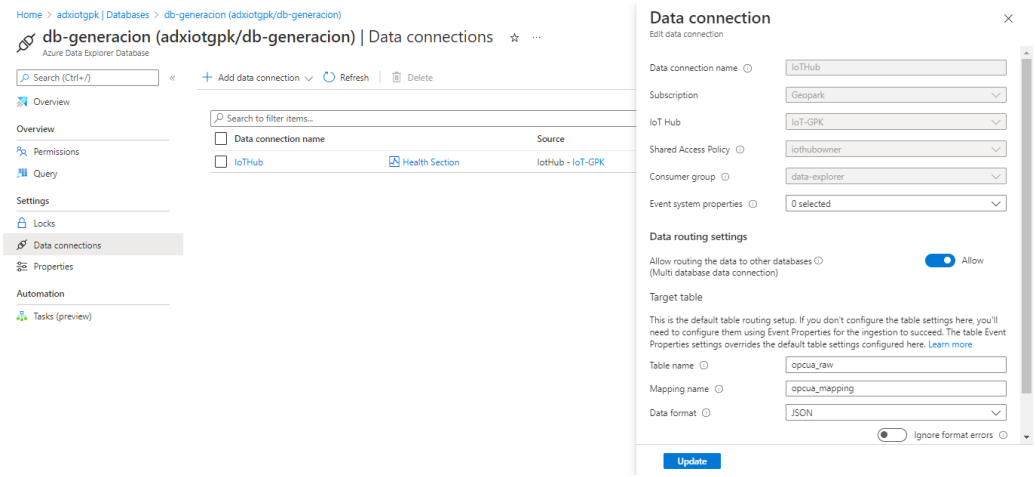

Figura 32. Edición de Conexión con IoT Hub.

Antes de continuar con las últimas tres configuraciones correspondientes a la configuración del enrutamiento de los datos (***Data routing settings***), se crea una tabla nueva en la base de datos que almacenara las tramas en formato JSON obtenidas del IoT Hub. Para esto ***Azure Data Explorer implementa*** el lenguaje ***KQL***. En la pestaña ***Query*** ubicada en la barra lateral,

se procede a crear la tabla ***opc_raw*** como se muestra en la Figura 33 con el comando ***KQL*** de control:

```
.create table opcua_raw(payload: dynamic).
```

Este define una tabla con una única columna tipo ***dynamic***, siendo este tipo de dato el adecuado para albergar las tramas en formato ***JSON***.

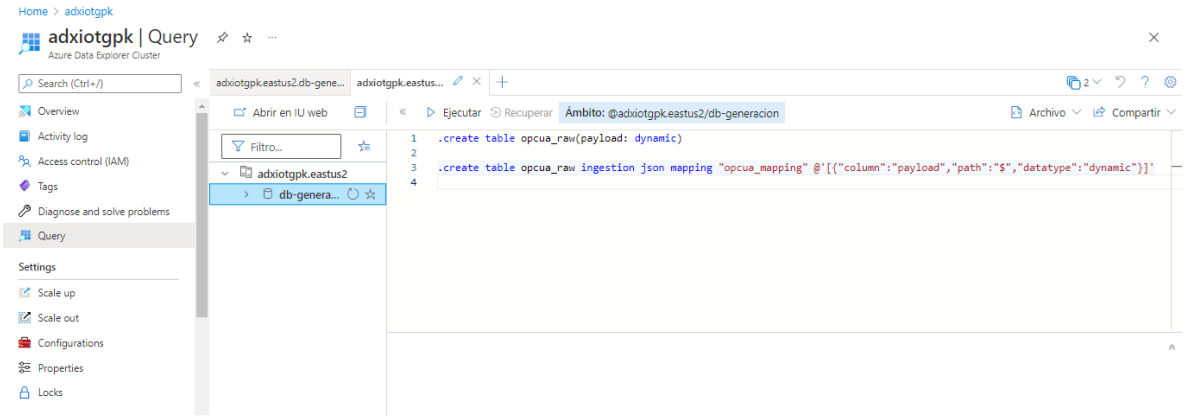

Figura 33. Pestaña de Consultas.

Luego se realiza el mapeo de los datos haciendo uso de del comando:

```
.create table opcua_raw ingestion json mapping “opcua_mapping” @’[{”column”:”payload”,”path”:”$”,”datatype”:”dynamic”}]’
```

Este permite tomar los datos provenientes del IoT Hub y almacenarlos en la tabla ***opc_raw***.

En este punto se continua con la edición de la conexión con el IoT Hub, procediendo a llenar los campos ***Table name***, ***Mapping Name*** y ***Data Format*** como se muestra en la Figura 32 y presionando ***Update***.

9.3. Creación de Tablas Intermedias

Para procesar con facilidad los datos almacenados en la tabla principal, se requiere la creación de tablas intermedias que permitan la descomposición de los paquetes en formato JSON. Para llevar los datos desde la tabla principal a otras tablas se requieren tres pasos:

1. Crear la tabla de destino
2. Crear una función almacenada
3. Actualizar las políticas de ingesta de la nueva tabla con la función creada

A continuación, se describen las diferentes tablas intermedias requeridas para la expansión de los mensajes JSON y las tablas finales asociadas a las necesidades del negocio.

Con el fin de separar los datos almacenados en la tabla principal en múltiples columnas, se crea una tabla intermedia en la cual se realiza la primera expansión con el comando:

```
.create table opcua_intermediate(DataSetWriterID: string, payload: dynamic)
```

Se procede a crear la función que realizara la expansión:

```
.create-or-alter function OPCUARawExpand() {
    opcua_raw
    |mv-expand records = payload.Messages
    |project
        DataSetWriterID = tostring(records[”DataSetWriterId”]),
        Payload = todynamic(records[”Payload”])
}
```

Se realiza la actualización de la política de ingesta de la tabla creada para que los datos recibidos en ***opc_raw*** pasen expandidos a ***ocupa_intermediate*** en tiempo real con el siguiente comando:

```
.alter table opcua_intermediate policy update @’[{”Source”: “opcua_raw”, “Query”: “OPCUARawExpand()”, “IsEnabled”: “True”}]
```

Finalmente se realiza la creación de la tabla final en donde se contará con cada una de las variables extendidas junto a su correspondiente valor y estampilla de tiempo:

```
.create table opcua_telemetry (DataSetWriterID: string, ExpandedNodeID: string, Value: dynamic, SourceTimestamp: datetime)
```

Se crea la función de descomposición de la carga útil (***payload***):

```
.create-or-alter function OPCUADatasetExpand() {
    opcua_intermediate
    | mv-apply payload on (
        extend key = tostring(bag_keys(payload)[0])
        | extend p = payload[key]
     | project ExpandedNodeId = key, Value = todynamic(p.Value), SourceTimestamp = todatetime(p.SourceTimestamp)
    )
}
```

De nuevo se actualiza la política de ingesta para la tabla final de la telemetría con la función creada:

.alter table opcua_telemetry policy update @'[{"Source": "opcua_intermediate", "Query": "OPCUADatasetExpand()", "IsEnabled": "True"}]'

Hasta este punto se cuenta con una tabla que contiene cada una de las muestras provenientes de los diferentes dispositivos, acompañadas de una estampilla de tiempo y el valor de dicha variable. A partir de este punto, se procede a dividir la información acorde a las necesidades del negocio.

9.4. Creación de Tablas por SCADA

Para facilitar el análisis de los diferentes datos, se crea una tabla para cada sistema SCADA las cuales almacenaran únicamente la telemetría de cada uno. El proceso para cada una consiste en los tres mismos pasos para llevar datos de una tabla a otra, pero en este caso las funciones almacenadas cuentan con comandos que permiten discriminar las variables que se requieren en cada tabla.

Dado que la telemetría obtenida solo cuenta con el nombre de la variable, el valor y la estampilla de tiempo, se realiza la ingesta de tablas de referencia adicionales, las cuales consisten en los diccionarios de datos de cada variable recibida. Esto permite enriquecer la telemetría y a su vez los diferentes tableros que se pueden crear con esta.

Carga de Diccionarios de Datos

Para la carga de tablas de referencia adicionales, se procede a acceder a la interfaz de usuario web ingresando a la pestaña ***Query*** en la barra lateral y pulsando ***Abrir en IU web***, se abrirá la página mostrada en la Figura 34.

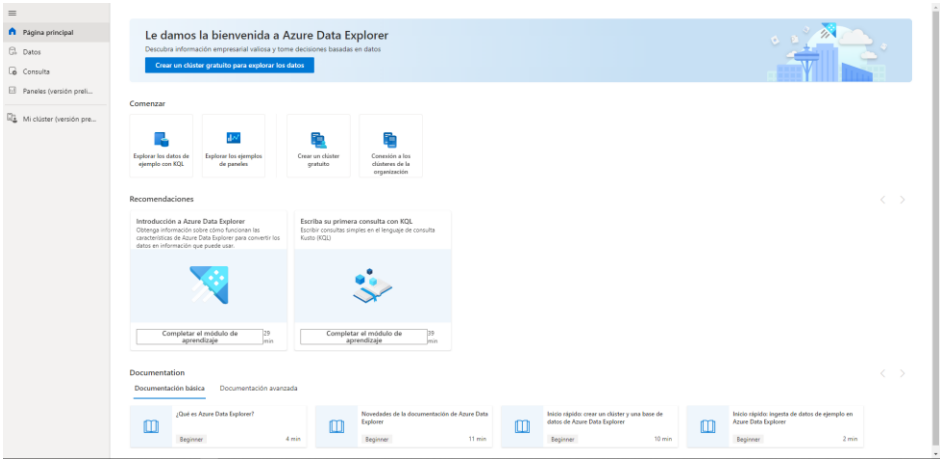

Figura 34. Interfaz de Usuario web Azure Data Explorer.

Se abre la pestaña Datos del panel lateral y se seleccionó ***Ingerir nuevos datos***:

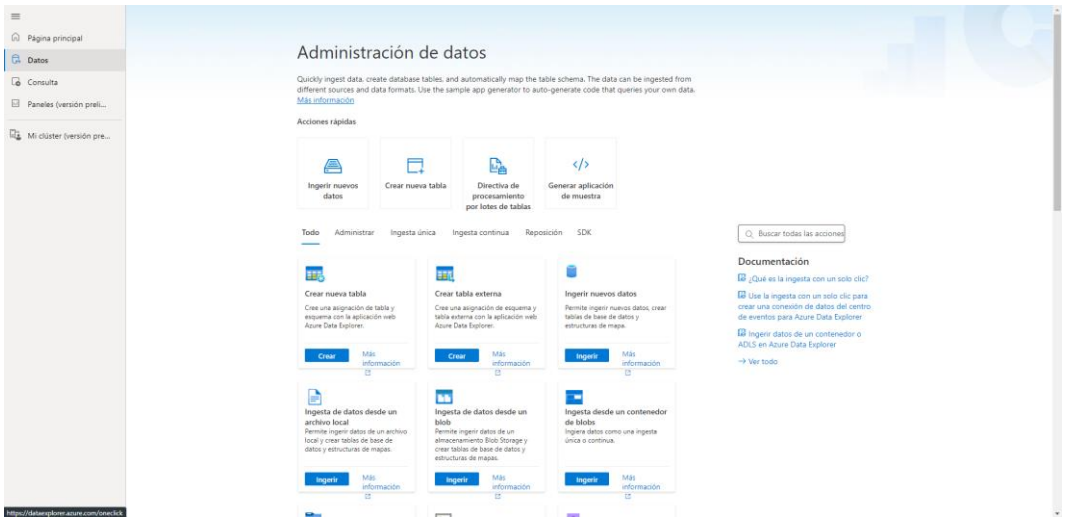

Figura 35. Administración de Datos.

Se define el nombre de la nueva tabla de referencia y se presiona siguiente:

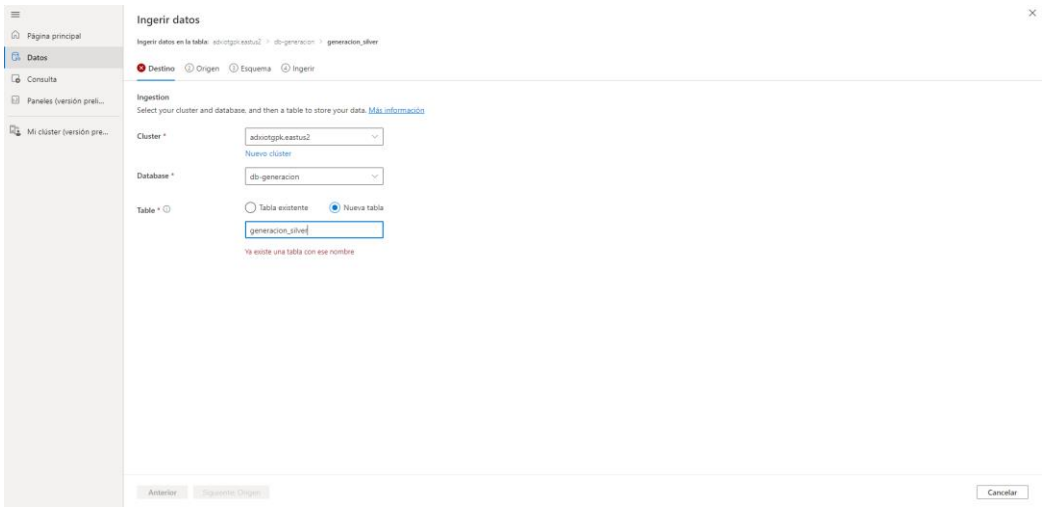

Figura 36. Definición de nueva Tabla para Diccionario de Datos

Se selecciona el archivo .csv de referencia y nuevamente se presiona siguiente:

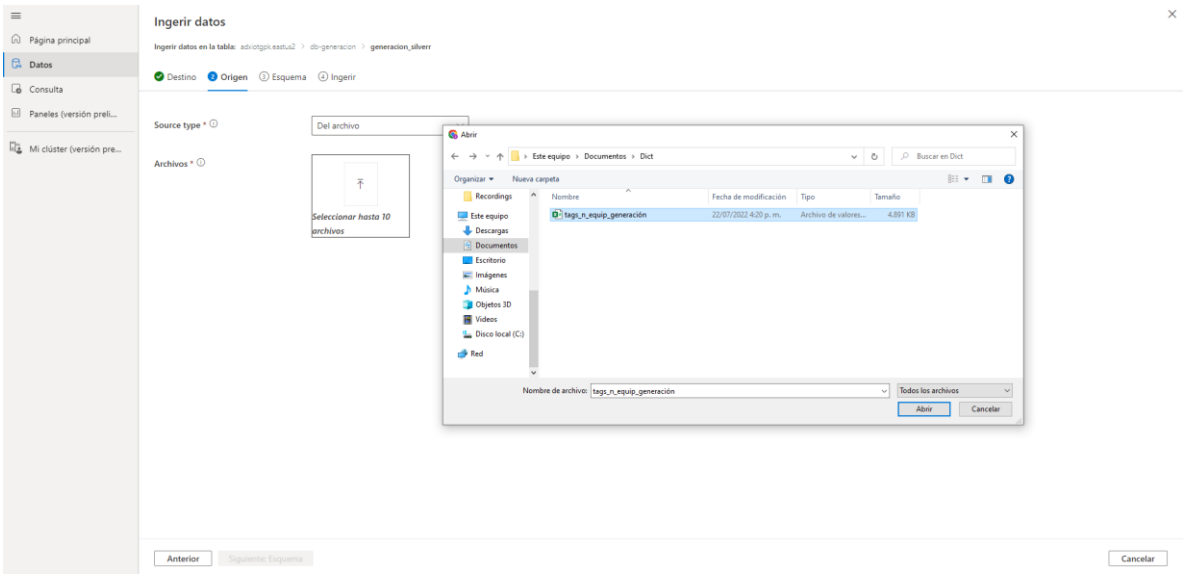

Figura 37. Carga de Diccionario de Datos.

El asistente de ingesta infiere automáticamente el esquema de la tabla cargada, además es posible descartar el primer registro en caso de que no tome los nombres de las columnas de manera automática. Finalmente se presiona iniciar ingesta para cargar la tabla de referencia.

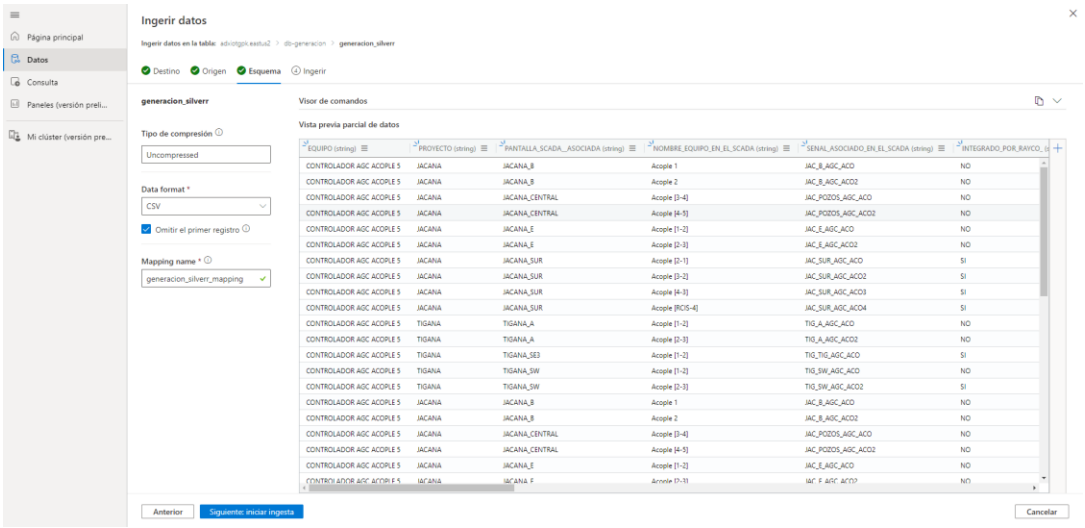

Figura 38. Comprobación de Esquema e Inicio de Ingesta.

Comandos y Consultas

Accediendo a la pestaña consulta se abre una ventana en la cual es posible ejecutar múltiples comandos ***KQL***, en esta se realiza la creación de las tablas necesarias para la distribución de los datos presentes en la tabla ***ocupa_telemetry***.

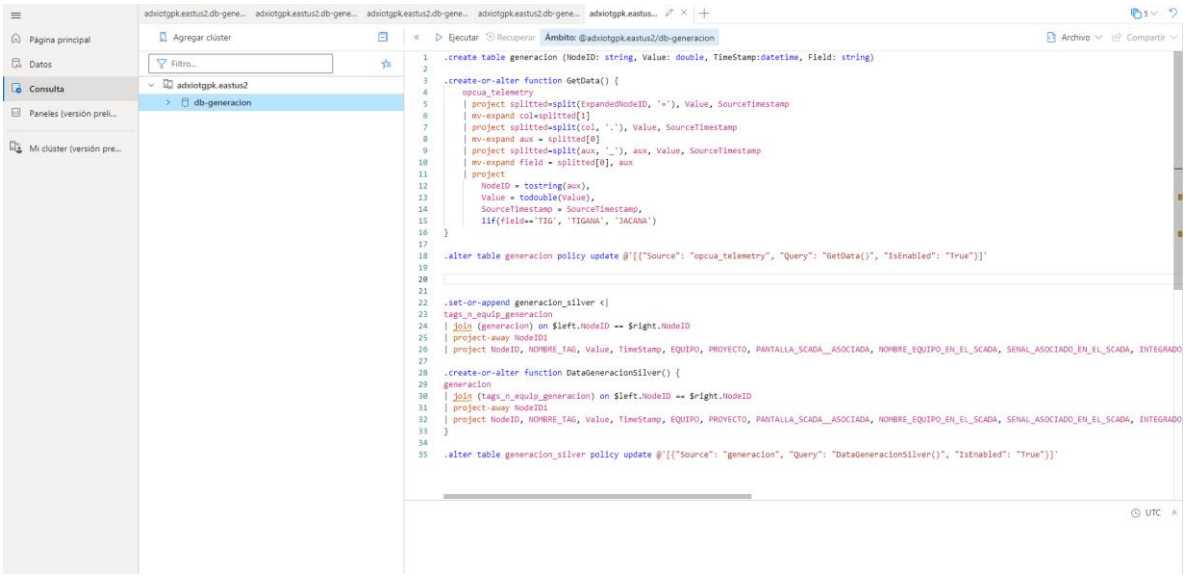

Figura 39. Ventana de consultas.

Son creadas las tablas ***generacion***, ***media_tension*** y ***variables_pozos*** en las que se distribuye la telemetría sin enriquecer desde la tabla ***ocupa_telemetry***. Adicionalmente se crean las funciones almacenadas que permitirán almacenar en tiempo real la telemetría proveniente del IoT Hub. A continuación, se muestran los tres pasos requeridos para particionar en tablas independientes por SCADA.

Comando Creación de Tabla Generación:

```
.create table generacion (NodeID: string, Value: double, TimeStamp:datetime, Field: string)
```

Función almacenada de ingesta en tabla generación desde tabla opcua_telemetry:

```
.create-or-alter function GetData() {
opcua_telemetry
| where ExpandedNodeID has('ge.com')
| project splitted = split(ExpandedNodeID, '='), Value, SourceTimestamp
| mv-expand col = splitted[1]
| project splitted = split(col, '.'), Value, SourceTimestamp
| mv-expand aux = splitted[0]
| project splitted = split(aux, '_'), aux, Value, SourceTimestamp
| mv-expand field = splitted[0], aux
| project
NodeID = tostring(aux),
Value = todouble(Value),
SourceTimestamp = SourceTimestamp,
```

```
iif(field==’TIG’, ‘TIGANA’, ‘JACANA’)
}
```

Actualización de política de ingesta de tabla generación:

```
.alter table generacion policy update @’[{”Source”: “opcua_telemetry”, “Query”: “GetData()”, “IsEnabled”: “True”}]’
```

En cuanto se cuenta con los datos de cada SCADA particionada en diferentes tablas, se procede a la creación de las tablas ***silver*** que contendrán la telemetría enriquecida con el diccionario almacenado previamente. La creación de las nuevas tablas se realiza a través de un comando que permite ingestar todo el histórico de las tablas creadas previamente.

Almacenado de histórico de telemetría, enriquecida con diccionario de datos:

```
.set-or-append generacion_silver <|
tags_n_equip_generacion
| join (generacion) on $left.NodeID == $right.NodeID
| project-away NodeID1
| project NodeID, NOMBRE_TAG, Value, TimeStamp, EQUIPO, PROYECTO,
    PANTALLA_SCADA__ASOCIADA, NOMBRE_EQUIPO_EN_EL_SCADA,
    SENAL_ASOCIADO_EN_EL_SCADA, INTEGRADO_POR_RAYCO_,
    ESTA_FUNCIONANDO_,PROTOCOLO_DE__COMUNICACION, DIRECCION_IP, ID,
    PUERTA_DE_ENLACE,MASCARA_DE_RED,REFERENCIA_DEL_EQUIPO,DATO,DIRECCIO
    N_MODBUS,OFFSET,UNIDADES,DESCRIPCION,EQUACION,TIPO_DE_DATO, Field
```

Función almacenada para ingesta enriquecida desde ***opcua_telemetry***:

```
.create-or-alter function DataGeneracionSilver() {
generacion
| join (tags_n_equip_generacion) on $left.NodeID == $right.NodeID
| project-away NodeID1
| project NodeID, NOMBRE_TAG, Value, TimeStamp, EQUIPO, PROYECTO,
PANTALLA_SCADA__ASOCIADA, NOMBRE_EQUIPO_EN_EL_SCADA,
SENAL_ASOCIADO_EN_EL_SCADA, INTEGRADO_POR_RAYCO_,
ESTA_FUNCIONANDO_,PROTOCOLO_DE__COMUNICACION, DIRECCION_IP, ID,
PUERTA_DE_ENLACE,MASCARA_DE_RED,REFERENCIA_DEL_EQUIPO,DATO,DIRECCION_
MODBUS,OFFSET,UNIDADES,DESCRIPCION,EQUACION,TIPO_DE_DATO, Field
}
```

Actualización de política de ingesta de tabla generación_silver:

```
.alter table generacion_silver policy update @’[{”Source”: “generacion”, “Query”: “DataGeneracionSilver()”, “IsEnabled”: “True”}]’
```

De manera que los datos sufren múltiples transformaciones y expansiones desde la tabla ***opc_raw*** hasta las tablas finales para ser consumidos por los tableros en la pestaña ***Paneles*** en el panel lateral.

9.5. Creación de Dashboard

Para la creación del ***dashboard***, se accede a la pestaña paneles en la barra lateral, se presiona ***New dashboard*** y se ingresa un nombre como se muestra en la Figura 39.

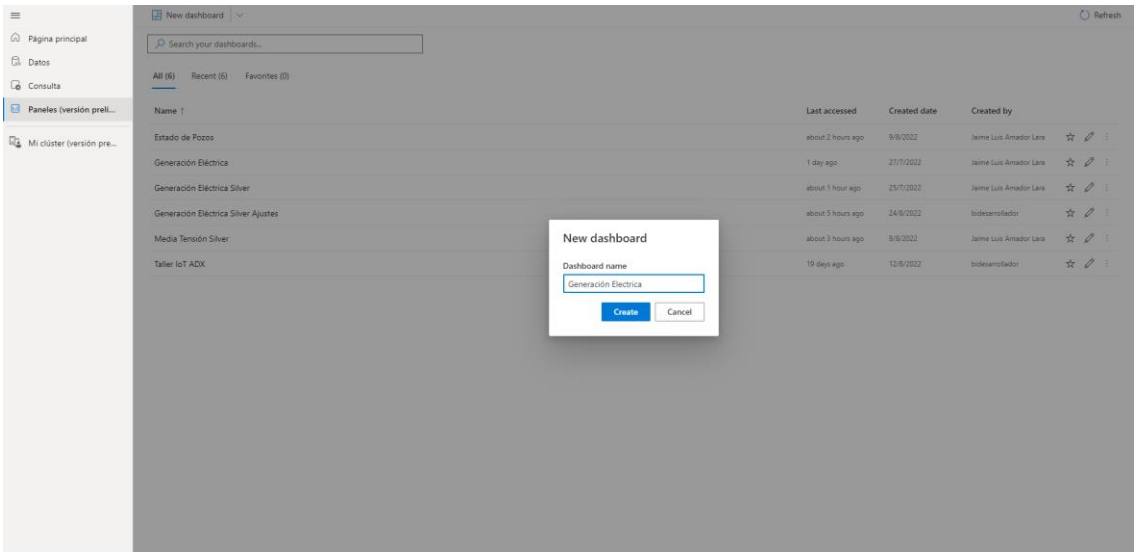

Figura 39. Creación de nuevo dashboard.

En este momento se accede a la ventana de edición del dashboard. En ella ***Azure Data Explorer*** ofrece una interfaz web de usuario que permite la creación de múltiples tableros que se conectan con la base de datos creada anteriormente. A través de estos es posible implementar mosaicos que utilizan fragmentos de ***KQL*** para recuperar datos y representarlos en diferentes visualizaciones entre las que se encuentran ***column charts***, ***time charts***, ***line charts***, ***pie charts***, tablas estadísticas, mapas, entre otros. Además, es posible parametrizar las consultas que constituyen los diferentes mosaicos, de esta manera es posible realizar análisis exploratorios sobre los diferentes datos que permitan tareas como la detección de anomalías, detección de patrones o inclusive predicción de valores futuros.

Al presionar en ***Add title*** como se muestra en la Figura 40, se solicita la creación de la conexión con la base de datos como se observa en la Figura 41, al presionar ***+ Data source***. Deberá nombrarse la conexión e ingresar la ***URI*** del cluster. Al presionar ***Apply*** se establecerá la conexión si todos los campos están correctos.

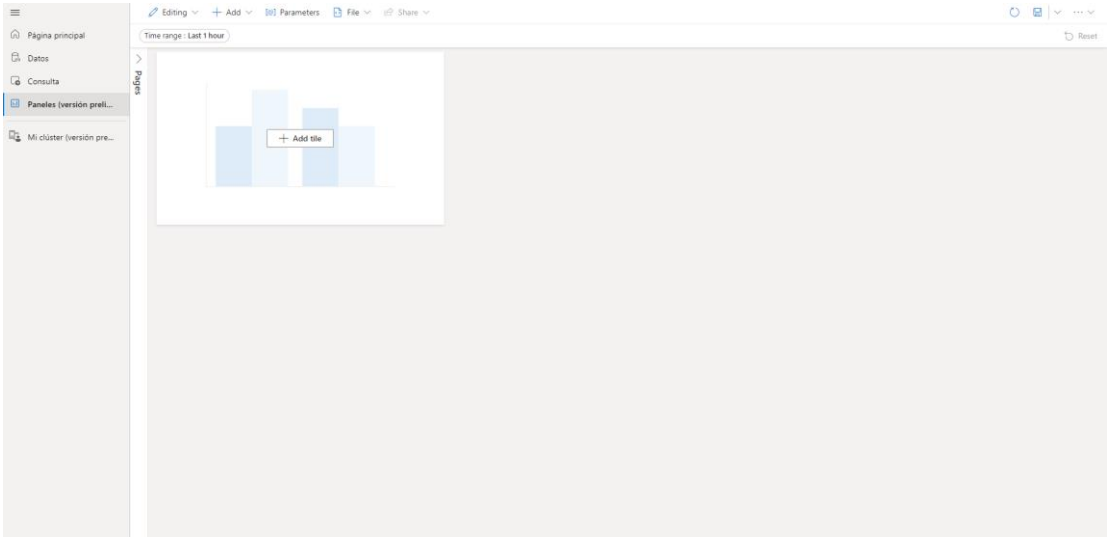

Figura 40. Añadir mosaico.

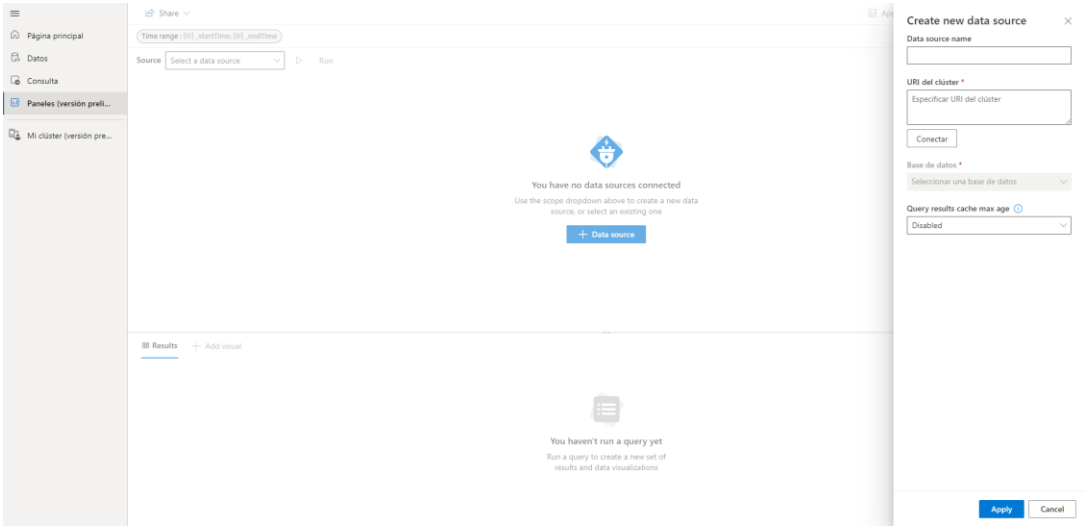

Figura 41. Conexión con base de datos.

Es posible encontrar la URI del cluster al acceder a la pestaña ***Overview*** . **Figura 42.**

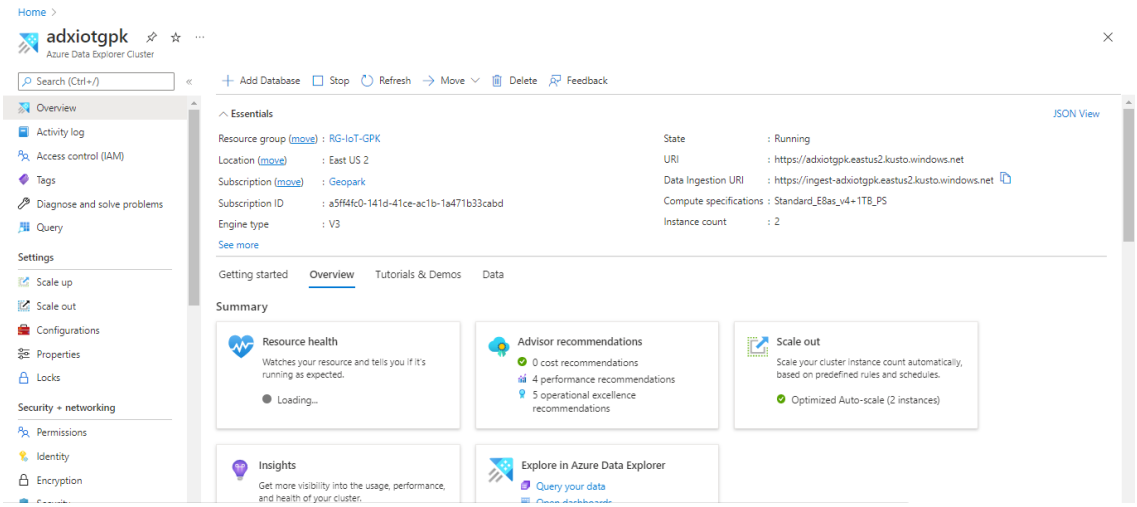

Figura 42. URI cluster.

Al contar con la fuente de datos, se procede a crear los diferentes fragmentos de ***KQL*** para realizar las consultas requeridas que serán mostradas con mosaicos. En la Figura 43, se aprecia una visual de ejemplo de lo que sería la construcción de una consulta.

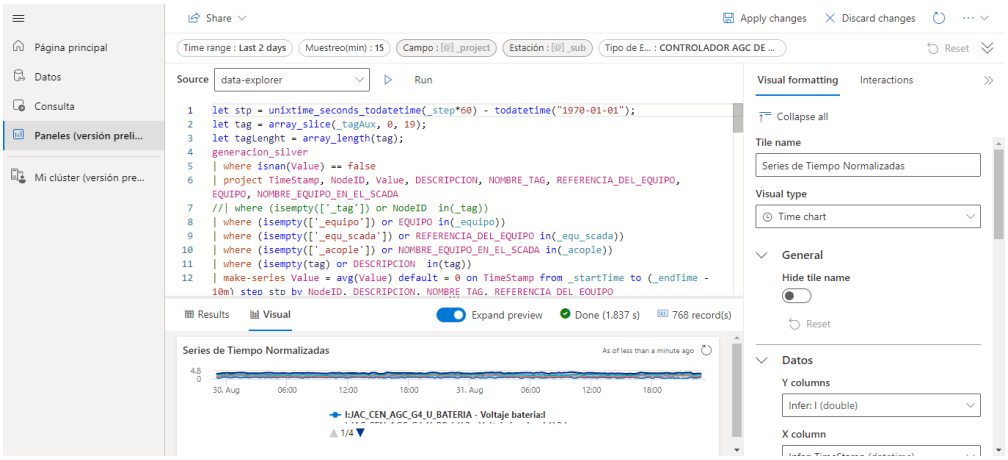

Figura 43. Creación de una consulta el *KQL*.

La parametrización de las consultas que gobiernan las visualizaciones consiste en consultas que permiten la generación de listas desplegables conformadas por una columna de una de las tablas de la base de datos. Estas listas desplegables pueden ser configuradas para que el usuario seleccione entre uno o múltiples valores disponibles. Para crear un parámetro se accede a la pestaña ***Parameters*** y se debe seleccionar ***New parameter*** de la barra lateral derecha emergente como se ilustra en la Figura 44.

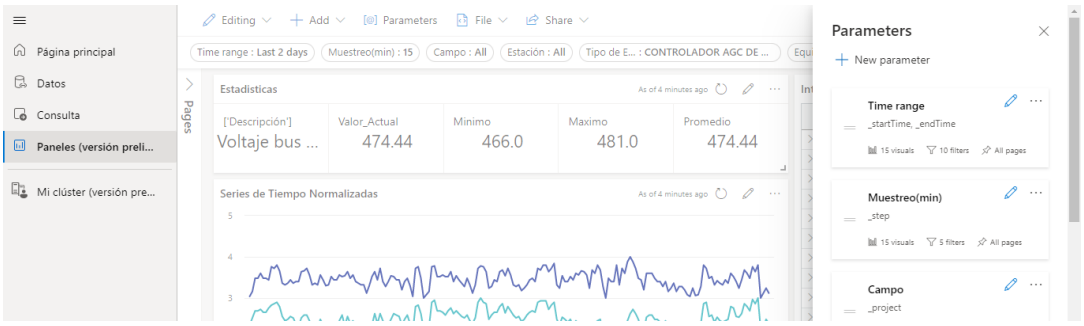

Figura 44. Creación nuevo parámetro.

Para configurar cada parámetro se debe asignar la etiqueta que aparecerá en el dasboard, así como el nombre de la variable a utilizar en las consultas y la cual contendrá el valor seleccionado de la lista desplegable final. Debe seleccionarse el tipo de parámetro acorde a su naturaleza, bien sea tipo texto, numérico o de tiempo.

Otra característica es que es posible agregar un valor fijo o que este se base en una consulta en ***KQL***. En la figura 45 se observa la ventana de configuración.

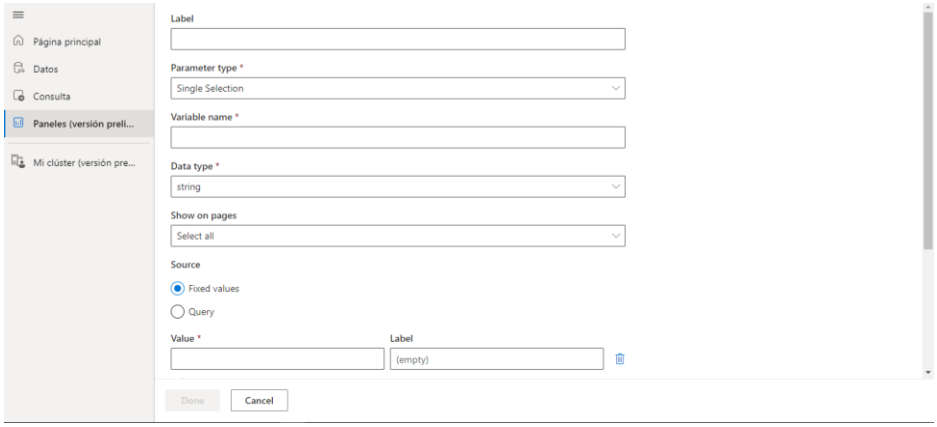

Figura 45. Configuración de parámetros.

A continuación, se detallan los diferentes patrones y mosaicos implementados en los tableros, así como ejemplos y uso de estos.

Parámetros

- Rango de Tiempo
- Muestreo
- Filtros

Pagina Vista General

- Estadísticas
- Series de Tiempo Normalizadas
- Serie de Tiempo Seleccionada
- Intervalo de Serie

- Valor Instantáneo de Variables

Pagina Análisis Detallado de Variables

- Estadísticas
- Serie de Tiempo
- Estacionalidad
- Detección de Anomalías
- Pronostico
- Instante
- Intervalo
- Significancia de Periodicidad (%)
- Frecuencia de Repetición (min)
- Periodos Detectados

Rango de Datos: Permite seleccionar el rango de tiempo en el que se requiere visualizar los datos.

Muestreo: Dado que cada variable es reportada con una periodicidad diferente, se permite la configuración del periodo de muestreo en el que van a ser mostrados los datos.

Filtros: De acuerdo con las columnas de la tabla de cada SCADA en la base de datos, se cuentan con múltiples parámetros que permiten acotar la búsqueda de una variable en específico.

Tomando como ejemplo generación eléctrica, se tomaron las columnas de campo, estación, equipo, tipo de equipo, acople y descripción para filtrar la búsqueda de una señal especifica, esto mismo ocurre en los demás ***dashboard*** pero acorde a las columnas de cada uno.

Estadísticas: Ofrece una descripción general de una variable seleccionada en un rango de tiempo. En esta se incluye: valor actual, valor mínimo, valor máximo y promedio.

Series de Tiempo Normalizadas: En la gráfica se pueden visualizar hasta 20 series de tiempo normalizadas, esto con el fin de poder contar con una visual de múltiples variables en una sola gráfica y facilitar el análisis de estas.

Serie de Tiempo Seleccionada: Permite visualizar una serie de tiempo en concreto sin normalizar para así tener un mayor detalle de los valores reales de la telemetría de cada tag.

Intervalo de Serie: Es una tabla que muestra los datos presentes en las series de tiempo normalizadas, con las demás columnas correspondientes a cada tag. Esta tabla cambia acorde al rango de datos seleccionado en el gráfico de las series normalizadas arrastrando el puntero.

Valor Instantáneo de Variables: Muestra el valor de todas las series de tiempo normalizadas visualizadas de un momento dado. Es posible escoger el momento al presionar en cualquier punto a lo largo de la gráfica.

Serie de Tiempo: Idéntica a serie de tiempo seleccionada. Esta se ubica en la página de análisis detallado de variables para contar en todo momento con la visual de la serie que se está analizando.

Estacionalidad: Se realiza la descomposición de la serie de tiempo seleccionada y se obtiene el componente de estacionalidad.

Detección de Anomalías: A partir de la estacionalidad detectada, la fuerza de esta y el residuo de la serie temporal, se determina la ocurrencia de anomalías y el momento en el que esta ocurre.

Pronostico: Nuevamente se parte de la estacionalidad detectada y se utilizan las funciones de pronóstico integradas en el ADX para determinar posibles futuros valores en un rango de 7 días.

Instante: Valor instantáneo en forma de tabla de la serie de tiempo analizada.

Intervalo: Rango de datos en forma de tabla de la serie de tiempo analizada. Esta se ajusta al igual que el intervalo de series al arrastrar el puntero sobre la gráfica de la serie que se esté visualizando.

Significancia de Periodicidad: Indica la fuerza de la estacionalidad detectada en la serie de tiempo analizada. Toma valores entre 0 y 100 %.

Frecuencia de repetición: Muestra en minutos cada cuanto tiempo se repite el patrón detectado en la serie de tiempo.

Periodos Detectados: Cada serie de tiempo puede contar con cero o múltiples patrones en un rango de tiempo dado. Esta tabla muestra los dos patrones más significativos e indica cada cuanta muestra se repite dicho cada patrón.

En las Figuras 46 y 47 se puede observar el tablero implementado para generación eléctrica.

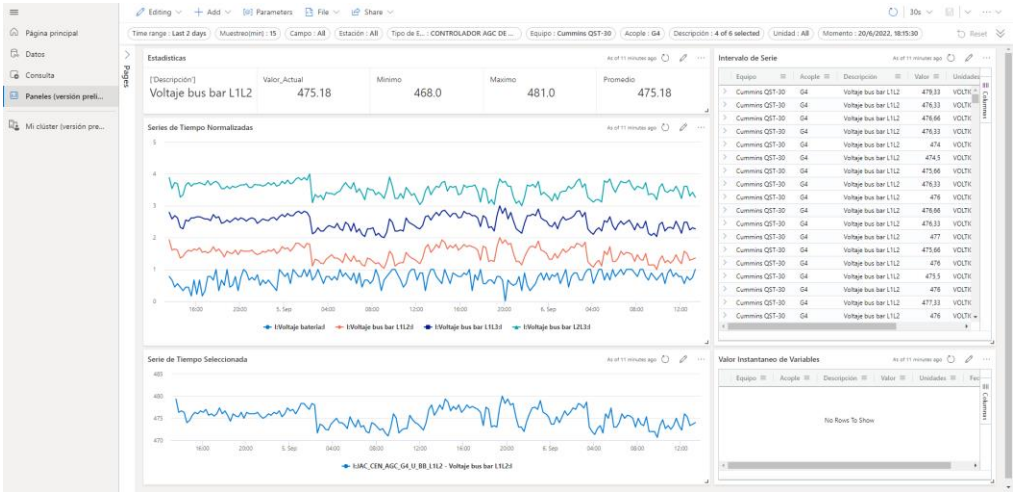

Figura 46. Pagina vista general.

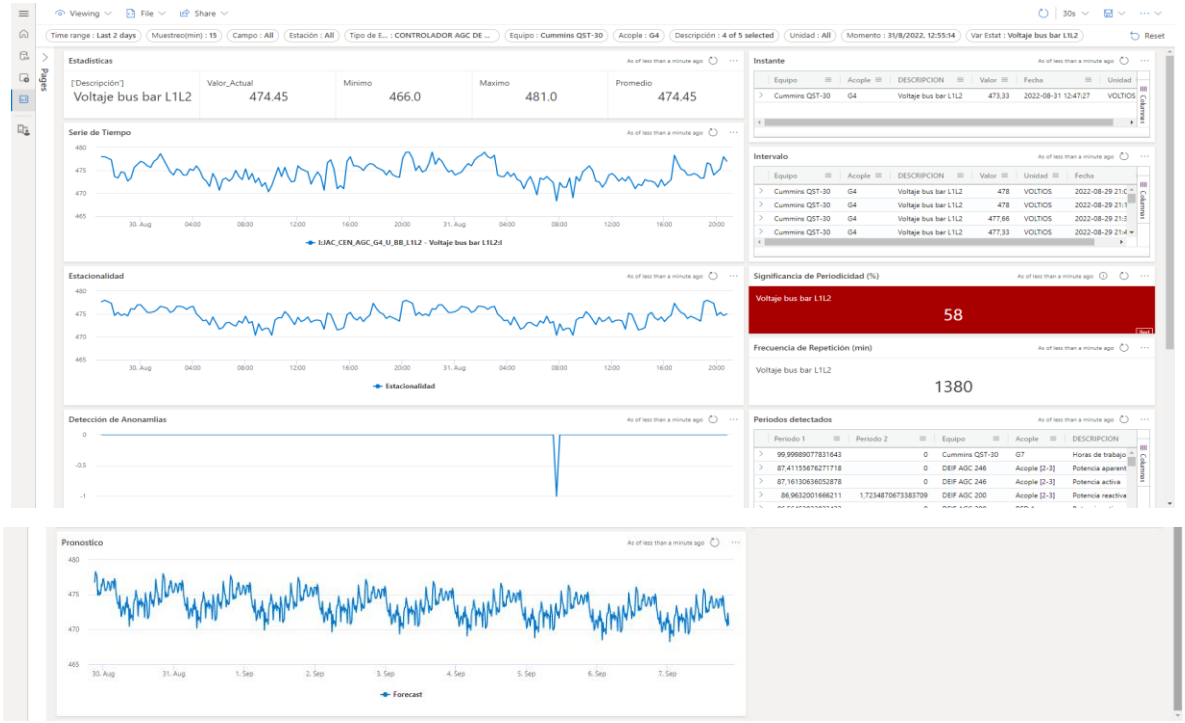

Figura 47. Página de análisis detallado de variables.

10. Matriz de Roles y tareas

La siguiente Matriz RACI resume los roles y tareas comunes en cada uno de los servicios que hacen parte de la suscripción, así como la acción que debe realizar cada uno según sus tareas. Adicionalmente, dentro del portal de Azure se encuentra como está actualmente la asignación de roles según la suscripción

		ROLES					
		Colaborador de ADX	Lector de ADX	Admin de ciberseguridad	Propietario	Colaborador IoT Hub -Edge	Admin de accesos
TAREAS	Administrador de todas las bases de datos Puede mostrar y modificar ciertas políticas a nivel de clúster. Editor del tablero.	R	I	C	A	R	I
	usuario de la base de datos y visor de todas las bases de datos y tableros.	A	R	I	A	I	C
	Administración de la seguridad y auditar la infraestructura tanto on-premises como en la nube.	I	I	R	A	I	I
	Acceso a todos los modulos y devices, permisos para asignar roles, todas las acciones.	I	I	C	A	R	R
	Permite el acceso de lectura y escritura a todos los módulos gemelos y dispositivos de IoT Hub.	I	I	C	A	R	R
	Control de Accesos a los servicios.	I	I	C	A	I	R

R	RESPONSABLE
A	APROBADOR
C	CONSULTADO
I	INFORMADO

11. Conclusiones

- Con los servicios de IoT se logró crear la telemetría para extraer los datos en un tiempo cercano al real para el análisis y visualización de estos. Debido a esto se debe contar con un servicio de almacenamiento en la nube donde se pueda almacenar grandes volúmenes de datos, actualmente se almacena alrededor de 0.5 GB diarias, lo que es mucha información, por lo tanto, este servicio de almacenamiento se depura cada 6 meses

- Es posible que los servicios en la nube fallen y no procesen los datos, para reportar cualquier tipo de fallo se crea un proceso de notificación que envía un correo electrónico notificando la causa del error para corregirlo en el menor tiempo posible y garantizar el funcionamiento de la solución

- Los reportes creados son monitoreados por técnicos encargados de la operación, que están capacitados para detectar cualquier tipo de falla con base en los datos procesados

- La creación del proyecto sirve como base para futuros proyectos, donde los datos recopilados en el datalake no solo sirven para la visualización de estos, si no que se puede utilizar para crear un modelo predictivo donde se pueda indicar una fecha de mantenimiento para cada uno de los generados y evitar que estos se dañen en cualquier momento; o se pueden utilizar para modelos analíticos dependiendo la necesidad de la operación.

- Un beneficio de guardar los datos que se ingesta desde las SCADAS, es lograr centralizar la información de la compañía en un mismo servicio de almacenamiento. Otros proyectos transversales al realizado guardan la información en el lago de datos en diferentes capas de información y con una herramienta de analítica se logran centralizar la información y crear un data marts con tablas de diferentes áreas con el fin de crear una bodega de datos.

12. Bibliografía

- Create and provision an IoT Edge for Linux on Windows device using symmetric keys. Product documentation Microsoft. Recuperado de: https://docs.microsoft.com/en-us/azure/iot-edge/how-to-provision-single-device-linux-on-windows-symmetric?view=iotedge-2020-11&tabs=azure-portal%2Cpowershell, 2022-09-01

- Nested virtualization for Azure IoT Edge for Linux on Windows. Product documentation Microsoft. Recuperado de: https://learn.microsoft.com/en-us/azure/iot-edge/nested-virtualization?view=iotedge-2020-11, 2022-10-12

- Conceptos de IoT y Azure IoT Hub, Product documentation Microsoft. Recuperado de: https://learn.microsoft.com/es-es/azure/iot-hub/iot-concepts-and-iot-hub, 2022-10-12

- What is Azure IoT Edge, Product documentation Microsoft. Recuperado de: https://learn.microsoft.com/en-us/azure/iot-edge/about-iot-edge?view=iotedge-2020-11, 2022-10-16

- Prepare to deploy your IoT Edge solution in production, Product documentation Microsoft. Recuperado de: https://learn.microsoft.com/en-us/azure/iot-edge/production-checklist?view=iotedge-2020-11, 2022-10-16

- Using Industrial IoT to pull data into Azure Data Explorer, Product documentation Microsoft. Recuperado de: https://learn.microsoft.com/en-us/azure/industrial-iot/tutorial-data-explorer-import, 2022-10-18

- Understand extended offline capabilities for IoT Edge devices, modules, and child devices, Product documentation Microsoft. Recuperado de: https://learn.microsoft.com/en-us/azure/iot-edge/offline-capabilities?view=iotedge-2020-11, 2022-10-18