**IOT MIDDLEWARE**

**INTRODUCTION:**

**Introduction:**

The Internet of Things (IoT) is a network of physical devices that collect and exchange data over the internet. These devices can range from everyday household items like refrigerators and thermostats to industrial machines in factories. However, managing and integrating the vast number of devices and the complex data they generate can be a significant challenge. This is where **IoT middleware** comes in.

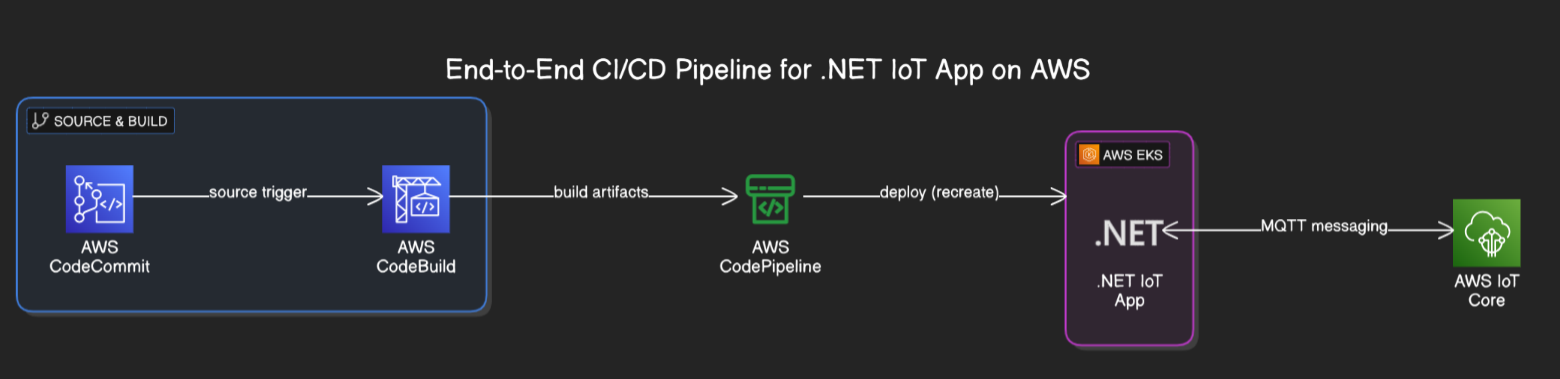
**IoT Middleware:**

IoT middleware is a software layer that sits between the IoT devices and applications, abstracting the complexities associated with device communication, data processing, and integration. Its main role is to facilitate the interoperability of diverse IoT devices, sensors, and applications that might otherwise be incompatible due to different communication protocols, data formats, or hardware constraints.

**Role of IoT Middleware:**

1. **Abstraction of Device Communication:**  
   Middleware helps to manage and abstract the communication between IoT devices and applications, regardless of their underlying protocols or hardware platforms. For instance, it can handle MQTT, CoAP, or HTTP protocols, allowing applications to communicate with a wide variety of devices without needing to know their specific protocols.
2. **Data Transformation & Processing:**  
   IoT middleware provides mechanisms for data collection, aggregation, filtering, and transformation, ensuring that data from devices is properly formatted and processed before it reaches the application layer. This allows for cleaner data flow and easier consumption by downstream systems.
3. **Security and Authentication:**  
   It plays a critical role in ensuring secure communication between devices and applications by implementing encryption, authentication, and authorization protocols to protect sensitive data.
4. **Scalability and Flexibility:**  
   Middleware can manage large-scale IoT ecosystems by providing features like device management, message routing, and load balancing, enabling applications to scale as more devices are added.
5. **Integration with Cloud and Other Services:**  
   It provides connectivity between IoT devices and cloud services, enabling IoT solutions to integrate with backend services, storage, and databases. Middleware also facilitates integration with third-party applications and APIs.

**SYSTEM ARCHITECTURE:**

****

This architecture, a fully automated CI/CD pipeline designed for a .NET-based IoT application using AWS services. The pipeline begins with AWS CodeCommit, which hosts the application source code. When changes are pushed to the repository, a source trigger activates AWS CodeBuild, initiating the build process.

CodeBuild compiles the application, runs tests, and generates build artifacts. These artifacts are then passed to AWS CodePipeline, which orchestrates the deployment workflow. The pipeline deploys the application to AWS EKS (Elastic Kubernetes Service), ensuring the .NET IoT app is containerized and managed within a scalable Kubernetes cluster.

Once deployed, the application communicates with AWS IoT Core using MQTT messaging, enabling low-latency, bidirectional communication between the IoT app and connected IoT devices. This architecture supports automated, repeatable deployments and seamless integration between cloud-native development tools and IoT services.

**Build & Push Docker Image (on your local machine)Source code:**

FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base

WORKDIR /app

EXPOSE 80

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build

WORKDIR /src

RUN dotnet publish -c Release -o /app

FROM base AS final

WORKDIR /app

COPY --from=build /app .

ENTRYPOINT ["dotnet", "App.dll"]

# 1. Build your Docker image

docker build -t dotnet-iot-app .

# 2. Tag the image for ECR

docker tag dotnet-iot-app:latest 123456789012.dkr.ecr.us-west-2.amazonaws.com/dotnet-iot-app

# 3. Authenticate Docker to your ECR

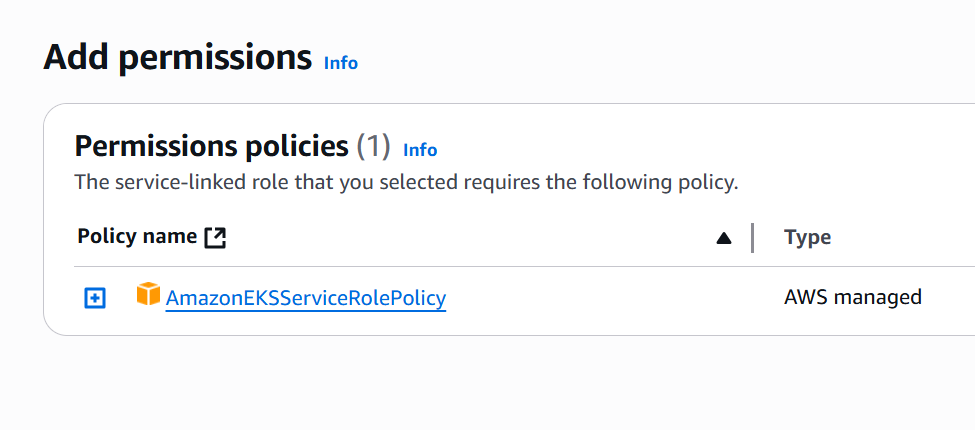
aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin 123456789012.dkr.ecr.us-west-2.amazonaws.com

# 4. Push the image to ECR

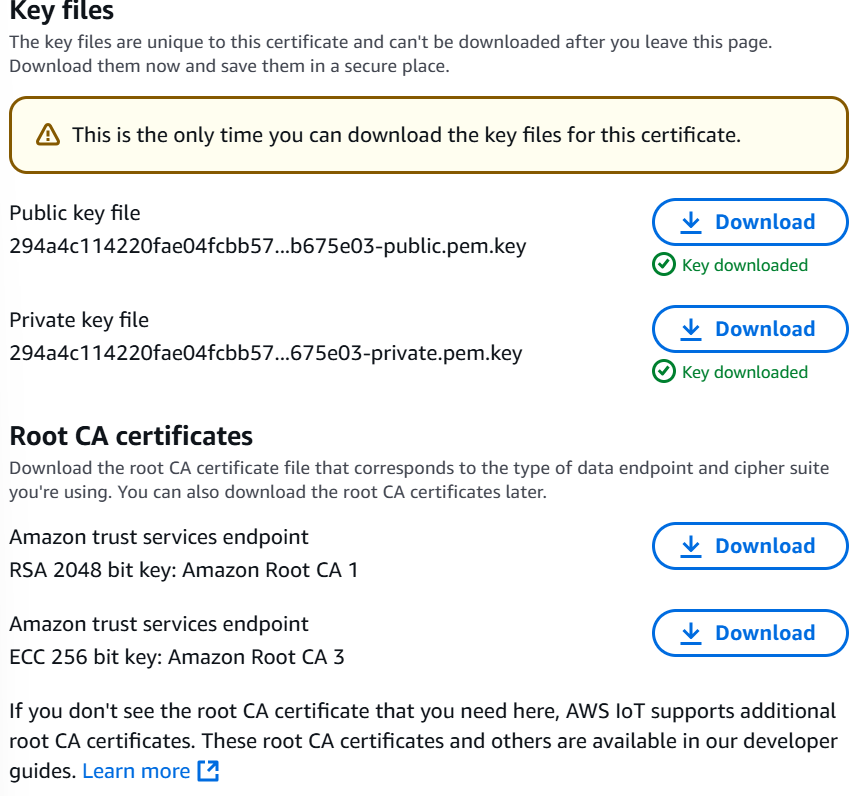
docker push 123456789012.dkr.ecr.us-west-2.amazonaws.com/dotnet-iot-app

**CREATED EKS:**

**AND ADDED ROLE:**

****

**IOT CORE CREATED AND DOWNLOAD CERTIFICATE:**



**Kubernetes Deployment YAML (deployment.yaml):**

apiVersion: apps/v1

kind: Deployment

metadata:

name: iot-app

spec:

replicas: 1

selector:

matchLabels:

app: iot

template:

metadata:

labels:

app: iot

spec:

containers:

- name: iot-app

image: 123456789012.dkr.ecr.us-west-2.amazonaws.com/dotnet-iot-app

ports:

- containerPort: 80

**Kubernetes Service YAML:**

apiVersion: v1

kind: Service

metadata:

name: iot-service

spec:

type: LoadBalancer

ports:

- port: 80

targetPort: 80

selector:

app: iot

**Conclusion:**

This experiment successfully demonstrated the implementation of an end-to-end CI/CD pipeline using AWS CodePipeline to automate the build, test, and deployment of a .NET-based IoT application with MQTT integration onto AWS Elastic Kubernetes Service (EKS). By integrating services like CodeCommit, CodeBuild, and CodeDeploy within the pipeline, we achieved a streamlined and repeatable deployment process. The use of EKS ensured scalable and resilient hosting of the containerized .NET IoT application, while MQTT integration enabled reliable and lightweight message communication suitable for IoT workloads.

Through this setup, we validated that AWS provides a robust ecosystem for managing CI/CD workflows and IoT applications, reducing operational overhead and enhancing deployment speed and reliability. This architecture also lays the foundation for scaling IoT applications in production environments with improved observability, version control, and rollback capabilities.

**DATE ENGINEERING:**

**Introduction to Data Engineering and Its Role in IoT Applications**

Data Engineering is the process of designing, constructing, and maintaining systems that collect, store, and analyze large volumes of data. It focuses on building robust data pipelines that allow for the efficient flow of data from various sources to storage and analysis systems, ensuring that the data is accessible, reliable, and scalable. Data engineering plays a pivotal role in IoT (Internet of Things) environments, as it abstracts the complexities of data collection, transformation, and storage across a wide range of IoT devices, making it easier for applications to interact with that data.

In the context of IoT, data engineering is essential for integrating multiple devices generating real-time data, storing it efficiently, and preparing it for analysis. The role of a data engineer in IoT systems includes:

1. Data Collection: IoT devices generate vast amounts of data, such as temperature readings, sensor values, or device status updates. Data engineering ensures these data streams are continuously collected and ingested into a central processing system.
2. Data Transformation: Raw data coming from devices needs to be processed and transformed into a usable format for analysis or decision-making. Data engineers design ETL (Extract, Transform, Load) pipelines that handle this transformation.
3. Data Storage: Storing massive volumes of time-series data generated by IoT devices requires specialized storage solutions that can scale effectively. Data engineering involves selecting appropriate storage systems, such as databases, data lakes, or time-series databases.
4. Data Accessibility: IoT applications need quick and reliable access to data for real-time analysis and decision-making. Data engineers create efficient data pipelines that ensure data is available on-demand, with low latency.

Through the use of these techniques, data engineering abstracts the complexities involved in managing the large-scale, real-time, and often heterogeneous data sources present in IoT ecosystems, ensuring that the data is usable and actionable for IoT applications.

Technologies to Be Used:

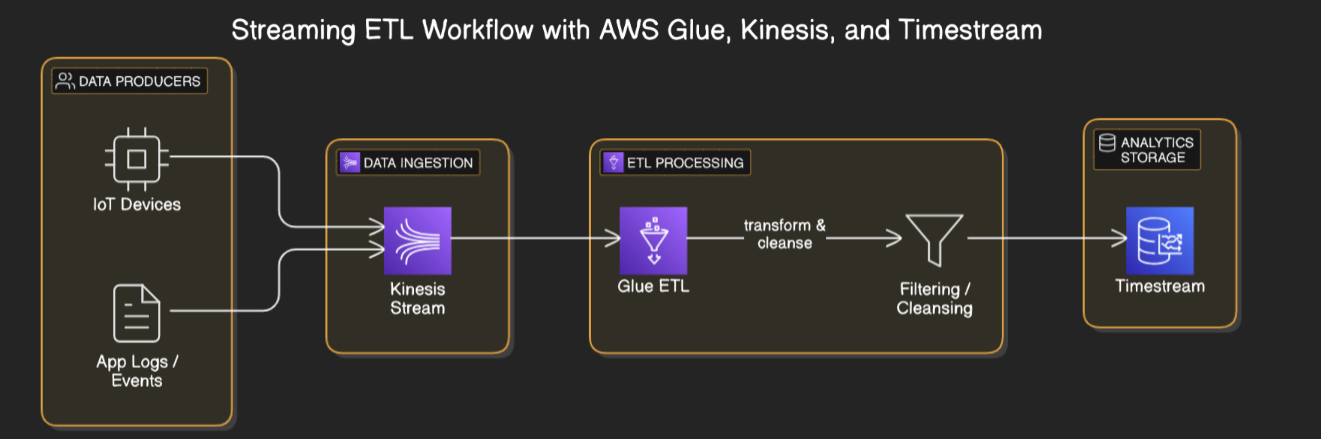
the following technologies will be leveraged to build a streaming ETL workflow from IoT devices to analysis platforms:

1. AWS Glue:
   * AWS Glue is a fully managed ETL (Extract, Transform, Load) service that simplifies the process of preparing data for analytics. It allows for easy integration between data sources like Kinesis (streaming data from IoT devices) and target data stores like Timestream (time-series data storage). It automates the data extraction and transformation processes, making it easier to manage large-scale data workflows.
2. Amazon Kinesis:
   * Kinesis is a platform for real-time data streaming that allows you to collect, process, and analyze data from IoT devices in real-time. It integrates seamlessly with other AWS services like AWS Glue and Timestream, enabling fast ingestion and processing of data as it's generated by IoT devices.
3. Amazon Timestream:
   * Timestream is a fully managed time-series database that is ideal for storing, analyzing, and querying time-stamped data, such as the data generated by IoT sensors. It is optimized for real-time analytics, making it perfect for IoT use cases where large amounts of time-series data are generated continuously.

By using these technologies in combination, this architecture will facilitate the seamless flow of data from IoT devices to an analytics platform, allowing for real-time monitoring, analysis, and actionable insights. The integration of Kinesis, Glue, and Timestream provides a powerful, scalable, and efficient solution for managing and analyzing IoT data.

**System Architecture Overview:**

**Diagram:**

****

**Streaming ETL Workflow with AWS Glue, Kinesis, and Timestream**

This architecture enables real-time data ingestion, transformation, and storage for analytics using AWS services.

**1. Data Producers**

* **IoT Devices** and **Application Logs/Events** act as real-time data sources.
* These sources generate high-frequency, structured/unstructured data in formats like JSON, CSV, or custom binary.
* Examples: temperature sensors, motion detectors, app error logs, and user interaction events.

**2. Data Ingestion – Amazon Kinesis Data Streams (KDS)**

* Kinesis Streams ingest the raw data from producers with millisecond latency.
* It supports parallel writes and scales via shards (each handling 1 MB/sec or 1,000 records/sec).
* Data is retained for a configurable window (default 24 hours, up to 7 days) to allow multiple consumers to read independently.

**3. ETL Processing – AWS Glue Streaming Jobs**

* **AWS Glue** connects directly to Kinesis to perform **real-time ETL (Extract, Transform, Load)** using a serverless **Apache Spark** engine.
* Processing includes:
  + **Schema detection** and data format parsing.
  + **Data transformation**, such as unit conversion, enrichment with metadata, timestamp formatting, or key derivation.
  + **Cleansing and filtering**, including removing nulls, deduplication, and threshold checks.
  + **Error handling** with automatic Dead Letter Queue (DLQ) support for corrupted records.
* Glue uses **checkpointing** to ensure **exactly-once processing**.

**4. Filtering / Cleansing (Inside Glue ETL)**

* Filtering logic ensures only high-quality, relevant data is processed further.
* Examples:
  + Remove readings outside expected sensor value ranges.
  + Normalize formats (e.g., date formats or measurement units).
  + Drop duplicate or irrelevant logs.
* All operations are handled within the ETL Spark script or AWS Glue Studio visual job editor.

**5. Analytics Storage – Amazon Timestream**

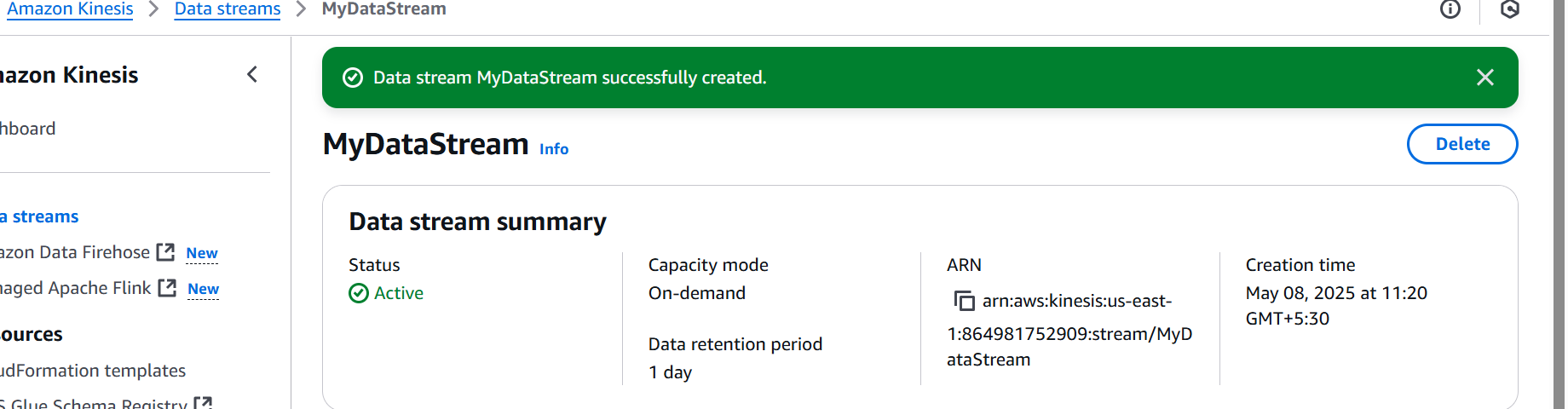
* **Amazon Timestream** is a managed time-series database built for IoT and operational analytics.
* **Features:**
  + **Serverless and auto-scaling**
  + **High ingestion rate** with time-stamped data support
  + **Built-in time-series functions** (like smoothing, interpolation, and aggregations)
  + **Tiered storage**: recent data in memory, older data moved to cost-efficient storage automatically
* Ideal for dashboards, anomaly detection, and time-series analytics (e.g., Grafana, Amazon QuickSight).

**End-to-End Flow**

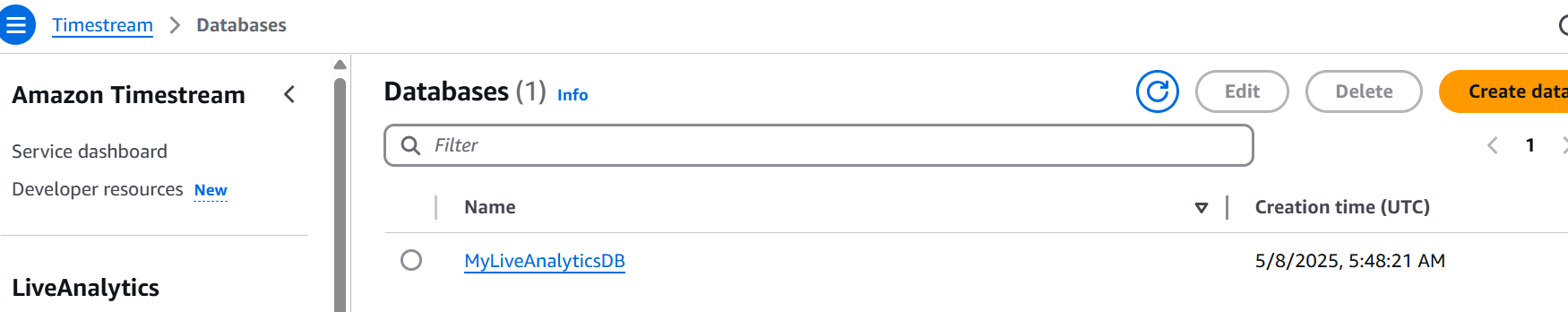
1. IoT Devices / App Logs send real-time data.
2. Data is ingested via Amazon Kinesis Streams.
3. AWS Glue Streaming ETL job consumes, transforms, and cleanses this data.
4. Cleaned data is filtered and structured within Glue.
5. Final output is stored in Amazon Timestream, ready for low-latency time-series querying and real-time dashboards.

**Data injested via kinesis:**

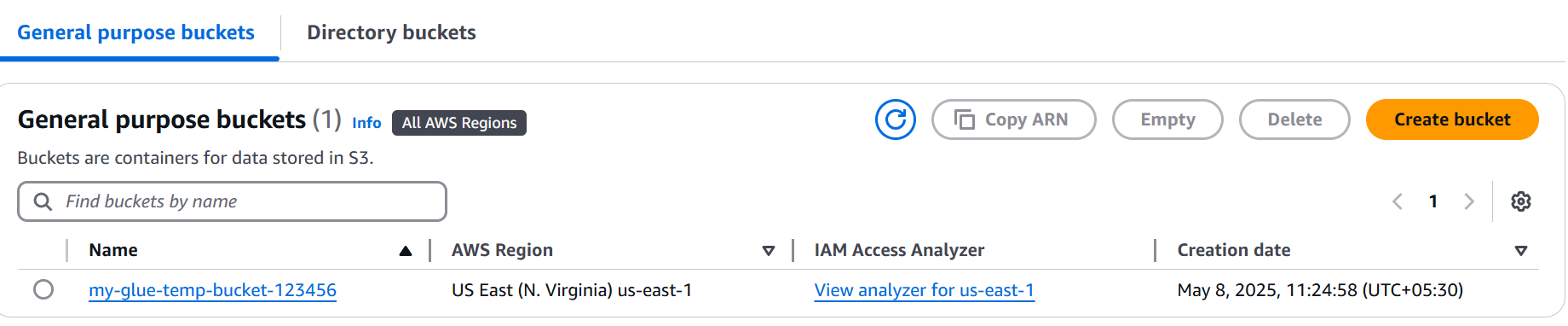
**Created kinesis:**

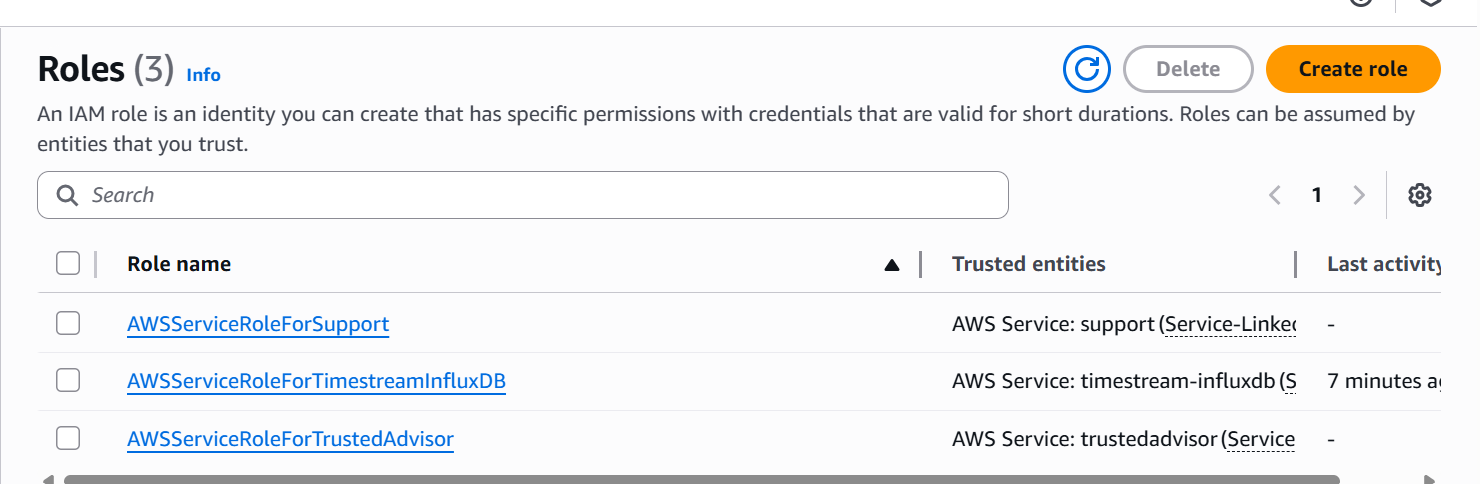
****

**Timestream (live analytic):**

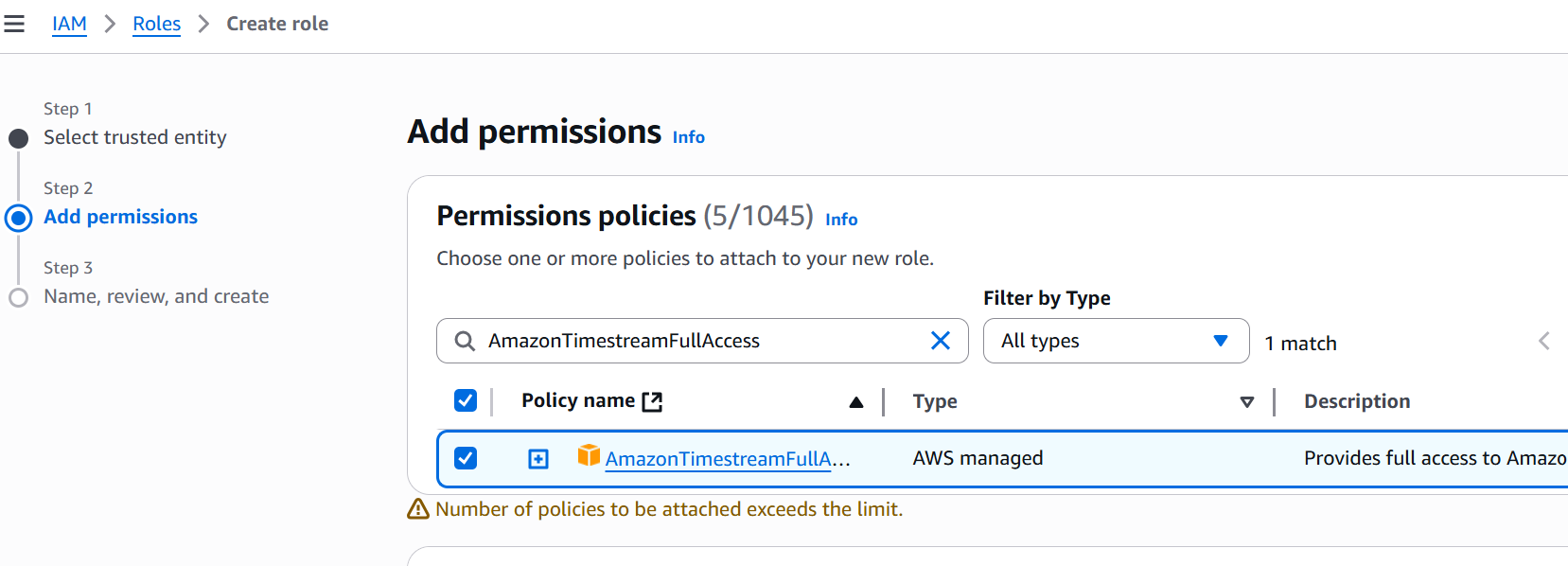
****

**S3 bucket:**

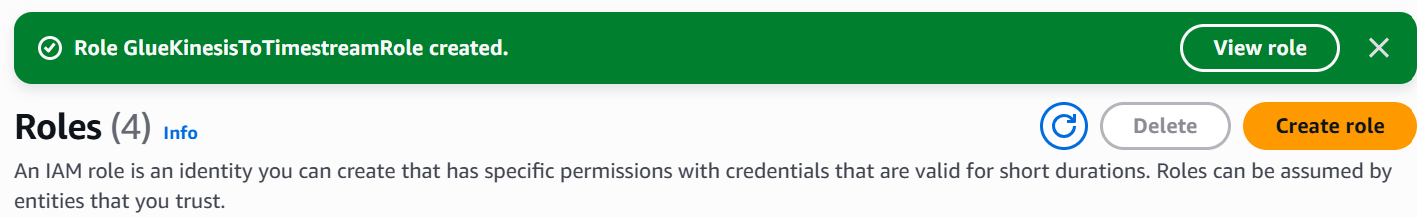
****

****

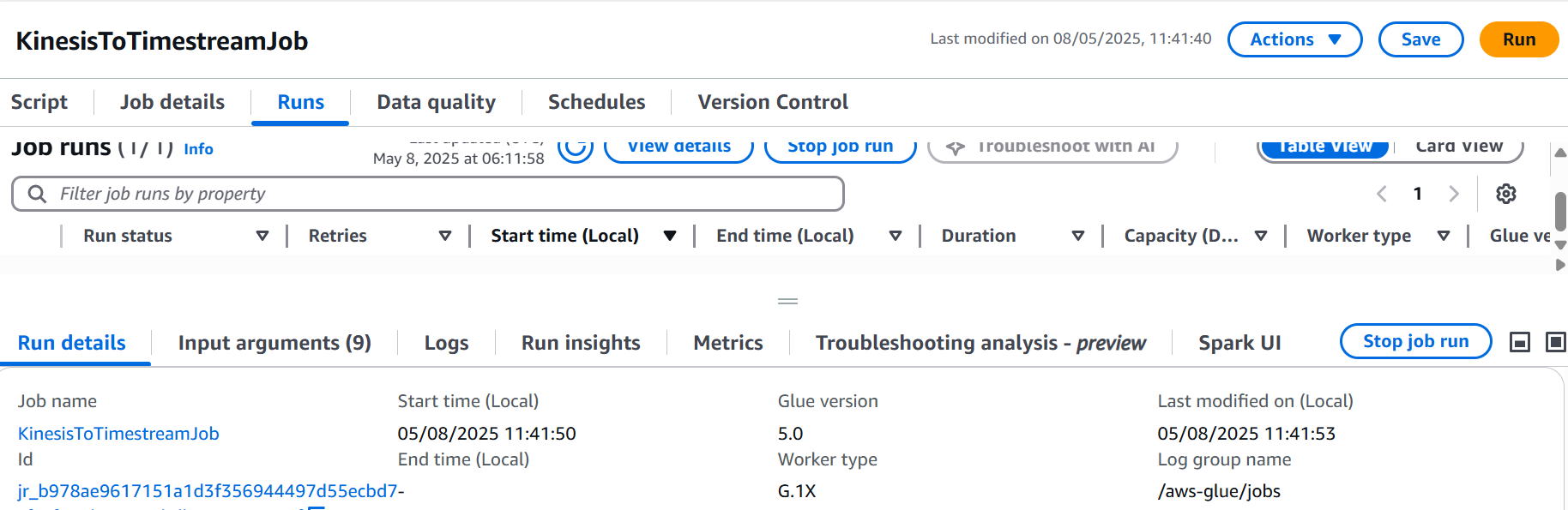
**IAM role added permission:**

****

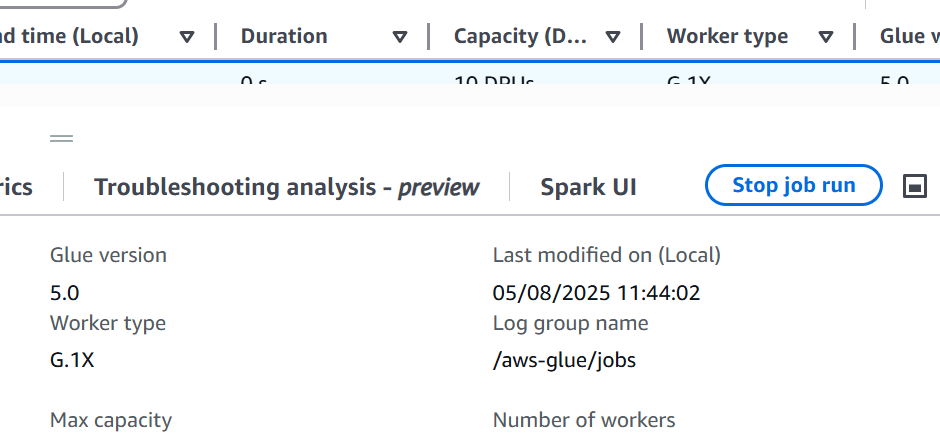
**IAM ROLE: (TIMESTREAM)**

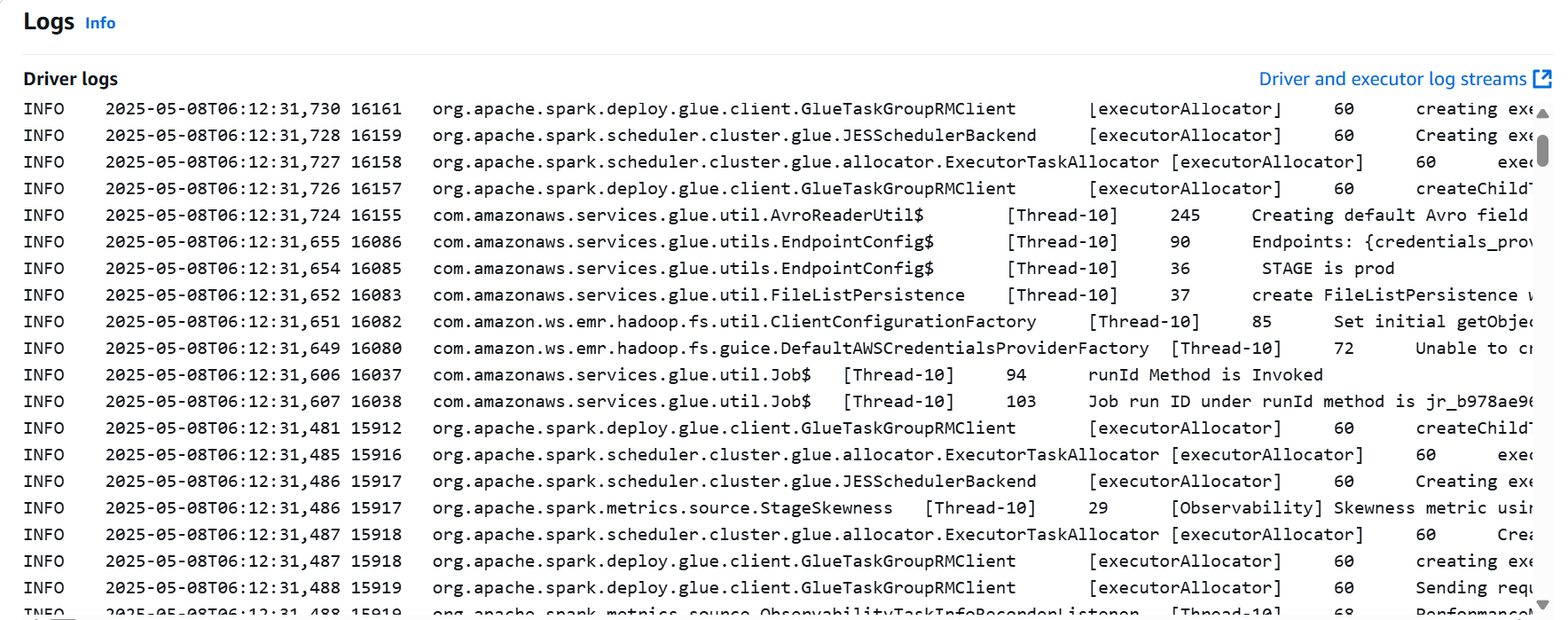
****

**Glue created the job and connected with the I AM role**

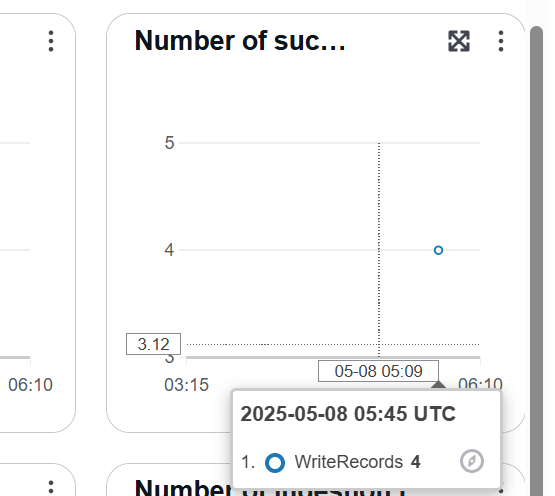
****

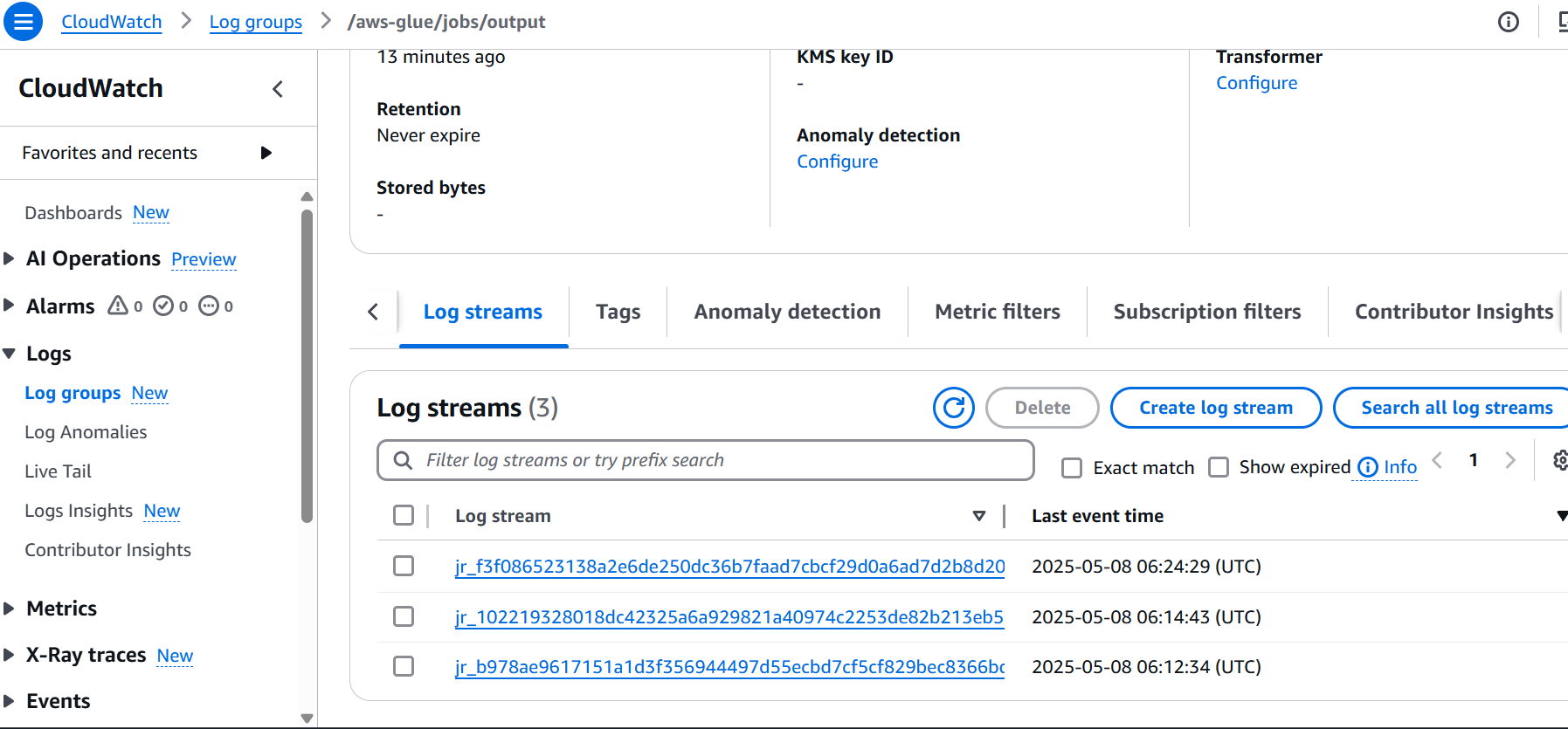
**Started job:**

****

****

**Connected to the time stream:**

****

**Connected to CloudWatch: Logs are Streamed**

**CONCLUSION:**

In this experiment, built a real-time streaming ETL pipeline using **AWS Kinesis**, **Glue**, and **Timestream** to handle and analyze IoT data efficiently. The workflow ingests data from IoT devices via Kinesis, processes it using AWS Glue, and stores it in Amazon Timestream for time-series analysis. This setup abstracts the complexities of handling continuous data streams and enables fast, scalable, and serverless data processing. It demonstrates the critical role of data engineering in turning raw IoT data into actionable insights with minimal manual effort.