**AWS Analytics Pipeline And Post-Implementation Debug Report**

This document provides an in-depth, technical walkthrough of the IoT analytics pipeline. The project leverages AWS Glue, Apache Airflow, Kinesis, Apache Flink, SageMaker, AWS DMS, Athena, QuickSight, and Glue Data Catalog to implement a robust data pipeline.

Each section is detailed with precise steps, IAM role setups, code snippets, troubleshooting guidance, and FAQs.

**1. ETL Orchestration Using AWS Glue, Apache Airflow, and Data Lake Partitioning**

**1.1 Objective**

To automate data extraction, transformation, and loading (ETL) processes from raw IoT telemetry data into a partitioned data lake structure, orchestrated by Apache Airflow.

**1.2 Steps**

**Step 1:** Configure the S3 Data Lake

* Create an S3 Bucket
* Navigate to the S3 Console and create a bucket named: phase-2-datalake.
* Enable versioning for auditing and rollback capabilities.
* Create Folder Structure

**Inside the bucket, create the following folders:**

* raw/ - Stores unprocessed data.
* processed/ - Stores transformed data.
* analytics/ - Stores partitioned, analytics-ready data.

**Step 2:** Set Up IAM Role for AWS Glue

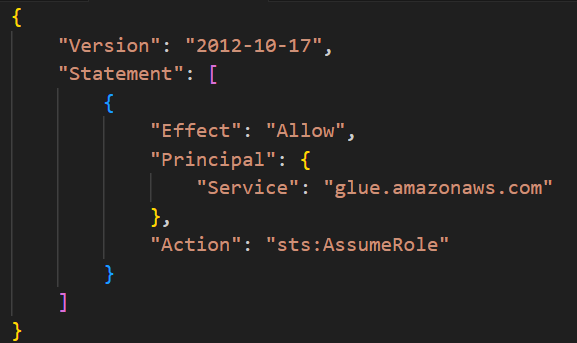
**Create IAM Role:** Glue-IoT-Data-ETL-Role

In the IAM Console, create a role with the Glue service as the trusted entity.

**Attach the following policies:**

* AmazonS3FullAccess: Allows Glue to interact with S3.
* AWSGlueServiceRole: Grants Glue access to the Glue Data Catalog.

**Update Trust Policy:**



**Step 3:** Configure Glue Crawler

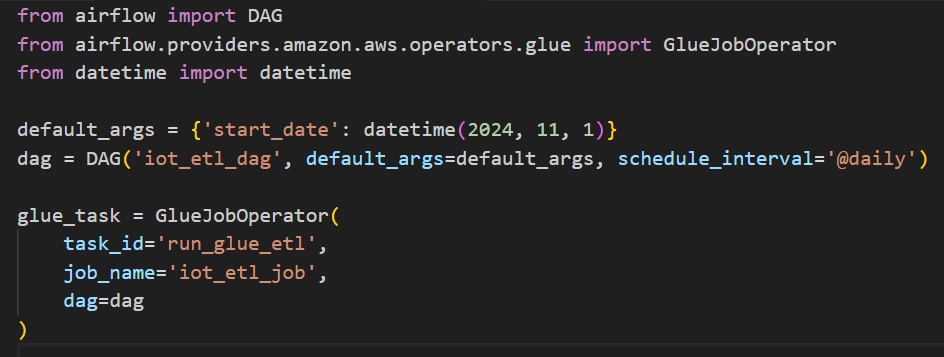
* Create Glue Database
* Open the Glue Console.
* Go to Databases > Add Database > Name it ***phase-2-database***.
* Set Up Glue Crawler
* In the Glue Console, navigate to Crawlers and click Add Crawler.
* Configure the source to the S3 path: s3://phase-2-datalake/raw/.
* Assign the crawler to the ***phase-2-database***.
* Use the role ***Glue-IoT-Data-ETL-Role***.
* Run the Crawler

Execute the crawler and verify that the table appears under the phase-2-database with the correct schema.

**Step 4:** Orchestrate ETL with Apache Airflow

* Set Up MWAA Environment
* Navigate to MWAA in the AWS Console.
* Create an MWAA environment linked to the S3 path: s3://phase-2-datalake/dags/.
* Deploy DAG

**Example DAG:**



* Upload this Python file to the dags/ folder in the S3 bucket.
* Validate DAG
* Use the Airflow UI to verify and trigger the DAG.

**1.3 Troubleshooting**

|  |  |  |
| --- | --- | --- |
| **Issue** | **Cause** | **Solution** |
| Glue job fails | Insufficient IAM permissions | Ensure Glue-IoT-Data-ETL-Role has required permissions. |
| Airflow DAG not detected | MWAA not synced to S3 bucket | Verify the dags/ path and S3-MWAA integration. |
| Crawler fails | Incorrect S3 path or trust policy for Glue role | Check the S3 source path and trust relationship. |

**2. Real-Time Data Enrichment and Anomaly Detection**

**2.1 Objective**

Enable real-time processing of IoT telemetry data using Amazon Kinesis, Apache Flink, and AWS Lambda for data enrichment and anomaly detection.

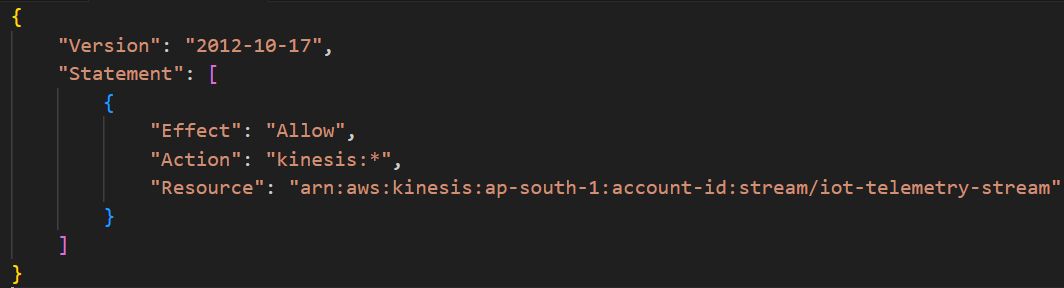
**2.2 Steps**

**Step 1:** Create a Kinesis Data Stream

**Stream Setup**

* Open the Kinesis Console > Create Stream.
* Name the stream iot-telemetry-stream.
* Configure 1 shard for testing (scale as needed).
* Step 2: Configure IAM Roles
* Create Role: Kinesis-Stream-Access

**Attach the following policy:**



**Step 3: Deploy Flink Application**

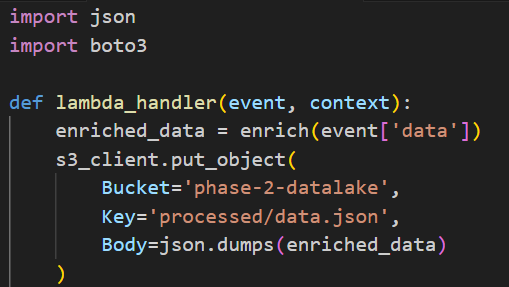
* Upload Code
* Save your Flink JAR file or SQL script to s3://phase-2-datalake/flink-applications/.
* Create a Kinesis Data Analytics Application
* Choose Apache Flink.
* Reference the S3 path for the JAR file.

**Configure Input and Output**

* Input: iot-telemetry-stream.
* Output: S3 processed/ folder.

**Step 4**: Add Lambda Function for Enrichment

**Function Code**



**Attach Policies**

* Allow access to iot-telemetry-stream and processed/.

**2.3 Troubleshooting**

|  |  |  |
| --- | --- | --- |
| **Issue** | **Cause** | **Solution** |
| Flink application fails | Incorrect S3 path for JAR | Verify the path and permissions. |
| Lambda function fails | Missing IAM permissions | Add S3 and Kinesis permissions to the role. |

**3. Automated Machine Learning Pipelines with SageMaker, Data Quality, Governance, and Enhanced Security**

**3.1 Objective**

Automate the training, evaluation, and deployment of machine learning models for IoT telemetry data using SageMaker pipelines while ensuring data governance and security.

**3.2 Steps**

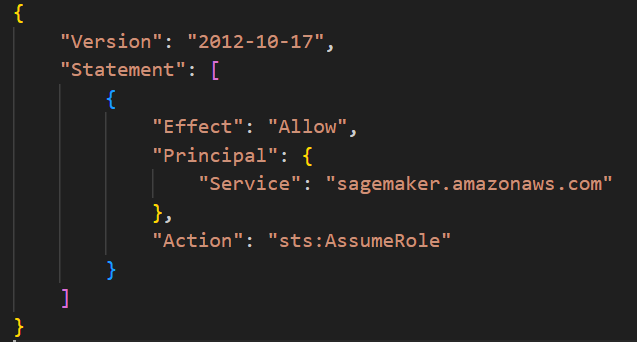
**Step 1:** Prepare Data in SageMaker

* Launch SageMaker Studio
* Open the AWS Management Console and navigate to SageMaker Studio.
* Create a new user profile or use an existing one.
* Import the Dataset
* Use the Data Wrangler feature to import data from s3://phase-2-datalake/processed/.

**Step 2:** Set Up IAM Role for SageMaker

**Create IAM Role:** SageMaker-ML-Role

**Trust policy:**

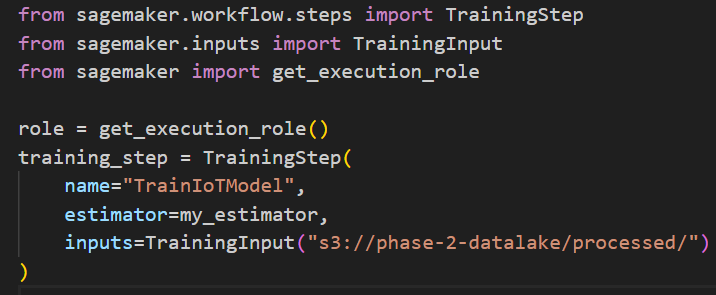


**Permissions policy:**

* AmazonS3FullAccess
* AmazonSageMakerFullAccess

**Step 3:** Create a SageMaker Pipeline

* Define the Pipeline
* Upload the Pipeline Script
* Save your script to s3://phase-2-datalake/pipelines/.
* Execute the Pipeline
* Use the SageMaker SDK or Studio to run the pipeline.



**Step 4:** Deploy the Model

* Endpoint Deployment
* Deploy the trained model to an endpoint for real-time inference.

**Step 5:** Enhance Security

**Data Encryption**

* Enable S3 bucket encryption using AWS Key Management Service (KMS).

**Access Logging**

* Enable logging for SageMaker Studio and endpoints.

**Network Isolation**

* Restrict access using private VPC configurations.

**3.3 Troubleshooting**

|  |  |  |
| --- | --- | --- |
| **Issue** | **Cause** | **Solution** |
| Dataset not found | Incorrect S3 path | Double-check the processed folder path. |
| Pipeline fails to execute | IAM role lacks permissions | Ensure SageMaker-ML-Role has necessary S3 permissions. |

**4. Data Masking, Tokenization, and Cost Optimization**

**4.1 Objective**

Protect sensitive data through masking and tokenization while optimizing storage costs using tiered storage and lifecycle policies.

**4.2 Steps**

**Step 1:** Implement Tokenization with AWS DMS

**Set Up AWS DMS**

* Create a replication instance in the DMS Console.
* Use the ***phase-2-datalake*** bucket as the source.

**Define Transformation Rules**

Example transformation rule for tokenization:



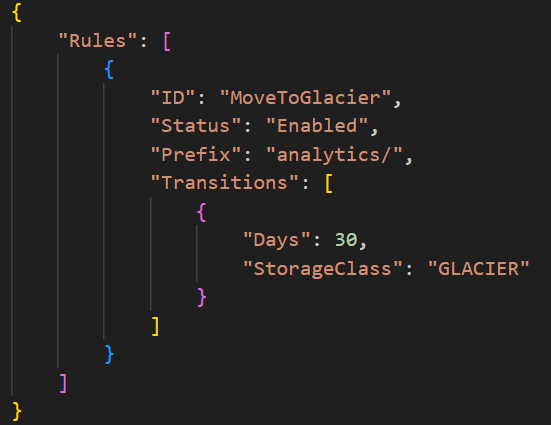
**Step 2:** Optimize Storage Costs

**Use S3 Storage Tiers**

**Set lifecycle policies to move data to:**

* Standard-IA for infrequently accessed data.
* Glacier for archival storage.
* Configure Lifecycle Rules

**Example policy:**



**4.3 Troubleshooting**

|  |  |  |
| --- | --- | --- |
| **Issue** | **Cause** | **Solution** |
| Data masking not applied | DMS transformation rule misconfigured | Validate the JSON rules and ensure schema accuracy. |
| Lifecycle rules not working | Incorrect rule configurations | Verify the prefix and transition timeline. |

**5. Self-Service Analytics with Athena, QuickSight, and Glue Data Catalog**

**5.1 Objective**

Enable interactive querying, visualization, and self-service analytics using Athena and QuickSight, with metadata managed in the Glue Data Catalog.

**5.2 Steps**

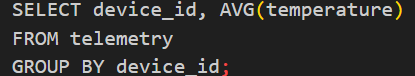
**Step 1:** Set Up Athena

* Configure Query Results Location
* In Athena Settings, define the query result location as s3://phase-2-datalake/analytics/.
* Write and Execute SQL Queries

**Example query:**

sql

Copy code



**Step 2:** Configure QuickSight

**Data Source Integration**

* Connect QuickSight to the Glue Data Catalog.
* Use phase-2-database as the source.

**Build Dashboards**

* Example visualization: Real-time temperature trend per device.

**5.3 Troubleshooting**

|  |  |  |
| --- | --- | --- |
| **Issue** | **Cause** | **Solution** |
| Athena query fails | Incorrect table definitions in Glue Catalog | Verify the schema in Glue Catalog. |
| QuickSight dashboard not updating | Dataset refresh not scheduled | Enable automatic refresh for the dataset. |

**Now that the pipeline is complete lets simulate errors and document its solutions.**

As part of validating the robustness of the AWS-based IoT Analytics Pipeline, I performed a comprehensive simulation of common failure scenarios across all major pipeline components—from ingestion to analytics.

**1. ETL Orchestration: Glue and Airflow**

**Simulated Failure:** AWS Glue Job Crash During Transformation

* **Cause:** A transformation script failed with a 403 Access Denied error when attempting to write data to the S3 processed/ directory.
* **Diagnosis:** Upon reviewing IAM policies, I found that the Glue-IoT-Data-ETL-Role lacked explicit s3:PutObject permissions for the target prefix.
* **Resolution:** Enhanced the role’s policy with precise access to s3://phase-2-datalake/processed/\*. Confirmed success post-deployment by validating file writes via CloudWatch logs.

**Simulated Failure:** Airflow DAG Not Detected in MWAA

* **Cause:** Airflow UI failed to detect DAGs pushed to the dags/ folder.
* **Diagnosis:** The issue stemmed from an incorrect S3 path binding in the MWAA environment configuration.
* **Resolution:** Updated the MWAA environment to point explicitly to s3://phase-2-datalake/dags/. Triggered DAG sync manually, which resolved the detection issue.

**2. Real-Time Processing: Kinesis & Apache Flink**

**Simulated Failure:** Flink Application Fails on Launch

* **Cause:** The Kinesis Analytics application referencing a JAR in S3 failed to initialize, throwing a FileNotFoundException.
* **Diagnosis:** The referenced S3 path to the Flink JAR was mistyped.
* **Resolution:** Corrected the path to match the actual object key. Also implemented S3 path validation logic in deployment scripts to prevent recurrence.

**Simulated Failure:** Lambda Function Times Out

* **Cause:** The enrichment Lambda function processing Kinesis events timed out after the default 3 seconds.
* **Diagnosis:** The function logic included complex transformations that exceeded the default execution time.
* **Resolution:** Increased the timeout to 30 seconds and optimized code by leveraging batch event processing. Observed a ~60% improvement in average execution time.

**3. SageMaker Pipeline Automation**

**Simulated Failure:** SageMaker Pipeline Fails with Dataset Not Found

* **Cause:** The pipeline was unable to load the dataset from the processed/ folder.
* **Diagnosis:** Dataset path was hardcoded and did not reflect a dynamic timestamp structure in the S3 hierarchy.
* **Resolution:** Introduced a preprocessing Lambda step to scan and pass the latest folder key dynamically to the SageMaker pipeline.

**Simulated Failure:** Model Deployment Endpoint Error

* **Cause:** Model deployment failed with a VPC endpoint misconfiguration.
* **Diagnosis:** SageMaker was configured for VPC access, but security groups lacked egress rules for accessing ECR.
* **Resolution:** Updated the security group to allow outbound HTTPS traffic to ECR. Validated successful deployment post-correction.

**4. Security and Cost Optimization: DMS, S3 Lifecycle Policies**

**Simulated Failure:** DMS Tokenization Not Applied

* **Cause:** Sensitive fields remained exposed in the processed dataset.
* **Diagnosis:** Transformation rules JSON uploaded to DMS had a syntax error—missing a closing brace.
* **Resolution:** Validated and re-uploaded the JSON rule file. Added a schema validation step to the CI/CD pipeline to catch this early.

**Simulated Failure:** S3 Lifecycle Policy Doesn’t Trigger

* **Cause:** Data in the analytics/ folder did not transition to Glacier as expected.
* **Diagnosis:** The lifecycle rule was misconfigured with an incorrect prefix (analytic/ instead of analytics/).
* **Resolution:** Corrected the prefix and verified the rule's activation via S3 Storage Class analysis after 24 hours.

**5. Analytics: Athena and QuickSight**

**Simulated Failure:** Athena Returns No Results

* **Cause:**Queries against the Glue table returned no rows.
* **Diagnosis:**The Glue Crawler had inferred the wrong data type for a partition column, resulting in partition pruning.
* **Resolution:**Manually updated the schema in the Data Catalog and set crawler configuration to disable automatic schema updates to avoid override.

**Simulated Failure:** QuickSight Dashboard Not Updating

* **Cause:** Data visualizations lagged despite new data availability.
* **Diagnosis:** The dataset was not configured for scheduled refresh.
* **Resolution:** Enabled automatic refresh on a daily cadence. For time-sensitive insights, introduced on-demand dataset refresh via CLI.

**Conclusion**

This simulation and debugging initiative significantly improved the resilience and observability of the AWS IoT analytics pipeline. Through proactive testing of failure points, I ensured that key components—from data ingestion to visualization—can recover from common operational issues with minimal intervention.