TECHNICAL DESIGN DOCUMENT

AWS Automated Data Pipeline

Preparation Date: 10-12-23

Prepared By: Ashutosh Kumar Maurya, Brhat B G

Epsilon

CONTENT

Revision History

Introduction:

The purpose of this document is to outline the technical specifications, design, configurations, and approaches for implementing an automated data pipeline POC involving data processing and analysis using AWS services and Google BigQuery. This POC aims to explore and demonstrate the capabilities of integrating AWS Lambda, Amazon EMR, AWS Glue, Amazon S3, and Google BigQuery to build a robust and scalable data processing and analysis solution.

This document will detail two primary approaches for the POC:

The first approach focuses on leveraging S3, AWS Lambda, Amazon EMR, and Google BigQuery. It outlines the process of using AWS Lambda for triggering data processing jobs, utilizing Amazon EMR for large-scale data processing tasks, and integrating Google BigQuery for advanced data analysis and insights.

The second approach explores the use of AWS Glue in conjunction with Amazon S3 and AWS Lambda. This method emphasizes the utilization of AWS Glue for data preparation and load operations, taking advantage of Amazon S3 for data storage, and employing AWS Lambda for orchestration and automation of data workflows.

Both approaches are designed to showcase the interoperability and strengths of using AWS services in combination with Google BigQuery for comprehensive data processing and analysis solutions. The document will provide detailed configurations, architectural designs, and step-by-step implementation guides for each approach, ensuring a clear path towards achieving the POC objectives.

Document Purpose & Overview:

This document is designed to offer a detailed guide to the architecture of the proposed POC, elaborating on the technical specifications, configurations, tools, and methodologies involved. It aims to furnish a comprehensive understanding of the step-by-step process for implementing a scalable and efficient data processing and analysis solution using AWS services (Lambda, S3,

EMR, Glue, CloudWatch Logs) in combination with Google BigQuery. Additionally, the document will address potential challenges and their solutions, ensuring a smooth execution of the POC.

Assumptions:

1. Availability of AWS and Google Cloud Platform (GCP) accounts for accessing and utilizing the services required for the POC.
2. Appropriate permissions granted for configuring and managing AWS services (AWS Lambda, Amazon S3, Amazon EMR, AWS Glue, AWS CloudWatch Logs) and Google BigQuery, facilitating seamless integration and data workflow.
3. Provision of sample data in CSV format to serve as the basis for testing and demonstrating the implementation process and data analysis capabilities.
4. A foundational understanding of AWS service functionalities and the mechanisms for data transfer between these services and Google BigQuery, enabling effective utilization of the proposed architecture.

Technical Specifications/Requirements:

1. **Data Source:**
    - CSV files uploaded to an S3 bucket.
2. **Processing Requirements:**
    - PySpark-based processing to integrate incoming file data with existing tables.
    - Dynamic EMR cluster scaling based on file size for further data processing.
3. **Data Destination:**
    - Google BigQuery for data storage and analysis.
4. **Roles and Policies:**
    - AWS IAM Roles: Specify roles required for each AWS service (Lambda, S3, EMR, Glue, CloudWatch Logs) involved in the POC, detailing their permissions for accessing other AWS services and resources.
    - Policies: Outline the policies attached to each role, defining the allowed and denied actions within specific AWS services. Include examples such as the

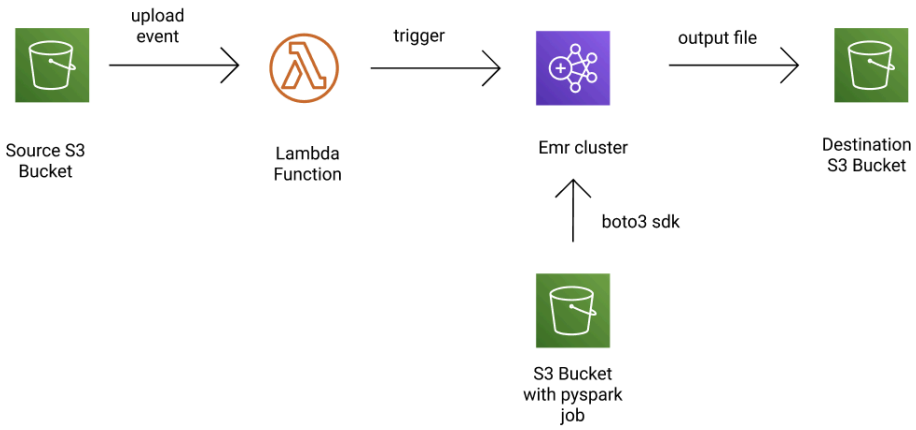
Lambda execution role with permissions to access S3 buckets, invoke EMR clusters, manage Glue jobs, and write logs to CloudWatch.

- Cross-service Permissions: Describe any necessary permissions for services to interact with Google BigQuery, including AWS to Google Cloud interconnectivity setups if applicable.

I. Approaches/Solution:

Approach 1:

1. Use AWS Lambda triggered by S3 events to process incoming files with PySpark.
2. Implement dynamic EMR cluster scaling based on calculated file size.
3. Utilize EMR for further complex data transformations and analysis.
4. Export processed data to Google BigQuery for analysis.



Step-by-Step Implementation:

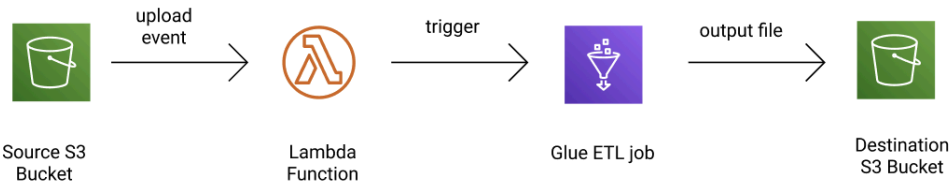
1. **Data Upload:** Data files are uploaded to the source Amazon S3 bucket. This triggers an event notification.
2. **Event Triggering:** The S3 upload event triggers an AWS Lambda function. The Lambda function is configured to respond to 'put' events in the source S3 bucket.

3. **Lambda Function Execution:** The triggered Lambda function executes the logic defined within it, which typically includes pre-processing tasks or the setup and configuration for the next steps of the data processing workflow.
4. **Job Submission:** The Lambda function submits a job to an Amazon EMR cluster. This job could be a PySpark job if the Lambda function fetches the PySpark script from another S3 bucket (as indicated by the 'S3 Bucket with pyspark job').
5. **EMR Processing:** The EMR cluster processes the data using the submitted PySpark job. EMR can dynamically scale based on the file size or the computing resources needed for processing the data efficiently.
6. **Data Output:** Upon completion of the data processing job, the results are saved back to a destination S3 bucket, which could be used for further analysis or as an input to another system.
7. **Further Analysis:** The processed data in the destination S3 bucket can then be transferred to Google BigQuery for storage and advanced analysis.

This example provides a general idea of how the POC could be executed using PySpark. In practice, we need to adapt the code according to our specific environment, actual data sources, and requirements.

Approach 2:

This approach leverages AWS services to create an event-driven ETL (Extract, Transform, Load) pipeline. It utilizes AWS Lambda to trigger AWS Glue jobs upon file uploads to S3, processes incoming CSV files, and stores the transformed data in a designated S3 bucket for downstream usage or further analysis.



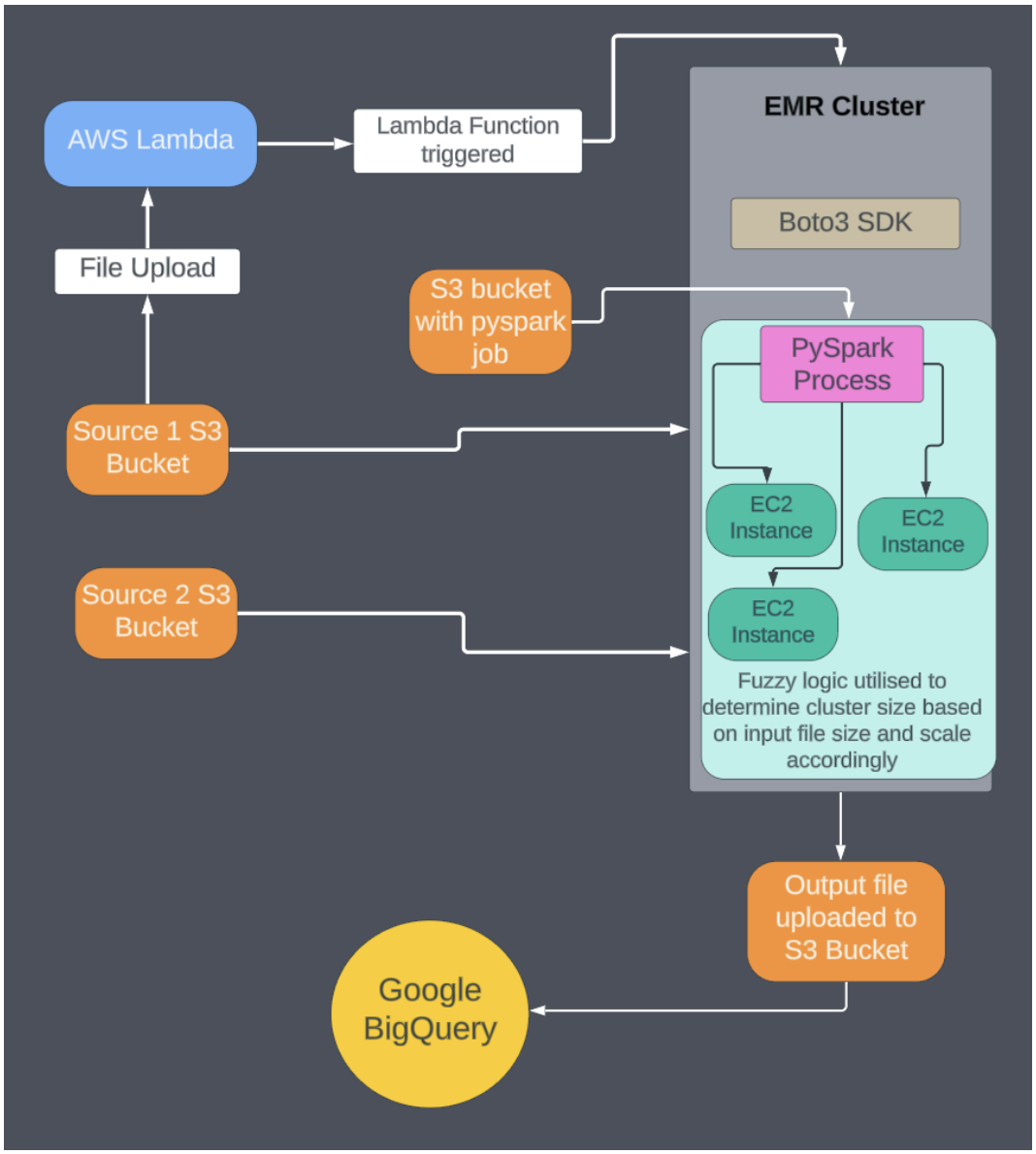
Step-by-Step Implementation:

1. **AWS Infrastructure Setup:**
    - **Source S3 Bucket Configuration:** Establish an S3 bucket to collect incoming CSV files, configuring event notifications to trigger a Lambda function.
    - **Destination S3 Bucket Setup:** Prepare another S3 bucket dedicated to storing the processed data.
2. **Lambda Function for Glue Job Triggering:**
    - **Lambda Function Creation:** Implement an AWS Lambda function that will be activated by 'put' or 'post' events in the source S3 bucket.
    - **Glue Job Invocation:** Configure the Lambda function to call an AWS Glue job to handle the CSV file received in the source S3 bucket.
3. **Glue Job Execution:**
    - **Glue ETL Job Configuration:** Design an AWS Glue ETL job that picks up the new CSV file from the source S3 bucket.
    - **ETL Logic Definition:** Within the Glue job, specify the transformation logic necessary to convert the data into the desired format or structure.
    - **Transformed Data Storage:** Direct the Glue job to deposit the resulting transformed data into the destination S3 bucket.
4. **Monitoring and Logging:**
    - Implement monitoring and logging to track the execution status of Lambda functions and Glue jobs.
    - Ensure that there are mechanisms in place for alerting in case of failures or anomalies.

By following this structure, you can implement a reliable and scalable event-driven ETL pipeline using AWS services that efficiently manage the processing of CSV files. This approach ensures that data is ready for subsequent stages of data analysis or machine learning pipelines.

II. Low-Level Design:

Approach 1:



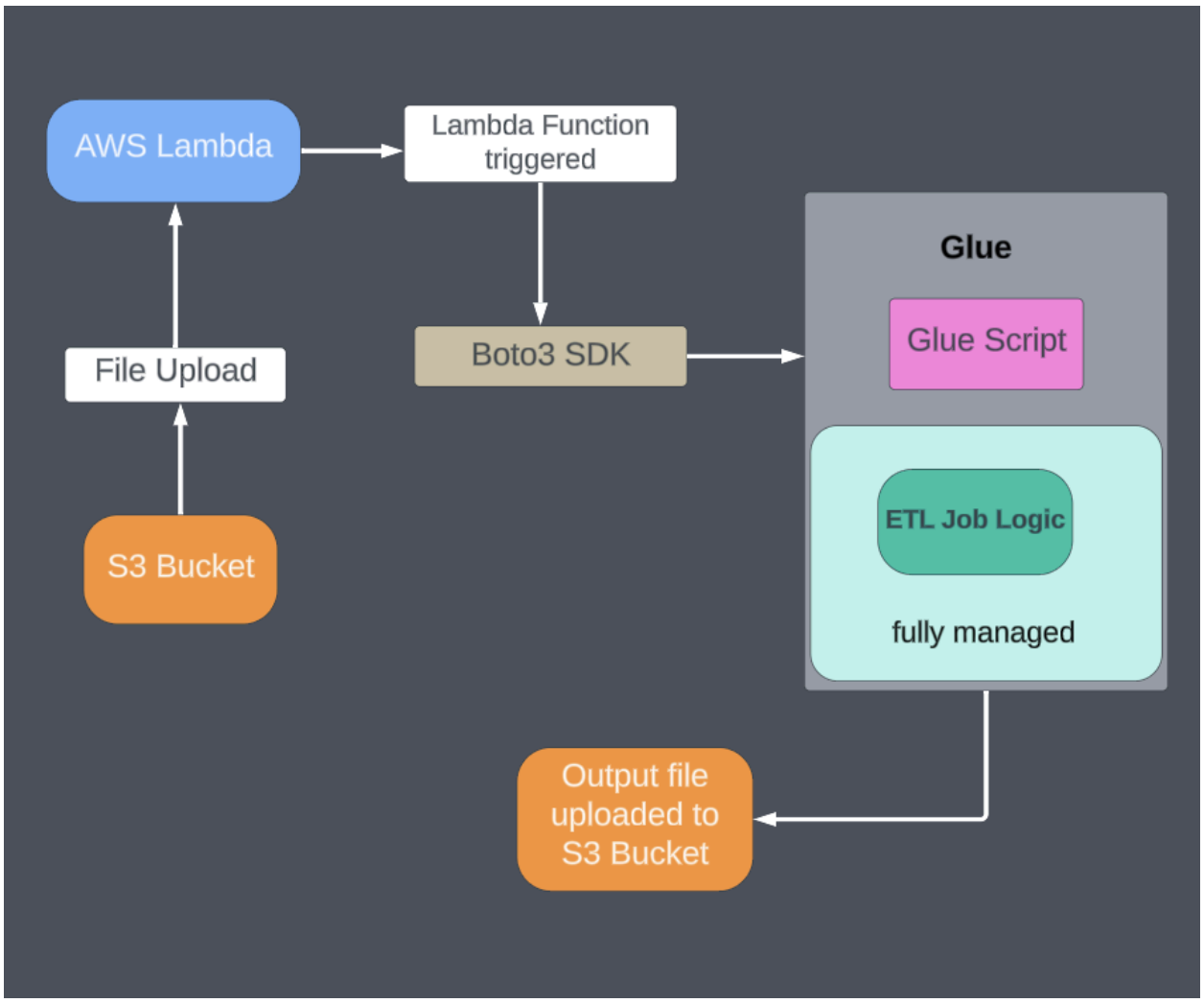
1. **Source 1 S3 Bucket:**
    - Set up this bucket to store incoming data files. Configure event notifications so that when a file is uploaded, it triggers the AWS Lambda function.
2. **AWS Lambda Function:**

- This function is invoked by the event triggered through file uploads to the Source 1 S3 Bucket.
- The Lambda function contains the implementation of fuzzy logic. This logic determines the appropriate size and configuration of the EMR cluster based on the size or content of the uploaded file0.
- Use AWS SDK (Boto3) within the Lambda function to programmatically interact with other AWS services, such as initiating an EMR cluster.

3. **Source 2 S3 Bucket:**
   - Contains pre-existing data that will be used in conjunction with the new data from Source 1 during the processing job in EMR.
4. **EMR Cluster:**
   - Once the Lambda function determines the configuration, it uses Boto3 to set up the EMR cluster or add a step to an existing cluster.
   - The cluster's task is to run a PySpark job that will process the data. The specifics of the task will depend on the requirements of the job itself, including any transformations or analyses to be performed.
5. **PySpark Job:**
   - The EMR cluster executes the PySpark job, which should be written to:
     i. Access and read data from Source 1 S3 Bucket and Source 2 S3 Bucket.
     ii. Perform necessary data processing, combining the new data with the existing data, and carry out any required transformations or analytics.
     iii. Output the result of the data processing to a specified destination S3 bucket.
6. **Destination S3 Bucket:**
   - After the PySpark job is completed, the results (output files) are stored in this bucket. This bucket acts as a bridge between the AWS environment and Google BigQuery.
   - Ensure that permissions are set so that the EMR cluster can write to this bucket and that any subsequent services (like a data transfer service to Google BigQuery) have read access.
7. **Google BigQuery:**

- Utilize a service or process (which may be an additional AWS Lambda function or AWS Glue) to transfer the output data from the destination S3 bucket to Google BigQuery.
- In BigQuery, the data can then be used for further analysis, querying, and reporting.

Approach 2:



1. **S3 Bucket (Source):**
   - Configure the S3 bucket to store incoming CSV data files.

- Set up event notifications so that when a CSV file is uploaded, it automatically triggers the configured AWS Lambda function.

2. **AWS Lambda Function:**
   - This function is activated upon the event of a CSV file upload to the S3 bucket.
   - The function uses the Boto3 SDK to call the AWS Glue service, triggering an ETL job.
3. **AWS Glue ETL Job:**
   - Once invoked, the AWS Glue job begins executing the predefined ETL script.
   - This script is responsible for the Extract, Transform, and Load process. It reads the input CSV file from the S3 bucket, applies transformations as per the business logic, and writes the transformed data back to another S3 bucket (destination).
4. **S3 Bucket (Destination):**
   - This is where the output of the Glue ETL job is stored.
   - The transformed CSV file is written to this bucket, ready for further use or processing.

III. Application Configurations:

Approach 1:

1. **AWS Configurations:**
   - Set up S3 bucket event notifications triggering Lambda.
   - Configure Lambda with necessary permissions to access S3, EMR, and other services.
2. **EMR Configurations:**
   - Define Spark jobs or steps for data processing on EMR.
   - Configure EMR with proper security settings and instance types.
3. **Google BigQuery Configurations:**
   - Set up dataset and tables for loading processed data from S3.

Approach 2:

1. **AWS Configurations:**
   - Ensure the Lambda function has the necessary permissions to access both S3 buckets and to trigger Glue jobs.

- Set up event notifications in the source S3 bucket to trigger the Lambda function upon file arrivals.

2. **Glue Job Configurations:**
   - Configure the Glue job with the necessary ETL scripts and data mappings.
   - Ensure the Glue job has permissions to read from the source S3 bucket and write to the destination S3 bucket.

IV. Tools and Technologies:

Approach 1:

1. **AWS Services:**
   - AWS Lambda, S3, EMR (Elastic MapReduce).
2. **Data Processing Tools:**
   - PySpark for data processing within EMR.
3. **Data Storage and Analytics:**
   - Google BigQuery for storing and analyzing processed data.

Approach 2:

1. **AWS Services:**
   - AWS Lambda, S3, AWS Glue.
2. **Data Processing:**
   - AWS Glue for ETL operations.

V. Issues/Resolution:

1. **Performance Optimization:**
   - **Issue:** Potential performance bottlenecks in EMR processing.
   - **Resolution:** Optimize Spark jobs, instance types, and partitioning strategies for efficient processing.
2. **Data Transfer:**
   - **Issue:** Challenges in efficient data transfer between EMR and Google BigQuery.
   - **Resolution:** Optimize data export strategies, explore intermediate storage options for seamless transfer.
3. **ETL Job Optimization:**

- **Issue:** Potential inefficiencies or bottlenecks in the Glue ETL process.
- **Resolution:** Optimize the ETL scripts, monitor job performance metrics, and adjust resources as necessary.

4. **Data Consistency and Integrity:**
   - **Issue:** Ensuring data consistency and integrity throughout the ETL process.
   - **Resolution:** Implement data validation checks within the Glue job, and handle any anomalies or errors gracefully.

Glossary of Terms:

Term / Acronym	Definition
Amazon S3 (Simple Storage Service)	A scalable cloud storage service provided by AWS that allows for the storage and retrieval of any amount of data from anywhere on the web.
Pandas	An open-source data analysis and manipulation tool, built on top of the Python programming language.
PySpark	The Python API for Apache Spark, which allows for easy integration of Python with Spark's big data processing capabilities.
Spark Framework	An open-source, distributed computing system that provides a simple and comprehensive programming model for big data processing.
EMR	Amazon EMR is the industry-leading cloud big data solution for petabyte-scale data processing, interactive analytics, and machine learning using open-source frameworks such as Apache Spark, Apache Hive, and Presto
EC2	Amazon EC2 is a web service that provides sizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.
Glue	AWS Glue is a serverless data integration service that makes it easier to discover, prepare, move, and integrate data from multiple sources for analytics, machine learning (ML), and application development.

Lambda	Compute service that runs your code in response to events and automatically manages the compute resources.

Appendix:

- https://docs.aws.amazon.com/lambda/
- https://docs.aws.amazon.com/ec2/
- https://docs.aws.amazon.com/s3/
- https://docs.aws.amazon.com/emr/
- https://docs.aws.amazon.com/glue/