AWS Data Lake and Analytics Platform

Overview

I designed and implemented a data lake solution on AWS, enabling the organization to store, process, and analyze large volumes of structured and unstructured data.

Key Components

- S3: For data storage
- Glue: For ETL jobs and data catalog
- Athena: For SQL queries on S3 data
- QuickSight: For data visualization
- Lambda: For data processing and transformations

Implementation Highlights

Glue ETL Job (PySpark)

python
Copy

```python
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

glueContext = GlueContext(SparkContext.getOrCreate())
job = Job(glueContext)

# Read data from S3
datasource0 = glueContext.create_dynamic_frame.from_catalog(
    database = "my_data_lake_db",
    table_name = "raw_data"
)

# Apply transformations
applymapping1 = ApplyMapping.apply(
    frame = datasource0,
    mappings = [
        ("col1", "string", "transformed_col1", "string"),
        ("col2", "long", "transformed_col2", "int"),
        ("col3", "string", "transformed_col3", "string")
    ]
)

# Write transformed data back to S3
glueContext.write_dynamic_frame.from_options(
    frame = applymapping1,
    connection_type = "s3",
    connection_options = {"path": "s3://my-data-lake-bucket/transformed/"},
    format = "parquet"
)

job.commit()
```

Lambda Function for Data Processing

python
Copy

```
import json
import boto3

s3 = boto3.client('s3')
athena = boto3.client('athena')

def lambda_handler(event, context):
    # Get the object from the event
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']

    # Process the data (example: count words)
    response = s3.get_object(Bucket=bucket, Key=key)
    content = response['Body'].read().decode('utf-8')
    word_count = len(content.split())

    # Write results back to S3
    result = json.dumps({'file': key, 'word_count': word_count})
    s3.put_object(Bucket=bucket, Key=f'processed/{key}_count.json', Body=result)

    # Trigger Athena query
    query = f"ALTER TABLE processed_data ADD PARTITION (dt='{key.split('/')[0]}')"
    athena.start_query_execution(
        QueryString=query,
        QueryExecutionContext={'Database': 'my_data_lake_db'},
        ResultConfiguration={'OutputLocation': 's3://my-athena-results/'}
    )

    return {
        'statusCode': 200,
        'body': json.dumps('Processing complete!')
    }
```

Challenges and Solutions

- Data partitioning: Implemented a partitioning strategy in S3 for improved query performance.
- Cost optimization: Set up S3 lifecycle policies and used Athena workgroups to manage costs.

Outcome

This project demonstrated my ability to architect and implement big data solutions on AWS, showcasing skills in data engineering and analytics.