LEVEL1_Solution Architecture_Report

BTTF Data Engineer Assignment

Objective

Design a secure, modular, and scalable data platform architecture for **BTTF Logistics** as per the assignment brief. The architecture ensures:

- Seamless ingestion from various internal and external sources
- A layered, organized data lakehouse design
- Distributed ETL processing using PySpark
- A query and reporting layer ready for stakeholder analytics
- Integration of observability and scalability principles

NOTE: This report aligns with Deliverable #1 from the assignment document.

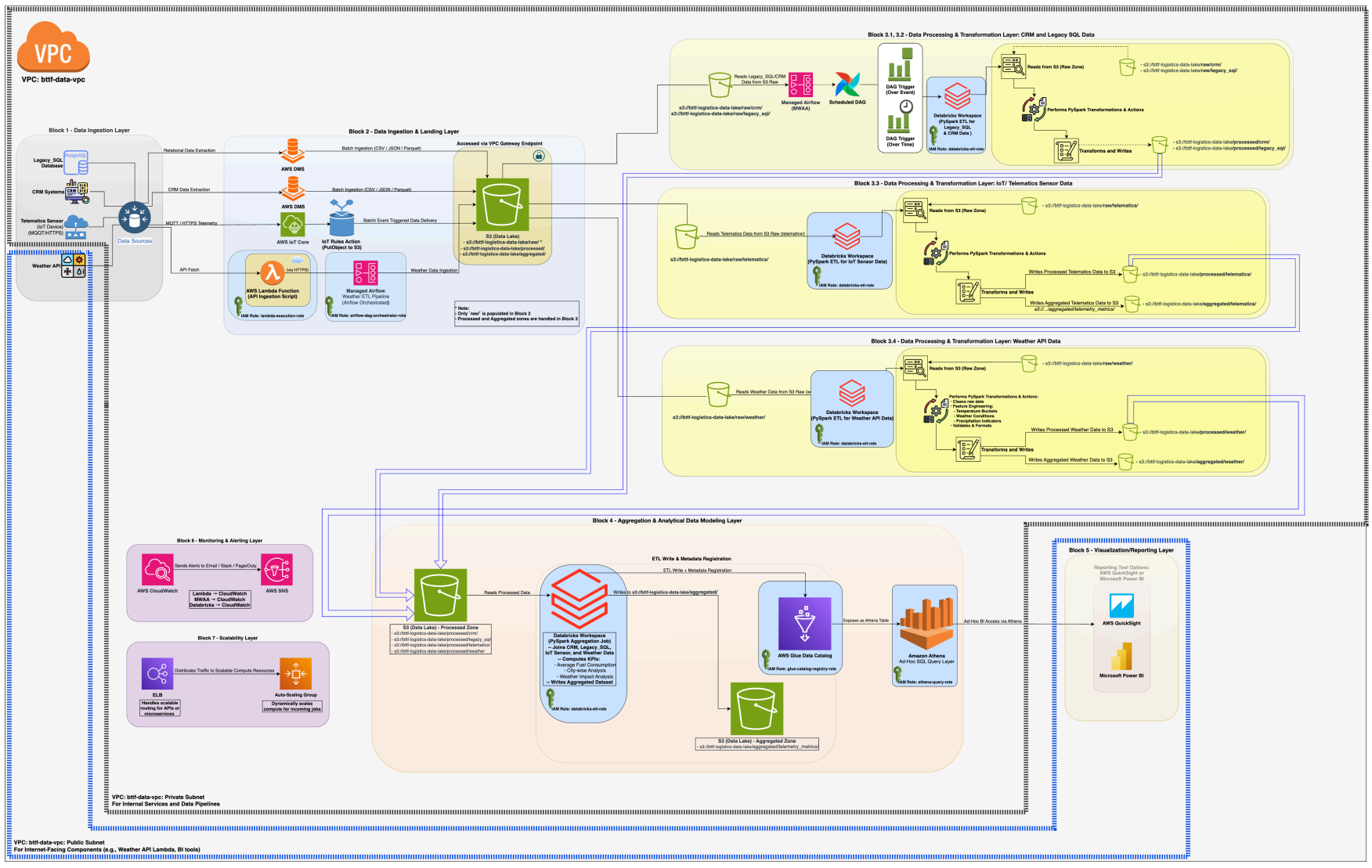
Problem Statement Recap

- Data is scattered across CRM, legacy SQL databases, and sensor-generated telemetry.
- Fleet generates GPS, load, maintenance, and other vehicle data but it's currently unprocessed.
- Future integration of external weather data is mandatory.
- Business goal: democratize data access, improve analytical capabilities, and optimize fuel consumption analysis.

Architecture Summary Table

Layer	Tools / Services Involved
Ingestion	AWS DMS, AWS IoT Core, Lambda, MWAA
Storage	Amazon S3 with `/raw/`, `/processed/`, `/aggregated/` zones
ETL Processing	Databricks PySpark Notebooks orchestrated by Airflow
Data Modeling	PostgreSQL (local testing) + Conceptual Redshift extension
Query Layer	AWS Glue Catalog, Amazon Athena
BI / Reporting	Power BI, QuickSight, Tableau
Monitoring	AWS CloudWatch + SNS Alerts
Scalability	ELB, Auto Scaling Group (for compute workloads)
Security	IAM Roles, VPC Subnets, Gateway Endpoints

Project-Proposed Architecture:



Block 1: Data Ingestion Sources

Description

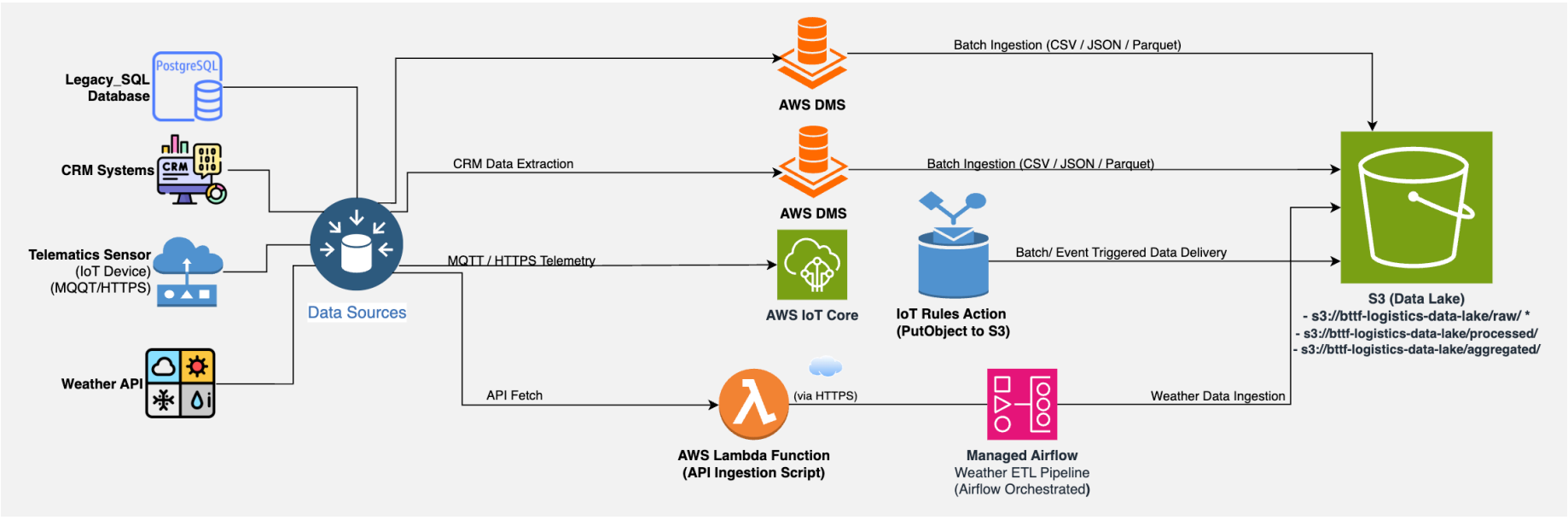
Handles structured (CRM, SQL) and semi-structured (IoT, Weather API) ingestion. Each source writes to S3 in the `raw/` zone.

Tools Used

- **CRM & SQL**: AWS DMS for batch migration
- **IoT Sensors**: AWS IoT Core → IoT Rule Action → S3 (PutObject)
- **Weather API**: Python script triggered by Lambda (via MWAA DAG)

Flow:

```
CRM / SQL          → AWS DMS     → S3 (raw/crm/, raw/legacy_sql/)
IoT Telematics     → IoT Core    → IoT Rule Action → S3 (raw/telematics/)
Weather API        → Lambda      → S3 (raw/weather/)
```



Block 2: S3 Landing – Raw Zone

Description

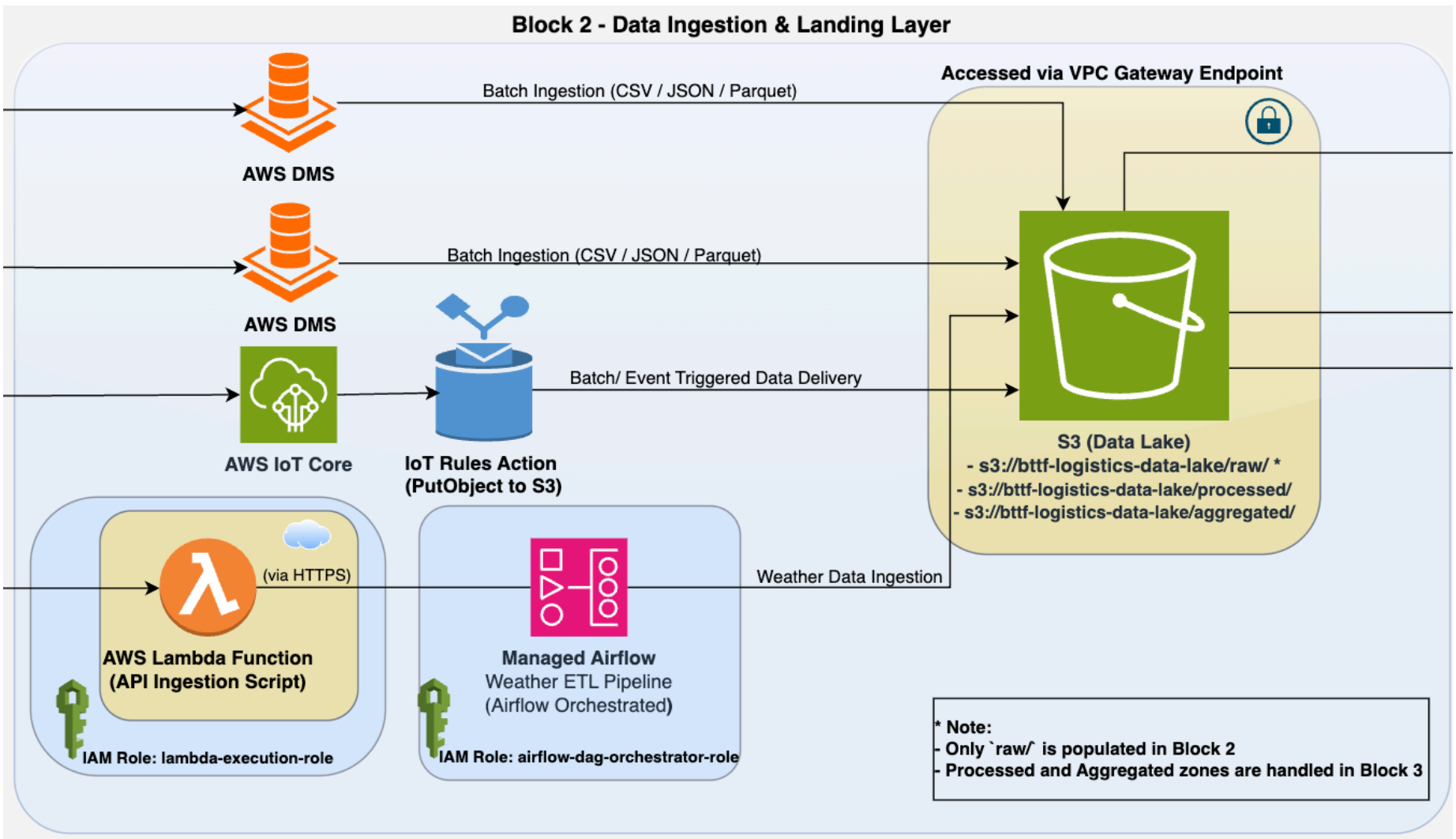
Serves as the single-entry point for all ingested data.

Structure:

```
s3://bttf-logistics-data-lake/
|── raw/
|   ├── crm/
|   ├── legacy_sql/
|   ├── telematics/
|   └── weather/
```

Notes:

- Private subnet access enforced (VPC Endpoint)
- IAM roles: `lambda-execution-role`, `dms-s3-write-role`, `iot-rule-role`



Block 3: Data Processing (ETL Layer)

🔍 Description

Data from the raw zone is cleaned, validated, enriched, and written to `processed/` and `aggregated/` zones.

Tools:

- **Databricks** (PySpark Notebooks)
- **MWAA (Airflow)**: orchestrates DAGs for ETL pipelines

Pipelines Overview:

3.1 CRM ETL

```
Read: s3://.../raw/crm/
Transform: Clean customer records, drop nulls
Write: s3://.../processed/crm/
```

3.2 Legacy SQL ETL

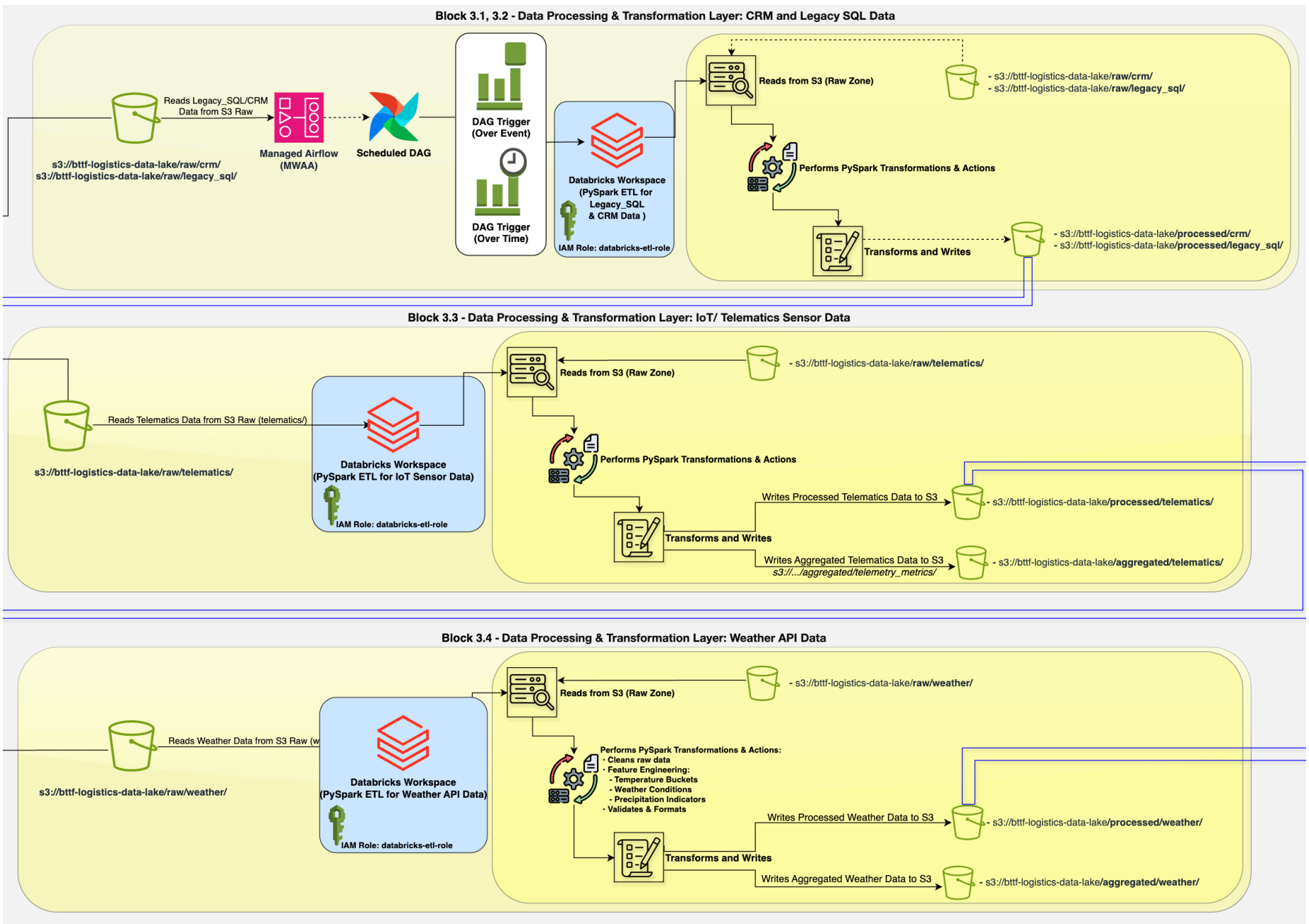
```
Read: s3://.../raw/legacy_sql/
Transform: Normalize columns, unify timestamp formats
Write: s3://.../processed/legacy_sql/
```

3.3 Telematics ETL

```
Read: s3://.../raw/telematics/
Transform: Extract GPS, fuel metrics, compute load features
Write: processed/ and aggregated/
```

3.4 Weather ETL

```
Read: s3://.../raw/weather/
Transform: Feature engineering, temp bucketing, missing handling
Write: s3://.../processed/weather/
```



Block 4: Aggregation Layer

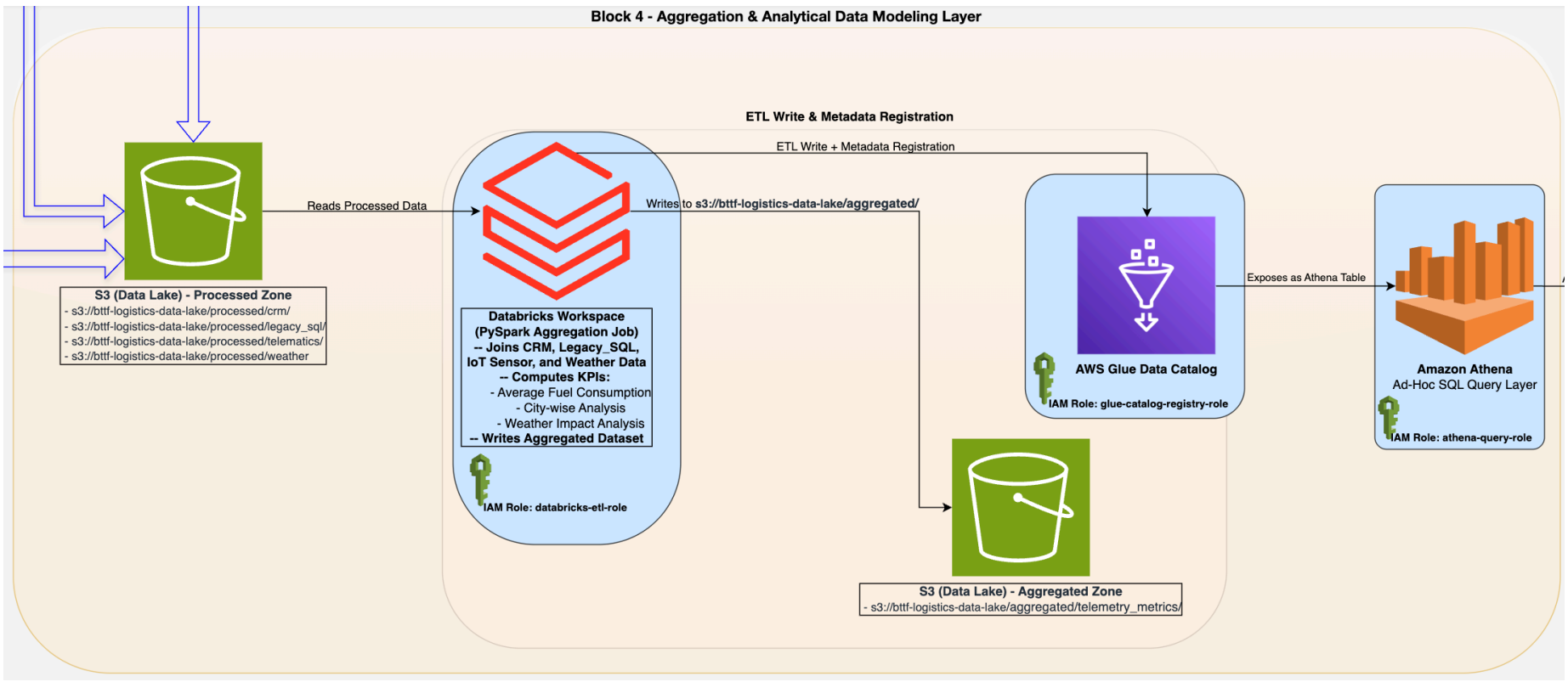
🔍 Description

ETL outputs are joined to create the analytical model:

KPIs Computed:

- Average fuel consumption by:
  - City
  - Temperature range
  - Weather condition

```
Join: CRM + Telematics + Weather + Shipment
Output: s3://.../aggregated/telemetry_metrics/
```



Block 5: Query Layer (Glue + Athena)

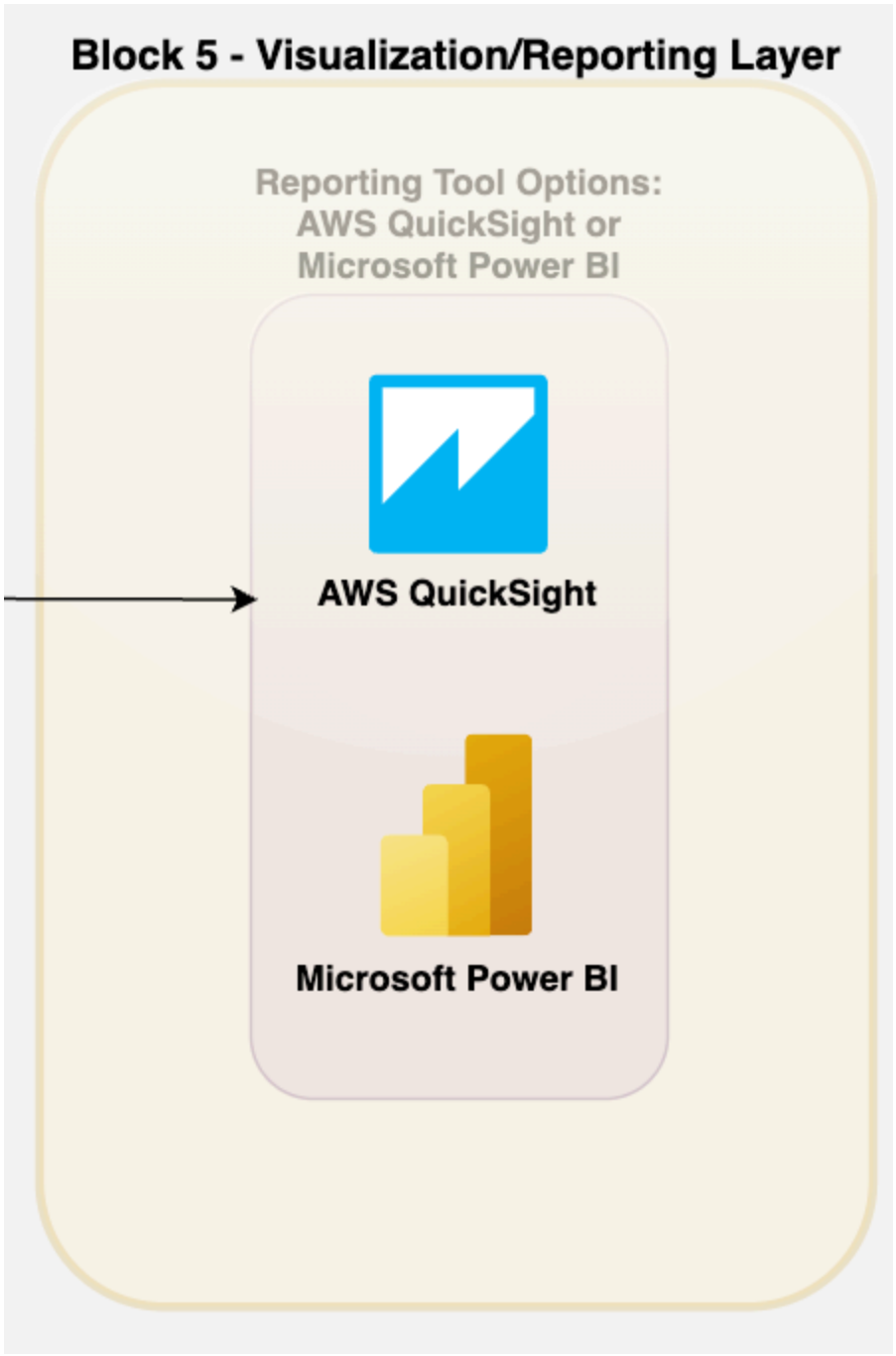
Description

Data is queried and exposed for BI tools.

Tools:

- Glue Data Catalog (registers telemetry_metrics)
- Amazon Athena (runs SQL queries directly on S3)
- Connected to BI tools via Athena Connector

```
Query: SELECT * FROM telemetry_metrics WHERE temp_range = 'Low';
```



Block 6: Monitoring & Alerting Layer

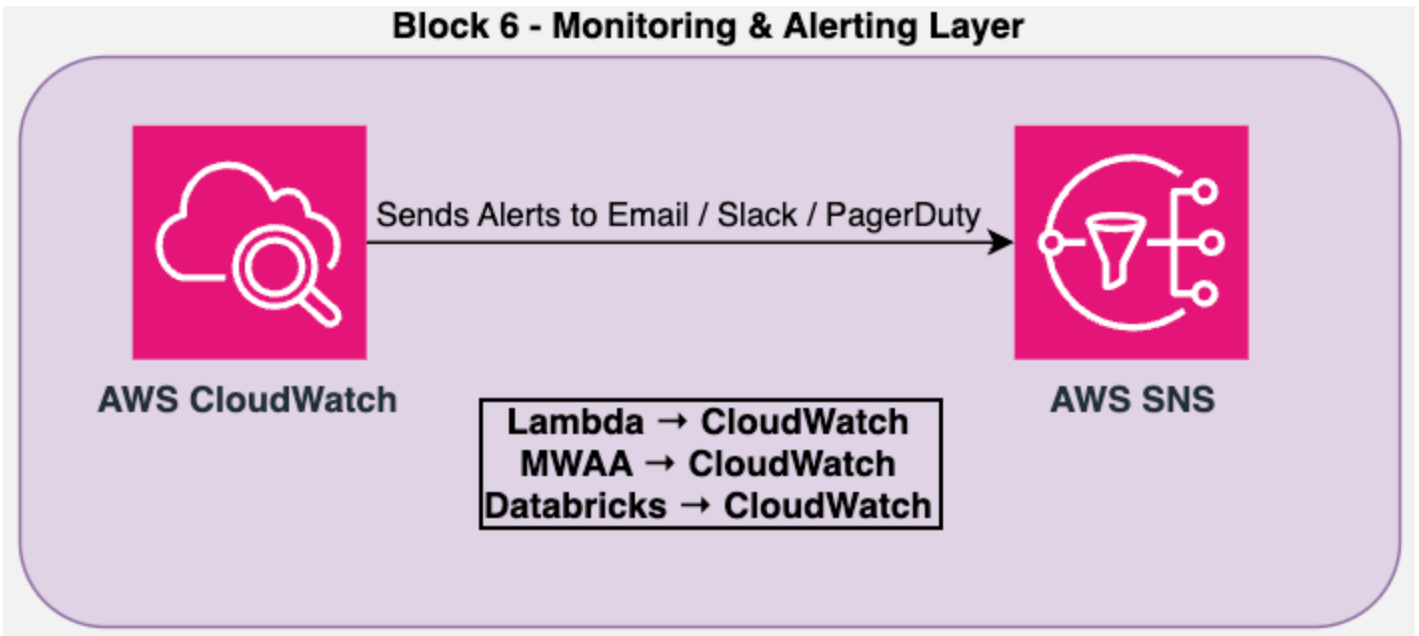
Description

Monitors system health, logs errors, and alerts stakeholders.

Flow:

```
Lambda / MWAA / Databricks → CloudWatch Logs
CloudWatch → Alarms → SNS Alerts
```

NOTE: Alerts routed via email, Slack, or PagerDuty



Block 7: Scalability Layer

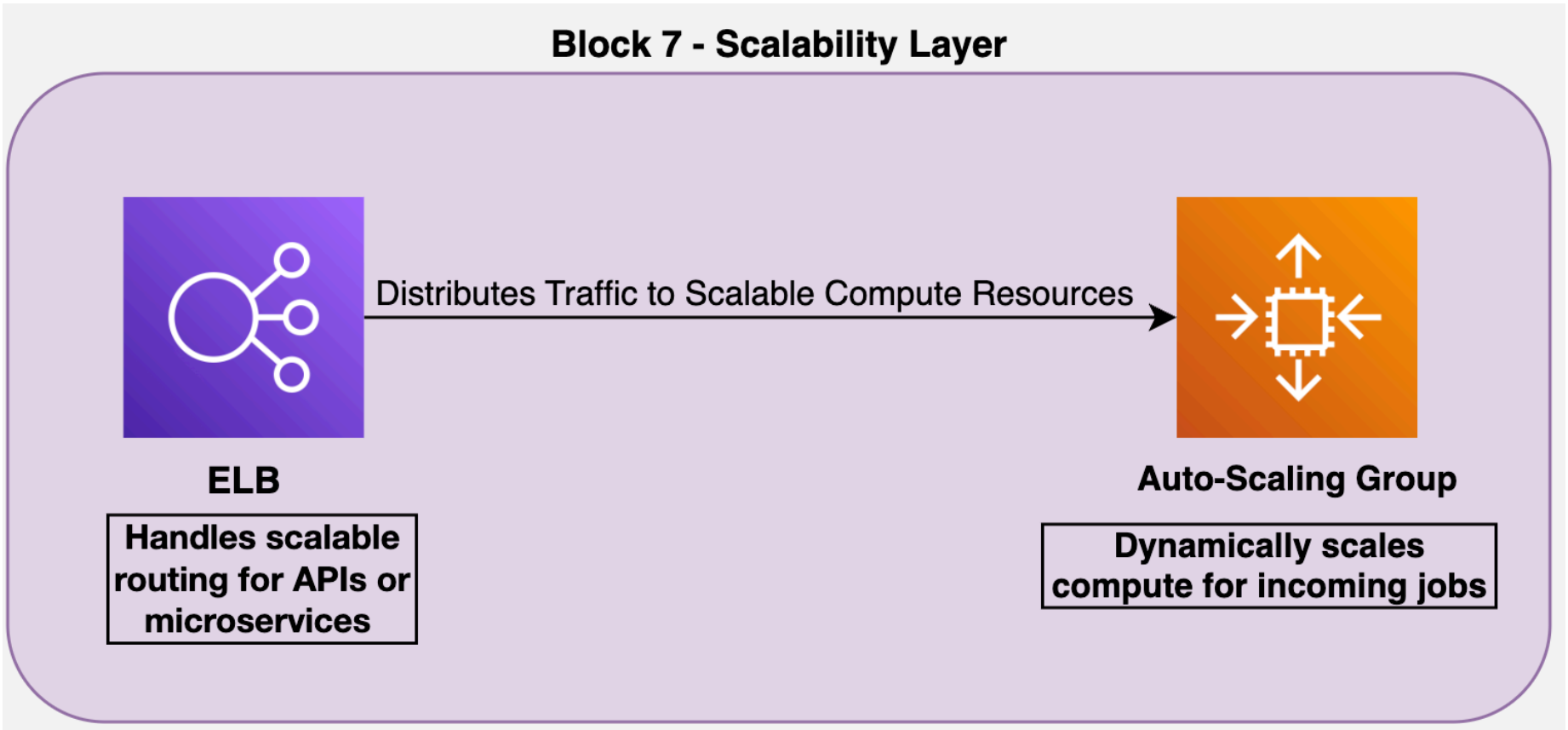
Description

Ensures elasticity under load and cost-optimized compute usage.

Components:

- **Elastic Load Balancer (ELB)**
- **Auto Scaling Group (ASG)**
- **Job trigger: DAG time/event**

NOTE: *ELB manages incoming ETL workloads, ASG scales compute dynamically*



Security Design

IAM Roles per Component:

Component	IAM Role Name	Permissions
Lambda	`lambda-execution-role`	S3 write, CloudWatch
MWAA	`airflow-dag-orchestrator`	S3 read/write, trigger notebooks
Databricks	`databricks-etl-role`	S3 + Glue access
Glue Catalog	`glue-catalog-role`	Register external tables
Athena	`athena-query-role`	SELECT permissions on aggregated dataset

Scaling Considerations

- S3 scales automatically as data grows
- PySpark is inherently distributed on Databricks
- Auto Scaling enabled for notebook clusters (future-ready)
- Glue + Athena are serverless and handle multi-user queries easily

---

Business Impact Alignment

This architecture helps BTTF:

- Democratize data across teams via query layer
- Forecast and optimize fuel consumption patterns
- Enable fast diagnostics and reporting across fleets

Next Step

→ Proceed to LEVEL2_Weather_Data_Ingestion – Implement and test the weather data collection script.