[Descriptive Analysis 1](#_Toc1019794092)

[Project Description: 1](#_Toc1430229208)

[Project Title: 1](#_Toc1649424798)

[Objective: 1](#_Toc1985794350)

[Dataset: 1](#_Toc253398588)

[Methodology: 1](#_Toc1805645163)

[Tools and Technologies: 2](#_Toc242875872)

[Deliverables: 2](#_Toc1995117632)

[AWS Deployment and Service Models 2](#_Toc914194637)

[AWS Cost Analysis 2](#_Toc2032893621)

[AWS Global infrastructure 3](#_Toc1070462277)

[AWS IAM 3](#_Toc1138330697)

[AWS VPC 3](#_Toc158185041)

[AWS Lambda 3](#_Toc164704258)

[AWS EBS 3](#_Toc1021751728)

# **Descriptive Analysis**

***Project 1***: This project focuses on analyzing the relationship between employee training attendance and performance metrics. The HR department seeks to determine which training programs yield the greatest impact on employee development and organizational performance. Leveraging cloud-based data lake architecture on AWS, the project follows a structured approach of data analysis, system design, and cloud implementation.

***Project Title*:** Analysis and Evaluation of Training Programs Impact on Employee Performance

***Objective***: To evaluate which training programs are most effective in improving employee skills and job performance.

***Dataset:*** Two primary datasets were provided by the HR department:

**Training Attendance Records** (CSV/JSON): Detailed information on employee participation in various training programs.

**Employee Performance Metrics** (CSV/JSON): Records of periodic performance evaluations of employees across departments.

## ***Methodology:***

**1. Data Analysis**

My first step was to thoroughly understand the business questions. In the case of Human Resources, the initial question was:

“Which training programs are most effective in improving employee performance?”

After receiving the datasets in CSV and JSON formats, I began exploring the data, familiarizing myself with the variables and selecting the relevant fields.

To visualize the possible causes affecting the effectiveness of training programs, I designed a fishbone diagram, where I classified the factors into six categories: policies, tools, processes, evaluations, environment, and people.

Among the problems identified, I found, for example:

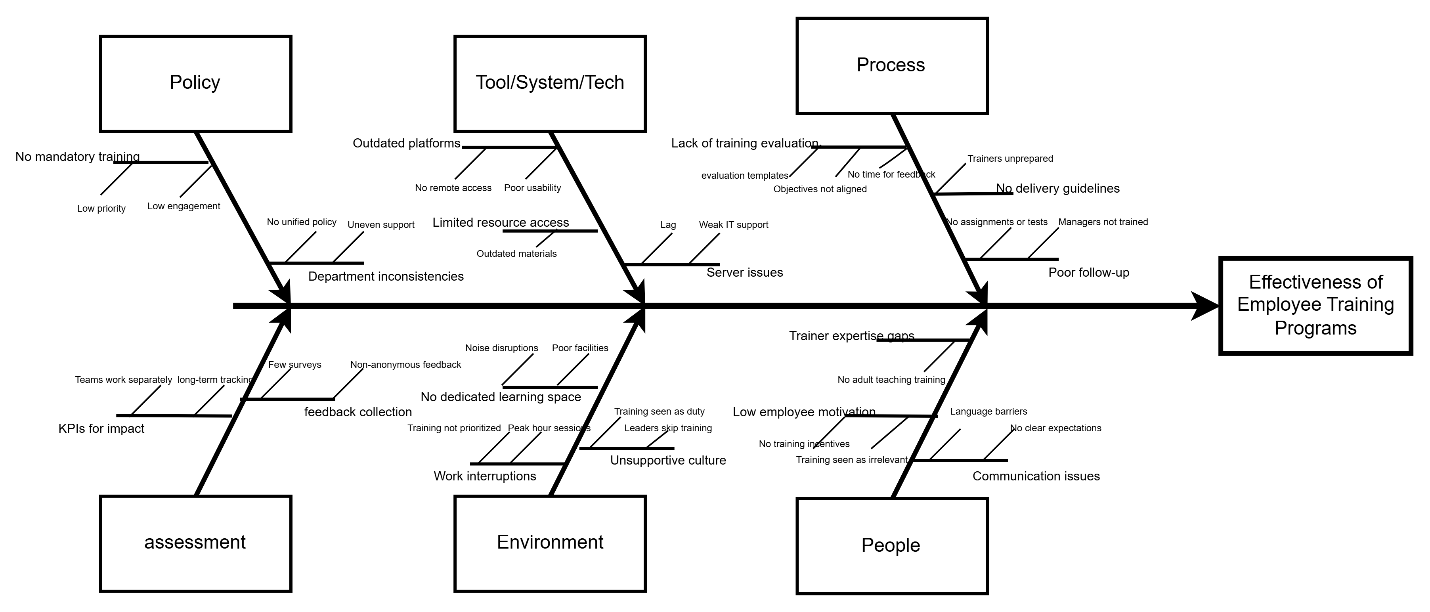
• Lack of mandatory training policies.

• Obsolete platforms with limited access.

• Inconsistent evaluations without adequate follow-up.

• Low employee motivation.

This first stage allowed me to clearly establish the objectives and root causes and prepare the solution design.



The root causes identified in the Fishbone analysis highlighted multiple areas that may contribute to varying training outcomes such as:

Lack of mandatory training policies

Outdated platforms and limited access to learning resources

Inconsistent evaluation processes and poor follow-up

Low employee motivation and communication barriers

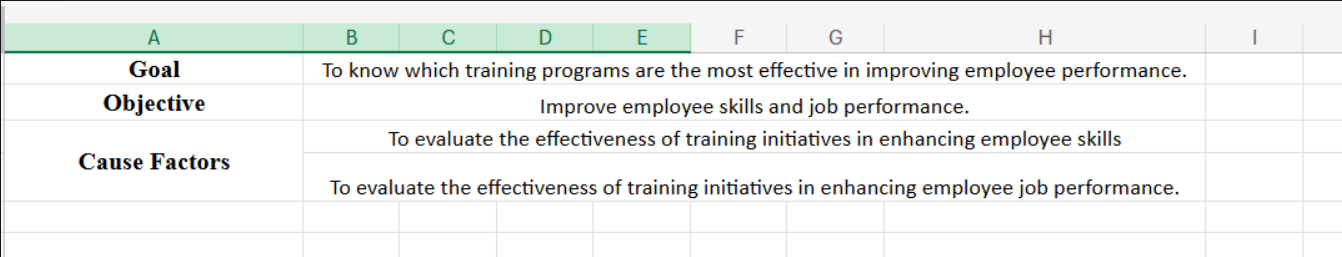
The analysis also summarized **Goal, Objective, and Cause Factors** (see Figure 2), which clarified the purpose of the analysis:

**Goal:** Identify which training programs improve performance.

**Objective:** Improve skills and job performance.

**Cause Factors:** Evaluate how training influences both skills and performance.

* This initial understanding provided a strong foundation for designing an appropriate data lake solution for subsequent processing and deeper analysis.



***Tools and Technologies*:**

* Microsoft Excel for initial data exploration and Fishbone diagram development.
* Draw.io for visual representation of analysis models.

***Deliverables*:**

* Business question breakdown and root cause analysis documentation.
* Visual Fishbone diagrams.
* Problem definition report for design phase alignment.

**2. Data Lake Design**

Based on the analysis outcomes, a cloud-based data lake architecture was designed using AWS services to ensure scalability, security, and operational efficiency. The design phase included:

**Data Storage Planning:**

Two separate S3 buckets were created to store the raw datasets, with folders organized hierarchically by year, quarter, month, and week.

Property tags were applied to facilitate cost monitoring, resource identification, and long-term manageability.

Storage classes were selected according to the data access patterns:

S3 Standard for frequently accessed datasets.

S3 One-Zone-IA (Infrequent Access) for less frequently accessed historical data.

**Data Analytics Platform (DAP) Setup:**

A Virtual Private Cloud (VPC) was configured to securely isolate cloud resources.

Within the VPC, a subnet, internet gateway, and routing tables were created to enable controlled access.

EC2 instances (t3.micro) were provisioned for compute capacity, with Elastic Block Store (EBS) volumes attached for temporary data processing needs.

Security groups were defined with restricted inbound/outbound rules to protect data ingestion endpoints.

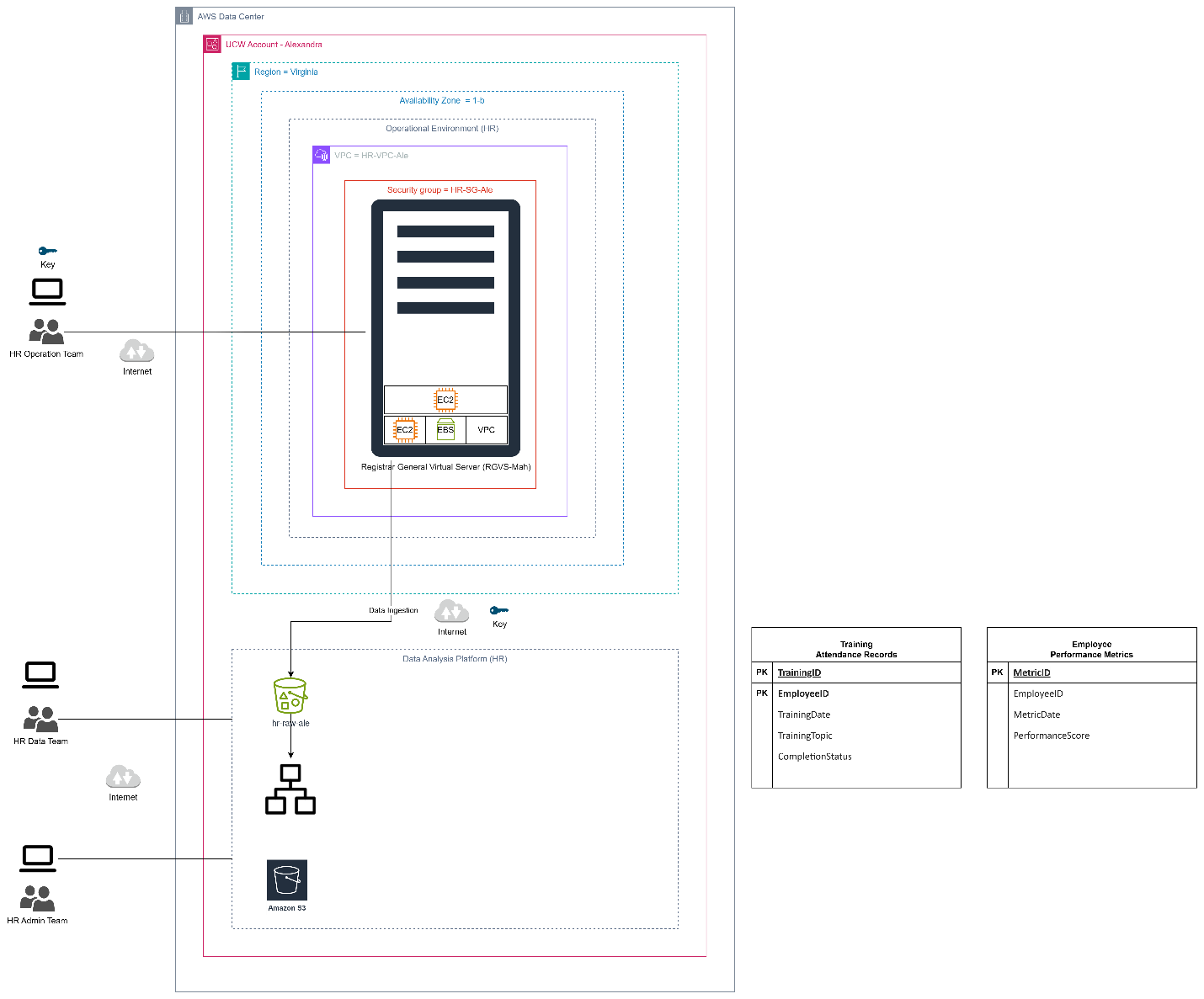
**Architecture Visualization:**  
The full architectural design was documented through a comprehensive architecture diagram (see Figure 3), illustrating:

AWS global infrastructure hierarchy.

VPC and EC2 resource allocation.

Data ingestion flow from HR users to S3 storage.

Metadata and tagging strategy.



***Tools and Technologies*:**

* AWS S3
* AWS VPC
* AWS EC2
* AWS EBS
* AWS Security Groups
* Draw.io for architecture visualization.

***Deliverables:***

* Complete data lake architecture design.
* AWS infrastructure deployed and validated.
* Visual architecture diagrams documenting cloud environment.

**3. Implementation**

The design was successfully deployed on AWS following best practices:

Commands were used to automate EC2 instance creation, security group configuration, and metadata setup. An example of the command executed:

aws ec2 run-instances --image-id "ami-09cb80360d5069de4" --instance-type "t3.micro" --key-name "vockey" \

--network-interfaces '{"AssociatePublicIpAddress":true,"DeviceIndex":0,"Groups":["sg-026d481f23eb7e45b"]}' \

--credit-specification '{"CpuCredits":"standard"}' \

--tag-specifications '{"ResourceType":"instance","Tags":[{"Key":"Name","Value":"RGVS-Mah"}]}' \

--metadata-options '{"HttpEndpoint":"enabled","HttpPutResponseHopLimit":2,"HttpTokens":"required"}' \

--private-dns-name-options '{"HostnameType":"ip-name","EnableResourceNameDnsARecord":true,"EnableResourceNameDnsAAAARecord":false}' \

--count "1"

**Lifecycle policies** were applied to S3 buckets to automate archival and cost optimization over time.

**Security configurations** ensured that only authorized HR personnel had access to sensitive data during ingestion.

This implementation phase transformed the initial design into a fully operational cloud-based data ingestion pipeline.

**3. Cost Evaluation and Data Cleaning Preparation**

Following the successful ingestion and initial setup of the data lake, the team proceeded to evaluate the operational costs of data storage and initiated the data quality management process. This phase was critical to ensure data integrity before proceeding to any analytical tasks.

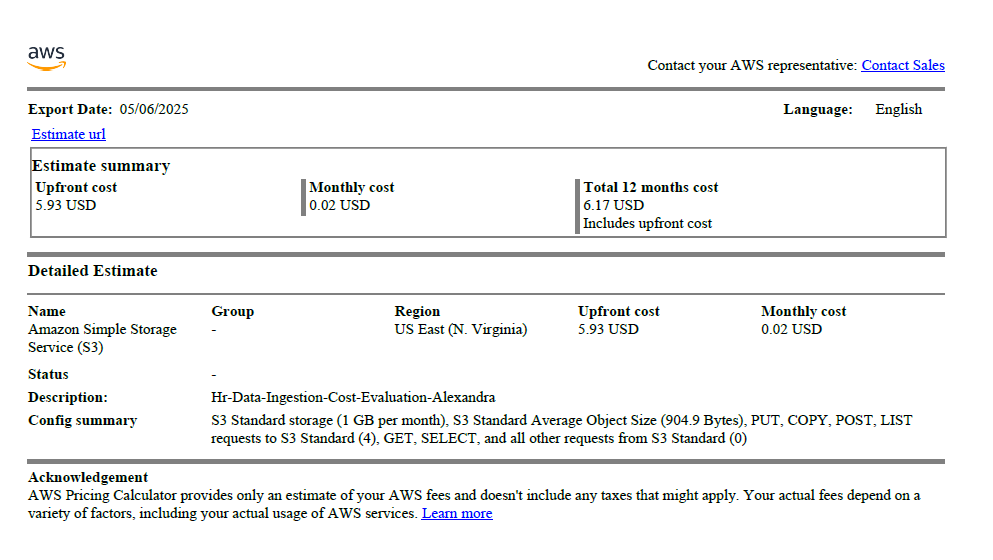
**3.1 Cost Evaluation of Dataset Ingestion**

Using the **AWS Pricing Calculator**, an initial cost estimation was conducted for the datasets stored in Amazon S3.

The pricing exercise demonstrated that, by correctly selecting storage classes (e.g., S3 Standard, One-Zone IA) and by quickly deleting unnecessary files, costs remained minimal and fully optimized from the start.

This activity enhanced understanding of AWS cost control mechanisms and reinforced the importance of proper data governance during ingestion.

*Visual evidence of the cost evaluation was captured directly from the AWS Pricing Calculator (see Figure 4).*



**3.2 Dataset Cleaning Analysis & Design**

The team employed **AWS Glue DataBrew** to assess the quality of the ingested datasets:

**Employee Performance Metrics**

**Training Attendance Records**

Dataset profiling was performed by creating projects and data workspaces within Glue DataBrew.

Key activities during profiling included:

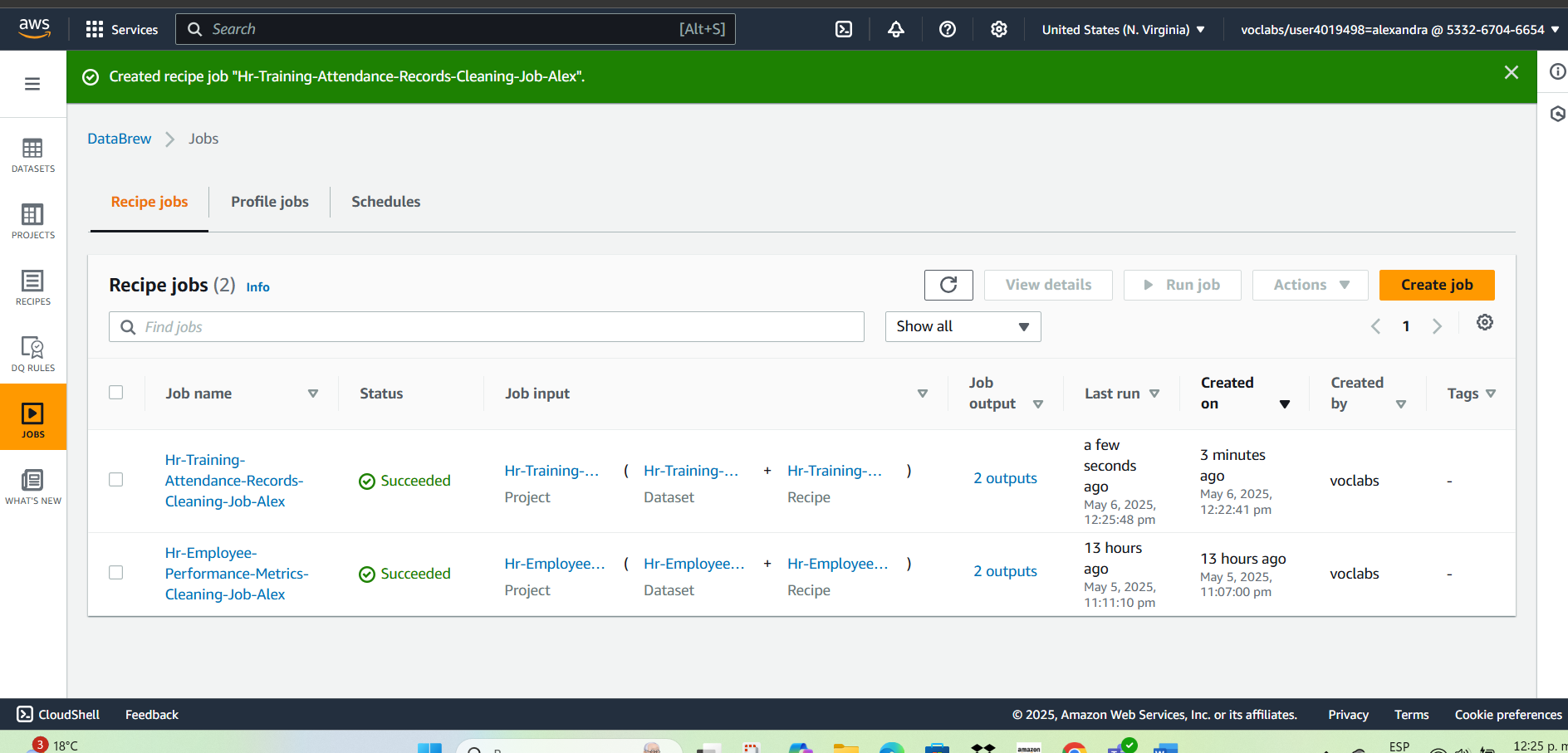
Identification and removal of redundant or unnecessary data fields.

Standardization of column names and formats for consistency.

Preparation of transformation rules to address datatype inconsistencies, null values, and incorrect formats.

This design phase ensured that data cleaning rules were well-defined before automated processing.

*The configuration of the DataBrew cleaning project is shown in Figure 5.*



**3.3 Dataset Cleaning Implementation**

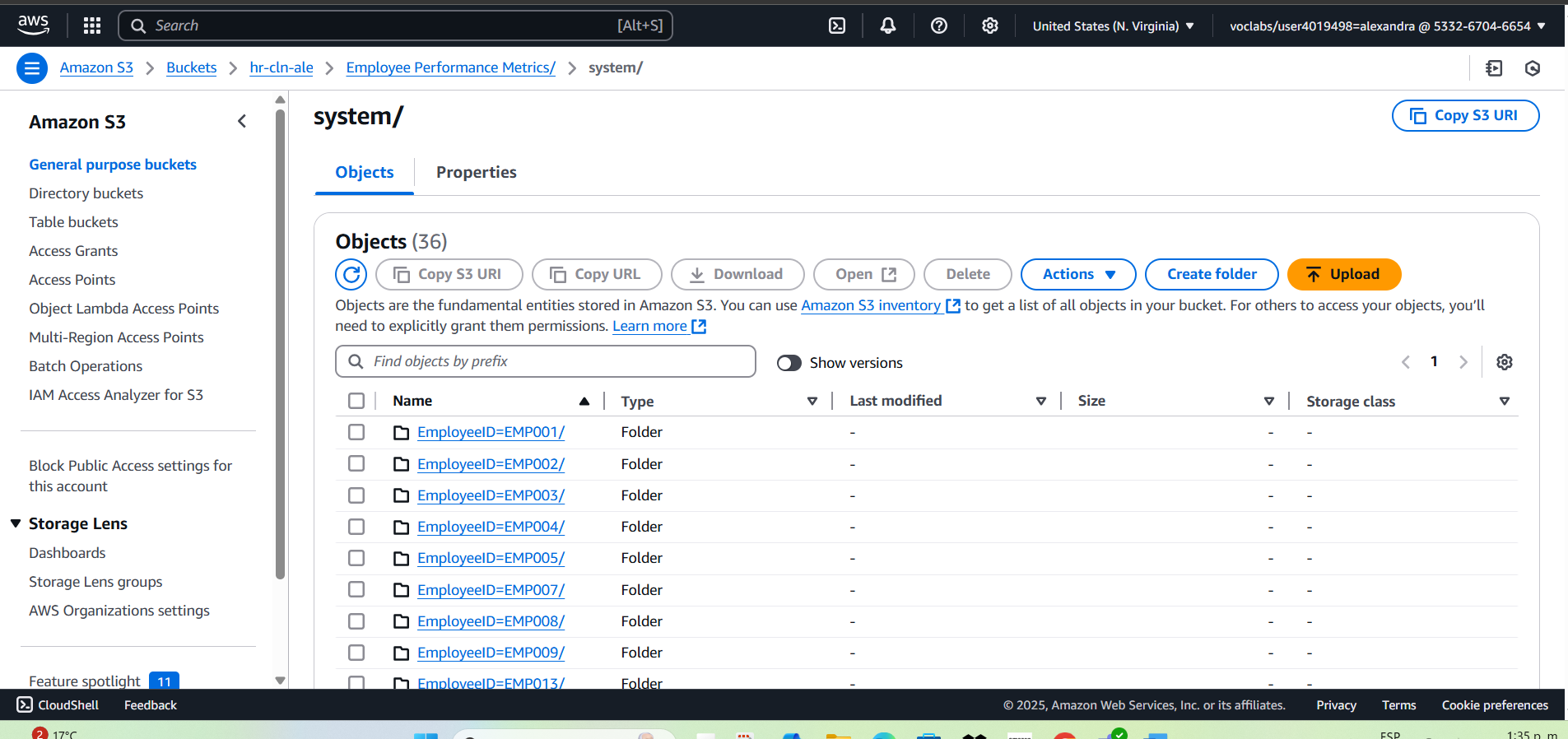
Using the pre-configured transformations, Glue DataBrew executed the data cleaning workflows automatically.

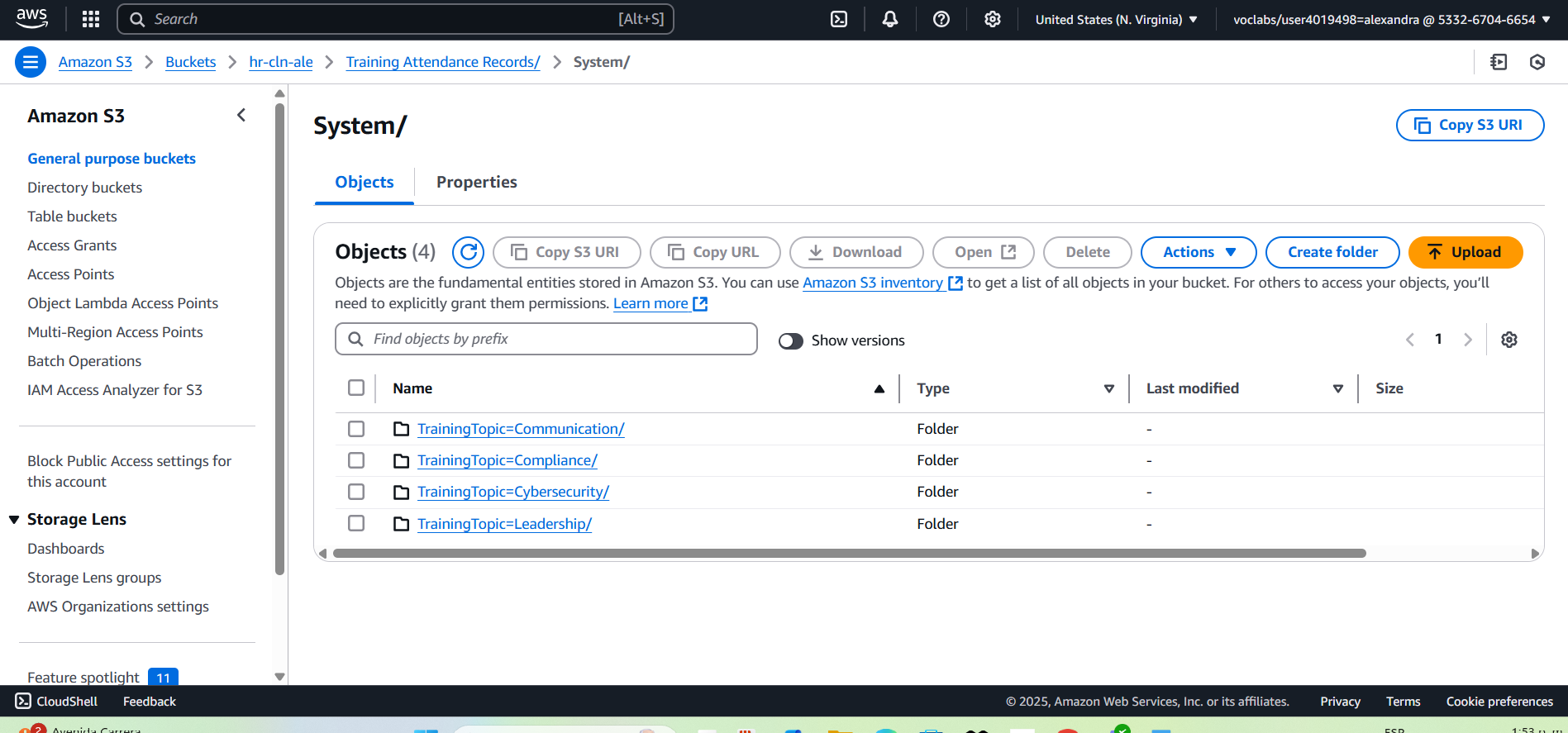
Upon successful execution, new **cleaned datasets** were generated and saved back into Amazon S3 for future analytical processing.

The validation of results was conducted via AWS S3 console to ensure all transformations were applied as intended.

This serverless approach not only accelerated processing time but also eliminated the need for managing compute infrastructure.

*Screenshots of the successful cleaning executions for both datasets are presented in Figures 6 and 7.*





***Tools and Technologies*:**

* AWS Pricing Calculator for cost analysis.
* AWS Glue DataBrew for data profiling and cleaning rule configuration.

***Deliverables*:**

* Cost estimation report for data ingestion and storage.
* Cleaned datasets stored in AWS S3.
* Data profiling summary with identified quality issues and solutions.

**4. Curated Data Processing and Advanced Cost Evaluation**

Following the successful ingestion and cleaning of raw data, the next phase involved transforming the cleaned datasets into curated, analysis-ready datasets through serverless ETL pipelines, while maintaining continuous monitoring of storage costs.

**4.1 Refining Business Question and Metrics Design**

The team revisited the original business question:

**"Which training programs are most effective in improving employee performance?"**

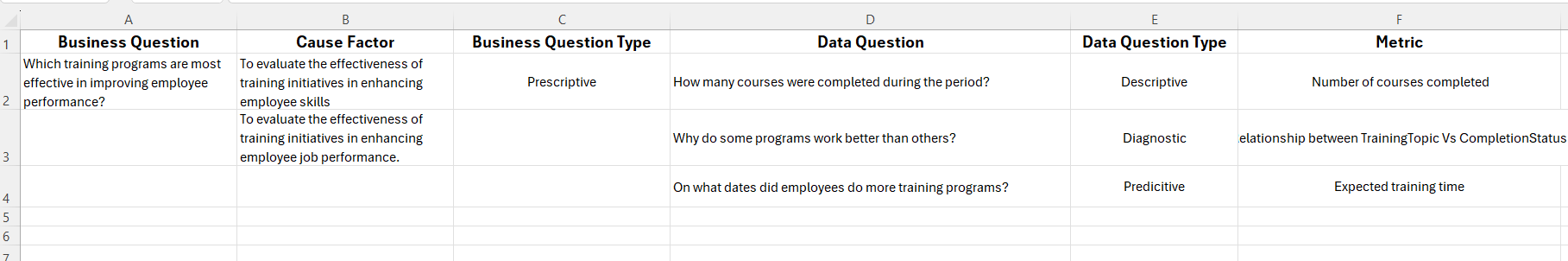
A data science framework was applied to categorize the problem across analytics types:

**Descriptive** (what happened),

**Diagnostic** (why it happened),

**Predictive** (what might happen),

**Prescriptive** (what should be done).



This categorization led to the formulation of additional sub-questions, allowing for richer metric design and deeper exploration of factors contributing to training effectiveness.

**4.2 ETL Pipeline Design with AWS Glue**

Using **AWS Glue**, an ETL (Extract, Transform, Load) job was developed to automatically process the cleaned data stored in S3.

The workflow performed the following tasks:

Extracted data from the "Cleaning" folder in S3.

Applied further transformations, including data normalization and enrichment.

Stored the processed outputs into a curated S3 bucket under the ETL zone hr-ETL-ale.

Cataloged the resulting curated datasets in the AWS Glue Data Catalog for easy discoverability.

**4.3 Data Exploration with Amazon Athena**

Leveraging the curated datasets, **Amazon Athena** was used to perform serverless SQL-based queries directly on the data stored in S3.

Simple and exploratory queries were executed to validate data quality, extract initial insights, and practice interactive querying without needing dedicated compute resources.

This approach allowed real-time data exploration while maintaining cost-efficiency.

**4.4 Updated Cost Evaluation**

A revised **cost evaluation** was conducted after these additional processing steps.

AWS usage metrics were reviewed, including:

Storage volume,

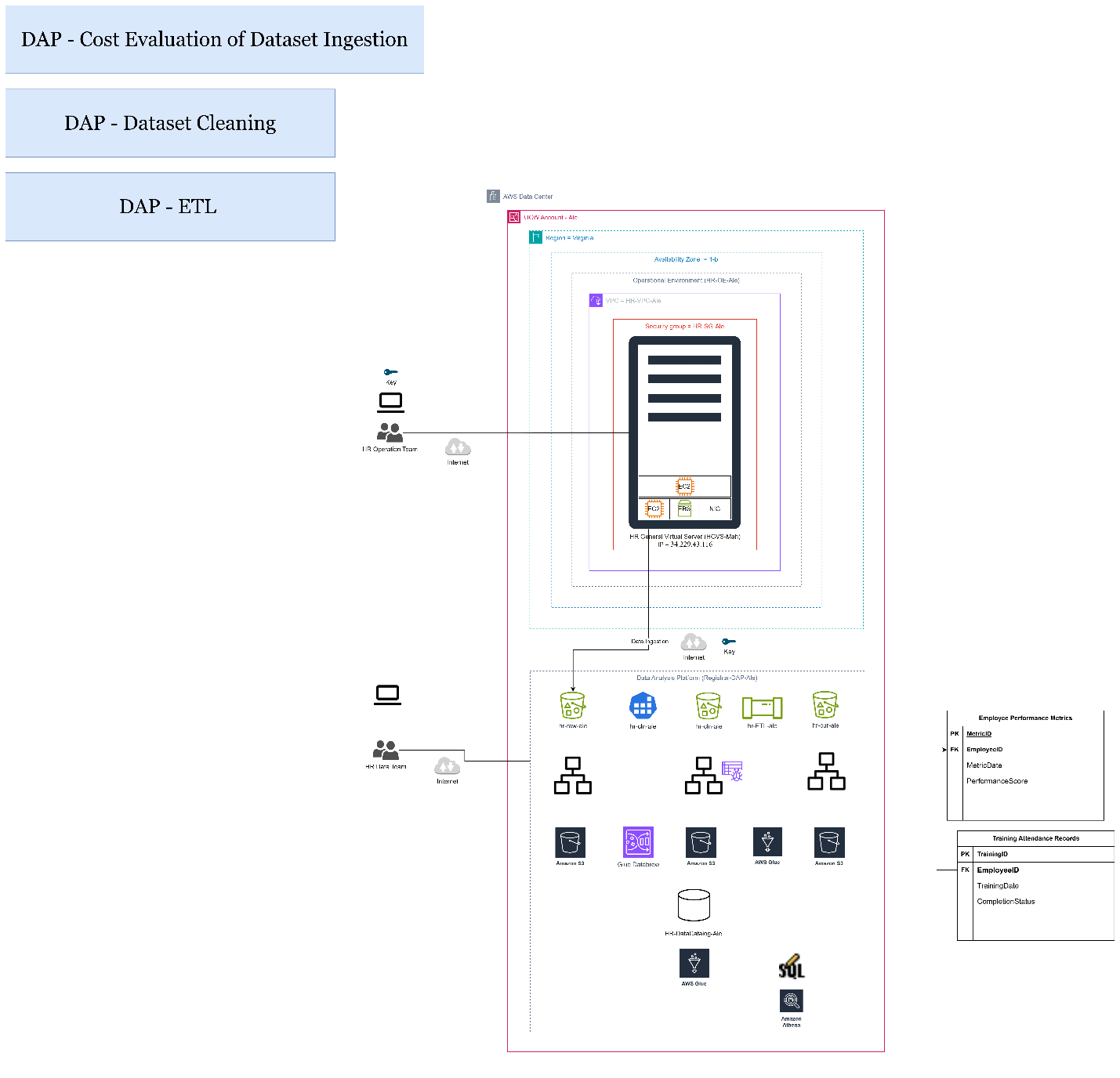
Execution minutes for Glue ETL jobs,

Query costs from Athena usage.

Preventive **cost alerts** were configured to proactively monitor spending thresholds.

Again, proper data lifecycle management and efficient service selection resulted in minimal incremental costs.





***Tools and Technologies:***

* AWS Glue Studio for ETL workflow design.
* AWS Glue Data Catalog for metadata management.
* Amazon Athena for SQL-based querying.
* CloudWatch for cost alarm monitoring.

***Deliverables:***

* ETL pipelines convert cleaned data into curated datasets.
* Metadata catalog for query-ready data.
* Diagrams from Draw.io

**5. Candidate Engagement Analysis**

As part of the advanced practice, the HR team members were assigned complementary business questions to reinforce skills developed in previous weeks. For this extended scenario, a new business problem was addressed:

**Business Question:**  
*"How does candidate engagement vary across application stages and platforms?"*

**5.1 Analysis**

A full operational architecture design was prepared starting from a detailed Excel structure which was later translated into a visual architecture diagram using **Draw.io**.

The architecture design separates public-facing data from internal protected datasets, ensuring compliance with security standards while enabling flexible data processing workflows for candidate engagement tracking.

The design documents include:

Fishbone diagram to analyze causes affecting candidate engagement (Appendix A)

Full Data Analytics Platform (DAP) Solution Architecture (Appendix B)

**5.2 Design**

Visual diagrams were finalized using **Draw.io**, representing both the data flow and the system architecture.

Supporting Excel design files were created to define:

Data lake folder structure

Cleaning rules and schema definitions

ETL process design

This systematic design provided a comprehensive blueprint for subsequent cloud deployment.

**5.3 Implementation**

The initial data ingestion process was implemented by creating the first **S3 bucket**, respecting the designed folder hierarchy.

Four distinct datasets were ingested:

Web Server Page Visit Logs

General Server Platform Engagement Summary

Beanstalk Application Interaction Logs

Lambda Engagement Tags

Proper tagging and metadata were applied to all datasets, facilitating future resource governance.

**Lifecycle rules** were configured in S3 to optimize storage costs according to access frequency.

**5.4 Operational Environment Architecture**

The final architecture incorporates additional AWS services such as:

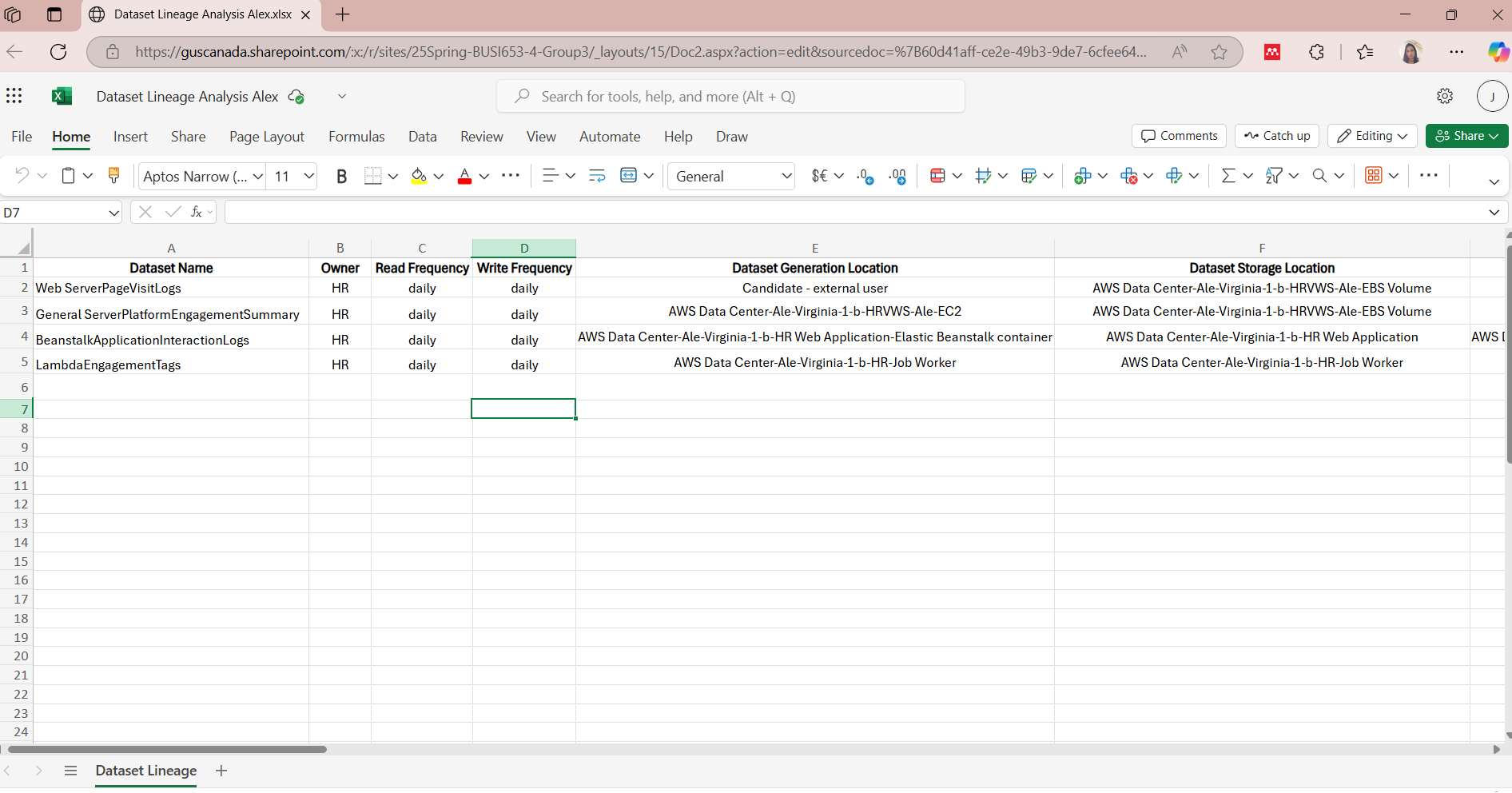
EC2 instances within both public and private subnets

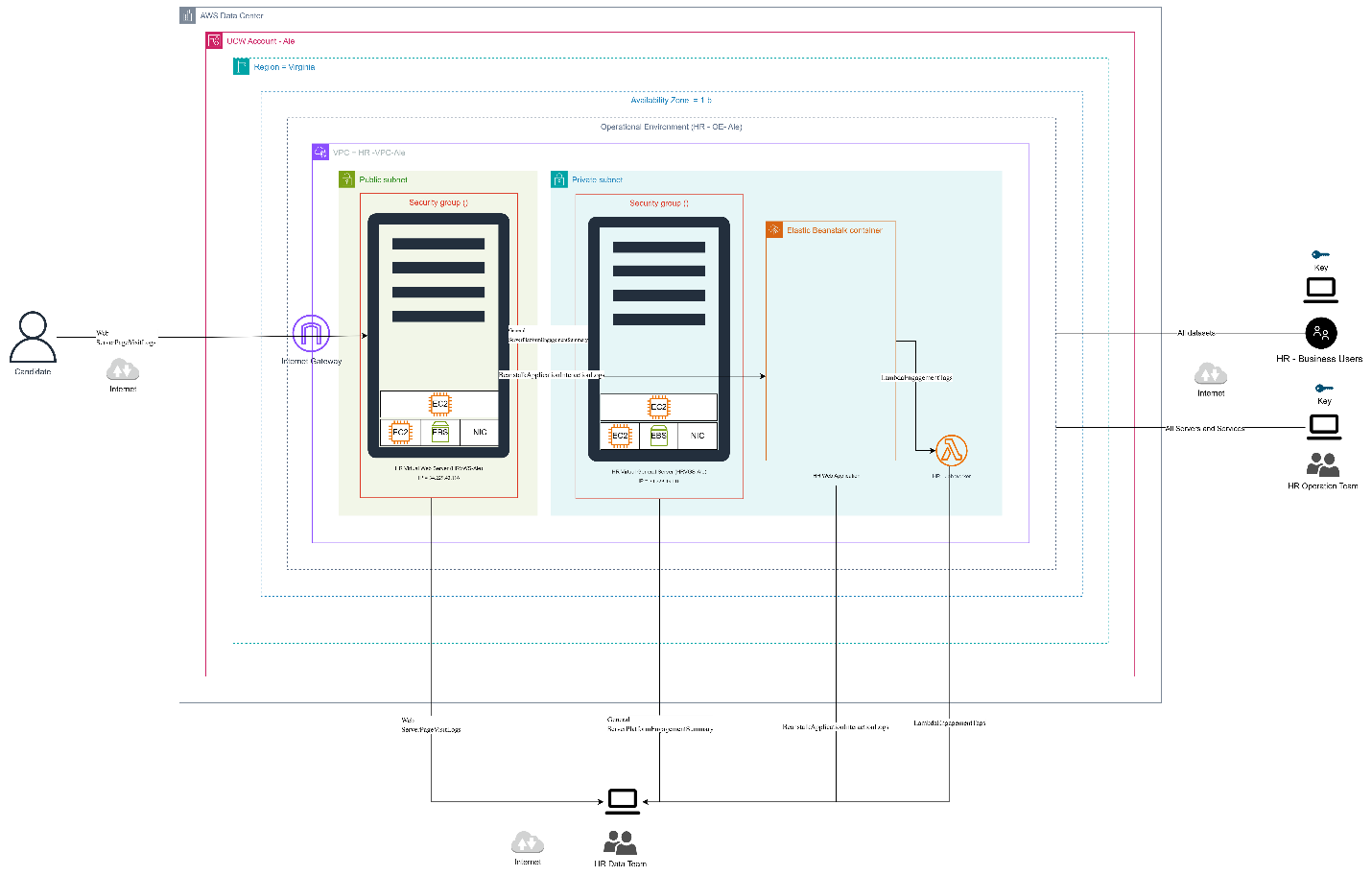
Elastic Beanstalk containers for application hosting

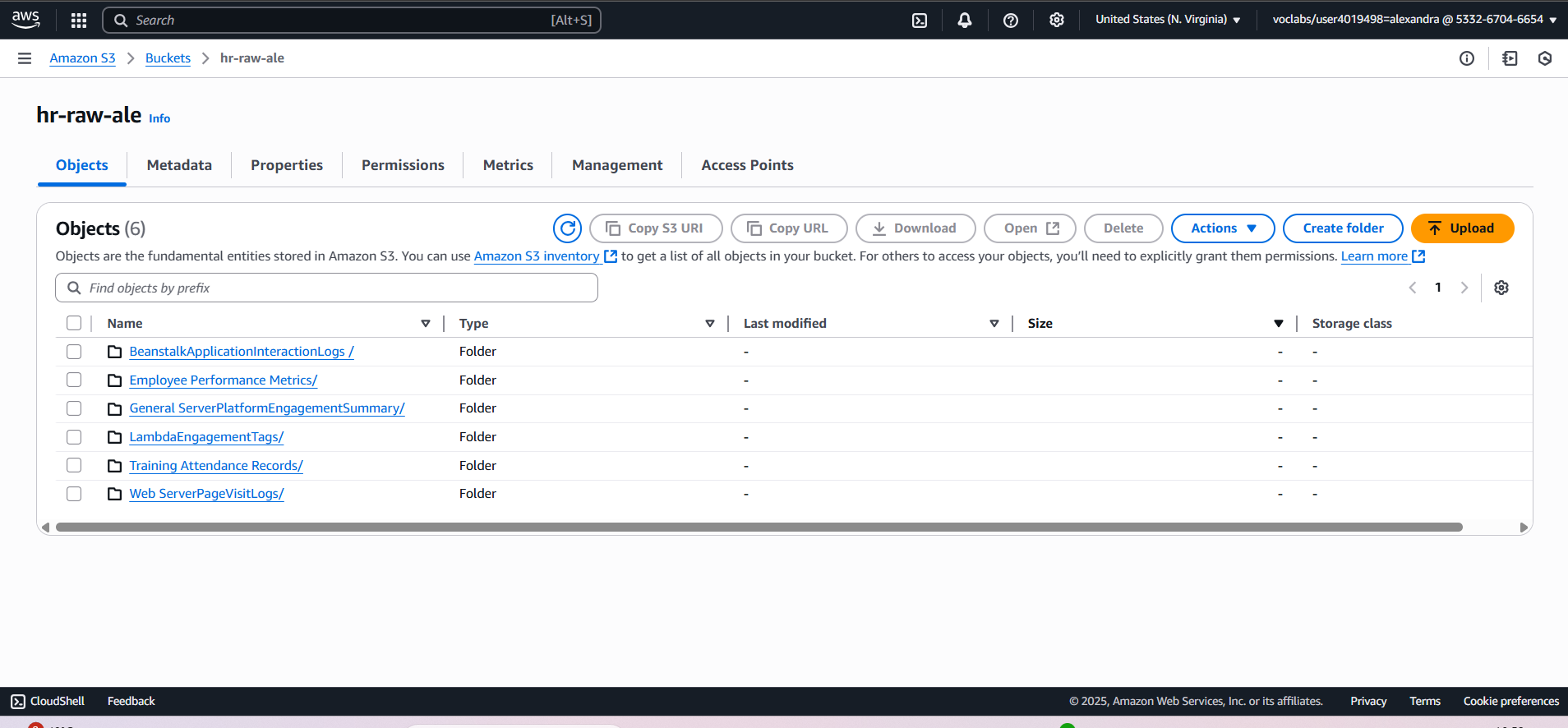
Lambda functions for tag management and event-driven processing

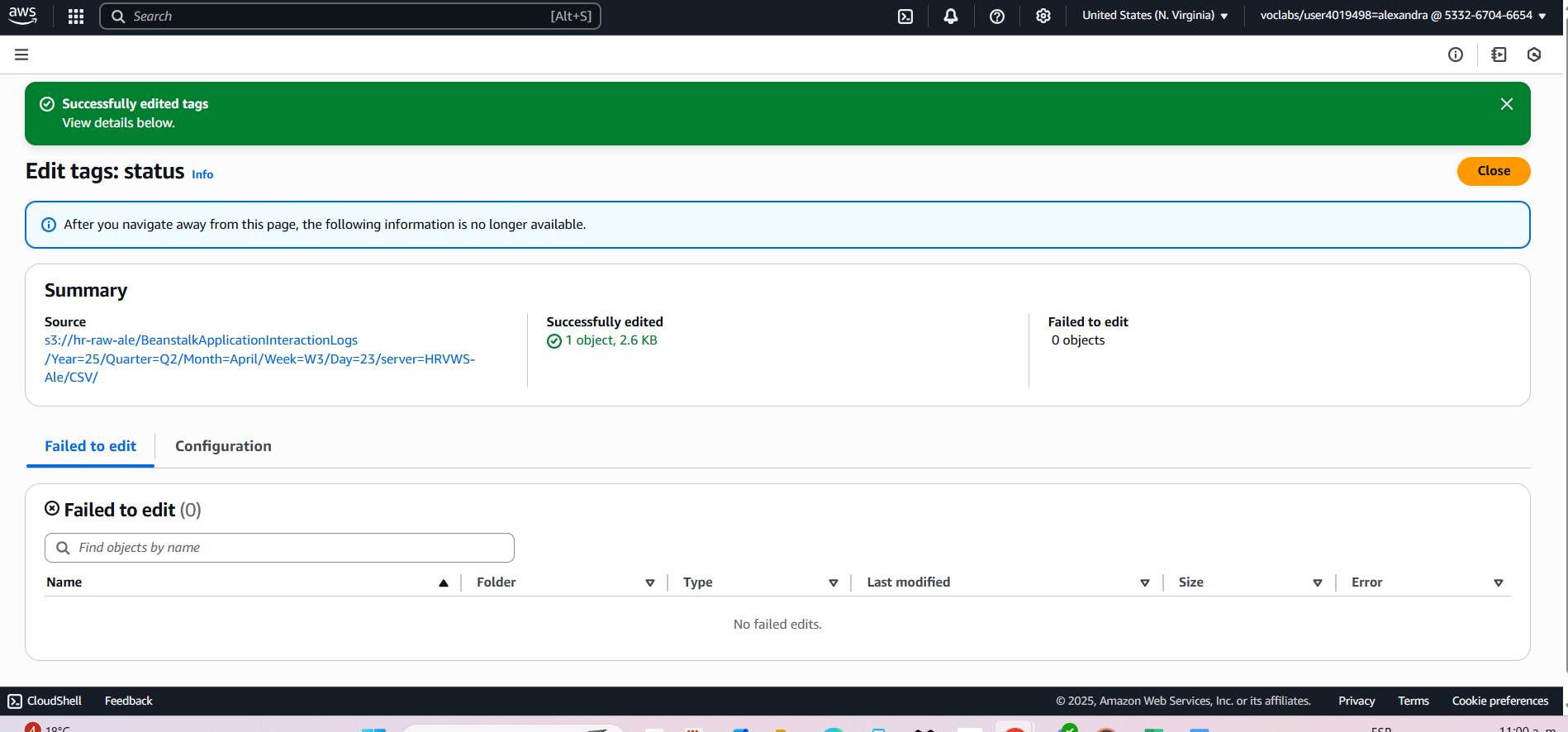
Internet Gateway for external access control

*Architecture diagrams and supporting data files are provided in next images*









***Tools and Technologies:***

* Draw.io for architecture modeling.
* Excel for data preparation designs.
* AWS Glue
* AWS Athena
* AWS S3
* AWS Pricing Calculator
* AWS CloudWatch
* SQL

***Deliverables:***

* Cost assessment report.
* Visual diagrams of data architecture and ETL flow.
* A fully automated ETL pipeline for HR data.
* Analytical results accessible through Athena.
* Cost-effective architecture with continuous monitoring.
* Key insights that support strategic HR decisions.

***Project 2***: City of Vancouver Capital Budget Analysis

***Project Title:*** Descriptive analysis of data on Capital Budget in City of Vancouver

***Objective:*** The purpose is to present the design and analysis of the data platform focused on understanding the budget changes made by the City of Vancouver in its 2023–2026 Capital Plan and 2025 Capital Budget. For this analysis, a public dataset was selected from Vancouver's open data portal, which contains detailed information on the original budget allocations, changes approved to date, and the final revised budget, categorized by service areas (such as arts and culture, transportation, civic facilities, community facilities, among others)

***Dataset:*** 2023–2026 Capital Plan and 2025 Capital Budget Obtenido del portal de datos abiertos de la Ciudad de Vancouver

## ***Methodology:***

Como ejercicio final, yo aplique el ciclo completo de desarrollo de la plataforma de datos a un nuevo conjunto de datos del mundo real proporcionado por el portal de la ciudad de Vancouver. Este proyecto me permitió consolidar habilidades en arquitectura de datos en la nube, ingesta, limpieza, enriquecimiento y análisis utilizando los servicios de AWS.

**Business Question:**  
*"How have the original budgets changed compared to the revised plan in the City of Vancouver's 2023–2026 Capital Plan and 2025 Capital Budget?"*

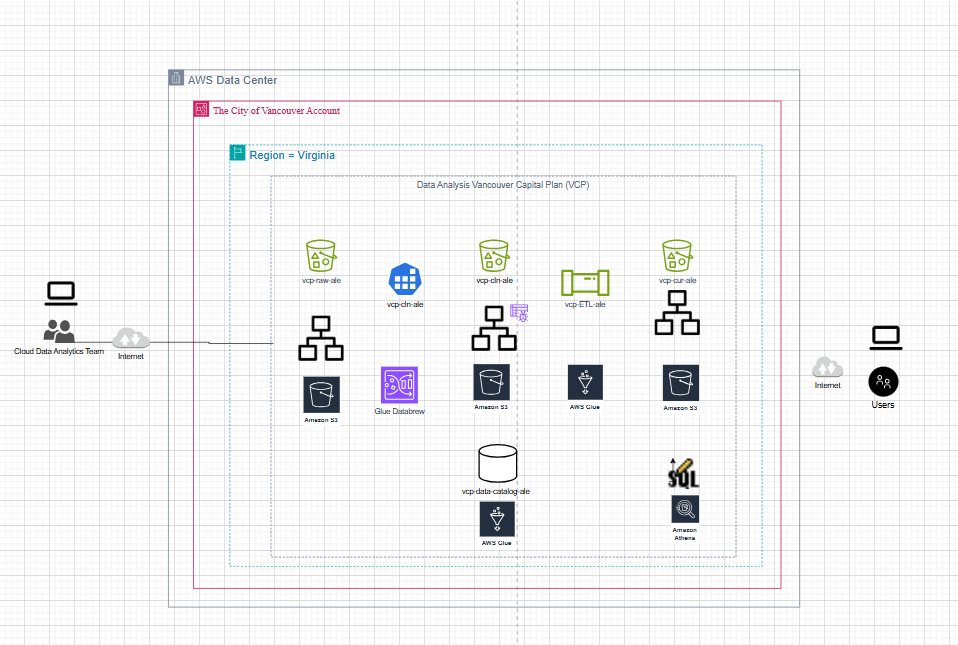
***Dataset Source:***  
Public dataset obtained from **Vancouver Open Data Portal**, including:

Original capital plan allocations

Approved revisions

Final revised capital budgets

Categorized by service area (e.g., transportation, arts and culture, civic facilities, community facilities.)



**Data Ingestion**

The dataset was ingested into AWS using **Amazon S3**, with the creation of a dedicated bucket: vcp-raw-ale.

The original file, in CSV format, was organized into structured subfolders following best practices of data lake organization.

**Data Profiling**

Using **AWS Glue DataBrew**, a full data profiling was executed, including:

File size: 362 rows x 17 columns.

Data type distribution: text, numeric, dates.

Identification of null values and inconsistencies.

This profiling guided the data cleaning process ensuring data quality prior to analysis.

**Data Cleaning**

A new S3 bucket vcp-cln-ale was provisioned to store cleaned data.

Within Glue DataBrew:

Empty text fields (e.g., Service Category 3, Project Name) were replaced with "Undefined".

Numeric columns were cleaned by replacing missing numeric values with "0".

Negative values (representing budget reductions) were kept intact after validating their meaning.

The complete cleaning workflow was encapsulated in an automated **DataBrew Job** (vcp-cleaning job-ale), which created a reliable cleaned dataset for subsequent processing.

**Data Cataloging**

A **Glue Crawler** (vcp-crawler-ale) was created to catalog the cleaned data.

The cleaned dataset was registered in a new Glue Database (vcp-data-catalog-ale) and converted into **Parquet** format for efficient querying.

The dataset was now fully discoverable and ready for advanced querying via Athena.

**Data Enrichment**

Using **Glue Studio**:

A new calculated column Budget-difference was created by subtracting the original budget from the revised budget for each record.

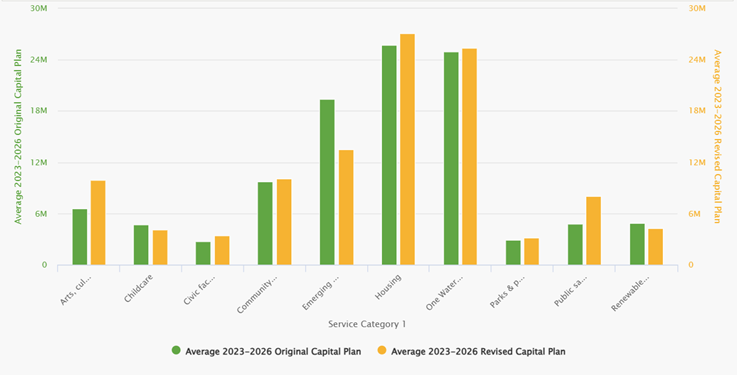
Aggregations were applied to calculate:

Average budget difference per service category.

Total budget difference.

Project count per category.

This transformation was executed successfully and validated using the Glue Output Schema Preview.

**

**Data Summarization and Querying**

The curated dataset was queried using **Amazon Athena**:

The query aggregated the data to summarize average and total budget differences by service category.

Output results included: Service Category, Average Budget Difference, Total Budget Difference, and Project Count.

Queries were saved, and results exported as CSV for reporting and visualization.

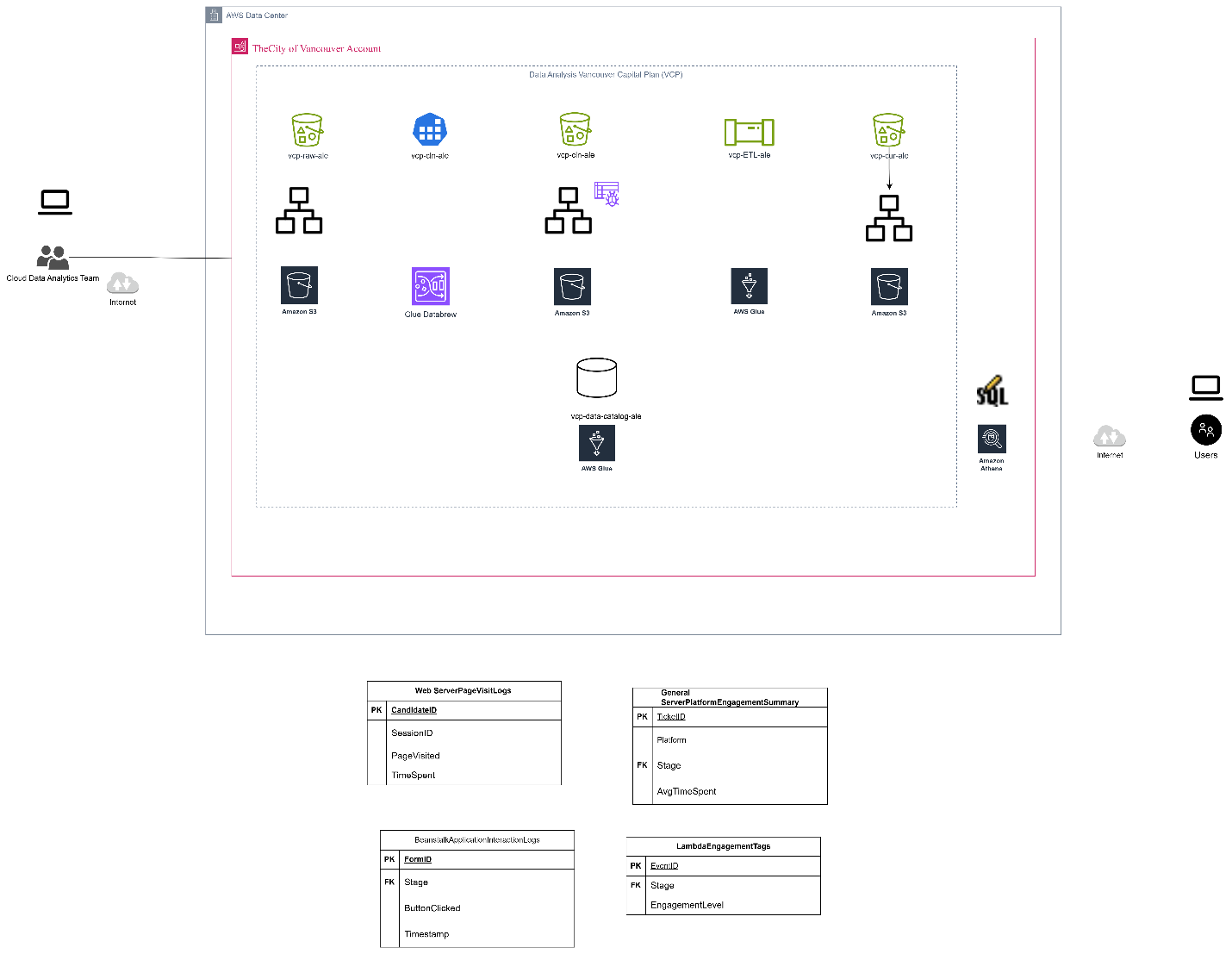
These final outputs provided direct answers to the business question, offering insights into how specific service areas experienced budget increases or reductions.

***Tools and Technologies:***

* AWS Glue DataBrew, Glue Studio, Glue Catalog.
* Amazon Athena for analytical queries.
* Draw.io for process diagrams.

***Deliverables:***

* Complete enriched dataset with budget difference metrics.
* Athena query outputs answering capital budget analysis.
* Summary report with service area budget variations.



**Security, Governance, and Monitoring Implementation**

Following the successful ingestion, cleaning, enrichment, and querying stages for the Vancouver Capital Budget Analysis, the project advanced into deeper levels of cloud security, governance, monitoring, and alignment with AWS Well-Architected Framework best practices.

**Data Security**

**AWS Key Management Service (KMS):**

A symmetric encryption key (vcp-key-ale) was created.

Default encryption was activated on all S3 buckets involved (vcp-raw-ale, vcp-cln-ale, and vcp-cur-ale), ensuring data-at-rest protection.

**Versioning and Replication:**

S3 versioning was enabled to maintain file change history.

Cross-bucket replication rules were configured to create automatic copies, strengthening data resilience.

**Data Governance and Quality Controls**

**Data Quality Rules in AWS Glue Studio:**

A dedicated ETL flow (vcp-QC-ale) was developed to validate data completeness.

Quality thresholds were applied across multiple columns (service categories and budget values).

Records were automatically classified into Passed or Failed folders based on validation results.

At runtime, no failures were detected, confirming high dataset consistency.

**ETL Adjustments:**

The governance flow was streamlined into a single processing block based on the limitations of the provided public dataset.

**Data Monitoring**

**AWS CloudWatch:**

A custom dashboard (vcp-MCR-ale) was created to monitor bucket size metrics and Glue resource usage.

Budget alarms (vcp-alarm-ale) were configured to notify when cost thresholds were approached.

**AWS CloudTrail:**

A trail (vcp-tra-ale) was configured to enable full traceability of account activities, logging all API calls and resource changes.

***Tools and Technologies:***

* AWS KMS, S3 Encryption, S3 Versioning & Replication.
* AWS Glue Quality Checks.
* AWS CloudWatch, CloudTrail for monitoring and auditing.

***Deliverables:***

* Full security configuration documentation.
* Governance quality control results.
* Monitoring dashboards, alarms, and audit logs.

**Alignment with AWS Well-Architected Framework**

The implementation was evaluated against AWS’s five pillars:

**Operational Excellence:**  
Applied iterative design, error recovery, and incremental validation but lacked full operations-as-code and failure anticipation.

**Security:**  
Strong data-at-rest encryption and traceability enabled. IAM role-based security and incident preparedness were identified as areas for future improvement.

**Reliability:**  
Data flow dependencies were well-illustrated; partial monitoring and basic alerting configured. Full disaster recovery documentation remains pending.

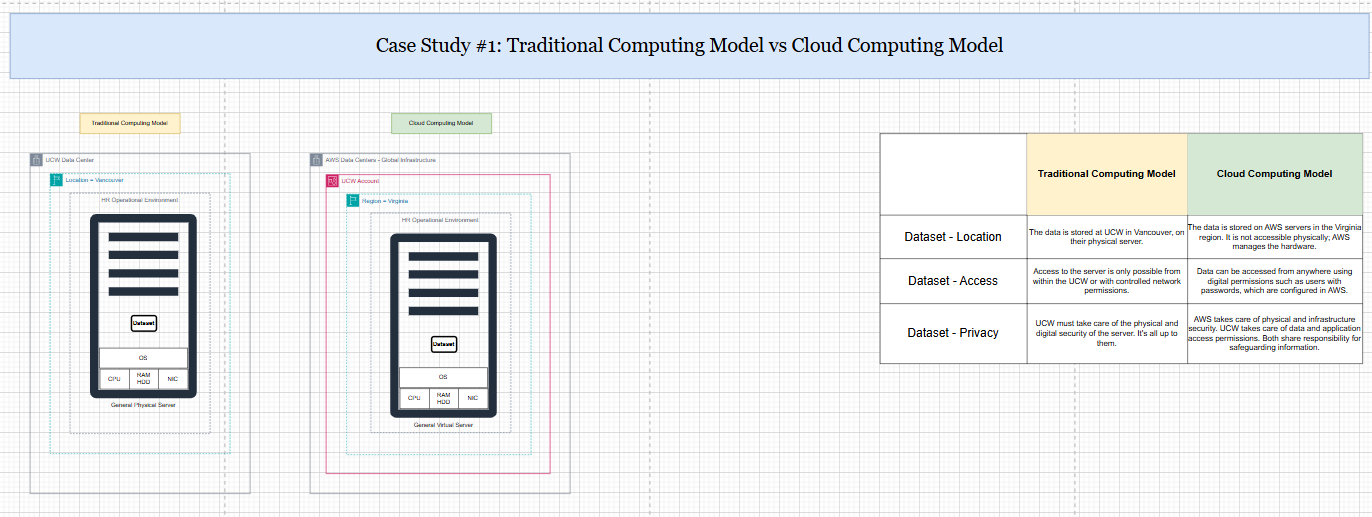
**Performance Efficiency:**  
Resource provisioning and storage optimization decisions were properly aligned. Advanced partitioning, compression, and fine-tuning opportunities were identified for future optimization.

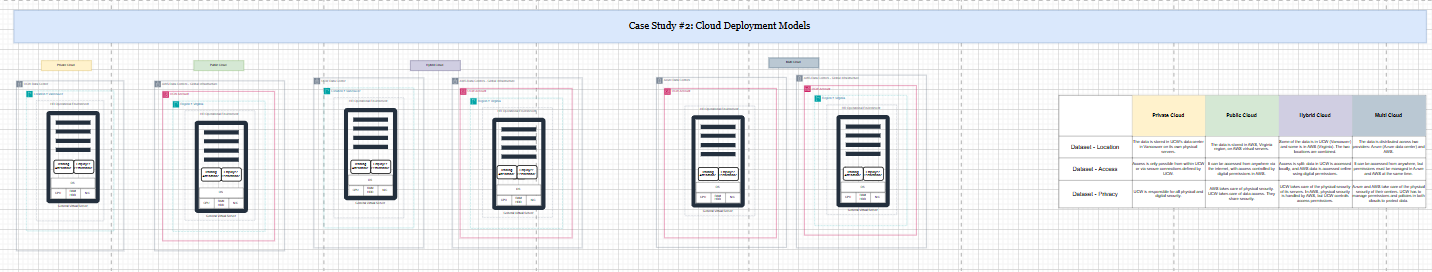
**Cost Optimization:**  
Consumption models with Glue and Athena (serverless services) reduced operational costs. Lifecycle policies, tagging, and cost monitoring were applied, but additional savings opportunities remain via data partitioning and automated reporting.

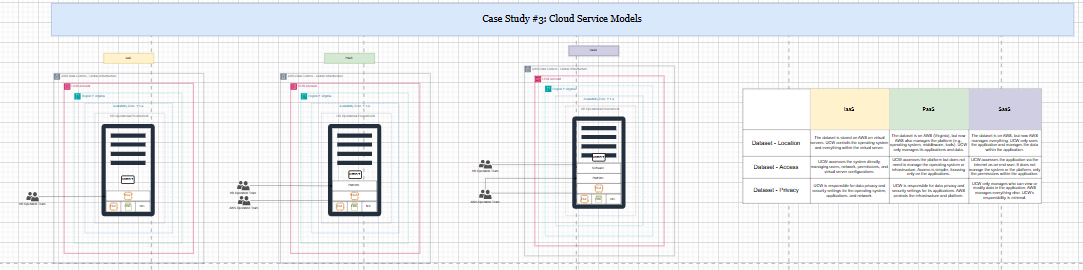
**Recommendations**

* Implement formal disaster recovery documentation.
* Fully automate security best practices (IAM roles, automated incident response).
* Optimize Athena queries via data partitioning and compression.
* Expand tagging and billing analysis using AWS Cost and Usage Reports (CUR).
* Further optimize Glue job sizing (DPUs) to balance performance and cost.

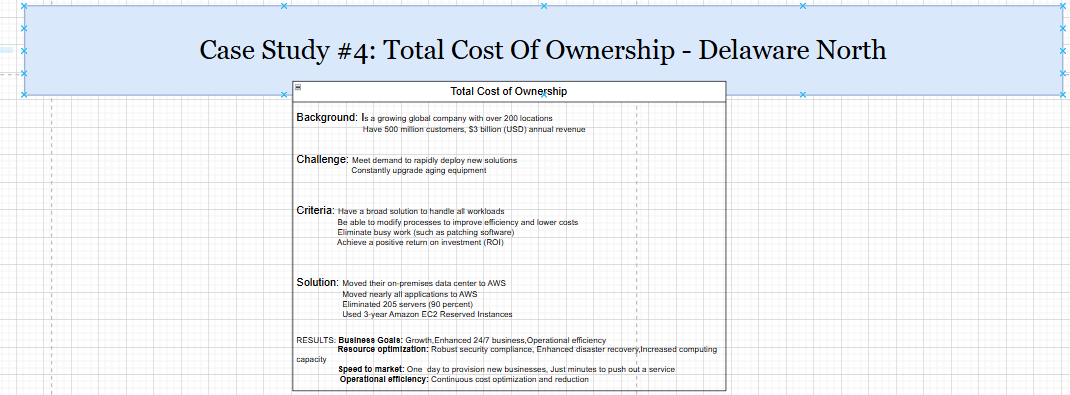
# **AWS Deployment and Service Models**

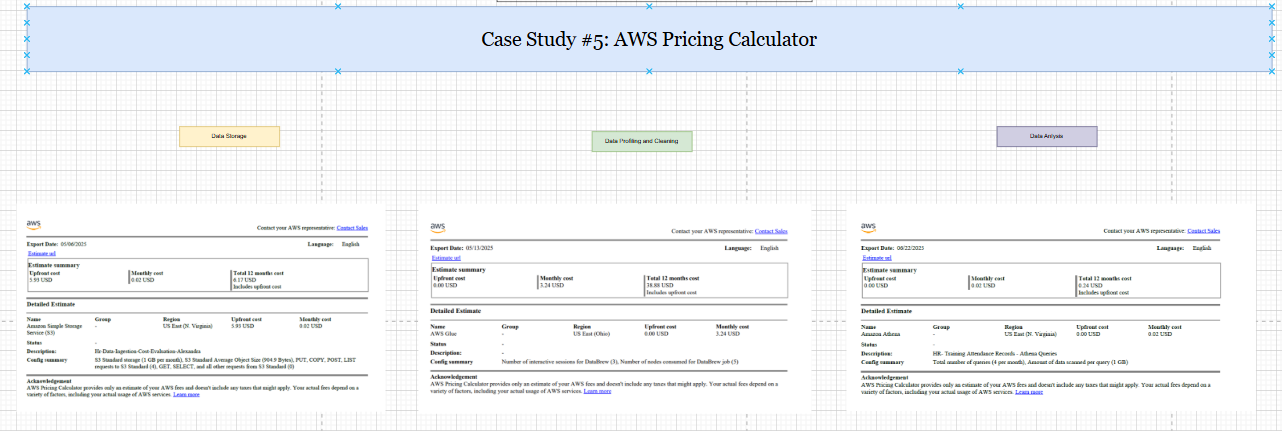
In this module, I understood how the cloud makes it easier for companies to use technology without having to worry so much about physical hardware. Before, companies like UCW had to buy servers, maintain them, and take care of the entire data center. With AWS, they can now use virtual servers and cloud storage, while AWS takes care of the physical infrastructure and security. UCW only has to manage its applications, users, and data. I also learned that depending on the model chosen private, public, hybrid, or multicloud it is possible to combine what is in the cloud and what remains on-premises. I also saw how responsibility changes depending on the type of service: in IaaS, the company manages almost everything; in PaaS, only the applications and data; and in SaaS, it is practically only responsible for using the application. Thanks to these models, companies save costs, grow faster, and focus on what is important: using technology for their business, without having to deal with all the technical details.

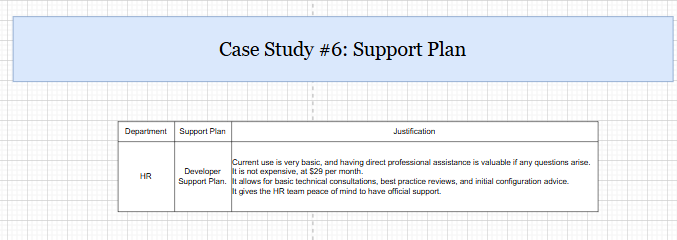




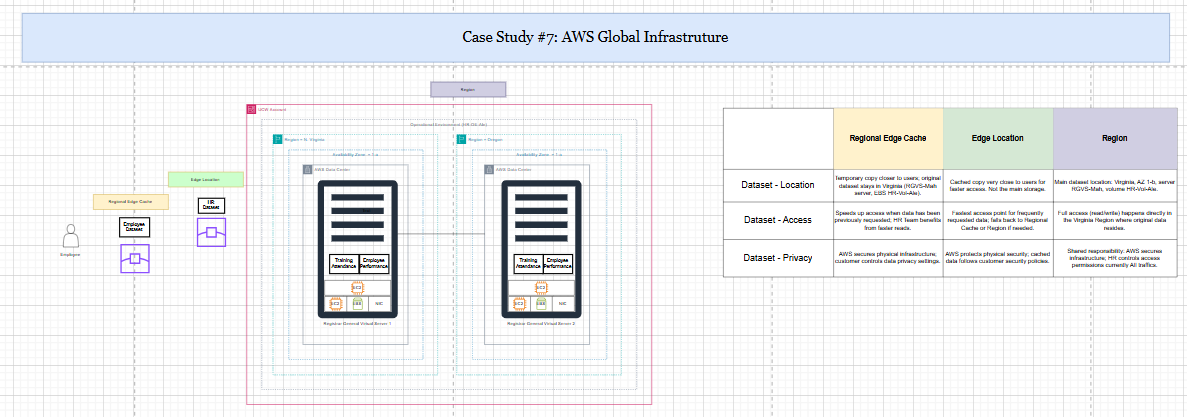
# **AWS Cost Analysis**

In this module, I understood how the cloud not only facilitates the use of technology, but also allows you to save money in a smart way. Before, when a company wanted to grow, it had to buy more servers, worry about physical space, and maintenance. Now, with AWS, it's much easier: companies only pay for what they actually use, they can increase or reduce resources as needed, and they avoid overspending. We looked at the case of Delaware North, where by moving to the cloud they were able to eliminate most of their physical servers, increase their capacity, improve security, and be better prepared for emergencies. In addition, I learned that AWS offers different types of technical support depending on what each company needs. For example, for the Human Resources team, which has simpler and less critical workloads, the Developer Support plan is sufficient, as it gives them access to professional help when they need it, but without having to pay for advanced services they would not use.



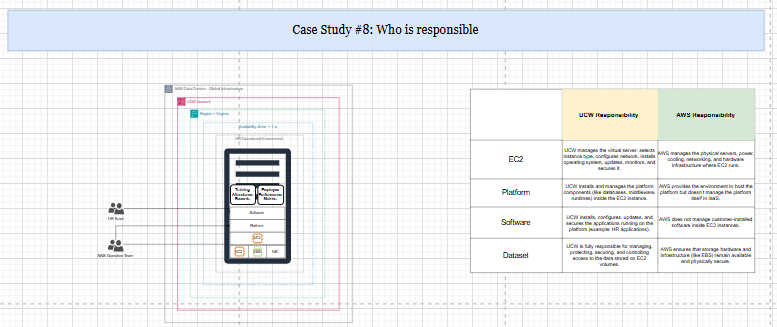


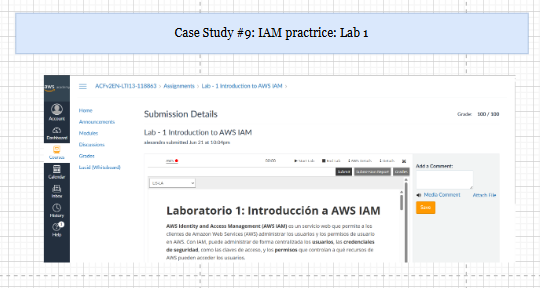
# **AWS Global infrastructure**

AWS has strategic points for storing the information most frequently accessed by its affiliates. This way, when someone requests that data again, the response is much faster because it is already closer and does not need to go to the main data center. Meanwhile, AWS is responsible for physically protecting the entire infrastructure, and the client company continues to decide who can view or modify its information.

# **AWS IAM**

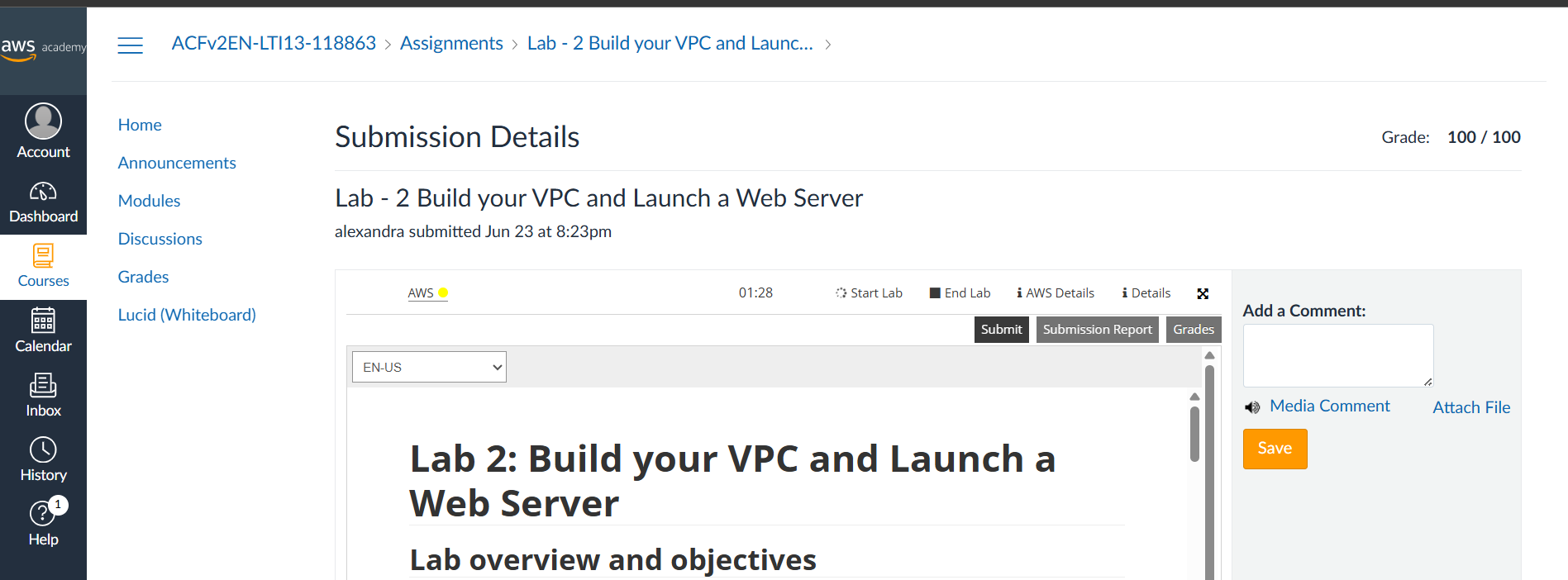
In this module, I learned how security and access management work in AWS using IAM. I practiced creating users, groups, and policies, and assigning permissions based on each person's role. For example, some users can only view EC2 or S3 resources, while others have permissions to manage servers. I also understood the difference between managed policies (which can apply to multiple users or groups) and inline policies (which are specific to a single user or group). In addition, we reviewed the shared responsibility model, where AWS is responsible for the physical infrastructure and basic security, while the company is responsible for managing the operating system, software, applications, and data within the virtual servers. This practice allowed me to see how, in the cloud, it is possible to give each person highly controlled access, based on what they actually need to do, while maintaining security and control over the data.





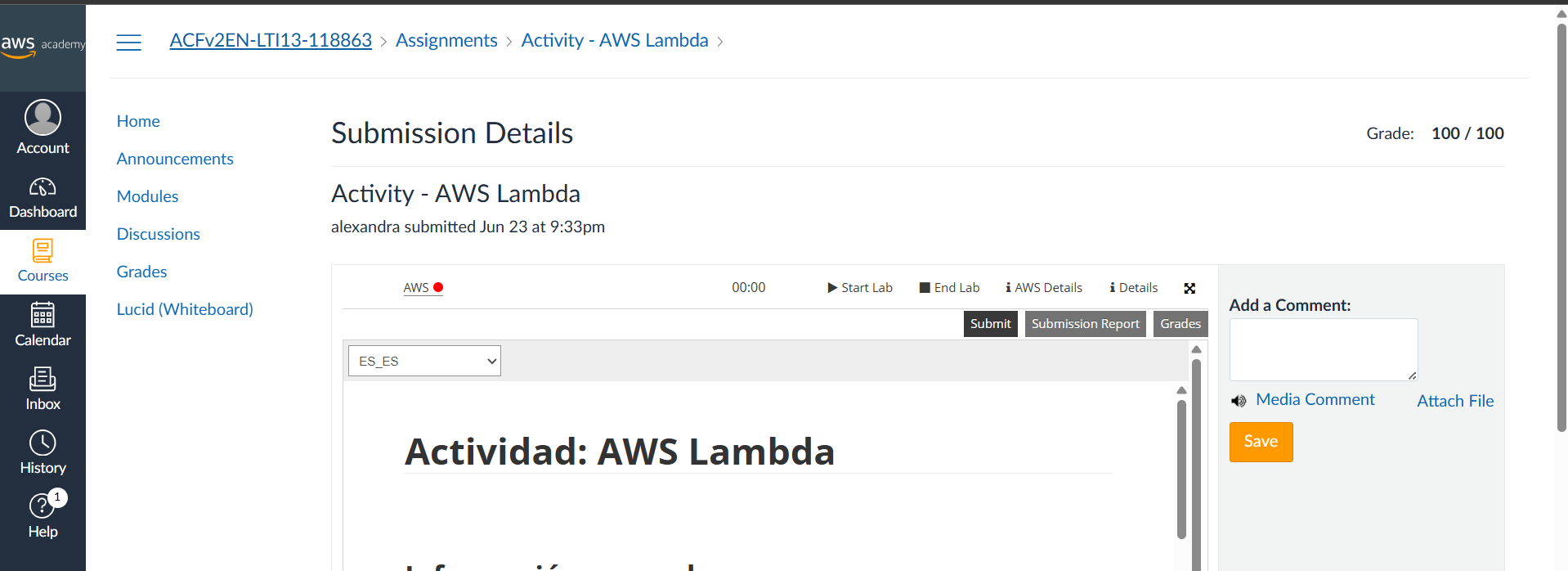
# **AWS VPC**

In this module, I learned how to create a complete network within AWS to host applications. I designed the network by dividing the servers between public and private zones, controlling how they connect to the internet and to each other. I also configured security rules to protect access. Finally, I launched a web server within that network and verified that it was working correctly. Through this exercise, I understood how to build secure and organized infrastructures in the cloud, similar to traditional physical networks, but taking advantage of the flexibility offered by AWS.



# **AWS Lambda**

In this module, I learned how to create a complete network within AWS to host applications. I designed the network by dividing the servers between public and private zones, controlling how they connect to the internet and to each other. I also configured security rules to protect access. Finally, I launched a web server within that network and verified that it was working correctly. Through this exercise, I understood how to build secure and organized infrastructures in the cloud, similar to traditional physical networks, but taking advantage of the flexibility offered by AWS.



# **AWS EBS**

Please add the result of module 7 case studies in the AWS assignment – part 2 with explain of the result in your portfolio