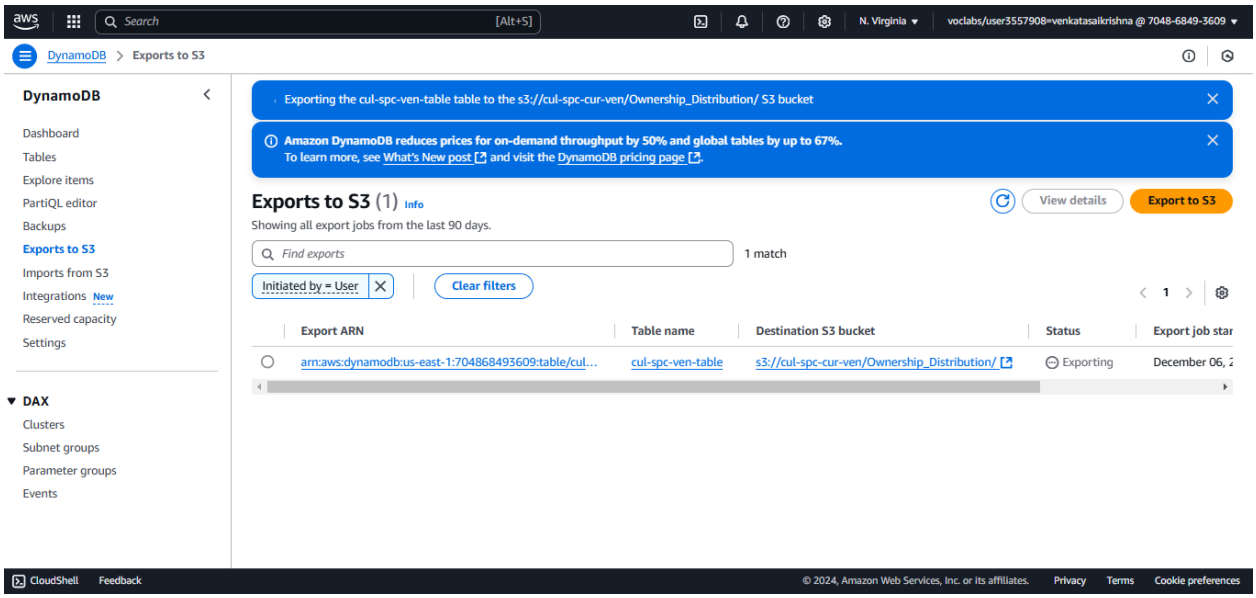
DAP Implementation (Individual work) – Venkata Sai Krishna Bhimavarapu

Step 5: Data Enriching

I focused on enriching the data to prepare it for further analysis. The first step was the creation of a DynamoDB database where I stored the data. It is scalable, flexible, and allows for the managing of big volumes of data. Once the database was populated, data was then exported to S3 for secure and cheap storage. Storage of the data in S3 also enabled direct analytics using AWS's analytics tools, such as AWS Glue and Athena, without necessarily having to transfer data across yet another database.

Figure 1

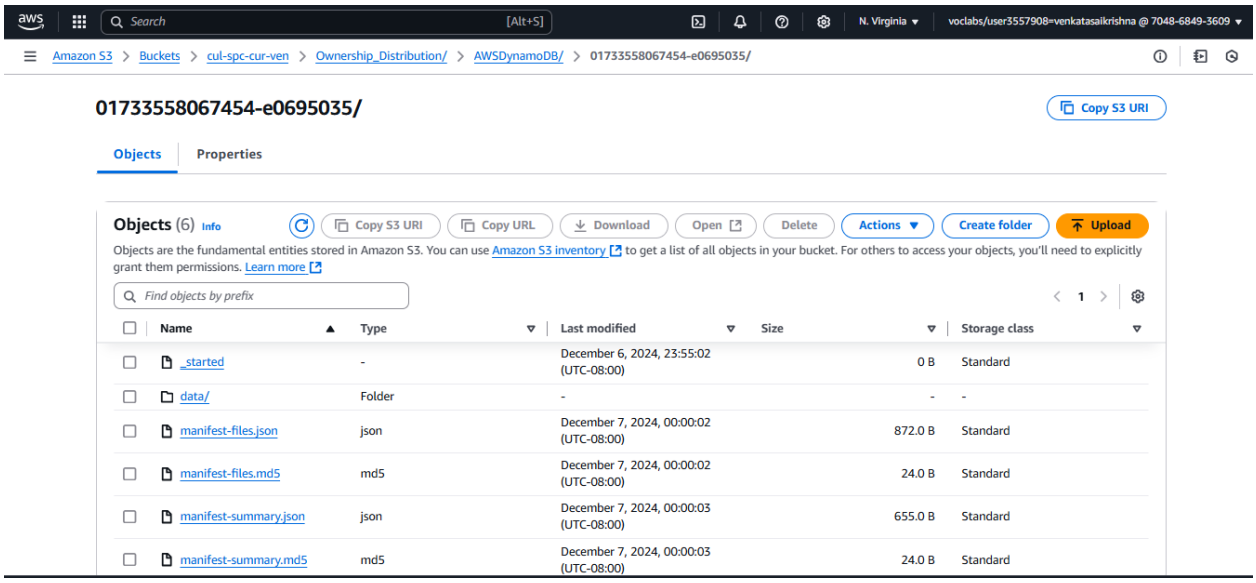
Exports to S3 from Dynamo DB



Note. Screenshot captured from AWS Console

Figure 2

Output of the exported objects

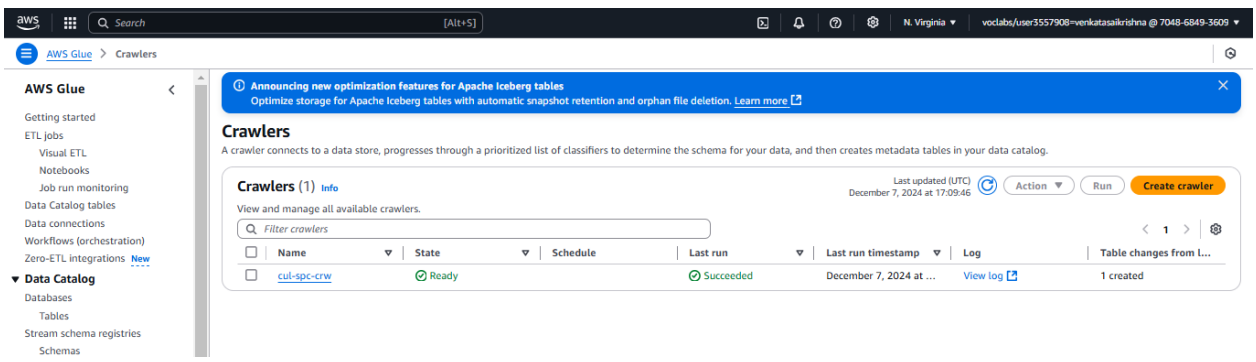


Note. Screenshot captured from AWS Console

Next, I created an AWS Glue crawler that scanned the data in the S3 bucket. The crawler automatically inferred the schema, structure, and format of the data and automatically populated the AWS Glue Data Catalog with metadata. This precluded the need for explicitly defining any schema, and the data was immediately query-ready. Running the AWS Glue crawler ensured the data was in a form that could be easily accessed for analysis.

.**Figure 3**

Crawler



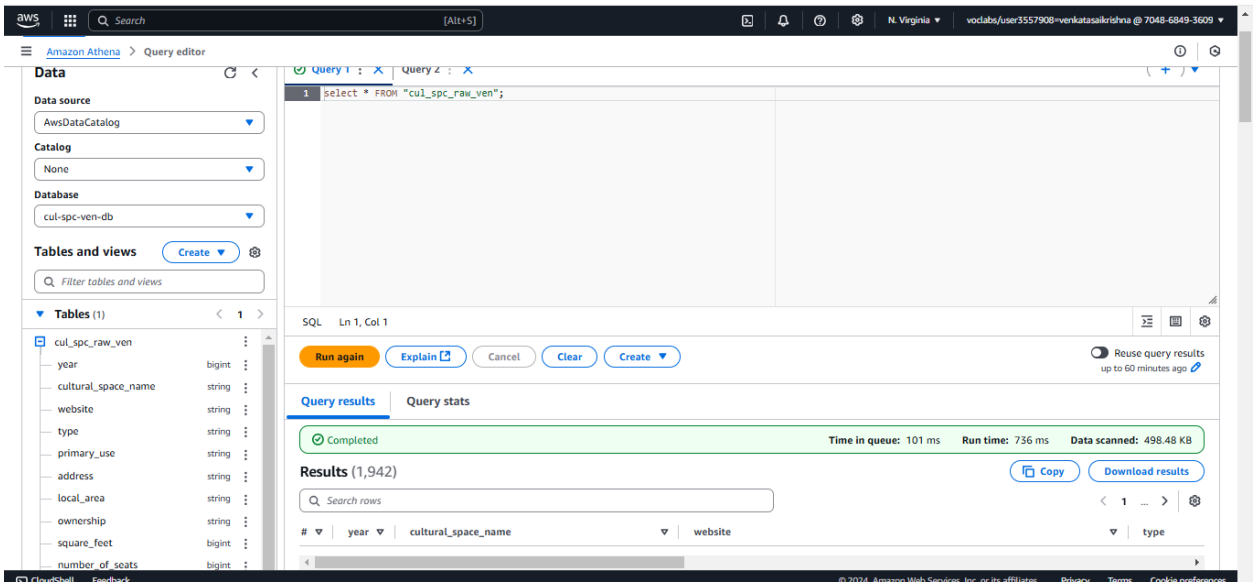
Note. Screenshot captured from AWS Console

Once the data was cataloged, I used AWS Athena to run SQL queries on the data stored in S3. With Athena, I analyzed the data using SQL, and it did not need to be loaded into a separate database system. The queries could be executed to fetch and process the data for certain analysis requirements.

More importantly, I made sure the integrity of the data was okay regarding completeness, consistency, and uniqueness. This was important to guarantee that the data analyzed would be accurate, reliable, and ready for insights. These together enabled me to enhance the data so that it became more useful for analysis and subsequent decision-making.

Figure 4

SQL Query execution in AWS Athena



Note. Screenshot captured from AWS Console

Step 6: Data Protection

Encryption Techniques for Confidentiality

Encryption plays a critical role in ensuring data confidentiality. For this implementation, we use AWS Key Management Service (KMS), which enables easy and secure management of cryptographic keys.

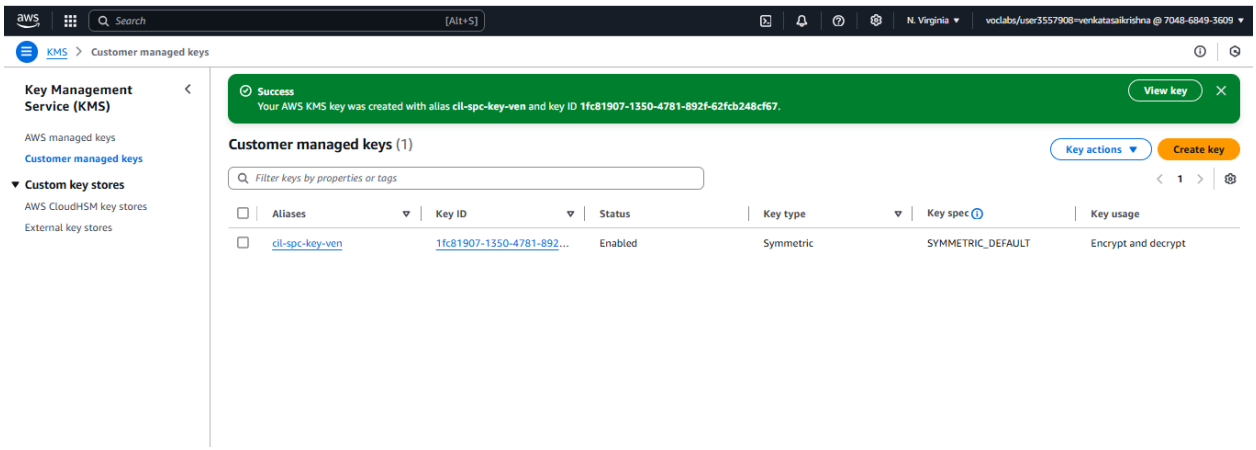
I have created a symmetric encryption key in AWS KMS. This is a symmetric key, hence very efficient since the same key would be used during the encryption and decryption processes. Using this key, I can then encrypt data at rest that is sensitive and decrypt when needed so that it doesn't remain in plain text without protection from unauthorized access.

AWS KMS not only makes key management a breeze but also works in tandem with other AWS services to provide end-to-end encryption capabilities while sticking to strict security standards. This setup ensures that industry best practices are followed to keep data confidential.

Encrypting the data within the AWS services, such as S3, ensures that sensitive information is kept out of reach and that only authorized users or systems can access it.

Figure 5

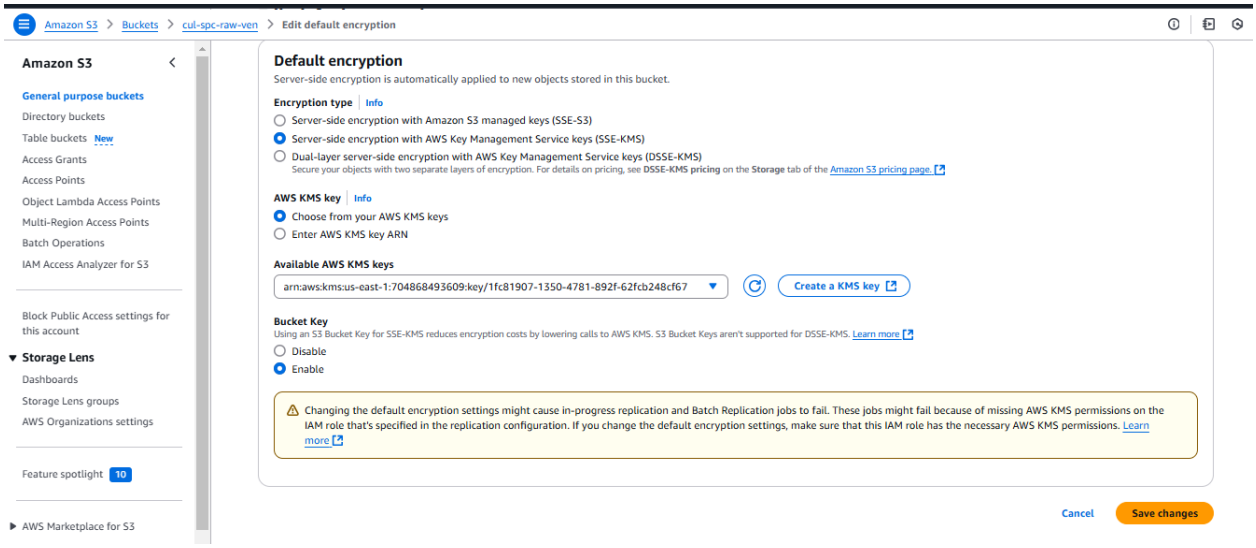
Customer Managed Keys



Note. Screenshot captured from AWS Console

Figure 6

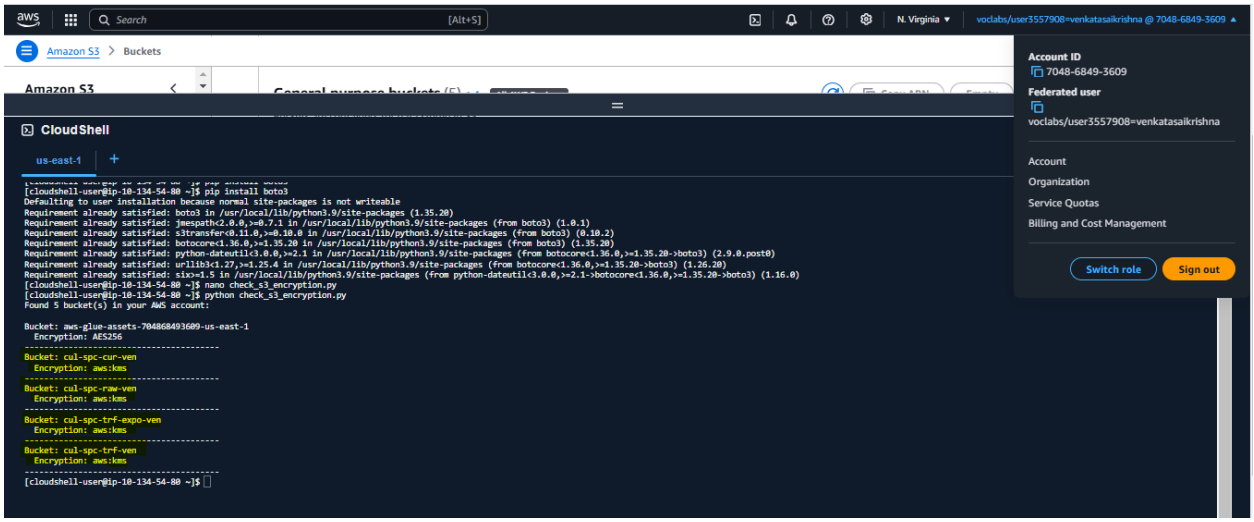
Bucket level encryption



Note. Screenshot captured from AWS Console

Figure 7

Status of all buckets



Note. Screenshot captured from AWS Console

Integration by Means of Bucket Versioning

As a way of adding data reliability and integrity, I have introduced versioning in the AWS S3 bucket. Bucket versioning is that form of qualitative functionality that allows us to maintain multiple versions of the objects stored in an S3 bucket. This ensures that no data is accidentally overwritten or deleted, as every change made in an object gets saved as a new version.

By enabling versioning, we achieve the following:

1. Data Recovery: In case an object gets accidentally deleted or updated, we can recover its previous versions smoothly.

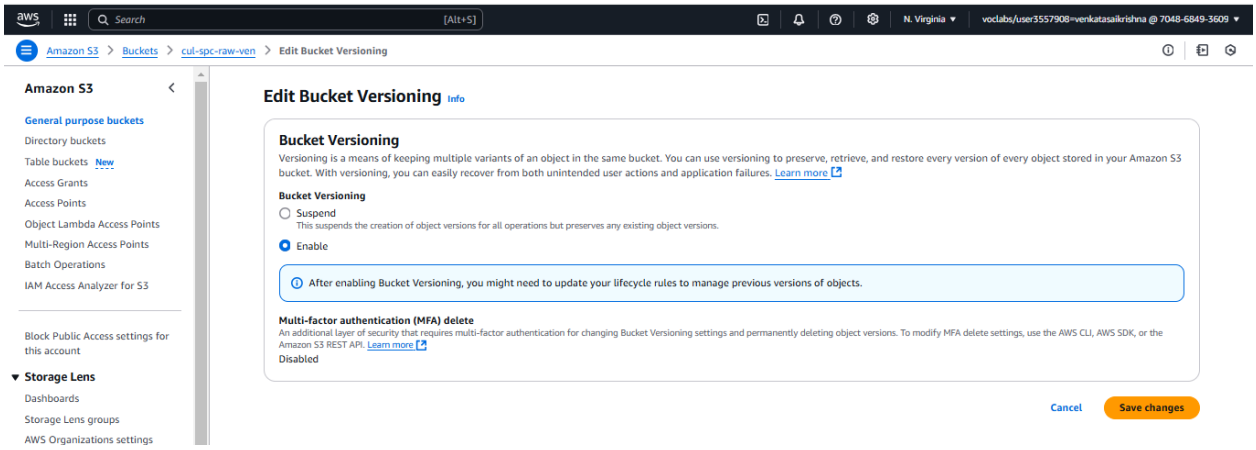
2. Audit Trail: Maintaining many versions of data provides a very clear history of changes, hence very useful for audit and compliance purposes.

3. Protection against Errors: Helps mitigate risks caused by human errors or application bugs that may overwrite or delete critical data.

Bucket versioning also befits our goal of ensuring that data is durable and available. This setup, combined with encryption, strengthens the security and resilience of the data pipeline, ensuring that cultural or sensitive datasets remain safeguarded and version-controlled for future use or reference.

Figure 8

Enabling bucket versioning



Note. Screenshot captured from AWS Console

Availability through Replication Techniques

I can implement replication in AWS S3 to ensure the high availability and durability of data. Replication is a robust mechanism that facilitates the automatic and asynchronous copying of objects from one bucket (source) to another bucket (destination), potentially across different AWS regions.

I have created some backup buckets in this regard for replication purposes. I also configured the replication rules so that the data in the source bucket gets seamlessly copied to the destination bucket. This provides a number of advantages those are mentioned below

1. Disaster Recovery: Since the data is replicated in another region, it may be accessed even when there's a failure in the primary hosting region.

2. Increased Accessibility: Data to be replicated can be then fetched from a local region, improving response times for users or an application.

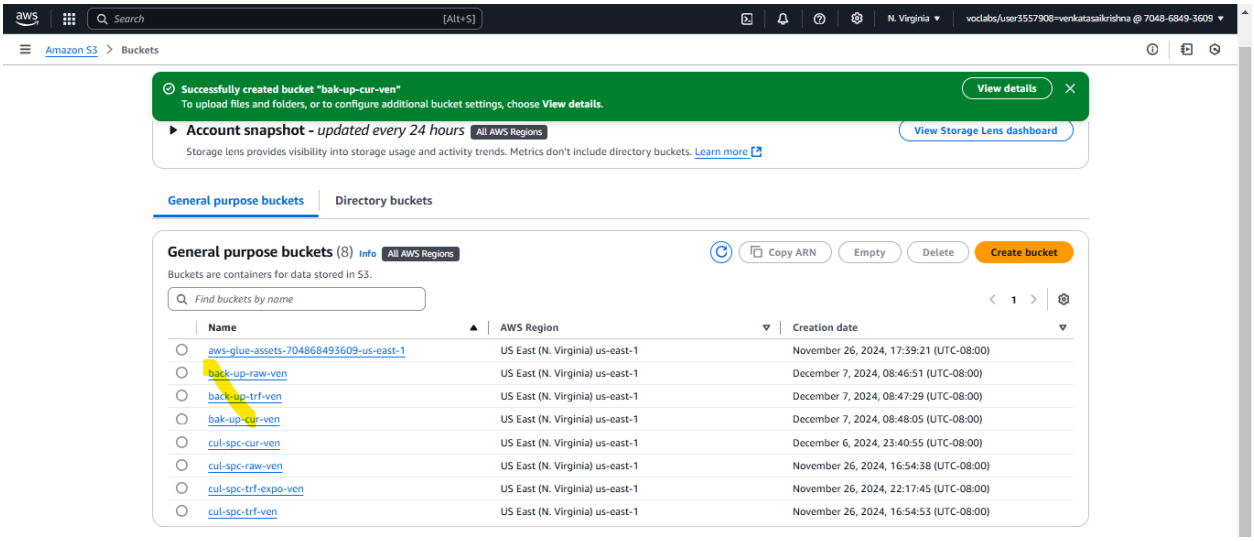
3. Compliance and Data Residency: Helps in meeting legal or compliance requirements by storing copies of data in specific geographic locations.

4. Improved Redundancy: I can make sure that a backup of critical data is always available, safeguarding against accidental deletions or corruption.

Being in a position to create replica buckets and use replication rules can indeed guarantee that datasets are safe and highly available for access, analysis, and recoveries. This will therefore strengthen the general reliability and accessibility of the data pipeline, ensuring that there will be no interruptions of services and better service delivery.

Figure 9

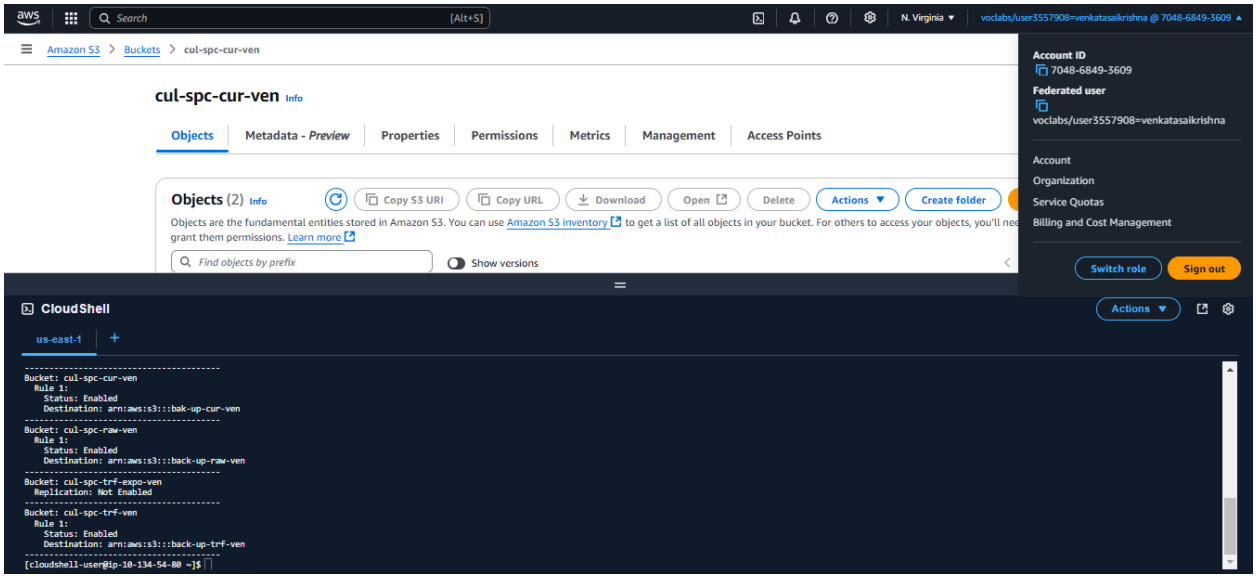
Replicated Buckets



Note. Screenshot captured from AWS Console

Figure 10

Enabled replication rules for all buckets



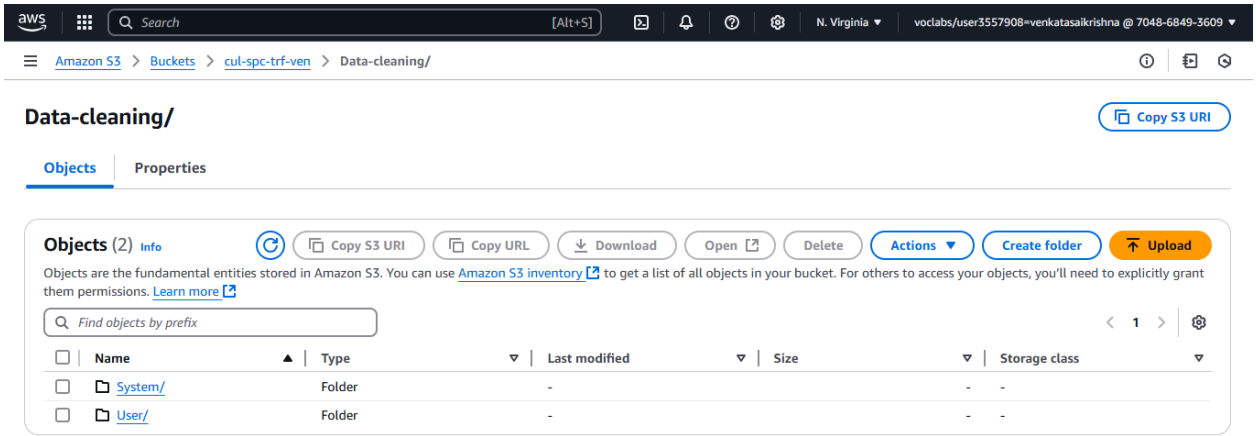
Note. Screenshot captured from AWS Console

Step 7: Data Governance

Here, I will use the cleaned dataset from Part 1 of my project. The dataset is in the system folder, and the bucket name is: "cul-spc-trf-ven." Since the dataset has already been cleaned, the next step involves additional checks to make sure data is protected and of quality. This includes the protection of sensitive information and the treatment of any PII in the dataset to maintain compliance and data security.

Figure 11

Source



Note. Screenshot captured from AWS Console

I am going to utilize AWS Glue for the construction of the ETL pipeline that would make sure the dataset is of high quality. It is another essential step for maintaining the

integrity of the information, confidentiality, and preparedness for advanced analysis of the dataset.

Here's why I am doing this

1. To Safeguard Sensitive Information: Identifying sensitive information like PII and taking due care by either masking or encrypting to keep it compliant with the Data Protection Acts.

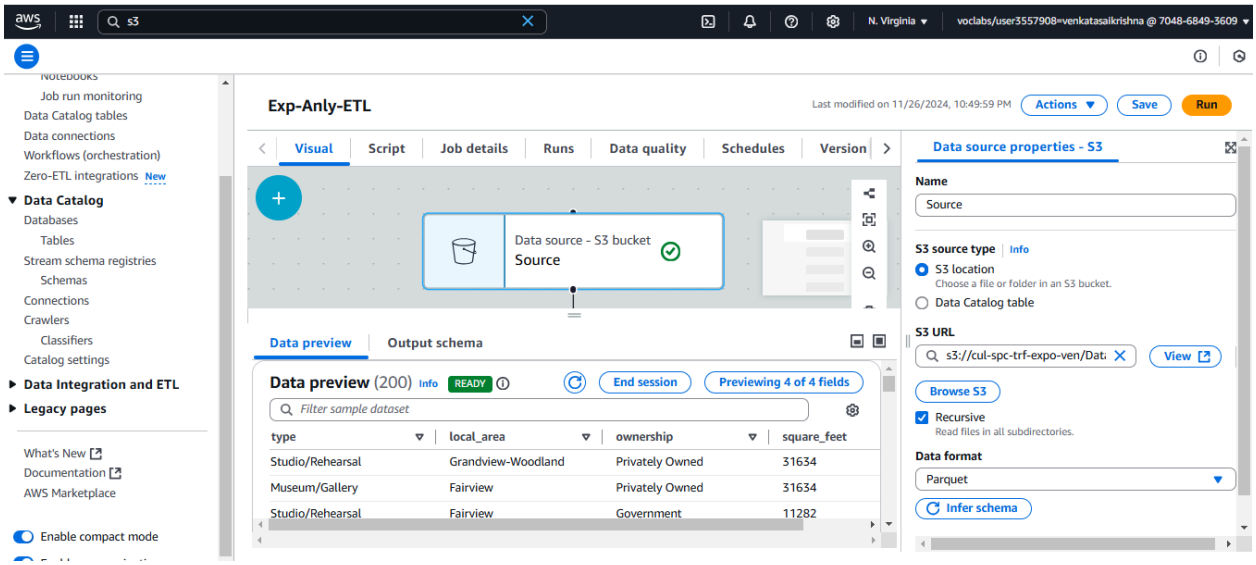
2. To Analyze Data Quality: It needs to be fresh, complete, and unique to become a quality dataset that assists in reliable decision-making.

3. To Organize Processed Data: Transferring the data that has been validated and enriched to a [data-quality] folder in the same S3 bucket keeps my data organized for reference.

This pipeline will take in as its source the dataset created in Part 1, and put its processed data into `s3://cul-spc-trf-ven/[data-quality]/`. This therefore warrants that only quality, securely gated data will be delivered at each subsequent step throughout the project. Next is to create the said pipeline.

Figure 12

Source dataset in ETL Pipeline



Note. Screenshot captured from AWS Console

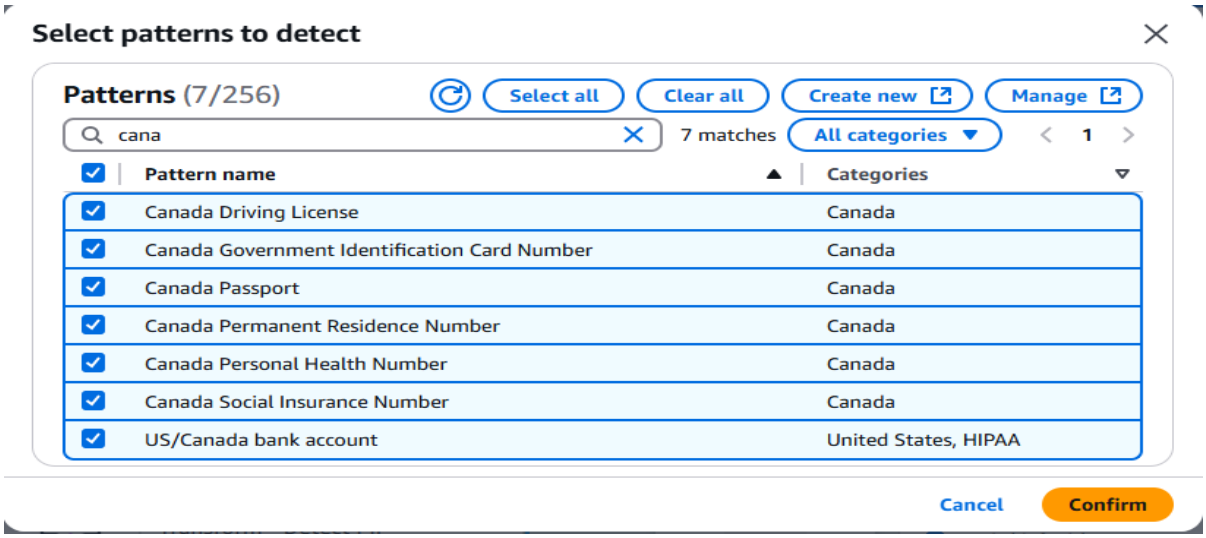
My aim is to find out which area has what type of sculptures and who owns them, with a keen interest in knowing who owns the most cultural sculptures. The data cleaning is done, and it consists of three columns: Type, Area, and Ownership and square feet. This data is further analyzed to show ownership patterns, classify sculptures according to their type and area, and determine the owners with the most cultural sculptures. In this way, it can show the trend and insights about sculpture distribution and ownership across different areas.

Detecting Sensitive Data

In this step, I'll check whether the dataset contains sensitive information. More specifically, I look for patterns that could include sensitive data such as passport numbers, phone numbers, or any other PII that may apply within the context of Canada. My dataset does not include sensitive data, so I move to the next step.

Figure 13

Detecting Sensitive Data



Note. Screenshot captured from AWS Console

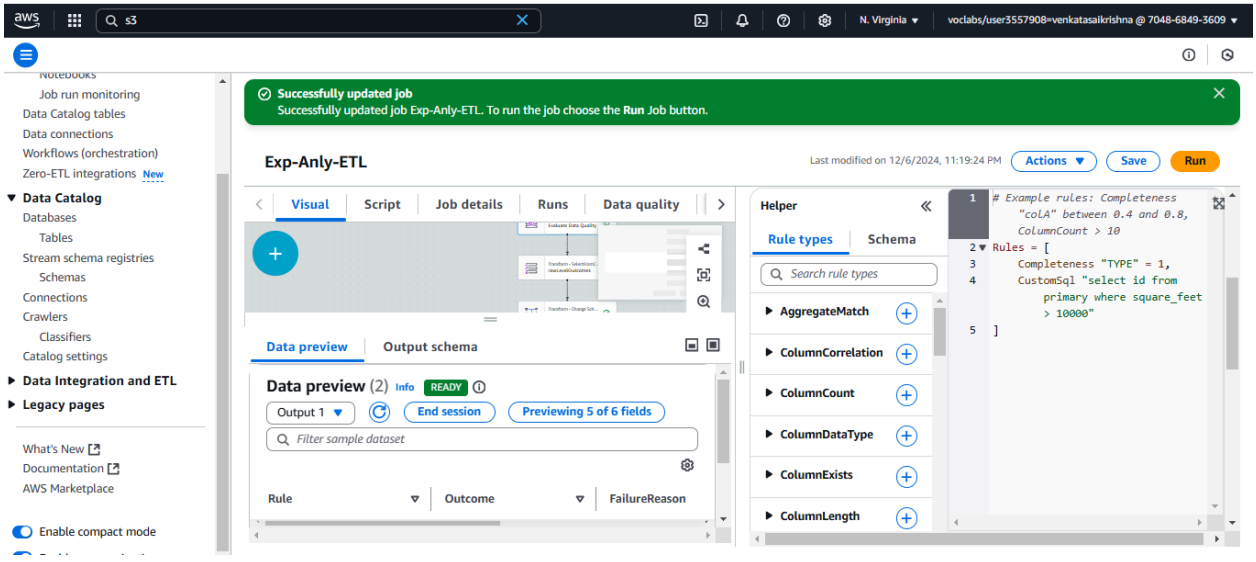
Data Quality Evaluation

This step is going to involve assessing the quality of the data according to the columns [Type, Location, Ownership, SquareFeet]. I will check for completeness and ensure

that the data is bound within specific criteria. I want to see if the range of Square Feet is large enough and analyze the data from an area size perspective. More precisely, I will analyze what kind of cultural spaces are represented in these areas, focusing particularly on those that have areas over 10,000 square feet.

Figure 14

ETL Pipeline for data quality analysis



Note. Screenshot captured from AWS Console

Here, I will be using the Custom Sql rule to check for land areas greater than 10,000 square feet. The SQL query I'll apply is:

Sql-query

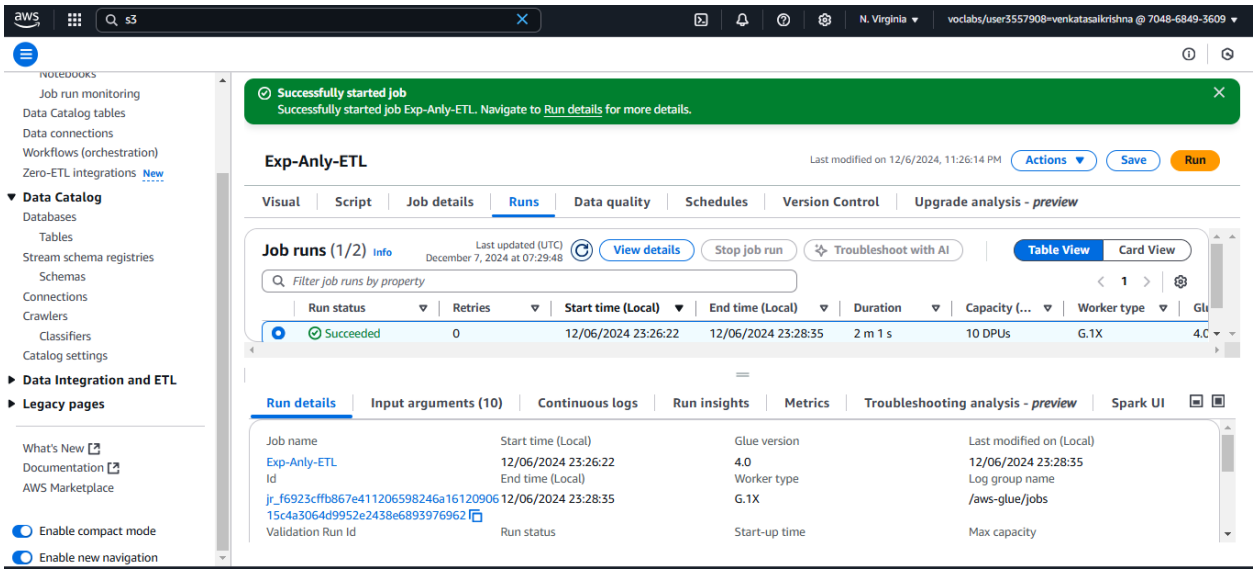
SELECT id FROM primary WHERE square_feet > 10000;

The data will be sorted based on the land area: I use the Completeness rule to make sure that in column TYPE, the value is complete-1.

Next, I will segregate the data into two separate folders, depending on whether the square_feet value is greater than or less than 10,000. To do this, I will use Conditional Routing to route the records into two different folders: one with land areas greater than 10,000 square feet and another for those smaller than 10,000 square feet. Finally, both pieces of data will be forwarded to their respective destinations for further processing.

Figure 15

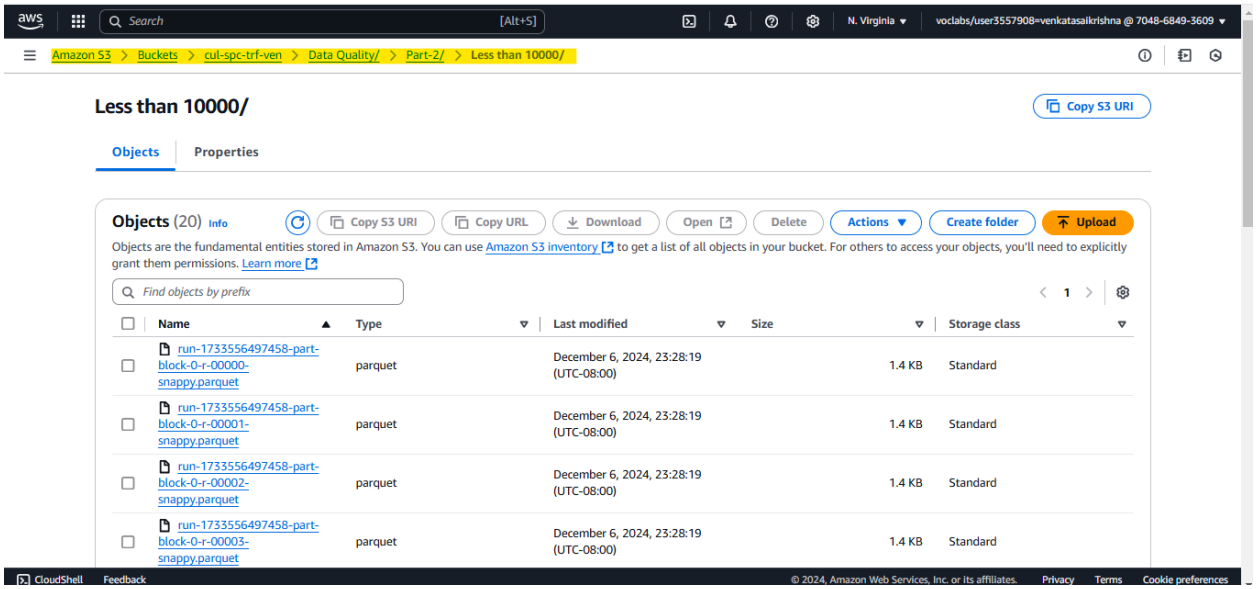
Successful job run



Note. Screenshot captured from AWS Console

Figure 16

Output of data quality objects

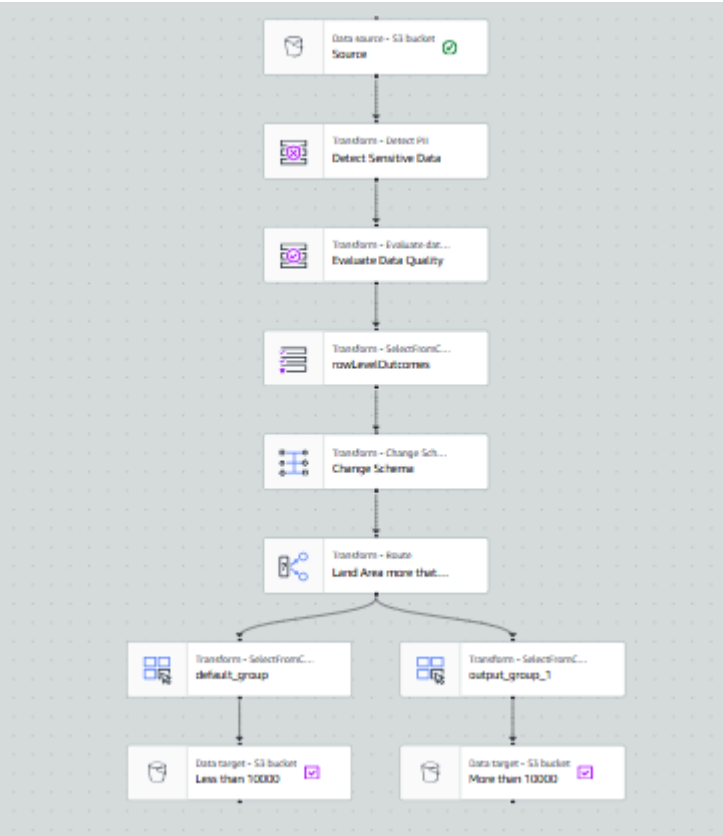


Note. Screenshot captured from AWS Console

Finally, the cleaned and analyzed data, devoid of sensitive information and appropriately categorized, is delivered to the Data Quality Data folder for further analysis or processing, ensuring that data is safe, compliant, and of a high quality prior to subsequent steps in the analytics pipeline.

Figure 17

The pipeline



Note. Screenshot captured from AWS Console

Step 8: Data Observability

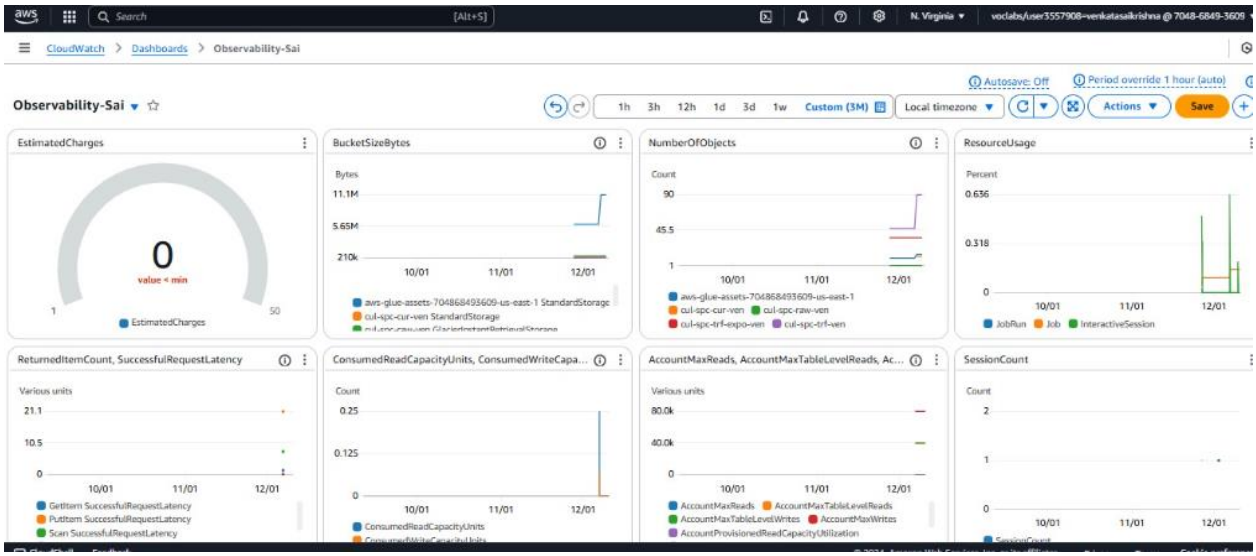
Data observability refers to the capability to observe and understand the health, quality, and flow of data throughout its life cycle. It involves monitoring key metrics, such as volume, quality, and freshness of data, and the detection of anomalies and issues in real time. Key aspects are monitoring, lineage tracking, quality assurance, anomaly detection, and auditing data access. It helps organizations ensure reliable, accurate data, thus enabling faster problem detection and better decision-making.

I will utilize AWS CloudWatch to observe the health and performance of the services that were implemented in Part-1 and Part-2 of the project. CloudWatch provides me a way to collect and track metrics, collect and track log data, and set alarms on anomalous activity. Further, I will develop a CloudWatch dashboard for visualization to view the most important

key metrics and data from all the implemented services, ensuring a centralized overview of the whole data analytics platform. This will further enable me to ensure the smooth operation of the system, quickly detecting the root cause and resolving problems.

Figure 18

Dashboard



Note. Screenshot captured from AWS Console

I have prepared this dashboard in AWS CloudWatch to monitor different metrics about my AWS resources. By setting up these visualizations, I am able to track the estimated charges for my AWS services over time, which helps me in keeping an eye on my spending. Further, I am able to get the size of my S3 buckets in storage use and count of objects within such a bucket so that I can get how my storage is growing, hence able to manage. I can get the resource usage graph, which gives an overview of the performance monitoring of AWS resources like an EC2 instance or a Lambda function. Finally, combining the resource counts with the estimated charges lets me know how my used resources are adding to the overall cost. This dashboard gives full insight into the AWS environment and eases the burden of cost management, performance, and resource utilization.