**Secure Dataset Storage in Amazon S3 (Free Tier)**

# Objective

The goal of this task was to securely store a dataset in the cloud using Amazon Web Services (AWS), ensuring that both encryption and access control were enforced. To align with budget constraints, the implementation was designed to remain within the AWS Free Tier.

# **Infrastructure Overview**

**Service Used:** Amazon S3 (Simple Storage Service)

**Encryption Method:** Server-Side Encryption with Amazon S3-managed keys (SSE-S3)

**Access Management:** IAM policies restricting access to authorized users only

**Region:** Asia Pacific (Mumbai) — ap-south-1

**Dataset:** CSV file uploaded as a test object

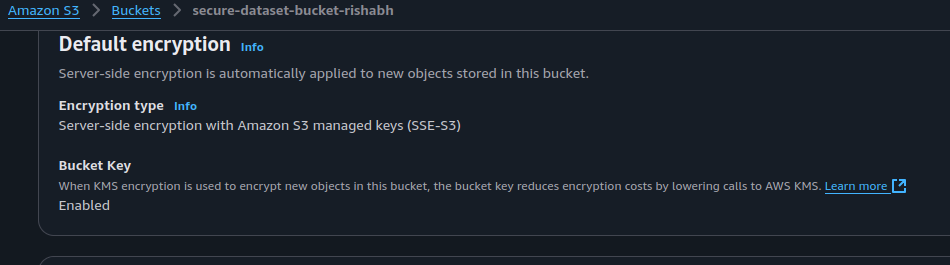
# **Security Implementation**

## **1. S3 Bucket Creation**

I created a uniquely named S3 bucket (secure-dataset-bucket-rishabh) through the AWS Console, with default configurations. To ensure baseline privacy, I confirmed that block public access settings were fully enabled at the time of creation, preventing any accidental public exposure.

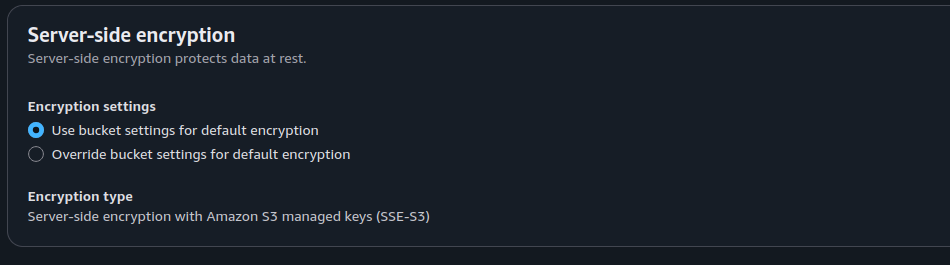
## 2. Encryption at Rest

To enforce encryption without incurring additional charges (as would be the case with AWS KMS), I enabled server-side encryption using Amazon S3-managed keys (SSE-S3). This automatically applies AES-256 encryption to all objects uploaded to the bucket, ensuring that the dataset remains encrypted at rest.



## 3. Dataset Upload

I uploaded a sample dataset (Dataset.csv) through the AWS Console. During the upload process, I ensured that encryption remained active and that all metadata properties were appropriately inherited from the bucket configuration.

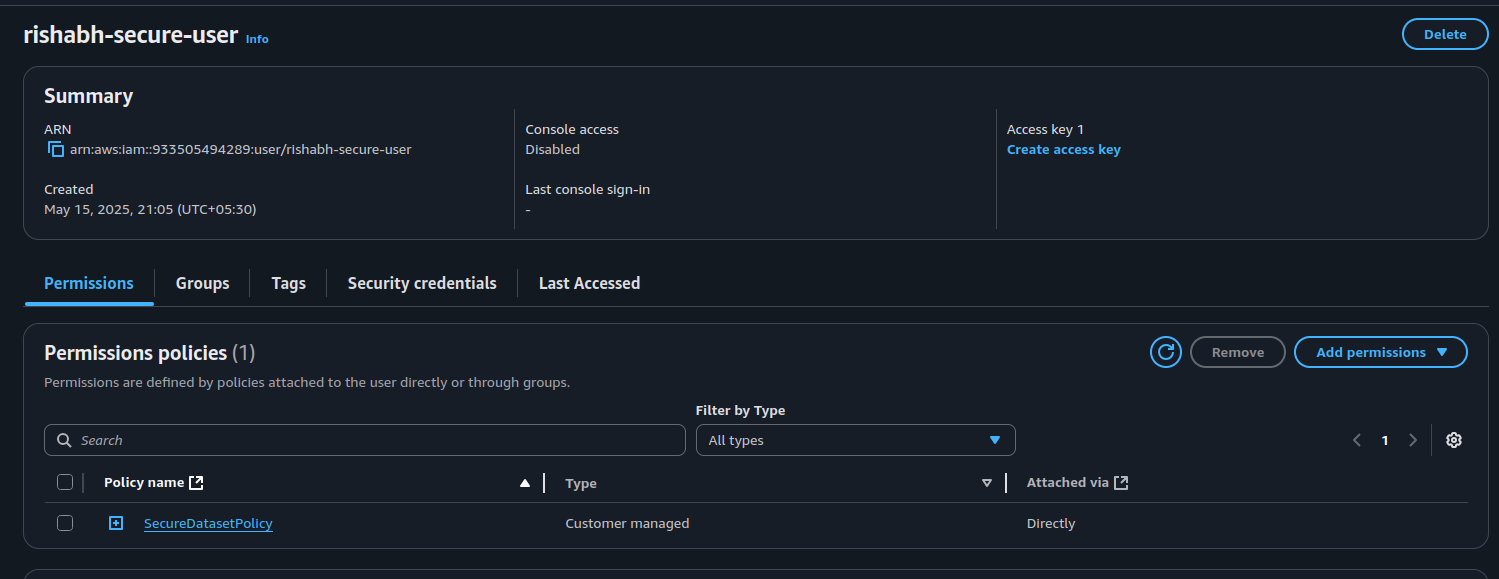


## 4. Access Control via IAM

To maintain fine-grained control over who can access the dataset, I created a dedicated IAM user (rishabh-secure-user). Instead of using Amazon's managed policies, I defined a custom IAM policy that granted read/write access only to the specific bucket and its contents.

The custom policy followed the principle of least privilege:

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Sid": "AllowLimitedBucketAccess",  
 "Effect": "Allow",  
 "Action": ["s3:GetObject", "s3:PutObject"],  
 "Resource": [  
 "arn:aws:s3:::secure-dataset-bucket-rishabh",  
 "arn:aws:s3:::secure-dataset-bucket-rishabh/\*"  
 ]  
 }  
 ]  
}



This policy was then attached to the IAM user, ensuring that only authenticated and explicitly authorized access to the dataset is allowed.

## 5. Public Access Blocking

I verified that all public access settings remained blocked after upload. This included settings for:  
- Blocking public ACLs  
- Blocking public bucket policies  
- Restricting public access via policies  
This measure ensures that no object in the bucket can be accessed by anonymous users under any circumstance.

## 6. Versioning

In addition to the core security setup, I also enabled bucket versioning. This allows for future rollback or recovery of objects in the event of accidental overwrites or deletions, further improving the resiliency of the dataset storage mechanism.

# Outcome Summary

|  |  |  |
| --- | --- | --- |
| Feature | Implementation Details | Status |
| Bucket Created | secure-dataset-bucket-rishabh in ap-south-1 | Complete |
| Encryption at Rest | SSE-S3 (AES-256, managed by Amazon) | Enabled |
| IAM User Access | Custom IAM policy applied to specific bucket only | Secured |
| Public Access Restriction | All block public access settings enabled | Enforced |
| Dataset Upload | Sample dataset securely uploaded with encryption | Done |
| Versioning | Enabled for change tracking and rollback | Optional |