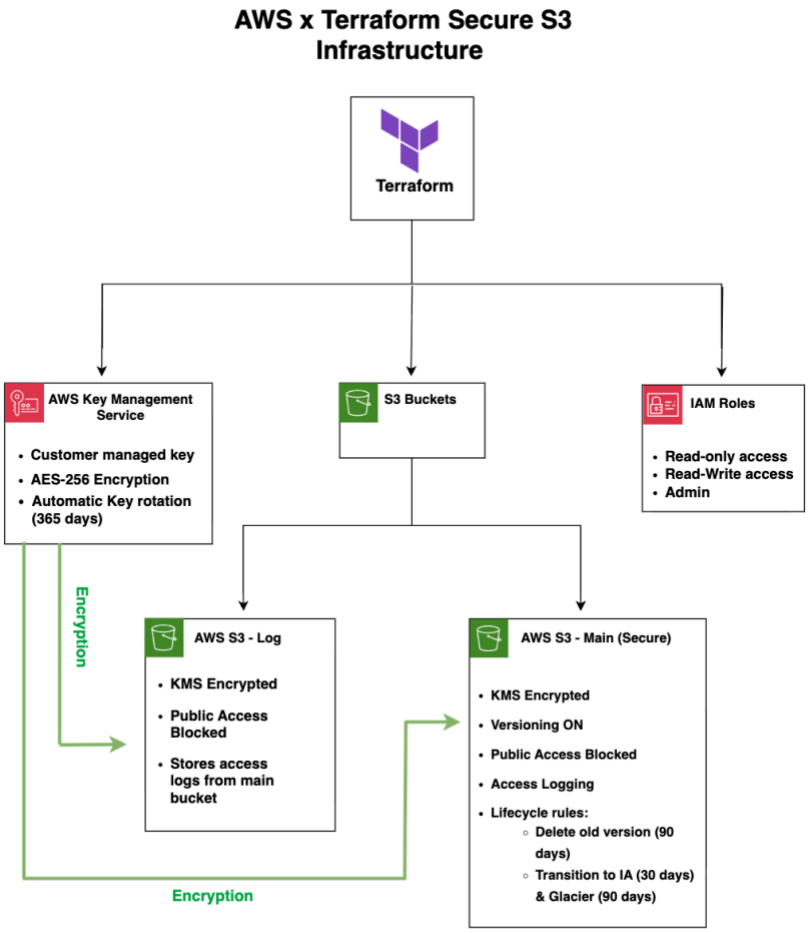
Secure S3 Bucket Infrastructure with Terraform

Overview

This project implements a secure S3 storage solution with KMS encryption, access controls, and audit logging. All infrastructure is created using Terraform, making set ups repeatable, testable, version controlled, automatable, and cost-efficient.

Architecture



Security Features:

Encryption

- **KMS Customer-Managed Keys**: Full control over encryption keys
- **Automatic Key Rotation**: Annual rotation enabled for compliance
- **S3 Bucket Keys**: 99% reduction in KMS request costs
- **Encryption at Rest**: All data encrypted using AES-256 via KMS

Access Controls

- **Least Privilege IAM Roles**: Three distinct roles with minimal permissions
- **Read-Only**: List and read objects only
- **Read-Write**: Upload and read, but cannot delete
- **Admin**: Full access (tightly controlled)
- **Instance Profiles**: Secure role assumption for EC2 instances

Audit & Compliance

- **Access Logging**: All bucket access logged to separate log bucket
- **Versioning**: Object versions preserved for recovery and compliance
- **Lifecycle Policies**: Automated data retention and cost optimization
- **Separation of Concerns**: Logs stored in dedicated bucket

Data Protection

- **Versioning**: Protects against accidental deletion or malicious changes
- **Lifecycle Management**: Old versions deleted after 90 days & Data transitions to cheaper storage classes (IA after 30 days, Glacier after 90 days)

Prerequisites

- AWS Account with appropriate permissions
- AWS CLI installed & configured with credentials
- Terraform version >= 1.0
- Basic understanding of AWS IAM and S3

Deployment Instructions:

1. Clone and Configure

```bash
# Navigate to project directory
cd secure-s3-terraform

# Initialize Terraform
terraform init
```

2. Review the Plan

```bash
# Preview what will be created
terraform plan
```

3. Deploy Infrastructure

```bash
# Create all resources
terraform apply

# Type 'yes' when prompted
```

4. Verify Deployment

```bash
# List your buckets
aws s3 ls

# Check encryption settings
aws s3api get-bucket-encryption --bucket <your-bucket-name>

# List IAM roles created
aws iam list-roles --query 'Roles[?contains(RoleName, `s3-bucket`)].RoleName'
```

Outputs

After deployment, Terraform should display:

```
bucket_name = "my-secure-bucket-xxxxxxxx" bucket_arn =
"arn:aws:s3:::my-secure-bucket-xxxxxxxx" log_bucket_name = "my-log-bucket-
xxxxxxxx" kms_key_id = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" kms_key_arn
= "arn:aws:kms:us-east-1:ACCOUNT:key/KEY_ID" s3_read_only_role_arn =
"arn:aws:iam::ACCOUNT:role/s3-bucket-read-only-role" s3_read_write_role_arn =
"arn:aws:iam::ACCOUNT:role/s3-bucket-read-write-role" s3_admin_role_arn =
"arn:aws:iam::ACCOUNT:role/s3-bucket-admin-role"
```

Cost Estimation

Service	Monthly Cost (estimate)
KMS Key	~$1.00
S3 Storage (first 50GB)	~$1.15
S3 Requests	~$0.01
Total	**~$2.16/month**

* Costs vary based on actual usage and data stored

Cleanup

To destroy all resources and avoid AWS charges:

```bash
# Preview what will be deleted
terraform destroy --dry-run

# Delete all resources
terraform destroy

# Type 'yes' when prompted
```

Warning: This permanently deletes all buckets and their contents. Make sure you have backups if needed.

DO NOT FORGET to "Terraform destroy" after completion of project

What I Learned

Through this project, I gained hands-on experience with:

- **Infrastructure as Code**: Writing Terraform configurations with HCL
- **AWS Security Services**: KMS, IAM, S3 security features
- **Access Control**: Implementing least-privilege IAM policies
- **Encryption**: Customer-managed keys vs AWS managed keys
- **Compliance Requirements**: Logging, versioning, and audit trails
- **Cost Optimization**: S3 lifecycle policies and storage classes
- **Security Best Practices**: Defense in depth, separation of concerns, least-privileges