# Lab 19 – Production-Grade Secure S3 Bucket (Versioning, KMS Encryption, Access Logging)

**Creator:** Sandeep Kumar Sharma

---

# 🔍Scenario – Real Production Requirement

Your company is building a data ingestion pipeline and needs a **highly secure S3 bucket** for storing critical data. The compliance team has shared strict security requirements:

### The S3 bucket must:

- **Enable versioning** to protect against accidental deletion or overwrites.
- Use **SSE-KMS encryption** with a customer-managed KMS key.
- **Block all public access**.
- **Enable access logging** to a dedicated logging bucket.
- Support **bucket policies** that restrict access to specific IAM roles.

This is a very common real-world requirement for: - Financial data - Healthcare data - ML datasets - Compliance-heavy organizations

Your task: **Create a production-ready secure S3 bucket using Terraform.**

---

# 🕐Learning Objectives

- Understand how to create secure S3 buckets for production.
- Learn how to enable versioning and KMS encryption.
- Learn how to configure S3 access logging.
- Learn how to create KMS keys with proper policies.
- Learn how to attach security-focused S3 bucket policies.

---

# 🔗Learning Outcomes

By the end of this lab, the learner will be able to: - Deploy a secure S3 bucket following AWS best practices. - Use Terraform to build KMS keys and S3 bucket policies. - Configure logging buckets for compliance. - Block public access at the bucket level. - Make S3 ready for production use.

---

# Concept Explanation (Natural Style)

S3 bucket creation in demos is usually simple:

```
resource "aws_s3_bucket" "demo" {}
```

But **real-world S3 buckets are never created like this**.

Real companies require: - No public access - Encryption at rest (KMS) - Access logs stored safely - Versioning enabled - Bucket policies for controlled access

This lab shows how S3 is created **in actual production**.

---

# 🛠️Part 1 – Project Setup

```
mkdir terraform-lab19-secure-s3
cd terraform-lab19-secure-s3
```

Create files:

```
touch main.tf variables.tf outputs.tf
```

---

# 🛠️Part 2 – Terraform Code

Below is a complete production-grade S3 configuration.

## main.tf

```
provider "aws" {
  region = "ap-south-1"
}

# ------------------------------------------
# 1. KMS Key for S3 Encryption
# ------------------------------------------
resource "aws_kms_key" "lab19_kms" {
  description             = "KMS key for S3 bucket encryption"
```

```
  deletion_window_in_days = 10
  enable_key_rotation     = true
}

# Logging bucket (stores all access logs)
resource "aws_s3_bucket" "lab19_logging" {
  bucket = "lab19-logging-bucket-${random_id.suffix.hex}"
  force_destroy = true
}

resource "aws_s3_bucket_public_access_block" "logging_block" {
  bucket = aws_s3_bucket.lab19_logging.id

  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

# Random suffix to avoid bucket name conflict
resource "random_id" "suffix" {
  byte_length = 3
}

# ------------------------------------------
# 2. Main Secure S3 Bucket
# ------------------------------------------
resource "aws_s3_bucket" "lab19_main" {
  bucket = "lab19-secure-bucket-${random_id.suffix.hex}"
  force_destroy = false
}

# Block public access
resource "aws_s3_bucket_public_access_block" "main_s3_block" {
  bucket = aws_s3_bucket.lab19_main.id

  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

# Versioning
resource "aws_s3_bucket_versioning" "main_versioning" {
  bucket = aws_s3_bucket.lab19_main.id

  versioning_configuration {
    status = "Enabled"
```

```
  }
}

# Server-side encryption using KMS
resource "aws_s3_bucket_server_side_encryption_configuration" "main_encryption"
{
  bucket = aws_s3_bucket.lab19_main.id

  rule {
    apply_server_side_encryption_by_default {
      kms_master_key_id = aws_kms_key.lab19_kms.arn
      sse_algorithm     = "aws:kms"
    }
  }
}

# ------------------------------------------
# 3. Enable Access Logging
# ------------------------------------------
resource "aws_s3_bucket_logging" "main_logging" {
  bucket = aws_s3_bucket.lab19_main.id

  target_bucket = aws_s3_bucket.lab19_logging.id
  target_prefix = "lab19-logs/"
}

# ------------------------------------------
# 4. Bucket Policy (Allow Only IAM Role Access)
# ------------------------------------------
resource "aws_s3_bucket_policy" "lab19_policy" {
  bucket = aws_s3_bucket.lab19_main.id

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Deny"
        Principal = "*"
        Action = "s3:*"
        Resource = [
          "${aws_s3_bucket.lab19_main.arn}",
          "${aws_s3_bucket.lab19_main.arn}/*"
        ]
        Condition = {
          Bool = {
            "aws:SecureTransport" = "false"
          }
        }
```

```
      }
    ]
  })
}
```

---

# 🧪Step 3 – Initialize Terraform

```
terraform init
```

---

# 🧪Step 4 – Plan

```
terraform plan
```

Terraform shows: - KMS key creation - Logging bucket - Secure main bucket - Public access blocking - Versioning - Encryption - Logging configuration

---

# 🧪Step 5 – Apply

```
terraform apply
```

Type **yes**.

---

# 🔍Step 6 – Validate in AWS Console

### 🔧Go to S3 → Buckets

You will see: - `lab19-secure-bucket-xxxx` - `lab19-logging-bucket-xxxx`

### 🔧Check Versioning

Inside bucket → Properties → Versioning → Enabled

### 🔧Check Encryption

Properties → Default Encryption → KMS (your key)

### 🔧Check Block Public Access

All 4 options should be TRUE.

### 🔧Check Bucket Policy

It should deny non-HTTPS access.

### 🔧Check Logging

S3 → Logging Bucket → Look for `lab19-logs/` prefix.

---

# 🗑️Step 7 – Destroy (Optional)

```
terraform destroy
```

Type **yes**.

---

# 📝Summary

In Lab 19, you learned how production teams secure S3 using: - KMS encryption - Versioning - Access logging - Public access block - Bucket policies

This is exactly how companies configure S3 buckets for compliance and audit requirements.

In the next lab, we will configure a **Production-Ready RDS Database** inside the private DB subnet created in Lab 17.

---

**End of Lab 19 – Secure S3 Setup**