UNIT 3: Storage Service

AWS S3 (Simple Storage Service) – Quick Overview 🚀
AWS S3 (Simple Storage Service) is a scalable, durable, and secure object storage service used for storing and retrieving data. It is designed for high availability (99.99%) and 11 nines (99.999999999%) durability.

Key Features of S3

- Object Storage – Stores data as objects (files + metadata) in buckets.
- Unlimited Storage – No total storage limit; individual objects can be up to 5 TB.
- Security & Access Control – Uses IAM policies, Bucket Policies, ACLs, and Encryption for protection.
- Versioning – Keeps multiple versions of an object for backup & recovery.
- Lifecycle Policies – Automatically moves data between storage classes or deletes objects.
- Replication – Cross-Region Replication (CRR) & Same-Region Replication (SRR) for redundancy.
- Static Website Hosting – Can serve websites directly from S3.

Amazon S3 Use cases

- Backup and storage
- Disaster Recovery
- Archive
- Hybrid Cloud storage
- Application hosting
- Media hosting
- Data lakes & big data analytics
- Static website

Amazon S3 - Objects

- Objects (files) have a Key
- The key is the FULL path:
    - s3://my-bucket/my_file.txt
    - s3://my-bucket/my_folder1/another_folder/my_file.txt
- The key is composed of prefix + object name
    - s3://my-bucket/my_folder1/another_folder/my_file.txt
- There's no concept of "directories" within buckets

(although the UI will trick you to think otherwise)

- Just keys with very long names that contain slashes ("/")

- Object values are the content of the body:
    - Max. Object Size is 5TB (5000GB)
    - If uploading more than 5GB, must use "multi-part upload"
- Metadata (list of text key / value pairs – system or user metadata)
- Tags (Unicode key / value pair – up to 10) – useful for security / lifecycle
- Version ID (if versioning is enabled)

Amazon S3 – Security

- User-Based
    - IAM Policies – which API calls should be allowed for a specific user from IAM

- Resource-Based
    - Bucket Policies – bucket wide rules from the S3 console - allows cross account
    - Object Access Control List (ACL) – finer grain (can be disabled)
    - Bucket Access Control List (ACL) – less common (can be disabled)

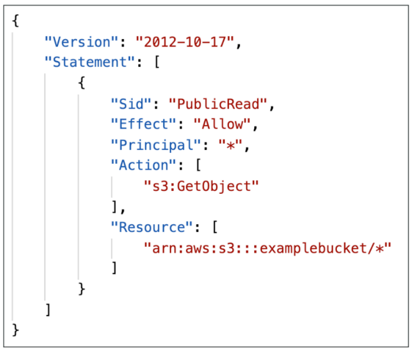
- Note: an IAM principal can access an S3 object if
    - The user IAM permissions ALLOW it OR the resource policy ALLOWS it
    - AND there's no explicit DENY

- Encryption: encrypt objects in Amazon S3 using encryption keys

S3 Bucket Policies

- JSON based policies
    - Resources: buckets and objects
    - Effect: Allow / Deny
    - Actions: Set of API to Allow or Deny
    - Principal: The account or user to apply the policy to

- Use S3 bucket for policy to:
    - Grant public access to the bucket
    - Force objects to be encrypted at upload
    - Grant access to another account (Cross Account)



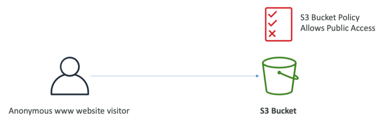
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket/*"
            ]
        }
    ]
}
```

Another Bucket Level Policy: which deny List/put/get objects to all the users except UserA

```
{
  "Version": "2012-10-17",
  "Id": "Policy1741273722082",
  "Statement": [
    {
      "Sid": "AllowUserA",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::854844598681:user/userA"
      },
      "Action": [
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::shrikant-demo-123",
        "arn:aws:s3:::shrikant-demo-123/*"
      ]
    },
    {
      "Sid": "DenyAllOthers",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::shrikant-demo-123",
        "arn:aws:s3:::shrikant-demo-123/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalArn": "arn:aws:iam::854844598681:user/userA"
        }
      }
    }
  ]
}
```

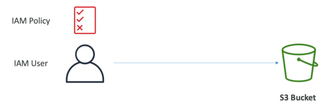
Public Access (Not IAM User):

Example: Public Access - Use Bucket Policy



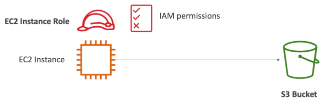
Access to IAM User

Example: User Access to S3 – IAM permissions

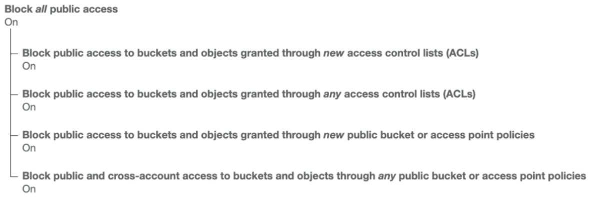


EC2 Accessing S3

Example: EC2 instance access - Use IAM Roles



Bucket settings for Block Public Access



Block *all* public access
On

- Block public access to buckets and objects granted through *new* access control lists (ACLs)
On
- Block public access to buckets and objects granted through *any* access control lists (ACLs)
On
- Block public access to buckets and objects granted through *new* public bucket or access point policies
On
- Block public and cross-account access to buckets and objects through *any* public bucket or access point policies
On

- These settings were created to prevent company data leaks
- If you know your bucket should never be public, leave these on
- Can be set at the account level

Amazon S3 - Versioning

- You can version your files in Amazon S3
- It is enabled at the bucket level
- Same key overwrite will change the “version”: 1, 2, 3….
- It is best practice to version your buckets
  - Protect against unintended deletes (ability to restore a version)
  - Easy roll back to previous version
- Notes:
  - Any file that is not versioned prior to enabling versioning will have version “null”
  - Suspending versioning does not delete the previous versions

S3 Storage Classes

- S3 Standard – High performance, frequently accessed data.
- S3 Intelligent-Tiering – Auto-moves data between storage classes based on access patterns.
- S3 Standard-IA (Infrequent Access) – Cheaper for less accessed data.
- S3 One Zone-IA – Lower cost, but stored in one availability zone.
- S3 Glacier – For long-term archival storage (retrieval in minutes to hours).
- S3 Glacier Deep Archive – Lowest cost, retrieval in hours.

Pricing Model

- Pay-as-you-go model based on:
  - Storage used (GBs/TBs)
  - Data retrieval & transfer
  - Number of requests (PUT, GET, DELETE, etc.)

- Versioning
- Cross-region replication
- Life Cycle Management
- Security & Encryption

Static Web-hosting with S3 bucket: Steps to Host a Static Website on S3

Events configuration on S3 buckets - Events configuration on S3 buckets

Events configuration on S3 buckets - Events configuration on S3 buckets

• Enabling cross-account access for S3
• S3 Data management and backup using 3rd Party applications.
• S3 Cross-Account Access and Pre-Signed URLs

Storage Gateway:

AWS **Storage Gateway** is a **hybrid cloud storage service** that connects your **on-premises** applications with AWS **cloud storage** (S3, EBS, Glacier). It enables seamless **data transfer** between your local environment and AWS for backup, archiving, disaster recovery, and hybrid cloud workloads.

Key Features

✅ **Extend On-Prem Storage to AWS** (For backup, disaster recovery, archiving)
✅ **Low-Latency Access to Cloud Storage** (Local caching speeds up access)
✅ **AWS Integration** (Works with S3, EBS, Glacier, FSx, etc.)
✅ **Encryption & Security** (Data is encrypted in transit and at rest)

Exercise:

Restrict Access to an S3 Bucket Using IAM Policy

Scenario

An employee should have **read-only access** to an S3 bucket, but should **not be able to delete files**.

Steps to Solve

1. **Create an IAM Policy**
   - Go to **IAM Console** → **Policies** → **Create Policy**
   - Use the following policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::shrikant-demo-123/*",
                "arn:aws:s3:::shrikant-demo-123"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Deny",
            "Action": "s3:DeleteObject",
            "Resource": [
              "arn:aws:s3:::shrikant-demo-123/*",
                "arn:aws:s3:::shrikant-demo-123"
            ]
        }
    ]
}
```

Attach the Policy to an IAM User

- Go to **IAM Users** → Select user → **Attach Policy**

Note : You Cannot Attach an IAM Policy to the Root User

AWS **does not allow** attaching **IAM policies** directly to the **root user**. However, you can **prevent the root user from deleting objects** using **Service Control Policies (SCPs)** if your AWS account is part of **AWS Organizations**.

Enable Cross-Region Replication (CRR) for Disaster Recovery

Scenario

You need to replicate objects from an S3 bucket in **us-east-1** to another bucket in **us-west-1**.

Steps to Solve

1. **Enable Versioning on Both Buckets**
   - Source bucket: my-source-bucket
   - Destination bucket: my-destination-bucket
2. **Set Up Replication Rule**
   - Go to **Management** → **Create Replication Rule**
   - Choose **Destination Bucket**
   - Select **IAM Role** or create a new one
   - Enable **Replicate Existing Objects**

Use S3 Lifecycle Policies to Move Files to Glacier

Scenario

A company wants to archive objects older than **30 days** to **S3 Glacier**. Configure an S3 Lifecycle Rule.

Steps to Solve

1. **Go to Lifecycle Configuration**
   - In **S3 Console**, select the bucket
   - Navigate to **Management** → **Create Lifecycle Rule**
2. **Define Rule**
   - Name: MoveToGlacierRule
   - Select **All objects in the bucket**
   - Set **Transition** → Move objects to **S3 Glacier after 30 days**
   - Save the rule and test it after 30 days

Enable Cross-Region Replication (CRR) for Disaster Recovery

Scenario

You need to replicate objects from an S3 bucket in **us-east-1** to another bucket in

You need to replicate objects from an S3 bucket in **us-east-1** to another bucket in **us-west-1**.

Steps to Solve

1. **Enable Versioning on Both Buckets**
   - Source bucket: my-source-bucket
   - Destination bucket: my-destination-bucket
2. **Set Up Replication Rule**
   - Go to **Management** → **Create Replication Rule**
   - Choose **Destination Bucket**
   - Select **IAM Role** or create a new one
   - Enable **Replicate Existing Objects**

Configure S3 Event Notifications to Trigger a Lambda Function

Scenario

You need to **trigger a Lambda function** whenever a new file is uploaded to an S3 bucket.

Steps to Solve

1. **Create an S3 Event Notification**
   - Go to **S3 Console** → Select the bucket
   - Navigate to **Properties** → **Event Notifications**
   - Create a new event for PUT operations
   - Choose **Lambda Function** as the destination
2. **Create a Lambda Function**
   - Go to **Lambda Console** → Create a function
   - Use the following Python code to print object details:

```
import json
def lambda_handler(event, context):
    print("New object uploaded:", json.dumps(event, indent=4))
```

lambda_handler is the entry point function that AWS Lambda automatically invokes when the function is triggered.

- The function receives **two parameters**:

1. **event** – Contains **data** about the event that triggered the Lambda function (e.g., an S3 file upload event, an API Gateway request, etc.).
2. **context** – Provides **runtime metadata** (e.g., function name, memory limit, request ID).

AWS **calls this function whenever the Lambda is triggered** (e.g., when an object is uploaded to an S3 bucket).

3. print("New object uploaded:", json.dumps(event, indent=4))

   - **print()** logs messages to **Amazon CloudWatch Logs**, helping with debugging.
   - "New object uploaded:" is a **simple log message**.
   - json.dumps(event, indent=4):
     - □ Converts the **event dictionary** to a **formatted JSON string**.
     - □ indent=4 makes it **more readable** (pretty-printed format).
   - This allows you to **see details** about the event in CloudWatch Logs.

3. **Test by Uploading a File**

Whats get printed :

```
{
    "Records": [
        {
            "eventVersion": "2.1",
            "eventSource": "aws:s3",
            "eventName": "ObjectCreated:Put",
            "s3": {
                "bucket": {
                    "name": "my-upload-bucket"
                },
                "object": {
                    "key": "image.jpg",
                    "size": 51234
                }
            }
        }
    ]
}
```

Output in CloudWatch Logs:

```
New object uploaded: {
    "Records": [
        {
            "eventVersion": "2.1",
            "eventSource": "aws:s3",
            "eventName": "ObjectCreated:Put",
            "s3": {
                "bucket": {
                    "name": "my-upload-bucket"
                },
                "object": {
                    "key": "image.jpg",
                    "size": 51234
                }
            }
        }
    ]
}
```

Modified Code to Log the Uploader's Identity

Modified Code to Log the Uploader's Identity

```
import json

def lambda_handler(event, context):
    # Extract S3 bucket and object key
    bucket_name = event["Records"][0]["s3"]["bucket"]["name"]
    object_key = event["Records"][0]["s3"]["object"]["key"]

    # Extract uploader identity (IAM user, role, or assumed identity)
    user_identity = event["Records"][0].get("userIdentity", {}).get("principalId",
"Unknown User")

    # Log message with uploader details
    print(f"New object uploaded to S3 bucket '{bucket_name}': '{object_key}' by
{user_identity}")

    # Pretty-print full event for debugging
    print("Event Details:", json.dumps(event, indent=4))
```

Configure S3 to Require MFA Delete

Scenario

To prevent accidental file deletions, configure **MFA Delete** for an S3 bucket.

Steps to Solve

1. **Enable MFA Delete** (via AWS CLI)
    - Run the following command:

```
aws s3api put-bucket-versioning --bucket my-secure-bucket --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa "SERIAL_NUMBER
MFA_CODE"
```

- Replace SERIAL_NUMBER and MFA_CODE with actual values

2. **Test by Trying to Delete a File**

Enable MFA for deleting objects(this can not be done via Console)

```
 aws s3api put-bucket-versioning --bucket shrikant-123-123-123 --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa
"arn:aws:iam::854844598681:mfa/root-account-mfa-device 121221"
```

Replace serial number with MFA device ARN Number and CURRENT MFA CODE
Your bucket Name -> shrikant-123-123-123

```
aws s3api delete-object \
    --bucket shrikant-123-123-123 \
    --key appl_stock.csv \
    --version-id 1NkKxoXSEnpy2xub3.0yFXGjEgXn0KqK \
    --mfa "arn:aws:iam::854844598681:mfa/root-account-mfa-device 704869"
```