Introduction to Amazon S3

Overview

Amazon S3 (Simple Storage Service) is a highly scalable, reliable, and low-latency data storage infrastructure that is a foundational component of AWS. It supports a vast range of use cases such as backup, disaster recovery, archiving, hybrid cloud storage, hosting applications, media storage, data lakes, big data analytics, software delivery, and static website hosting.

Key Features and Use Cases

- **Backup and Storage**: Store files, disk images, and backups.
- **Disaster Recovery**: Backup data to another region to ensure availability in case of regional failures.
- **Archival Storage**: Archive data at a lower cost and retrieve it when needed.
- **Hybrid Cloud Storage**: Extend on-premises storage to the cloud.
- **Application and Media Hosting**: Host applications, videos, images, etc.
- **Data Lakes and Big Data Analytics**: Store large amounts of data for analytics.
- **Software Updates Delivery**: Distribute software updates.
- **Static Website Hosting**: Host static websites directly from S3.

Real-World Examples

- **Nasdaq**: Stores seven years of data in S3 Glacier, Amazon's archival storage service.
- **Sysco**: Performs analytics on its data stored in S3 for business insights.

Amazon S3 Structure

Buckets

- **Buckets**: The top-level directories in S3, where data is stored.
- **Naming**: Buckets must have globally unique names (unique across all AWS regions and accounts).
  - Naming rules: No uppercase letters, no underscores, must be 3-63 characters long, cannot be an IP address, and must start with a lowercase letter or number.
- **Regions**: Buckets are created in specific AWS regions.

Objects

- **Objects**: Files stored in S3, identified by a unique key.
- **Object Key**: The full path to the object, which includes any "folder" structure.
  - Example: `my-folder1/another-folder/my-file.txt`
  - The key consists of a prefix (`my-folder1/another-folder`) and an object name (`my-file.txt`).
- **Object Value**: The content or body of the object (the actual data).
  - Maximum object size is 5 TB.
  - For files larger than 5 GB, multipart upload must be used.

Metadata, Tags, and Versioning

- **Metadata**: Key-value pairs associated with the object to provide additional information.
- **Tags**: Unicode key-value pairs (up to 10) for managing and organizing objects.
- **Versioning**: Allows multiple versions of an object to be stored, identified by version IDs (if versioning is enabled).

Summary

Amazon S3 is a versatile and powerful storage solution that supports a wide array of applications and use cases. Its structure of buckets and objects, along with features like metadata, tags, and versioning, makes it a critical component for data storage and management in the cloud.

Amazon S3 Security

User-Based Security

1. **IAM Policies**: Define which API calls are allowed for specific IAM users. These policies are associated with users to control their access to S3 resources.

Resource-Based Security

1. **S3 Bucket Policies**:
    - These are bucket-wide rules that can be assigned directly from the S3 console.
    - They allow specific users or accounts (cross-account access) to access S3 buckets.
    - Bucket policies are the primary method to make S3 buckets public.
2. **Object Access Control Lists (ACLs)**:
    - Provide finer-grained security controls at the object level.
    - Can be disabled if not needed.
3. **Bucket Access Control Lists (ACLs)**:
    - Less common than object ACLs.
    - Can also be disabled.

Determining Access

An IAM principal can access an S3 object if:

- The IAM permissions allow it.
- The resource policies allow it.
- There is no explicit deny action.

Encryption

- **Object Encryption**: Enhances security by encrypting objects using encryption keys.

S3 Bucket Policies

Structure of Bucket Policy

1. **Resource**: Specifies which buckets and objects the policy applies to. For example, `"Resource": "arn:aws:s3:::example-bucket/*"` applies to all objects in the "example-bucket".
2. **Effect**: Indicates whether the policy allows or denies the specified action (Allow/Deny).
3. **Action**: Specifies which API calls are allowed or denied. For example, `"Action": "s3:GetObject"` allows the GetObject action.
4. **Principal**: Specifies the user or account to which the policy applies. `"Principal": "*"` means it applies to anyone.

Example Policy for Public Access

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

This policy allows public read access to all objects in the "example-bucket".

Scenarios for S3 Access

1. **Public Access**
    - Users on the web can access files in the S3 bucket by attaching a public access bucket policy.
2. **IAM User Access**
    - Assign IAM permissions to users within your account, allowing them to access S3 buckets.
3. **EC2 Instance Access**
    - Use IAM roles instead of IAM users to grant EC2 instances access to S3 buckets.
    - Create an EC2 instance role with the appropriate IAM permissions.
4. **Cross-Account Access**

- Create a bucket policy that grants access to an IAM user from another AWS account.

Additional Security Settings

1. **Block Public Access Settings**
   - These settings provide an extra layer of security to prevent unintended public access.
   - Even if a bucket policy allows public access, enabling these settings will ensure the bucket remains private.
   - Can be set at both the bucket and account levels to prevent data leaks.

Summary

Amazon S3 security involves a combination of user-based and resource-based policies to control access. IAM policies manage permissions for individual users, while bucket policies handle broader access rules. Object and bucket ACLs offer more granular control, but bucket policies are generally preferred. Additional security can be enforced through encryption and block public access settings to prevent unintended data exposure.

Creating a Public Bucket Policy in Amazon S3

Step-by-Step Guide:

1. **Allow Public Access in Bucket Settings**:
   - Go to the **Permissions** tab of your S3 bucket.
   - Edit the settings to allow public access by unticking the block public access options.
   - Confirm this change. Be cautious as this can lead to potential data leaks if sensitive data is exposed.
2. **Review Access**:
   - Ensure that objects in the bucket can now be publicly accessible.
3. **Create a Bucket Policy**:
   - Scroll down to the **Bucket Policy** section, which will currently be empty.
   - You can use the **Policy Generator** or the **documentation** to create a policy.
4. **Using the Policy Generator**:
   - Open the **AWS Policy Generator**.
   - Select the type as **S3 Bucket Policy**.
   - Set the **Effect** to `Allow`.
   - Set the **Principal** to `*` (asterisk) to allow access to anyone.
   - For the **Action**, choose `GetObject` to allow read access to the objects in the bucket.

- The **Amazon Resource Name (ARN)** must be specified correctly. This includes the bucket name followed by `/*` to represent all objects within the bucket.
  - Copy the bucket ARN from the S3 console.
  - Append `/*` to the ARN to cover all objects.

5. **Example Policy**:

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

   - Replace `your-bucket-name` with the actual name of your bucket.

6. **Add and Save the Policy**:

   - Copy the generated policy into the **Bucket Policy** editor in the S3 console.
   - Ensure there are no extra spaces or errors in the policy JSON.
   - Save the changes.

7. **Confirm Public Access**:

   - Check under the **Permissions** tab that the bucket access is now set to public.
   - A warning will appear indicating the bucket is publicly accessible from the internet.

8. **Verify Public URL Access**:

   - Go to the object (e.g., `coffee.jpg`) in your bucket.
   - Copy the **public URL** provided by S3.
   - Paste the URL into a browser and refresh the page. The object should now be visible publicly.

By following these steps, you can successfully make your S3 bucket and its objects publicly accessible. This process involves changing the bucket permissions, creating a suitable bucket policy using the AWS Policy Generator, and verifying the changes by accessing the objects through their public URLs.

Hosting Static Websites on Amazon S3

Amazon S3 can be used to host static websites. These websites are comprised of static files such as HTML, CSS, and JavaScript, and can be accessed via a URL that is dependent on the AWS region where the bucket is created.

URL Formats

Depending on the AWS region, the URL format for accessing the static website will be:

- `http://your-bucket-name.s3-website-region.amazonaws.com`
- `http://your-bucket-name.s3-website.region.amazonaws.com`

The only difference is the presence of a dash or dot in the URL format, which is region-specific.

Steps to Host a Static Website on S3

1. **Create an S3 Bucket**:
    - Create a new S3 bucket that will contain your website files, such as HTML, CSS, JavaScript, and images.
2. **Upload Website Files**:
    - Upload all your website files to the newly created S3 bucket.
3. **Enable Static Website Hosting**:
    - Go to the **Properties** tab of your S3 bucket.
    - Under **Static website hosting**, select **Enable**.
    - Specify the index document (e.g., `index.html`) and an optional error document (e.g., `error.html`).
4. **Set Permissions for Public Access**:
    - To make your website publicly accessible, you need to set the appropriate permissions.
    - Ensure that the bucket has a policy that allows public read access.
5. **Create a Bucket Policy for Public Access**:
    - Go to the **Permissions** tab of your S3 bucket.
    - Create a bucket policy using the AWS Policy Generator or manually.
    - Example policy for public read access:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::your-bucket-name/*"
```

```
      }
    ]
}
```

- Replace `your-bucket-name` with the actual name of your bucket.
- Save the policy.

6. **Accessing the Website**:
   - Once the bucket policy is in place and the website hosting is enabled, you can access your static website using the provided S3 website URL.
   - If you encounter a 403 Forbidden error, ensure the bucket policy is correctly set to allow public read access.

By following these steps, you can easily host a static website on Amazon S3 and make it publicly accessible through the appropriate URL format. This process involves creating an S3 bucket, uploading website files, enabling static website hosting, and setting the correct permissions to ensure public access.

Enabling S3 Bucket for Static Website Hosting

To enable your Amazon S3 bucket for static website hosting and upload necessary files, follow these steps:

Step-by-Step Guide

1. **Upload Files to S3 Bucket**:
   - Upload any additional files you need, such as images. For instance, upload `beach.jpg` to your S3 bucket.
   - Ensure your bucket contains all the necessary files for your website.

2. **Enable Static Website Hosting**:
   - Navigate to the **Properties** tab of your S3 bucket.
   - Scroll down to find **Static website hosting**.
   - Click on **Edit**.
   - Select **Enable** and choose **Host a static website**.
   - Specify the **Index document** as `index.html`. This will be the default or homepage of your website.
   - Save the changes.

3. **Upload the Index File**:
   - Go back to the **Objects** tab.
   - Upload the `index.html` file by clicking on **Upload**, then **Add files**, select `index.html`, and **Upload**.

- Ensure that `index.html` is now listed among your bucket's files.

4. **Ensure Public Read Access**:
   - As mentioned in the previous steps, your bucket content must be publicly readable. Ensure that your bucket policy allows public access.
   - You should have already set a public read access policy.

5. **Access the Website**:
   - Navigate back to the **Properties** tab.
   - Scroll down to **Static website hosting** and find your **Bucket website endpoint**.
   - Copy the provided URL and paste it into your browser. You should see your `index.html` content displayed.
   - For instance, if your `index.html` file says "I love coffee. Hello world!" and references `coffee.jpg`, you should see that text and image.

6. **Accessing Uploaded Images**:
   - You can also directly access any uploaded images via their public URLs. For example:
     - `http://your-bucket-name.s3-website-region.amazonaws.com/coffee.jpg`
     - `http://your-bucket-name.s3-website-region.amazonaws.com/beach.jpg`

By following these steps, you have successfully enabled static website hosting on your S3 bucket. You can now access your static website through the provided S3 URL and view all your uploaded content publicly. This setup is essential for hosting static websites on AWS S3 and making sure all resources are accessible as intended.

Amazon S3 Versioning

Amazon S3 versioning is a feature that allows you to manage and preserve multiple versions of objects (files) in your S3 bucket. This is particularly useful for maintaining the history of an object, protecting against unintended deletions, and enabling easy rollbacks to previous versions.

Enabling Versioning

1. **Enable Versioning on a Bucket**:
   - Navigate to the S3 console.
   - Select the bucket you want to enable versioning for.
   - Go to the **Properties** tab.
   - Scroll down to **Bucket Versioning**.
   - Click on **Edit**.
   - Enable versioning and save the changes.

2. **Behavior of Versioning**:

- Once versioning is enabled, every time an object is uploaded to the bucket, S3 assigns a unique version ID to the object.
- If you upload an object with the same key (name), S3 will create a new version of that object rather than overwriting the existing one.
- Previous versions of the object remain stored in the bucket and can be accessed or restored at any time.

Benefits of Versioning

1. **Protection Against Unintended Deletes**:
   - If an object is deleted, S3 adds a delete marker instead of removing the object permanently.
   - This allows you to restore the object by removing the delete marker.
2. **Easy Rollback to Previous Versions**:
   - You can easily revert to a previous version of an object if needed.
   - This is useful for recovering from accidental changes or deletions.

Important Notes

- **Initial Versioning**:
  - Objects that were in the bucket before versioning was enabled will have a version ID of `null`.
  - This helps distinguish between objects created before and after versioning was enabled.
- **Suspending Versioning**:
  - You can suspend versioning on a bucket without deleting existing versions.
  - When versioning is suspended, new objects will not be versioned, but existing versions remain intact.

Using Versioning in the Console

1. **Upload a New Version of an Object**:
   - Upload a file to the versioned bucket.
   - Note the version ID assigned to the object.
   - Upload the same file again (with the same key/name). S3 will create a new version with a different version ID.
2. **Viewing Object Versions**:
   - In the S3 console, navigate to your bucket.
   - Click on **List versions** to view all versions of your objects.
   - You will see multiple versions of the same object, each with a unique version ID.
3. **Restoring a Previous Version**:
   - To restore a previous version, find the version you want to restore.

- You can either download that version or copy its version ID for reference.

4. **Deleting an Object Version**:
   - Deleting an object adds a delete marker.
   - To permanently delete an object, you need to delete all its versions including the delete marker.

By following these steps and understanding the benefits, you can effectively use versioning in Amazon S3 to manage object versions, protect your data, and enable easy recovery from accidental deletions or changes.

Playing with S3 Versioning

Enabling Versioning

1. **Enable Bucket Versioning**:
   - Navigate to your S3 bucket in the AWS console.
   - Go to the **Properties** tab.
   - Scroll to the **Bucket Versioning** section and click **Edit**.
   - Enable versioning and save the changes.

Updating Files with Versioning

1. **Modify and Upload a File**:
   - Edit your `index.html` file (e.g., change "I love coffee" to "I REALLY love coffee").
   - Upload the updated `index.html` file to your bucket.
   - Check the website URL to confirm the update ("I REALLY love coffee").

2. **View File Versions**:
   - In the S3 console, click on **Show versions**.
   - Notice the versions of `index.html`:
     - The initial upload has a `null` version ID (before versioning was enabled).
     - The updated file has a new version ID.

Rolling Back to a Previous Version

1. **Delete the Current Version**:
   - Ensure **Show versions** is enabled.
   - Select the current version ID of `index.html` and delete it.
   - Confirm the permanent delete by typing "permanently delete" and clicking **Delete objects**.

2. **Verify Rollback**:

- Refresh your website to see the original content ("I love coffee").

Deleting an Object

1. **Delete an Object**:
   - Disable **Show versions**.
   - Delete `coffee.jpg` by selecting it and clicking **Delete**.
   - Confirm the delete by typing "delete" and clicking **Delete objects**.
2. **Check for Delete Marker**:
   - Enable **Show versions**.
   - Notice a delete marker for `coffee.jpg` and the original version with its version ID.

Restoring a Deleted Object

1. **Remove the Delete Marker**:
   - Select the delete marker for `coffee.jpg` and delete it permanently.
   - This restores the previous version of the object.
2. **Verify Restoration**:
   - Refresh your website to see `coffee.jpg` reappear.

Playing Around with Versioning

- **Experiment with Versioning**:
  - Upload different versions of files and observe the creation of new version IDs.
  - Delete specific versions and understand the impact of delete markers versus permanent deletes.
  - Practice restoring previous versions by managing version IDs and delete markers.

By enabling and utilizing versioning, you can manage multiple versions of objects in your S3 bucket, protect against unintended deletions, and easily revert to previous states, ensuring data integrity and recovery capabilities.

Amazon S3 Replication

Amazon S3 offers two types of replication: Cross-Region Replication (CRR) and Same-Region Replication (SRR). These replication mechanisms allow you to copy data between S3 buckets asynchronously, providing various benefits such as compliance, latency reduction, and data redundancy.

Types of Replication

1. **Cross-Region Replication (CRR)**:
    - **Definition**: Replicates data between S3 buckets in different AWS regions.
    - **Use Cases**:
        - Compliance with regulatory requirements.
        - Reducing latency by placing data closer to users in different regions.
        - Data redundancy across geographical locations.
    - **Requirements**:
        - Source and destination buckets must be in different AWS regions.
        - Versioning must be enabled in both the source and destination buckets.
2. **Same-Region Replication (SRR)**:
    - **Definition**: Replicates data between S3 buckets within the same AWS region.
    - **Use Cases**:
        - Aggregating logs from multiple buckets.
        - Creating a live replication between production and test environments.
        - Redundancy within the same region for increased data durability.
    - **Requirements**:
        - Source and destination buckets must be in the same AWS region.
        - Versioning must be enabled in both the source and destination buckets.

Setting Up Replication

1. **Enable Versioning**:
    - Ensure versioning is enabled on both the source and destination buckets.
2. **IAM Permissions**:
    - Proper IAM permissions must be set to allow the S3 service to read from the source bucket and write to the destination bucket.
3. **Configuration**:
    - Define the replication configuration, specifying the source and destination buckets, IAM role, and replication rules.

Mechanism

- **Asynchronous Copying**:
    - Data is copied asynchronously in the background, ensuring minimal impact on the performance of ongoing operations in the source bucket.

Use Cases and Benefits

1. **CRR Use Cases**:
    - **Compliance**: Ensuring data is stored in different regions to meet regulatory requirements.
    - **Latency Reduction**: Placing data closer to end-users in different geographical locations.

- **Data Redundancy**: Ensuring data is available in multiple regions for disaster recovery.

2. **SRR Use Cases**:
    - **Log Aggregation**: Consolidating logs from various sources into a single bucket.
    - **Environment Replication**: Creating a mirrored test environment from a production environment for testing purposes.
    - **Data Redundancy**: Providing additional data protection within the same region.

By leveraging S3 Replication, organizations can enhance their data management strategies, improve accessibility, and ensure compliance with various regulatory requirements.

Additional Notes on Amazon S3 Replication

Replication Scope

- **New Objects Only**: Once replication is enabled, it applies only to new objects added to the source bucket. Existing objects will not be automatically replicated.
- **Batch Replication for Existing Objects**: To replicate existing objects and any objects that failed initial replication, use the S3 Batch Replication feature.

Handling Delete Operations

- **Replication of Delete Markers**: You can optionally replicate delete markers from the source bucket to the target bucket. This ensures that deletions in the source bucket are reflected in the target bucket.
- **Permanent Deletions**: Permanent deletions with version IDs are not replicated to prevent accidental or malicious deletions from propagating across buckets.

Replication Rules and Configuration

- **No Chaining of Replications**: Replication is direct between two buckets. If bucket one replicates to bucket two and bucket two replicates to bucket three, objects in bucket one will not automatically replicate to bucket three. Each replication rule is independent.

Summary

- **Enabling Replication**: Only new objects are replicated; use S3 Batch Replication for existing objects.
- **Delete Markers**: Can be replicated optionally; permanent deletions are not replicated.
- **Replication Configuration**: No chaining; each bucket pair's replication rule is independent.

These features and considerations ensure that S3 replication is flexible and secure, allowing for efficient data management and compliance with organizational requirements.

Notes on Amazon S3 Replication

Replication Scope

- **New Objects Only**: When replication is enabled, it applies only to new objects added to the source bucket. Existing objects will not be automatically replicated.

Replicating Existing Objects

- **S3 Batch Replication**: To replicate existing objects and any objects that failed initial replication, you can use the S3 Batch Replication feature. This feature ensures that all objects, regardless of when they were added, can be replicated to the target bucket.

Handling Delete Operations

- **Delete Markers Replication**: You can optionally replicate delete markers from the source bucket to the target bucket. This ensures that when an object is deleted in the source bucket, the deletion is also reflected in the target bucket.
- **Permanent Deletions**: Permanent deletions with version IDs are not replicated. This is to prevent accidental or malicious deletions from being propagated across buckets, providing an additional layer of data protection.

Replication Rules and Configuration

- **No Chaining of Replications**: Replication rules are direct and do not chain. For example, if bucket one replicates to bucket two and bucket two replicates to bucket three, objects in bucket one will not be automatically replicated to bucket three. Each replication rule is independent and applies only between the specified source and target buckets.

Summary

- **Enabling Replication**: Only new objects are replicated; use S3 Batch Replication for existing objects.
- **Delete Markers**: Can be replicated optionally; permanent deletions are not replicated.
- **Replication Configuration**: No chaining; each bucket pair's replication rule is independent.

These features and considerations ensure that S3 replication is flexible, efficient, and secure, facilitating robust data management and compliance with organizational requirements.

Practice with Amazon S3 Replication

Step-by-Step Process for Setting Up S3 Replication

1. **Create the Origin Bucket**:
    - Name: `s3-stephane-bucket-origin-v2`
    - Region: `eu-west-1`
    - Enable versioning on this bucket.
2. **Create the Target Bucket**:
    - Name: `s3-stephane-bucket-replica-v2`
    - Region: `us-east-1` (or another desired region)
    - Enable versioning on this bucket.
3. **Upload Initial File to Origin Bucket**:
    - Upload a file named `beach.jpg` to the origin bucket.
    - Verify the file upload.
4. **Set Up Replication**:
    - Go to the **Management** tab in the origin bucket.
    - Scroll down to **Replication Rules** and create a new rule.
    - Name the rule, e.g., `DemoReplicationRule`.
    - Set the rule as enabled.
    - Apply the rule to all objects in the bucket.
    - Specify the target bucket (in this example, `s3-stephane-bucket-replica-v2`).
    - Create a new IAM role for the replication process.
    - Choose not to replicate existing objects at this time.
5. **Upload a New File to Trigger Replication**:
    - Upload a file named `coffee.jpg` to the origin bucket.
    - Check the target bucket after a few seconds to ensure the file is replicated.
    - Verify that the version ID of `coffee.jpg` in the target bucket matches the origin bucket.
6. **Replication of Delete Markers**:
    - Edit the replication rule to enable delete marker replication.
    - Delete `coffee.jpg` from the origin bucket to create a delete marker.
    - Verify that the delete marker is replicated to the target bucket.
7. **Handling Permanent Deletions**:
    - Delete a specific version of `beach.jpg` from the origin bucket.
    - Verify that the permanent delete is not replicated to the target bucket, ensuring that only delete markers are replicated.

Key Points to Remember

- **New Objects Only**: Replication only affects new objects added after replication is enabled.
- **S3 Batch Replication**: Use this feature to replicate existing objects.
- **Delete Marker Replication**: Optional and needs to be explicitly enabled.
- **No Chaining of Replications**: Objects in one bucket are only replicated to the directly specified target bucket.

By following these steps, you ensure that your S3 buckets are set up for replication correctly, providing data redundancy and increased availability across different regions.

Amazon S3 Storage Classes

Amazon S3 offers various storage classes tailored for different use cases, focusing on cost-efficiency, availability, and retrieval times. Here's an in-depth look at each storage class:

1. Amazon S3 Standard-General Purpose

- **Use Case**: Frequently accessed data.
- **Availability**: 99.99%
- **Durability**: 99.999999999% (11 nines)
- **Characteristics**: Low latency, high throughput. Can sustain the loss of two concurrent facilities.
- **Examples**: Big data analytics, mobile and gaming applications, content distribution.

2. Amazon S3 Standard-Infrequent Access (IA)

- **Use Case**: Data that is less frequently accessed but requires rapid access when needed.
- **Availability**: 99.9%
- **Durability**: 11 nines
- **Characteristics**: Lower cost than Standard, retrieval fee applies.
- **Examples**: Disaster recovery, backups.

3. Amazon S3 One Zone-Infrequent Access (One Zone-IA)

- **Use Case**: Data that can be recreated if lost, stored in a single Availability Zone.
- **Availability**: 99.5%
- **Durability**: 11 nines within a single AZ.
- **Characteristics**: Lower cost, data is lost if the AZ is destroyed.
- **Examples**: Secondary copies of backups, on-premises data copies.

4. Amazon S3 Glacier Instant Retrieval

- **Use Case**: Archiving data with millisecond retrieval.
- **Availability**: N/A (asynchronous retrieval)

- **Durability**: 11 nines
- **Characteristics**: Low cost, minimum storage duration of 90 days, retrieval in milliseconds.
- **Examples**: Data accessed once a quarter.

5. Amazon S3 Glacier Flexible Retrieval

- **Use Case**: Archiving data with flexible retrieval times.
- **Availability**: N/A (asynchronous retrieval)
- **Durability**: 11 nines
- **Characteristics**: Low cost, minimum storage duration of 90 days, retrieval options:
  - **Expedited**: 1-5 minutes
  - **Standard**: 3-5 hours
  - **Bulk**: 5-12 hours
- **Examples**: Archive data requiring various retrieval speeds.

6. Amazon S3 Glacier Deep Archive

- **Use Case**: Long-term archival storage.
- **Availability**: N/A (asynchronous retrieval)
- **Durability**: 11 nines
- **Characteristics**: Lowest cost, minimum storage duration of 180 days, retrieval options:
  - **Standard**: 12 hours
  - **Bulk**: 48 hours
- **Examples**: Data that can be accessed infrequently over a long period.

7. Amazon S3 Intelligent-Tiering

- **Use Case**: Automatically moves data between access tiers based on usage patterns.
- **Availability**: Same as S3 Standard for frequently accessed data, other tiers as needed.
- **Durability**: 11 nines
- **Characteristics**: Small monitoring and auto-tiering fee, no retrieval charges, multiple tiers:
  - **Frequent Access**: Default tier.
  - **Infrequent Access**: For objects not accessed for 30 days.
  - **Archive Instant Access**: For objects not accessed for 90 days.
  - **Archive Access**: Optional, configurable for 90-700+ days.
  - **Deep Archive Access**: Optional, configurable for 180-700+ days.
- **Examples**: Data with unknown or unpredictable access patterns.

Key Concepts: Durability and Availability

- **Durability**: Indicates the likelihood of data loss. S3 offers 11 nines of durability, meaning you might lose one object out of 10 million every 10,000 years.
- **Availability**: Indicates the readiness of the service. For example, S3 Standard has 99.99% availability, equating to about 53 minutes of downtime per year.

Managing Storage Classes

- **Choosing a Storage Class**: When creating an object in S3, you can select its storage class.
- **Modifying Storage Classes**: You can manually change the storage class of an object or use S3 Lifecycle configurations to automatically move objects between storage classes based on predefined rules and access patterns.

Summary

Understanding the various S3 storage classes allows you to optimize costs while meeting your data access and retrieval requirements. Each class is designed to handle specific use cases, from frequent access needs to long-term archival storage, ensuring flexibility and cost-efficiency in data management.