Moving Objects Between AWS S3 Storage Classes

Overview of Storage Class Transitions

AWS S3 offers various storage classes, each designed for different use cases based on access frequency and cost requirements. Transitioning objects between these storage classes can help optimize costs. Here's a visual guide to the transitions:

- **Standard** to **Standard-IA** (Infrequent Access)
- **Standard-IA** to **Intelligent Tiering**
- **Standard-IA** to **One-Zone-IA**
- **One-Zone-IA** to **Glacier Flexible Retrieval**
- **One-Zone-IA** to **Glacier Deep Archive**

If you expect your objects to be accessed infrequently, move them to **Standard-IA**. For archiving, move objects to **Glacier** or **Deep Archive**.

Automating Transitions with Lifecycle Rules

Lifecycle rules can automate the transition of objects between storage classes. These rules consist of:

1. **Transition Actions:**
   - Specify when an object should move to a different storage class.
   - Example: Move to **Standard-IA** after 60 days or to **Glacier** after 6 months.
2. **Expiration Actions:**
   - Define when objects should be deleted.
   - Example: Delete access log files after 365 days or delete old versions of files if versioning is enabled.
3. **Prefixes and Tags:**
   - Apply rules to specific paths (prefixes) or based on object tags.
   - Example: Apply a rule only to objects with the tag `department=finance`.

Practical Scenarios

1. **Image Thumbnails Example:**
   - **Source Images:**
     - Stored in **Standard** class.
     - Transition to **Glacier** after 60 days.
   - **Thumbnails:**
     - Stored in **One-Zone-IA**.
     - Delete after 60 days.
   - Use a prefix to differentiate between source images and thumbnails.

2. **Recoverable Deleted Objects Example:**
    - Enable S3 versioning to keep object versions.
    - Create a rule to transition non-current versions to **Standard-IA** after 30 days.
    - Further transition non-current versions to **Glacier Deep Archive** for long-term archival.

Determining Optimal Transition Periods

- Use **Amazon S3 Analytics** to analyze usage patterns and get recommendations for transitioning objects from **Standard** to **Standard-IA**.
- S3 Analytics provides a CSV report updated daily with insights and recommendations, which can help in configuring effective lifecycle rules.
- Note that S3 Analytics does not support **One-Zone-IA** or **Glacier**.

Summary

Using lifecycle rules in AWS S3 allows for efficient and cost-effective management of data by automating the transition and expiration of objects based on predefined criteria. Employing S3 Analytics can further optimize these processes by providing data-driven insights into access patterns and transition timing.

Creating a Lifecycle Rule for S3 Buckets

To efficiently manage the storage of your S3 objects, you can create lifecycle rules that automate the transition and expiration of objects based on specified criteria. Here's a step-by-step guide to creating a lifecycle rule:

1. **Access Lifecycle Management:**
    - Go to the S3 console.
    - Select the bucket for which you want to create a lifecycle rule.
    - Navigate to the "Management" tab.
    - Click on "Create lifecycle rule."
2. **Define the Rule:**
    - Name your rule (e.g., "demo rule").
    - Apply the rule to all objects in the bucket.
    - Acknowledge the application of the rule to all objects.
3. **Configure Rule Actions:**
    - **Move Current Versions of Objects Between Storage Classes:**
        - Example transitions:
            - Standard IA after 30 days
            - Intelligent-Tiering after 60 days
            - Glacier Instant Retrieval after 90 days
            - Glacier Flexible Retrieval after 180 days
            - Glacier Deep Archive after 365 days
    - **Move Non-Current Versions of Objects Between Storage Classes:**
        - Example transition:
            - Move non-current versions to Glacier Flexible Retrieval after 90 days.
    - **Expire Current Versions of Objects:**
        - Set expiration for current versions (e.g., after 700 days).
    - **Permanently Delete Non-Current Versions of Objects:**
        - Permanently delete non-current versions (e.g., after 700 days).
    - **Delete Expired Objects, Delete Markers, or Incomplete Multi-Part Uploads:**
        - Configure to handle these scenarios as needed.
4. **Review and Confirm Actions:**
    - Review the timeline that outlines the transitions and expiration actions for both current and non-current versions of your objects.
    - Ensure that the actions align with your data management policies and cost optimization strategies.
5. **Create the Rule:**
    - Once satisfied with the configuration, create the rule.

    - The rule will operate in the background, automatically transitioning and expiring objects as specified.

By creating and applying lifecycle rules, you can automate the management of your S3 objects, optimizing storage costs and adhering to data retention policies without manual intervention. This process ensures that objects are transitioned to more cost-effective storage classes based on their access patterns and are eventually deleted when they are no longer needed.

S3 Event Notifications

Amazon S3 Event Notifications enable you to automatically respond to certain events occurring within your S3 bucket. Here's a detailed breakdown of how these notifications work and how they can be configured.

Types of Events

Events in S3 include:

- Object creation
- Object removal
- Object restoration
- Replication events

Filtering Events

You can filter events to only consider specific objects. For example, you might filter to only respond to objects that end with `.jpeg`.

Use Cases

A common use case is to generate thumbnails for images uploaded to S3. When an image is uploaded, an event notification triggers a function that creates the thumbnail.

Destinations for Event Notifications

You can send S3 event notifications to:

- **SNS Topic**: Simple Notification Service for broadcasting messages.
- **SQS Queue**: Simple Queue Service for message queuing.
- **Lambda Function**: For executing code in response to the event.

IAM Permissions

To enable S3 to send notifications, you must configure resource access policies:

- **SNS Resource Policy**: Allows S3 to send messages to an SNS topic.
- **SQS Resource Policy**: Authorizes S3 to send messages to an SQS queue.
- **Lambda Resource Policy**: Grants S3 permission to invoke a Lambda function.

Configuration Example

For example, if you want to create an event notification to generate thumbnails of images uploaded to an S3 bucket:

1. **Create the Event Notification**:
    - Define the event type (e.g., object creation).

    - Set up filters (e.g., objects ending with `.jpeg`).
2. **Specify the Destination**:
    - Choose SNS, SQS, or Lambda.
3. **Set Up IAM Policies**:
    - Attach the necessary resource policy to the SNS topic, SQS queue, or Lambda function.

Advanced Integration with EventBridge

Amazon EventBridge is another powerful tool for handling S3 events. All S3 events automatically flow into EventBridge, where you can set up rules and advanced filtering. With EventBridge, you can:

- Send events to over 18 AWS services, such as Step Functions, Kinesis Data Streams, or Firehose.
- Filter by metadata, object size, and name.
- Archive and replay events.
- Ensure more reliable event delivery.

Summary

S3 Event Notifications allow you to react to various events within your S3 bucket by sending them to destinations like SQS, SNS, Lambda, or EventBridge. This automation enhances the functionality and responsiveness of your AWS environment, enabling you to build sophisticated workflows and applications.

Demonstration of S3 Event Notifications

Let's go through the process of setting up and demonstrating S3 Event Notifications. We'll create an S3 bucket, configure event notifications to send messages to an SQS queue, and ensure that the notifications work as expected.

1. **Create an S3 Bucket**:
    - Create a bucket named "stephane-v3-events-notifications".
    - Go to the S3 service, choose the Ireland region, and create the bucket.
2. **Set Up Event Notifications**:
    - Navigate to the bucket properties.
    - Scroll down to find the "Event Notifications" section.
    - There are two options: creating an event notification or enabling Amazon EventBridge integration.
    - For simplicity, we'll create an event notification.
3. **Create an Event Notification**:
    - Name the notification "DemoEventNotification".
    - Skip prefix and suffix filters for simplicity.
    - Choose event types: Select "All object create events".
    - You can also choose other event types like object removals, restorations, etc.
4. **Select Destination**:
    - Choose SQS queue as the destination.
    - First, create an SQS queue.
5. **Create an SQS Queue**:
    - Go to the SQS service and create a queue named "DemoS3Notification".
6. **Configure SQS Access Policy**:
    - To allow the S3 bucket to send messages to the SQS queue, adjust the access policy.
    - Initially, attempting to link the S3 event notification to the SQS queue may result in an error because the queue does not yet accept messages from the S3 bucket.
7. **Update SQS Access Policy**:
    - Use the Policy Generator to create a policy allowing the S3 bucket to send messages to the SQS queue.
    - Generate a policy that grants "SendMessage" permission to anyone (for demonstration purposes).
    - Attach this policy to the SQS queue.
8. **Link S3 Event Notification to SQS Queue**:

- Go back to the S3 event notification configuration.
- Select the newly created SQS queue.
- Save the changes.

9. **Test the Event Notification**:

   - Verify the configuration by checking the SQS queue for a test message sent by S3 to ensure connectivity.
   - Now, upload an object to the S3 bucket to trigger the event notification.

10. **Upload an Object**:

    - Upload a file (e.g., "coffee.jpg") to the S3 bucket.
    - Confirm the file is uploaded by checking the bucket.

11. **Check SQS Queue**:

    - Go to the SQS queue and poll for messages.
    - You should see a message indicating that an object was created in the S3 bucket.
    - The message will contain details such as the event name ("ObjectCreated:Put") and the key of the uploaded object ("coffee.jpg").

This demonstration shows how S3 Event Notifications can be used to automatically react to events in an S3 bucket by sending messages to an SQS queue. This setup can be extended to perform various automated tasks, such as generating thumbnails for uploaded images. Additionally, remember that event notifications can also be sent to SNS topics, Lambda functions, or Amazon EventBridge for further processing and more advanced workflows.

S3 Baseline Performance

Amazon S3 is designed to automatically scale and handle a large number of requests with low latency. Here's a detailed explanation of S3's baseline performance and optimization techniques:

Baseline Performance

1. **Low Latency**:
   - S3 provides very low latency, typically between 100 and 200 milliseconds to retrieve the first byte.
2. **High Request Rates**:
   - **PUT/COPY/POST/DELETE**: 3,500 requests per second per prefix.
   - **GET/HEAD**: 5,500 requests per second per prefix.

Understanding Prefixes

- A prefix in S3 is essentially the part of the object key (file path) before the actual file name. For example, in the object key `bucket/folder1/subfolder1/file`, the prefix is `folder1/subfolder1/`.
- The high request rate per prefix means that if you organize your data into multiple prefixes, you can achieve much higher overall performance. For instance:
  - `bucket/folder1/sub1/file`
  - `bucket/folder1/sub2/file`
  - Each prefix (`folder1/sub1/` and `folder1/sub2/`) can independently handle 3,500 PUT and 5,500 GET requests per second.

By distributing your objects across multiple prefixes, you can effectively increase the number of requests per second that your bucket can handle.

Performance Optimization Techniques

1. **Multi-Part Upload**:
   - Recommended for files over 100 MB and mandatory for files over 5 GB.
   - Multi-part upload divides a file into smaller parts and uploads them in parallel, speeding up the transfer and maximizing bandwidth usage.
   - Example: A large file is split into smaller parts, each part is uploaded concurrently, and S3 assembles the parts back into the original file after all parts are uploaded.
2. **S3 Transfer Acceleration**:
   - Increases transfer speed by routing data through AWS edge locations.
   - Edge locations are numerous and closer to users, reducing latency.
   - Example: Uploading a file from the USA to an S3 bucket in Australia via an edge location in the USA. The file first travels quickly to the edge location, then over AWS's fast private

network to the destination bucket, minimizing the time spent on the slower public internet.

3. **S3 Byte-Range Fetches**:
   - Parallelizes download requests by fetching specific byte ranges of a file.
   - Allows for faster downloads by requesting multiple parts of a file simultaneously.
   - Example: A large file can be split into byte ranges, with each range requested in parallel. This technique can also be used to fetch only necessary parts of a file, such as headers, to reduce data transfer and latency.

Use Cases and Examples

1. **Multi-Part Upload**:
   - **Scenario**: Uploading a 1 GB video file.
   - **Process**: Split the file into smaller chunks (e.g., 100 MB each), upload each chunk in parallel, and let S3 assemble the file after all chunks are uploaded.
   - **Benefit**: Faster upload due to parallel processing.
2. **S3 Transfer Acceleration**:
   - **Scenario**: Uploading data from a user in New York to an S3 bucket in Sydney.
   - **Process**: Upload data to a nearby edge location in New York, which then uses AWS's private network to transfer the data to Sydney.
   - **Benefit**: Reduced upload time and higher reliability.
3. **S3 Byte-Range Fetches**:
   - **Scenario**: Downloading a 2 GB file but only needing the first 100 MB.
   - **Process**: Request only the first 100 MB using a byte-range request.
   - **Benefit**: Faster access to the required data and reduced data transfer costs.

By understanding and implementing these optimization techniques, you can significantly improve the performance and efficiency of your interactions with Amazon S3, ensuring that your applications run smoothly and cost-effectively.

S3 Select and Glacier Select

Overview

S3 Select and Glacier Select allow you to retrieve only the necessary subsets of data from S3 objects and Glacier archives using SQL queries. This server-side filtering reduces the amount of data transferred over the network and lowers the client-side CPU cost for processing data.

Traditional Data Retrieval vs. S3 Select

- **Traditional Method**:
  - Retrieve the entire file from S3.
  - Filter the data on the application side.
  - This approach often results in retrieving large amounts of data, even if only a small portion is needed.
- **S3 Select Method**:
  - Perform server-side filtering using SQL queries.
  - Retrieve only the required subset of data.
  - This method significantly reduces data transfer and processing time.

Benefits of S3 Select

- **Performance**: Up to 400% faster retrieval because only the necessary data is transferred.
- **Cost**: Up to 80% cheaper by reducing the amount of data transferred and lowering processing costs.

How S3 Select Works

1. **SQL Query**: Use SQL queries to specify the filtering criteria (e.g., specific rows or columns).
2. **Server-Side Filtering**: S3 processes the query and filters the data on the server.
3. **Data Transfer**: Only the filtered data is sent back to the client.

Example

Imagine you have a large CSV file stored in S3, but you only need a few rows that match certain conditions:

- **Without S3 Select**:
  - Retrieve the entire CSV file from S3.
  - Filter the data on the client side.
  - This results in a large amount of data transfer and higher processing costs.
- **With S3 Select**:
  - Execute a SQL query on S3 to retrieve only the matching rows.

    - S3 processes the query and filters the data.
    - Only the filtered rows are sent back, significantly reducing data transfer and processing time.

Diagram

1. **Before S3 Select**:
    - Retrieve entire file from S3.
    - Filter data client-side.
    - High data transfer and processing cost.
2. **With S3 Select**:
    - Execute SQL query on S3.
    - S3 filters data server-side.
    - Transfer only filtered data.
    - Reduced data transfer and processing cost.

Glacier Select

- Glacier Select works similarly to S3 Select but is used for data stored in Glacier, which is typically for archival purposes.
- Use SQL queries to retrieve only the necessary data from Glacier archives.
- Reduces retrieval costs and speeds up access to archived data.

Use Cases

- **Data Analytics**: Quickly retrieve specific rows and columns needed for analysis without downloading entire datasets.
- **Log Processing**: Filter logs stored in S3 to retrieve only relevant entries.
- **Archival Data Access**: Use Glacier Select to access specific parts of archived data without incurring high retrieval costs.

By leveraging S3 Select and Glacier Select, you can optimize data retrieval, reduce costs, and improve performance for various applications that interact with data stored in S3 and Glacier.

User-Defined Object Metadata and S3 Object Tags

User-Defined Object Metadata

- **Definition**: Key-value pairs attached to your objects in S3.
- **Format**: User-defined metadata keys must start with `x-amz-meta-`. This differentiates them from system-defined metadata, which is generated by AWS.
- **Usage**: Provides additional information about the object, which can be retrieved when accessing the object.

Example:

- **Content-Length**: `7.5 kilobytes` (AWS system-defined)
- **Content-Type**: `html` (AWS system-defined)
- **x-amz-meta-origin**: `paris` (User-defined)

User-defined metadata can be useful for storing custom information about the object, such as the origin or any other relevant data.

S3 Object Tags

- **Definition**: Key-value pairs that are used to categorize and manage your S3 objects.
- **Usage**: More commonly used than metadata for several reasons, including fine-grained permissions and analytics.

Example:

- **Project**: `Blue`
- **PHI**: `True` (Indicating the object contains personal health information)

Tags can help in:

- **Permissions**: Using tags for defining and controlling access to specific objects within AWS.
- **Analytics**: Grouping findings and performing analytics based on tags using tools like S3 Analytics.

Key Points to Remember

1. **Non-Searchable**: Neither metadata nor tags are searchable within S3. You cannot filter or search objects based on their metadata or tags directly in S3.
2. **External Indexing**: To search objects based on metadata or tags, you need to build an external index using a database like DynamoDB.

External Indexing with DynamoDB

1. **Create an Index**: Store the metadata and tags in a searchable index within DynamoDB.
2. **Perform Searches**: Conduct your searches on DynamoDB to find the relevant objects.
3. **Retrieve Results**: Use the results from DynamoDB to extract the corresponding objects from S3.

Architecture:

- **Metadata/Tags Storage**: Store the metadata and tags in DynamoDB when objects are uploaded to S3.
- **Search**: Query DynamoDB to find objects based on metadata or tags.
- **Fetch**: Use the DynamoDB results to retrieve the objects from S3.

This approach allows you to efficiently search and manage your S3 objects based on their metadata and tags without any direct filtering capability in S3 itself.