Chapter 2
Amazon Simple Storage Service (Amazon S3) and Amazon Glacier Storage

THE AWS CERTIFIED SOLUTIONS ARCHITECT ASSOCIATE EXAM OBJECTIVES COVERED IN THIS CHAPTER MAY INCLUDE, BUT ARE NOT LIMITED TO, THE FOLLOWING:

Domain 1.0: Designing highly available, cost-efficient, fault-tolerant, scalable systems

✓ **1.1 Identify and recognize cloud architecture considerations, such as fundamental components and effective designs.**

Content may include the following:

- How to design cloud services
- Planning and design
- Monitoring and logging
- Familiarity with:
  - Best practices for AWS architecture
  - Developing to client specifications, including pricing/cost (e.g., On Demand vs. Reserved vs. Spot; Recovery Time Objective [RTO] and Recovery Point Objective [RPO] disaster recovery design)
  - Architectural trade-off decisions (e.g., high availability vs. cost)
  - Hybrid IT architectures
  - Elasticity and scalability

Domain 2.0: Implementation/Deployment

✓ **2.1 Identify the appropriate techniques and methods using Amazon Simple Storage Service (Amazon S3) to code and implement a cloud solution.**

Content may include the following:

- Configure services to support compliance requirements in the cloud.
- Launch instances across the AWS global infrastructure.
- Configure AWS Identity and Access Management (IAM) policies and best practices.

Domain 3.0: Data Security

✓ **3.1 Recognize and implement secure practices for optimum cloud deployment and maintenance**

Content may include the following:

- Security Architecture with AWS
  - “Core” Amazon S3 security feature sets
  - Encryption solutions (e.g., key services)
  - Complex access controls (building sophisticated security groups, Access Control Lists [ACLs], etc.)

Introduction

This chapter is intended to provide you with a basic understanding of the core object storage services available on AWS: Amazon Simple Storage Service (Amazon S3) and Amazon Glacier.

Amazon S3 provides developers and IT teams with secure, durable, and highly-scalable cloud storage. Amazon S3 is easy-to-use *object storage* with a simple web service interface that you can use to store and retrieve any amount of data from anywhere on the web. Amazon S3 also allows you to pay only for the storage you actually use, which eliminates the capacity planning and capacity constraints associated with traditional storage.

Amazon S3 is one of first services introduced by AWS, and it serves as one of the foundational web services—nearly any application running in AWS uses Amazon S3, either directly or indirectly. Amazon S3 can be used alone or in conjunction with other AWS services, and it offers a very high level of integration with many other AWS cloud services. For example, Amazon S3 serves as the durable target storage for Amazon Kinesis and Amazon Elastic MapReduce (Amazon EMR), it is used as the storage for Amazon Elastic Block Store (Amazon EBS) and Amazon Relational Database Service (Amazon RDS) snapshots, and it is used as a data staging or loading storage mechanism for Amazon Redshift and Amazon DynamoDB, among many other functions. Because Amazon S3 is so flexible, so highly integrated, and so commonly used, it is important to understand this service in detail.

Common use cases for Amazon S3 storage include:

- Backup and archive for on-premises or cloud data
- Content, media, and software storage and distribution
- Big data analytics
- Static website hosting
- Cloud-native mobile and Internet application hosting
- Disaster recovery

To support these use cases and many more, Amazon S3 offers a range of *storage classes* designed for various generic use cases: general purpose, infrequent access, and archive. To help manage data through its lifecycle, Amazon S3 offers configurable lifecycle policies. By using lifecycle policies, you can have your data automatically migrate to the most appropriate storage class, without modifying your application code. In order to control who has access to your data, Amazon S3 provides a rich set of permissions, access controls, and encryption options.

Amazon Glacier is another cloud storage service related to Amazon S3, but optimized for data archiving and long-term backup at extremely low cost. Amazon Glacier is suitable for “cold data,” which is data that is rarely accessed and for which a retrieval time of three to five hours is acceptable. Amazon Glacier can be used both as a storage class of Amazon S3 (see Storage Classes and Object Lifecycle Management topics in the Amazon S3 Advanced Features section), and as an independent archival storage service (see the Amazon Glacier section).

Object Storage versus Traditional Block and File Storage

In traditional IT environments, two kinds of storage dominate: *block storage* and *file storage*. Block storage operates at a lower level—the raw storage device level—and manages data as a set of numbered, fixed-size blocks. File storage operates at a higher level—the operating system level—and manages data as a named hierarchy of files and folders. Block and file storage are often accessed over a network in the form of a Storage Area Network (SAN) for block storage, using protocols such as iSCSI or Fibre Channel, or as a Network Attached Storage (NAS) file server or "filer" for file storage, using protocols such as Common Internet File System (CIFS) or Network File System (NFS). Whether directly-attached or network-attached, block or file, this kind of storage is very closely associated with the server and the operating system that is using the storage.

Amazon S3 object storage is something quite different. Amazon S3 is cloud *object storage*. Instead of being closely associated with a server, Amazon S3 storage is independent of a server and is accessed over the Internet. Instead of managing data as blocks or files using SCSI, CIFS, or NFS protocols, data is managed as objects using an Application Program Interface (API) built on standard HTTP verbs.

Each Amazon S3 object contains both data and metadata. Objects reside in containers called *buckets*, and each object is identified by a unique user-specified key (filename). Buckets are a simple flat folder with no file system hierarchy. That is, you can have multiple buckets, but you can't have a sub-bucket within a bucket. Each bucket can hold an unlimited number of objects.

It is easy to think of an Amazon S3 object (or the data portion of an object) as a file, and the key as the filename. However, keep in mind that Amazon S3 is not a traditional file system and differs in significant ways. In Amazon S3, you `GET` an object or `PUT` an object, operating on the whole object at once, instead of incrementally updating portions of the object as you would with a file. You can't "mount" a bucket, "open" an object, install an operating system on Amazon S3, or run a database on it.

Instead of a file system, Amazon S3 is highly-durable and highly-scalable object storage that is optimized for reads and is built with an intentionally minimalistic feature set. It provides a simple and robust abstraction for file storage that frees you from many underlying details that you normally do have to deal with in traditional storage. For example, with Amazon S3 you don't have to worry about device or file system storage limits and capacity planning—a single bucket can store an unlimited number of files. You also don't need to worry about data durability or replication across availability zones—Amazon S3 objects are automatically replicated on multiple devices in multiple facilities within a region. The same with scalability—if your request rate grows steadily, Amazon S3 automatically partitions buckets to support very high request rates and simultaneous access by many clients.


If you need traditional block or file storage in addition to Amazon S3 storage, AWS provides options. The Amazon EBS service provides block level storage for Amazon Elastic Compute Cloud (Amazon EC2) instances. Amazon Elastic File System (AWS EFS) provides network-attached shared file storage (NAS storage) using the NFS v4 protocol.

Amazon Simple Storage Service (Amazon S3) Basics

Now that you have an understanding of some of the key differences between traditional block and file storage versus cloud object storage, we can explore the basics of Amazon S3 in more detail.

Buckets

A *bucket* is a container (web folder) for objects (files) stored in Amazon S3. Every Amazon S3 object is contained in a bucket. Buckets form the top-level namespace for Amazon S3, and bucket names are global. This means that your bucket names must be unique across all AWS accounts, much like Domain Name System (DNS) domain names, not just within your own account. Bucket names can contain up to 63 lowercase letters, numbers, hyphens, and periods. You can create and use multiple buckets; you can have up to 100 per account by default.

 It is a best practice to use bucket names that contain your domain name and conform to the rules for DNS names. This ensures that your bucket names are your own, can be used in all regions, and can host static websites.

AWS Regions

Even though the namespace for Amazon S3 buckets is global, each Amazon S3 bucket is created in a specific region that you choose. This lets you control where your data is stored. You can create and use buckets that are located close to a particular set of end users or customers in order to minimize latency, or located in a particular region to satisfy data locality and sovereignty concerns, or located far away from your primary facilities in order to satisfy disaster recovery and compliance needs. You control the location of your data; data in an Amazon S3 bucket is stored in that region unless you explicitly copy it to another bucket located in a different region.

Objects

Objects are the entities or files stored in Amazon S3 buckets. An object can store virtually any kind of data in any format. Objects can range in size from 0 bytes up to 5TB, and a single bucket can store an unlimited number of objects. This means that Amazon S3 can store a virtually unlimited amount of data.

Each object consists of data (the file itself) and *metadata* (data about the file). The data portion of an Amazon S3 object is opaque to Amazon S3. This means that an object's data is treated as simply a stream of bytes—Amazon S3 doesn't know or care what type of data you are storing, and the service doesn't act differently for text data versus binary data.

The metadata associated with an Amazon S3 object is a set of name/value pairs that describe the object. There are two types of metadata: system metadata and user metadata. System metadata is created and used by Amazon S3 itself, and it includes things like the date last modified, object size, MD5 digest, and HTTP Content-Type. User metadata is optional, and it can only be specified at the time an object is created. You can use custom metadata to tag your data with attributes that are meaningful to you.

Keys

Every object stored in an S3 bucket is identified by a unique identifier called a *key*. You can think of the key as a filename. A key can be up to 1024 bytes of Unicode UTF-8 characters, including embedded slashes, backslashes, dots, and dashes.

Keys must be unique within a single bucket, but different buckets can contain objects with the same key. The combination of bucket, key, and optional version ID uniquely identifies an Amazon S3 object.

Object URL

Amazon S3 is storage for the Internet, and every Amazon S3 object can be addressed by a unique URL formed using the web services endpoint, the bucket name, and the object key. For example, with the URL:

`http://mybucket.s3.amazonaws.com/jack.doc`

`mybucket` is the S3 bucket name, and `jack.doc` is the key or filename. If another object is created, for instance:

`http://mybucket.s3.amazonaws.com/fee/fi/fo/fum/jack.doc`

then the bucket name is still `mybucket`, but now the key or filename is the string `fee/fi/fo/fum/jack.doc`. A key may contain delimiter characters like slashes or backslashes to help you name and logically organize your Amazon S3 objects, but to Amazon S3 it is simply a long key name in a flat namespace. There is no actual file and folder hierarchy. See the topic “Prefixes and Delimiters” in the “Amazon S3 Advanced Features” section that follows for more information.

For convenience, the Amazon S3 console and the Prefix and Delimiter feature allow you to navigate within an Amazon S3 bucket as if there were a folder hierarchy. However, remember that a bucket is a single flat namespace of keys with no structure.

Amazon S3 Operations

The Amazon S3 API is intentionally simple, with only a handful of common operations. They include:

- Create/delete a bucket
- Write an object
- Read an object
- Delete an object
- List keys in a bucket

REST Interface

The native interface for Amazon S3 is a *REST (Representational State Transfer)* API. With the REST interface, you use standard HTTP or HTTPS requests to create and delete buckets, list keys, and read and write objects. REST maps standard HTTP “verbs” (HTTP methods) to

the familiar CRUD (Create, Read, Update, Delete) operations. Create is `HTTP PUT` (and sometimes `POST`); read is `HTTP GET`; delete is `HTTP DELETE`; and update is `HTTP POST` (or sometimes `PUT`).

 Always use HTTPS for Amazon S3 API requests to ensure that your requests and data are secure.

In most cases, users do not use the REST interface directly, but instead interact with Amazon S3 using one of the higher-level interfaces available. These include the AWS Software Development Kits (SDKs) (wrapper libraries) for iOS, Android, JavaScript, Java, .NET, Node.js, PHP, Python, Ruby, Go, and C++, the AWS Command Line Interface (CLI), and the AWS Management Console.

 Amazon S3 originally supported a SOAP (Simple Object Access Protocol) API in addition to the REST API, but you should use the REST API. The legacy HTTPS endpoint is still available, but new features are not supported.

Durability and Availability

Data *durability* and *availability* are related but slightly different concepts. Durability addresses the question, "Will my data still be there in the future?" Availability addresses the question, "Can I access my data right now?" Amazon S3 is designed to provide both very high durability and very high availability for your data.

Amazon S3 standard storage is designed for 99.999999999% durability and 99.99% availability of objects over a given year. For example, if you store 10,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000,000 years. Amazon S3 achieves high durability by automatically storing data redundantly on multiple devices in multiple facilities within a region. It is designed to sustain the concurrent loss of data in two facilities without loss of user data. Amazon S3 provides a highly durable storage infrastructure designed for mission-critical and primary data storage.

If you need to store non-critical or easily reproducible derived data (such as image thumbnails) that doesn't require this high level of durability, you can choose to use Reduced Redundancy Storage (RRS) at a lower cost. RRS offers 99.99% durability with a lower cost of storage than traditional Amazon S3 storage.

 Even though Amazon S3 storage offers very high durability at the infrastructure level, it is still a best practice to protect against user-level accidental deletion or overwriting of data by using additional features such as versioning, cross-region replication, and MFA Delete.

Data Consistency

Amazon S3 is an *eventually consistent* system. Because your data is automatically replicated across multiple servers and locations within a region, changes in your data may take some time to propagate to all locations. As a result, there are some situations where information that you read immediately after an update may return stale data.

For PUTs to new objects, this is not a concern—in this case, Amazon S3 provides read-after-write consistency. However, for PUTs to existing objects (object overwrite to an existing key) and for object DELETEs, Amazon S3 provides *eventual consistency*.

Eventual consistency means that if you PUT new data to an existing key, a subsequent GET might return the old data. Similarly, if you DELETE an object, a subsequent GET for that object might still read the deleted object. In all cases, updates to a single key are atomic—for eventually-consistent reads, you will get the new data or the old data, but never an inconsistent mix of data.

Access Control

Amazon S3 is secure by default; when you create a bucket or object in Amazon S3, only you have access. To allow you to give controlled access to others, Amazon S3 provides both coarse-grained access controls (Amazon S3 Access Control Lists [ACLs]), and fine-grained access controls (Amazon S3 bucket policies, AWS Identity and Access Management [IAM] policies, and query-string authentication).

Amazon S3 ACLs allow you to grant certain coarse-grained permissions: READ, WRITE, or FULL-CONTROL at the object or bucket level. ACLs are a legacy access control mechanism, created before IAM existed. ACLs are best used today for a limited set of use cases, such as enabling bucket logging or making a bucket that hosts a static website be world-readable.

Amazon S3 bucket policies are the recommended access control mechanism for Amazon S3 and provide much finer-grained control. Amazon S3 bucket policies are very similar to IAM policies, which were discussed in Chapter 6, “AWS Identity and Access Management (IAM),” but are subtly different in that:

- They are associated with the bucket resource instead of an IAM principal.
- They include an explicit reference to the IAM principal in the policy. This principal can be associated with a different AWS account, so Amazon S3 bucket policies allow you to assign cross-account access to Amazon S3 resources.

Using an Amazon S3 bucket policy, you can specify who can access the bucket, from where (by Classless Inter-Domain Routing [CIDR] block or IP address), and during what time of day.

Finally, IAM policies may be associated directly with IAM principals that grant access to an Amazon S3 bucket, just as it can grant access to any AWS service and resource. Obviously, you can only assign IAM policies to principals in AWS accounts that you control.

Static Website Hosting

A very common use case for Amazon S3 storage is *static website* hosting. Many websites, particularly micro-sites, don’t need the services of a full web server. A static website means

that all of the pages of the website contain only static content and do not require server-side processing such as PHP, ASP.NET, or JSP. (Note that this does not mean that the website cannot be interactive and dynamic; this can be accomplished with client-side scripts, such as JavaScript embedded in static HTML webpages.) Static websites have many advantages: they are very fast, very scalable, and can be more secure than a typical dynamic website. If you host a static website on Amazon S3, you can also leverage the security, durability, availability, and scalability of Amazon S3.

Because every Amazon S3 object has a URL, it is relatively straightforward to turn a bucket into a website. To host a static website, you simply configure a bucket for website hosting and then upload the content of the static website to the bucket.

To configure an Amazon S3 bucket for static website hosting:

1. Create a bucket with the same name as the desired website hostname.
2. Upload the static files to the bucket.
3. Make all the files public (world readable).
4. Enable static website hosting for the bucket. This includes specifying an Index document and an Error document.
5. The website will now be available at the S3 website URL:

   `<bucket-name>.s3-website-<AWS-region>.amazonaws.com`.
6. Create a friendly DNS name in your own domain for the website using a DNS CNAME, or an Amazon Route 53 alias that resolves to the Amazon S3 website URL.
7. The website will now be available at your website domain name.

Amazon S3 Advanced Features

Beyond the basics, there are some advanced features of Amazon S3 that you should also be familiar with.

Prefixes and Delimiters

While Amazon S3 uses a flat structure in a bucket, it supports the use of *prefix* and *delimiter* parameters when listing key names. This feature lets you organize, browse, and retrieve the objects within a bucket hierarchically. Typically, you would use a slash (`/`) or backslash (`\`) as a delimiter and then use key names with embedded delimiters to emulate a file and folder hierarchy within the flat object key namespace of a bucket.

For example, you might want to store a series of server logs by server name (such as server42), but organized by year and month, like so:

```
logs/2016/January/server42.log
logs/2016/February/server42.log
logs/2016/March/server42.log
```

The REST API, wrapper SDKs, AWS CLI, and the Amazon Management Console all support the use of delimiters and prefixes. This feature lets you logically organize new data and easily maintain the hierarchical folder-and-file structure of existing data uploaded or backed up from traditional file systems. Used together with IAM or Amazon S3 bucket policies, prefixes and delimiters also allow you to create the equivalent of departmental “subdirectories” or user “home directories” within a single bucket, restricting or sharing access to these “subdirectories” (defined by prefixes) as needed.

Use delimiters and object prefixes to hierarchically organize the objects in your Amazon S3 buckets, but always remember that Amazon S3 is not really a file system.

Storage Classes

Amazon S3 offers a range of *storage classes* suitable for various use cases.

Amazon S3 Standard offers high durability, high availability, low latency, and high performance object storage for general purpose use. Because it delivers low first-byte latency and high throughput, Standard is well-suited for short-term or long-term storage of frequently accessed data. For most general purpose use cases, Amazon S3 Standard is the place to start.

Amazon S3 Standard – Infrequent Access (Standard-IA) offers the same durability, low latency, and high throughput as Amazon S3 Standard, but is designed for long-lived, less frequently accessed data. Standard-IA has a lower per GB-month storage cost than Standard, but the price model also includes a minimum object size (128KB), minimum duration (30 days), and per-GB retrieval costs, so it is best suited for infrequently accessed data that is stored for longer than 30 days.

Amazon S3 Reduced Redundancy Storage (RRS) offers slightly lower durability (4 nines) than Standard or Standard-IA at a reduced cost. It is most appropriate for derived data that can be easily reproduced, such as image thumbnails.

Finally, the *Amazon Glacier* storage class offers secure, durable, and extremely low-cost cloud storage for data that does not require real-time access, such as archives and long-term backups. To keep costs low, Amazon Glacier is optimized for infrequently accessed data where a retrieval time of several hours is suitable. To retrieve an Amazon Glacier object, you issue a restore command using one of the Amazon S3 APIs; three to five hours later, the Amazon Glacier object is copied to Amazon S3 RRS. Note that the restore simply creates a copy in Amazon S3 RRS; the original data object remains in Amazon Glacier until explicitly deleted. Also be aware that Amazon Glacier allows you to retrieve up to 5% of the Amazon S3 data stored in Amazon Glacier for free each month; restores beyond the daily restore allowance incur a restore fee. Refer to the Amazon Glacier pricing page on the AWS website for full details.

In addition to acting as a storage tier in Amazon S3, Amazon Glacier is also a standalone storage service with a separate API and some unique characteristics. However, when you use Amazon Glacier as a storage class of Amazon S3, you always interact with the data via the Amazon S3 APIs. Refer to the Amazon Glacier section for more details.


Set a data retrieval policy to limit restores to the free tier or to a maximum GB-per-hour limit to avoid or minimize Amazon Glacier restore fees.

Object Lifecycle Management

Amazon S3 *Object Lifecycle Management* is roughly equivalent to automated *storage tiering* in traditional IT storage infrastructures. In many cases, data has a natural lifecycle, starting out as "hot" (frequently accessed) data, moving to "warm" (less frequently accessed) data as it ages, and ending its life as "cold" (long-term backup or archive) data before eventual deletion.

For example, many business documents are frequently accessed when they are created, then become much less frequently accessed over time. In many cases, however, compliance rules require business documents to be archived and kept accessible for years. Similarly, studies show that file, operating system, and database backups are most frequently accessed in the first few days after they are created, usually to restore after an inadvertent error. After a week or two, these backups remain a critical asset, but they are much less likely to be accessed for a restore. In many cases, compliance rules require that a certain number of backups be kept for several years.

Using Amazon S3 lifecycle configuration rules, you can significantly reduce your storage costs by automatically transitioning data from one storage class to another or even automatically deleting data after a period of time. For example, the lifecycle rules for backup data might be:

- Store backup data initially in Amazon S3 Standard.
- After 30 days, transition to Amazon Standard-IA.
- After 90 days, transition to Amazon Glacier.

- After 3 years, delete.

Lifecycle configurations are attached to the bucket and can apply to all objects in the bucket or only to objects specified by a prefix.

Encryption

It is strongly recommended that all sensitive data stored in Amazon S3 be encrypted, both in flight and at rest.

To encrypt your Amazon S3 data in flight, you can use the Amazon S3 Secure Sockets Layer (SSL) API endpoints. This ensures that all data sent to and from Amazon S3 is encrypted while in transit using the HTTPS protocol.

To encrypt your Amazon S3 data at rest, you can use several variations of *Server-Side Encryption (SSE)*. Amazon S3 encrypts your data at the object level as it writes it to disks in its data centers and decrypts it for you when you access it. All SSE performed by Amazon S3 and AWS Key Management Service (Amazon KMS) uses the 256-bit Advanced Encryption Standard (AES). You can also encrypt your Amazon S3 data at rest using *Client-Side Encryption*, encrypting your data on the client before sending it to Amazon S3.

SSE-S3 (AWS-Managed Keys)

This is a fully integrated "check-box-style" encryption solution where AWS handles the key management and key protection for Amazon S3. Every object is encrypted with a unique key. The actual object key itself is then further encrypted by a separate master key. A new master key is issued at least monthly, with AWS rotating the keys. Encrypted data, encryption keys, and master keys are all stored separately on secure hosts, further enhancing protection.

SSE-KMS (AWS KMS Keys)

This is a fully integrated solution where Amazon handles your key management and protection for Amazon S3, but where you manage the keys. SSE-KMS offers several additional benefits compared to SSE-S3. Using SSE-KMS, there are separate permissions for using the master key, which provide protection against unauthorized access to your objects stored in Amazon S3 and an additional layer of control. AWS KMS also provides auditing, so you can see who used your key to access which object and when they tried to access this object. AWS KMS also allows you to view any failed attempts to access data from users who did not have permission to decrypt the data.

SSE-C (Customer-Provided Keys)

This is used when you want to maintain your own encryption keys but don't want to manage or implement your own client-side encryption library. With SSE-C, AWS will do the encryption/decryption of your objects while you maintain full control of the keys used to encrypt/decrypt the objects in Amazon S3.

Client-Side Encryption

Client-side encryption refers to encrypting data on the client side of your application before sending it to Amazon S3. You have the following two options for using data encryption keys:

- Use an AWS KMS-managed customer master key.
- Use a client-side master key.

When using client-side encryption, you retain end-to-end control of the encryption process, including management of the encryption keys.

 For maximum simplicity and ease of use, use server-side encryption with AWS-managed keys (SSE-S3 or SSE-KMS).

Versioning

Amazon S3 versioning helps protects your data against accidental or malicious deletion by keeping multiple versions of each object in the bucket, identified by a unique version ID. Versioning allows you to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. If a user makes an accidental change or even maliciously deletes an object in your S3 bucket, you can restore the object to its original state simply by referencing the version ID in addition to the bucket and object key. Versioning is turned on at the bucket level. Once enabled, versioning cannot be removed from a bucket; it can only be suspended.

MFA Delete

MFA Delete adds another layer of data protection on top of bucket versioning. MFA Delete requires additional authentication in order to permanently delete an object version or change the versioning state of a bucket. In addition to your normal security credentials, MFA Delete requires an authentication code (a temporary, one-time password) generated by a hardware or virtual Multi-Factor Authentication (MFA) device. Note that MFA Delete can only be enabled by the root account.

Pre-Signed URLs

All Amazon S3 objects by default are private, meaning that only the owner has access. However, the object owner can optionally share objects with others by creating a *pre-signed URL*, using their own security credentials to grant time-limited permission to download the objects. When you create a pre-signed URL for your object, you must provide your security credentials and specify a bucket name, an object key, the HTTP method (`GET` to download the object), and an expiration date and time. The pre-signed URLs are valid only for the specified duration. This is particularly useful to protect against "content scraping" of web content such as media files stored in Amazon S3.

Multipart Upload

To better support uploading or copying of large objects, Amazon S3 provides the Multipart Upload API. This allows you to upload large objects as a set of parts, which generally gives better network utilization (through parallel transfers), the ability to pause and resume, and the ability to upload objects where the size is initially unknown.

Multipart upload is a three-step process: initiation, uploading the parts, and completion (or

abort). Parts can be uploaded independently in arbitrary order, with retransmission if needed. After all of the parts are uploaded, Amazon S3 assembles the parts in order to create an object.

In general, you *should* use multipart upload for objects larger than 100 Mbytes, and you *must* use multipart upload for objects larger than 5GB. When using the low-level APIs, you must break the file to be uploaded into parts and keep track of the parts. When using the high-level APIs and the high-level Amazon S3 commands in the AWS CLI (`aws s3 cp`, `aws s3 mv`, and `aws s3 sync`), multipart upload is automatically performed for large objects.

 You can set an object lifecycle policy on a bucket to abort incomplete multipart uploads after a specified number of days. This will minimize the storage costs associated with multipart uploads that were not completed.

Range GETs

It is possible to download (`GET`) only a portion of an object in both Amazon S3 and Amazon Glacier by using something called a *Range `GET`*. Using the Range HTTP header in the `GET` request or equivalent parameters in one of the SDK wrapper libraries, you specify a range of bytes of the object. This can be useful in dealing with large objects when you have poor connectivity or to download only a known portion of a large Amazon Glacier backup.

Cross-Region Replication

Cross-region replication is a feature of Amazon S3 that allows you to asynchronously replicate all new objects in the source bucket in one AWS region to a target bucket in another region. Any metadata and ACLs associated with the object are also part of the replication. After you set up cross-region replication on your source bucket, any changes to the data, metadata, or ACLs on an object trigger a new replication to the destination bucket. To enable cross-region replication, versioning must be turned on for both source and destination buckets, and you must use an IAM policy to give Amazon S3 permission to replicate objects on your behalf.

Cross-region replication is commonly used to reduce the latency required to access objects in Amazon S3 by placing objects closer to a set of users or to meet requirements to store backup data at a certain distance from the original source data.

 If turned on in an existing bucket, cross-region replication will only replicate new objects. Existing objects will not be replicated and must be copied to the new bucket via a separate command.

Logging

In order to track requests to your Amazon S3 bucket, you can enable Amazon S3 server access logs. Logging is off by default, but it can easily be enabled. When you enable logging for a

bucket (the source bucket), you must choose where the logs will be stored (the target bucket). You can store access logs in the same bucket or in a different bucket. Either way, it is optional (but a best practice) to specify a prefix, such as `logs/` or `yourbucketname/logs/`, so that you can more easily identify your logs.

Once enabled, logs are delivered on a best-effort basis with a slight delay. Logs include information such as:

- Requestor account and IP address
- Bucket name
- Request time
- Action (`GET`, `PUT`, `LIST`, and so forth)
- Response status or error code

Event Notifications

Amazon S3 *event notifications* can be sent in response to actions taken on objects uploaded or stored in Amazon S3. Event notifications enable you to run workflows, send alerts, or perform other actions in response to changes in your objects stored in Amazon S3. You can use Amazon S3 event notifications to set up triggers to perform actions, such as transcoding media files when they are uploaded, processing data files when they become available, and synchronizing Amazon S3 objects with other data stores.

Amazon S3 event notifications are set up at the bucket level, and you can configure them through the Amazon S3 console, through the REST API, or by using an AWS SDK. Amazon S3 can publish notifications when new objects are created (by a `PUT`, `POST`, `COPY`, or multipart upload completion), when objects are removed (by a `DELETE`), or when Amazon S3 detects that an RRS object was lost. You can also set up event notifications based on object name prefixes and suffixes. Notification messages can be sent through either Amazon Simple Notification Service (Amazon SNS) or Amazon Simple Queue Service (Amazon SQS) or delivered directly to AWS Lambda to invoke AWS Lambda functions.

Best Practices, Patterns, and Performance

It is a common pattern to use Amazon S3 storage in hybrid IT environments and applications. For example, data in on-premises file systems, databases, and compliance archives can easily be backed up over the Internet to Amazon S3 or Amazon Glacier, while the primary application or database storage remains on-premises.

Another common pattern is to use Amazon S3 as bulk “blob” storage for data, while keeping an index to that data in another service, such as Amazon DynamoDB or Amazon RDS. This allows quick searches and complex queries on key names without listing keys continually.

Amazon S3 will scale automatically to support very high request rates, automatically re-partitioning your buckets as needed. If you need request rates higher than 100 requests per second, you may want to review the Amazon S3 best practices guidelines in the Developer Guide. To support higher request rates, it is best to ensure some level of random distribution of keys, for example by including a hash as a prefix to key names.

 If you are using Amazon S3 in a `GET`-intensive mode, such as a static website hosting, for best performance you should consider using an Amazon CloudFront distribution as a caching layer in front of your Amazon S3 bucket.

Amazon Glacier

Amazon Glacier is an extremely low-cost storage service that provides durable, secure, and flexible storage for data archiving and online backup. To keep costs low, Amazon Glacier is designed for infrequently accessed data where a retrieval time of three to five hours is acceptable.

Amazon Glacier can store an unlimited amount of virtually any kind of data, in any format. Common use cases for Amazon Glacier include replacement of traditional tape solutions for long-term backup and archive and storage of data required for compliance purposes. In most cases, the data stored in Amazon Glacier consists of large TAR (Tape Archive) or ZIP files.

Like Amazon S3, Amazon Glacier is extremely durable, storing data on multiple devices across multiple facilities in a region. Amazon Glacier is designed for 99.999999999% durability of objects over a given year.

Archives

In Amazon Glacier, data is stored in *archives*. An archive can contain up to 40TB of data, and you can have an unlimited number of archives. Each archive is assigned a unique archive ID at the time of creation. (Unlike an Amazon S3 object key, you cannot specify a user-friendly archive name.) All archives are automatically encrypted, and archives are immutable—after an archive is created, it cannot be modified.

Vaults

Vaults are containers for archives. Each AWS account can have up to 1,000 vaults. You can control access to your vaults and the actions allowed using IAM policies or vault access policies.

Vaults Locks

You can easily deploy and enforce compliance controls for individual Amazon Glacier vaults with a *vault lock* policy. You can specify controls such as Write Once Read Many (WORM) in a vault lock policy and lock the policy from future edits. Once locked, the policy can no longer be changed.

Data Retrieval

You can retrieve up to 5% of your data stored in Amazon Glacier for free each month, calculated on a daily prorated basis. If you retrieve more than 5%, you will incur retrieval fees based on your maximum retrieval rate. To eliminate or minimize those fees, you can set a data retrieval policy on a vault to limit your retrievals to the free tier or to a specified data rate.

Amazon Glacier versus Amazon Simple Storage Service (Amazon S3)

Amazon Glacier is similar to Amazon S3, but it differs in several key aspects. Amazon Glacier supports 40TB archives versus 5TB objects in Amazon S3. Archives in Amazon Glacier are

identified by system-generated archive IDs, while Amazon S3 lets you use “friendly” key names. Amazon Glacier archives are automatically encrypted, while encryption at rest is optional in Amazon S3. However, by using Amazon Glacier as an Amazon S3 storage class together with object lifecycle policies, you can use the Amazon S3 interface to get most of the benefits of Amazon Glacier without learning a new interface.

Summary

Amazon S3 is the core object storage service on AWS, allowing you to store an unlimited amount of data with very high durability.

Common Amazon S3 use cases include backup and archive, web content, big data analytics, static website hosting, mobile and cloud-native application hosting, and disaster recovery.

Amazon S3 is integrated with many other AWS cloud services, including AWS IAM, AWS KMS, Amazon EC2, Amazon EBS, Amazon EMR, Amazon DynamoDB, Amazon Redshift, Amazon SQS, AWS Lambda, and Amazon CloudFront.

Object storage differs from traditional block and file storage. Block storage manages data at a device level as addressable blocks, while file storage manages data at the operating system level as files and folders. Object storage manages data as objects that contain both data and metadata, manipulated by an API.

Amazon S3 buckets are containers for objects stored in Amazon S3. Bucket names must be globally unique. Each bucket is created in a specific region, and data does not leave the region unless explicitly copied by the user.

Amazon S3 objects are files stored in buckets. Objects can be up to 5TB and can contain any kind of data. Objects contain both data and metadata and are identified by keys. Each Amazon S3 object can be addressed by a unique URL formed by the web services endpoint, the bucket name, and the object key.

Amazon S3 has a minimalistic API—create/delete a bucket, read/write/delete objects, list keys in a bucket—and uses a REST interface based on standard HTTP verbs—`GET`, `PUT`, `POST`, and `DELETE`. You can also use SDK wrapper libraries, the AWS CLI, and the AWS Management Console to work with Amazon S3.

Amazon S3 is highly durable and highly available, designed for 11 nines of durability of objects in a given year and four nines of availability.

Amazon S3 is eventually consistent, but offers read-after-write consistency for new object `PUTS`.

Amazon S3 objects are private by default, accessible only to the owner. Objects can be marked public readable to make them accessible on the web. Controlled access may be provided to others using ACLs and AWS IAM and Amazon S3 bucket policies.

Static websites can be hosted in an Amazon S3 bucket.

Prefixes and delimiters may be used in key names to organize and navigate data hierarchically much like a traditional file system.

Amazon S3 offers several storage classes suited to different use cases: Standard is designed for general-purpose data needing high performance and low latency. Standard-IA is for less frequently accessed data. RRS offers lower redundancy at lower cost for easily reproduced data. Amazon Glacier offers low-cost durable storage for archive and long-term backups that can are rarely accessed and can accept a three- to five-hour retrieval time.

Object lifecycle management policies can be used to automatically move data between

storage classes based on time.

Amazon S3 data can be encrypted using server-side or client-side encryption, and encryption keys can be managed with Amazon KMS.

Versioning and MFA Delete can be used to protect against accidental deletion.

Cross-region replication can be used to automatically copy new objects from a source bucket in one region to a target bucket in another region.

Pre-signed URLs grant time-limited permission to download objects and can be used to protect media and other web content from unauthorized "web scraping."

Multipart upload can be used to upload large objects, and Range GETs can be used to download portions of an Amazon S3 object or Amazon Glacier archive.

Server access logs can be enabled on a bucket to track requestor, object, action, and response.

Amazon S3 event notifications can be used to send an Amazon SQS or Amazon SNS message or to trigger an AWS Lambda function when an object is created or deleted.

Amazon Glacier can be used as a standalone service or as a storage class in Amazon S3.

Amazon Glacier stores data in archives, which are contained in vaults. You can have up to 1,000 vaults, and each vault can store an unlimited number of archives.

Amazon Glacier vaults can be locked for compliance purposes.

Exam Essentials

Know what amazon s3 is and what it is commonly used for. Amazon S3 is secure, durable, and highly scalable cloud storage that can be used to store an unlimited amount of data in almost any format using a simple web services interface. Common use cases include backup and archive, content storage and distribution, big data analytics, static website hosting, cloud-native application hosting, and disaster recovery.

Understand how object storage differs from block and file storage. Amazon S3 cloud object storage manages data at the application level as objects using a REST API built on HTTP. Block storage manages data at the operating system level as numbered addressable blocks using protocols such as SCSI or Fibre Channel. File storage manages data as shared files at the operating system level using a protocol such as CIFS or NFS.

Understand the basics of Amazon S3. Amazon S3 stores data in objects that contain data and metadata. Objects are identified by a user-defined key and are stored in a simple flat folder called a bucket. Interfaces include a native REST interface, SDKs for many languages, an AWS CLI, and the AWS Management Console.

Know how to create a bucket; how to upload, download, and delete objects; how to make objects public; and how to open an object URL.

Understand the durability, availability, and data consistency model of Amazon S3. Amazon S3 standard storage is designed for 11 nines durability and four nines availability of objects over a year. Other storage classes differ. Amazon S3 is eventually consistent, but offers read-after-write consistency for PUTS to new objects.

Know how to enable static website hosting on Amazon S3. To create a static website on Amazon S3, you must create a bucket with the website hostname, upload your static content and make it public, enable static website hosting on the bucket, and indicate the index and error page objects.

Know how to protect your data on Amazon S3. Encrypt data in flight using HTTPS and at rest using SSE or client-side encryption. Enable versioning to keep multiple versions of an object in a bucket. Enable MFA Delete to protect against accidental deletion. Use ACLs Amazon S3 bucket policies and AWS IAM policies for access control. Use pre-signed URLs for time-limited download access. Use cross-region replication to automatically replicate data to another region.

Know the use case for each of the Amazon S3 storage classes. Standard is for general purpose data that needs high durability, high performance, and low latency access. Standard-IA is for data that is less frequently accessed, but that needs the same performance and availability when accessed. RRS offers lower durability at lower cost for easily replicated data. Amazon Glacier is for storing rarely accessed archival data at lowest cost, when three- to five-hour retrieval time is acceptable.

Know how to use lifecycle configuration rules. Lifecycle rules can be configured in the AWS Management Console or the APIs. Lifecycle configuration rules define actions to transition objects from one storage class to another based on time.

Know how to use Amazon S3 event notifications. Event notifications are set at the

bucket level and can trigger a message in Amazon SNS or Amazon SQS or an action in AWS Lambda in response to an upload or a delete of an object.

Know the basics of amazon glacier as a standalone service. Data is stored in encrypted archives that can be as large as 40TB. Archives typically contain TAR or ZIP files. Vaults are containers for archives, and vaults can be locked for compliance.

Exercises

For assistance in completing the following exercises, reference the following documentation:

- Getting started with Amazon S3: http://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html
- Setting up a static website: http://docs.aws.amazon.com/AmazonS3/latest/dev/HostingWebsiteOnS3Setup.html
- Using versioning: http://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html
- Object Lifecycle Management: http://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html

EXERCISE 2.1

Create an Amazon Simple Storage Service (Amazon S3) Bucket

In this exercise, you will create a new Amazon S3 bucket in your selected region. You will use this bucket in the following exercises.

1. Log in to the AWS Management Console.
2. Choose an appropriate region, such as US West (Oregon).
3. Navigate to the Amazon S3 console. Notice that the region indicator now says Global. Remember that Amazon S3 buckets form a global namespace, even though each bucket is created in a specific region.
4. Start the create bucket process.
5. When prompted for Bucket Name, use **`mynewbucket`**.
6. Choose a region, such as US West (Oregon).
7. Try to create the bucket. You almost surely will get a message that the requested bucket name is not available. Remember that a bucket name must be unique globally.
8. Try again using your surname followed by a hyphen and then today's date in a six-digit format as the bucket name (a bucket name that is not likely to exist already).

You should now have a new Amazon S3 bucket.

EXERCISE 2.2

Upload, Make Public, Rename, and Delete Objects in Your Bucket

In this exercise, you will upload a new object to your bucket. You will then make this object public and view the object in your browser. You will then rename the object and finally delete it from the bucket.

Upload an Object

1. Load your new bucket in the Amazon S3 console.
2. Select Upload, then Add Files.
3. Locate a file on your PC that you are okay with uploading to Amazon S3 and making public to the Internet. (We suggest using a non-personal image file for the purposes of this exercise.)
4. Select a suitable file, then Start Upload. You will see the status of your file in the Transfers section.
5. After your file is uploaded, the status should change to Done.

The file you uploaded is now stored as an Amazon S3 object and should be now listed in the contents of your bucket.

Open the Amazon S3 URL

6. Now open the properties for the object. The properties should include bucket, name, and link.
7. Copy the Amazon S3 URL for the object.
8. Paste the URL in the address bar of a new browser window or tab.

You should get a message with an XML error code `AccessDenied`. Even though the object has a URL, it is private by default, so it cannot be accessed by a web browser.

Make the Object Public

9. Go back to the Amazon S3 Console and select Make Public. (Equivalently, you can change the object's permissions and add grantee Everyone and permissions Open/Download.)
10. Copy the Amazon S3 URL again and try to open it in a browser or tab. Your public image file should now display in the browser or browser tab.

Rename Object

11. In the Amazon S3 console, select Rename.
12. Rename the object, but keep the same file extension.
13. Copy the new Amazon S3 URL and try to open it in a browser or tab. You should see the same image file.

Delete the Object

14. In the Amazon S3 console, select Delete. Select OK when prompted if you want to delete the object.
15. The object has now been deleted.
16. To verify, try to reload the deleted object's Amazon S3 URL.

You should once again get the XML `AccessDenied` error message.

EXERCISE 2.3

Enable Version Control

In this exercise, you will enable version control on your newly created bucket.

Enable Versioning

1. In the Amazon S3 console, load the properties of your bucket. Don't open the bucket.
2. Enable versioning in the properties and select OK to verify. Your bucket now has versioning enabled. (Note that versioning can be suspended, but not turned off.)

Create Multiple Versions of an Object

3. Create a text file named `foo.txt` on your computer and write the word **`blue`** in the text file.
4. Save the text file to a location of your choosing.
5. Upload the text file to your bucket. This will be version 1.
6. After you have uploaded the text file to your bucket, open the copy on your local computer and change the word **`blue`** to **`red`**. Save the text file with the original filename.
7. Upload the modified file to your bucket.
8. Select Show Versions on the uploaded object.

You will now see two different versions of the object with different Version IDs and possibly different sizes. Note that when you select Show Version, the Amazon S3 URL now includes the version ID in the query string after the object name.

EXERCISE 2.4

Delete an Object and Then Restore It

In this exercise, you will delete an object in your Amazon S3 bucket and then restore it.

Delete an Object

1. Open the bucket containing the text file for which you now have two versions.
2. Select Hide Versions.
3. Select Delete, and then select OK to verify.
4. Your object will now be deleted, and you can no longer see the object.
5. Select Show Versions.

Both versions of the object now show their version IDs.

Restore an Object

6. Open your bucket.
7. Select Show Versions.
8. Select the oldest version and download the object. Note that the filename is simply `foo.txt` with no version indicator.
9. Upload `foo.txt` to the same bucket.
10. Select Hide Versions, and the file `foo.txt` should re-appear.

To restore a version, you copy the desired version into the same bucket. In the Amazon S3 console, this requires a download then re-upload of the object. Using APIs, SDKs, or AWS CLI, you can copy a version directly without downloading and re-uploading.

EXERCISE 2.5

Lifecycle Management

In this exercise, you will explore the various options for lifecycle management.

1. Select your bucket in the Amazon S3 console.
2. Under Properties, add a Lifecycle Rule.
3. Explore the various options to add lifecycle rules to objects in this bucket. It is recommended that you do not implement any of these options, as you may incur additional costs. After you have finished, click the Cancel button.

Most lifecycle rules require some number of days to expire before the transition takes effect. For example, it takes a minimum of 30 days to transition from Amazon S3 Standard to Amazon S3 Standard-IA. This makes it impractical to create a lifecycle rule and see the actual result in an exercise.

EXERCISE 2.6

Enable Static Hosting on Your Bucket

In this exercise, you will enable static hosting on your newly created bucket.

1. Select your bucket in the Amazon S3 console.
2. In the Properties section, select Enable Website Hosting.
3. For the index document name, enter **`index.txt`**, and for the error document name, enter **`error.txt`**.
4. Use a text editor to create two text files and save them as **`index.txt`** and **`error.txt`**. In the `index.txt` file, write the phrase "**`Hello World`**," and in the `error.txt` file, write the phrase "**`Error Page`**." Save both text files and upload them to your bucket.
5. Make the two objects public.
6. Copy the Endpoint: link under Static Website Hosting and paste it in a browser window or tab. You should now see the phrase `"Hello World"` displayed.
7. In the address bar in your browser, try adding a forward slash followed by a made-up filename (for example, `/test.html`). You should now see the phrase `"Error Page"` displayed.
8. To clean up, delete all of the objects in your bucket and then delete the bucket itself.

Review Questions

1. In what ways does Amazon Simple Storage Service (Amazon S3) object storage differ from block and file storage? (Choose 2 answers)
   A. Amazon S3 stores data in fixed size blocks.
   B. Objects are identified by a numbered address.
   C. Objects can be any size.
   D. Objects contain both data and metadata.
   E. Objects are stored in buckets.
2. Which of the following are not appropriates use cases for Amazon Simple Storage Service (Amazon S3)? (Choose 2 answers)
   A. Storing web content
   B. Storing a file system mounted to an Amazon Elastic Compute Cloud (Amazon EC2) instance
   C. Storing backups for a relational database
   D. Primary storage for a database
   E. Storing logs for analytics
3. What are some of the key characteristics of Amazon Simple Storage Service (Amazon S3)? (Choose 3 answers)
   A. All objects have a URL.
   B. Amazon S3 can store unlimited amounts of data.
   C. Objects are world-readable by default.
   D. Amazon S3 uses a REST (Representational State Transfer) Application Program Interface (API).
   E. You must pre-allocate the storage in a bucket.
4. Which features can be used to restrict access to Amazon Simple Storage Service (Amazon S3) data? (Choose 3 answers)
   A. Enable static website hosting on the bucket.
   B. Create a pre-signed URL for an object.
   C. Use an Amazon S3 Access Control List (ACL) on a bucket or object.
   D. Use a lifecycle policy.
   E. Use an Amazon S3 bucket policy.
5. Your application stores critical data in Amazon Simple Storage Service (Amazon S3), which must be protected against inadvertent or intentional deletion. How can this data be protected? (Choose 2 answers)

A. Use cross-region replication to copy data to another bucket automatically.

B. Set a vault lock.

C. Enable versioning on the bucket.

D. Use a lifecycle policy to migrate data to Amazon Glacier.

E. Enable MFA Delete on the bucket.

6. Your company stores documents in Amazon Simple Storage Service (Amazon S3), but it wants to minimize cost. Most documents are used actively for only about a month, then much less frequently. However, all data needs to be available within minutes when requested. How can you meet these requirements?

   A. Migrate the data to Amazon S3 Reduced Redundancy Storage (RRS) after 30 days.

   B. Migrate the data to Amazon Glacier after 30 days.

   C. Migrate the data to Amazon S3 Standard – Infrequent Access (IA) after 30 days.

   D. Turn on versioning, then migrate the older version to Amazon Glacier.

7. How is data stored in Amazon Simple Storage Service (Amazon S3) for high durability?

   A. Data is automatically replicated to other regions.

   B. Data is automatically replicated within a region.

   C. Data is replicated only if versioning is enabled on the bucket.

   D. Data is automatically backed up on tape and restored if needed.

8. Based on the following Amazon Simple Storage Service (Amazon S3) URL, which one of the following statements is correct?

   https://bucket1.abc.com.s3.amazonaws.com/folderx/myfile.doc

   A. The object “myfile.doc” is stored in the folder “folderx” in the bucket “bucket1.abc.com.”

   B. The object “myfile.doc” is stored in the bucket “bucket1.abc.com.”

   C. The object “folderx/myfile.doc” is stored in the bucket “bucket1.abc.com.”

   D. The object “myfile.doc” is stored in the bucket “bucket1.”

9. To have a record of who accessed your Amazon Simple Storage Service (Amazon S3) data and from where, you should do what?

   A. Enable versioning on the bucket.

   B. Enable website hosting on the bucket.

   C. Enable server access logs on the bucket.

   D. Create an AWS Identity and Access Management (IAM) bucket policy.

   E. Enable Amazon CloudWatch logs.

10. What are some reasons to enable cross-region replication on an Amazon Simple Storage Service (Amazon S3) bucket? (Choose 2 answers)

A. You want a backup of your data in case of accidental deletion.

B. You have a set of users or customers who can access the second bucket with lower latency.

C. For compliance reasons, you need to store data in a location at least 300 miles away from the first region.

D. Your data needs at least five nines of durability.

11. Your company requires that all data sent to external storage be encrypted before being sent. Which Amazon Simple Storage Service (Amazon S3) encryption solution will meet this requirement?

A. Server-Side Encryption (SSE) with AWS-managed keys (SSE-S3)

B. SSE with customer-provided keys (SSE-C)

C. Client-side encryption with customer-managed keys

D. Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)

12. You have a popular web application that accesses data stored in an Amazon Simple Storage Service (Amazon S3) bucket. You expect the access to be very read-intensive, with expected request rates of up to 500 GETs per second from many clients. How can you increase the performance and scalability of Amazon S3 in this case?

A. Turn on cross-region replication to ensure that data is served from multiple locations.

B. Ensure randomness in the namespace by including a hash prefix to key names.

C. Turn on server access logging.

D. Ensure that key names are sequential to enable pre-fetch.

13. What is needed before you can enable cross-region replication on an Amazon Simple Storage Service (Amazon S3) bucket? (Choose 2 answers)

A. Enable versioning on the bucket.

B. Enable a lifecycle rule to migrate data to the second region.

C. Enable static website hosting.

D. Create an AWS Identity and Access Management (IAM) policy to allow Amazon S3 to replicate objects on your behalf.

14. Your company has 100TB of financial records that need to be stored for seven years by law. Experience has shown that any record more than one-year old is unlikely to be accessed. Which of the following storage plans meets these needs in the most cost efficient manner?

A. Store the data on Amazon Elastic Block Store (Amazon EBS) volumes attached to t2.micro instances.

B. Store the data on Amazon Simple Storage Service (Amazon S3) with lifecycle policies that change the storage class to Amazon Glacier after one year and delete the object

after seven years.

C. Store the data in Amazon DynamoDB and run daily script to delete data older than seven years.

D. Store the data in Amazon Elastic MapReduce (Amazon EMR).

15. Amazon Simple Storage Service (S3) bucket policies can restrict access to an Amazon S3 bucket and objects by which of the following? (Choose 3 answers)

A. Company name

B. IP address range

C. AWS account

D. Country of origin

E. Objects with a specific prefix

16. Amazon Simple Storage Service (Amazon S3) is an eventually consistent storage system. For what kinds of operations is it possible to get stale data as a result of eventual consistency? (Choose 2 answers)

A. GET after PUT of a new object

B. GET or LIST after a DELETE

C. GET after overwrite PUT (PUT to an existing key)

D. DELETE after PUT of new object

17. What must be done to host a static website in an Amazon Simple Storage Service (Amazon S3) bucket? (Choose 3 answers)

A. Configure the bucket for static hosting and specify an index and error document.

B. Create a bucket with the same name as the website.

C. Enable File Transfer Protocol (FTP) on the bucket.

D. Make the objects in the bucket world-readable.

E. Enable HTTP on the bucket.

18. You have valuable media files hosted on AWS and want them to be served only to authenticated users of your web application. You are concerned that your content could be stolen and distributed for free. How can you protect your content?

A. Use static web hosting.

B. Generate pre-signed URLs for content in the web application.

C. Use AWS Identity and Access Management (IAM) policies to restrict access.

D. Use logging to track your content.

19. Amazon Glacier is well-suited to data that is which of the following? (Choose 2 answers)

A. Is infrequently or rarely accessed

B. Must be immediately available when needed

C. Is available after a three- to five-hour restore period

D. Is frequently erased within 30 days

20. Which statements about Amazon Glacier are true? (Choose 3 answers)

A. Amazon Glacier stores data in objects that live in archives.

B. Amazon Glacier archives are identified by user-specified key names.

C. Amazon Glacier archives take three to five hours to restore.

D. Amazon Glacier vaults can be locked.

E. Amazon Glacier can be used as a standalone service and as an Amazon S3 storage class.