MODULE 8 · STUDY NOTES

Storage and Archiving

The four AWS storage categories, EBS volumes and snapshots, instance store, EFS, S3 objects and storage classes, Glacier archiving, Storage Gateway, and data-migration services

Every application holds its information somewhere, and AWS offers a complete range of storage services to hold it — the craft is matching the service to the workload. This module sorts AWS storage into four categories: **block** storage (Amazon EBS) for a single instance's disk, **object** storage (Amazon S3 and S3 Glacier) for internet-accessible files and long-term archives, **file** storage (Amazon EFS and Amazon FSx) for shared file systems, and **hybrid** storage (AWS Storage Gateway) that bridges an on-premises data center to the cloud. Along the way it covers how EBS volumes are created, tuned, and backed up with incremental snapshots; the temporary speed of instance store; the durability, versioning, storage classes, and lifecycle rules that make S3 flexible and cheap to operate; how Glacier archives data with retrieval by request; and the services — AWS Transfer for SFTP, DataSync, and Snowball — that move data into the cloud in the first place.

LEARNING OBJECTIVES

- Differentiate the AWS data storage options and explain their purpose and benefits
- Create and manage Amazon EBS snapshots
- Store, retrieve, and archive Amazon S3 objects
- Identify AWS data migration services

8.1 Cloud storage overview

Cloud storage is a critical component of cloud computing — it holds the information that applications use. Big data analytics, data warehouses, the Internet of Things, databases, and backup and archive applications all rely on some form of data storage. AWS groups its storage services into **four categories**, and choosing well starts with recognizing which category a workload needs: a disk for one instance is **block** storage, internet-reachable files are **objects**, a shared file system is **file** storage, and a link back to a data center is **hybrid** storage.

TABLE 8.1 The four AWS storage categories and their services

CATEGORY	SERVICE	PURPOSE
Block	Amazon EBS	Persistent, low-latency block volumes for a single EC2 instance
Object	Amazon S3	Secure, durable, scalable storage for objects, over the internet
Object (archive)	Amazon S3 Glacier	Low-cost, highly durable long-term backup and archive
File	Amazon EFS	Simple, scalable, elastic NFS file system for Linux workloads
File (Windows)	Amazon FSx for Windows File Server	Fully managed native Windows file system for Windows apps
Hybrid	AWS Storage Gateway	Links an on-premises environment to AWS cloud storage

The object category holds two services because they solve different problems: **Amazon S3** stores objects for active, internet-facing access, while **Amazon S3 Glacier** stores them cheaply for long-term archive. The file category likewise pairs **Amazon EFS** for Linux with **Amazon FSx for Windows File Server** for Windows applications (the FSx family also includes **Amazon FSx for Lustre**). Together these services address the common storage scenarios: storing, retrieving, and managing application data; persisting log files from transient EC2 instances for later analysis; backing up EC2 instances and their attached volumes; and backing up on-premises data for disaster recovery.

AWS also provides **data transfer and migration** services to move on-premises data into the cloud, covered fully in the last section: **AWS Transfer for SFTP** (transfer files directly into and out of S3 using SFTP), **AWS DataSync** (automate moving data between on-premises storage and S3 or EFS), and **AWS Snowball** (a large-scale, physical-device transport for very large datasets).

8.2 Amazon Elastic Block Store (Amazon EBS)

Amazon EBS provides block-level storage volumes for use with EC2 instances. The volumes are highly available and reliable, and are **network-attached** to a running instance in the **same Availability Zone**. Crucially, an EBS volume attached to an instance is exposed as a storage volume that **persists independently from the life of the instance** — the data survives even if the instance stops or is terminated. EBS supports both **boot** volumes and **data** volumes.

EBS offers several **volume types** that differ in performance and price so you can tailor storage to a workload. They are identified by a three-character name and fall into two families: **solid-state drives (SSD)**, ideal for IOPS-intensive workloads, and **hard disk drives (HDD)**, suited to MB/s (throughput)-intensive workloads.

TABLE 8.2 Amazon EBS volume types

TYPE	FAMILY	IDEAL FOR
Provisioned IOPS SSD (io1)	SSD	The highest-performance option; critical, I/O-intensive database and application workloads, and throughput-intensive data warehouse work that needs extremely low latency
General Purpose SSD (gp2)	SSD	The default type for EC2 instances; a broad range of transactional workloads, dev/test, low-latency interactive apps, and boot volumes
Throughput Optimized HDD (st1)	HDD	Frequently accessed, throughput-intensive workloads with large datasets and large I/O sizes
Cold HDD (sc1)	HDD	The lowest cost per GB of all EBS types; less frequently accessed workloads with large, cold datasets

CREATING AND ATTACHING A VOLUME (AWS CLI OR CONSOLE)

- A volume exists in one **Availability Zone**. Create it with `create-volume` and set the `--availability-zone` option; if it will attach to an existing instance, create it in **that instance's AZ**.
- Example: `aws ec2 create-volume --size 80 --availability-zone us-east-1a --volume-type gp2` makes an 80 GiB `gp2` volume.
- Inspect a volume with `describe-volumes --volume-ids`.
- Attach it to a **logical mount point** in the OS using `attach-volume` with `--device` (for example `--device /dev/sdf`); the instance must be in the same AZ.

Certain EC2 instance types can be **EBS-optimized**, giving the instance a **dedicated network connection** to its EBS volumes that raises EBS I/O performance and stabilizes it by minimizing contention with other traffic (dedicated

bandwidth ranges from **425 Mbps to 14,000 Mbps** by instance type). Optimization is **enabled by default** on instance types categorized as EBS-optimized and must be **enabled manually** on other supported types; instances built on the **AWS Nitro System**, which eliminates virtualization overhead, gain additional performance.

You back up an EBS volume by taking a **snapshot**, which copies **block-level data to Amazon S3 infrastructure** where it is redundantly stored (able to sustain the concurrent loss of data in two facilities). Take snapshots of development, test, and production data on a **regular basis** so a volume can be restored if the underlying hardware fails or the volume is accidentally deleted. The **first snapshot is a full copy** of the disk; every snapshot after that is **incremental**, capturing only the blocks that changed since the previous snapshot. Each new snapshot is effectively a recipe combining unchanged blocks from earlier snapshots with the newly changed blocks.

MANAGING SNAPSHOTS WITH THE CLI

- **Create** — `create-snapshot --volume-id ... --description "..."`. The command returns asynchronously: the point-in-time snapshot is created immediately but stays **pending** until all modified blocks reach S3.
- Before snapshotting, consider **stopping the instance or unmounting the volume** for completeness (otherwise cached, un-flushed writes may be excluded) — strongly recommended for **root-device** volumes and critical for **database servers and RAID** configurations.
- **Copy** — snapshots are **Region-scoped**; use `copy-snapshot` to move one to another Region or duplicate it in the same one. Data in transit is **automatically encrypted** (S3 256-bit AES), and the copy gets a **new snapshot ID**.
- **Restore** — a snapshot is stored in S3 but **not directly accessible**; use `create-volume` with `--snapshot-id` to restore it to a new volume. Check progress with `describe-volume-status`.

COMMON PITFALL

Volumes restored from a snapshot carry a **first-access penalty**: each block must be initialized — pulled down from S3 and written to the volume — the first time it is accessed, which spikes I/O latency. For production, eliminate the penalty by **warming** the volume: read every block that holds data before putting the volume into service.

To automate backups, use **Amazon Data Lifecycle Manager (Amazon DLM)**, which handles the **creation, retention, and deletion** of EBS snapshots. Automating snapshots enforces a regular backup schedule, retains backups for auditors or internal compliance, and reduces storage costs by deleting outdated backups. You **tag** an EBS volume and create a **lifecycle policy** whose core settings are the **resource type** (EBS volumes), a **target tag** (the tag a volume must carry to be managed), and a **schedule** (how often to create snapshots and the maximum number to keep — when a new snapshot exceeds that maximum, the oldest is deleted). Create the IAM role DLM needs with `aws dlm create-default-role`, then the policy with `aws dlm create-lifecycle-policy`.

8.3 Instance store

An **instance store** provides **temporary, non-persistent block-level storage** for an instance on disks that are **physically attached to the host computer**. It is **fast** and is **reclaimed when the instance is terminated**, making it ideal for information that changes frequently and does not need to survive the instance — **buffers, caches, scratch data**, and other temporary content. An instance store consists of one or more volumes exposed as block devices; while the store is dedicated to a particular instance, the underlying disk subsystem is shared among instances on the host.

INSTANCE STORE VOLUMES

- Available for **many instance types but not all**; the **number, size, and type (HDD vs. SSD)** vary by instance type.
- Volumes that use **NVMe** (Non-Volatile Memory express) or SATA-based SSDs deliver **higher random I/O** performance — good when you need very low latency but don't need the data to persist.
- You must **specify instance store volumes at launch** (except NVMe volumes, which are available by default); you **can't add** one after launch.
- A volume must be **mounted** before you can access it — auto-mounted by the **Windows EC2Launch** service, and typically **mounted manually on Linux**.

COMMON PITFALL

Instance store is not a backup medium. **Instance-store-backed instances cannot be stopped — only terminated** — and terminating reclaims the store, so the data is gone. Anything that must survive belongs on EBS, S3, or another persistent service.

8.4 Amazon Elastic File System (Amazon EFS)

Amazon EFS is a fully managed file-system storage service for Linux-based workloads. Files are reached through a standard **file-system interface** (using ordinary OS file I/O APIs) with full file-system semantics such as **strong consistency and file locking**. EFS uses the **Network File System (NFS) version 4.x** protocol and works with any EC2 AMI that supports it. **Multiple EC2 instances can access the same file system at the same time** — even across multiple Availability Zones in a Region — so EFS becomes a common data source for scaled-out applications, and it **automatically scales from gigabytes to petabytes** without provisioning storage. With **AWS Direct Connect**, you can even mount an EFS file system on on-premises servers to migrate or back up data.

SETTING UP AN EFS FILE SYSTEM

- **Create the file system** — it is scoped at the **Region** level.
- **Create a mount target** in the instance's VPC — it provides the **IP address for an NFSv4 endpoint** where you mount the file system.
- **Mount the file system** on the mount target.
- **Connect the EC2 instance** to the mount target. Best practice: access the file system from a mount target in the **same Availability Zone** as the instance.

MAIN EFS USE CASES

- **Shared user storage** — home directories and datasets for many users who need a common source.
- **Enterprise, web, and content applications** — scalable, durable shared storage that eases lift-and-shift migration and backs content management and web serving.
- **Dev/test and database backups** — a common code repository, and an NFSv4 target for portable database backups.
- **Big data analytics and media workflows** — high throughput with read-after-write consistency for analysis, video editing, and rendering.

EFS exposes three **performance attributes** you set to control the file system's behavior. Choose them at creation time — note that the **performance mode cannot be changed after the file system is created**.

TABLE 8.3 Amazon EFS performance attributes

ATTRIBUTE	OPTIONS	NOTES
Performance Mode	General Purpose (default), Max I/O	General Purpose suits latency-sensitive uses; Max I/O scales to higher aggregate throughput and operations with slightly higher latency (big data, media). Fixed at creation.
Storage Class	Standard, Infrequent Access	Standard is for active, frequently accessed files; Infrequent Access is a lower-cost class for long-lived, rarely accessed files.
Throughput Mode	Bursting (default), Provisioned	Bursting scales throughput with file-system size; Provisioned sets throughput in MiB/s independent of stored data, to grow beyond Bursting limits.

8.5 Amazon S3

Amazon S3 is object storage built to store and retrieve data — it is not a file system. It provides fast, durable, highly available **key-based** access to objects. S3 is **regionally provisioned** and stores content in **buckets**: a bucket is a large space that can hold any number of objects, and bucket names are **globally unique across all AWS accounts**. To access data, use the **S3 API or direct HTTP calls** — for example, an `aws s3 cp` command issues an HTTP GET through the S3 API to retrieve an object.

AMAZON S3 CHARACTERISTICS

- Can be accessed **anywhere** — inside and outside AWS; every object gets a **unique URL** built from the bucket name and the object's key.
- **Highly durable** — designed for **99.999999999% (eleven 9s)** durability through redundant storage across devices and Regions.
- **Highly available** — designed for **99.99%** availability.
- **No limit** to the amount of data stored, with up to **5 TB per object**.
- **Private by default** — all objects are protected by permissions to keep them from being exposed publicly.

S3 versioning protects objects from accidental overwrites and deletes. You enable it at the **bucket** level; each object then carries a **version ID** (its value is null when versioning is off). A key rule for the exam: once versioning has been **enabled on a bucket, it can never be disabled — only suspended**.

VERSIONING ENABLED	VERSIONING DISABLED OR SUSPENDED
history is preserved	*no new history*
▪ Uploading the same key **keeps the old object** and creates a new one with a new version ID; both are retrievable by version ID	▪ Uploading the same key **overwrites** the object; the previous one is no longer retrievable
▪ Deleting adds a **delete marker**, but the prior version is still retrievable by version ID	▪ Deleting **permanently deletes** the object
	▪ Suspended keeps existing versions but behaves like disabled for new writes

S3 offers a **range of storage classes** for different access patterns: **S3 Standard** for frequently accessed, general-purpose data (the default); **S3 Standard-IA** and **S3 One Zone-IA** for long-lived, infrequently accessed data (One Zone-IA keeps a single copy in one AZ for non-critical data); **S3 Glacier** for long-term archive and digital preservation; and **S3 Intelligent-Tiering** for unpredictable access. Intelligent-Tiering is worth special note: it **automatically moves data to the most cost-effective access tier** with no performance impact or operational overhead, keeping objects in a frequent-access tier and a lower-cost infrequent-access tier and moving any object **not accessed for 30 consecutive days** to the infrequent tier — then back to frequent access the moment it is accessed again. That makes it ideal for **long-lived data with unknown or unpredictable access patterns**.

By default, new buckets and objects **do not allow public access**, but users can loosen that with bucket policies or object permissions. **S3 Block Public Access** provides account- and bucket-level settings that **override** those resource-based policies to keep resources private, enforced regardless of how a resource was created (available from the console, CLI, S3 APIs, and CloudFormation). Best practice: allow public access **only for S3 web hosting**.

THE FOUR BLOCK PUBLIC ACCESS SETTINGS

- **Block new public ACLs and public objects** — future PUTs that include public ACLs fail; existing resources are unaffected.
- **Remove public access granted through public ACLs** — S3 ignores every public ACL when authorizing, so ACLs can't make anything public.
- **Block new public bucket policies** — a bucket policy can't be updated to grant public access.
- **Block public and cross-account access to buckets with public policies** — access to such buckets is limited to the bucket owner and AWS services.

S3 Object Lock stores objects using the **write once, read many (WORM)** model — preventing an object from being deleted or overwritten for a fixed period or indefinitely, to meet regulatory requirements. It must be **enabled when the bucket is created**. Retention is managed two ways: a **retention period** (a fixed span during which the object is WORM-protected) and a **legal hold** (the same protection with **no expiration**, staying until you explicitly remove it — independent of any retention period). A retention period also has a **mode**: in **Governance** mode, only users with special permissions can change lock settings or delete the object; in **Compliance** mode, **no user — not even the account root user** — can overwrite or delete it, and the retention period cannot be shortened.

The **event notification** feature configures a bucket to issue notifications when an object is **added, deleted, or overwritten**. Notifications go directly to **Amazon SQS queues, Amazon SNS topics, or AWS Lambda functions**, and can be **filtered by key prefix and suffix** — for example, sending only `.jpg` uploads to an SNS topic, or objects under a `logs/` prefix to a Lambda function.

TABLE 8.4 Additional Amazon S3 features

FEATURE	PURPOSE	HOW TO USE IT
Object lifecycle management	Store objects cost-effectively across their lifecycle	Create a lifecycle configuration whose rules define transition actions (change storage class) and expiration actions (delete)
Presigned object URL	Share a private object with someone who has no AWS credentials or permissions	Generate the presigned URL programmatically; the recipient uses it to access the object for a set duration
Cross-Origin Resource Sharing (CORS)	Let a bucket that hosts a static website serve requests from other origins	Add a CORS configuration (an XML document) listing the allowed origins and HTTP operations

WORKING WITH S3 FROM THE CLI

- `aws s3 ls` lists buckets; `aws s3 mb s3://mybucket` makes a bucket. Address objects with the `s3://` URI prefix.
- `aws s3 cp` copies a file, `aws s3 sync` synchronizes a directory and prefix, and `aws s3 rm` removes an object.
- Set a class at write time with `--storage-class` (values include `STANDARD`, `STANDARD_IA`, `INTELLIGENT_TIERING`, `ONEZONE_IA`, and `GLACIER`).
- `aws s3api` gives **direct access to the S3 APIs** and enables operations the high-level `aws s3` command doesn't expose — such as `put-bucket-policy`, `put-object-acl`, `create-multipart-upload`, and `list-object-versions`.

8.6 Amazon S3 Glacier

Amazon S3 Glacier is a secure, durable, and extremely low-cost service for data archiving and long-term backup. Like S3, it is designed for **eleven 9s (99.999999999%) durability** and provides strong security and compliance capabilities. You store and retrieve data using the **AWS CLI** or code that calls the **S3 Glacier APIs**, and you can use **S3 lifecycle rules** to automatically move objects from S3 into Glacier. One important limitation: there is **no console support for archive operations** — retrieval is always **by request**, not immediate access.

KEY TERMS

Archive
The basic unit of data in Glacier — a document, video, image, or any file — each assigned a unique ID by AWS.

Vault
A collection of archives, addressed by a unique name; an AWS account may create up to 1,000 vaults. An archive's URL combines the account ID, the vault name, and the archive ID.

There are **two ways to interact** with Glacier. Using an **S3 lifecycle policy**, a transition action changes an object's storage class to `GLACIER`; that data is **managed by S3 on your behalf** and **cannot be retrieved through the Glacier API** (it does not appear as a vault or archive) — instead you restore it through the S3 console or S3 API. Files added **directly through the Glacier API** are retrieved through the Glacier API: retrieval is **asynchronous** — you **initiate a job** (the Initiate Job / POST jobs REST API) and download the output after it completes, polling with the Describe Job API or receiving an **Amazon SNS** completion notification before downloading the "thawed" data. Glacier also offers **Glacier Select**, a query-in-place feature that runs analytics directly on archived data at rest without restoring it to S3.

TABLE 8.5 Amazon S3 Glacier archive retrieval options

OPTION	TYPICAL TIME	USE IT FOR
Expedited	1–5 minutes	Occasional urgent access to a subset of archives (for all but the largest archives, 250 MB+). Comes in On-Demand and guaranteed Provisioned Capacity forms.
Standard	3–5 hours	Access to any archive within several hours — the default when no option is specified.
Bulk	5–12 hours	The lowest-cost option; retrieve large amounts of data (even petabytes) inexpensively within a day.

COMMON PITFALL

Don't confuse the two paths into Glacier. Data placed in Glacier by an **S3 lifecycle transition** is S3-managed and is restored via **S3**, not the Glacier API; archives you upload **directly with the Glacier API** are the ones that live in vaults and are retrieved via the Glacier API. And remember the default retrieval is **Standard (3–5 hours)** — plan for the wait.

8.7 AWS Storage Gateway

AWS Storage Gateway is a hybrid storage service that lets on-premises applications seamlessly use AWS cloud storage — for backup and archiving, disaster recovery, cloud data processing, storage tiering, and migration. Applications connect to a **virtual-machine or hardware gateway appliance** using standard protocols (**NFS, SMB, iSCSI**), and the gateway connects to AWS services such as **Amazon S3, S3 Glacier, and EBS**. It includes bandwidth management, automated network resilience, efficient data transfer, and a **local cache** for low-latency access to the most active data. Storage Gateway comes in **three interface types**.

TABLE 8.6 The three Storage Gateway interfaces

GATEWAY	PROTOCOL	HOW IT STORES DATA IN AWS
File gateway	NFS, SMB	Stores and retrieves files as native Amazon S3 objects, reached through an NFS mount point — with versioning, lifecycle to Glacier, and cross-Region replication
Volume gateway	iSCSI	Presents block volumes backed up as point-in-time EBS snapshots; runs in cached or stored mode
Tape gateway	iSCSI	Presents a virtual tape library to existing backup software; tapes are stored in S3 and archived to S3 Glacier

The **volume gateway** runs in two modes. In **cached mode**, primary data lives in **Amazon S3** while frequently accessed data is retained **locally** — cutting on-premises storage cost while keeping low-latency access. In **stored mode**, the **entire dataset is stored locally** and asynchronously backed up to S3, providing durable, inexpensive offsite backups you can recover locally or from EC2.

8.8 AWS data transfer and migration services

The module closes with three services for moving data into (and out of) AWS. They differ by scale and mechanism: a managed protocol endpoint, an automated network transfer agent, and a physical shipping appliance for the very largest datasets.

TABLE 8.7 AWS data transfer and migration services

SERVICE	WHAT IT DOES	TRANSPORT
AWS Transfer for SFTP	A fully managed, highly available SFTP server that sends files directly into and out of Amazon S3	SFTP (SSH File Transfer Protocol) over the network
AWS DataSync	Simplifies, automates, and accelerates moving and replicating data between on-premises storage and S3 or EFS	An on-premises agent over the internet or AWS Direct Connect, encrypted with TLS
AWS Snowball	A petabyte-scale transport for moving very large datasets into or out of AWS	Secure physical appliances (50/80 TB) shipped to and from your site

AWS Transfer for SFTP creates a managed SFTP endpoint that integrates with your existing **Active Directory** or other identity system and with services like **Route 53**, so you migrate SFTP workflows without changing authentication systems, domains, or hostnames — and external partners keep using their existing clients. **AWS DataSync** is fully managed and automatically handles **encryption, network optimization, and data-integrity validation** with a purpose-built, multi-threaded protocol; to use it you deploy the **DataSync agent** on-premises, connect it to your storage over **NFS**, and select S3 or EFS as the destination (encrypted with TLS in transit). **AWS Snowball** answers large-scale challenges — high network costs, long transfer times, security concerns — at as little as **one-fifth** the price of high-speed internet: AWS ships a secure appliance you load and return, with an **E Ink** shipping label, job tracking through **Amazon SNS**, and protection from tamper-resistant enclosures, **256-bit encryption**, and a **Trusted Platform Module (TPM)** for full chain-of-custody.

Module summary

- AWS storage falls into four categories — block (Amazon EBS), object (Amazon S3 and S3 Glacier), file (Amazon EFS and Amazon FSx), and hybrid (AWS Storage Gateway); pick the category that matches the workload.
- Amazon EBS gives one EC2 instance network-attached, persistent block storage in the same AZ; volume types (io1, gp2, st1, sc1) trade IOPS against throughput and cost, with gp2 the default and io1 the highest performance.
- EBS snapshots back a volume up to Amazon S3 — the first is full, later ones are incremental; they are Region-scoped, encrypted when copied, restored to a new volume with create-volume (warm it to avoid the first-access penalty), and automated with Amazon DLM lifecycle policies.
- Instance store is temporary, physically attached block storage that is reclaimed on termination — instance-store-backed instances can only be terminated, never stopped — so use it only for buffers, cache, and scratch data.
- Amazon EFS is a fully managed, elastic NFS v4 file system for Linux, shareable by many instances across AZs, set up through mount targets, and tuned with three performance attributes (performance mode is fixed at creation).
- Amazon S3 is regional object storage in globally unique buckets, designed for eleven 9s durability and 99.99% availability, unlimited data at up to 5 TB per object, private by default; versioning (enable-only-then-suspend) and Object Lock (WORM) protect objects.
- S3 storage classes plus object lifecycle rules (transition and expiration) control cost; Intelligent-Tiering auto-moves objects after 30 idle days, Block Public Access overrides permissive policies, and event notifications fire to SQS, SNS, or Lambda.

- Amazon S3 Glacier archives data cheaply (eleven 9s durable) with retrieval by request — Expedited (1–5 min), Standard (3–5 hr, default), or Bulk (5–12 hr); AWS Storage Gateway (file/volume/tape) bridges on-premises to AWS, and Transfer for SFTP, DataSync, and Snowball migrate data into the cloud.

Review questions

1. A workload needs a persistent disk for a single database server on one EC2 instance, and a separate workload needs a file system shared by a fleet of Linux instances. Which storage service fits each?

Answer: Amazon EBS for the single-instance database disk (persistent, network-attached block storage in the instance's AZ); Amazon EFS for the shared Linux file system (a managed NFS file system many instances can mount at once).

2. Explain why only the first EBS snapshot is a full copy, and where snapshots are stored.

Answer: Snapshots are incremental: the first captures the entire disk, and each later snapshot copies only the blocks that changed since the previous one, referencing the unchanged blocks from earlier snapshots. All are stored in Amazon S3 infrastructure.

3. After you enable versioning on an S3 bucket, what happens on (a) re-uploading the same key and (b) deleting an object — and can you turn versioning off?

Answer: (a) S3 keeps the old object and creates a new one with a new version ID; both are retrievable. (b) S3 adds a delete marker but the prior version is still retrievable by version ID. Versioning can never be disabled again — only suspended.

4. A customer wants long-lived data stored as cheaply as possible but with unpredictable access, and separately wants archives kept for years. Which S3 storage class and which service, and what's the retrieval catch on the archive?

Answer: S3 Intelligent-Tiering for the unpredictable-access data (it auto-tiers, demoting after 30 idle days); Amazon S3 Glacier for the multi-year archive. Glacier retrieval is by request and asynchronous — Standard takes 3–5 hours by default, Expedited 1–5 minutes, Bulk 5–12 hours.

5. A regulated business must guarantee that certain objects cannot be deleted by anyone — including administrators — for seven years. What S3 feature and setting deliver that, and when must it be turned on?

Answer: S3 Object Lock in Compliance mode with a seven-year retention period — in Compliance mode no user, not even the account root user, can overwrite or delete the object, and the period can't be shortened. Object Lock must be enabled when the bucket is created.

6. Name the three AWS data transfer and migration services and the situation each is built for.

Answer: AWS Transfer for SFTP — a managed SFTP endpoint to move files into and out of S3 while keeping existing identity/hostnames; AWS DataSync — automated, accelerated network transfer between on-premises storage and S3 or EFS; AWS Snowball — physical appliances for petabyte-scale transfers where the network would be too slow or costly.