Module 5 – Storage and Databases

Contents

Learning objectives

In this module, you will learn how to:

- Summarize the basic concept of storage and databases.
- Describe the benefits of Amazon Elastic Block Store (Amazon EBS).
- Describe the benefits of Amazon Simple Storage Service (Amazon S3).
- Describe the benefits of Amazon Elastic File System (Amazon EFS).
- Summarize various storage solutions.
- Describe the benefits of Amazon Relational Database Service (Amazon RDS).
- Describe the benefits of Amazon DynamoDB.
- Summarize various database services.

With what we have learnt earlier, we can now have an **elastic, scalable, disaster resistant, cost optimized architecture that now has a global, highly secured network that can be deployed entirely programmatically.** Now let's talk about storage and databases.

Instance stores

When you're using Amazon EC2 to run your business applications, those applications need access to CPU, memory, network, and storage. EC2 instances give you access to all those different components, and right now, let's focus on the **storage access**. As applications run, they will oftentimes need access to **block-level storage**.

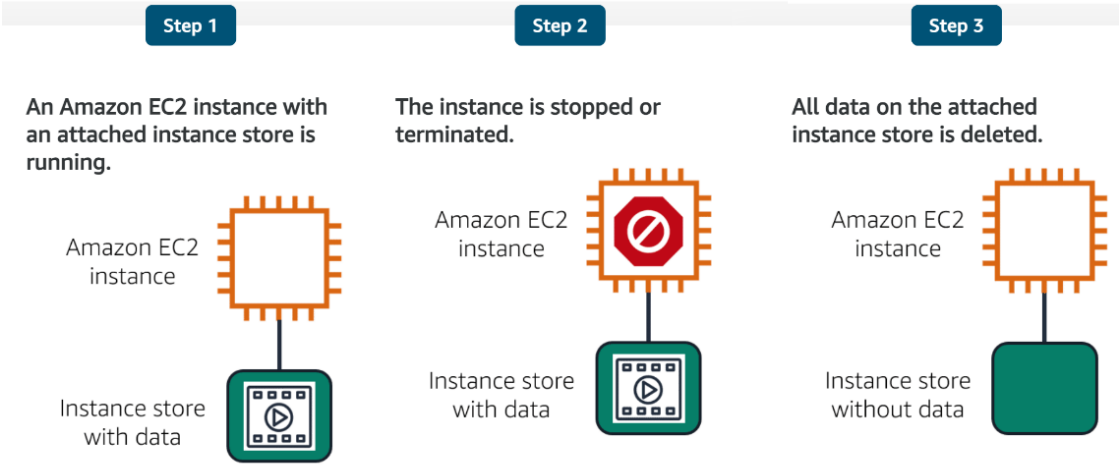
You can think of **block-level storage as a place to store files**. A file being a series of bytes that are stored in blocks on disc. When a file is updated, the whole series of blocks aren't all overwritten. Instead, it updates just the pieces that change. This makes it an efficient storage type when working with applications like databases, enterprise software, or file systems.

When you use your laptop or personal computer, you are accessing block-level storage. All block-level storage is in this case is your hard drive. EC2 **block-level storage volumes** behave like physical hard drives.

An **instance store** provides **temporary block-level storage** for an Amazon **EC2** instance. An instance store is disk storage that is **physically attached** to the host computer for an EC2 instance, and therefore has the same lifespan as the instance. The catch here is that since this volume is attached to the underlying physical host, if you **stop or terminate** your EC2 instance, all data written to the instance store volume will be deleted. This means that when the instance is terminated, you lose any data in the instance store.

The reason for this, is that if you **start your instance from a stop state**, it's likely that EC2 instance will **start up on another host**. A host where that volume does not exist. Remember **EC2 instances are virtual machines**, and therefore the **underlying host can change** between

stopping and starting an instance.



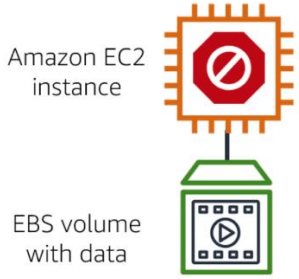
Amazon EC2 instances are virtual servers. If you start an instance from a stopped state, the instance might **start on another host**, where the previously used instance store volume does not exist. Therefore, AWS recommends instance stores for use cases that involve temporary data that you do not need in the long term.

Because of this **ephemeral or temporary nature of instance store volumes**, they are useful in situations where you can lose the data being written to the drive. Such as temporary files, scratch data, and data that can be easily recreated without consequence.

Amazon Elastic Block Store

Amazon Elastic Block Store (Amazon EBS) is a service that provides block-level storage volumes that you can use with Amazon EC2 instances. If you stop or terminate an Amazon EC2 instance, **all the data on the attached EBS volume remains available**.

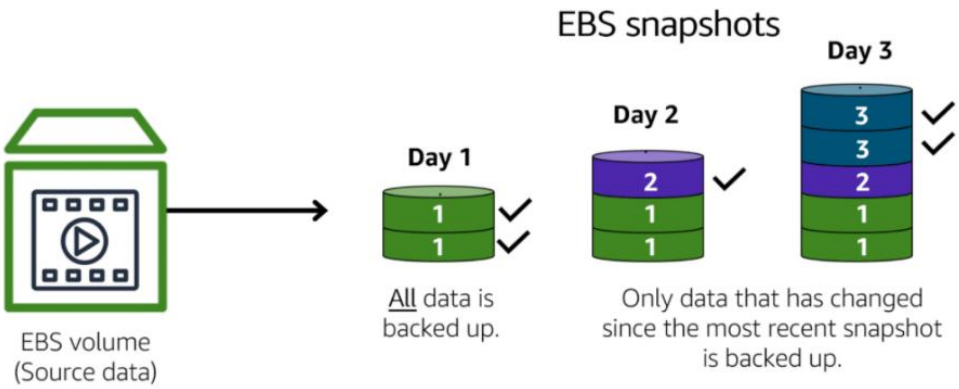
With EBS, you can create virtual hard drives that we call EBS volumes that you can attach to your EC2 instances. These are separate drives from the local instance store volumes, and they aren't tied directly to the host that your EC2 instance is running on. This means that the data that you write to an EBS volume **persists between stops and starts of an EC2 instance** (i.e. beyond the lifecycle of an EC2 instance).



To create an EBS volume, you define the configuration (such as volume size and type) and provision it. After you create an EBS volume, it can **attach** to an Amazon EC2 instance.

Because EBS volumes are for data that needs to persist, it's important to back up the data. You can take incremental **backups** of EBS volumes by creating Amazon **EBS snapshots**. It's very important that you take regular snapshots of your EBS volumes. This way, if a drive ever becomes corrupted, you haven't lost your data. And you can restore that data from a snapshot.

Amazon EBS snapshots



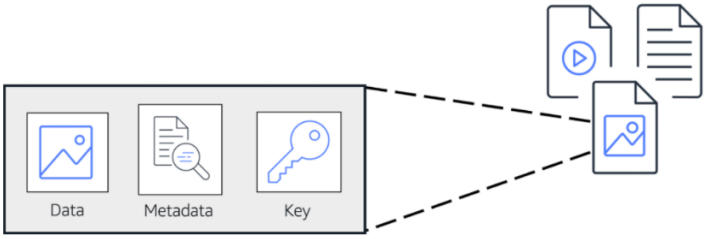
An **EBS snapshot** is an **incremental** backup. This means that the first backup taken of a volume copies all the data. For subsequent backups, **only the blocks of data that have changed** since the most recent snapshot are saved.

Incremental backups are different from full backups, in which all the data in a storage volume copies each time a backup occurs. The **full backup includes data that has not changed since the most recent backup.**

Object storage

Most businesses have data that needs to be stored somewhere. For the coffee shop, this could be receipts, images, Excel spreadsheets, employee training videos, and even text files, among others. Storing these files is where S3 comes in handy because it is a data store that allows you to **store and retrieve an unlimited amount of data at any scale**.

Data is stored as objects, but instead of storing them in a file directory, you store them in what we call **buckets**. Think of a file sitting on your hard drive, which is an object and think of a **file directory, which is the bucket**.



In **object storage**, each object consists of **data, metadata, and a key**.

The data might be an image, video, text document, or any other type of file. Metadata contains information about what the data is, how it is used, the object size, and so on. An object's **key is its unique identifier**.

Amazon Simple Storage Service (Amazon S3)

Amazon Simple Storage Service (Amazon S3) is a service that provides object-level storage. Amazon S3 stores data as objects in buckets.

You can upload any type of file to Amazon S3, such as images, videos, text files, and so on. For example, you might use Amazon S3 to store backup files, media files for a website, or archived documents. Amazon S3 offers unlimited storage space. The **maximum file size for an object in Amazon S3 is 5 TB.**

When you upload a file to Amazon S3, you can set permissions to control visibility and access to it. You can also use the Amazon S3 **versioning** feature to track changes to your objects over time. What this means is that you always retain the previous versions of an object, as like a paper trail.

You can even create multiple buckets and store them across different classes or tiers of data. You can then create permissions to limit who can see or even access objects.

Amazon S3 storage classes

With Amazon S3, you pay only for what you use. You can choose from a range of storage classes to select a fit for your business and cost needs. When selecting an Amazon S3 storage class, consider these two factors:

- How often you plan to retrieve your data
- How available you need your data to be

S3 Standard

- Designed for frequently accessed data
- Stores data in a **minimum of three Availability Zones**

S3 Standard provides **high availability** for objects. This makes it a good choice for a wide range of use cases, such as websites, content distribution, and data analytics. S3 Standard has a higher cost than other storage classes intended for infrequently accessed data and archival storage.

It comes with **11 nines of durability**. That means an object stored in S3 Standard has a 99.999999999 percentage probability that it will remain intact after a period of one year. That's pretty high.

Furthermore, data is stored in such a way that AWS can sustain the concurrent loss of data in two separate storage facilities. This is because data is stored in **at least three facilities**, so multiple copies reside across locations.

Another useful way to use S3 is **static website hosting**, where a static website is a collection of HTML files and each file is akin to a physical page of the actual site. You can do this by simply uploading all your HTML, static web assets, and so forth into a bucket and then checking a box to host it as a static website. You can then enter the bucket's URL and ba-bam! Instant website. And we say static, but that doesn't mean you can't have animations and moving parts to your website.

S3 Standard-Infrequent Access (S3 Standard-IA)

- Ideal for infrequently accessed data
- Similar to S3 Standard but has a lower storage price and higher retrieval price

S3 Standard-IA is ideal for data infrequently accessed but requires high availability (i.e. rapid access) when needed. It's a perfect place to store **backups**, **disaster recovery files**, or any object that requires a long-term storage.

Both S3 Standard and S3 Standard-IA store data in a **minimum of three Availability Zones**. S3 Standard-IA provides the same level of availability as S3 Standard but with a **lower storage price and a higher retrieval price.**

S3 One Zone-Infrequent Access (S3 One Zone-IA)

- Stores data in a **single Availability Zone**
- Has a lower storage price than S3 Standard-IA

Compared to S3 Standard and S3 Standard-IA, which store data in a minimum of three Availability Zones, S3 One Zone-IA stores data in a **single Availability Zone**. This makes it a good storage class to consider if the following conditions apply:

- You want to save costs on storage.
- You **can easily reproduce your data** in the event of an Availability Zone failure.

S3 Intelligent-Tiering

- Ideal for data with unknown or changing access patterns
- Requires a small monthly monitoring and automation fee per object

In the S3 Intelligent-Tiering storage class, Amazon S3 monitors objects' access patterns. If you haven't accessed an object for **30 consecutive days**, Amazon S3 automatically moves it to the infrequent access tier, **S3 Standard-IA**. If you access an object in the infrequent access tier, Amazon S3 automatically moves it to the frequent access tier, **S3 Standard**.

S3 Glacier

- Low-cost storage designed for data archiving
- Able to retrieve objects within a **few minutes to hours**

Say, we need to **retain data for several years for auditing** purposes. And we don't need it to be retrieved very rapidly. Well, then you can use Amazon S3 Glacier to archive that data.

S3 Glacier is a low-cost storage class that is ideal for data archiving. For example, you might use this storage class to store archived customer records or older photos and video files.

To use Glacier, you can simply move data to it, or you can create vaults and then populate them with archives. And if you have **compliance requirements** around retaining data for, say, a certain period of time, you can employ an S3 Glacier **vault lock policy** and lock your vault. You can specify controls such as **write once/ read many**, or WORM, in a vault lock policy and **lock the policy from future edits**. Once locked, the policy can no longer be changed. You also have three options for retrieval, which range from minutes to hours, and you have the option of uploading directly to Glacier or using S3 Lifecycle policies.

Lifecycle policies are policies you can create that can **move data automatically between tiers.** For example, say we need to keep an object in S3 Standard for 90 days, and then we want to move it to S3-IA for the next 30 days. Then after 120 days total, we want it to be moved to S3 Glacier. With Lifecycle policies, you create that configuration without changing your application code and it will perform those moves for you automatically.

S3 Glacier Deep Archive

- Lowest-cost object storage class ideal for archiving
- Able to retrieve objects within 12 hours

When deciding between Amazon S3 Glacier and Amazon S3 Glacier Deep Archive, consider how quickly you need to retrieve archived objects. You can retrieve objects stored in the S3 Glacier storage class within a few minutes to a few hours. By comparison, you can retrieve objects stored in the S3 Glacier Deep Archive storage class **within 12 hours**.

Comparing Amazon EBS and Amazon S3

Let's say you're running a photo analysis website where users upload a photo of themselves, and your application finds the animals that look just like them. You have potentially millions of animal pictures that all need to be indexed and possibly viewed by thousands of people at once. This is the perfect use case for S3. **S3 is already web enabled**. **Every object already has a URL** that you can control access rights to who can see or manage the image.

S3 is **regionally distributed**, which means that it has 11 nines of durability, so no need to worry about backup strategies. S3 is your **backup strategy**. Plus the cost savings is substantial overrunning the same storage load on EBS. **With the additional advantage of being serverless, no Amazon EC2 instances are needed**. Sounds like S3 is the judge's winner here for this round.

But wait, round two, you have an 80-gigabyte video file that you're making **edit corrections** on. To know the best storage class here, we need to understand the difference between object storage and block storage. **Object storage treats any file as a complete, discreet object.** Now this is great for documents, and images, and video files that get uploaded and consumed as entire objects, but every time there's a change to the object, you must **re-upload the entire file**. There are **no delta updates**.

Block storage breaks those files down to small component parts or blocks. This means, for that 80-gigabyte file, when you make an edit to one scene in the film and save that change, the engine only updates the blocks where those bits live. If you're making a bunch of micro edits, using **EBS, elastic block storage**, is the perfect use case. If you were using S3, every time you saved the changes, the system would have to upload all 80 gigabytes, the whole thing, every time. EBS clearly wins round two.

This means, if you are using **complete objects or only occasional changes, S3 is victorious**. If you are doing **complex read, write, change functions, then, absolutely, EBS is your knockout winner**. Your winner depends on your individual workload. Each service is the right service for specific needs. Once you understand what you need, you will know which service is your champion!

File storage

In **file storage**, multiple clients (such as users, applications, servers, and so on) can **access data that is stored in shared file folders**. In this approach, a storage server uses block storage with a local file system to organize files. Clients access data through file paths.

It's extremely common for businesses to have shared file systems across their applications. For example, you might have multiple servers running analytics on large amounts of data being stored in a shared file system.

Compared to block storage and object storage, **file storage is ideal for use cases in which a large number of services and resources need to access the same data at the same time.**

Amazon Elastic File System (Amazon EFS) is a scalable file system used with AWS Cloud services and on-premises resources. As you add and remove files, Amazon EFS grows and shrinks automatically. It can scale on demand to petabytes without disrupting applications.

EFS allows you to have multiple instances accessing the data in EFS at the same time. It scales up and down as needed without you needing to do anything to make that scaling happen.

Comparing Amazon EBS and Amazon EFS

Amazon EBS volumes attach to EC2 instances and are an Availability Zone-level resource i.e. an Amazon **EBS volume stores data in a single Availability Zone**. To attach an Amazon EC2 instance to an EBS volume, both the Amazon EC2 instance and the **EBS volume must reside within the same Availability Zone**. You can save files on it. You can also run a database on top of it. Or store applications on it. It's a hard drive. If you provision a two terabyte EBS volume and fill it up, it doesn't automatically scale to give you more storage. So that's EBS.

Amazon EFS is a **regional** service. It stores data in and across **multiple** Availability Zones. The **duplicate storage** enables you to **access data concurrently from all the Availability Zones** in the Region where a file system is located, meaning that Amazon EFS can have multiple instances reading and writing from it at the same time. Additionally, **on-premises servers can access Amazon EFS using AWS Direct Connect.**

Relational databases

In a **relational database**, data is stored in a way that relates it to other pieces of data.

An example of a relational database might be the coffee shop's inventory management system. Each record in the database would include data for a single item, such as product name, size, price, and so on.

Relational databases use **structured query language (SQL)** to store and query data. This approach allows data to be stored in an easily understandable, consistent, and scalable way. For example, the coffee shop owners can write a SQL query to identify all the customers whose most frequently purchased drink is a medium latte.

Example of data in a relational database:

ID	Product name	Size	Price
1	Medium roast ground coffee	12 oz.	$5.30
2	Dark roast ground coffee	20 oz.	$9.27

Amazon Relational Database Service

Amazon Relational Database Service (Amazon RDS) is a service that enables you to **run relational databases** in the AWS Cloud.

Amazon RDS is a **managed service** that automates tasks such as hardware provisioning, database setup, patching, and backups. With these capabilities, you can spend less time completing administrative tasks and more time using data to innovate your applications. You can integrate Amazon RDS with other services to fulfil your business and operational needs, such as using **AWS Lambda** to **query your database** from a serverless application.

Amazon RDS provides a number of different security options. Many Amazon RDS database engines offer encryption at rest (protecting data while it is stored) and encryption in transit (protecting data while it is being sent and received).

Amazon RDS database engines

Amazon RDS is available on six database engines, which optimize for memory, performance, or input/output (I/O). Supported database engines include:

- Amazon Aurora
- PostgreSQL
- MySQL
- MariaDB
- Oracle Database
- Microsoft SQL Server

The benefits of Amazon RDS include **automated patching, backups, redundancy, failover, disaster recovery**, all of which you normally have to manage for yourself. This makes it an extremely attractive option to AWS customers, as it allows you to focus on business problems and not maintaining databases.

Amazon Aurora

Amazon Aurora is an **enterprise-class relational database**. It is compatible with MySQL and PostgreSQL relational databases. It is up to **five times faster than standard MySQL** databases and up to three times faster than standard PostgreSQL databases.

Amazon Aurora is AWS' **most managed relational database option**. It comes in two forms, **MySQL** and **PostgreSQL**. And is priced is 1/10th the cost of commercial grade databases. That's a pretty cost effective database.

Amazon Aurora helps to reduce your database costs by reducing unnecessary input/output (I/O) operations, while ensuring that your database resources remain reliable and available.

Consider **Amazon Aurora** if your workloads require high availability. It replicates **six copies of your data across three Availability Zones and continuously backs up your data to Amazon S3.** You therefore also get point in time recovery, so you can recover data from a specific period.

You can also deploy up to 15 read replicas, so you can offload your reads and scale performance.

Nonrelational databases

In a **nonrelational database**, you create tables. A table is a place where you can store and query data.

Relational databases works great for a lot of use cases, and has been the standard type of database historically. However, these types of rigid SQL databases, can **have performance and scaling issues when under stress**. The rigid schema also makes it so that you cannot have any variation in the types of data that you store in a table. So, it might not be the best fit for a dataset that is a little bit less rigid, and is being accessed at a very high rate. This is where non-relational, or NoSQL, databases come in.

Nonrelational databases are sometimes referred to as "NoSQL databases" because they use structures other than rows and columns to organize data. Non-relational databases tend to have simple flexible schemas, not complex rigid schemas, laying out multiple tables that all relate to each other.

One type of structural approach for nonrelational databases is key-value pairs. With key-value pairs, data is organized into items (keys), and items have attributes (values). You can think of attributes as being different features of your data.

In a key-value database, you can add or remove attributes from items in the table at any time. Additionally, not every item in the table has to have the same attributes.

Example of data in a nonrelational database:

Key	Value
1	**Name**: John Doe **Address**: 123 Any Street **Favourite drink**: Medium latte
2	**Name**: Mary Major **Address**: 100 Main Street **Birthday**: July 5, 1994

Amazon DynamoDB

Amazon DynamoDB is a key-value database service. It delivers single-digit millisecond performance at any scale. It is a **non-relational, NoSQL database**.

With DynamoDB, you create tables. A DynamoDB table, is just a place where you can store and query data. Data is organized into items, and items have attributes. Attributes are just different features of your data. If you have one item in your table, or 2 million items in your table, DynamoDB manages the underlying storage for you.

DynamoDB is **serverless**, which means that you **do not have to provision, patch, or manage servers. You also do not have to install, maintain, or operate software**.

DynamoDB, **beyond being massively scalable, is also highly performant**. DynamoDB has a millisecond response time. And when you have applications with potentially millions of users, having scalability and reliable lightning fast response times is important. As the size of your database shrinks or grows, DynamoDB **automatically scales** to adjust for changes in capacity while maintaining consistent performance. This makes it a suitable choice for use cases that require high performance while scaling.

DynamoDB stores this data redundantly **across availability zones** and mirrors the data across multiple drives under the hood for you. This makes the burden of operating a highly available database, much lower.

Comparing Amazon RDS and Amazon DynamoDB

Relational databases have been around since the moment businesses started using computers. Being able to build complex analysis of data spread across multiple tables, is the strength of any relational system. In this round, you have a **sales supply chain management system** that you have to analyse for weak spots. Using RDS is the clear winner here because it's built for business analytics, because you **need complex relational joins**. Round one easily goes to RDS.

Round two, the use case, pretty much anything else. Now that sounds weird, but despite what your standalone legacy database vendor would have you believe**, most of what people use expensive relational databases for, has nothing to do with complex relationships**. In fact, **a lot of what people put into these databases ends up just being look-up tables.**

For this round, imagine you have an employee contact list: names, phone numbers, emails, employee IDs. Well, this is all single table territory. I could use a relational database for this, but the things that make relational databases great, **all of that complex functionality, creates overhead and lag and expense if you're not actually using it**.

This is where non-relational databases, Dynamo DB, delivers the knockout punch. By eliminating all the overhead, **DynamoDB allows you to build powerful, incredibly fast databases where you don't need complex joint functionality**. DynamoDB comes out the undisputed champion.

Once again, the winner depends on your individual workload. Each service is the right service for specific needs. And once you understand what you need, you will know again, which service is your champion.

Amazon Redshift

We just spent a lot of time discussing the kinds of workflow that require fast, reliable, current data. Databases that can handle 1,000s of transactions per second, storage that is highly available and massively durable. But sometimes, we have a business need that goes outside what is happening right now to what did happen. This data analysis is the realm of a whole different class of databases. Sure, you could use the one size fits all model of a single database for everything, but modern databases that are engineered for high speed, real time ingestion, and queries may not be the best fit.

Let me explain. In order to handle the **velocity** of real time read/write functionality, most relational databases tend to function fabulously at certain capacities. How much content it actually stores. The problem with historical analytics, data that answers questions like, "Show me how production has improved since we started", is the data collection never stops. In fact, with modern telemetry and the explosion of IoT, the **volume** of data will overwhelm even the beefiest traditional relational database.

It gets worse. Not just the volume, but the **variety** of data can be a problem. You want to run business intelligence or BI projects against data coming from different data stores like your inventory, your financial, and your retail sales systems? **A single query against multiple databases sounds nice, but traditional databases don't handle them easily.**

Once data becomes too complex to handle with traditional relational databases, you've entered the world of data warehouses. Data warehouses are engineered specifically for this kind of **big data**, where you are looking at historical analytics as opposed to operational analysis.

Now, let's be clear. Historical may be as soon as: show me **last hour's** sales numbers across all the stores. The key thing is, the data is now set. We're not selling any more from the last hour because that is now in the past. Compare that question to, "How many bags of coffee do we still have in our inventory right now?" Which could be changing as we speak. **As long as your business question is looking backwards at all, then a data warehouse is the right solution for that line of business intelligence.**

A data warehouse is a system that pulls together data from many different sources within an organization for reporting and analysis. The reports created from complex queries

within a data warehouse are used to make business decisions. A data warehouse stores **historical data** about your business so that you can analyse and extract insights from it. It does **not** store current information, nor is it updated in real-time.

Now there are many data warehouse solutions out on the market. If you already have a favourite one, running it on AWS is just a matter of getting the data transferred. But beyond that, there may still be a lot of undifferentiated heavy lifting that goes into **keeping a data warehouse tuned, resilient, and continuously scaling**. Wouldn't it be nice if your data warehouse team could focus on the data instead of the unavoidable care and feeding of the engine?

Introducing Amazon Redshift. **Amazon Redshift** is a **data warehousing service** that you can use **for big data analytics**. It offers the ability to collect data from many sources and helps you to understand relationships and trends across your data. This is **data warehousing as a service**. It's massively scalable. Redshift nodes in multiple petabyte sizes is very common. In fact, in cooperation with Amazon Redshift Spectrum, you can **directly run a single SQL query against exabytes of unstructured data running in data lakes.**

But it's more than just being able to handle massively larger data sets. Redshift uses a variety of innovations that allow you to achieve up to 10 times higher performance than traditional databases, when it comes to these kinds of business intelligence workloads.

We have whole classes that you or your data teams can take that explain how it is built, and why it can return such improved results. The key for you is to understand that when you need **big data BI solutions**, Redshift allows you to get started with a single API call. Less time waiting for results, more time getting answers.

AWS Database Migration Service (AWS DMS)

AWS Database Migration Service (AWS DMS) enables you to migrate **relational** databases, **nonrelational** databases, and other types of data stores.

DMS helps customers migrate existing databases onto AWS in a secure and easy fashion. With AWS DMS, you move data between a source database and a target database. The source and target databases can be of the same type (**homogenous** migration) or different types (**heterogeneous** migration). During the migration, your **source database remains operational**, reducing downtime for any applications that rely on the database.

For heterogeneous migrations, it's a **two-step** process. Since the schema structures, data types, and database code are different between source and target, we first need to convert them using the **AWS Schema Conversion Tool**. This will convert the source schema and code to match that of the target database. The next step is then to use DMS to migrate data from the source database to the target database.

For example, suppose that you have a MySQL database that is stored on premises in an Amazon EC2 instance or in Amazon RDS. Consider the MySQL database to be your source database. Using AWS DMS, you could migrate your data to a target database, such as an Amazon Aurora database.

Other use cases for AWS DMS

Development and test database migrations: Enabling developers to test applications against production data without affecting production users. In this case, you use DMS to migrate a copy of your production database to your **dev or test environments**, either once-off or continuously.

Database consolidation: Combining several databases into a **single central** database

Continuous replication: Sending ongoing copies of your data to other target sources instead of doing a one-time migration. This could be for disaster recovery or because of geographic separation.

Additional database services

Amazon DocumentDB is a document database service that supports **MongoDB** workloads. (MongoDB is a document database program). This is great for content management, catalogs, and user profiles.

Amazon Neptune is a **graph database** service. You can use Amazon Neptune to build and run applications that work with highly connected datasets, such as social networking and recommendation engines, fraud detection, and knowledge graphs.

Amazon Quantum Ledger Database (Amazon QLDB) is an **immutable ledger database** service. You can use Amazon QLDB to review a complete history of all the changes that have been made to your application data.

Amazon Managed Blockchain is a service that you can use to create and manage **blockchain networks** with open-source frameworks. Blockchain is a distributed ledger system that lets multiple parties run transactions and share data without a central authority.

Amazon ElastiCache is a service that **adds caching layers on top of your databases to help improve the read times of common requests.** It supports two types of data stores: **Redis** and **Memcached**.

Amazon DynamoDB Accelerator (DAX) is an **in-memory cache for DynamoDB**. It helps to dramatically **improve response (read) times** for your nonrelational data from single-digit milliseconds to microseconds.

Summary

You learned about all the different types of AWS storage mechanisms. Let's recap them, shall we?

The first one we learned about is **Elastic Block Store (EBS)** volumes, and you attach those to EC2 instances so you have local storage that is not **ephemeral**.

You learned about how **S3** and how you can store objects in AWS with the click of a button or call of an API.

We even discussed the various relational database options available on AWS, such as Amazon **RDS**. Or for the workloads that just need a key-value pair, we have the non-relational offering called **DynamoDB**.

Next up was **EFS** for file storage use cases. We then have **Amazon Redshift** for all our data warehouse needs. And to aid in migration of existing databases, we have **DMS** or Database Migration Service.

We also touched upon the lesser known storage services, like **DocumentDB**, **Neptune**, **QLDB**, and Amazon **Managed Blockchain**. Lastly, we talked about how caching solutions like **ElastiCache** and DynamoDB Accelerator (**DAX**) can be used.

That's a lot of places to store different types of data, and hopefully you've learned the correct place to store each type.

Quiz

Which Amazon S3 storage classes are optimized for archival data? (Select TWO.)

- S3 Glacier
- S3 Glacier Deep Archive
- Note: Objects stored in the S3 Glacier storage class can be retrieved within a few minutes to a few hours. By comparison, objects that are stored in the S3 Glacier Deep Archive storage class can be retrieved within 12 hours.

Which statement or statements are TRUE about Amazon EBS volumes and Amazon EFS file systems?

- EBS volumes store data within a single Availability Zone. Amazon EFS file systems store data across multiple Availability Zones. An EBS volume must be located in the same Availability Zone as the Amazon EC2 instance to which it is attached. Data in an Amazon EFS file system can be accessed concurrently from all the Availability Zones in the Region where the file system is located.

You want to store data in an object storage service. Which AWS service is best for this type of storage?

- Amazon Simple Storage Service (Amazon S3)

Which statement best describes Amazon DynamoDB?

- A serverless key-value database service. Amazon DynamoDB is a key-value database service. It is serverless, which means that you do not have to provision, patch, or manage servers.

Which service is used to query and analyse data across a data warehouse?

- Amazon Redshift. It is a data warehousing service that you can use for big data analytics. Use Amazon Redshift to collect data from many sources and help you understand relationships and trends across your data.

Additional resources

To learn more about the concepts that were explored in Module 5, review these resources.

- Cloud Storage on AWS
- AWS Storage Blog
- Hands-On Tutorials: Storage
- AWS Customer Stories: Storage
- AWS Database Migration Service
- Databases on AWS
- Category Deep Dive: Databases
- AWS Database Blog
- AWS Customer Stories: Databases