



Storing and Organizing Data

AWS Academy Data Engineering

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| **Slide number 1**

| **Instructor notes**

|

| **Student notes**

Welcome to the Storing and Organizing Data module.



| **Slide number 2**

| **Instructor notes**

|

| **Student notes**

This introduction section describes the content of this module.

Module objectives



This module prepares you to do the following:

- Define storage types that are found in a modern data architecture.
- Distinguish between data storage types.
- Select data storage options that match your storage needs.
- Implement secure storage practices for cloud-based data.

| Slide number 3

| Instructor notes

|

| Student notes

This module defines the storage types that are found in a modern data architecture. You will learn to distinguish between the storage types and select the option that matches the requirements and constraints of your use case. And finally, you will learn about secure storage practices for cloud-based data.

Module overview

Presentation sections

- Storage in the modern data architecture
- Data lake storage
- Data warehouse storage
- Purpose-built databases
- Storage in support of the pipeline
- Securing storage

Lab

- Storing and Analyzing Data by Using Amazon Redshift

Knowledge checks

- Online knowledge check
- Sample exam question



| Slide number 4

| Instructor notes

| Each module has an introduction, content sections, and a wrap-up. The wrap-up for this module contains a sample exam question for you to review with the students.

|

| Student notes

The objectives of this module are presented across six sections.

You will also complete a hands-on lab that uses Amazon Redshift to store and analyze data.

The module wraps up with a sample exam question and an online knowledge check that covers the presented material.

Storage in the modern data architecture

Storing and Organizing Data



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| Slide number 5

| Instructor notes

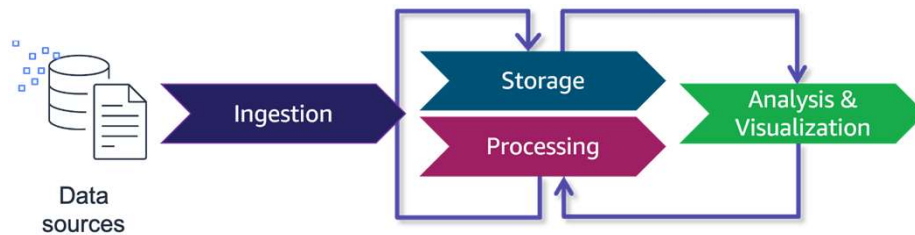
| This section covers the basics of cloud storage, how they apply to the iterative data pipeline, and the part they play in modern data architecture.

|

| Student notes

This section provides an overview of storage concepts and how storage is used to support a modern data architecture.

The simplified iterative data pipeline



| Slide number 6

| Instructor notes

| Review

|

| Student notes

You have learned that, in practice, storing data is directly related to the tasks of ingesting and processing data.

Storage in the iterative data pipeline is far from a linear process. When data is ingested into the pipeline, it is placed in storage. The data can then be removed from storage, processed, and returned to storage for later use or to await additional processing and transformation. This in-and-out cycle of data illustrates that storage is key to your data pipeline. Therefore, your planning efforts should focus on how storage will be used to support your pipeline goals.

In this module, you will learn how to select the appropriate storage for your needs, and how data is stored and organized to support your data analytics and machine learning (ML) pipelines.

Storage in the AWS modern data architecture



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

7

| Slide number 7

| Instructor notes

| Note to students that all the items that are listed on this slide are parts of a modern data architecture. Point out that the circles represent AWS services, while the inner ring of floating words are the elements of the modern data architecture that those services handle.

|

| Student notes

In the Design Principles and Patterns for Data Pipelines module, you learned about the AWS modern data architecture, including the tools and services that you can use to manage and govern your resources. In the diagram on this slide, note that storage is at the center—it's the core of this architecture. In the AWS Cloud, Amazon Simple Storage Service (Amazon S3) provides you with the object storage that is central to your needs. Amazon Redshift provides additional storage as data warehousing. The service stores the structured and semistructured data from relational databases, such as Amazon Aurora, and nonrelational databases, such as Amazon DynamoDB. Note that nonrelational databases are sometimes called NoSQL databases.

The type and volume of data is growing at an unprecedented rate as technology evolves. Organizations want to capture all of this data to derive value from it as quickly as possible.

A modern data architecture will give you the best of both data lakes and purpose-built data stores, providing you with a low-cost storage solution for any amount of data using open standards-based data formats. A modern data architecture isn't restricted by data silos, which enables you to empower individuals or groups to run analytics and ML workloads by using their preferred tools or techniques.

Types of cloud storage

Block storage

- Offers dedicated, low-latency storage
- Is scalable and offers high performance
- Is similar to local direct attached storage or a storage area network (SAN)
- Example: Amazon Elastic Block Storage (Amazon EBS)

File storage

- Stores data as files
- Is highly scalable
- Is ideal for storage such as content repositories and media stores
- Example: Amazon Elastic File System (Amazon EFS)

Object storage

- Stores unstructured, semistructured, or structured data
- Is highly scalable
- Uses a unique identifier for each object
- Has a lower cost than traditional storage
- Example: Amazon Simple Storage Service (Amazon S3)



| Slide number 8

| Instructor notes

|

| Student notes

There are three types of cloud storage, each with their own advantages, disadvantages, and use cases. The three cloud storage types are block storage, file storage, and object storage.

Block storage provides dedicated, low-latency storage for a host. Block storage is dedicated to an instance and behaves like a directly attached storage (DAS) solution, such as a physical hard drive, or a storage area network (SAN).

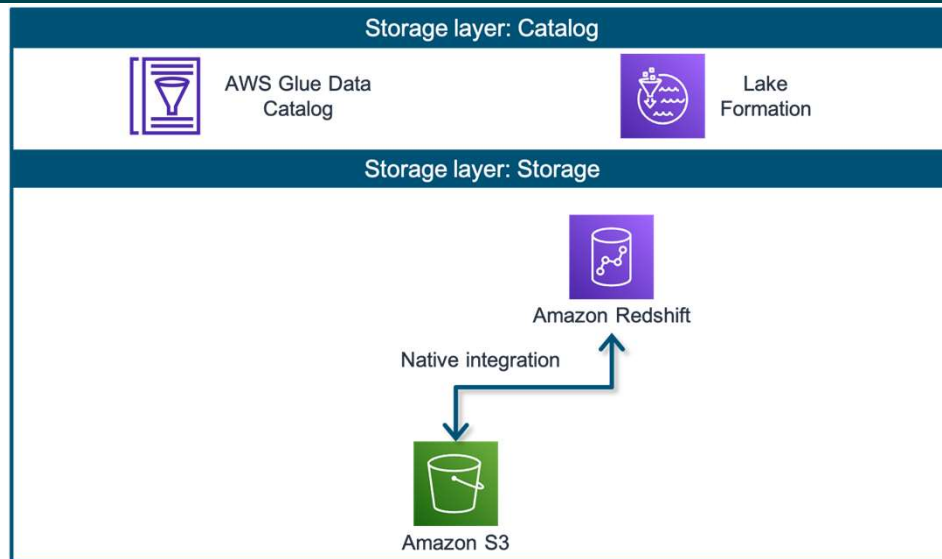
File storage stores data in the form of files. This highly scalable storage format is the ideal solution for data such as content repositories, media stores, and development environments. File storage is often supported with a network attached storage (NAS) server.

Object storage can be used to store unstructured, semistructured, and structured data. The AWS solution for cloud object storage is Amazon S3, which you will learn more about in this module. The service is highly scalable and has near-unlimited storage capability.

Data in cloud object storage is handled as objects. Each object is assigned a key, which is a

unique identifier. When the key is paired with metadata that is attached to the objects, other AWS services can use the information to unlock a multitude of capabilities. Thanks to economies of scale, cloud object storage comes at a lower cost than traditional storage.

Storage layer



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

9

| Slide number 9

| Instructor notes

|

| Student notes

The data storage layer provides durable, scalable, and cost-effective components to store and manage vast quantities of data. In the AWS architecture, Amazon Redshift and Amazon S3 provide unified, natively integrated storage.

The catalog layer in the storage layer stores business and technical metadata about datasets that are hosted in the storage layer. This metadata supports the ability to find and query data that is stored in the data lake and the data warehouse. In the AWS architecture, AWS Lake Formation and AWS Glue work together to collect and store metadata and make it available when needed. The catalog makes it easier for consumers to search for and explore the available data.

Amazon S3 provides you with object storage for structured and unstructured data. With S3 as the foundation of your data lake architecture, you can easily store, access, and query data from a multitude of data sources. You can then use this data immediately or place it in a low-cost, long-term storage solution, such as Amazon S3 Glacier, for later use and analysis. With the scalable, flexible, and durable storage that Amazon S3 provides, you can support the five Vs that were discussed in The Elements of Data module.

Amazon Redshift uses SQL to analyze structured and semistructured data across data warehouses, operational databases, and data lakes. The service uses AWS designed hardware and ML to deliver the best price performance at any scale. Because the service is integrated with AWS services, including database and ML services, Amazon Redshift can help you handle complete analytics workflows.

Comparing data lakes and data warehouses

Characteristic	Data Warehouse	Data Lake
Data	Relational data from transactional systems, operational databases, and line of business applications	Nonrelational and relational data from Internet of Things (IoT) devices, websites, mobile apps, social media, and corporate applications
Schema	Designed prior to the data warehouse implementation (schema on write)	Written at the time of analysis (schema on read)
Price and Performance	Fastest query results using higher cost storage	Query results getting faster using low-cost storage
Data Quality	Highly curated data that serves as the central version of the truth	Any data, which might or might not be curated (for example, raw data)
Users	Business analysts	Data scientists, data developers, and business analysts (using curated data)
Analytics	Batch reporting, business intelligence (BI), and visualizations	ML, predictive analytics, and data discovery and profiling



| Slide number 10

| Instructor notes

| Review this slide with students and discuss how the differences would apply to various scenarios. Relay to students that data lakes and data warehouses are complementary—rather than competing—storage types. Draw students' attention to the data quality characteristic, which distinguishes them from each other.

|

| Student notes

Take a moment to review the differences between data lakes and data warehouses. They are complementary storage types, with data warehouses drawing much of their data from the vast data repositories that are found in data lakes. The key difference lies in the data quality characteristic. A data warehouse stores highly curated data, whereas the data in a data lake can be raw, untransformed, and uncurated.

Data lake storage

Storing and Organizing Data



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| Slide number 11

| Instructor notes

| This section provides an introduction to data lakes. The topic of data storage based on type—structured, unstructured, or semistructured—is discussed. In addition to the concept of data lakes, this section describes the AWS services that support each of them.

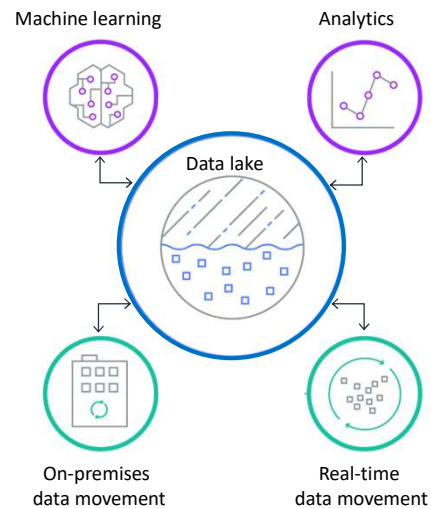
|

| Student notes

This section discusses the storage types that are available for large sets of data, the type of data that they store, and the AWS services that are available for each type.

Data lakes

- Provide a centralized repository
- Store both structured and unstructured data
- Catalog and index data for analysis without data movement
- Store, secure, and protect data at unlimited scale
- Offer in-place transformation and querying of data assets
- Are built using Amazon S3



| Slide number 12

| Instructor notes

|

| Student notes

Data lakes are centralized repositories where you can store structured and unstructured data, regardless of scale. Unlike with a data warehouse, you can store data as-is in a data lake. It isn't necessary to structure the data before you begin to run analytics.

Data lakes can store both relational data, such as data from a line of business application, and nonrelational data, from sources such as mobile apps, Internet of Things (IoT) devices, and social media. You can store data without a particular focus on design or usage plans because the structure of the data or schema is not defined when the data is captured.

Data lakes in AWS provide you with secure, scalable, and durable storage. You can employ AWS services to run your analytics and ML processing, or you can select a preferred third-party analytics platform from the AWS Partner Network (APN). After you migrate your data into an S3 data lake, you can employ purpose-built analytics services, such as Amazon Athena and Amazon EMR; launch artificial intelligence (AI) and ML jobs; and transform and query data in place. Additionally, data lakes in AWS can take advantage of Amazon S3 storage classes, which are purpose built to provide the lowest cost storage for different

access patterns.

Amazon Simple Storage Service (Amazon S3)

- Is secure, scalable, and durable
- Provides a low-cost storage solution
- Stores structured and unstructured data
- Offers in-place transformation and querying
- Uses object storage classes
- Has a strong data consistency model
- Supports multipart upload
- Is the basis of data lake creation



| Slide number 13

| Instructor notes

|

| Student notes

Amazon S3 is the core storage service for AWS. The service provides you with a secure, scalable, and durable low-cost storage solution for a virtually unlimited amount of data. Amazon S3 can store structured, semistructured, and unstructured data alike. With the service, you can read and write data at the object level and use in-place queries.








You can use Amazon S3 storage classes to store your data based on the data access, resiliency, and cost requirements of your workloads. The storage classes are purpose built to provide you with the lowest storage costs based on the access patterns that you need to support.

Amazon S3 simplifies the migration of on-premises analytics workloads by automatically providing strong read-after-write and list consistency for all applications. This capability removes the need for changes to applications and reduces costs by removing the need for additional infrastructure to provide strong consistency. Single key updates are atomic. This means that if you make a PUT request to an existing key in one thread and make a GET request on the same key in a second thread concurrently, you will get either old data or the new data, but never partial or corrupt data. It's important to note that S3 bucket configurations use an *eventual* consistency model. This means that if you delete a bucket and immediately list all of your buckets, the deleted bucket might still appear in your list.

Multipart upload is an Amazon S3 feature that you can use to upload large objects in parts. When you start a multipart upload, Amazon S3 logically divides the object into parts, which can be uploaded independently and in any order. Any transmission failures that you incur during the upload can be corrected without affecting the other parts of the object. After all object parts are uploaded, Amazon S3 assembles the independent parts to create the object. Multipart upload is a three-step process: you initiate the upload, upload all of the object parts, and then complete the multipart upload. You can then access the object in your S3 bucket in the same way as other objects. By default, multipart uploads utilize an MD5 checksum algorithm to verify data integrity, though you specify an additional checksum algorithm if desired. Checksums are performed on the individual object parts rather than the entire object as a single entity.

Amazon S3 storage classes



General purpose	Infrequent access	Archive
 S3 Standard	 S3 Standard-Infrequent Access	 S3 Glacier Instant Retrieval
Unknown or changing access	 S3 One Zone-Infrequent Access	 S3 Glacier Flexible Retrieval
 S3 Intelligent-Tiering		 S3 Glacier Deep Archive



| Slide number 14

| Instructor notes

|

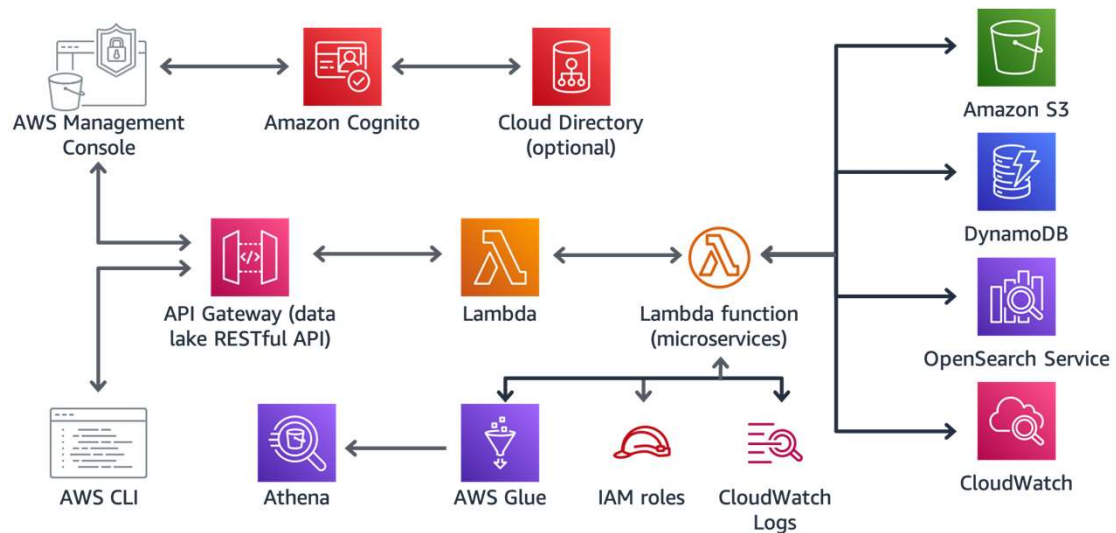
| Student notes

By using Amazon S3 storage classes, you can choose how to store data based on access needs and cost considerations. Analytics and ML workloads require large quantities of data, and you want to ensure that you store that data in the most cost-effective manner as possible.

The storage classes include S3 Intelligent-Tiering for automatic cost savings for data with unknown or changing access patterns and S3 Standard for frequently accessed data. S3 Standard-Infrequent Access (S3 Standard-IA) and S3 One Zone-Infrequent Access (S3 One Zone-IA) are available for less frequently accessed data. S3 Glacier Instant Retrieval is available to archive data that needs immediate access, and S3 Glacier Flexible Retrieval (formerly S3 Glacier) is available for rarely accessed long-term data that doesn't require immediate access. For long-term archive and digital preservation, with retrieval in hours at the lowest cost storage in the cloud, use S3 Glacier Deep Archive.

If you have data residency requirements that can't be met by an existing AWS Region, you can use the S3 Outposts storage class to store your S3 data on premises. Amazon S3 also offers capabilities to manage your data throughout its lifecycle. For example, you can set a lifecycle policy to automatically transfer data to a different storage class without any changes to your application. For more information about Amazon S3 storage classes, see the resources for this module in the Content Resources page of your course.

Example architecture of a data lake



| Slide number 15

| Instructor notes

|

| Student notes

This slide shows an example architecture for a data lake and the services that are common in it. This example uses AWS Lambda microservices (functions), Amazon OpenSearch Service for search capabilities, Amazon Cognito for user authentication, AWS Glue for data transformation, and Amazon Athena for data analysis.

If you would like to explore this data lake architecture more fully, the code to configure this example is available for download on GitHub. A link is available in the Content Resources page of your course.

AWS Lake Formation

- Is a fully managed service
- Provides the ability to build, secure, and manage data lakes
- Automates elements of data lake creation
- Augments the AWS Identity and Access Management (IAM) permissions model
- Supports atomic, consistent, isolated, and durable (ACID) transactions by using governed tables
- Integrates with AWS analytics and ML services



| Slide number 16

| Instructor notes

|

| Student notes

Lake Formation is a fully managed service that simplifies the process to build, secure, and manage data lakes by automating many of the complex manual steps that are required. Lake Formation provides a centrally defined permissions model, which augments your AWS Identity and Access Management (IAM) permissions model. These permissions are enforced using granular controls at the column, row, and cell levels across AWS analytics and ML services.

With Lake Formation, you can identify your existing relational and nonrelational databases or Amazon S3 data stores. You can also import them new. Lake Formation can then crawl and read your data sources to catalog and label the data within, enabling users to find what they need in a more efficient manner. You can use Lake Formation to transform data to ensure consistency for analytics use, and you can use the FindMatches ML transform to clean and deduplicate data. Finally, Lake Formation provides storage optimization for governed tables through capabilities such as data compaction and garbage collection.

Governed tables are metadata tables that are unique to Lake Formation. Governed tables are enabled through new APIs and support the four key properties of transactions—ensuring that they are atomic, consistent, isolated, and durable (ACID). This capability enables multiple users to concurrently insert and delete data across tables. Other users can simultaneously run analytical queries and ML models on those same datasets, all while returning consistent, up-to-date results.

Key takeaways: Data lake storage



- Data lakes store data as-is. You don't need to structure the data before you begin to run analytics.
- Amazon S3 promotes data integrity through strong data consistency and multipart uploads.
- With Lake Formation, you can use governed tables to enable concurrent data inserts and edits across tables.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

17

| Slide number 17

| Instructor notes

|

| Student notes

Here are a few key points to summarize this section.

Data lakes store structured, semistructured, and unstructured data in object storage. You don't need to structure the data before you begin to run analytics.

Amazon S3 promotes data integrity through strong data consistency and multipart uploads. These features ensure that data that is retrieved from storage isn't partial or corrupted and that large files can be uploaded in parts without corruption.

AWS Lake Formation allows you to use governed tables to enable concurrent data inserts and edits across tables. This allows users to update datasets while other users simultaneously run analytical queries and ML models on those same datasets, all while returning consistent, up-to-date results.

Data warehouse storage

Storing and Organizing Data



| Slide number 18

| Instructor notes

| Building on the concentric circles seen in the modern data architecture diagram, this section focuses on the data warehousing aspect. This section includes an overview of Amazon Redshift, which is the AWS solution for data warehousing.

|

| Student notes

The module discusses the data warehouse storage concept with a focus on the Amazon Redshift data warehouse service.

Data warehouses

- Provide a centralized repository
- Store structured and semistructured data
- Store data in one of two ways:
 - Frequently accessed data in fast storage
 - Infrequently accessed data in cheap storage
- Might contain multiple databases that are organized into tables and columns
- Separate analytics processing from transactional databases
- Example: Amazon Redshift



| Slide number 19

| Instructor notes

|

| Student notes

Data warehouses are centralized data repositories. The data that is contained within the data warehouses can be processed and analyzed to make more informed decisions. Structured and semistructured data from sources such as transactional systems and relational databases flows into the data warehouse in a regular cadence.

Data warehouses consist of three tiers. At the top is a front-end client, which presents your results through reporting, analysis, and data mining tools. The middle tier is made up of an analytics engine, which is used to access and analyze the data. The third tier consists of the database server, where your data is loaded and stored. This data is stored in one of two ways: in fast storage, such as SSD drives, if it is frequently accessed, or in low-cost object storage, such as Amazon S3, if it is infrequently accessed. Data can be automatically shifted between the two storage types based on how frequently it is accessed.

A data warehouse can contain multiple databases of structured and semistructured data—data that is organized into tables and columns. When data is ingested into the data warehouse, it can be stored in tables, each of which can be organized inside of schemas. Query tools use these schemas to determine which data tables need to be accessed and analyzed.

Amazon Redshift

- Provides a cloud-based data warehouse solution
- Is a fully managed service
- Supports near real-time data analysis
- Uses columnar storage
- Offers multiple node types for tailored solutions:
 - DC2
 - DS2
 - RA3



| Slide number 20

| Instructor notes

| We recommend that you discuss how Amazon Redshift differs from Amazon S3. Also, point out the differences between the Amazon Redshift node types and usage.

|

| Student notes

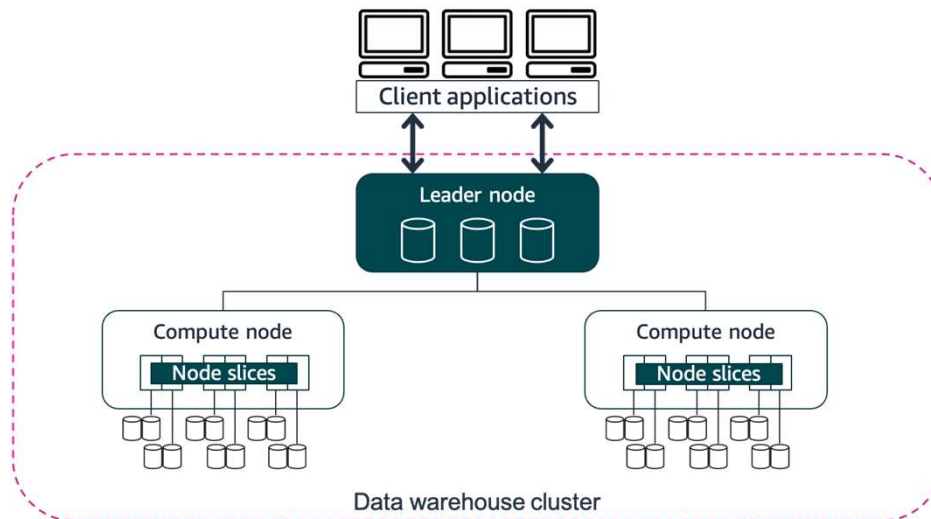
Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the AWS Cloud. The service consists of collections of computing resources called *nodes*. Nodes are organized into *clusters* that run an Amazon Redshift engine and contain one or more column-oriented databases. Each cluster has a leader node and one or more compute nodes. The leader node receives queries from client applications, and then parses the queries and develops query execution plans. The leader node then coordinates the parallel processing of the query plans with the compute nodes and aggregates the intermediate results. Finally, the results are returned to the client applications.

Amazon Redshift has multiple node types for you to choose from, so you can select the best node type to fit your specific requirement. Dense compute, or DC2, nodes are designed for demanding data warehousing workloads that require low latency and high throughput. With DC2 nodes, you can have compute-intensive data warehouses with local SSD storage included. DC2 nodes are often used when the amount of data in your workload is less than 500 GB.

You can use RA3 nodes to optimize your data warehouse by scaling and paying for compute and managed storage independently. By choosing the number of nodes based on performance requirements, you pay only for the managed storage that you use. Therefore, you can size your RA3 cluster based on the amount of data that is processed daily.

By using dense storage, or DS2, nodes, you can create large data warehouses by using low-cost hard disk drives (HDDs). AWS recommends that you upgrade DS2 nodes to RA3 nodes to get more storage and improved performance for the same cost. DS2 nodes are often used when the amount of data in your workload exceeds 500 GB.

Example architecture: Data warehouse in Amazon Redshift



| Slide number 21

| Instructor notes

|

| Student notes

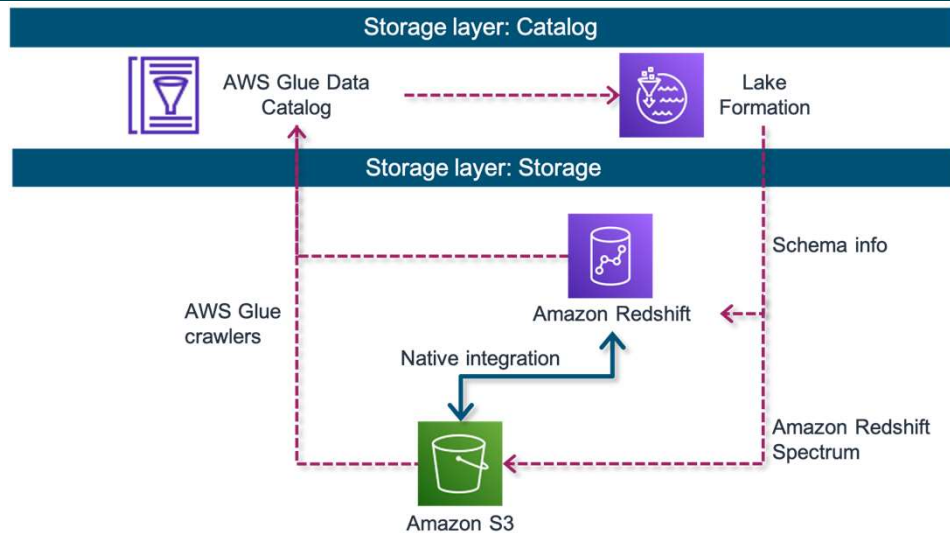
This diagram represents the elements of a data warehouse architecture in Amazon Redshift. The client applications are business intelligence (BI), reporting, data mining, and analytics tools. The data warehouse cluster consists of two compute nodes and a leader node.

Note that if a cluster is provisioned with two or more compute nodes, it will require a leader node to coordinate the compute nodes and handle external communication. When a leader node is present, the client applications only recognize the presence of the leader node and not that of the compute nodes.

As mentioned, the leader node coordinates the compute nodes within the cluster and communicates with the client applications. With the exception of direct queries to reference tables that are stored on the compute nodes, the leader node will run all queries. Certain SQL queries, such as the `CURRENT_SCHEMA` function, can only be run on the leader node. If such queries are run against a compute node, an error will indicate that the function isn't supported.

While the leader node is responsible for compiling code for the elements of the execution plan, the compute node is responsible for running the code. The compute node returns the intermediate results of the query back to the leader node for aggregation. Each compute node has dedicated CPU, memory, and attached disk storage based on node type, and each node is partitioned into *node slices*. These node slices are allocated a portion of the compute node's memory and disk space. The node slices work in parallel to complete operations that the leader node assigned to the computer node.

Amazon Redshift Spectrum



| Slide number 22

| Instructor notes

|

| Student notes

Using Amazon Redshift Spectrum, you can conduct fast, in-place querying against data that is stored in data lakes on Amazon S3. You can make Redshift Spectrum queries on a wide variety of data lake assets, including comma-separated value (CSV), tab-separated value (TSV), Parquet, Sequence, and RCFile formatted data.

With Redshift Spectrum, you can write SQL queries that combine data from the data lake and the data warehouse. When a user makes a query request that includes data from the data lake, Redshift Spectrum gets schema information from the Lake Formation catalog and uses it to query the data lake.

Key takeaways: Data warehouse storage



- Data warehouses consists of three tiers, and can store structured, curated, or transformed data.
- Amazon Redshift is a fully-managed data warehouse services that uses computing resources called *nodes*.
- You can use Redshift Spectrum to write SQL queries that combine data from both your data lake and your data warehouse.

| Slide number 23

| Instructor notes

|

| Student notes

Here are a few key points to summarize this section.

Data warehouses are centralized repositories that consist of three tiers and can store structured data. Data that is ingested into the data warehouse is organized into tables and columns.

Amazon Redshift is a fully-managed data warehouse service provided by AWS. It uses computing resources called *nodes* to provide parallel query processing for rapid results. Amazon Redshift Spectrum is a feature of Amazon Redshift that can be used to conduct fast, in-place querying against data that is stored in data lakes on Amazon S3. SQL queries can be run that combine data from both your data lake and your data warehouse.



| Slide number 24

| Instructor notes

| Now that the core of the modern data architecture, Amazon S3, has been discussed, let's move outward on the architecture diagram. Purpose-built databases build on the relational and nonrelational database concepts.

|

| Student notes

The module discusses the importance of purpose-built databases, how they support data analytics and machine learning, and the considerations necessary for selecting your database solution.

Choosing your purpose-built database

- Choosing the right database is key to supporting your application architecture.
- Your database will affect what your application can handle, how it will perform, and the operation that you are responsible for.
- Consider several factors:
 - Application workload
 - Data shape
 - Performance requirements
 - Operations burden



| Slide number 25

| Instructor notes

|

| Student notes

Choosing the right purpose-built database is key when selecting the application architecture that will support your analytics or ML workload. The database service that you choose will affect the volume and variety of what your application can handle, and determine what type of data is stored and the format it's stored in. Your database selection also determines whether your application can handle the volume and velocity of datasets, as well as corresponding customer demand. Finally, your database selection will determine the operations that you are responsible for.

Consider several factors when you select your purpose-built database.

First, consider the workload of your application. Will your database support ecommerce or content management, or will it strictly be used for analytical purposes?

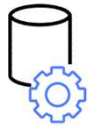
Next, determine the shape of the data that your application will be working with. Consider whether you expect your database to grow considerably over time and how the data is accessed and updated.

After data shape, consider the performance requirements of your application. This is where you will focus on the customer experience in terms of access and application performance.

Finally, you will need to consider the operations burden of your database, focusing on the support you will need to provide to your database throughout the application's lifecycle.

Let's inspect these database selection factors to gain a better understanding of what each entails.

Factors in choosing a purpose-built database



Application workload

- Is your workload transactional?
- Will your workload be used for analytics purposes?
- Does your workload need caching to improve response times?



Data shape

- How will your data be accessed?
- How often will your data be updated?



Performance

- How fast does your data access need to be?
- What is the average size of the records that you are using?
- How will end users use your service?



Operations burden

- How will you prepare for instance failures?
- How will you configure backups?
- What future upgrades might you need?



| Slide number 26

| Instructor notes

|

| Student notes

When choosing a purpose-built database, the first factor to consider is the workload of your application. Workload is about the type of data that your application stores and the data access patterns. There are three broad categories of workloads.

Transactional: A *transactional workload* is characterized by a high number of concurrent operations, where each operation reads or writes a small number of rows. Most user-facing applications, including ecommerce, mobile gaming, and social networking, are transactional. Transactional workloads are also referred to as online transactional processing, or OLTP. Online shopping services are a great example of an OLTP workload.

Analytical: *Analytical workloads* aggregate and summarize large volumes of data. There are usually far fewer concurrent queries in analytical data stores, but they are operating on many more rows per query. Analytical access patterns are usually for internal applications such as reporting. These workloads are also referred to as online analytical processing, or OLAP. Your organization can use OLAP workloads for strategic planning. Imagine your business sells widgets in multiple sizes, shapes, and colors nationwide. An OLAP workload can combine this data set with purchase with data sets that give information about purchase locations, quantities, and more to determine what types of widget should be kept in larger supply based on such factors.

Caching: In a *caching workload*, you compute and store frequently accessed data in a separate database for faster response times. This approach reduces the load on your transactional database and improves response times to your end users. The cache is a secondary source that stores derived data from your transactional workloads rather than acting as a primary source of data.

After considering your application workload, the next factor to consider is the shape of your data. When considering data shape, determine the types of entities that you will model and the relationships between your entities. Ask yourself how you will access your data and how often entities will be updated.

After you determine your data shape, choose the database type that best suits your needs. Here are some of the common data models and use cases for each of them.

- **Relational:** The relational database is a well-known format for many developers. In a relational database, you normalize your data into separate tables and assemble related entities together at query time. In the context of databases, the term *normalization* refers to the process of organizing the columns (attributes) and tables (relations) of a relational database to minimize data redundancy. A relational database is a good choice when you have multiple related entities with varying update patterns. The strict schema validation and normalized data model help you maintain data integrity across your application.
- **Key-value** or **wide-column:** These data models are designed for scale, with your data being split across multiple storage nodes. As your data grows, additional storage nodes can be added to accommodate. This partitioning scheme allows for nearly infinite scalability with no performance degradation.
- **Document:** A document data model uses large records called *documents* to assemble heterogeneous bits of data that are frequently accessed together. Rather than spreading this data across multiple tables, you can keep the data together in a document for faster access.
- **Graph:** With a graph data model, you emphasize relationships between data. Graph databases allow you to traverse relationships between objects to find hidden connections between data. This database is commonly used for social networking or fraud-detection services.







When choosing a purpose-built database, you should also consider the performance requirements of your application. Consider not only the speed of your data access and the size of your records, but also where your service will be used in reference to end users.

Speed is vital when your service is serving a critical workload for users who are awaiting a response. If this is applicable to you, you might want to use an in-memory cache to help decrease latency to users. Conversely, speed might be less of a consideration if your service is serving internal analytics or is doing background data processing. You might be more concerned with whether your service can handle the amount of data that is coming into your system.

In addition to these considerations, you should consider geographic requirements for your data. Database services, such as DynamoDB and Aurora, make it easy to replicate your data across the world to bring your data closer to your users and reduce response times.

Finally, consider the operations burden of your database. While it might be tempting to focus solely on developing against your database, you also need to ensure that you have prepared for instance failures, configured backups, and created a plan for upgrades.

Common database use cases

Relational	Key-value	Document	Graph
Traditional applications, enterprise resource planning (ERP), customer relationship management (CRM), ecommerce	High-traffic web applications, ecommerce systems, gaming applications	Content management, catalogs, user profiles	Fraud detection, social networking, recommendation engines
AWS services	AWS services	AWS services	AWS services
 Aurora  Amazon RDS  Amazon Redshift	 DynamoDB	 Amazon DocumentDB	 Neptune



| Slide number 27

| Instructor notes

|

| Student notes

Let's take a moment to look at some common database use cases and the AWS services that are available to support each of them. We'll start with the most common database type: relational databases.

Relational databases are commonly used in traditional applications, enterprise resource planning, customer relationship management, and ecommerce. Organizations use services such as Aurora and Amazon Relational Database Service (Amazon RDS) to streamline their customer experience. Amazon Redshift is another relational database management system. It's optimized for high performance and reporting of large datasets.

High-traffic web applications, ecommerce systems, and gaming applications can all be supported by a key-value database service such as DynamoDB. DynamoDB is a fully managed nonrelational database that provides fast, consistent performance at any scale. High-scale applications and serverless applications are two categories of applications that DynamoDB often supports, but the service can support nearly any OLTP workload.

Content management, catalogs, user profile management, and web and mobile applications depend on document databases. Amazon DocumentDB (with MongoDB compatibility) is a fast, reliable, and fully managed nonrelational document database. You can use the service to store and query rich documents in your applications.

Graph database services, such as Amazon Neptune, can support fraud detection, social networking, knowledge graphs, and recommendation engines with your application. These use cases are all examples of the highly connected data with rich relationship variety that Neptune supports.

When using AWS purpose-built databases, most of the operations burden is handled for you. AWS databases can automatically promote a replica instance in the event that your primary instance fails. Backups and restores are fully managed for you, and you don't need to think about software upgrades. By using fully managed databases from AWS, you can focus on developing features and innovating for your users.

Key takeaways: Purpose-built databases



- Your choice of database will affect what your application can handle, how it will perform, and the operations that you are responsible for.
- When choosing your database, consider several factors:
 - Application workload
 - Data shape
 - Performance
 - Operations burden

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

28

| Slide number 28

| Instructor notes

|

| Student notes

Here are a few key points to summarize this section.

Your choice of database will affect what your application can handle, how it will perform, and the operations that you are responsible for.

When choosing your database, consider several factors.

First, look at your application workload, including the type of data being stored and data access patterns.

Next, consider data shape and understand how you will access your data and how often entities will be updated.

Also consider performance. You need to understand the speed of your data access and the size of your records, as well as where end users will use your service.

And finally, look at the operations burden. Ensure that you have prepared for instance failures, configured backups, and created a plan for upgrades.

Storage in support of the pipeline

Storing and Organizing Data



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| **Slide number 29**

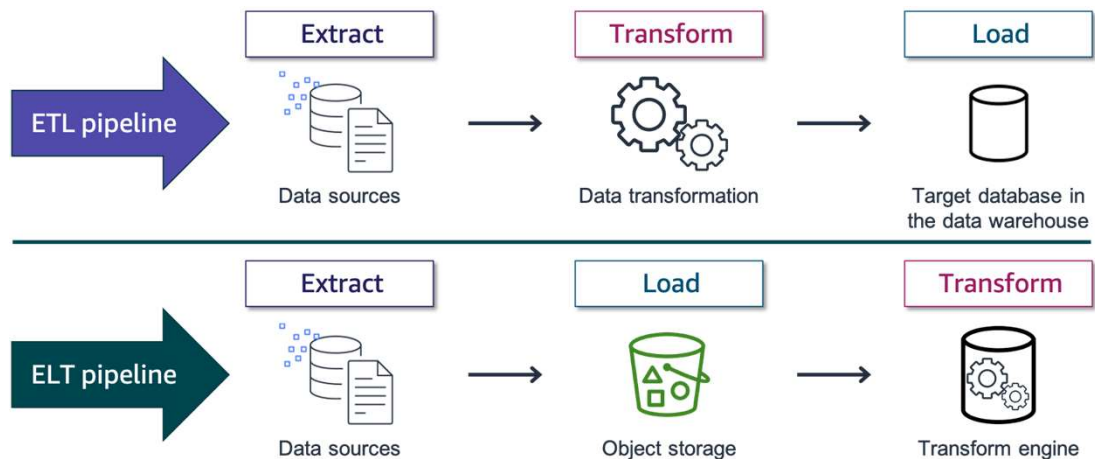
| **Instructor notes**

|

| **Student notes**

This section discusses how storage is used as part of your data pipeline.

Comparing storage in ETL and ELT pipelines



| Slide number 30

| Instructor notes

|

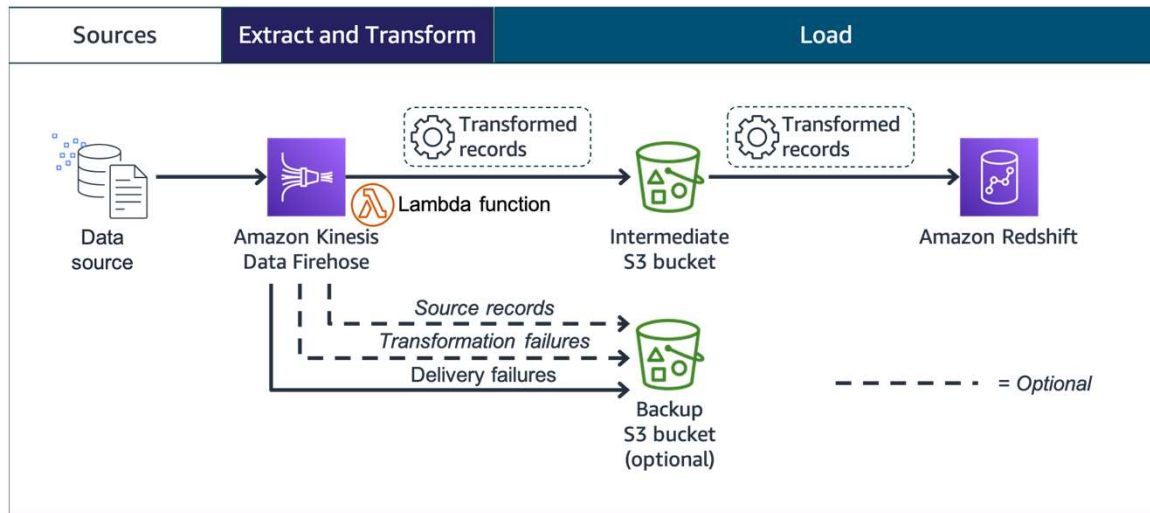
| Student notes

This diagram represents what you have previously learned about extract, transform, and load (ETL) and extract, load, and transform (ELT) pipelines. Recall that with the traditional ETL pipeline, data is extracted from its source and transformed into a structured format. The structured format is ready to be used in analytics applications by using tools such as AWS Glue, Apache Spark, and Apache Hive on Amazon EMR. This transformed data is then loaded into structured storage, such as a data warehouse, where data scientists or analysts can query the data warehouse and perform additional transformations and processing if needed. Note that the data is generally transformed while cached in memory, meaning that the transformation is completed *before* it is stored in a data warehouse.

The trend toward bulk collection of unstructured and semistructured data is shifting the paradigm from the traditional ETL pipeline to the ELT format. Data is extracted from its source and cleaned just enough to be stored in object storage, such as a data lake built on Amazon S3. Then, whichever data transformation engine is built into the data warehouse for relational and SQL workloads accesses the data. This pattern is powerful because it uses the highly optimized and scalable data storage and compute power of massively parallel processing (MPP) architecture.

Let's take a look at some example architectures for both pipeline types in action.

Example: Storage in support of an ETL pipeline



| Slide number 31

| Instructor notes

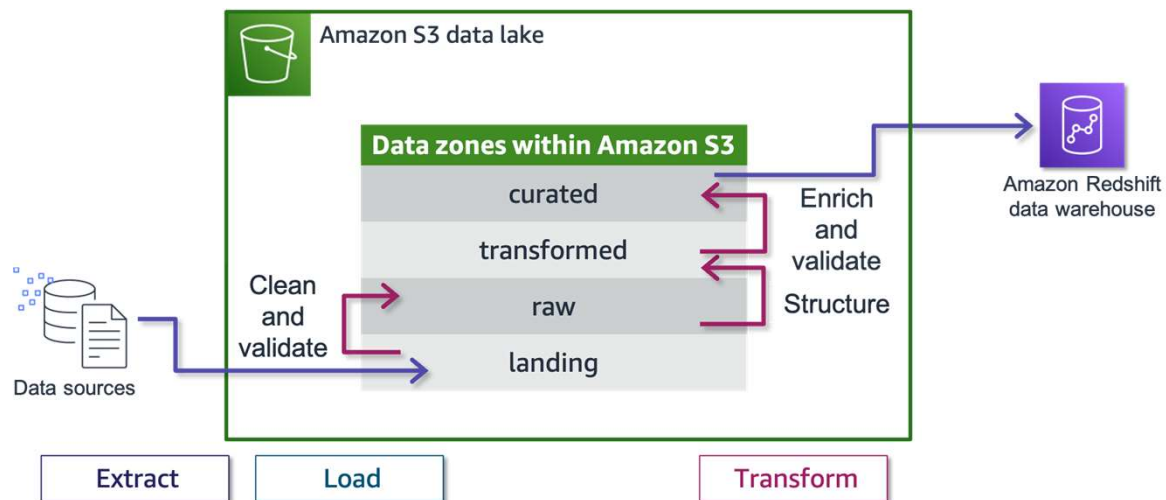
|

| Student notes

This slide provides a high-level example of an ETL pipeline. In this example, the pipeline is an Amazon Kinesis Data Firehose stream using the data transformation mode with data that is destined for a Redshift cluster. Kinesis Data Firehose is an ETL service that captures, transforms, and delivers streaming data to data lakes, data stores, and analytics services. When the data transformation mode is enabled, as Kinesis Data Firehose ingests data, it is first buffered—up to 3 MB data. Kinesis Data Firehose invokes the specified Lambda function, which transforms the data and sends it back to Kinesis Data Firehose. This transformed data is sent to an intermediate S3 bucket, where it waits for Kinesis Data Firehose to issue a COPY command to Amazon Redshift, which then loads data from the intermediate bucket into the Redshift cluster.

The transformed data must contain the recordID, result, and data parameters in order to be returned from Lambda to Kinesis Data Firehose; otherwise, records are treated as data transformation failures and can be sent to the optional backup S3 bucket. These records will contain useful metadata that you can use to troubleshoot and avoid future data transformation failures.

Example: Storage in support of an ELT pipeline



| Slide number 32

| Instructor notes

|

| Student notes

This high-level example provides a generalized ELT architecture that conforms with the storage principles of a modern data architecture. Data sources are ingested and immediately sent to an Amazon S3 landing zone within a data lake. This data is cleaned and validated, and then transferred within the data lake to a raw data zone. As data is further processed, it is loaded, transformed, and moved into the appropriate S3 zones until it is ready for implementation in a data warehouse setting or for use in an analytics engine.

While the ELT pipeline simplifies the architecture, the burden of the transformation workload is placed on the target system. This means that data isn't processed by using an interim transformation in buffered memory. Instead, dedicated compute resources process the data, which greatly improves performance in the pipeline. Thanks to the strong data consistency and in-place querying that Amazon S3 provides, you can efficiently use your data without worrying about corrupt data or migrations.

Key takeaways: Storage in support of the pipeline



- Storage plays an integral part in ELT and ETL pipelines.
- ETL pipelines transform data in buffered memory prior to loading data into a data lake or data warehouse for storage.
- ELT pipelines extract and load data into a data lake or data warehouse for storage without transformation.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

33

| **Slide number 33**

| **Instructor notes**

|

| **Student notes**

Here are a few key points to summarize this section.

Storage plays an integral part in ELT and ETL pipelines. Data often moves in and out of storage numerous times, based on pipeline type and workload type.

ETL pipelines transform data in buffered memory prior to loading data into a data lake or data warehouse for storage. Levels of buffered memory vary by service.

ELT pipelines extract and load data into data lake or data warehouse storage without transformation. The transformation of the data is part of the target system's workload.

Securing storage

Storing and Organizing Data



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| Slide number 34

| Instructor notes

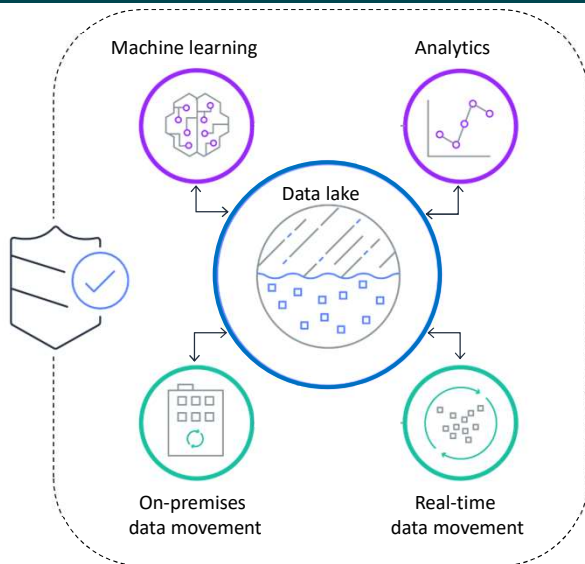
|

| Student notes

This section introduces concepts related to securing storage.

Secure, protect, and manage your AWS data lake

- Benefit from the built-in security of Amazon S3.
- Manage access through resource-based policies and user policies.
- Choose from multiple encryption options.
- Use tags to categorize and manage data and to manage access permissions.
- Use Lake Formation for centralized governance and access control.



| Slide number 35

| Instructor notes

|

| Student notes

Storing large quantities of data in a centralized repository makes security a key element of your architecture planning process. With Amazon S3 at the core of your data lake solution, several robust security features and capabilities are intrinsic to the service and are available to you. You can use these capabilities to help protect your data from both internal and external threats.

The durability of Amazon S3, combined with protections against malicious deletions and corruption, helps to provide a high level of data protection for your data lake. As your data lake grows, the need for fine-grained security controls to secure your data's processing and lifecycle grows. You can combine the inherent security features of Amazon S3 with other AWS services, such as IAM, AWS Key Management Service (AWS KMS), Amazon Cognito, and Amazon API Gateway. The capabilities of these services can help you to ensure that your S3 data lake meets stringent data security, compliance, privacy, and protection requirements.

Access policy options are a good way to manage access to your S3 resources. Access policy options are broadly categorized as resource based and user policies, and they can help you provide access to various AWS accounts and users.

Bucket policies and access control lists (ACLs) are two resource-based policies that are available to you to manage access to your buckets and objects. These policy types are highly customizable, and you can secure your S3 resources down to the object level.

While resource-based policies are an excellent choice to secure your data lake resources, AWS recommends using user policies for most data lake environments. With user policies, you can assign permissions through user roles and permissions to provide access to the data processing and analytics services and tools that your data lake users will use. When you implement user policies along with IAM, you can create IAM users, groups, and roles in accounts and then attach access policies to them that grant access to the AWS resources that make up your data lake.

By using encryption keys to encrypt and decrypt data assets, you can prevent users from inadvertently or maliciously gaining access to your data assets. With Amazon S3 at the core of your data lake, multiple server-side and client-side encryption options are available to you. You can use AWS KMS for centralized control over encryption keys and audit the usage of those keys through the integration of AWS KMS with Amazon CloudWatch. Finally, you can combine the capabilities of services such as API Gateway, Amazon Cognito, and IAM to create a marketplace model where users can check in and check out data lake assets.

In Amazon S3, you can use object tags to categorize and manage your S3 data assets. You can use object tags in conjunction with IAM to enable fine-grained access permissions. You can also use tags to manage S3 data lifecycle policies. Finally, you can combine object tags with CloudWatch metrics and AWS CloudTrail logs to monitor and audit data based on specific data asset tag filters.

In addition to the components and features of Lake Formation that were discussed earlier in this module, you can centrally govern your data lake by defining granular data access policies to the metadata and data through grant or revoke permissions models. The integration of Lake Formation and IAM provides this capability, and you can define both metadata access control and underlying data access control. You can grant access to named resources in your data lake by using tag-based access control (TBAC), which is the recommended method because of its ability to simplify access control while managing a large number of Data Catalog resources and principals.

Security for a data warehouse in Amazon Redshift

- Amazon Redshift database security is distinct from the security of the service itself.
- Amazon Redshift provides additional features to manage database security.
- Due to third-party auditing, Amazon Redshift can help to support applications that are required to meet international compliance standards.
- Amazon Redshift integrates with Amazon CloudWatch, AWS CloudTrail, and AWS Security Hub for monitoring and alerting.



| Slide number 36

| Instructor notes

|

| Student notes

Amazon Redshift handles database security separately from the security of the service itself. While access to Amazon Redshift is managed through identity and access management measures, the following additional security features are included to manage database security:

- **Sign-in credentials:** Your AWS account permissions control access to the Redshift console.
- **Access management:** To control access to specific Redshift resources, you define IAM accounts.
- **Cluster security groups:** To grant inbound access to a Redshift cluster to other users, you define a cluster security group and associate it with a cluster.
- **VPC:** To protect access to your cluster by using a virtual networking environment, you can launch your cluster in a virtual private cloud (VPC).
- **Cluster encryption:** To encrypt the data in all user-created tables, you can turn on cluster encryption when you launch the cluster.
- **SSL connections:** To encrypt the connection between your SQL client and your cluster, you can use SSL encryption.

- **Load data encryption:** To encrypt your table load data files when you upload them to Amazon S3, you can use either server-side encryption or client-side encryption. When you load from server-side encrypted data, Amazon S3 handles decryption transparently. When you load from client-side encrypted data, the Redshift COPY command decrypts the data as it loads the table.
- **Data in transit:** To protect your data in transit within the AWS Cloud, Amazon Redshift uses hardware accelerated SSL to communicate with Amazon S3 or DynamoDB for COPY, UNLOAD, backup, and restore operations.
- **Column-level access control:** To have column-level access control for data in Amazon Redshift, use column-level grant and revoke statements without needing to implement views-based access control or use another system.
- **Row-level security control:** To have row-level security control, define security policies at an object level.

Amazon Redshift uses third-party auditors to assess security and compliance. Therefore, using Amazon Redshift can help you to meet compliance and security requirements for handling sensitive data.

For monitoring and alerting, Amazon Redshift integrates with CloudWatch, CloudTrail, and AWS Security Hub to provide increased oversight and visibility over the service and its resources.

Key takeaways: Securing storage



- Security for data lake storage is built upon the intrinsic security features of Amazon S3.
- Access policies provide a highly customizable way to provide access to resources in your data lake.
- Data lakes that are built on AWS rely on server-side and client-side encryption.
- Amazon Redshift handles service security and database security as two distinct functions.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

37

| Slide number 37

| Instructor notes

|

| Student notes

Here are a few key points to summarize this section.

With Amazon S3 as the core of your data lake, the security features that are inherent to the service are a central part of your data lake architecture.

Resource-based and user access policies are highly customizable ways to provide or restrict access to resources in your data lake.

Data lakes that are built on AWS rely on server-side and client-side encryption.

Amazon Redshift handles the security of the service and the security of its databases as two distinct functions. This allows for greater security of the data that is stored within.

Lab: Storing and Analyzing Data by Using Amazon Redshift



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| Slide number 38

| Instructor notes

|

| Student notes

You will now complete a lab. The next slide summarizes what you will do in the lab, and you will find the detailed instructions in the lab environment.

Lab introduction: Storing and Analyzing Data by Using Amazon Redshift



- In this lab, you will build a proof of concept to use Amazon Redshift to address the need to query large datasets.
- You will use SQL queries for a music ticket sales dataset. These data files are already stored in Amazon S3 in another account, and you will load them into a Redshift database.
- Open your lab environment to start the lab and find additional details about the tasks that you will perform during this lab.

| Slide number 39

| Instructor notes

|

| Student notes

Access the lab environment through your online course to get additional details and complete the lab.

Debrief: Storing and Analyzing Data by Using Amazon Redshift

- What is a Redshift cluster? What is the purpose of the leader node and the compute nodes in a cluster?
 - Can an AWS admin user change the size or type of a Redshift node?
- Can you use Athena to query a Redshift database?
 - How can you query a Redshift database?
- What skills would a data engineer need to be able to use Amazon Redshift?
- What was the purpose of the MyRedshiftRole IAM role in this lab?



| Slide number 40

| Instructor notes

| Q1 – **Example strong response:** A cluster is the main infrastructure component of a Redshift data warehouse. A cluster has a leader node and one or more compute nodes. Client applications interact with the leader node. The leader node coordinates with the compute nodes to run queries and then returns the results to client applications. The compute nodes communicate with each other to complete queries. **Follow-up to Q1 response:** Yes, Redshift node instances vary by the needs of the workload.

| Q2 – **Example strong response:** No. You can use Athena to query an AWS Glue database but not a Redshift database. **Follow-up to Q2 response:** You can query a Redshift database from the query editor in the Redshift console and from the Redshift API by using either the AWS CLI or SDK.

| Q3 – **Example strong response:** A data engineer would need experience with relational databases, data types (such as character, numerical, and boolean), and writing SQL queries. They would also need experience with other AWS services, such as Amazon EC2, Amazon S3, and IAM. A data engineer would also need a good understanding of the AWS API and how to use it with Redshift in application integration.

| Q4 – **Example strong response:** The role contained permissions that allowed Amazon Redshift to access and read S3 buckets and the objects contained in them. This access was needed to load data into the database. The role also contained permissions to allow Redshift to describe Amazon EC2 and Amazon VPC resources. These permissions were needed to create and configure the Redshift cluster.

|

|Student notes

Your instructor might review these questions with you, or you might review them on your own. Use this opportunity to extend your thinking about the tasks that you performed during the lab.



| **Slide number 41**

| **Instructor notes**

|

| **Student notes**

This section summarizes what you have learned and brings the module to a close.

Module summary

This module prepared you to do the following:

- Define storage types that are found in a modern data architecture.
- Distinguish between data storage types.
- Select data storage options that match your storage needs.
- Implement secure storage practices for cloud-based data.



| Slide number 42

| Instructor notes

| This is a good opportunity to use an online group or discussion board to ask students to reflect on what they have learned. You might ask the students to recall a point from the module that aligns to one of the listed objectives. This provides a good segue to the knowledge check and sample exam question.

|

| Student notes

This module defined the storage types that are found in a modern data architecture. You also learnt to distinguish between the storage types and select the option that matches the requirements and constraints of your use case. And finally, you learnt about secure storage practices for cloud-based data.

Module knowledge check



- The knowledge check is delivered online within your course.
- The knowledge check includes 10 questions based on material presented on the slides and in the slide notes.
- You can retake the knowledge check as many times as you like.

| **Slide number 43**

| **Instructor notes**

|

| **Student notes**

Use your online course to access the knowledge check for this module.

Sample exam question

A data engineer is designing an infrastructure and wants users to be able to query data directly from files in the company's data lake, which is built on Amazon S3.

Which service feature would enable this capability?

Identify the key words and phrases before continuing.

The following are the key words and phrases:

- **Query data directly** from files in the company's **data lake**, which is **built on Amazon S3**
- **Service feature**



| Slide number 44

| Instructor notes

| The key words section is animated to be revealed on click.

|

| Student notes

The question implies the need for the capability to query data directly from files stored in a data lake that has been built using Amazon S3.

Sample exam question: Response options

A data engineer is designing an infrastructure and wants users to be able to **query data directly** from files in the company's **data lake**, which is **built on Amazon S3**.

Which **service feature** would enable this capability?

Choice	Response
A	AWS Lake Formation
B	Amazon Redshift Spectrum
C	AWS Glue crawlers
D	Amazon Neptune graph queries



| Slide number 45

| Instructor notes

|


| Student notes

Use the key words that you identified on the previous slide, and review each of the possible responses to determine which one best addresses the question.

Sample exam question: Answer

The correct answer is B.

Choice	Response
A	
B	Amazon Redshift Spectrum
C	
D	

 © 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved. 46

| Slide number 46

| Instructor notes

|

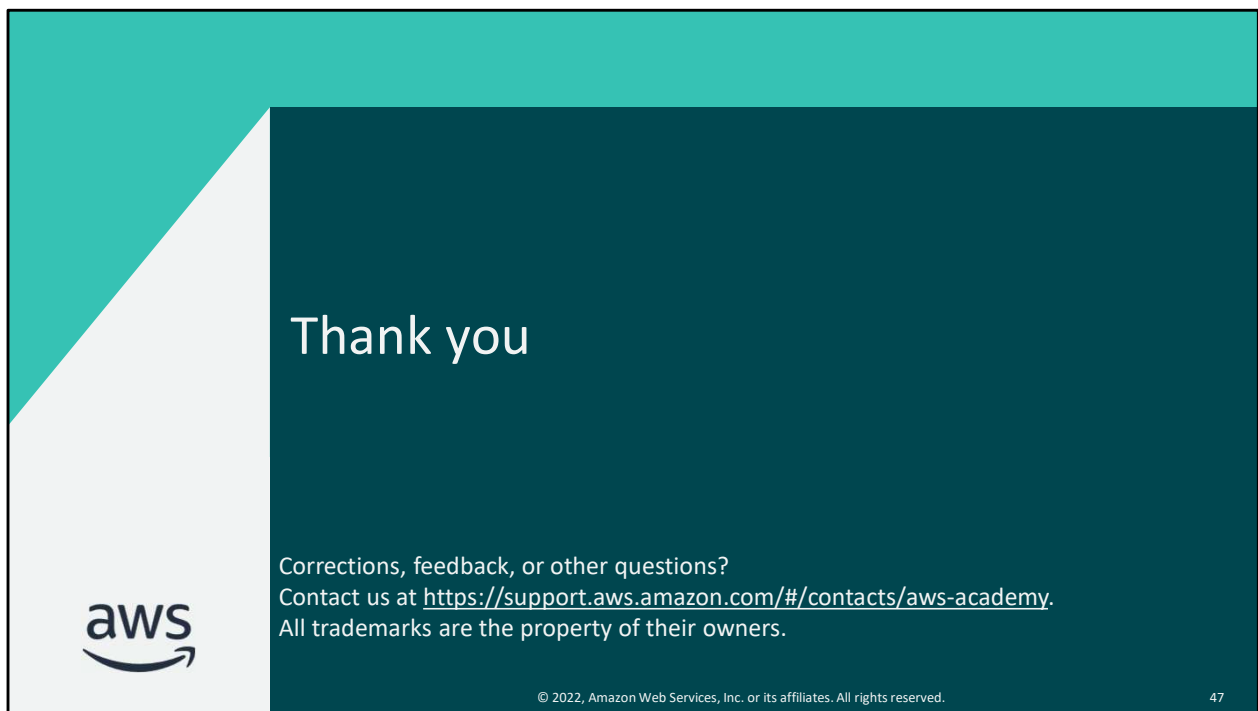
| Student notes

Choice A (AWS Lake Formation) is incorrect. Lake Formation is a service—not a specific service feature.

Choice C (AWS Glue crawlers) is incorrect. Crawlers are used to populate the AWS Glue Data Catalog with tables; they are not used to perform queries.

Choice D (Amazon Neptune graph queries) is incorrect. Data from Amazon S3 would first need to be loaded into Neptune and couldn't be queried directly from the data lake.

The correct answer is B (Amazon Redshift Spectrum). With Redshift Spectrum, you can extend your analytics beyond the data that is stored on local disks and perform queries on data that is stored in Amazon S3.



| Slide number 47

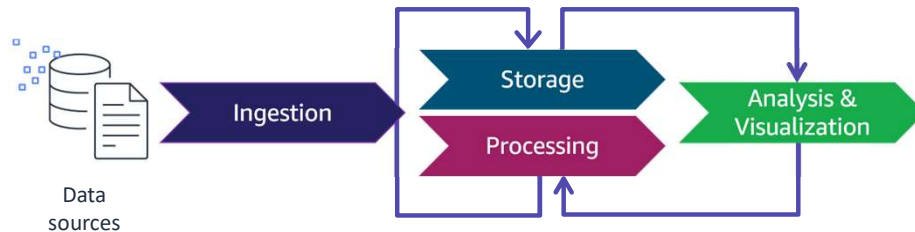
| Instructor notes

|

| Student notes

That concludes this module. The Content Resources page of your course includes links to additional resources that are related to this module.

The simplified iterative data pipeline



| Slide number 48

| Instructor notes

| Review

|

| Student notes

You have learned that, in practice, storing data is directly related to the tasks of ingesting and processing data.

Storage in the iterative data pipeline is far from a linear process. When data is ingested into the pipeline, it is placed in storage. The data can then be removed from storage, processed, and returned to storage for later use or to await additional processing and transformation. This in-and-out cycle of data illustrates that storage is key to your data pipeline. Therefore, your planning efforts should focus on how storage will be used to support your pipeline goals.

In this module, you will learn how to select the appropriate storage for your needs, and how data is stored and organized to support your data analytics and machine learning (ML) pipelines.

Storage in the AWS modern data architecture



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

49

| Slide number 49

| Instructor notes

| Note to students that all the items that are listed on this slide are parts of a modern data architecture. Point out that the circles represent AWS services, while the inner ring of floating words are the elements of the modern data architecture that those services handle.

|

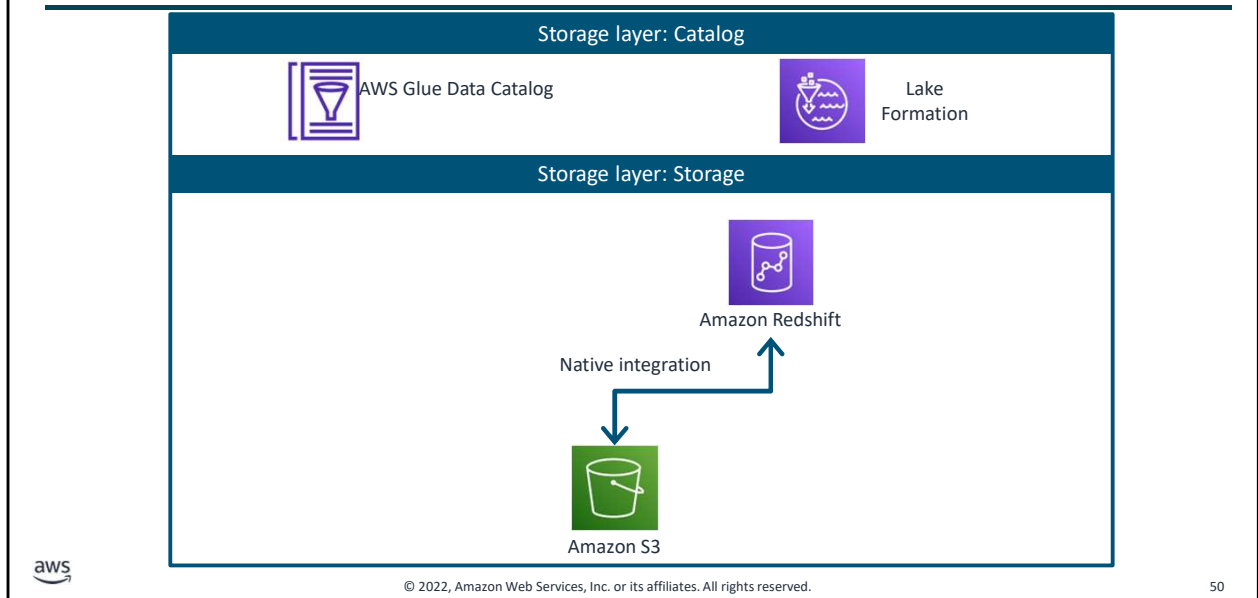
| Student notes

In the Design Principles and Patterns for Data Pipelines module, you learned about the AWS modern data architecture, including the tools and services that you can use to manage and govern your resources. In the diagram on this slide, note that storage is at the center—it's the core of this architecture. In the AWS Cloud, Amazon Simple Storage Service (Amazon S3) provides you with the object storage that is central to your needs. Amazon Redshift provides additional storage as data warehousing. The service stores the structured and semistructured data from relational databases, such as Amazon Aurora, and nonrelational databases, such as Amazon DynamoDB. Note that nonrelational databases are sometimes called NoSQL databases.

The type and volume of data is growing at an unprecedented rate as technology evolves. Organizations want to capture all of this data to derive value from it as quickly as possible.

A modern data architecture will give you the best of both data lakes and purpose-built data stores, providing you with a low-cost storage solution for any amount of data using open standards-based data formats. A modern data architecture isn't restricted by data silos, which enables you to empower individuals or groups to run analytics and ML workloads by using their preferred tools or techniques.

Storage layer



| Slide number 50

| Instructor notes

|

| Student notes

The data storage layer provides durable, scalable, and cost-effective components to store and manage vast quantities of data. In the AWS architecture, Amazon Redshift and Amazon S3 provide unified, natively integrated storage.

The catalog layer in the storage layer stores business and technical metadata about datasets that are hosted in the storage layer. This metadata supports the ability to find and query data that is stored in the data lake and the data warehouse. In the AWS architecture, AWS Lake Formation and AWS Glue work together to collect and store metadata and make it available when needed. The catalog makes it easier for consumers to search for and explore the available data.

Amazon S3 provides you with object storage for structured and unstructured data. With S3 as the foundation of your data lake architecture, you can easily store, access, and query data from a multitude of data sources. You can then use this data immediately or place it in a low-cost, long-term storage solution, such as Amazon S3 Glacier, for later use and analysis. With the scalable, flexible, and durable storage that Amazon S3 provides, you can

support the five Vs that were discussed in The Elements of Data module.

Amazon Redshift uses SQL to analyze structured and semistructured data across data warehouses, operational databases, and data lakes. The service uses AWS designed hardware and ML to deliver the best price performance at any scale. Because the service is integrated with AWS services, including database and ML services, Amazon Redshift can help you handle complete analytics workflows.

Comparing data lakes and data warehouses

Characteristic	Data Warehouse	Data Lake
Data	Relational data from transactional systems, operational databases, and line of business applications	Nonrelational and relational data from Internet of Things (IoT) devices, websites, mobile apps, social media, and corporate applications
Schema	Designed prior to the data warehouse implementation (schema on write)	Written at the time of analysis (schema on read)
Price and Performance	Fastest query results using higher cost storage	Query results getting faster using low-cost storage
Data Quality	Highly curated data that serves as the central version of the truth	Any data, which might or might not be curated (for example, raw data)
Users	Business analysts	Data scientists, data developers, and business analysts (using curated data)
Analytics	Batch reporting, business intelligence (BI), and visualizations	ML, predictive analytics, and data discovery and profiling



| Slide number 51

| Instructor notes

| Review this slide with students and discuss how the differences would apply to various scenarios. Relay to students that data lakes and data warehouses are complementary—rather than competing—storage types. Draw students' attention to the data quality characteristic, which distinguishes them from each other.

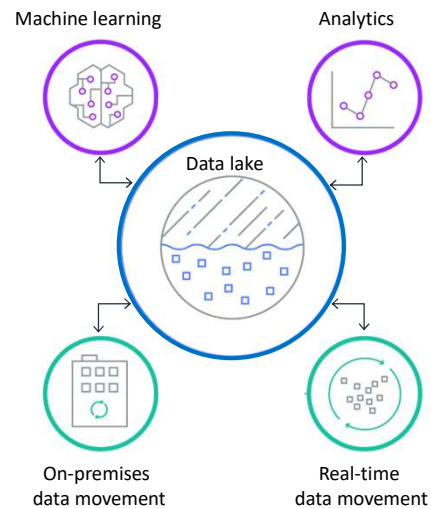
|

| Student notes

Take a moment to review the differences between data lakes and data warehouses. They are complementary storage types, with data warehouses drawing much of their data from the vast data repositories that are found in data lakes. The key difference lies in the data quality characteristic. A data warehouse stores highly curated data, whereas the data in a data lake can be raw, untransformed, and uncurated.

Data lakes

- Provide a centralized repository
- Store both structured and unstructured data
- Catalog and index data for analysis without data movement
- Store, secure, and protect data at unlimited scale
- Offer in-place transformation and querying of data assets
- Are built using Amazon S3



| Slide number 52

| Instructor notes

|

| Student notes

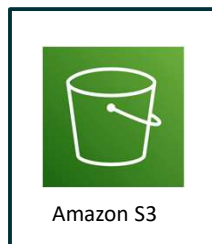
Data lakes are centralized repositories where you can store structured and unstructured data, regardless of scale. Unlike with a data warehouse, you can store data as-is in a data lake. It isn't necessary to structure the data before you begin to run analytics.

Data lakes can store both relational data, such as data from a line of business application, and nonrelational data, from sources such as mobile apps, Internet of Things (IoT) devices, and social media. You can store data without a particular focus on design or usage plans because the structure of the data or schema is not defined when the data is captured.

Data lakes in AWS provide you with secure, scalable, and durable storage. You can employ AWS services to run your analytics and ML processing, or you can select a preferred third-party analytics platform from the AWS Partner Network (APN). After you migrate your data into an S3 data lake, you can employ purpose-built analytics services, such as Amazon Athena and Amazon EMR; launch artificial intelligence (AI) and ML jobs; and transform and query data in place. Additionally, data lakes in AWS can take advantage of Amazon S3 storage classes, which are purpose built to provide the lowest cost storage for different

access patterns.

Amazon S3 storage classes



General purpose



S3 Standard

Infrequent access



S3 Standard-
Infrequent
Access

Archive



S3 Glacier
Instant
Retrieval

Unknown or changing access



S3 Intelligent-
Tiering



S3 One Zone-
Infrequent
Access



S3 Glacier
Flexible
Retrieval



S3 Glacier Deep
Archive



| Slide number 53

| Instructor notes

|

| Student notes

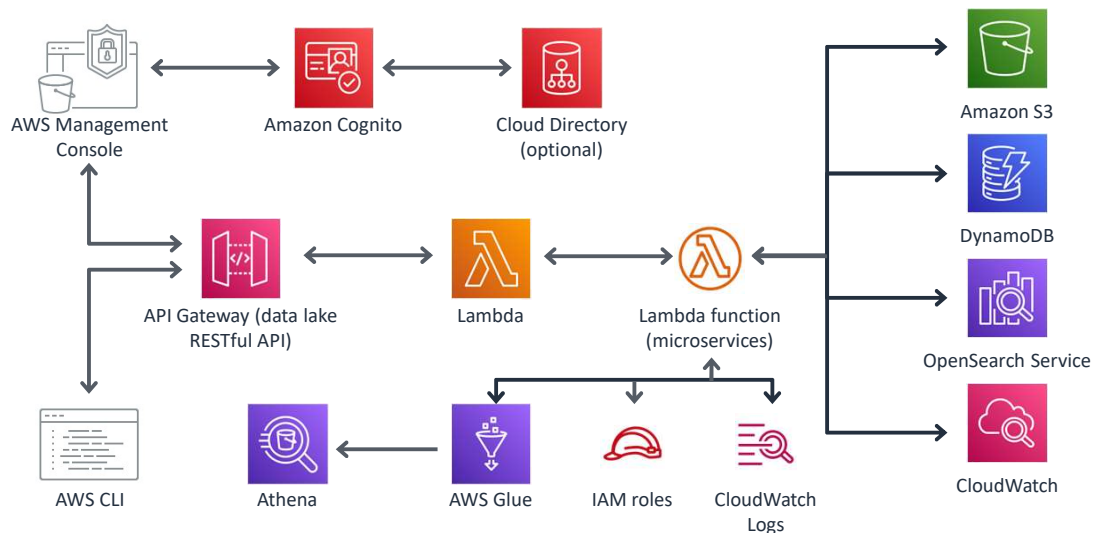
By using Amazon S3 storage classes, you can choose how to store data based on access needs and cost considerations. Analytics and ML workloads require large quantities of data, and you want to ensure that you store that data in the most cost-effective manner as possible.

The storage classes include S3 Intelligent-Tiering for automatic cost savings for data with unknown or changing access patterns and S3 Standard for frequently accessed data. S3 Standard-Infrequent Access (S3 Standard-IA) and S3 One Zone-Infrequent Access (S3 One Zone-IA) are available for less frequently accessed data. S3 Glacier Instant Retrieval is available to archive data that needs immediate access, and S3 Glacier Flexible Retrieval (formerly S3 Glacier) is available for rarely accessed long-term data that doesn't require immediate access. For long-term archive and digital preservation, with retrieval in hours at the lowest cost storage in the cloud, use S3 Glacier Deep Archive.

If you have data residency requirements that can't be met by an existing AWS Region, you can use the S3 Outposts storage class to store your S3 data on premises. Amazon S3 also

offers capabilities to manage your data throughout its lifecycle. For example, you can set a lifecycle policy to automatically transfer data to a different storage class without any changes to your application. For more information about Amazon S3 storage classes, see the resources for this module in the Content Resources page of your course.

Example architecture of a data lake



| Slide number 54

| Instructor notes

|

| Student notes

This slide shows an example architecture for a data lake and the services that are common in it. This example uses AWS Lambda microservices (functions), Amazon OpenSearch Service for search capabilities, Amazon Cognito for user authentication, AWS Glue for data transformation, and Amazon Athena for data analysis.

If you would like to explore this data lake architecture more fully, the code to configure this example is available for download on GitHub. A link is available in the Content Resources page of your course.

Data warehouses

- Provide a centralized repository
- Store structured and semistructured data
- Store data in one of two ways:
 - Frequently accessed data in fast storage
 - Infrequently accessed data in cheap storage
- Might contain multiple databases that are organized into tables and columns
- Separate analytics processing from transactional databases
- Example: Amazon Redshift



| Slide number 55

| Instructor notes

|

| Student notes

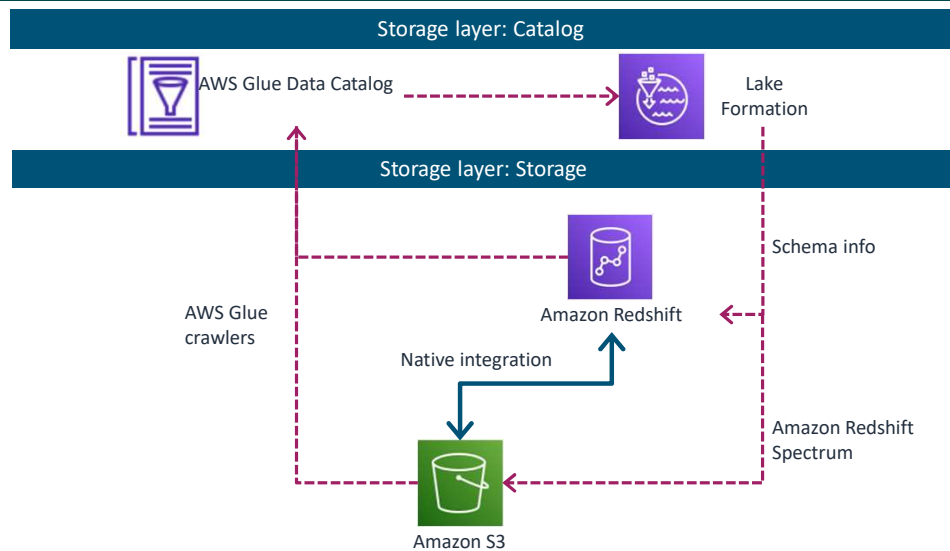
Data warehouses are centralized data repositories. The data that is contained within the data warehouses can be processed and analyzed to make more informed decisions. Structured and semistructured data from sources such as transactional systems and relational databases flows into the data warehouse in a regular cadence.

Data warehouses consist of three tiers. At the top is a front-end client, which presents your results through reporting, analysis, and data mining tools. The middle tier is made up of an analytics engine, which is used to access and analyze the data. The third tier consists of the database server, where your data is loaded and stored. This data is stored in one of two ways: in fast storage, such as SSD drives, if it is frequently accessed, or in low-cost object storage, such as Amazon S3, if it is infrequently accessed. Data can be automatically shifted between the two storage types based on how frequently it is accessed.

A data warehouse can contain multiple databases of structured and semistructured data—data that is organized into tables and columns. When data is ingested into the data warehouse, it can be stored in tables, each of which can be organized inside of schemas.

Query tools use these schemas to determine which data tables need to be accessed and analyzed.

Amazon Redshift Spectrum



| Slide number 56

| Instructor notes

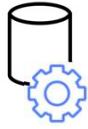
|

| Student notes

Using Amazon Redshift Spectrum, you can conduct fast, in-place querying against data that is stored in data lakes on Amazon S3. You can make Redshift Spectrum queries on a wide variety of data lake assets, including comma-separated value (CSV), tab-separated value (TSV), Parquet, Sequence, and RCFile formatted data.

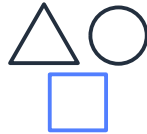
With Redshift Spectrum, you can write SQL queries that combine data from the data lake and the data warehouse. When a user makes a query request that includes data from the data lake, Redshift Spectrum gets schema information from the Lake Formation catalog and uses it to query the data lake.

Factors in choosing a purpose-built database



Application workload

- Is your workload transactional?
- Will your workload be used for analytics purposes?
- Does your workload need caching to improve response times?



Data shape

- How will your data be accessed?
- How often will your data be updated?



Performance

- How fast does your data access need to be?
- What is the average size of the records that you are using?
- How will end users use your service?



Operations burden

- How will you prepare for instance failures?
- How will you configure backups?
- What future upgrades might you need?



| Slide number 57

| Instructor notes

|

| Student notes

When choosing a purpose-built database, the first factor to consider is the workload of your application. Workload is about the type of data that your application stores and the data access patterns. There are three broad categories of workloads.

Transactional: A *transactional workload* is characterized by a high number of concurrent operations, where each operation reads or writes a small number of rows. Most user-facing applications, including ecommerce, mobile gaming, and social networking, are transactional. Transactional workloads are also referred to as online transactional processing, or OLTP. Online shopping services are a great example of an OLTP workload.

Analytical: *Analytical workloads* aggregate and summarize large volumes of data. There are usually far fewer concurrent queries in analytical data stores, but they are operating on many more rows per query. Analytical access patterns are usually for internal applications such as reporting. These workloads are also referred to as online analytical processing, or OLAP. Your organization can use OLAP workloads for strategic planning. Imagine your

business sells widgets in multiple sizes, shapes, and colors nationwide. An OLAP workload can combine this data set with purchase data sets that give information about purchase locations, quantities, and more to determine what types of widget should be kept in larger supply based on such factors.

Caching: In a *caching workload*, you compute and store frequently accessed data in a separate database for faster response times. This approach reduces the load on your transactional database and improves response times to your end users. The cache is a secondary source that stores derived data from your transactional workloads rather than acting as a primary source of data.

After considering your application workload, the next factor to consider is the shape of your data. When considering data shape, determine the types of entities that you will model and the relationships between your entities. Ask yourself how you will access your data and how often entities will be updated.

After you determine your data shape, choose the database type that best suits your needs. Here are some of the common data models and use cases for each of them.

- **Relational:** The relational database is a well-known format for many developers. In a relational database, you normalize your data into separate tables and assemble related entities together at query time. In the context of databases, the term *normalization* refers to the process of organizing the columns (attributes) and tables (relations) of a relational database to minimize data redundancy. A relational database is a good choice when you have multiple related entities with varying update patterns. The strict schema validation and normalized data model help you maintain data integrity across your application.
- **Key-value or wide-column:** These data models are designed for scale, with your data being split across multiple storage nodes. As your data grows, additional storage nodes can be added to accommodate. This partitioning scheme allows for nearly infinite scalability with no performance degradation.
- **Document:** A document data model uses large records called *documents* to assemble heterogeneous bits of data that are frequently accessed together. Rather than spreading this data across multiple tables, you can keep the data together in a document for faster access.
- **Graph:** With a graph data model, you emphasize relationships between data. Graph databases allow you to traverse relationships between objects to find hidden connections between data. This database is commonly used for social networking or fraud-detection services.

When choosing a purpose-built database, you should also consider the performance requirements of your application. Consider not only the speed of your data access and the size of your records, but also where your service will be used in reference to end users.

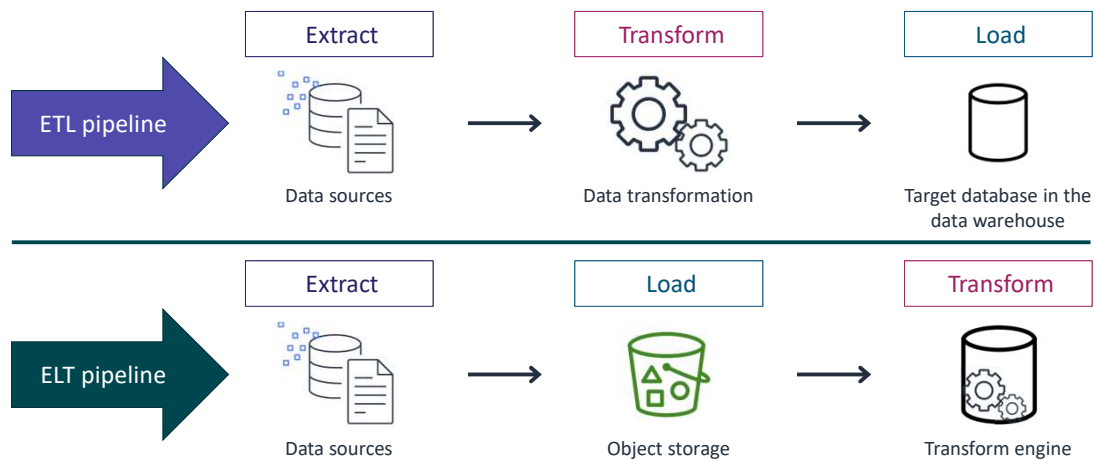
Speed is vital when your service is serving a critical workload for users who are awaiting a

response. If this is applicable to you, you might want to use an in-memory cache to help decrease latency to users. Conversely, speed might be less of a consideration if your service is serving internal analytics or is doing background data processing. You might be more concerned with whether your service can handle the amount of data that is coming into your system.

In addition to these considerations, you should consider geographic requirements for your data. Database services, such as DynamoDB and Aurora, make it easy to replicate your data across the world to bring your data closer to your users and reduce response times.

Finally, consider the operations burden of your database. While it might be tempting to focus solely on developing against your database, you also need to ensure that you have prepared for instance failures, configured backups, and created a plan for upgrades.

Comparing storage in ETL and ELT pipelines



| Slide number 58

| Instructor notes

|

| Student notes

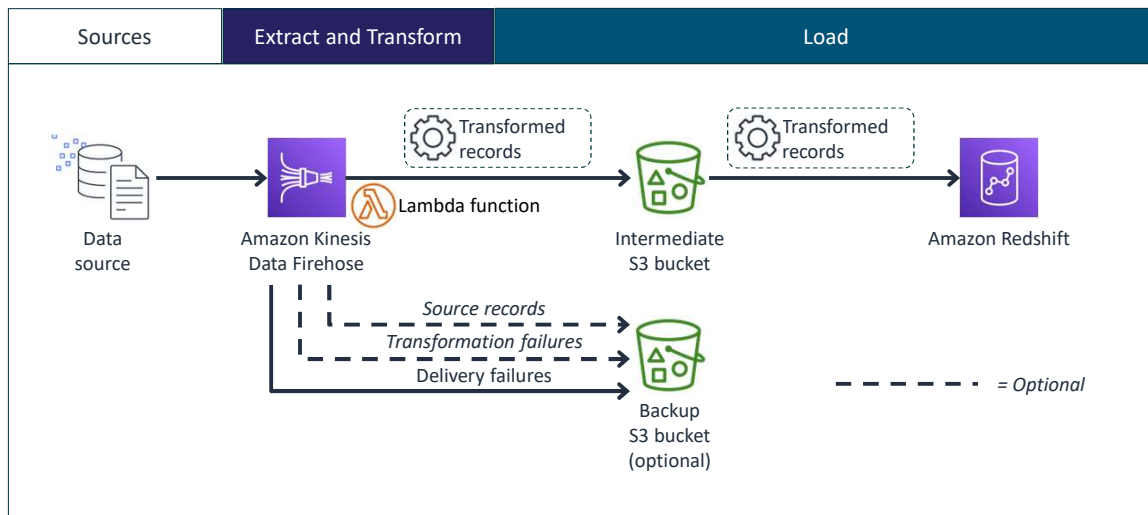
This diagram represents what you have previously learned about extract, transform, and load (ETL) and extract, load, and transform (ELT) pipelines. Recall that with the traditional ETL pipeline, data is extracted from its source and transformed into a structured format. The structured format is ready to be used in analytics applications by using tools such as AWS Glue, Apache Spark, and Apache Hive on Amazon EMR. This transformed data is then loaded into structured storage, such as a data warehouse, where data scientists or analysts can query the data warehouse and perform additional transformations and processing if needed. Note that the data is generally transformed while cached in memory, meaning that the transformation is completed *before* it is stored in a data warehouse.

The trend toward bulk collection of unstructured and semistructured data is shifting the paradigm from the traditional ETL pipeline to the ELT format. Data is extracted from its source and cleaned just enough to be stored in object storage, such as a data lake built on Amazon S3. Then, whichever data transformation engine is built into the data warehouse for relational and SQL workloads accesses the data. This pattern is powerful because it uses the highly optimized and scalable data storage and compute power of massively parallel

processing (MPP) architecture.

Let's take a look at some example architectures for both pipeline types in action.

Example: Storage in support of an ETL pipeline



| Slide number 59

| Instructor notes

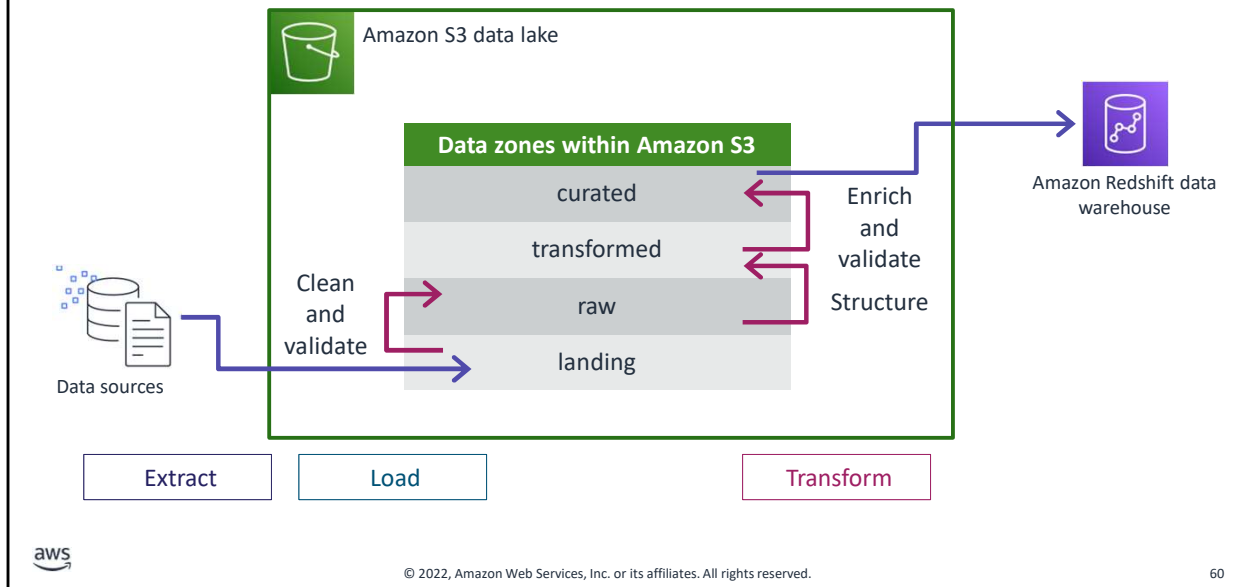
|

| Student notes

This slide provides a high-level example of an ETL pipeline. In this example, the pipeline is an Amazon Kinesis Data Firehose stream using the data transformation mode with data that is destined for a Redshift cluster. Kinesis Data Firehose is an ETL service that captures, transforms, and delivers streaming data to data lakes, data stores, and analytics services. When the data transformation mode is enabled, as Kinesis Data Firehose ingests data, it is first buffered—up to 3 MB data. Kinesis Data Firehose invokes the specified Lambda function, which transforms the data and sends it back to Kinesis Data Firehose. This transformed data is sent to an intermediate S3 bucket, where it waits for Kinesis Data Firehose to issue a COPY command to Amazon Redshift, which then loads data from the intermediate bucket into the Redshift cluster.

The transformed data must contain the recordID, result, and data parameters in order to be returned from Lambda to Kinesis Data Firehose; otherwise, records are treated as data transformation failures and can be sent to the optional backup S3 bucket. These records will contain useful metadata that you can use to troubleshoot and avoid future data transformation failures.

Example: Storage in support of an ELT pipeline



| Slide number 60

| Instructor notes

|

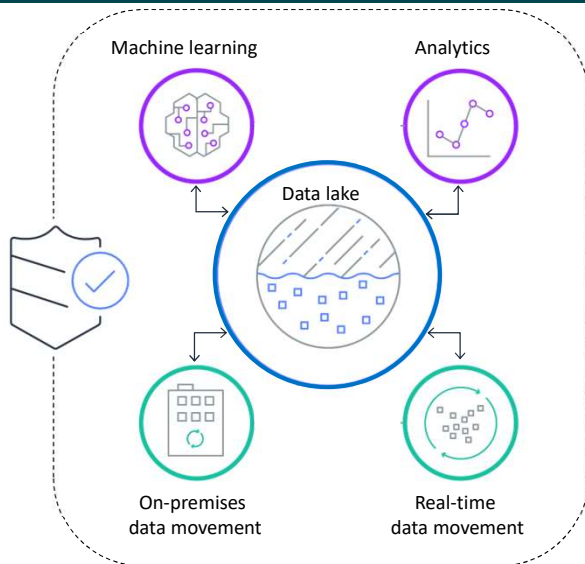
| Student notes

This high-level example provides a generalized ELT architecture that conforms with the storage principles of a modern data architecture. Data sources are ingested and immediately sent to an Amazon S3 landing zone within a data lake. This data is cleaned and validated, and then transferred within the data lake to a raw data zone. As data is further processed, it is loaded, transformed, and moved into the appropriate S3 zones until it is ready for implementation in a data warehouse setting or for use in an analytics engine.

While the ELT pipeline simplifies the architecture, the burden of the transformation workload is placed on the target system. This means that data isn't processed by using an interim transformation in buffered memory. Instead, dedicated compute resources process the data, which greatly improves performance in the pipeline. Thanks to the strong data consistency and in-place querying that Amazon S3 provides, you can efficiently use your data without worrying about corrupt data or migrations.

Secure, protect, and manage your AWS data lake

- Benefit from the built-in security of Amazon S3.
- Manage access through resource-based policies and user policies.
- Choose from multiple encryption options.
- Use tags to categorize and manage data and to manage access permissions.
- Use Lake Formation for centralized governance and access control.



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

61

| Slide number 61

| Instructor notes

|

| Student notes

Storing large quantities of data in a centralized repository makes security a key element of your architecture planning process. With Amazon S3 at the core of your data lake solution, several robust security features and capabilities are intrinsic to the service and are available to you. You can use these capabilities to help protect your data from both internal and external threats.

The durability of Amazon S3, combined with protections against malicious deletions and corruption, helps to provide a high level of data protection for your data lake. As your data lake grows, the need for fine-grained security controls to secure your data's processing and lifecycle grows. You can combine the inherent security features of Amazon S3 with other AWS services, such as IAM, AWS Key Management Service (AWS KMS), Amazon Cognito, and Amazon API Gateway. The capabilities of these services can help you to ensure that your S3 data lake meets stringent data security, compliance, privacy, and protection requirements.

Access policy options are a good way to manage access to your S3 resources. Access policy

options are broadly categorized as resource based and user policies, and they can help you provide access to various AWS accounts and users.

Bucket policies and access control lists (ACLs) are two resource-based policies that are available to you to manage access to your buckets and objects. These policy types are highly customizable, and you can secure your S3 resources down to the object level.

While resource-based policies are an excellent choice to secure your data lake resources, AWS recommends using user policies for most data lake environments. With user policies, you can assign permissions through user roles and permissions to provide access to the data processing and analytics services and tools that your data lake users will use. When you implement user policies along with IAM, you can create IAM users, groups, and roles in accounts and then attach access policies to them that grant access to the AWS resources that make up your data lake.

By using encryption keys to encrypt and decrypt data assets, you can prevent users from inadvertently or maliciously gaining access to your data assets. With Amazon S3 at the core of your data lake, multiple server-side and client-side encryption options are available to you. You can use AWS KMS for centralized control over encryption keys and audit the usage of those keys through the integration of AWS KMS with Amazon CloudWatch. Finally, you can combine the capabilities of services such as API Gateway, Amazon Cognito, and IAM to create a marketplace model where users can check in and check out data lake assets.

In Amazon S3, you can use object tags to categorize and manage your S3 data assets. You can use object tags in conjunction with IAM to enable fine-grained access permissions. You can also use tags to manage S3 data lifecycle policies. Finally, you can combine object tags with CloudWatch metrics and AWS CloudTrail logs to monitor and audit data based on specific data asset tag filters.

In addition to the components and features of Lake Formation that were discussed earlier in this module, you can centrally govern your data lake by defining granular data access policies to the metadata and data through grant or revoke permissions models. The integration of Lake Formation and IAM provides this capability, and you can define both metadata access control and underlying data access control. You can grant access to named resources in your data lake by using tag-based access control (TBAC), which is the recommended method because of its ability to simplify access control while managing a large number of Data Catalog resources and principals.