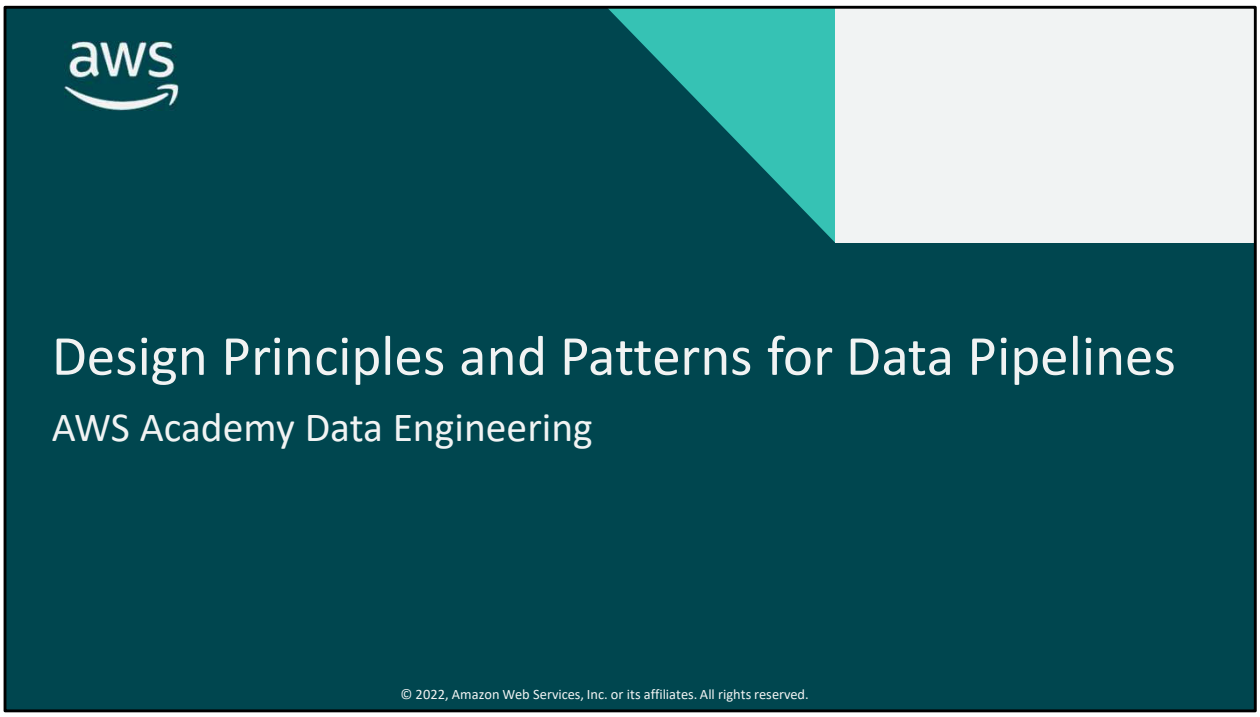


|Slide number 1

|Instructor notes

|This module introduces the AWS Well-Architected Framework and its Data Analytics Lens. You might find it helpful to review the Well-Architected documentation when preparing for the course and to keep the web-based resource open while going through this module. Links for the Well-Architected resources are provided in the Content Resources section of the online course.

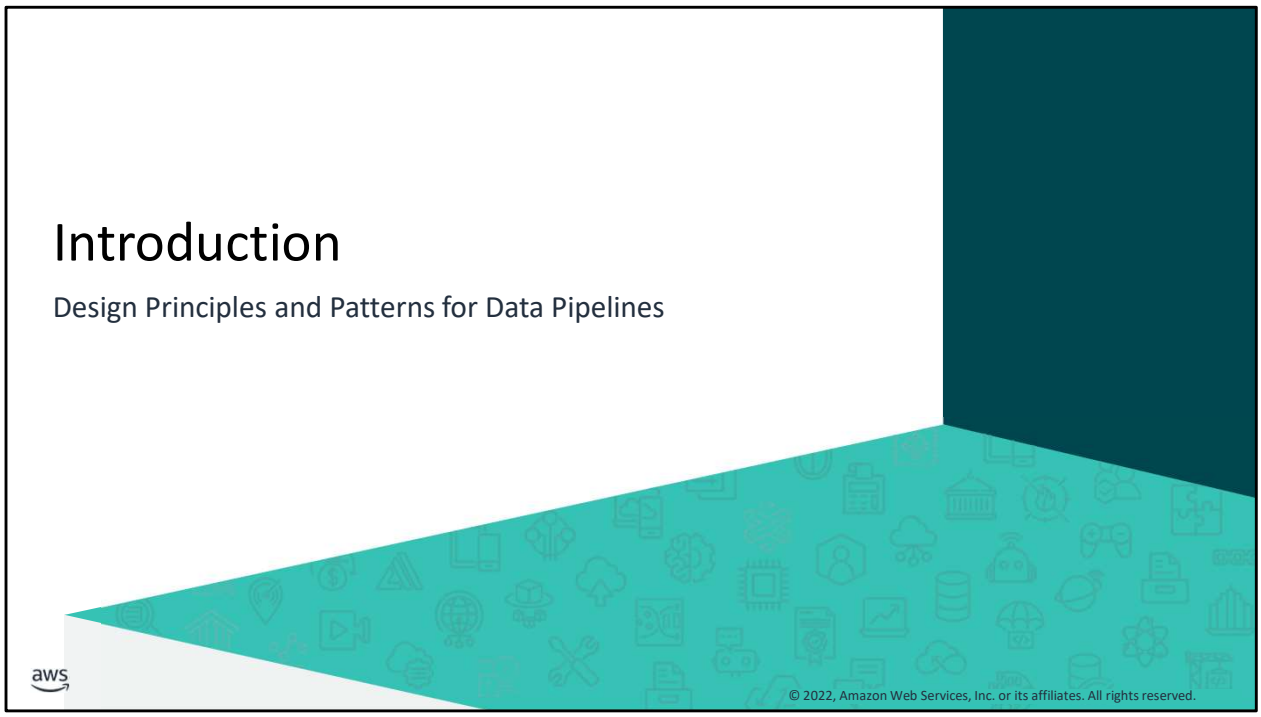
|The modern data architecture diagrams and content are derived from materials in the Well-Architected Framework as well as the whitepaper titled *Derive Insights from AWS Modern Data*. A blog post also discusses the concepts. In some references, you might still see this architecture referred to as the "Lake House," which reflects the data lake and movement in, out, and around the lake to other purpose-built data stores.

|The diagrams in this module touch on some AWS services briefly. The intent is to set initial recognition of concepts that will be built upon in modules that are focused on individual pipeline layers. One goal of this module is to get students familiar with using the Well-Architected resources to find and follow best practices. Many of the best practices that are outlined are extensible beyond AWS services and provide good checklists of how to think about designing architectures for the cloud. A secondary goal is to highlight how the concepts presented in modules 2 and 3 are reflected in the design choices that you make when actually building infrastructure.

|The reference architectures in the *Data Analytics Lens: AWS Well-Architected Framework* resource provide the source for the diagrams in this module. However, the slides have been simplified or separated into layers to help reduce the overhead of reviewing the diagrams in the lens resource.
|

|Student notes

Welcome to the Design Principles and Patterns for Data Pipelines module. This module focuses on design principles and patterns for data pipelines.



|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:

- Use the AWS Well-Architected Framework to inform the design of analytics workloads.
- Recount key milestones in the evolution of data stores and data architectures.
- Describe the components of modern data architectures on AWS.
- Cite AWS design considerations and key services for a streaming analytics pipeline.

|Slide number 3
|Instructor notes
|
|Student notes
This module builds on the introduction to data pipelines and the elements of data that were presented previously. The AWS Well-Architected Framework is a key resource, which you can use to design new pipelines or improve infrastructure that you are asked to work with. Understanding the evolution of data stores will provide context for the modern data architecture, which is presented in this module. The modern data architecture provides a framework to plan a wholistic solution that addresses the concepts of modernize, unify, and innovate, which are described in the Data-Driven Organizations module. Understanding key characteristics of streaming data pipelines will also provide context for many of the concepts and use cases presented in other modules.

Module overview

Presentation sections

- AWS Well-Architected Framework and Lenses
- The evolution of data architectures
- Modern data architecture on AWS
- Modern data architecture pipeline: Ingestion and storage
- Modern data architecture pipeline: Processing and consumption
- Streaming analytics pipeline

Activity

- Using the Well-Architected Framework

Lab

- Querying Data by Using Athena

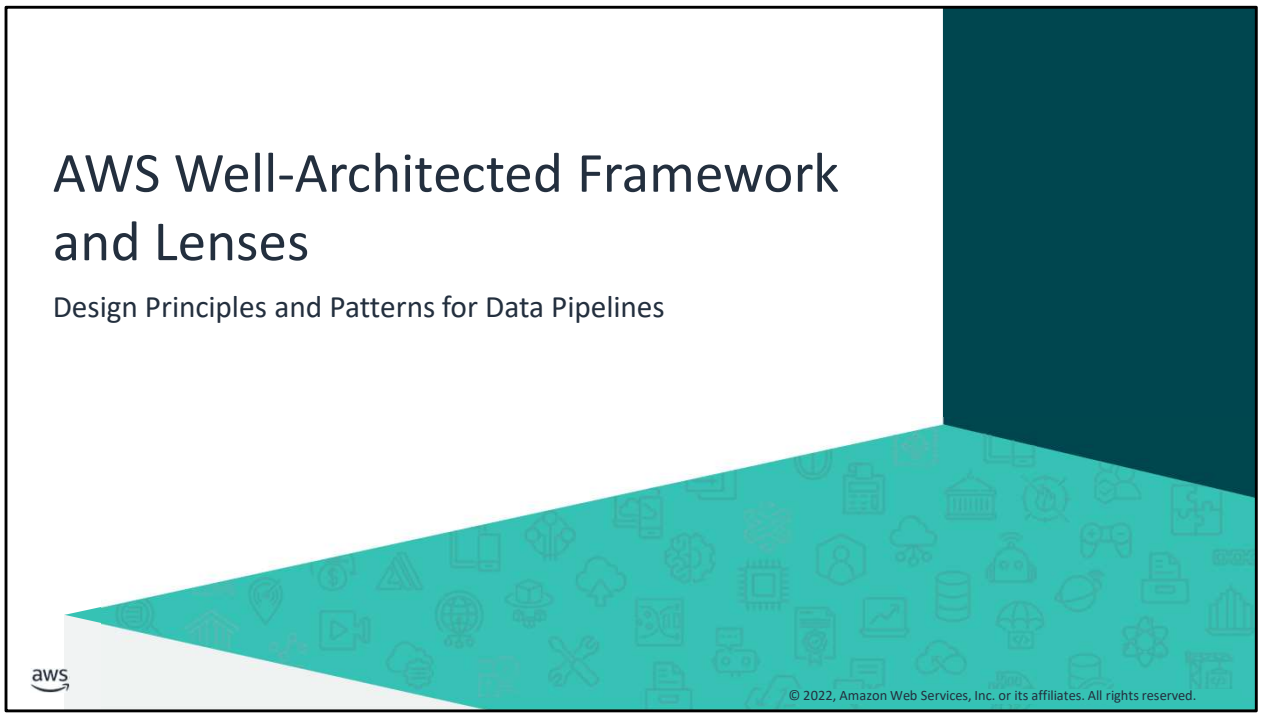
Knowledge checks

- 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 multiple sections.

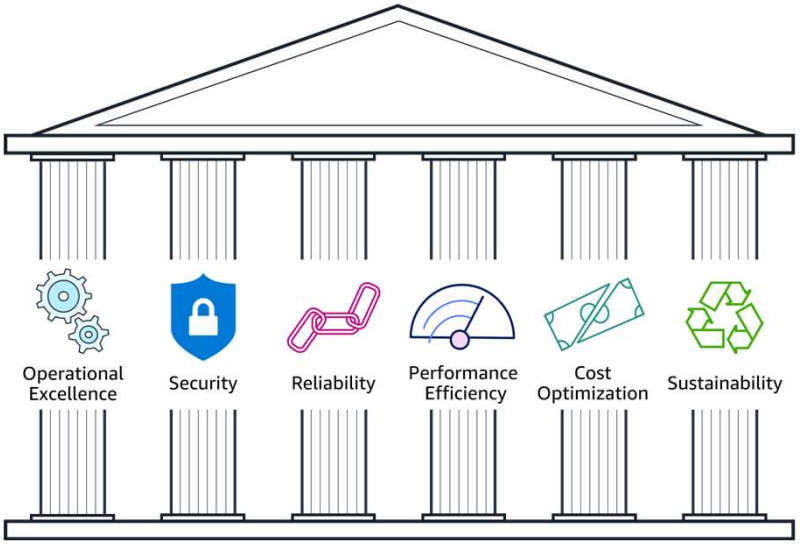
You will also complete an activity using the Well-Architected Framework.

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



|Slide number 5
|Instructor notes
|
|Student notes
This section introduces the Well-Architected Framework as a tool to design modern data architectures and analytics pipelines.

Well-Architected Framework pillars



|Slide number 6
|Instructor notes
|A suggestion for this slide is to go to the main AWS Well-Architected Framework website and show the pillars. Then, open the document for one pillar and show the design principles. In the AWS Well-Architected Framework document, go to the appendix and highlight the questions and best practices by pillar.
|
|Student notes
The AWS Academy Cloud Foundations course introduced the Well-Architected Framework. The framework is available on the AWS website, and you can find the link on the Content Resources page in your online course. You might find it helpful to open the website now to look at the way the pillars are organized.

The framework comprises six pillars: Operational Excellence, Security, Reliability, Performance Efficiency, Cost Optimization, and Sustainability. The framework provides best practices and design guidance across each pillar to help you make choices and review existing architectures.

Well-Architected Framework Lenses

Well-Architected Lenses	Data Analytics Lens	ML Lens
• Extend the AWS Well-Architected Framework guidance to specific domains • Contain insights from real-world case studies	• Provides key design elements of analytics workloads • Includes reference architectures for common scenarios	• Addresses differences between application and machine learning (ML) workloads • Provides a recommended ML lifecycle

|Slide number 7
|Instructor notes
|A suggestion for this slide is to have students go to the main AWS Well-Architected Framework site. In the AWS Well-Architected Lenses section of the page, they can open the documents for the various lens options.
|
|Student notes
The Well-Architected Framework includes lenses, which extend the guidance to specific domains. You will find a list of the lenses on the main Well-Architected Framework page. The Data Analytics Lens helps you learn the key design elements of well-architected analytics workloads and provides recommendations for improvement. The lens also includes reference architectures for common scenarios. The ML Lens calls out differences between non-ML and ML applications, and includes a recommended lifecycle for ML applications. The ML lens is discussed in the Processing Data for ML module.

Activity: Using the Well-Architected Framework

- In this activity, you will use the Data Analytics Lens from the Well-Architected Framework to identify cloud best practices that your data engineering team should follow when building their data pipelines.
- Use the detailed instructions that are provided in your online course to complete this activity.

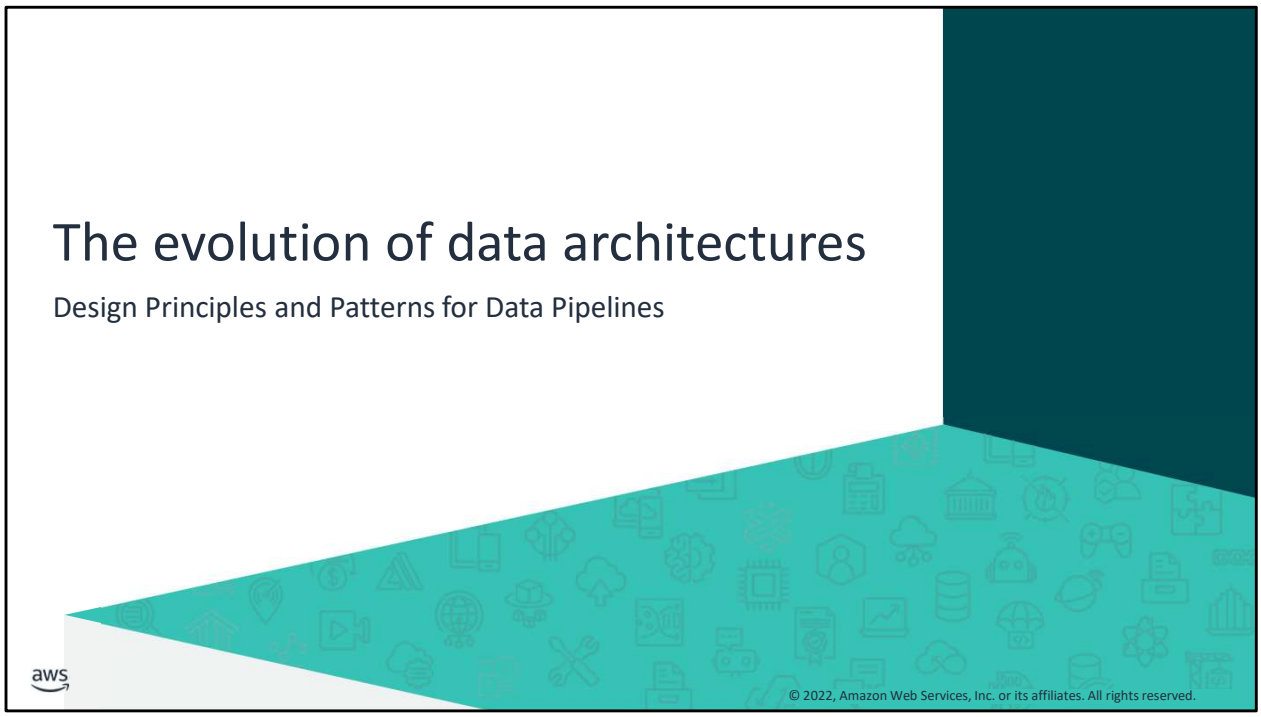
|Slide number 8
|Instructor notes
|The first page of the WellArchitectedActivity.pdf, which is available in the instructor files section of the course, provides more detailed instructions to be shared with students. The second page provides suggested answers. The directions suggest using discussion threads in the online course to capture student answers so that they can see and comment on each other's findings.
|
|Student notes
You might perform this activity with your class or on your own. Follow the instructions that your instructor provides or that are located in your online course for this module.

Key takeaways: AWS Well-Architected Framework and Lenses

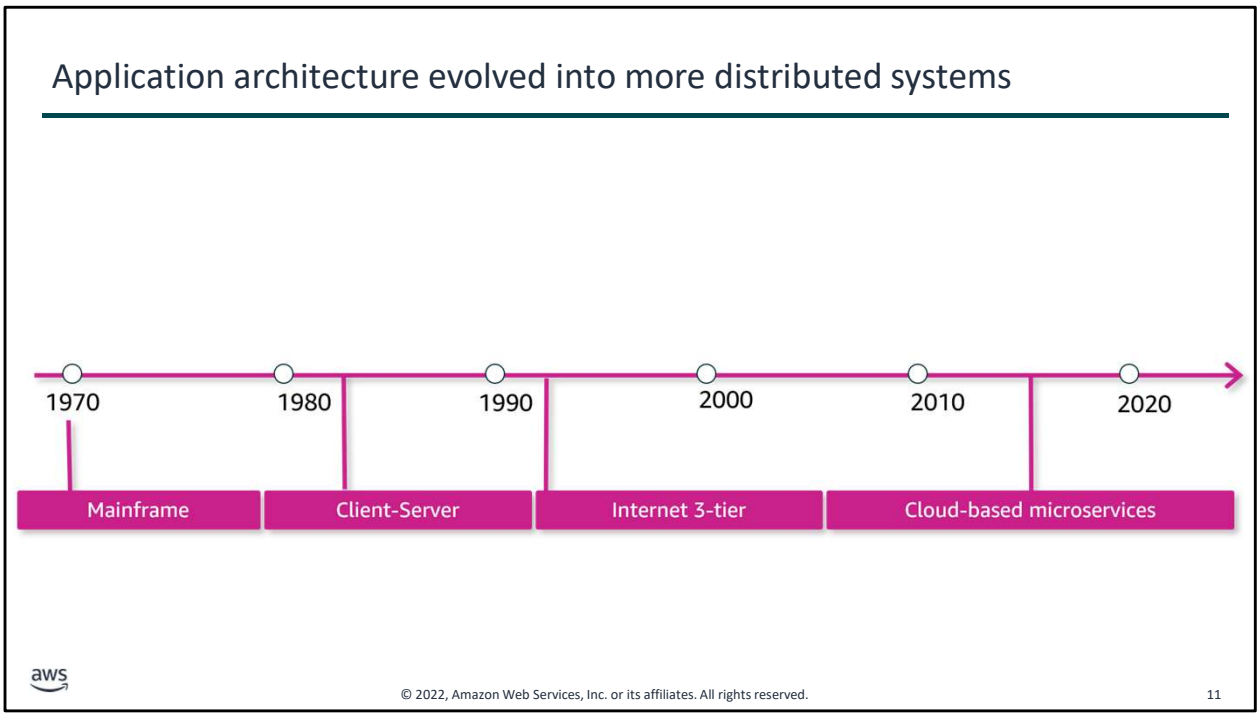
- The Well-Architected Framework provides best practices and design guidance across six pillars.
- The Well-Architected Framework Lenses extend guidance to focus on specific domains.
- The Data Analytics Lens provides guidance that helps with design decisions related to the elements of data (volume, velocity, variety, veracity, and value).

|Slide number 9
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

- The Well-Architected Framework provides best practices and design guidance, and its lenses extend guidance to focus on specific domains.
- You can use the Data Analytics Lens to guide your design of data pipelines to suit the characteristics of the data that you need to process.



|Slide number 10
|Instructor notes
|
|Student notes
This section describes the evolution of data stores from 1970 to present day.



|Slide number 11

|Instructor notes

|You can spend more or less time on this information depending on the knowledge level and technical background of students. For students who have experienced pre-internet applications, you might ask for examples of what they remember about the challenges of those earlier days of computing.

|

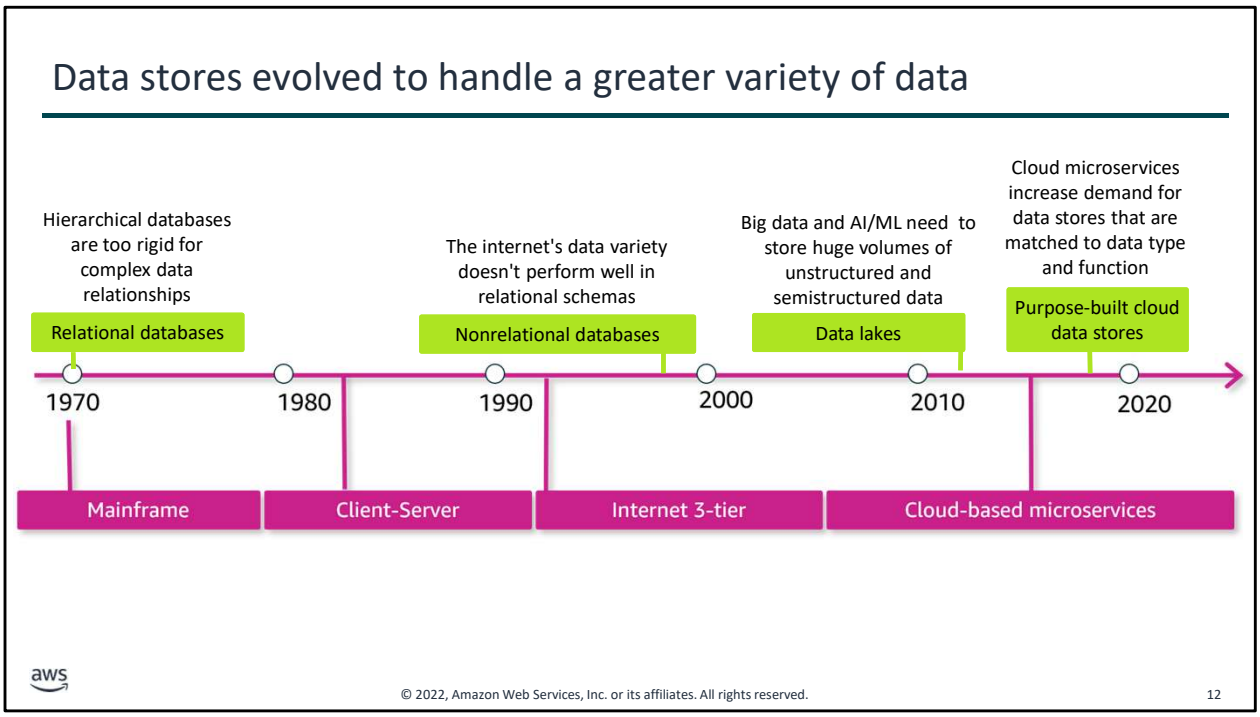
|Student notes

Application architectures have changed quite a bit in recent decades. Originally, companies would use on-premises mainframes to handle their critical applications. These mainframes combined all aspects of an application—compute and storage—and ran for years without interruption.

Applications then were split into pieces using a client-server architecture. The server would respond to requests from various clients, which allowed for more distributed systems. The clients and servers could be on the same computer, but the separation allowed for more scalable systems when needed. Clients and servers could be split up so they would not compete for resources.

With the rise of the internet, the three-tier architecture became prominent. Applications were split into groups by function: a presentation tier served as the user interface, an application tier handled the business logic and processing, and a data tier provided long-term, persistent storage. Again this increased the scalability of applications because each tier could be scaled independently.

Each of these moves focused on splitting up your application to improve scaling and resiliency. The most recent architectural trend, which aligns well to cloud best practices, is to move to microservices. With microservices, you split your application into different services based on functionality. For example, rather than having one application that handles both your inventory data and order history data, you might split those into two separate services that are focused on their domain. This split allows the two services to scale independently and provides more agility between development teams.



|Slide number 12

|Instructor notes

|The callouts are animated so that each problem description appears on click and the green callout follows after .5 seconds.

|Student notes

Parallel to the evolution of application architecture is the evolution of how we store and access data. This evolution is worth a quick review to provide context to the tools and best practices that apply to modern data architectures.

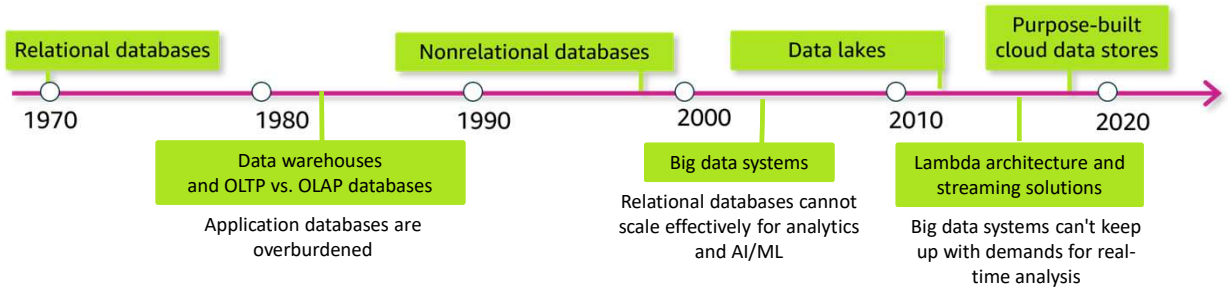
Relational databases replaced hierarchical ones, which had limited abilities to define relationships among data. Relational databases have been around since 1970 and continue to be the workhorse of many applications.

As the internet era took off, organizations started to get more varied types of data, which they wanted to be able to analyze. This new variety of data was less structured and needed less rigid rules. This led to the development of nonrelational databases in the late 90s.

With the growth of big data and artificial intelligence and machine learning (AI/ML) applications, organizations realized that they wanted to collect as much data as possible for potential use, without the rigors of transforming it into a formal structure. Organizations wanted to simplify collection and querying of data in its most raw form. With a data lake, raw data could be loaded into the data store, where it could be held for a variety of use cases.

Cloud computing gave organizations the freedom to scale data stores up and down based on actual usage. Coupled with a move toward microservices, the flexibility of the cloud made it attractive to connect different application components to different databases, rather than relying on a single multipurpose one. Purpose-built data stores emerged to let developers optimize the storage that is connected to a component to match the data type and processing of that component. For example, developers might use a ledger database for financial transactions.

Data architectures evolved to handle volume and velocity

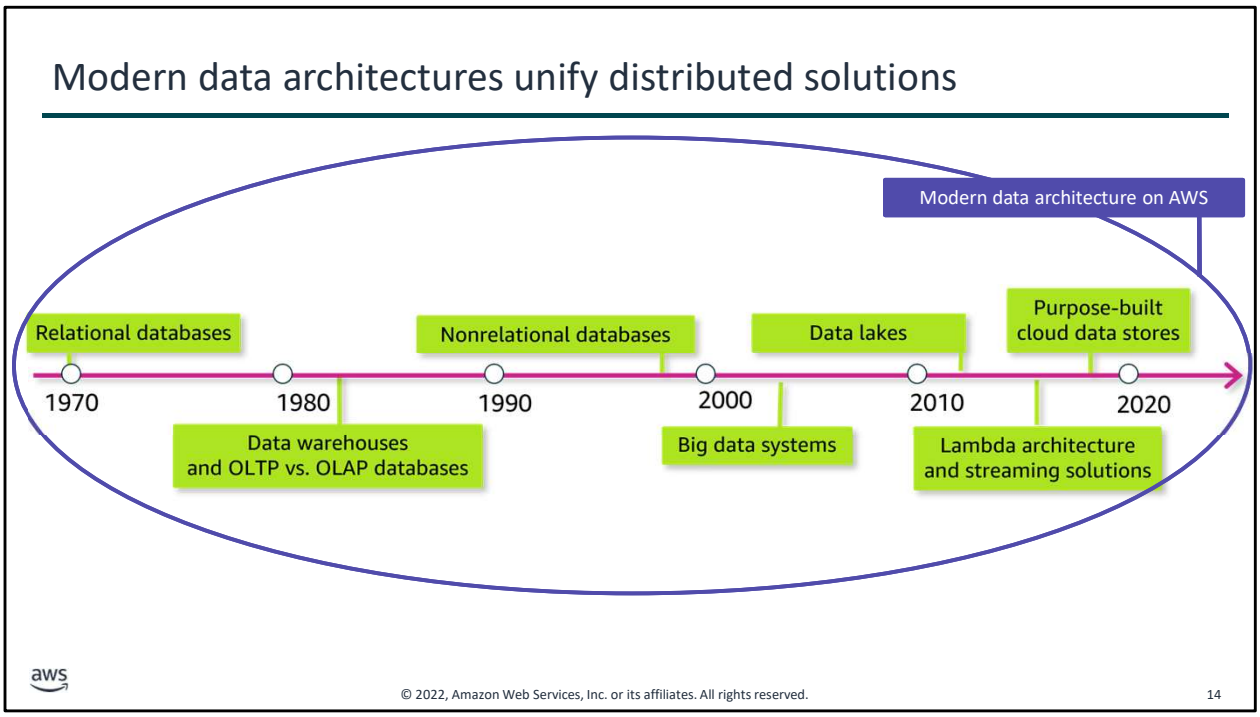


|Slide number 13
|Instructor notes
| The animation on this slide works the same as the prior one. The callouts are animated so that each problem description appears on click and the green callout follows after .5 seconds.
|It might be worth pointing out to students that the AWS Lambda service and Lambda functions are not related to the lambda architecture which is a standard in big data processing.
|This course doesn't cover the lambda architecture as its own topic. However, if it fits with your curriculum, the book *Big Data: Principles and Best Practices of Scalable Realtime Data Systems* by Nathan Marz and James Warren covers it in great detail.
|
|Student notes
While data stores were evolving to handle different types of data, data architectures were also evolving to handle a continued increase in volume and velocity.

In the 1980s, data warehouses evolved as a way to separate operational reporting, which requires read-heavy querying across a full dataset, from the application's transactional database, which is focused on fast reads and writes in a much smaller set of records. Data warehouses are still relational databases, but they have been optimized for reporting and analytics. Online analytical processing (OLAP) databases are optimized for reporting whereas online transaction processing (OLTP) databases are designed for transactions, such as creating an order or making an ATM withdrawal. The extract, transform, and load (ETL) process was introduced to extract data from OLTP databases, transform it, and then load it into the data warehouse.

As noted earlier, the rise of the internet brought new data to be collected and analyzed. Even with a data warehouse dedicated to analysis, keeping up with the volume and velocity of incoming data created database bottlenecks. Administrators could scale vertically—that is, increase the size and speed of the database—but there wasn’t an easy way to scale horizontally—that is, to distribute the load across multiple databases. Big data systems or frameworks addressed this shortcoming in the 2000s. Big data frameworks were designed to distribute data across multiple nodes and handle any failures automatically. These frameworks also allowed the big data systems to handle many ETL transformations, which helped to increase the speed with which analysis could be done.

Work was still mostly done by batching at some scheduled interval. Batch processing created a lag between the arrival of new data and its being included in analytics results. The continually increasing volume and velocity of new data meant that the time between batches created an increasingly large gap in the timeliness of data being processed. The volume and velocity also prevented more real-time analysis and decisions. As noted earlier, data has its greatest value when it can be analyzed as close as possible to when it was created. But as you also learned, the most valuable, predictive insights are more difficult to derive. To find a balance between value and complexity, Nathan Marz proposed the lambda architecture—an approach which combines the use of batch processing with stream processing to support close-to-real-time insights. This approach has become a standard way to process big data.



|Slide number 14

|Instructor notes

|The animation adds a circle around all elements in the slide. You might pause and ask the students which aspects they think are still part of modern architectures before revealing that all of them are still relevant. Circle reveals on click.

|

|Student notes

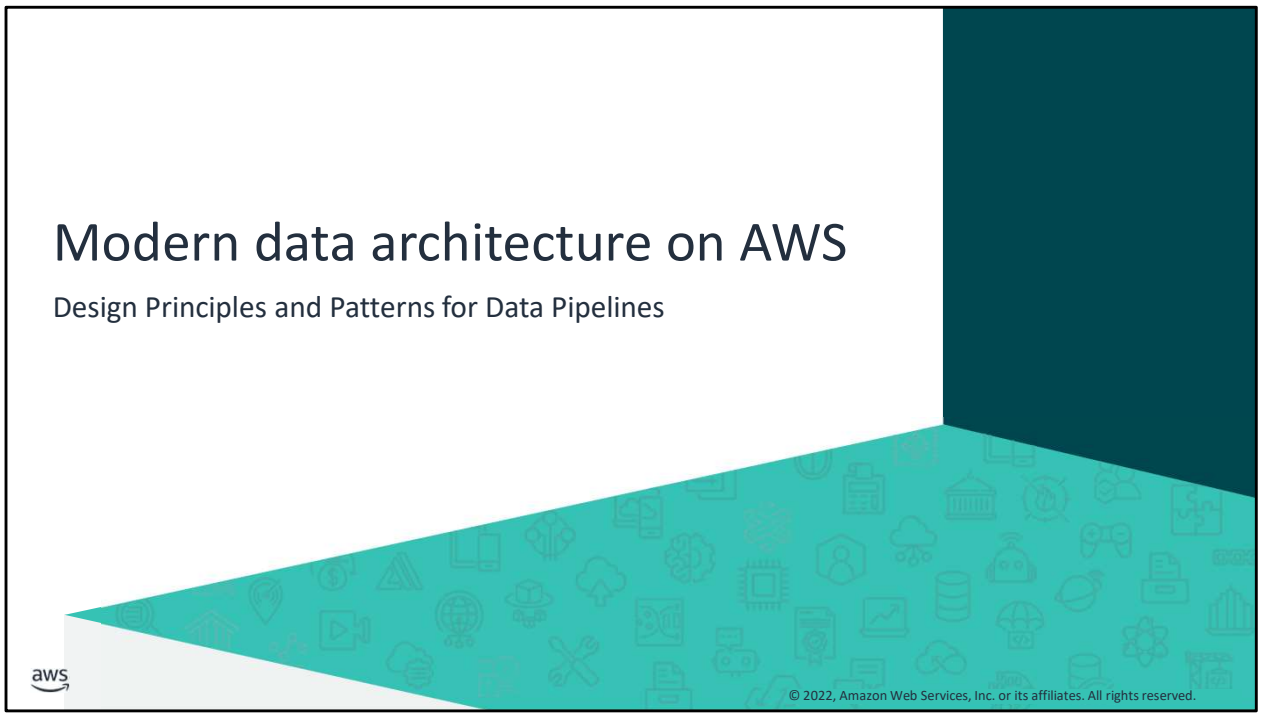
So, which of these data stores or data architectures is the best one for your data pipeline? The reality is that a modern architecture might include all of these elements. The key to a modern data architecture is to apply the three-pronged strategy that you learned about earlier. *Modernize* the technology that you are using. *Unify* your data sources to create a single source of truth that can be accessed and used across the organization. And *innovate* to get higher value analysis from the data that you have.

Key takeaways: The evolution of data architectures

- Data stores and architectures evolved to adapt to increasing demands of data volume, variety, and velocity.
- Modern data architectures continue to use different types of data stores to suit different use cases.
- The goal of the modern architecture is to unify disparate sources to maintain a single source of truth.

|Slide number 15
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

- Data stores and architectures evolved to keep up with continually increasing volume, variety, and velocity of the data being generated.
- Modern data architectures will continue to use different types of data stores to provide the best fit for each use case, but they need to find a way to unify disparate sources to maintain a single source of truth.

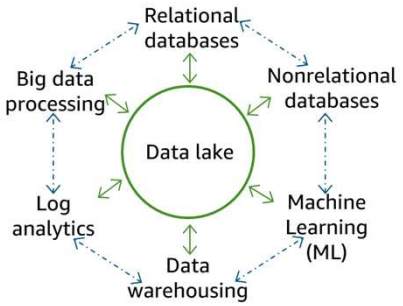


|Slide number 16
|Instructor notes
|This section aligns to the modern data architecture resources that were noted at the start of this module. This section attempts to pull key concepts from each of them. You might compare across the three presentations of this architecture and decide if you would prefer to use one of those online versions rather than the slides.
|
|Student notes
This section provides an overview of the components of a modern data architecture.

Modern data architecture

Key design considerations

- Scalable data lake
- Performant and cost-effective components
- Seamless data movement
- Unified governance



|Slide number 17
|Instructor notes
|
|Student notes

The goal of the modern data architecture is to store data in a centralized location and make it available to all consumers to perform analytics and run AI/ML applications. But that doesn't mean that you have one data store—only that you have a single source of truth. As illustrated on the slide, a data lake provides the centralized repository of data and is integrated with the other types of data stores and data processing systems that were described in the previous section. Data might be queried directly from the lake, or it might be moved to and from other purpose-built tools for processing.

The diagram on this slide provides a conceptual look at the components of a modern data architecture and how they are connected to each other. The diagram doesn't reflect the actual architecture that accommodates these connections.

The movement of data among the lake and other integrated services falls into three general types: outside in, inside out, and around the perimeter.

Outside in is when an organization that stores data in purpose-built data stores, such as a data warehouse or a database, moves data into the lake to run analysis on it. For example, they copy query results for product sales in a given region from their data warehouse into their data lake to run product recommendation algorithms against a larger dataset by using ML.

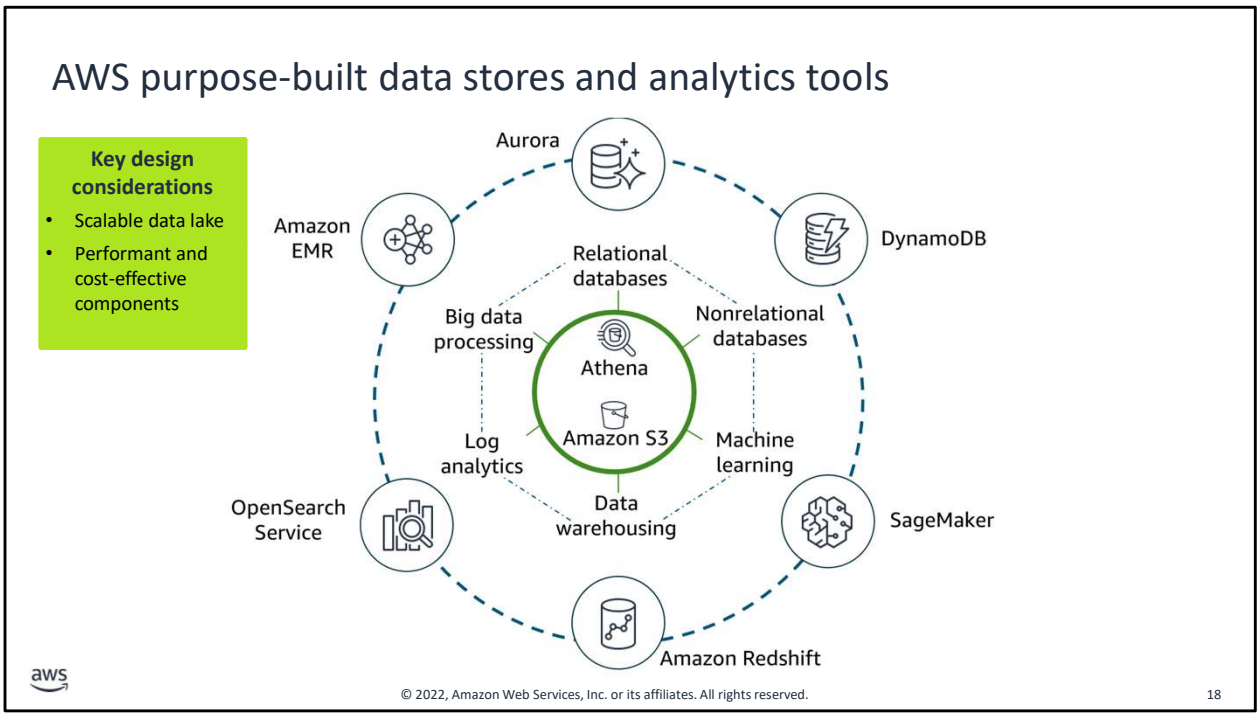
Inside out is when an organization stores data in a data lake and then moves a portion of that data to a purpose-built data store to do additional ML or analytics. For example, they collect clickstream data from web applications directly into a data lake and then move a portion of that data out to a data warehouse for daily reporting.

Around the perimeter is when an organization moves data directly between the other data store components that are integrated with the data lake without needing to access the data lake. For example, they might copy the product catalog data that is stored in their database to their search service to make it easier to look through their product catalog and offload the search queries from the database.

The modern data architecture approach addresses the strategies of modernizing, unifying, and innovating with your data architecture, but it also introduces a lot of complexity to constructing your data pipeline. The main challenge with a data lake architecture is that raw data is stored with no oversight of the contents. For a data lake to make data usable, it needs to have defined mechanisms to catalog and secure data. Without these elements, data cannot be found or trusted, which results in a “data swamp."

To build a successful modern data architecture, your design must incorporate the following features:

- The data lake should be able to scale easily as data grows. Use a scalable, durable data store that supports multiple ways to bring data in. For each component, select scalable services that balance the fastest performance at the lowest cost for the use case. Choose purpose-built tools. And continually assess options that could increase performance or reduce costs.
- Your design should also support seamless data movement into and out of the data lake and around the perimeter. And whenever possible, provide direct access to the data.
- Your design must also ensure consistency of data across all components of the architecture. As data in your various data stores and data lake continues to grow, it becomes more difficult to move all of that data around securely and in a governed way. This is referred to as *data gravity*. In this architecture, it’s critical to have robust mechanisms for authorization and auditing with a centralized location to define policies and enforce them.



|Slide number 18
|Instructor notes
|
|Student notes
In this version of the conceptual diagram, you see AWS services that map to the conceptual elements of the modern data architecture and support its key design considerations. You will learn more about each of these components throughout the course and will see how these connections are reflected in reference architectures that support this conceptual view.

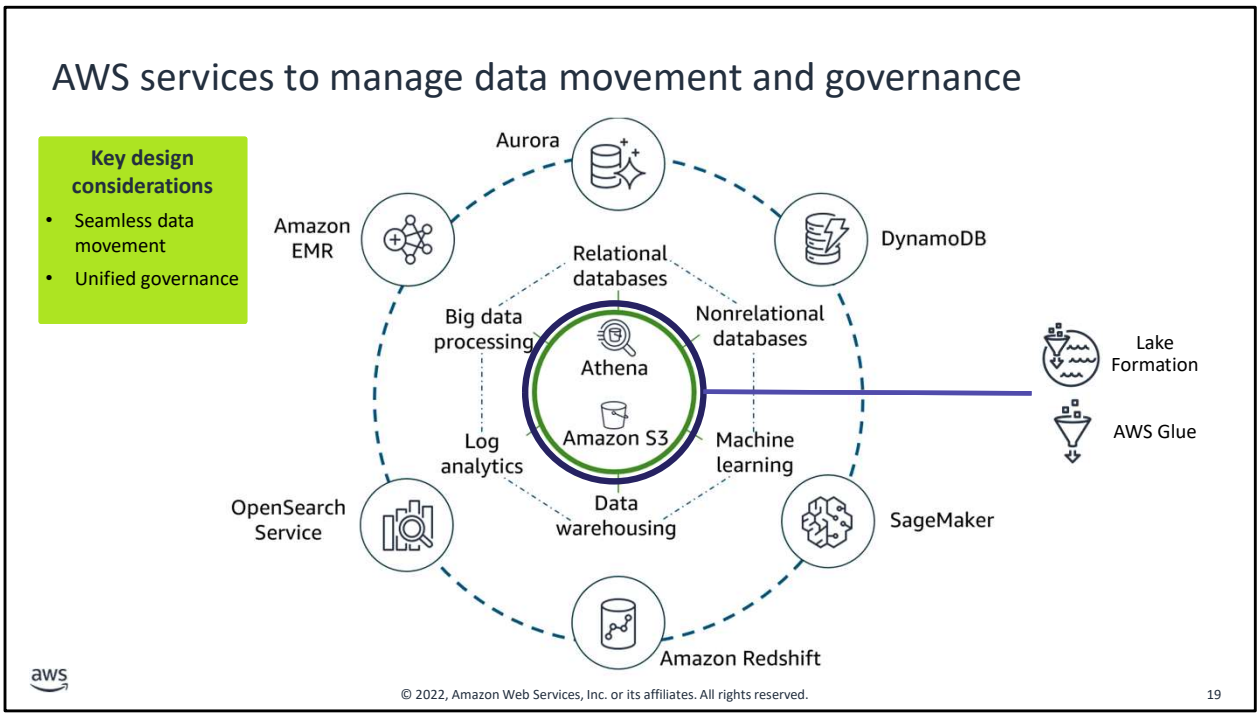
Amazon Simple Storage Service (Amazon S3) provides storage for structured and unstructured data and is the storage service of choice to build a data lake on AWS. With Amazon S3, you can cost-effectively build and scale a data lake of any size in a secure environment with high durability. With Amazon S3, you can also use native AWS services to run big data analytics and AI/ML applications.

Amazon Athena is shown directly on the data lake to illustrate that it provides interactive querying of data directly in Amazon S3.

The architecture illustrates the following other AWS purpose-built services that integrate with Amazon S3 and map to each component that was described on the previous slide:

- Amazon Redshift is a fully managed data warehouse service.
- Amazon OpenSearch Service is a purpose-built data store and search engine that is optimized for real-time analytics, including log analytics.
- Amazon EMR provides big data processing and simplifies some of the most complex elements of setting up big data processing.
- Amazon Aurora provides a relational database engine that was built for the cloud.
- Amazon DynamoDB is a fully managed nonrelational database that is designed to run high-performance applications.
- Amazon SageMaker is an AI/ML service that democratizes access to ML processing.

To meet the goal of seamless data movement, Athena and Amazon Redshift both support federated queries and the ability to run queries across different data stores without needing to move the data between stores to perform the queries.



|Slide number 19

|Instructor notes

|The Querying Data by Using Athena lab uses AWS Glue that is suggested to be used as part of this module. AWS Glue is also discussed in greater detail in the Ingesting by Batch or by Stream module.

|

|Student notes

Two additional services provide movement and governance for this architecture. AWS Glue facilitates data movement and transformation between data stores, which helps to prepare data for analytics and ML much more quickly than traditional ETL methods.

AWS Lake Formation was built to make it easier to manage time-consuming tasks that are related to loading, monitoring, and managing data lakes. Lake Formation helps to catalog data and classify and secure it for different types of access.

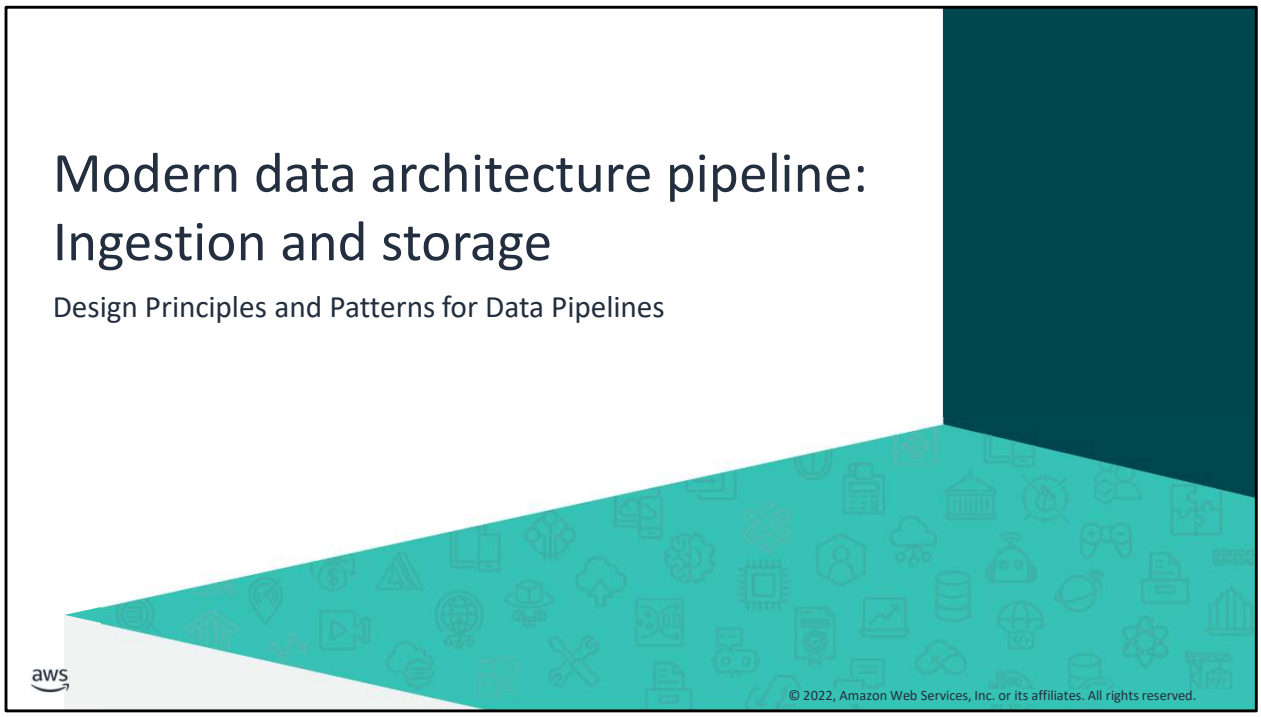
This conceptual view represents a lot of different infrastructure and pipeline components and isn’t meant to convey a detailed architectural view of what you build or and how the integrations are achieved. But it's a good reference point to keep in mind when you think about the goals of how you store and make data available within your organization.

Key takeaways: Modern data architecture on AWS

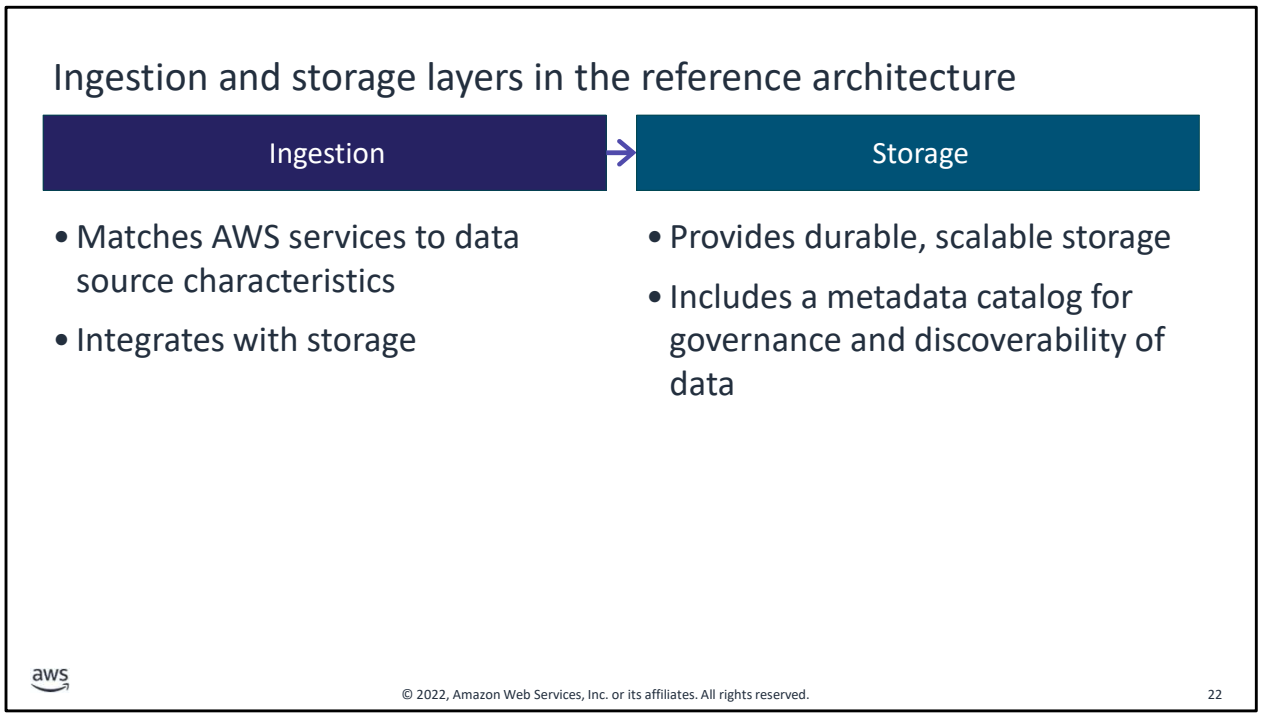
- A centralized data lake provides data that can be available to all consumers.
- Purpose-built data stores and processing tools integrate with the lake to read and write data.
- The architecture supports three types of data movement: outside in, inside out, and around the perimeter.
- AWS services that are key to seamless access to a centralized lake include Amazon S3, Lake Formation, and AWS Glue.

|Slide number 20
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

- A data lake provides centralized storage and is integrated with purpose-built data stores and processing tools.
- Data moves into the data lake (outside in), from the data lake to other stores (inside out), and might also move directly between purpose-built stores (around the perimeter).
- Amazon S3 provides the data lake, and Lake Formation and AWS Glue help to provide seamless access.

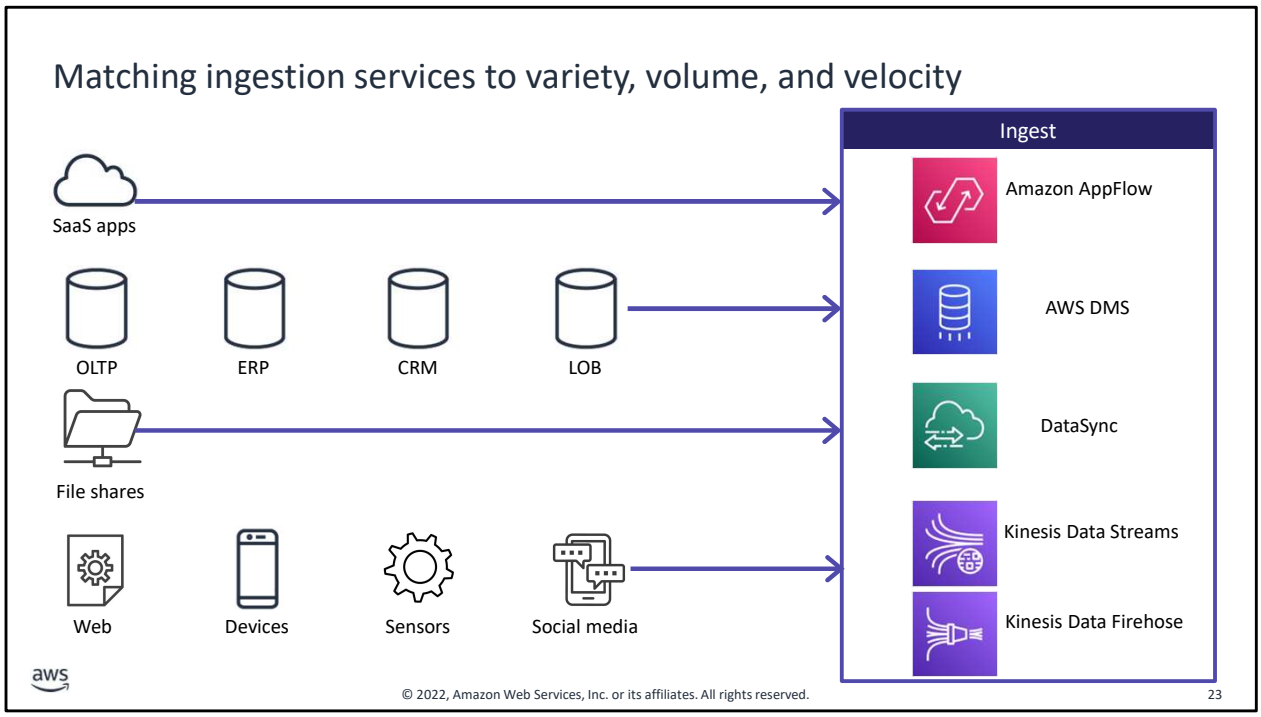


|Slide number 21
|Instructor notes
|Suggest that students open the *Data Analytics Lens: AWS Well-Architected Framework* resource and go to **Scenarios** > **Modern data architecture** > **Reference architecture**. The direct link is provided in the Content Resources section of the course.
|
|Student notes
The next two sections highlight aspects of how the modern data architecture is reflected in the reference architecture that is presented in the Well-Architected Framework. This section focuses on ingesting and storing data. You might find it helpful to open *Data Analytics Lens: AWS Well-Architected Framework* in a browser window and go to the Modern Data Architecture section for additional details. The slides provide a high-level summary. The direct link is provided in the course resources.



|Slide number 22
|Instructor notes
|
|Student notes
The layers of the reference architecture align to the pipeline layers that were introduced earlier in the course. The ingestion and storage layers make data available for processing and analysis.

In the ingestion layer, you can match AWS services to source data types so that you can more easily bring in different types of data. Most services also integrate directly with storage layer components. The storage layer has two sublayers. One provides storage, and the other provides the catalog that holds the metadata about the datasets that are being stored. The next few slides look at the ingestion and storage layers in more detail.



|Slide number 23

|Instructor notes

|The reference architecture shows all of these components but doesn't specifically highlight the connection between source type and the service that is used to ingest it. If you want to teach directly from the Well-Architected Framework image, you could just annotate the connections that are shown on this slide.

|The Ingesting by Batch or by Stream module will revisit ingestion services.

|

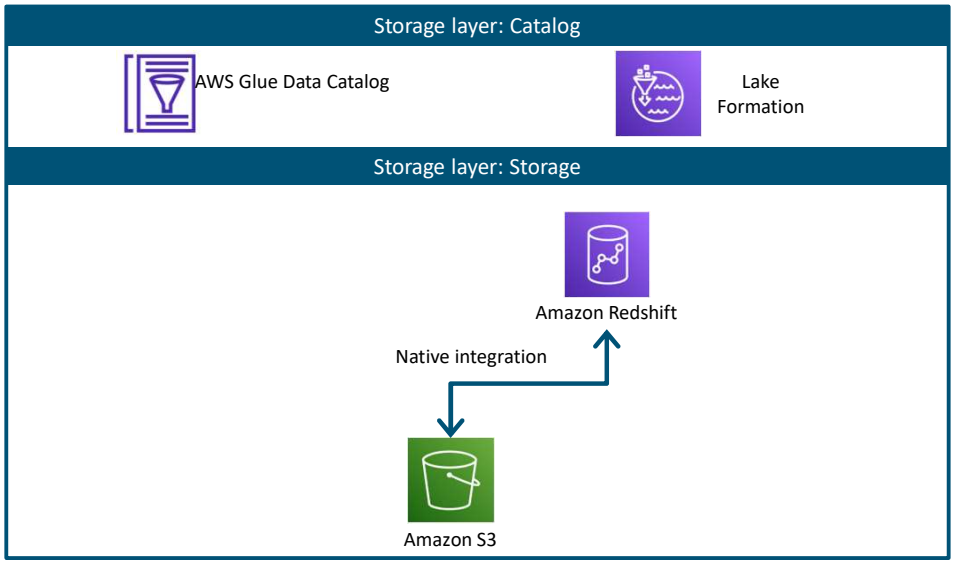
|Student notes

The ingestion layer uses individual purpose-built AWS services to match the unique connectivity, data format, data structure, and data velocity requirements of source types and to deliver them to the storage layer components. The services include the following:

- Amazon AppFlow can ingest from Software as a service (SaaS) applications, such as Salesforce or Zendesk.
- AWS Database Migration Service (AWS DMS) can ingest from operational databases like online transaction processing (OLTP), enterprise resource planning (ERP), customer relationship management (CRM) and line of business (LOB) databases.
- AWS DataSync can ingest from file shares.
- Amazon Kinesis Data Streams and Amazon Kinesis Data Firehose can ingest from streaming data sources.

In the Ingesting by Batch or by Stream module, you will learn more about these services and how they support ingestion of different types of data.

Modern data architecture storage layer

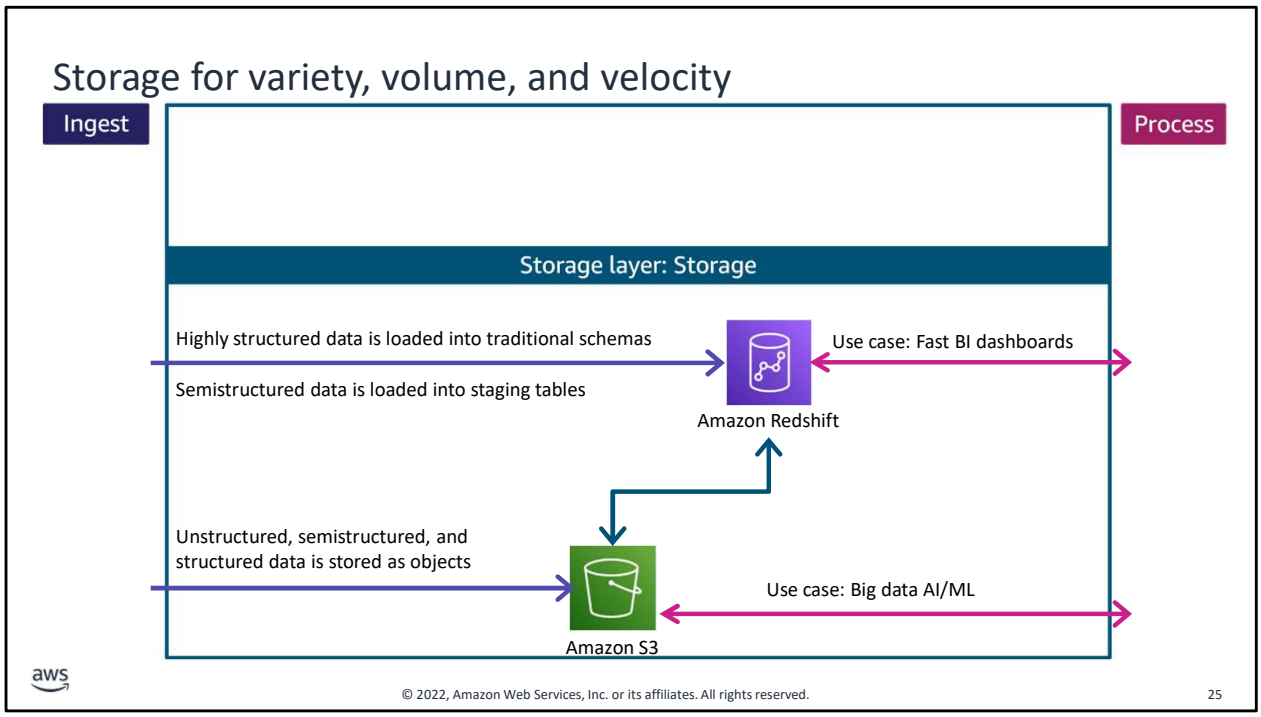


|Slide number 24
|Instructor notes
|The Storing and Organizing Large Datasets module will revisit storage details. This section serves as the introduction and provides scaffolding for the details that will follow in that module.
|
|Student notes
Data is ingested into the storage layer.

The data storage layer is responsible for providing 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 is responsible for storing 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, 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.

The next slides look at the relationships between the data lake, data warehouse, and data catalog.



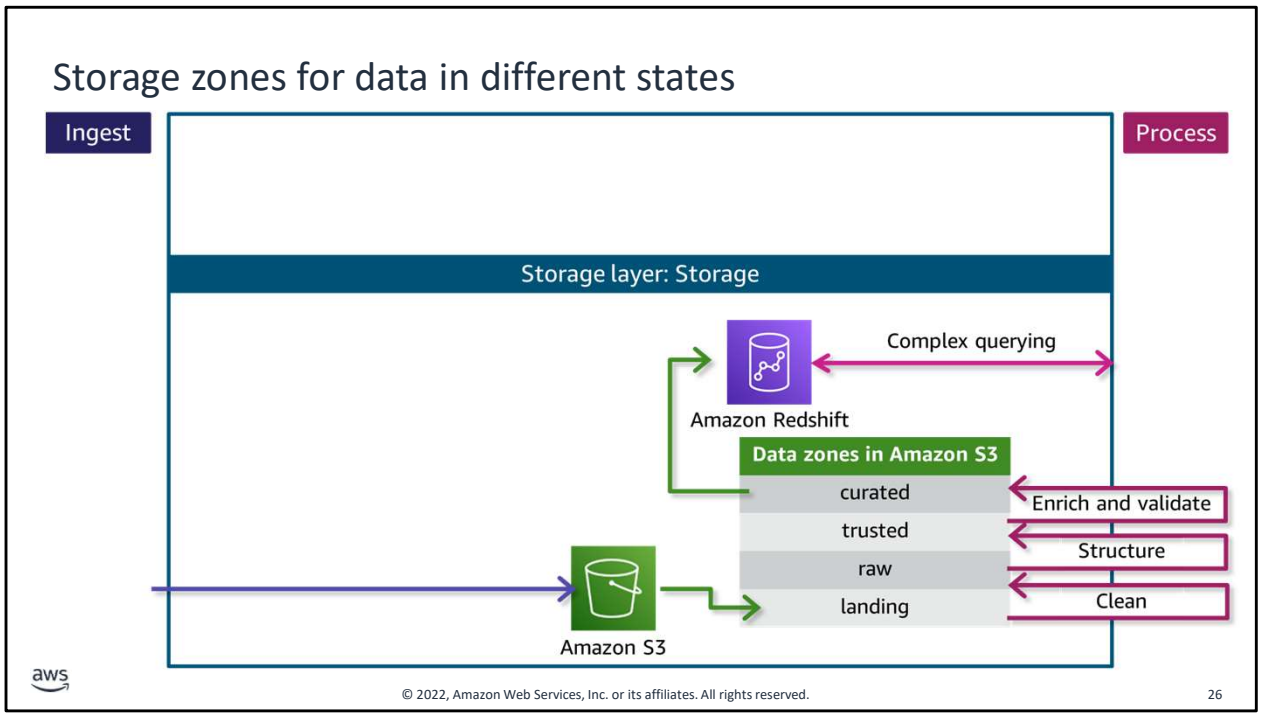
|Slide number 25
|Instructor notes
|
|Student notes
Highly structured data in the warehouse typically powers interactive queries and highly trusted, fast business intelligence (BI) dashboards. A modern cloud-native data warehouse, such as Amazon Redshift, provides low-latency turnaround of complex SQL queries.

Data in a warehouse is normally ingested from highly structured sources, such as transactional systems and relational databases, on a regular cadence. Amazon Redshift also supports ingestion of semistructured data into staging tables. To be accessed for analytics, data in the warehouse must be highly trusted and structured into traditional schemas.

Data in the data lake typically drives ML, data science, and big data processing use cases. The data lake enables analysis of diverse datasets by using diverse methods.

The Amazon S3 data lake supports storage of data in structured, semistructured, and unstructured formats and can scale automatically. Data can be ingested into the lake without first defining a schema, so this speeds the time needed to ingest the data and make it available for exploration. Both structured and unstructured data is stored as S3 objects.

The native integration between Amazon S3 and Amazon Redshift means that you can ingest data as is into Amazon S3 and then prepare it for the data warehouse as needed. This lets you offload historical data from warehouse storage into a more cost-efficient storage tier in Amazon S3, which reduces the cost of storage.



|Slide number 26
|Instructor notes
|
|Student notes
This slide looks closer at how data can be ingested as is into Amazon S3 and then prepared for other use cases, including storage in the data warehouse, by using the concept of *storage zones*.

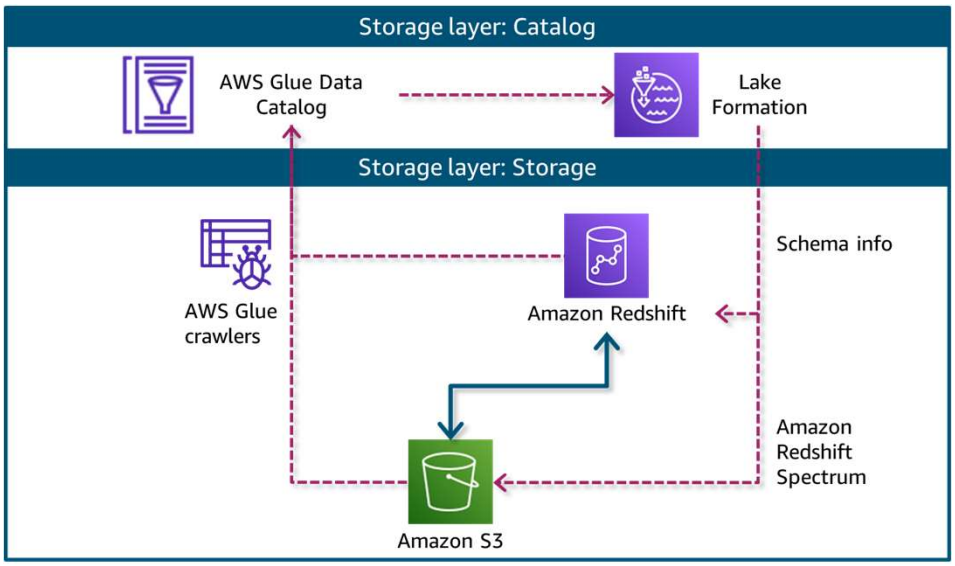
Each zone represents a different state of data and is represented by a bucket or prefix in Amazon S3. Zones include landing, raw, trusted, and curated. Data might pass through each zone as it is cleansed, normalized, augmented, or transformed in some other way. Transformed data is saved into a zone that matches its readiness to be consumed.

Data being ingested into the Amazon S3 data lake arrives at the landing zone, where it is first cleaned and stored into the raw zone for permanent storage. Because data that is destined for the data warehouse needs to be highly trusted and conformed to a schema, the data needs to be processed further.

Additional transformations would include applying the schema and partitioning (structuring) as well as other transformations that are required to make the data conform to requirements that are established for the trusted zone. Finally, the processing layer prepares the data for the curated zone by modeling and augmenting it to be joined with other datasets (enrichment) and then stores the transformed, validated data in the curated layer. Datasets from the curated layer are ready to be ingested into the data warehouse to make them available for low-latency access or complex SQL querying.

This process reflects the iterative nature of the pipeline processing that you learned about earlier in the course. This process also highlights how smaller pipelines are at work within the larger architecture. The Ingesting and Preparing Data module discusses these data transformation processes in more detail.

Catalog layer for governance and discoverability



|Slide number 27
|Instructor notes
|
|Student notes
This slide illustrates how the catalog layer supports governing access and finding data in the lake or warehouse.

The AWS Glue service helps you simplify data movement and transformation within your pipeline. You can use AWS Glue to generate schemas for your data sources, which are then stored in an AWS Glue Data Catalog with other metadata about your sources. Another feature—AWS Glue data crawlers—can automatically discover schemas and metadata about data in the data lake and data warehouse and save those updates to the Data Catalog. You will use AWS Glue in the lab for this module, and you will learn more about AWS Glue in the Ingesting by Batch or by Stream module.

The AWS Glue Data Catalog can also send the schema and metadata information to Lake Formation, as depicted in this slide. The Lake Formation service is designed to simplify setting up, accessing, and securing data lakes. Lake Formation provides the data lake administrator with a central place to set up granular permissions for databases and tables that are hosted in the data lake.

In this architecture, Lake Formation provides the central catalog for all datasets that are hosted in Amazon S3 and Amazon Redshift. The catalog includes both business attributes, such as data owner and column business definitions, and technical metadata, such as versioned table schemas and timestamps.

By using Amazon Redshift Spectrum, a feature of Amazon Redshift, users 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.

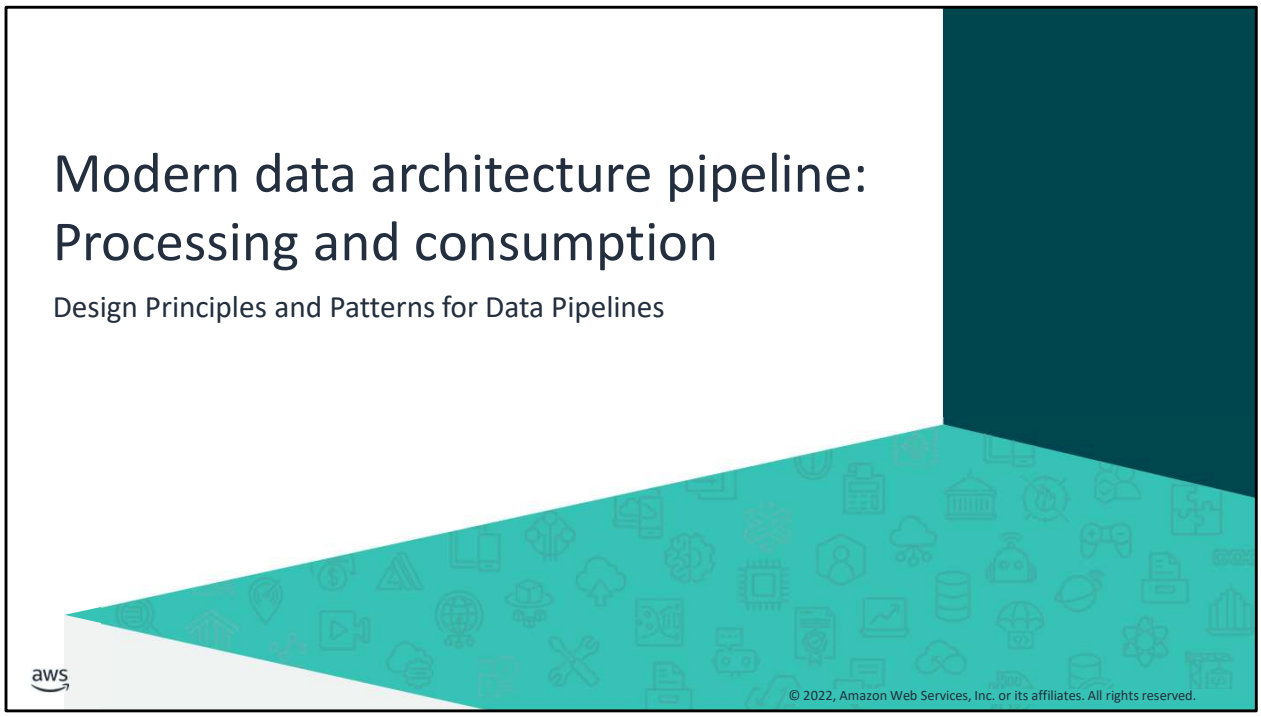
The integration between AWS Glue, Lake Formation, Amazon Redshift, and Amazon S3 makes it easier to manage and use data that is ingested into your pipeline.

Key takeaways: Modern data architecture pipeline: Ingestion and storage

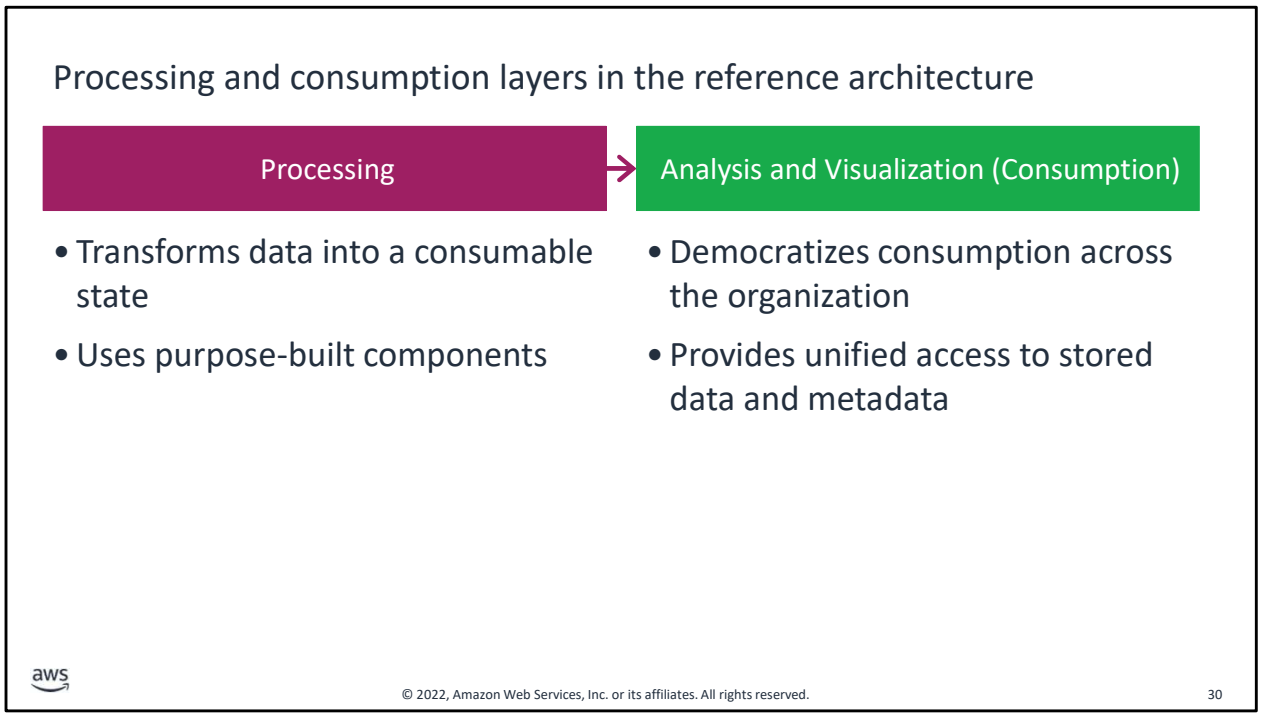
- The AWS modern data architecture uses purpose-built tools to ingest data based on characteristics of the data.
- The storage layer uses Amazon Redshift as its data warehouse and Amazon S3 for its data lake.
- The Amazon S3 data lake uses prefixes or individual buckets as zones to organize data in different states, from landing to curated.
- AWS Glue and Lake Formation are used in a catalog layer to store metadata.
- With the catalog, Amazon Redshift Spectrum can query data in Amazon S3 directly.

|Slide number 28
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

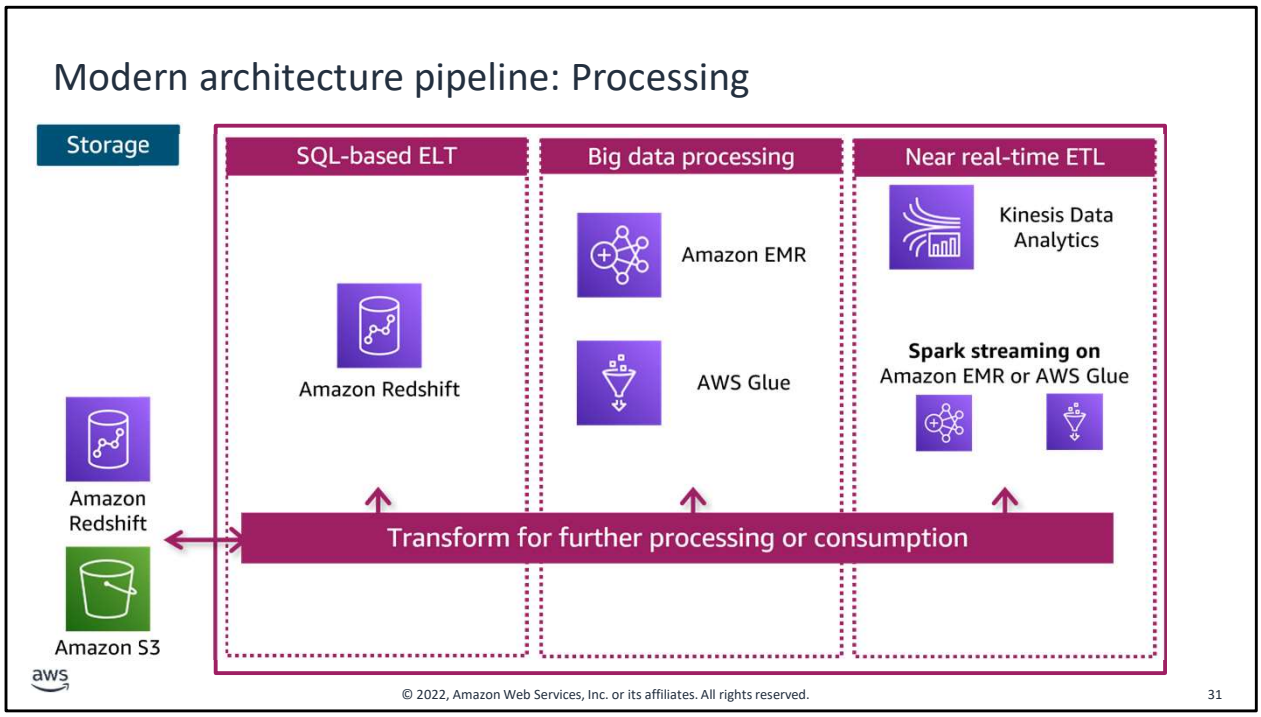
- The AWS modern data architecture uses purpose-built tools to ingest data based on characteristics of the data.
- The storage layer includes two layers. The first is a storage layer that uses Amazon Redshift as its data warehouse and Amazon S3 for its data lake. The second is a catalog layer that uses AWS Glue and Lake Formation.
- The catalog maintains metadata and also provides schemas-on-read, which Redshift Spectrum uses to read data directly from Amazon S3.
- Normally, data within the lake is segmented into zones to represent different states of processing. Data arrives in a landing zone and might move up to curated as it's processed for different types of consumption. You create zones by using prefixes or bucket names for each zone in Amazon S3.



|Slide number 29
|Instructor notes
|Suggest that students open the *Data Analytics Lens: AWS Well-Architected Framework* resource and go to **Scenarios** > **Modern data architecture** > **Reference architecture**. The direct link is provided in the Content Resources section of the course.
|
|Student notes
This section continues the discussion about the modern data architecture and focuses on the processing and consumption layers as documented in *Data Analytics Lens: AWS Well-Architected Framework*.



|Slide number 30
|Instructor notes
|
|Student notes
The processing and consumption layers in the modern data architecture prepare data and make it available to consumers. The consumption layer equates to the analysis and visualization layer of a data pipeline, as identified earlier in the course. In this reference architecture, it's referred to as the consumption layer. This reflects that the data available in the pipeline can be used and consumed in a variety of ways—either by end users or by other downstream systems that consume outputs of data processing.



|Slide number 31

|Instructor notes

|The processing layer components will be expanded upon through the Ingesting and Preparing Data, Ingesting by Batch or by Stream, Processing Big Data, and Processing Data for ML modules. This section serves as the introduction to the types of processing that occur and the fact that individual ETL (or ELT) pipelines are within the larger processing architecture. These ETL or ELT pipelines extract and transform data from the storage layer and store it again in its transformed state.

|This section also references concepts that were presented in the Elements of Data module with respect to actions taken on data and the iterative nature of the pipeline.

|

|Student notes

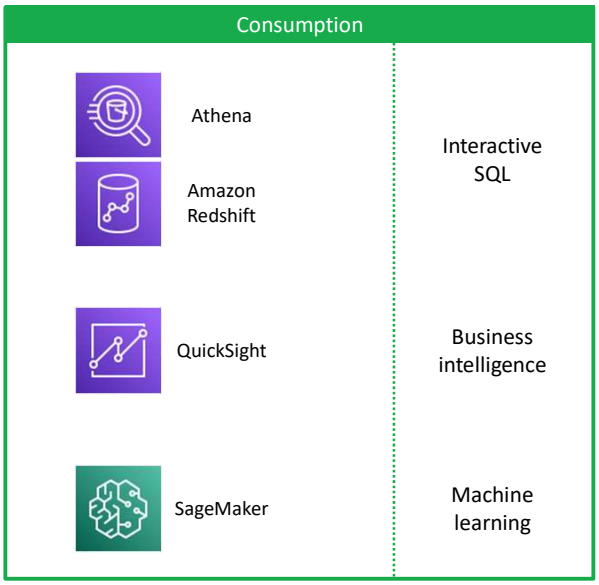
As noted on the previous slide, components in the data processing layer are responsible for transforming data into a consumable state. The processing layer provides purpose-built components that enable a variety of data types, velocities, and transformations. Each component can read and write data to both Amazon S3 and Amazon Redshift in the storage layer, and all can scale to very high data volumes.

Each pipeline reads data from the storage layer, processes it using temporary storage as needed, and then writes it to the appropriate location in the storage layer. The transformations are grouped into three main types, which are aligned to the use case:

- SQL-based processing using a data warehouse (in this case, Amazon RedShift)
- Big data processing using big data tools (in this case, Amazon EMR and AWS Glue)
- Near real-time processing using streaming (in this case, Amazon Kinesis Data Analytics or Spark streaming on Amazon EMR or AWS Glue)

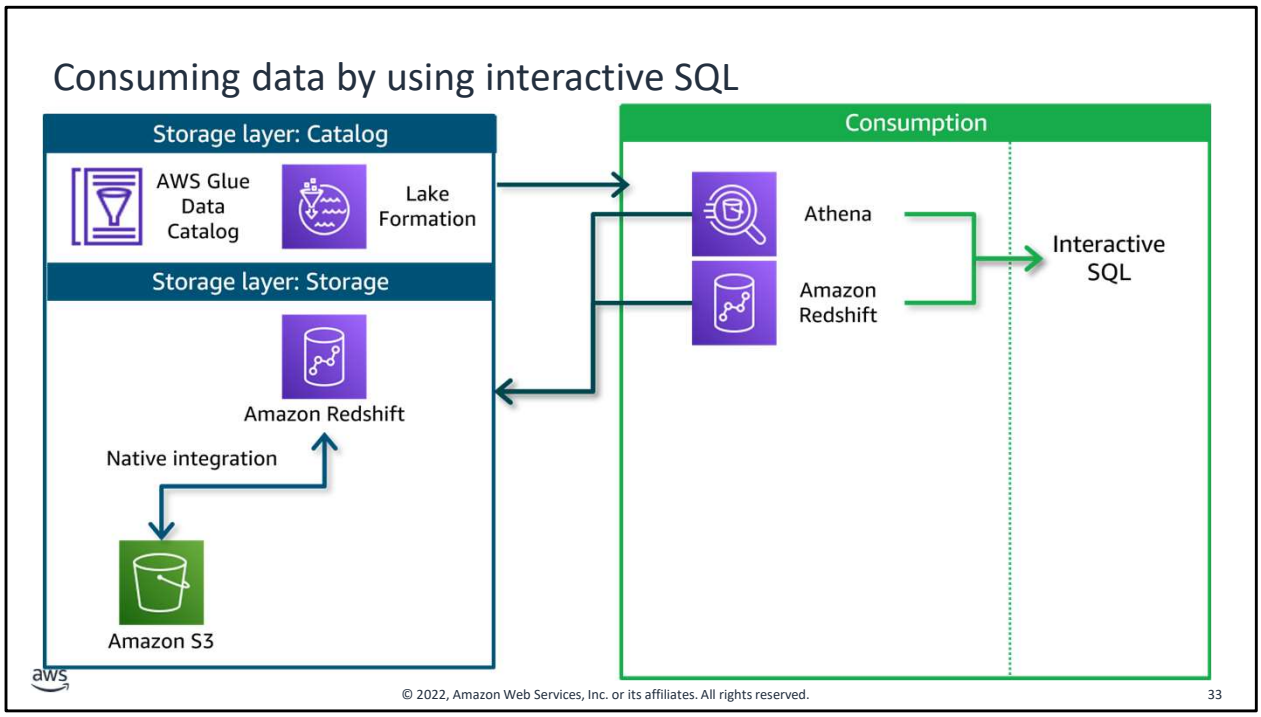
The details provided here are intended to give you a sense of the types of processing that could be implemented and the types of tools that you might need. Effectively, there are tools for SQL querying, tools that are designed for big data processing, and tools that are designed to process streaming data. The Ingesting and Preparing Data, Ingesting by Batch or by Stream, Processing Big Data, and Processing Data for ML modules discuss these types of processing in more detail.

Modern data architecture: Consumption layer

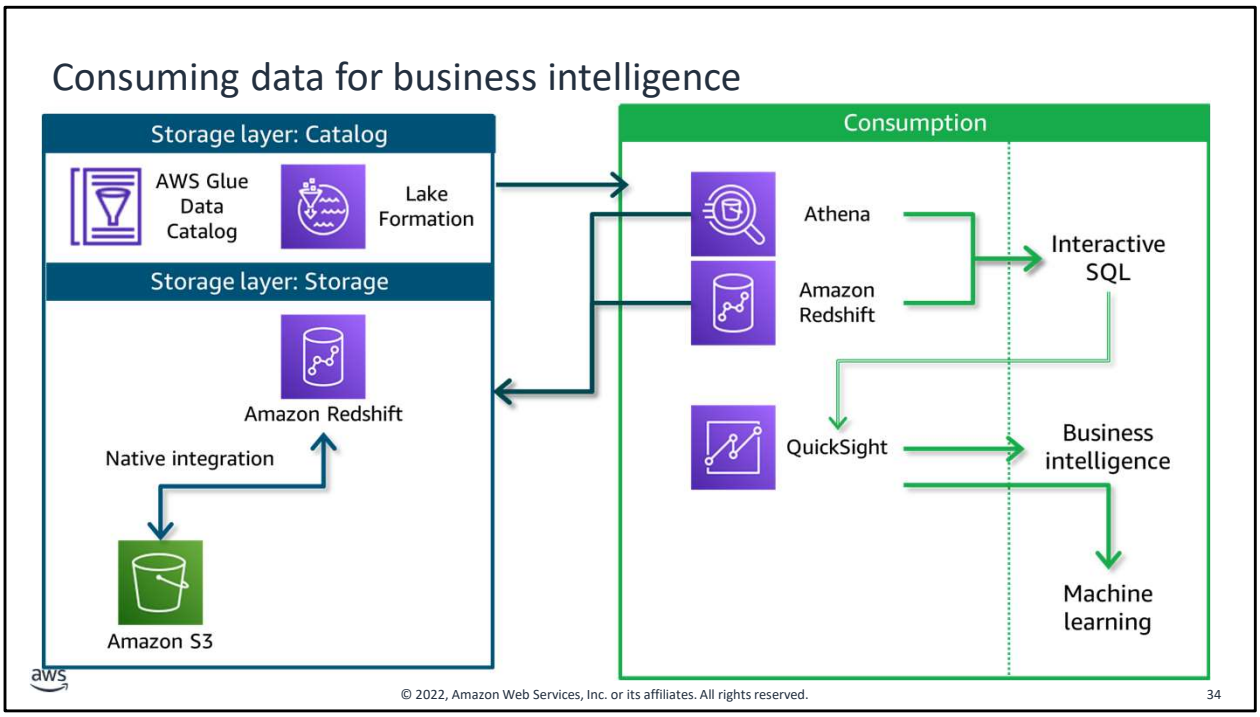


|Slide number 32
|Instructor notes
|The consumption layer maps to the Analyzing and Visualizing Data module in this course, and the tools and methods introduced here are addressed at a deeper level in that module. The services introduced here will also be used within lab activities. The Processing Data for ML module will provide additional coverage of Amazon SageMaker and the specific tasks (such as feature engineering) that are part of ML processing.
|
|Student notes
The data consumption layer is responsible for providing scalable and performant components that use unified interfaces to access all the data and metadata in the storage layer. The consumption layer democratizes access to datasets for different types of users across the organization and enables different analysis methods. Each method has access to combine data from the data warehouse, which is stored in traditional schemas, and data in the lake, which is stored in open formats.

The Analyzing and Visualizing Data module and the Processing Data for ML module discuss these methods and components in more detail, but the next few slides provide a brief overview of how the modern data architecture addresses three analysis methods: interactive SQL queries, BI dashboards, and ML.



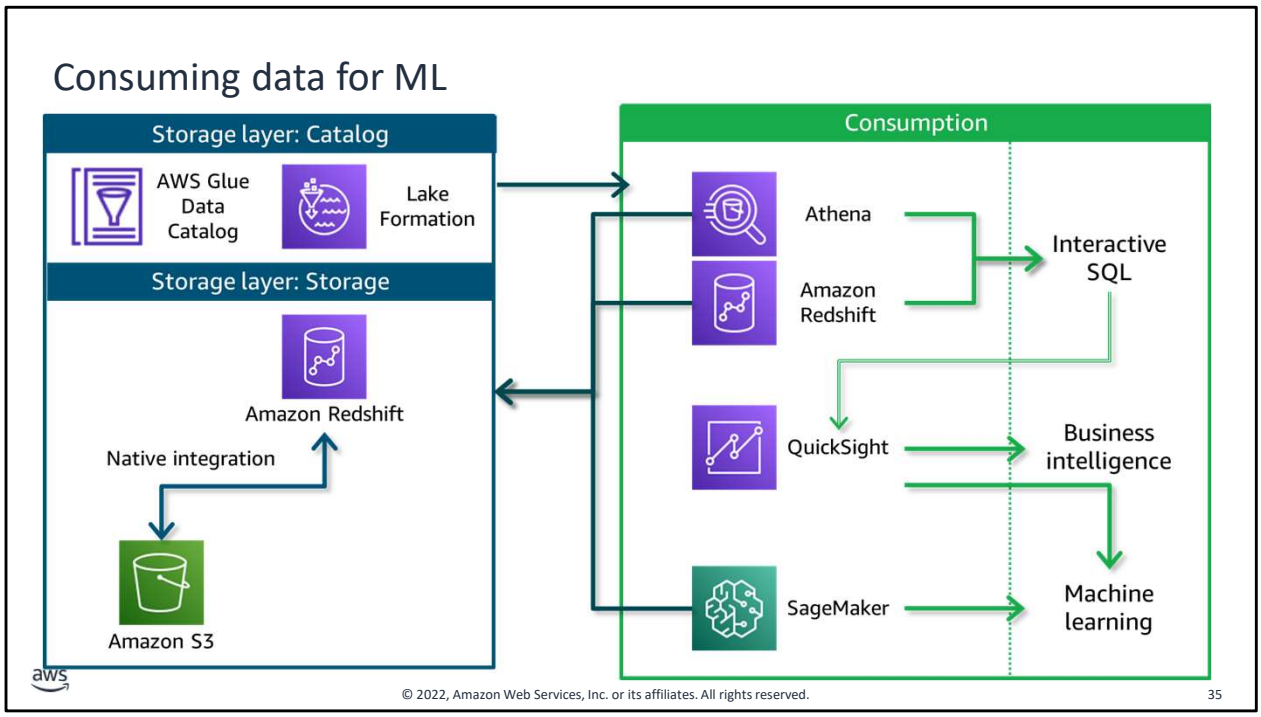
|Slide number 33
|Instructor notes
|
|Student notes
Business analysts and data scientists can use Amazon Redshift (with Redshift Spectrum) or Athena to explore all data in the storage layer by using interactive SQL.



|Slide number 34
|Instructor notes
|
|Student notes
Business analysts can use the Athena or Amazon Redshift interactive SQL interface to power Amazon QuickSight dashboards with data from the storage layer. The lab in this module will give you practice using Athena and AWS Glue to query data that is stored in Amazon S3.

QuickSight is serverless and helps you create and publish interactive BI dashboards. QuickSight is one example of a BI tool, and organizations might use a variety of tools or integrations to derive business intelligence.

QuickSight also enriches dashboards and visuals with automatically generated ML insights such as forecasting and anomaly detection.



|Slide number 35
|Instructor notes
|
|Student notes
Data scientists typically need to explore, wrangle, and perform feature engineering on a variety of structured and unstructured datasets to prepare the data to train ML models.

The interfaces that are available in this architecture simplify data preparation steps. After a data scientist has prepared the data, they can develop, train, and deploy ML models by connecting ML tools to the storage layer. In this AWS architecture, they could use SageMaker to connect to the storage layer to access their training feature sets.

SageMaker is a fully managed service that provides components to build, train, and deploy ML models by using an integrated development environment (IDE) called Amazon SageMaker Studio. In SageMaker Studio, you can upload data, create new notebooks, train and tune models, move back and forth between steps to adjust experiments, compare results, and deploy models to production all in one place. ML models are trained on SageMaker managed compute instances.

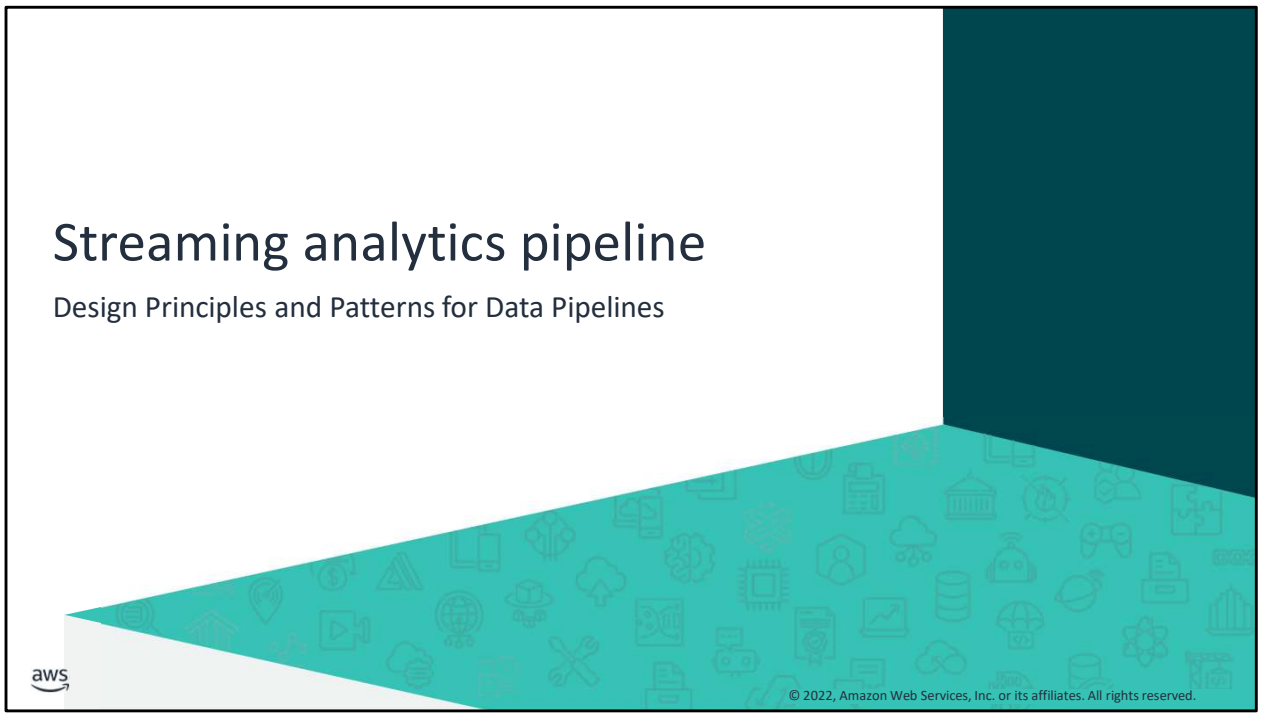
You will learn more about SageMaker in the Processing Data for ML module.

Key takeaways: Modern data architecture pipeline: Processing and consumption

- Components in the processing layer are responsible to transform data into a consumable state.
- The processing layer supports three types of processing: SQL-based ELT, big data processing, and near real-time ETL.
- The consumption layer provides unified interfaces to access all the data and metadata in the storage layer.
- The consumption layer supports three analysis methods: interactive SQL queries, BI dashboards, and ML.

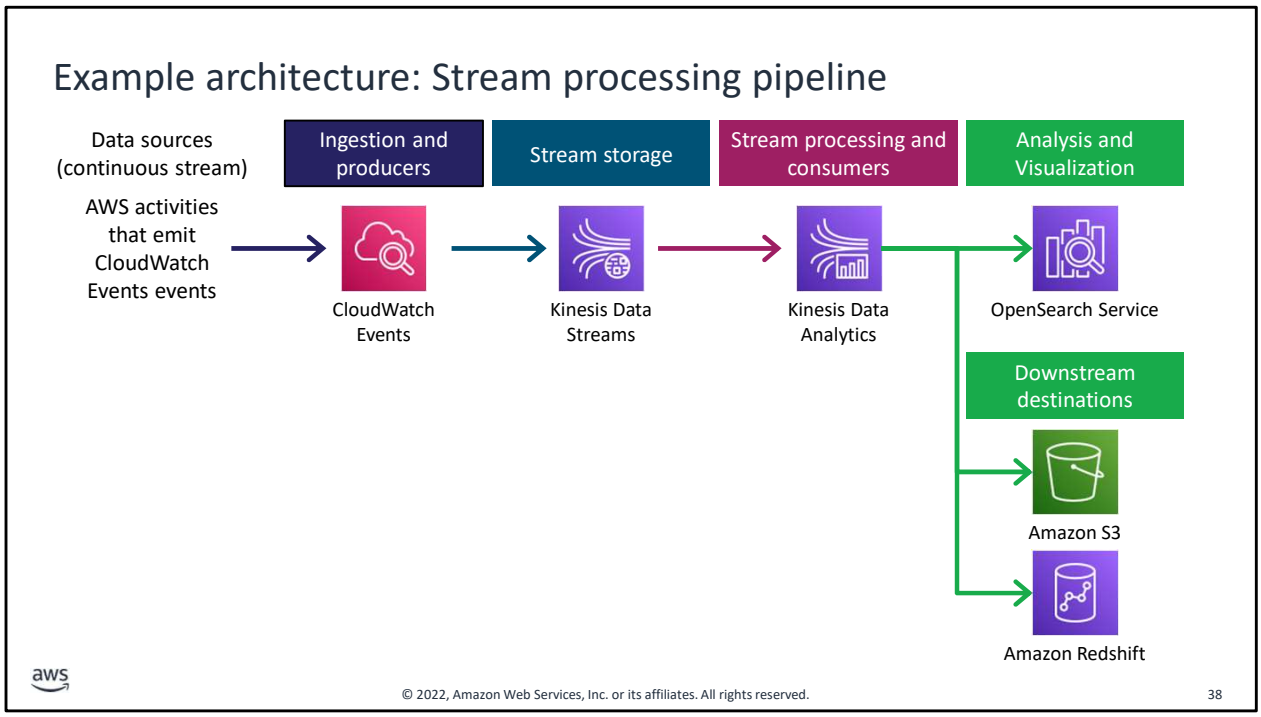
|Slide number 36
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

- The processing layer is where data is transformed for some type of consumption. The modern data architecture supports three general types of processing: SQL-based ELT, big data processing, and near real-time ETL.
- The consumption layer includes components that access the data and metadata in the storage layer (including the data that is transformed by the processing layer). The consumption layer supports three analysis methods: interactive SQL queries, BI dashboards, and ML.



|Slide number 37
|Instructor notes
|Suggest that students open the *Data Analytics Lens: AWS Well-Architected Framework* resource and go to **Scenarios** > **Streaming ingest and stream processing** > **Reference architecture**. The direct link is provided in the Content Resources section of the course.
|
|Student notes
The modern data architecture layers that you reviewed touch on the concepts of ingesting and processing streaming data as part of the ingestion and processing layers, respectively. The Amazon Kinesis services were highlighted to perform these tasks.

But it's worth taking a quick look at a stream processing scenario from end to end to highlight how the distinct layers that you have been learning about blur together when you ingest and process streaming data. The Ingesting by Batch or by Stream module describes streaming pipelines in more detail. The pipeline that is depicted on the slide is based on the reference architecture for streaming ingest and stream processing in *Data Analytics Lens: AWS Well-Architected Framework*.



|Slide number 38
|Instructor notes
|
|Student notes
Streaming pipelines follow the same general layers as other pipelines, but there are some unique considerations. Data sources include clickstream logs, mobile apps, existing application databases, or Internet of Things (IoT) sensors. You might want to respond to this data in real time or use it for analysis later.

Producers ingest records onto the stream. Producers are integrations that collect data from a source and load it onto the stream. *Consumers* process records. Consumers read data from the stream and perform their own processing on it. The stream itself provides a temporary but durable storage layer for the streaming solution.

In the pipeline that is depicted in this slide, Amazon CloudWatch Events is the producer that puts CloudWatch Events event data onto the stream. Kinesis Data Streams provides the storage. The data is then available to multiple consumers.

With real-time streaming analytics, records on the stream are typically processed sequentially and incrementally by record over sliding time windows. In the pipeline that is

depicted on the slide, Kinesis Data Analytics is a consumer of the stream and processes streaming data by using custom applications or standard SQL. In this example, results are sent to OpenSearch Service, where they can be used to visualize real-time insights with OpenSearch Dashboards immediately.

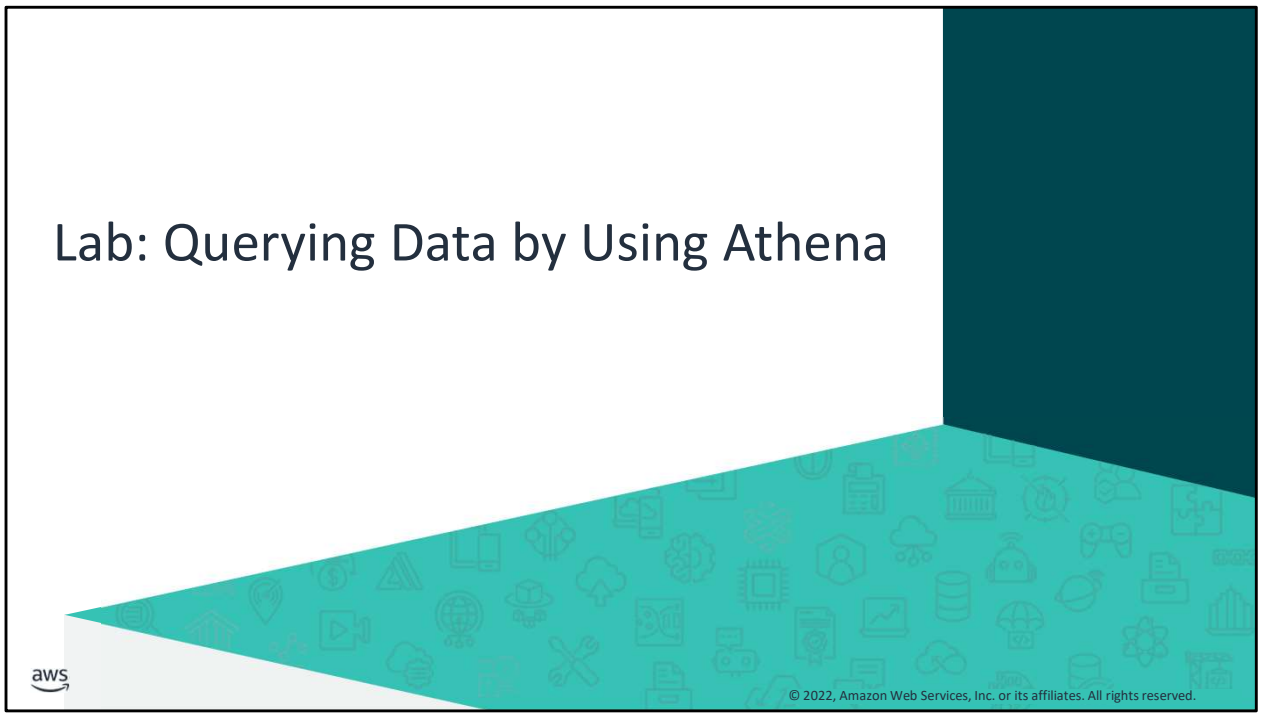
In this scenario, Amazon S3 and Amazon Redshift also consume the data that Kinesis Data Analytics processes. These downstream destinations aren't being used for real-time analytics but could be used for serving applications such as one-time analytics and ML. This is an example of how the modern data architecture supports the goal of making ingested data available to let different consumers perform different types of analytics and run AI/ML applications.

Key takeaways: Streaming analytics pipeline

- Streaming analytics includes producers and consumers.
- A stream provides temporary storage to process incoming data in real time.
- The results of streaming analytics might also be saved to downstream destinations.

|Slide number 39
|Instructor notes
|
|Student notes
Here are a few key points to summarize this section.

- Streaming analytics includes producers who put things on the stream and consumers who get things off the stream.
- A stream provides temporary storage to process incoming data in real time for delivery to real-time applications, such as real-time dashboards.
- The results of streaming analytics might also be saved to more durable storage for additional processing downstream.



|Slide number 40
|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: Querying Data by Using Athena

- In this lab, you will learn how to use Athena and AWS Glue to query data that is stored in Amazon S3.
- You will optimize Athena queries and create views to simplify use by multiple team members.
- You will use AWS CloudFormation to integrate Athena into your infrastructure and AWS Identity and Access Management (IAM) to provide the appropriate level of access to your users.

|Slide number 41
|Instructor notes
|
|Student notes
Access the lab environment through your online course to get additional details and complete the lab.

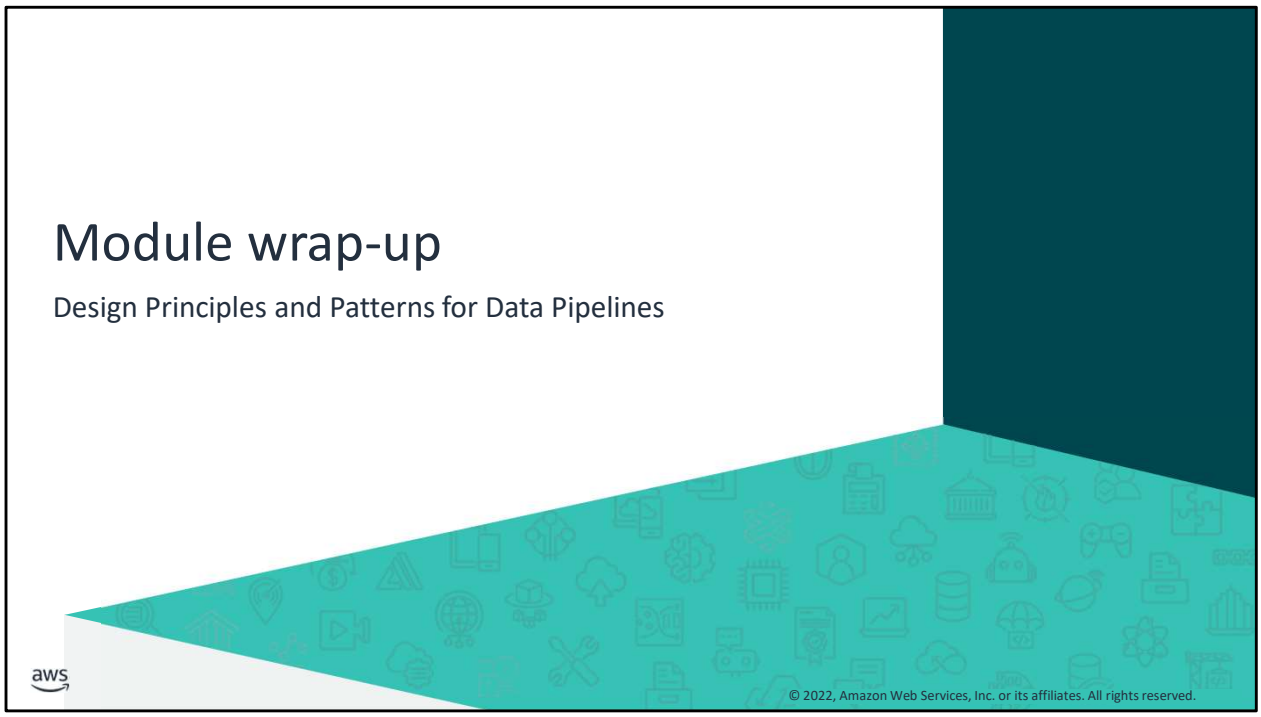
Debrief: Querying Data by Using Athena

- How could you use Athena and AWS Glue in other businesses with analytics workloads?
- What security considerations do you need to address before you implement solutions like those in the lab with a team?
- How could you address concepts such as least privilege and data governance with these types of solutions?
- What is one way that a data analyst could run a named query that you created and provided to the team?

aws

|Slide number 42
|Instructor notes
|Q1 - **Example strong response:** Using Athena and AWS Glue is an example of using a relational database to capture and analyze data that is formatted as columns and rows. Any company could effectively use this type of database to track any part of its business, such as sales, support cases, and customer contact information.
|Q2 - **Example strong response:** You should only share data that is stored in the AWS Glue database with those who need access to perform a specific job function. This follows the principle of least privilege and is one of the principles of the AWS Well-Architected Framework.
|Q3 - **Example strong response:** You should store data in a centralized data lake with versioning. Only provide access to those who it to perform a specific job function. You could create a Center of Excellence team to introduce, enforce, and maintain best practices for the overall data and analytics solution. This team would help to ensure that best practices are maintained, data is secured, and data remains of high quality so that the best insights can be taken from the data.
|Q4 - **Example strong response:** They could run a named query directly from the Athena console query editor by using the Saved Queries tab.
|
|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 43
|Instructor notes
|
|Student notes
This section summarizes what you have learned and brings the Design Principles and Patterns for Data Pipelines module to a close.

Module summary

This module prepared you to do the following:

- Use the AWS Well-Architected Framework to inform the design of analytics workloads.
- Recount key milestones in the evolution of data stores and data architectures.
- Describe the components of modern data architectures on AWS.
- Cite AWS design considerations and key services for a streaming analytics pipeline.

|Slide number 44
|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 focused on the design principles and patterns that you will use to build data pipelines. You learned about how the evolution of data stores and data architectures informed design principles of the modern data architecture. You used the Well-Architected Framework to find design principles and recommendations that are related to building analytics pipelines. In addition to the modern data architecture, you were introduced to key characteristics of streaming data pipelines.

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 45
|Instructor notes
|
|Student notes
Use your online course to access the knowledge check for this module.

Sample exam question

A data engineer has implemented the AWS modern data architecture as described in the Well-Architected Framework. An analyst wants to combine and explore customer sales data from the data warehouse with customer support ticket data for the last 6 months. The support ticket data is available as a JSON extract from their SaaS support system (Zendesk).

How could the engineer meet this need and simplify the effort for the engineer and the analyst? (Select TWO.)

Identify the key words and phrases before continuing.

The following are the key words and phrases:

- **AWS modern data architecture**
- **Combine and explore** customer sales in the **data warehouse** with **SaaS data** available as a **JSON extract**
- **Data for the last 6 months**
- **Simplify** the effort

|Slide number 46
|Instructor notes
|The key words section is animated to be revealed on click.
|**Slide titled Matching ingestion services to variety, volume, and velocity** includes information about Amazon AppFlow. **Slide titled Catalog layer for governance and discoverability** discusses using Amazon Redshift Spectrum to combine data from the data lake and warehouse.
|
|Student notes
The question notes that the modern data architecture has been implemented and the analyst wants to combine data in the warehouse with data that needs to be ingested into the pipeline from a SaaS system. The SaaS system data is in JSON format, which indicates that it is semistructured. The range of data to be ingested is 6 months of ticket data. Finally, the goal is to simplify the work for both the engineer and the analyst.

Sample exam question: Response choices

A data engineer has implemented the **AWS modern data architecture** as described in the Well-Architected Framework. An analyst wants to **combine and explore customer sales data** from the **data warehouse** with customer support ticket data for the **last 6 months**. The support ticket data is available as a **JSON extract** from their **SaaS support system** (Zendesk).

How could the engineer meet this need and **simplify** the effort for the engineer and the analyst? (Select TWO.)

Choice	Response
A	Use Amazon AppFlow to ingest the SaaS data into Amazon S3.
B	Use Kinesis Data Firehose to stream the SaaS data into Amazon S3.
C	Use OpenSearch Service to index the customer ticket data so that the analyst can search the data.
D	Create an ETL process that transforms the data that is ingested into Amazon S3 to a curated format. Then, load the data into the data warehouse.
E	Use Amazon Redshift Spectrum to write a query that includes data from the data lake and the data warehouse.

|Slide number 47
|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 two best address the question.

Sample exam question: Answer

The correct answers are A and E.

Choice	Response
A	Use Amazon AppFlow to ingest the SaaS data into Amazon S3.
B	
C	
D	
E	Use Amazon Redshift Spectrum to write a query that includes data from the data lake and the data warehouse.

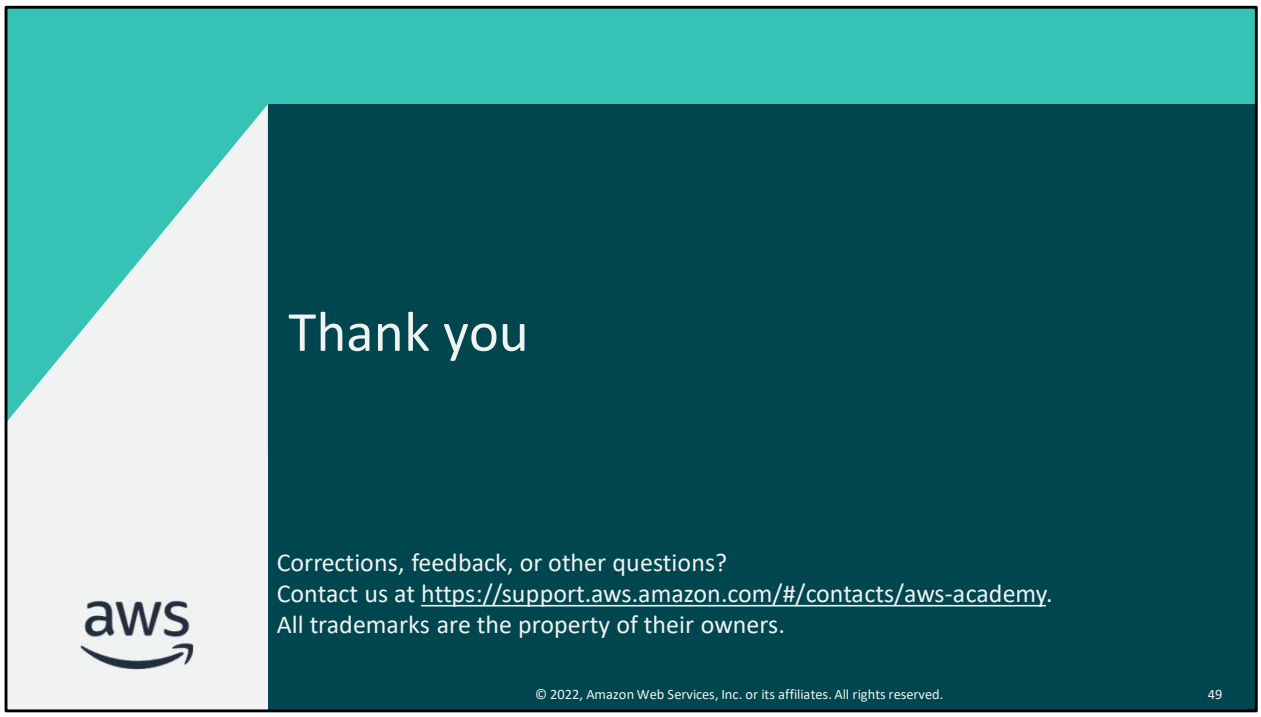
|Slide number 48
|Instructor notes
|
|Student notes
Choice B (Use Kinesis Data Firehose to stream the SaaS data into Amazon S3) doesn't fit the scenario. Kinesis Data Firehose would be appropriate for streaming data, but this request is for a dataset comprised of 6 months of ticket data.

Choice C (Use OpenSearch Service to index the customer ticket data so that the analyst can search it) doesn't fit the scenario. OpenSearch Service is a purpose-built data store and search engine that is optimized for real-time analytics. The dataset doesn't require real-time analytics, and the analyst would need to search for ticket data and then combine it with the sales data separately.

Choice D (Create an ETL process to transform the data that is ingested into Amazon S3 to a curated format and load the data into a data warehouse) doesn't fit the scenario. The goal is to find the simplest way to combine the data. Developing a method to transform the JSON extract into a schema within the data warehouse would create a lot of additional effort. Because the goal is to explore the data instead of a near real-time dashboard, there doesn't seem to be value in conforming the data to fit into the data warehouse. This would

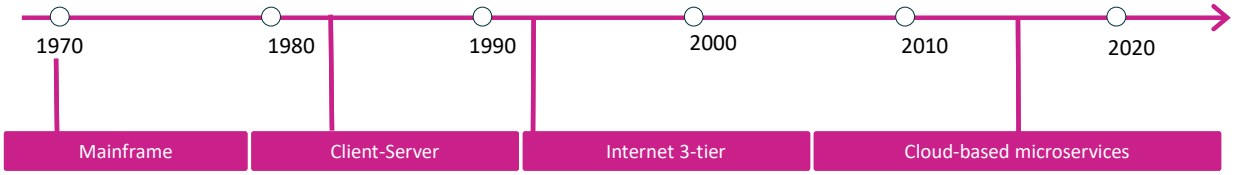
not simplify the approach.

The correct answers are A and E. Amazon AppFlow is a purpose-built service that integrates with SaaS applications including Zendesk and can deliver data directly to Amazon S3. As noted in the storage layer of the modern data architecture, Redshift Spectrum can get schema data from the Lake Formation catalog for objects that are stored in Amazon S3. Redshift Spectrum can also combine data from the lake with data from the warehouse in a single query. This is the simplest approach for the engineer and the analyst.



|Slide number 49
|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.

Application architecture evolved into more distributed systems



Mainframe | Client-Server | Internet 3-tier | Cloud-based microservices

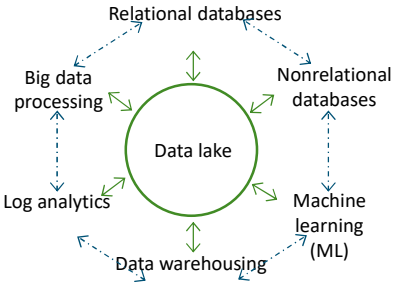
aws

Source for timeline.

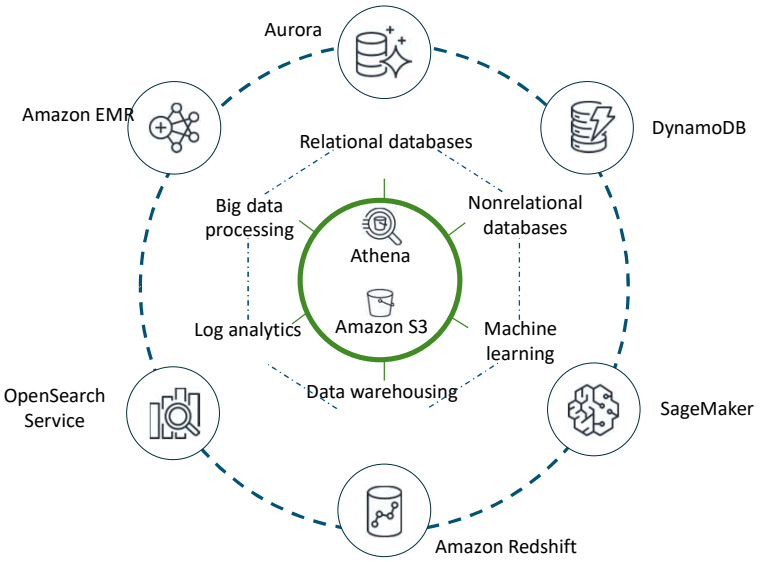
Modern data architecture

Key design considerations

- Scalable data lake
- Performant and cost-effective components
- Seamless data movement
- Unified governance



Modern data architecture



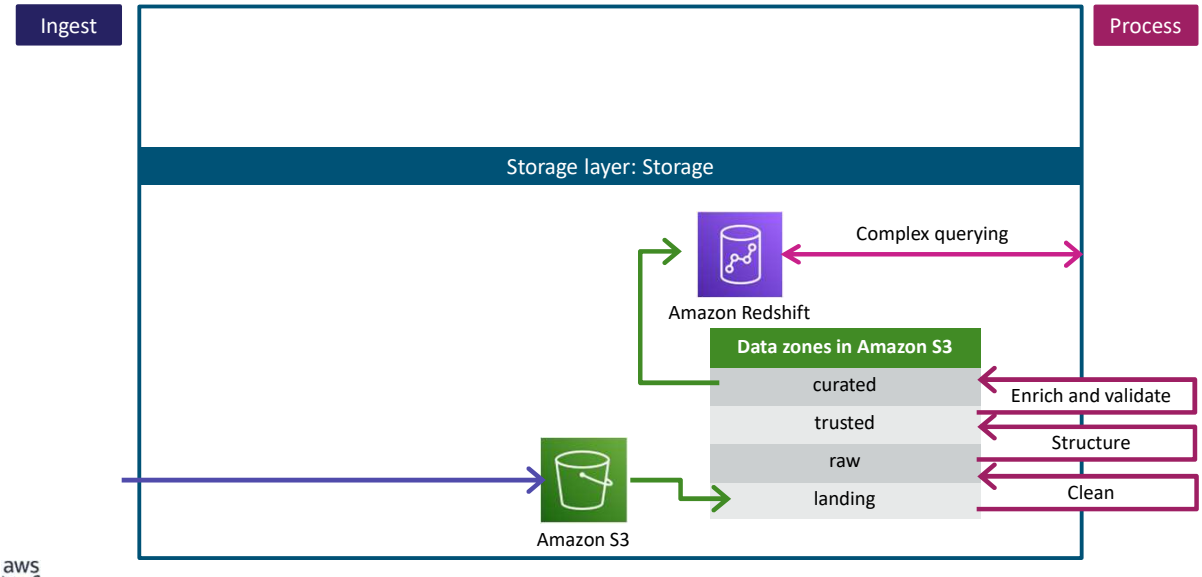
Source for AWS arch

* Aurora
* DynamoDB
* SageMaker
* Amazon Redshift
* OpenSearch Service
* Amazon EMR

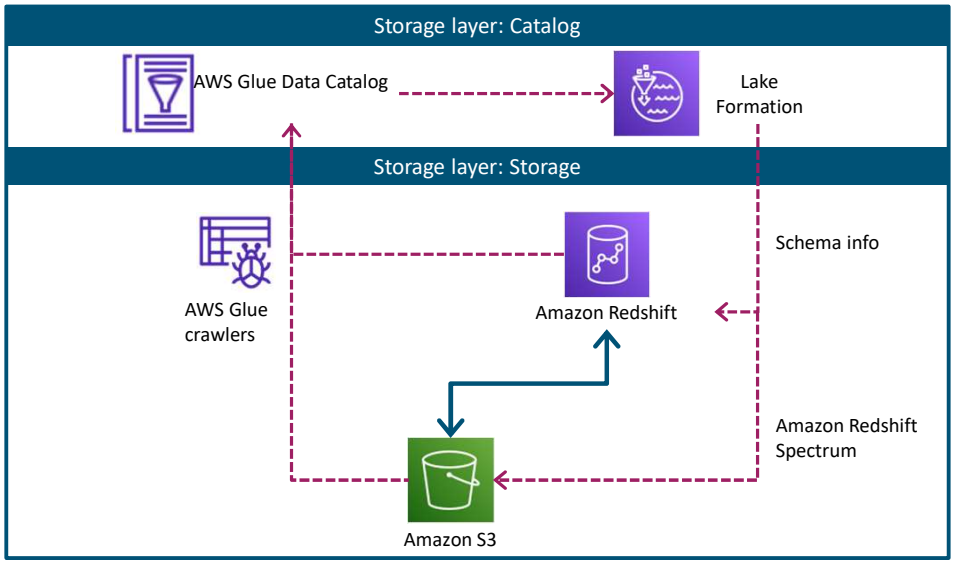
In the middle of the diagram, the labels would be:

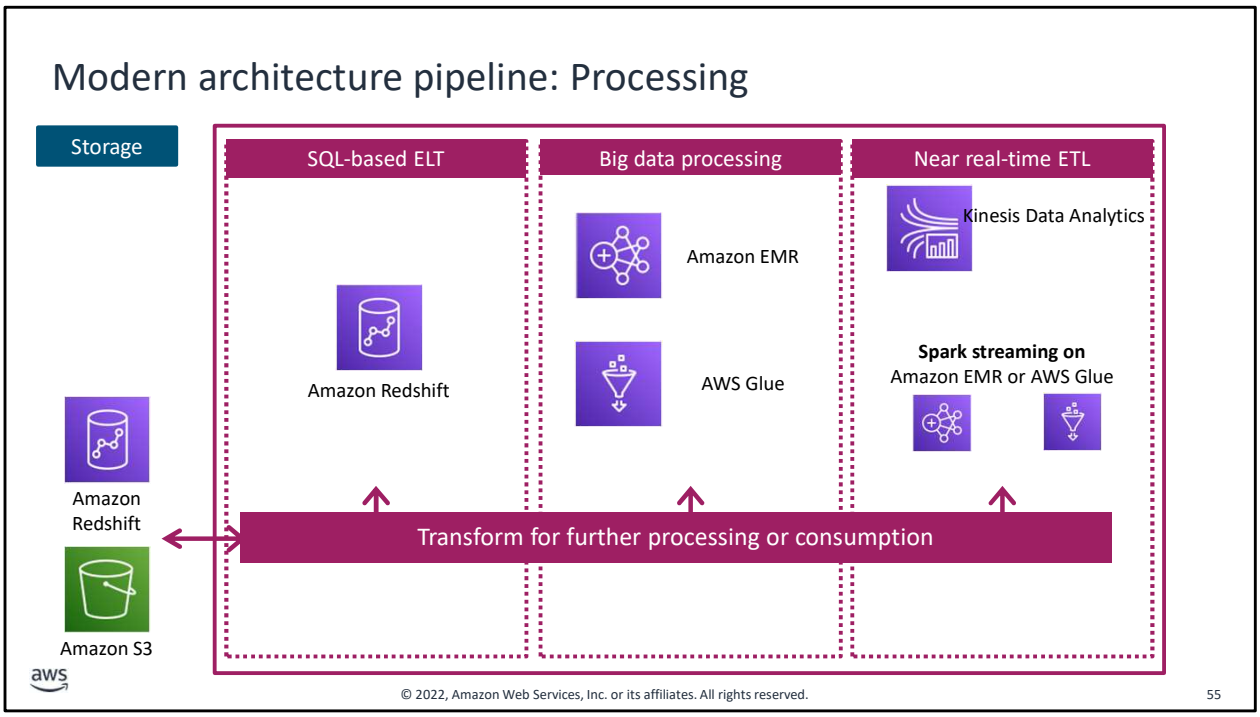
* Athena
* Amazon S3

Storage zones for data in different states



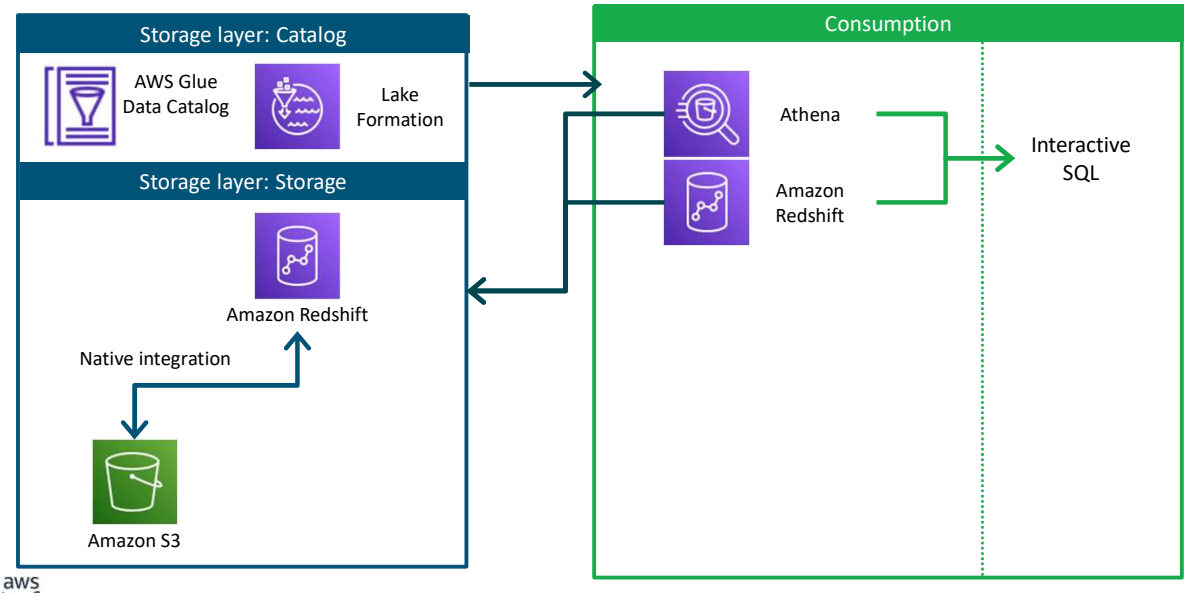
Catalog layer for governance and discoverability



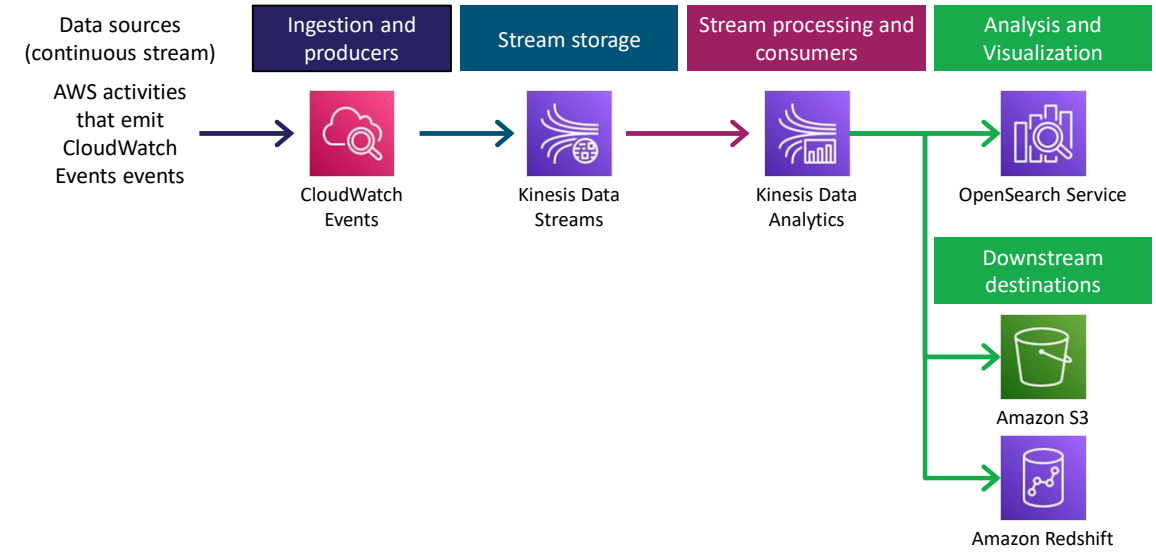

Modern architecture pipeline: Processing
Storage
SQL-based ELT
Big data processing
Near real-time ETL
Amazon EMR
Kinesis Data Analytics
Amazon Redshift
AWS Glue
Spark streaming on
Amazon EMR or AWS Glue
Amazon
Redshift
Amazon S3
Transform for further processing or consumption
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.
55

Source

Consuming data



Example architecture: Stream processing pipeline



Source