AWS Whitepaper

Modern Data Architecture Rationales on AWS

Modern Data Architecture Rationales on AWS: AWS Whitepaper

Table of Contents

Abstract and introduction

Publication date: **February 15, 2022** (*Document revisions*)

Abstract

Today, with the rapid growth in data from ever-expanding data producer platforms, organizations are looking for ways to modernize their data analytics platforms. This whitepaper lays out the traditional analytics approaches and their challenges. It then explains why a modern data architecture is required, and how it helps solve the challenges presented by traditional analytics.

This paper also provides context around how Amazon Web Service (AWS) Cloud analytics services provide a foundation for building this modern data architecture platform.

Introduction

Many large organizations use enterprise data warehouses (EDW) as the single version of truth, and a semantic layer for all the data in the organization. EDW acts as a harmonization layer for data originating from multiple sources into an integrated data model.

In contrast, a data lake is a storage repository that holds a vast amount of raw data in its native format until it is needed for analytics applications. While a traditional data warehouse stores data in hierarchical dimensions and tables, a data lake uses a flat architecture to store data, primarily in files or object storage.

EDW became popular in the last two decades because of the popularity of using business intelligence (BI) for strategic decision making, as well as for daily operational activities across the organization.

The two most popular approaches to realize a data warehouse over the years have been:

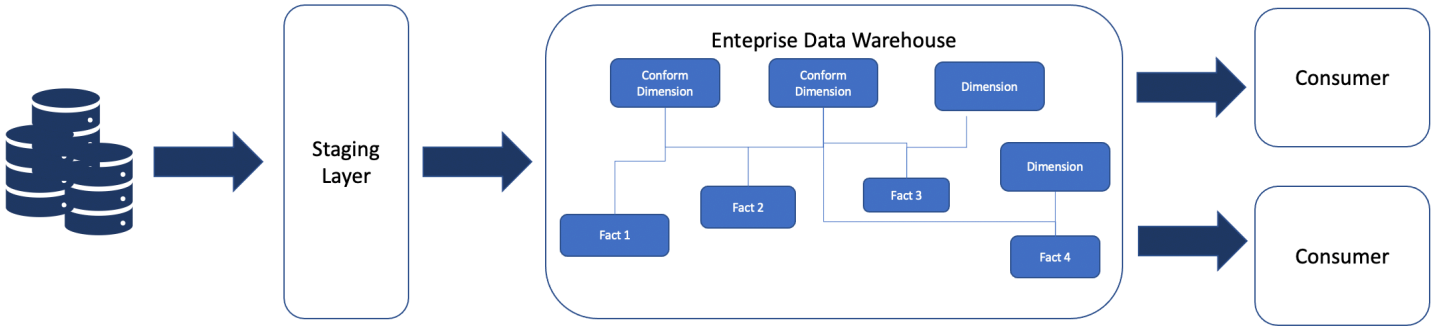
- **Kimball approach** – proposed by Ralph Kimball
- **Inmon approach** – proposed by Bill Inmon

Enterprise data warehouse

Kimball approach

Ralph Kimball's view is that data warehouse design is a bottom-up process, and the data warehouse is formed by several smaller data marts (star schemas) based on various business contexts with some key dimensions which are common across multiple business contexts.

This approach suggests gradual building of the data warehouse by adding new data marts. Data marts are added by adding new fact tables to represent transactional and event data, and by adding new dimensions or attributes to existing conformed dimensions.



Bottom-up EDW approach

The various combination of dimensions and facts form a logical data mart for a specific business use case.

Benefits:

- EDW is purpose-built with logical grouping of data marts; EDW is gradually realized based on business use cases and all the data gets stored in EDW.
- Data is highly denormalized and redundant, which makes it optimal for analytical purposes with generally better performance. In some cases, a performance layer is also created in terms of physical data marts to achieve optimal performance for use cases such as aggregated reports, trend reports, and time frame reports.
- EDW is easier to understand for non-IT users, because logical data marts are purpose-built subject areas for a particular business domain. It's easy to create and understand a business metadata layer.
- EDW is highly suitable for massively parallel processing Relational Database Management Systems (RDBMS), and also suitable for data lake implementations.

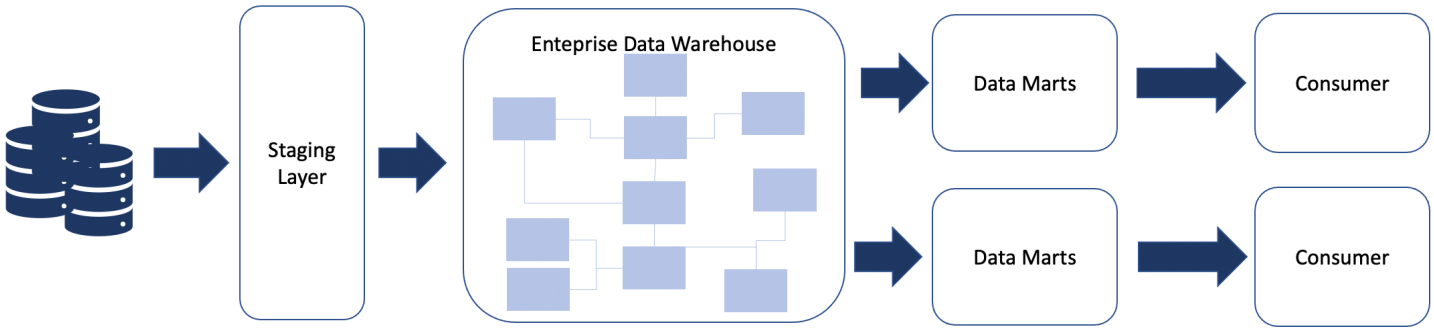
Shortcomings:

- Because EDW is use case driven, creating a fully harmonized integration layer might be difficult to achieve. Data duplication and data sync issues also make this task challenging.
- Tight coupling with consumers causes delay in changes to mitigate the impact on consumers.
- The data volume of the tables is high, and this results in heavy and long running extract, transform, and load (ETL) jobs. Additional jobs may be required to get a physical performant layer.

Inmon approach

Bill Inmon proposed a top-down approach to create a normalized integrated data model where all the data must first be harmonized and translated to a unified data model as per the organization's context.

According to this approach, data from all the source systems is unified, formatted, type casted, and semantically translated (mapped to the same vocabulary up front). To achieve this, organizations must put in significant amounts of time and effort to define a unified data model to align all the data elements from the source systems.



Top-down EDW approach

Benefits:

- Highly normalized and no/low data redundancy.
- Data relationships and granularity is clear with proper semantics.
- Decoupled from consumers, so it can evolve independently.
- Optimized for storage and historical data.
- More suitable for RDBS implementations.

Shortcomings:

- Data model is complex and data preparation effort is large.
- Integrated model leads to higher dependency in data sources and reduces parallelism.
- Third normal form (3NF) leads to a lot of joins for analysis, making it compute-heavy for doing ad-hoc analysis.
- Most of the time, source data needs to be broken down for the EDW model, and businesses may need to join multiple tables together to derive a final combined dataset across multiple sources.
- Usually a presentation layer (consisting of multiple data marts) is created on top, which is de-normalized in a star or snowflake schema for optimal performance for analytical queries.
- Less suitable for data lake implementation.

Because of these reasons and the exponential growth of data in recent years, data warehouses by themselves prove to be less agile and tend to increase the time to market to realize data products in a large organization.

Also, there is an increasing need for organizations to use several tools on this data to create modern data application, whereas traditional data warehouse patterns mostly supported SQL and BI workloads.

In recent years, with the advent of cloud and big data tools, organizations are making a shift towards a modern data architecture pattern.

Modern data architecture

As technology rapidly evolves, the type and volume of data also grows rapidly. Organizations want to capture all this data and derive value from it as fast as possible, to stay ahead of competition. A data warehouse is a type of data store that caters to a particular type of use case – Online Analytics Processing (OLAP). To meet other types of use cases, such as log analytics, predictive analytics, and big data processing, a one-size-fits-all data strategy creates rough edges and is challenging to scale for future growth.

A modern data architecture gives you the best of both data lakes and purpose-built data stores. It lets you store any amount of data you need at a low cost, and in open, standards-based data formats. It isn't restricted by data silos, and lets you empower people to run analytics or machine learning (ML) using their preferred tool or technique. Also, it lets you securely manage who has access to the data.

Organizations want to build a better data foundation to:

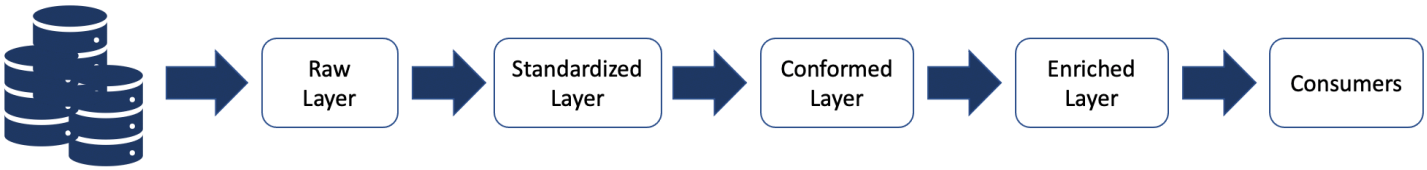
- Modernize their data infrastructure.
- Unify the best of both data lakes and purpose-built data stores.
- Innovate new experiences and reimagine old processes with AI/ML.

The main features of a modern data architecture are:

- Scalable, performant and cost-effective.
- Purpose-built data services.
- Support for open-data formats.
- Decoupled storage and compute.
- Seamless data movements.
- Support diverse consumption mechanisms.
- Secure and governed.

Modern data architecture ecosystem

Before looking into details of the tools that make up a modern data architecture, it's important to understand the different layers through which the data passes, and the significance of each of these layers.



Data layers of modern data architecture

Raw layer

The *raw layer* acts as the landing zone for all the source data in the format delivered by the source. The data in this layer can be stored for longer periods of time, and can be archived for audit and reproducibility perspective.

Standardized layer

Because the data that arrives in the raw layer can be in specific formats as delivered by the source, the *standardized layer* is used to store the data in a standard format (typically Apache Parquet file format) after performing schema validations, schema evolution control, data quality rules, tokenization, and cleansing rules for the data. A typical example of cleansing rule is to standardize the datetime format to a standard format (for example, ISO 8601).

The data stored in this layer is already optimized for analytical queries, because it is partitioned and stored in columnar format. This data is typically also stored in a central data catalog for discovery.

This layer acts as the consumption layer for standardized raw data in the organization.

Conformed layer

Typically, in any organization, there are some common entities and subject areas which are well defined, and are commonly understood and used across the organization. Such entities can be treated as *conformed entities*, and end up in the *conformed layer*.

The definition of these common entities needs to be governed centrally, because they are usually formed based on the primary data of an organization.

The entities in this layer can be created centrally, or the definitions can be created centrally and the operations of loading and maintaining can be delegated to various data engineering teams that need such entities based on a first-come-first-realize basis.

All these entities are also logged in a central data catalog with clear ownership and metadata with respect to Personal Identifiable Information/Payment Card Industry (PII/PCI), retention, purpose, and so on.

One of benefits of managing the conformed entities centrally is clear enterprise ownership. Because this data is used by several parties within the organization, if the ownership is distributed,

the definitions can become ambiguous, and maintenance and retention of history, along with governance and data management of these conformed entities, can become challenging.

Enriched layer

The *enriched layer* is more of a logical layer, because it is aimed at data engineering teams, who create their own data products combining conformed entities and standardized raw data.

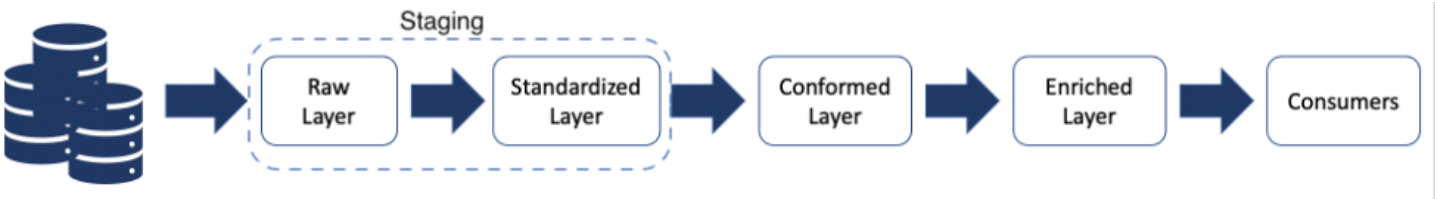
Mostly, these business domain-focused teams have many end products that are useful for particular business domains; however, in some cases these could also be products that are useful for other business domains. These are sometimes called *golden datasets* and can be offloaded to the data lake for sharing across the business.

All the end product datasets in this layer should also be added to the central data catalog with proper labels, metadata, and the purpose of the datasets.

Modern data architecture layers deep-dive

Following is an overview of each of the layers in the modern data architecture.

Staging layer



Staging layer of modern data architecture

The *staging layer* in traditional data warehouses resided on RDBMS. It was a schema on write approach, which rejected any incompatible changes and needed tables to be dropped and re-created when incompatible schema changes occurred. However, this also meant that a well-defined interface (contract) was ensured for the data pipelines, which ensured minimal disruption by stopping incompatible changes at the door of the data platform.

In a modern data architecture, the staging layer is represented by the raw layer plus the standardized layer. The raw layer is mainly used to land the data as-is, and is used for audit, exploration, and reproducibility purposes.

Standardized layer

The *standardized layer* is the interface for users (data engineers, data scientists, and so on) to explore and build use cases using raw data that is validated and conforming to certain defined standards. This layer is a schema on read layer. The data is stored in low-cost storage such as Amazon Simple Storage Service (Amazon S3) in optimized and compressed format.

The challenge of schema evolution

The standardized layer is the interface for self-service BI, exploration and Business as Usual (BAU) data pipelines refreshing the insights on hourly, daily or weekly basis which are used by ML models or business stakeholders to make critical decisions on a day-to-day basis. Consequently, these pipelines need an agreed-upon interface to consistently produce insights with limited issues.

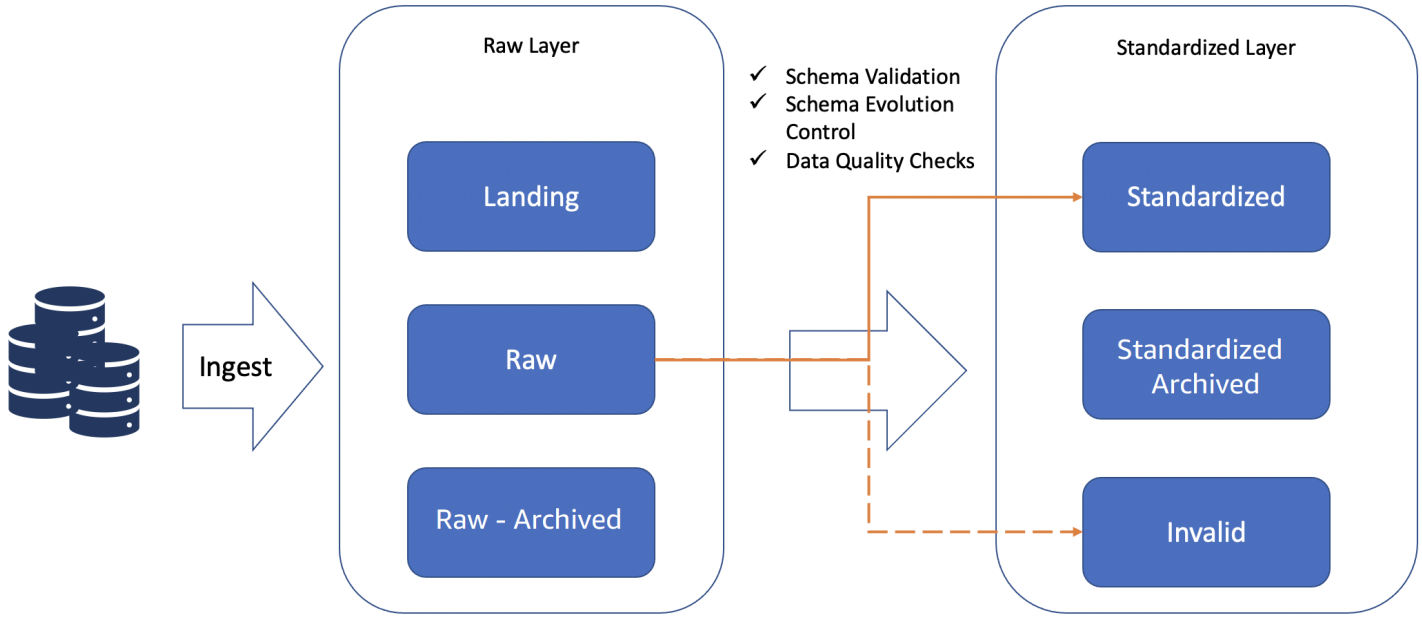
The challenge of schema evolution appears in a data lake due to the schema on read nature of implementation. Typically, in a data lake, the schema is enforced on read. This can make establishing contracts challenging. For example, a data pipeline might use a numeric column to do some calculations. If this column is changed to a string by the source, without notifying the consumer, the pipeline will certainly fail.

To avoid such schema change issues, a standardized layer is recommended. The standardized layer acts as the contract between data producers and consumers. All the data can land in the raw layer, making it an enterprise repository of data for exploration. However, only standardized and harmonized data with schema validation and evolution control is available in the standardized layer. This way, the standardized layer exposes only quality assured data to the consumers.

Schema validation and evolution control assures data consumers that any changes from producers that impact the consumers are flagged, so the data lake administrator can take appropriate actions.

Essentially, schema evolution controls schema drift that is based on the user's configuration, and should normally be configured so that only compatible changes are propagated further.

Any incompatible changes, flagged in the invalid zone, need to be evaluated by data lake administrators so they can make informed decisions (such as creating a new version of the dataset).



Standardized layer activities

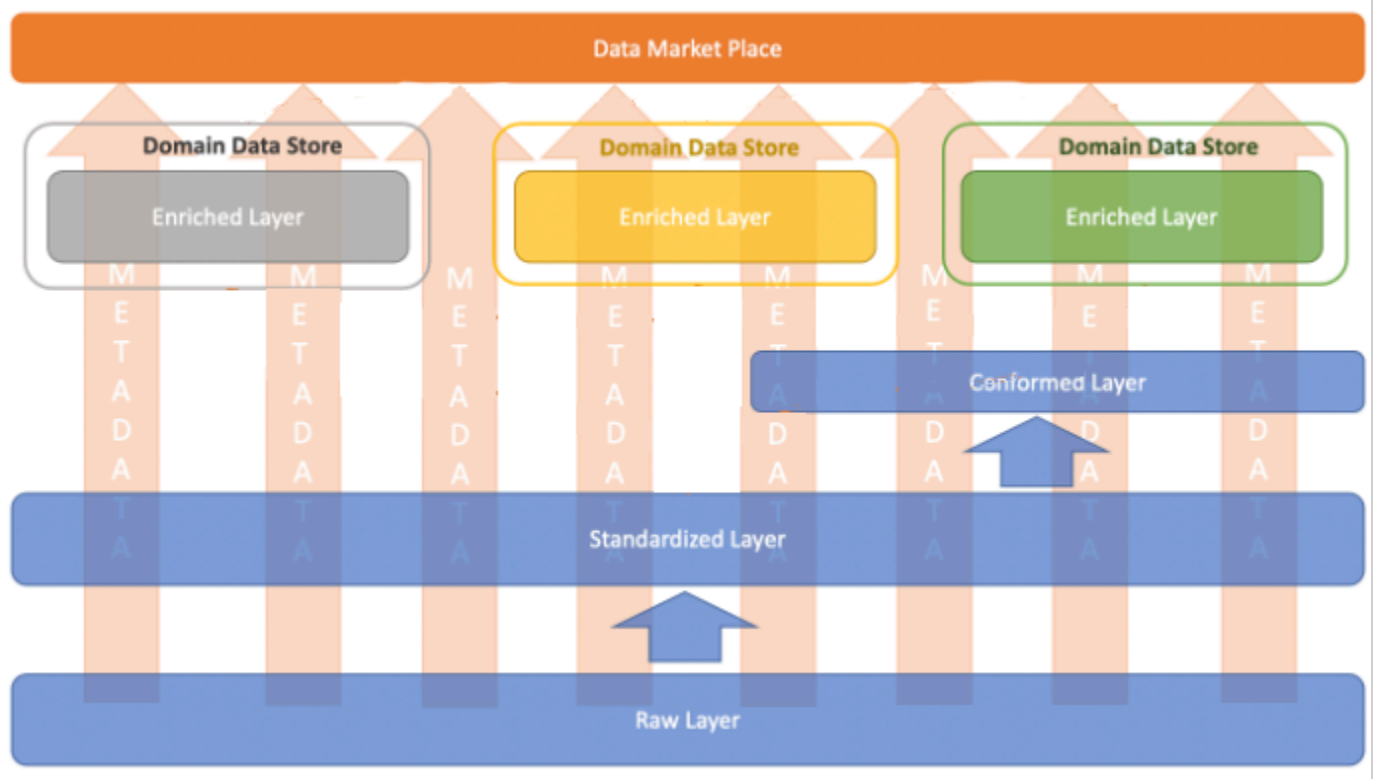
Conformed layer

In general, you can view the *conformed layer* as a repository of common entities which are well defined and are commonly understood and used across the organization, as explained in the conformed layer description. There are two main patterns to understand:

Centralized pattern

These are common entities and properties (attributes) that are generically used throughout the enterprise. This pattern generally limits the properties of such entities, and can be governed and made centrally available.

For example, in the case of a customer entity, these are generic properties such as name, age, profession, address, and so on. This makes it easier to centrally manage and provide primary data information, especially from a governance point of view (such as PII/PCI management, and lineage and lifecycle management of such sensitive data).



Centralized pattern for conformed layer

In the preceding image, metadata is logged in the central data catalog and data exchange is governed using the capabilities of marketplace data.

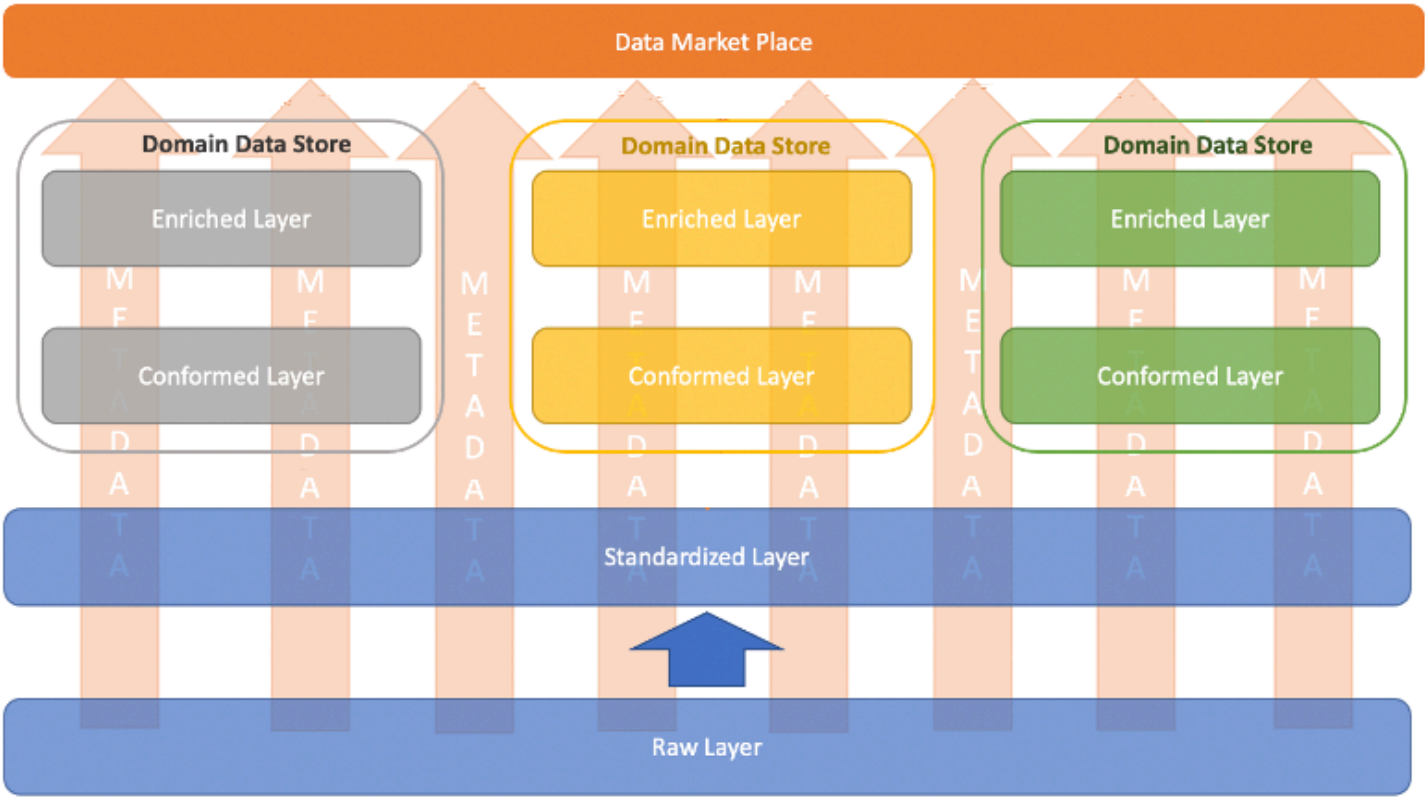
Different business units might have additional properties with respect to the customer entity; for example, from a marketing perspective, preferred channels, lifetime value, next best action, customer consent, net promoter score (NPS), and so on. These might be very important properties, but they are only relevant in the bounded context of marketing. Therefore, these properties could be derived and added by the marketing team in their enriched layer from the standardized raw layer. Deriving and managing all properties relevant for each bounded context at the enterprise level centrally often causes a bottleneck and increases time-to-market.

In this pattern, it should be possible to achieve a more denormalized dataset because of the limited number of attributes per entity.

Distributed pattern

Another pattern could be having a conformed layer and enriched layer per business line context, which could be useful in cases of large enterprises where each business line realizes entities which are generic in their context, such as marketing, will focus on customer segments, campaigns, impressions, conversions, and so on, while accounting will focus on orders, sales, revenue, net income, and so on.

This is more distributed conformed layer, which is more domain driven. Because of its distributed nature, you see multiple datasets emerge for the common enterprise entities with properties where different domains are the golden source for specific properties. A way to enable discoverability of these datasets is cataloging the metadata and enabling governance for data exchange using a central data marketplace, with a data catalog and marketplace at its core.



Distributed pattern for conformed layer

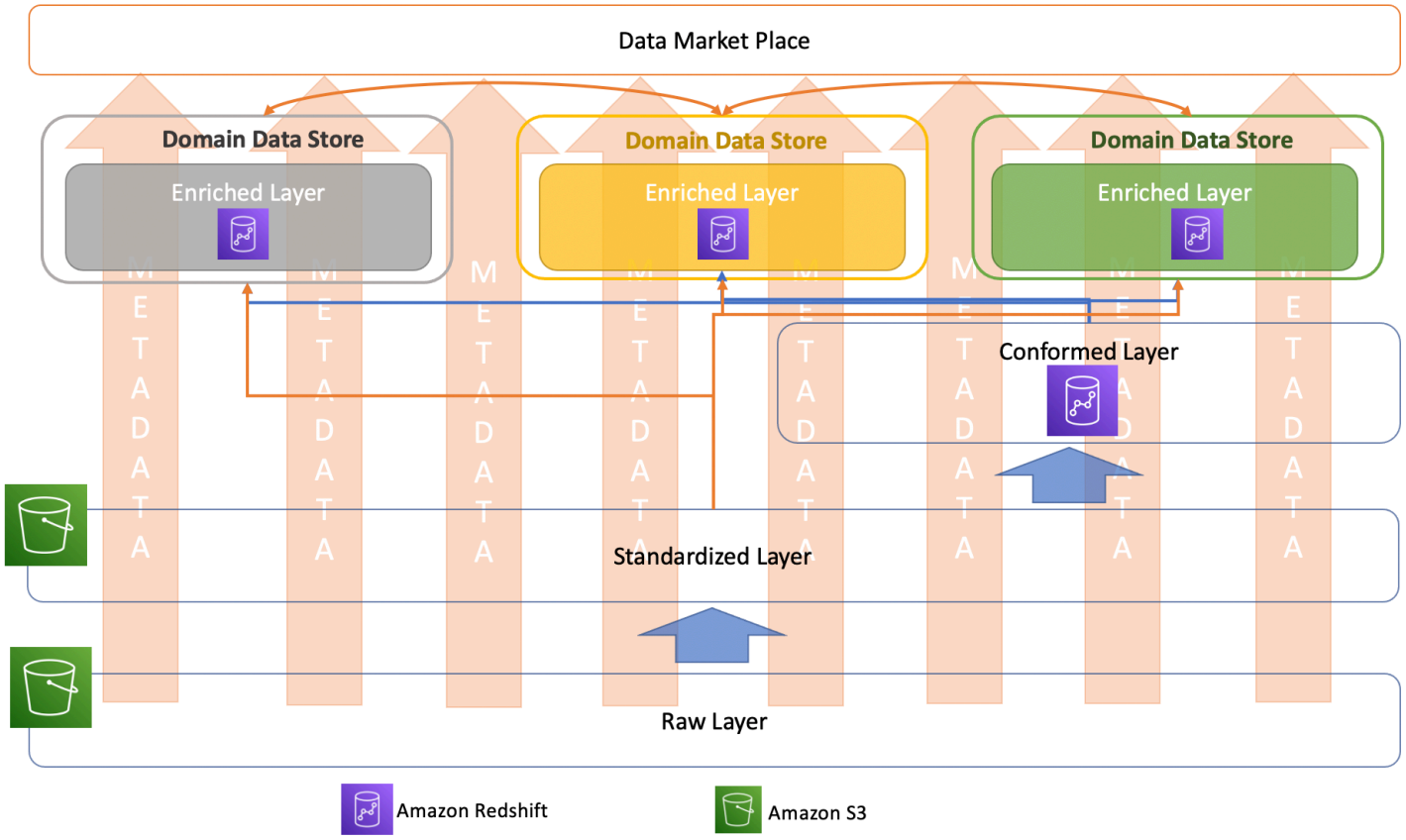
In the preceding diagram, the metadata is logged in the central data catalog and data exchange is governed using the capabilities of the marketplace data.

In both patterns, centralized and distributed, the conformed layer is not directly used for operational BI and other applications; however, it can be used for self-service BI, exploration, and analysis.

To maintain the slowly changing nature of such datasets and to optimize performance for both patterns, some suggested approaches follow:

Amazon Redshift-based approach

Based on the patterns discussed in the previous section, Amazon Redshift can be used to host conformed and enriched layers. Because it is a data warehouse solution, it allows for storing and managing data as per analytical models, and supports slowly changing dimensions.



Amazon Redshift approach for conformed layer

AWS Lake Formation governed table approach

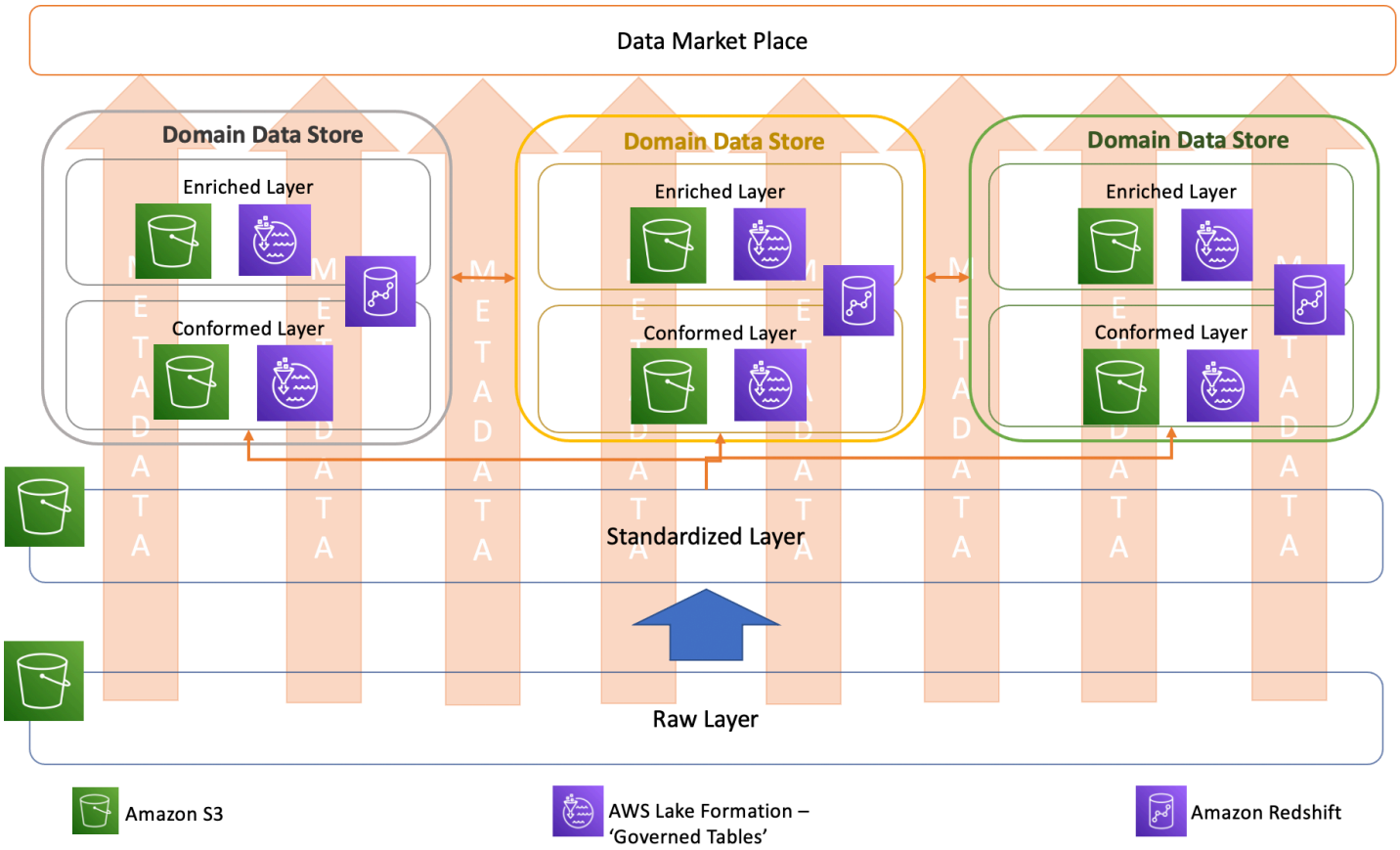
The consumers of the conformed layer or enriched layer may use different tools to analyze the data stored in those layers. These consumption patterns vary based on the use case and non-functional requirements. For business intelligence and analytical applications, having a data warehouse solution is suitable; however, for other use cases, such as a customer profile store exposed to online channels, a very low latency NoSQL solution would be more suitable.

One solution won't typically work for all the use cases. To address this, you can use the new feature of AWS Lake Formation known as governed tables.

AWS Lake Formation has introduced new APIs that support atomic, consistent, isolated, and durable (ACID) transactions using a new data lake table type, called a *governed table*. A governed table allows multiple users to concurrently insert, delete, and modify rows across tables, while still allowing other users to simultaneously run analytical queries and ML models on the same datasets that return consistent and up-to-date results. The ability to update and delete individual rows in governed tables, like a row (record) of customer data after they have asked to be forgotten, helps

users comply with *right to be forgotten* provisions in privacy laws like GDPR and CCPA. Governed tables also automatically compact smaller files to help improve performance.

This feature enables the creation of tables in a data lake (Amazon S3) and maintains history in those tables using slowly changing dimension (SCD) mechanisms. Because the transactions are ACID-compliant, the consumers get a consistent picture of the data from these tables.



AWS Lake Formation governed tabled approach for conformed layer

Enriched layer

The *enriched layer* can be seen as the layer which hosts the data product (data set). Depending on the consumption pattern for a use case, an appropriate service can be chosen to host the data to fulfill the non-functional requirements.

This is the final data product used in the business process. For example, it could be a dashboard used to make daily operational decisions or a customer profile with recommendations of products or next-best actions. The datasets can be hosted in different databases or applications, depending on the nonfunctional requirements such as latency, capabilities, scaling needs, and so on.

This data product or data set should be cataloged in the data marketplace for other users to find and request it as needed.

Need for other purpose-built data services

So far, this paper discussed the shortcomings of traditional analytics patterns, and how modern data architecture can help resolve some of those challenges. However, it has addressed only the use cases revolving around structured data that is cataloged and consumed by directly querying the data lake or the data warehouse. With the recent exponential growth of data coming from an ever-growing ecosystem of data stores, system logs, Software as a Service (SaaS) applications, machines, and Internet of Things (IoT) devices, a single type of system is not effective in meeting all business use cases.

Some of the challenges faced by organizations today are:

- Proliferating structured, semi-structured, and unstructured data.
- Performance requirements at sub-second latency.
- A large number of complex producers of data.
- Complex ETL pipelines to get the data to end state.
- Duplicate datasets across multiple systems.
- Difficulty in creating a centralized governance mechanism.

There is a wide variety of use case across organizations that need specific capabilities in terms of storing structured or semi-structured data, latency requirements, and query capabilities.

Examples of such use cases follow:

Self-service business intelligence and ad-hoc data analytics

With increasing democratization of data, the need for a petabyte-scale cloud data warehouse is evident. Users of the data warehouse want to analyze and combine the prebuilt data in the data warehouse with fresh data from the data lake, and in some cases, directly from the source of truth. This helps expedite the latest business insights to make accurate decisions.

So far in the modern data architecture pattern, this paper has placed the data warehouse at the end of the data value chain; for example, in conformed or enriched layers. These typically represent semantically transformed and enriched data with new business insights. These insights are generated by combining data from multiple sources, applying business rules, and using predictive

algorithms, leading to the final consumable data product. These steps often need movement of data via multiple ETL pipelines, which is a multi-step, multi-team effort, causing longer time-to-market. A modern data architecture platform should provide the ability to access data from disparate systems in a seamless manner from a single pane of glass.

Amazon Redshift Spectrum allows data in an Amazon S3 data lake to be analyzed from within Amazon Redshift. Amazon Redshift also provides federated queries, which allow data residing in relational databases like PostgreSQL and MySQL to be queried from within Amazon Redshift.

In a large enterprise, multiple business units often have their own data warehouse focused on a particular business domain. However, the need to share some derived insights with other business units is often required. A typical mechanism to achieve this pattern is by exporting the tables to the data lake and loading them into the destination data warehouse, resulting in copies of the data and delayed insights.

Amazon Redshift data sharing addresses this very challenge and enables sharing live tables from one data warehouse cluster to another without making copies of the data, maintaining a single source of truth.

In a modern data architecture, marketplace mechanisms also need to be in place to subscribe to external datasets for augmenting the enriched layer with third-party data. This plays a crucial role not only in improving the accuracy of the reports, but also deriving additional insights into the relationships between different entities.

AWS Data Exchange for Amazon Redshift enables you to find and subscribe to third-party data in AWS Data Exchange. You can query this data in an Amazon Redshift data warehouse in minutes. This feature empowers customers to quickly query, analyze, and build applications with these third-party datasets.

Another challenge organizations face while designing for a self-service analytics platform is the fact that data is produced and stored in a wide variety of systems, including on-premises facilities and cloud databases, along with an ever-growing list of SaaS based applications. Converging all these datasets into a single location to derive insights typically requires building complex ETL pipelines, and aligning the data to fit in the target data models. This process takes time to build, and is counterintuitive to self-service analytics. A modern data architecture should make it seamless to query data residing in these disparate systems from a single pane of glass.

Amazon Athena is an interactive query service that makes it easy to analyze data in an Amazon S3 data lake using standard SQL. If you have data in sources other than S3, you can use Amazon

Athena Federated Query to query the data in place, or build pipelines that extract data from multiple data sources and store them in S3. With Athena Federated Query, you can run SQL queries across data stored in relational, non-relational, object, and custom data sources, including SaaS applications. This ability plays a key part in enabling a self-service mechanism where data from all these disparate systems can be queried from a single place and be used in BI reports and dashboards.

Predictive analytics

Analyzing data to predict future outcomes is a key component of the overall platform. However, organizations often struggle in this area, because the AI/ML set of personas and the associated technologies are often decoupled from the main analytics projects; which creates a layer of friction. New ETL pipelines, new data quality and governance rules, and eventually new data silos emerge. A hallmark of a modern data architecture should be that predictive analytics become a seamless and integrated part of the overall analytics platform, and the tools and technologies used for analytics should agnostically support predictive analytics.

Amazon Redshift ML makes it easy for data analysts and database developers to create, train, and apply ML models using familiar SQL commands in Amazon Redshift data warehouses. With Amazon Redshift ML, you can use Amazon SageMaker, a fully managed ML service, without learning new tools or languages. Simply use SQL statements to create and train Amazon SageMaker ML models using your Amazon Redshift data, and then use these models to make predictions.

Process and log mining

Customers want to optimize their existing processes and identify rudimentary and unnecessary steps to make their business processes leaner and more efficient.

This requires loading petabytes of log and event data from various systems across the business process chain into an indexed store. Users can then analyze various steps of the process to identify patterns which make the business process inefficient.

A typical business process in a large organization has several intermediate steps (subsystems) handling part of the process chain. To analyze such business processes, both structured and semi-structured log data is often needed. Traditionally, this data had to be normalized using multi-step ETL cycles to flatten, combine, and store the data into a relational data store.

However, even after all these steps, customers are frequently unable to derive meaningful insights, because running analytics on normalized process data at scale is difficult using traditional tools. Moreover, any new information needs to be modeled and go through cycles of the ETL process change to be available for business needs, often making the whole process frustrating and error prone.

Amazon Opensearch Service is a fit for purpose solution for such use cases, because you can index all your structured and semi-structured log data into a single index store, and run search queries to combine this data at scale and identify patterns. As new data comes in, you naturally enhance your search queries to immediately include such insights, reducing time to market and improving the experience of your analysts.

User profile store for personalization

Businesses want to better engage with their customers by providing tailor-made recommendations of product and services based on the current and previous patterns of customer preferences.

To achieve such a high degree of personalization, data from various channels of user interaction needs to be combined with existing data about the customer. Typically, clickstream data from web and mobile channels, call scripts from call centers, chat scripts from chat bots, and other data from after sales and service feedback tools and social media needs to combined, related, and enriched with users' data from system of record to identify various properties of the users. These properties help determine the best actions and recommendations for the user, which can be fine-tuned using recent behavior.

This set of properties can be called a user profile.

Storing this profile in a traditional RDBMS is challenging, because on large ecommerce web channels, there can be tens of thousands to millions of users, each with a varying set of attributes associated with their profile. To meet such large scale and constantly changing attributes, these user profiles must be stored in a highly scalable and flexible sub-second latency data store, to support the millions of API calls of the personalization engine for profile lookups.

Amazon DynamoDB delivers this performance, with millions of API call lookups in sub-second latency at any scale.

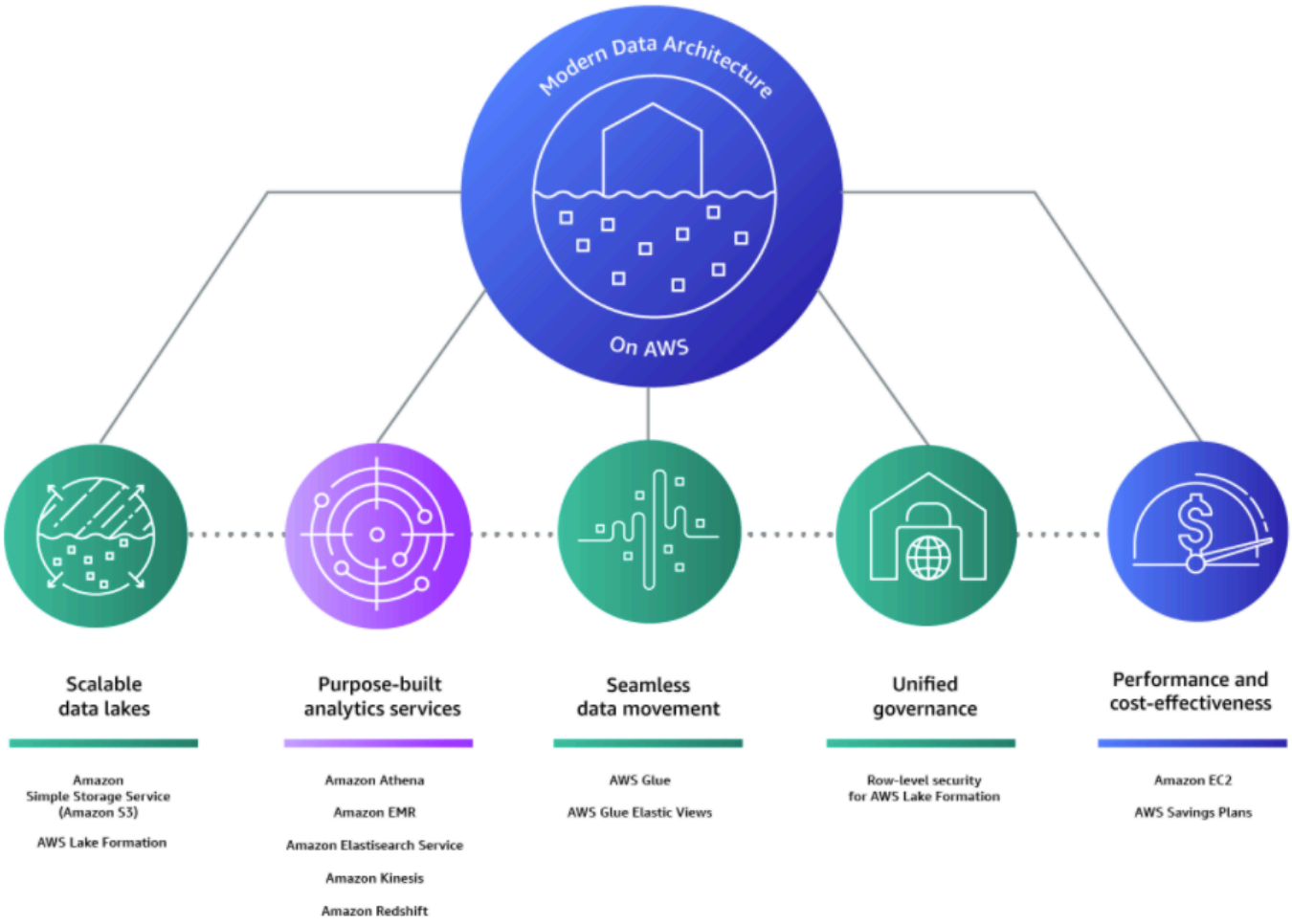
Modern data architecture on AWS

A modern data platform should address everything discussed in this paper so far, and offer features that enable different personas in an organization to build mechanisms for end users to consume the data in appropriate ways.

An AWS modern data architecture has 5 pillars:

- Scalable data lakes
- Purpose-built analytics services
- Unified data access
- Unified governance
- Performant and cost-effective

With a modern data architecture on AWS, customers can rapidly build scalable data lakes, use a broad and deep collection of purpose-built data services, ensure compliance with unified data access, security, and governance, scale their systems at a low cost without compromising performance, and easily share data across organizational boundaries, allowing them to make decisions with speed and agility at scale.



Modern data architecture on AWS

Further details on this modern data architecture on AWS, including patterns and other services and features, can be found in the resources mentioned in the Further reading section of this document.

Conclusion and contributors

Conclusion

As businesses evolve, the data they generate becomes more valuable and complex. Deriving timely insights from this data becomes critical for organizations to maintain their competitive edge in the market. A modern analytics platform caters to this need.

This whitepaper explored how the traditional approaches to data analytics fall short of growing business expectations. It introduced the concept of modern data architecture and how it helps solve new analytics use cases, how it makes the platform agile, and the data lifecycle seamless. It also highlighted how AWS analytics services help bring this modern data architecture to life.

Contributors

Contributors to this document include:

- Behram Irani, Sr. Solutions Architect, Amazon Web Services
- Abhishek Choudhary, Sr. Data Architect, Amazon Web Services

Further reading

Refer to the following resources for further insights in the modern data architecture:

- Modern Data Architecture on AWS
- Derive Insights from AWS Lake House (whitepaper)
- Build a Lake House Architecture on AWS (blog entry)
- Harness the power of your data with AWS Analytics (blog entry)

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Updated	First publication	February 15, 2022

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser you are using.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.