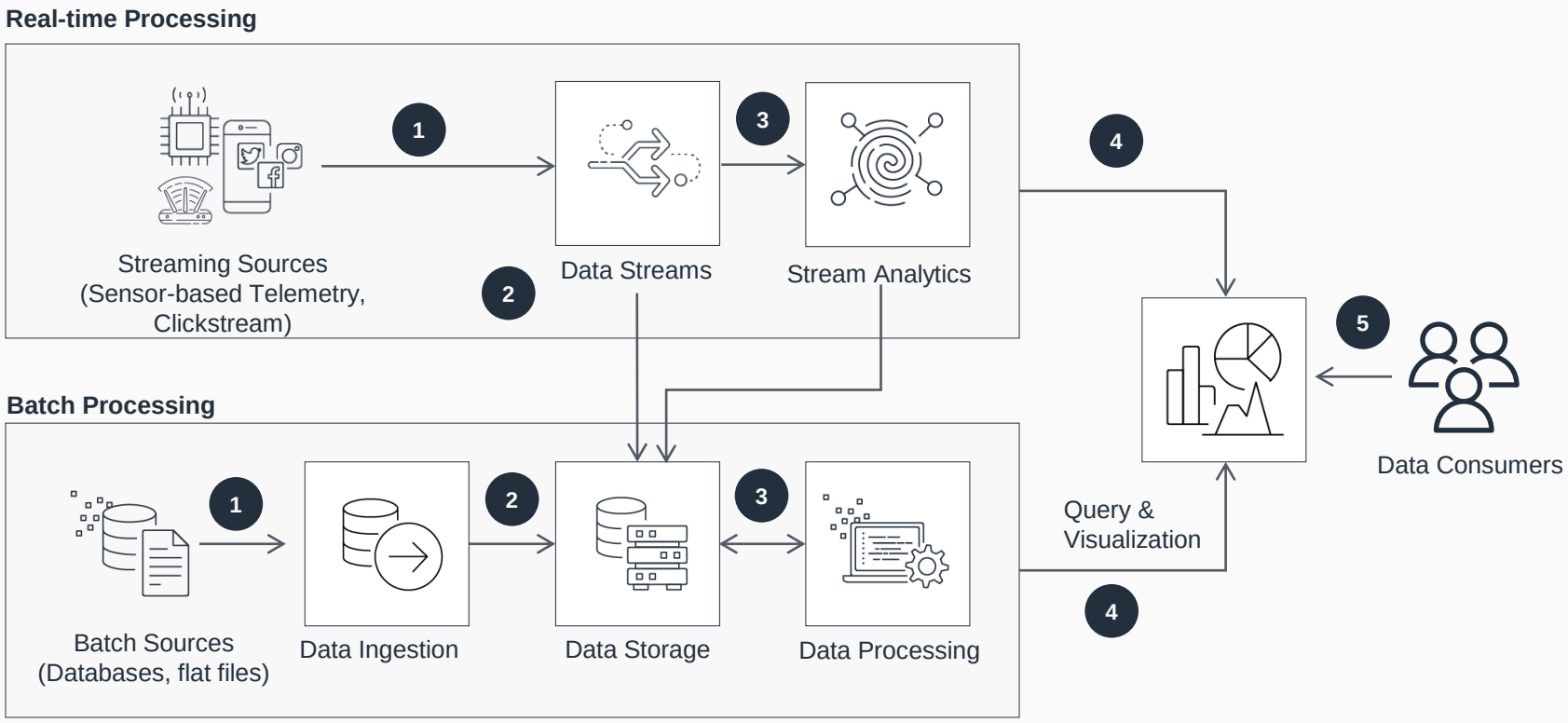
Guidance for Optimizing Data Architecture for Sustainability on AWS

Overview

An end-to-end data architecture delivering insights on both real-time and batch data.



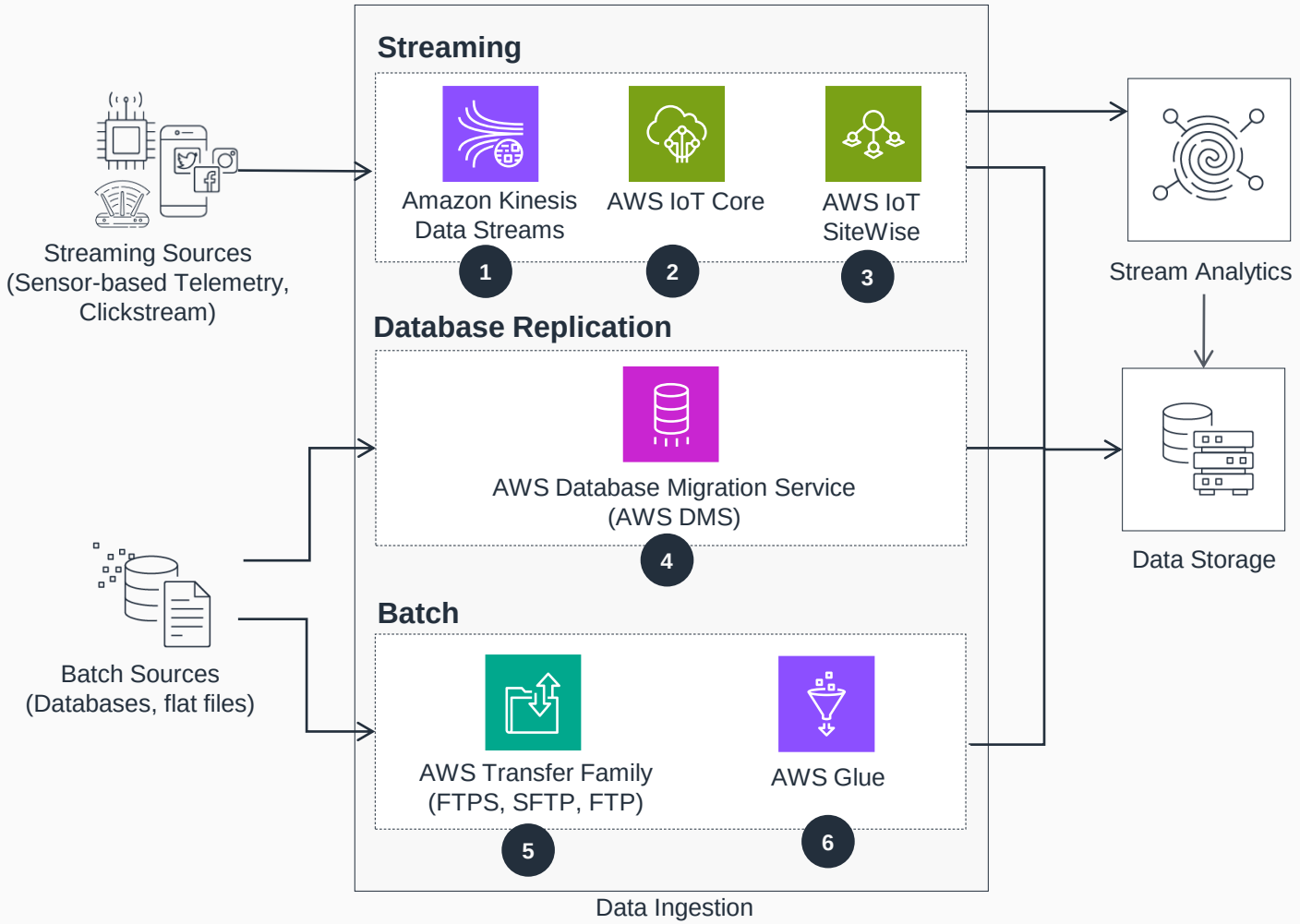
1. Organizations ingest data from streaming sources like sensors, devices, social media, or web applications, and in batches from database and file systems.
2. Streaming event data from data streams is stored for longer retention. Data from databases and file systems are stored in a colder storage layer for transformation and consumption.
3. A stream analytics system analyzes, filters, and transforms incoming data streams in real-time. The batch data processing layer transforms raw data by cleaning, combining, and aggregating the data for analytical purposes.
4. Streaming data is sent to downstream systems for consumers to query and visualize in real-time. Batch data is modeled and served for consumption for business intelligence.
5. Data consumers use query and visualization tools to analyze the data.

Reviewed for technical accuracy August 27, 2024

AWS Reference Architecture

Data Ingestion

This diagram shows a real-time and batch data ingestion pattern and a database replication pattern, along with with recommended AWS services that serve these capabilities.



1. Use managed services such as **Amazon Kinesis Data Streams** for streaming data, such as clickstream data from web applications. Managed services distribute the environmental impact of the service across many users because of the multi-tenant control planes.

2. Use managed services such as **AWS IoT Core** for consumer Internet of Things (IoT). It ensures that resources are used efficiently and distributes the environmental impact of the services across the user pool.

3. Use managed services such as **AWS IoT SiteWise** for industrial IoT streaming data. It ensures resources are used optimally and distributes the environmental impact of the services across the user pool.

4. Choose the right-sized **AWS Database Migration Service** (AWS DMS) instance type.

5. To reduce the waste of resources and to maximize the utilization through automation, use AWS Transfer Family managed workflows. It automates various processing steps such as copying, tagging, scanning, filtering, compressing or decompressing, and encrypting or decrypting the data that is transferred using **AWS Transfer Family**.

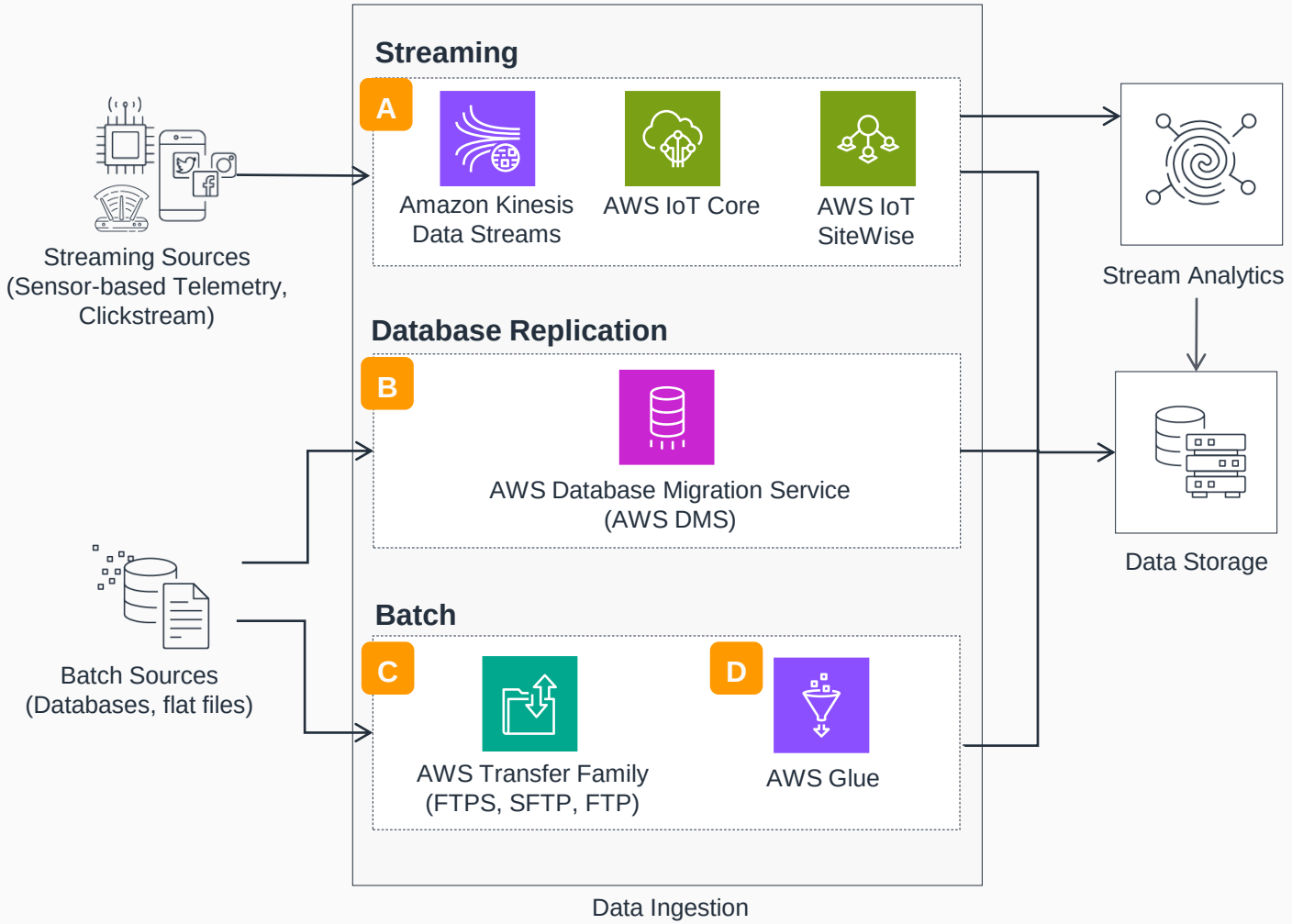
6. Use managed services like **AWS Glue** for better resource utilization and leverage AWS Glue Flex jobs for non-urgent ingestions.

Reviewed for technical accuracy August 27, 2024

AWS Reference Architecture

Data Ingestion

This diagram shows additional considerations for data ingestion. It includes a real-time and batch data ingestion pattern and database replication pattern with possible AWS services that serve these capabilities.



A **Consideration A**
Evaluate the service-level agreement (SLA) for data delivery and assess whether a continuous stream is necessary.

B **Consideration B**
Evaluate whether a workload is memory-intensive or CPU-intensive, and size resources accordingly.

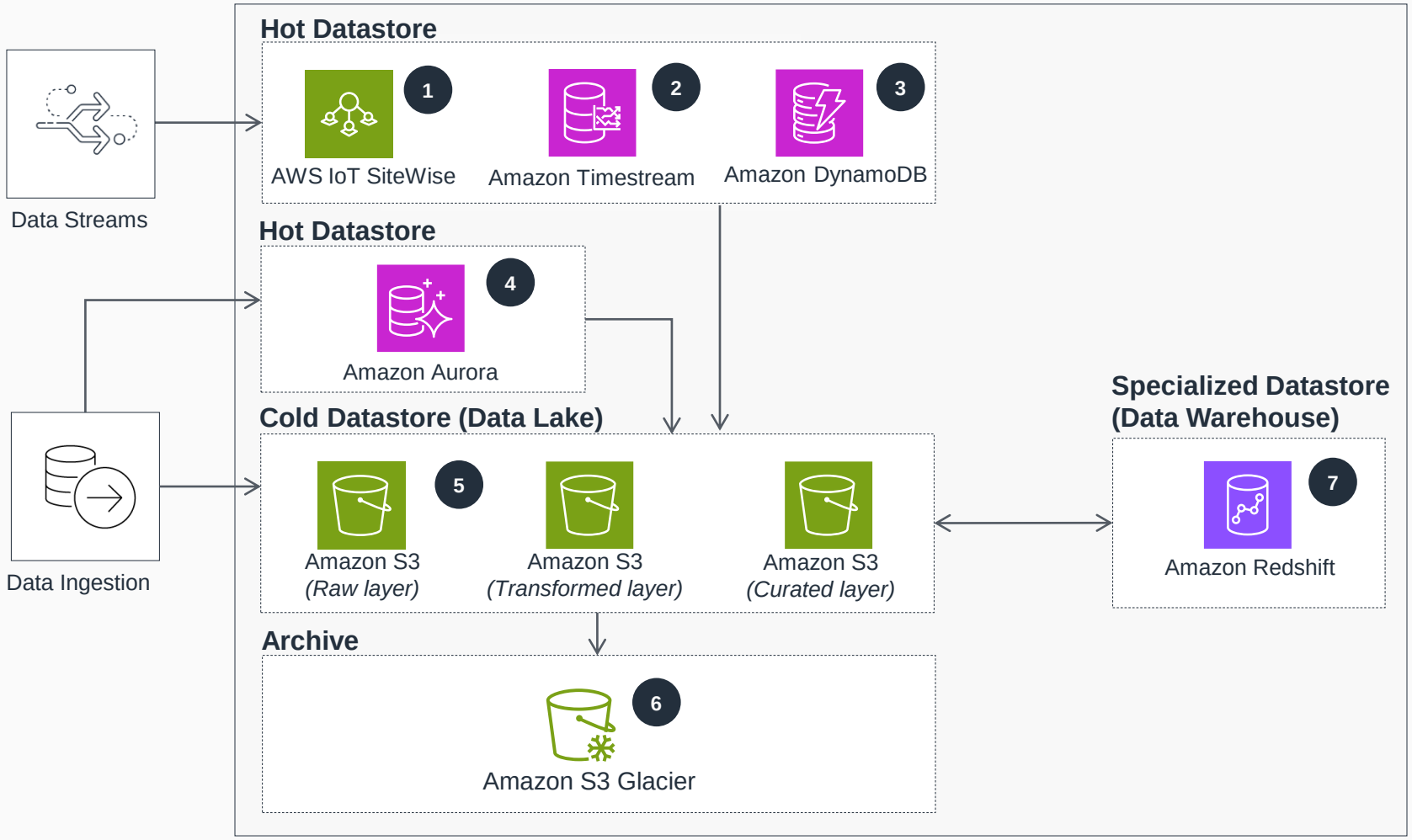
C **Consideration C**
Ingest valuable data from the source systems by filtering out data items that are of no value to consumers and remove redundant ingestion pipelines. Adopt an event-driven serverless architecture for your data ingestion, so it only provisions resources when work needs to be done.

D **Consideration D**
Review scheduled ingestions and remove unnecessary operations or reduce the operation's frequency where possible. Spread batch ingestion schedules over the day rather than running all at once overnight. This helps with reducing the provisioning of resources for peak usage when most batch jobs are scheduled overnight.

Reviewed for technical accuracy August 27, 2024

Data Storage

This diagram shows the storage layer with frequently accessed data stores for operational use, and two popular storage patterns for analytics use – the data lake and the data warehouse.



1. Use the managed database of **AWS IoT SiteWise** for storing data from industrial equipment at scale. Set a retention period for how long your data is stored in the hot tier before it's deleted. Move historical data to colder tiers in **Amazon Simple Storage Service (Amazon S3)**.

2. Use a fully managed, purpose-built time-series database to store time-series data. **Amazon Timestream** saves time and cost in managing the lifecycle of time-series data by keeping recent data in memory and moving historical data to a cost optimized storage tier based upon user-defined policies.

3. Consider using serverless databases for unpredictable or irregular workloads. For example, **Amazon DynamoDB** for non-relational data.

4. Consider using serverless databases for unpredictable or irregular workloads. For example, **Amazon Aurora** for relational data.

5. Consider using object storage such as **Amazon S3** to store large volumes of data. Move infrequently accessed data to colder tiers for energy efficiency. Use the **Amazon S3 Intelligent Tiering** storage class to optimize storage by automatically moving data to the most cost-effective access tier when access patterns change.

6. Use Amazon S3 Glacier storage classes for archiving data not queried frequently.

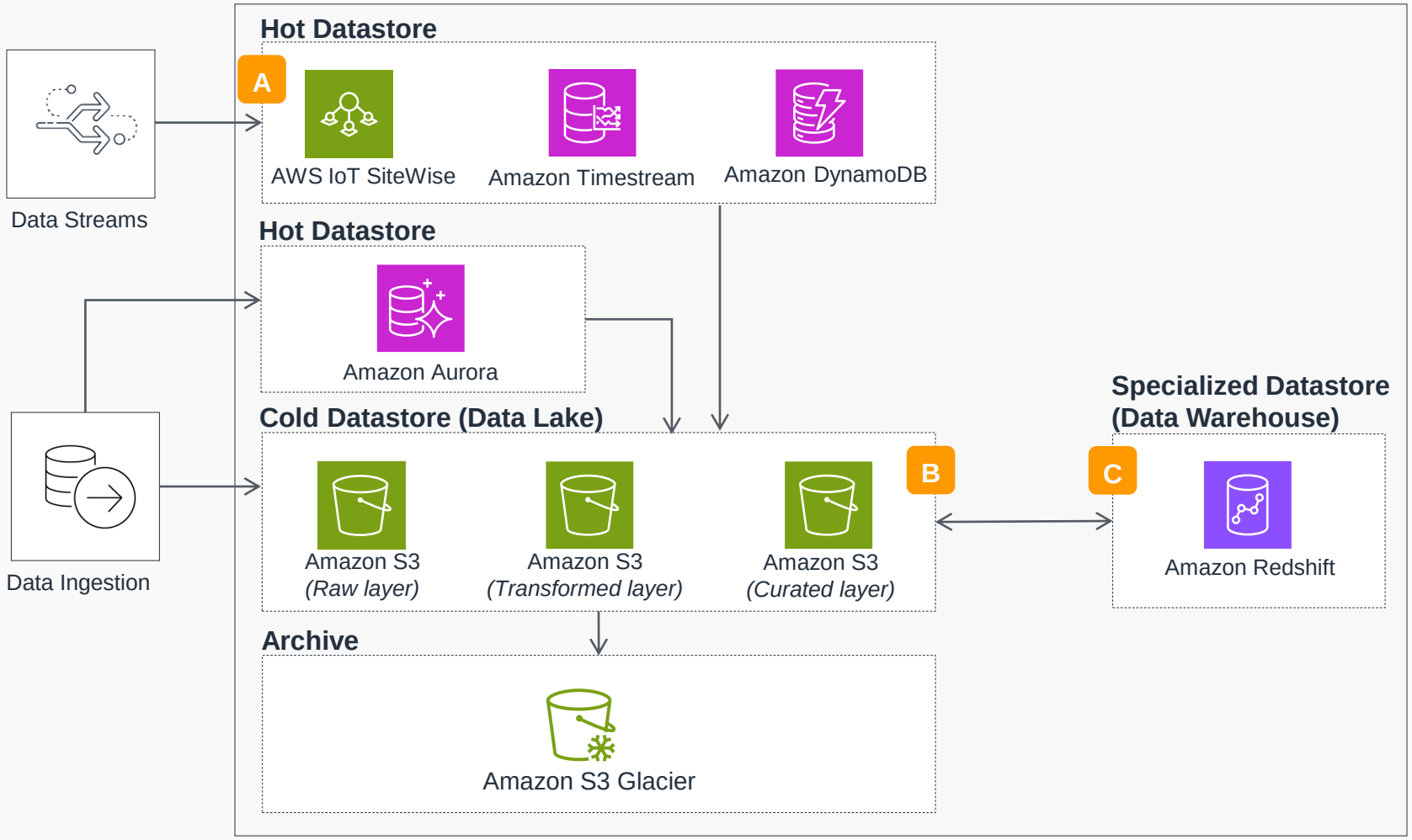
7. Use serverless services like **Amazon Redshift** Serverless for efficient utilization of resources and reduce engineering overhead on unpredictable or irregular data warehouse workloads. Use features in **Amazon Redshift** that provide automation, like Automatic Table Optimization (ATO).

Reviewed for technical accuracy August 27, 2024

AWS Reference Architecture

Data Storage

This diagram shows additional considerations for the data storage layer. It includes frequently accessed data stores for operational use and two popular storage patterns for analytics use – the data lake and the data warehouse.



Data Storage

A **Consideration A**

For hot data stores where data is queried frequently, choose the right database service for the right purpose to improve resource efficiency of your workloads. Right-size your database infrastructure to reduce wasting resources and use energy-efficient processors like Graviton where possible. Automate database backups so you don't have to take manual snapshots of your databases.

B **Consideration B**

When data is stored in large volumes in a data lake, reduce storage footprint by compressing data and deleting unused data. Classify data to understand its significance to business outcomes to determine when you can move data to colder storage layers or safely delete it. Use efficient file formats suited for consumption patterns (like **Parquet** for certain analytics use-cases) to improve utilization of downstream compute. Practice partitioning and bucketing design standards to reduce the amount of data scanned per query by downstream compute.

C **Consideration C**

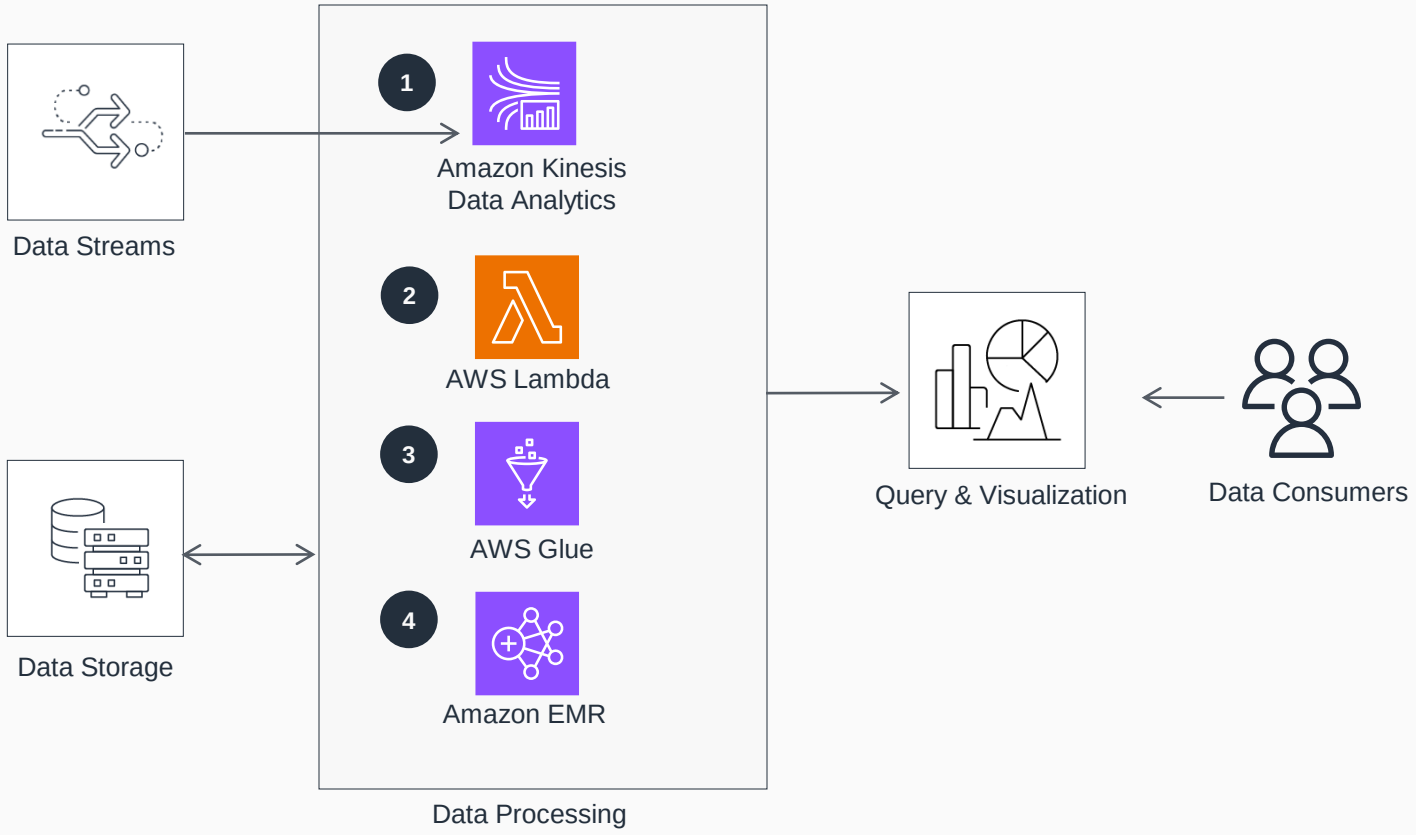
Follow data warehouse table design best practices like choosing suitable sort keys and distribution styles based on workloads. Use appropriate data types for the columns; for example, use date/time data types for date columns.

Reviewed for technical accuracy August 27, 2024

Data Processing

This diagram shows the data processing layers with different AWS services that could be used to process data in real-time or in batch processing mode. Use either managed services (option 1) or self-managed (option 2) as shown in subsequent slides.

Option 1: Managed Services



1. Use managed services like **Amazon Kinesis Data Analytics** to reduce overhead of managing infrastructure and risk of overprovisioning resources. Select the appropriate length for time-based windowing operations for streaming applications to reduce wastage of resources.

2. Use serverless services like **AWS Lambda** for the appropriate use. Design workloads to limit **Lambda** invocations to reduce resource usage. Run new and existing functions on Arm-based AWS Graviton2 processors.

3. Use **AWS Glue** for unpredictable batch data processing workloads on Apache Spark, Python shell, or Ray engines. Choose appropriate worker nodes for the job. Run non-critical jobs on **AWS Glue** Flex.

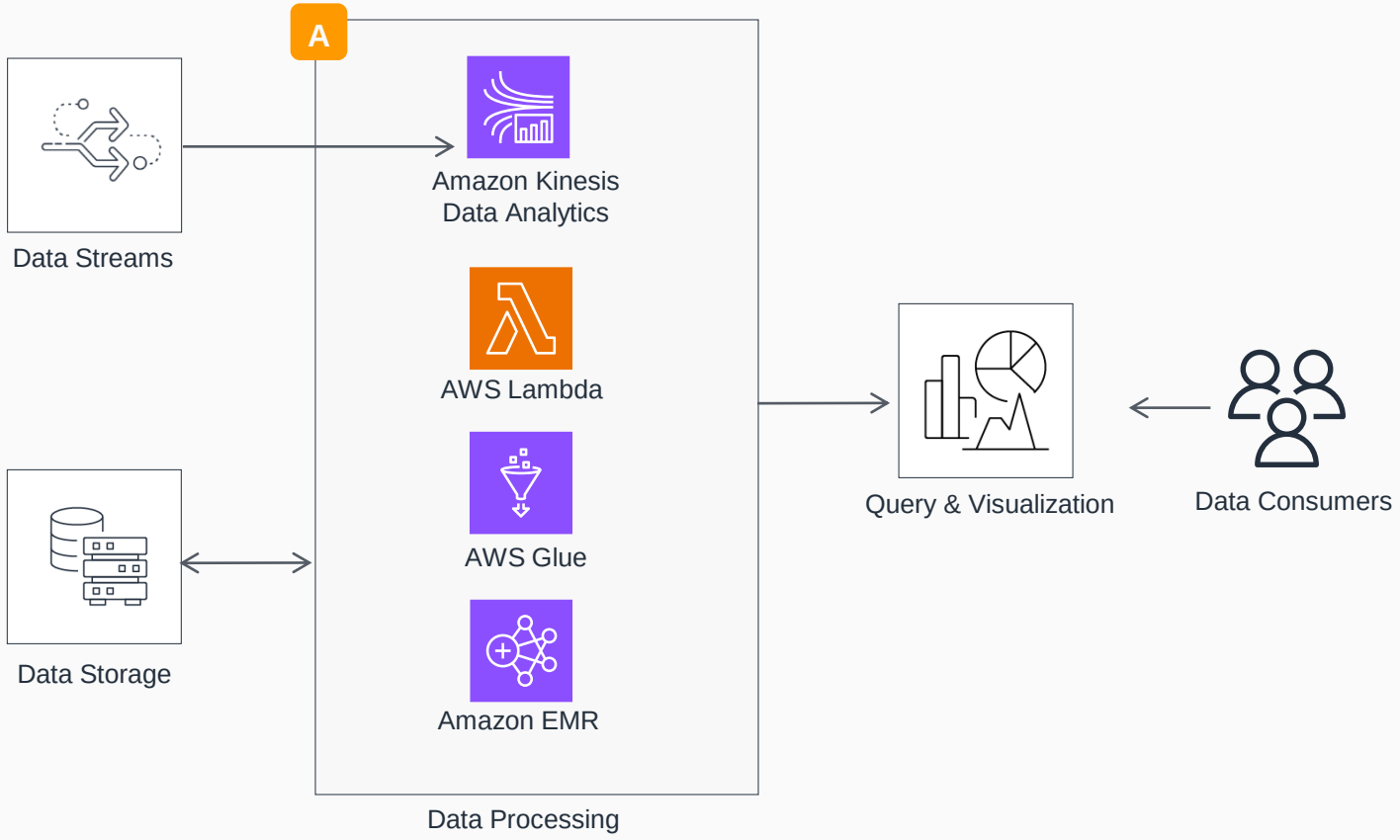
4. Run petabyte-scale data processing jobs on open-source frameworks such as Apache Spark, Apache Hive, and Presto using **Amazon EMR.** For unpredictable data processing workloads, use **Amazon EMR Serverless**. For **Amazon EMR** running on **Amazon Elastic Compute Cloud** (Amazon EC2) instances, consider using AWS Graviton-based Amazon EC2 instances, which use up to 60% less energy than comparable EC2 instances for the same performance. Design clusters to terminate on job completion for batch workloads. Consider using **EC2** Spot Instances for non-critical workloads. Leverage **Amazon EMR** managed scaling to automatically size cluster resources based on the workload for best resource utilization.

Reviewed for technical accuracy August 27, 2024

Data Processing

This diagram shows an additional consideration for the data processing layer. It includes different AWS services that could be used to process data in real-time or in batch processing mode.

Option 1: Managed Services

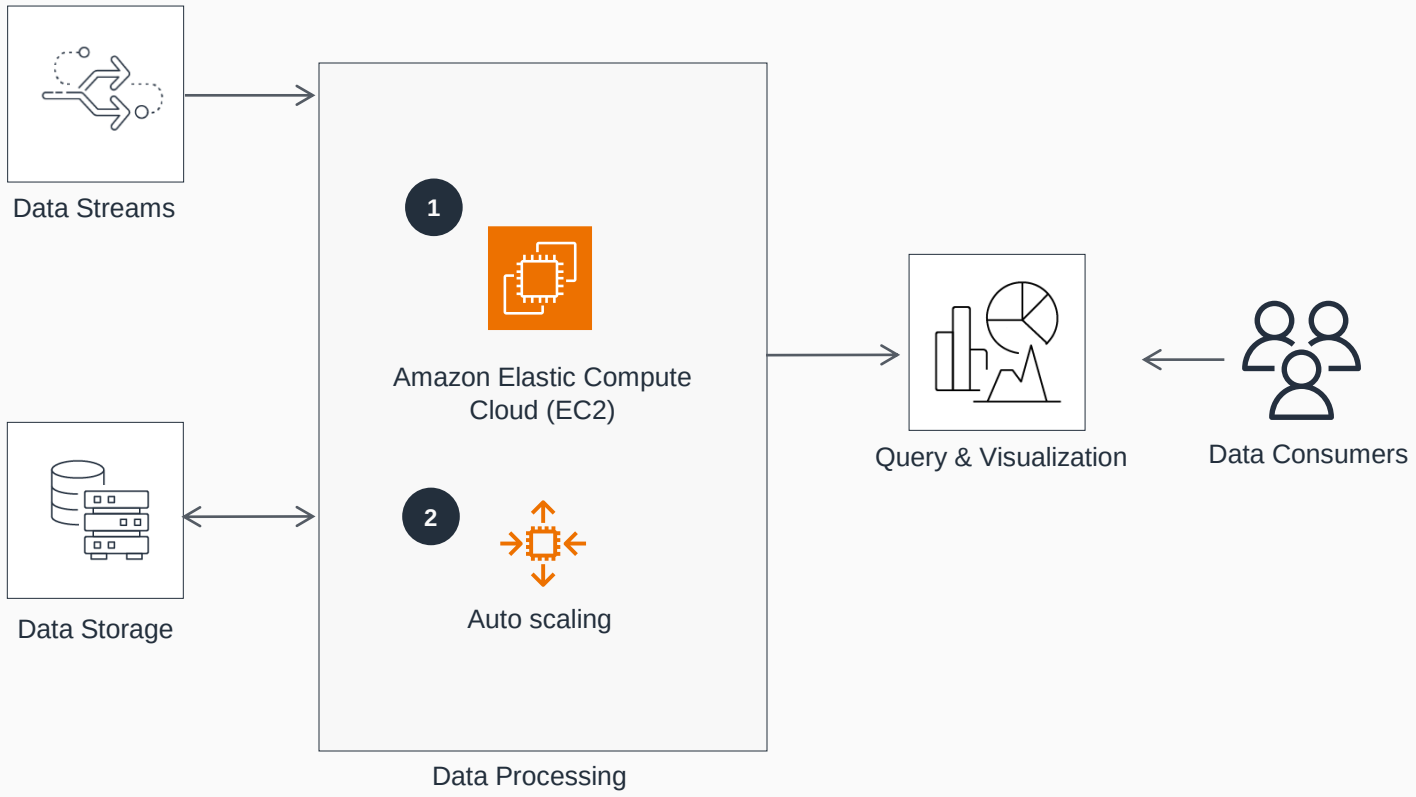


A **Consideration A**
Use predicate pushdown to reduce the amount of data moved between different layers during data processing. Implement an event-driven architecture to maximize overall resource utilization for asynchronous workloads.

Data Processing

This diagram shows the data processing layers with different AWS services that could be used to process data in real-time or in batch processing mode.

Option 2: Self-Managed

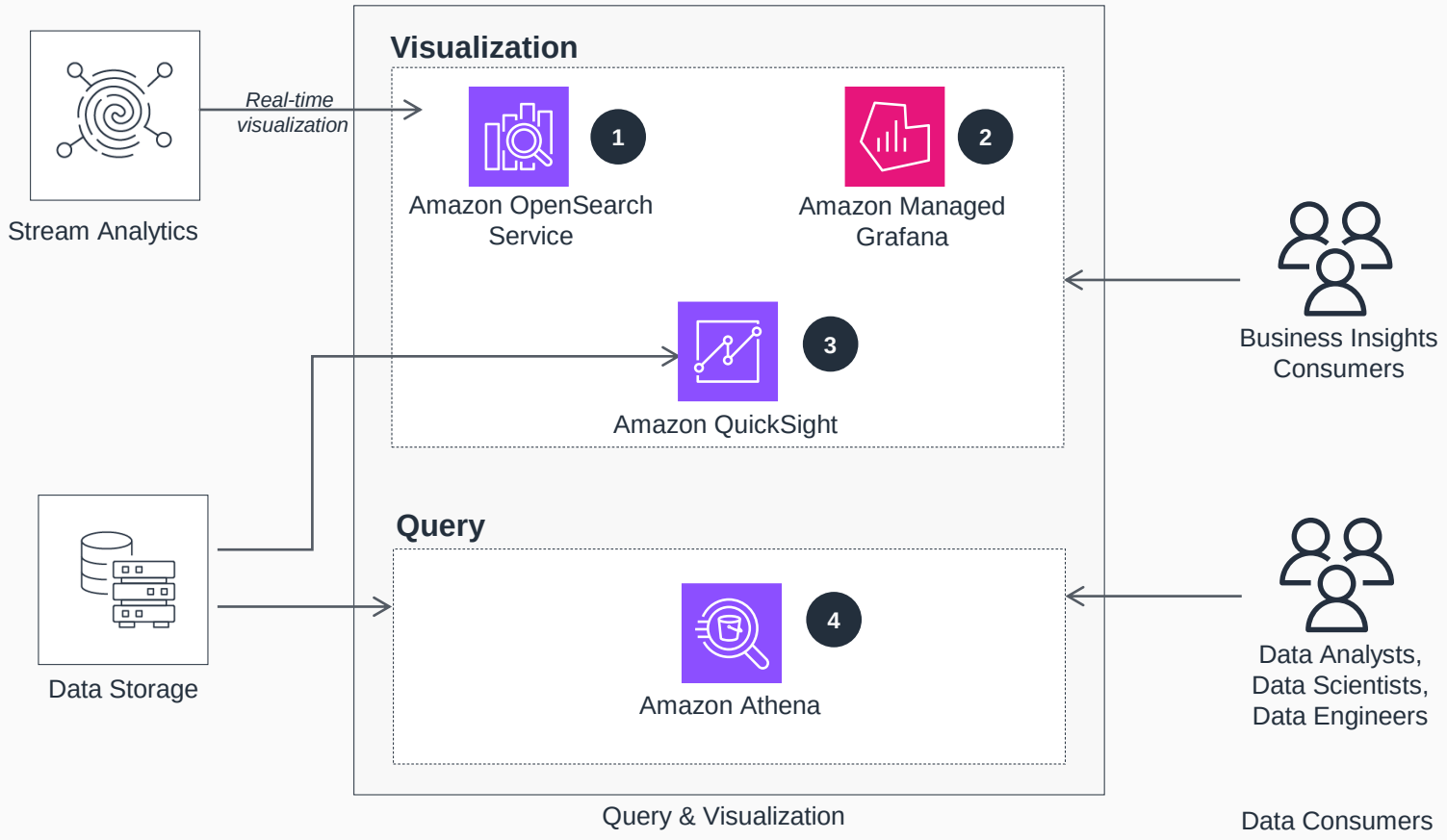


1. Use **Amazon Elastic Compute Cloud** (Amazon EC2) instances to build your own analytics application that is best suited for your business requirements. Run petabyte-scale data processing jobs on open-source analytics frameworks such as Apache Airflow, Apache Hive, Apache Kafka, Apache Flink, Apache Spark, Presto, and Trino, among others. Choose Graviton-based EC2 instances to reduce energy consumption for the same performance. AWS Graviton-based EC2 instances use up to 60% less energy than comparable EC2 instances for the same performance.

2. Consider using Amazon EC2 Auto Scaling to launch new EC2 instances seamlessly and automatically when demand increases, and terminate unneeded instances automatically to reduce wastage and save money when demand subsides.

Data Consumers

This diagram shows the data query and visualization layer with different AWS services that helps users to query and visualize data.



1. Use **Amazon OpenSearch Service** for real-time visualization. **Amazon OpenSearch Serverless** removes much of the complexity of managing **OpenSearch** clusters and capacity. It automatically sizes and tunes your clusters and takes care of shard and index lifecycle management, ensuring optimal utilization of resources. If you are using *provisioned* **OpenSearch** clusters, consider using instances with AWS Graviton processors.

2. Leverage the rich, interactive data visualizations in **Amazon Managed Grafana**. It is a multi-cloud, cross-project service used to analyze, monitor, and alarm on metrics, logs, and traces across multiple data sources. The infrastructure is fully managed by AWS, which helps reduce the environmental impact of the service.

3. Deliver insights to all your users through visualizations using the serverless, scalable business intelligence service, **Amazon QuickSight**. Use in-memory caching mechanisms like **Amazon QuickSight** SPICE to reduce contention with source data stores and improve query efficiency. Use built-in machine learning insights in **QuickSight** to eliminate the need of setting up separate machine learning pipelines and natural language processing (NLP) features like **Amazon Q in QuickSight** to reduce the need to develop visuals.

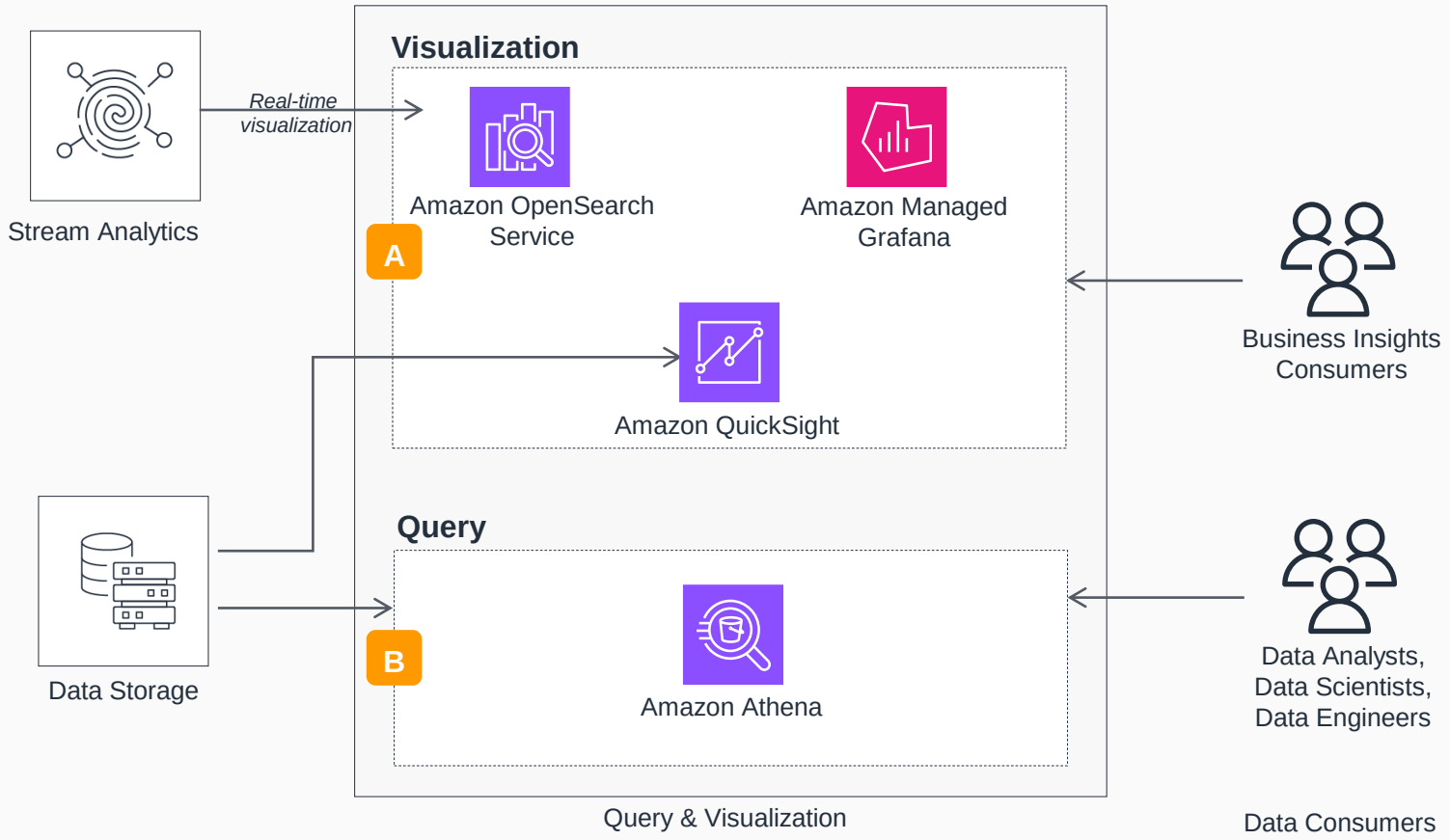
4. Implement data virtualization techniques to query information about where the data resides and avoid data movement. For example, use the federated querying feature in **Amazon Athena** while querying external data stores.

Reviewed for technical accuracy August 27, 2024

AWS Reference Architecture

Data Consumers

This diagram shows additional considerations for the data query and visualization layer. It includes different AWS services that helps users to query and visualize data.



A **Consideration A**

Consider reviewing reports and dashboard usage at regular intervals. Remove unused, redundant reports from dashboards.

B **Consideration B**

Optimize data consumption by reducing physical data movement to consuming systems. Use pre-compiled views to improve resource utilization while querying data. Identify and optimize long-running, resource-intensive queries. Use result-caching techniques to reduce I/O operations.

Reviewed for technical accuracy August 27, 2024