Building Data Architecture Using Amazon Web Services

Published: 9 June 2020 **ID:** G00451433

Analyst(s): Sumit Pal, Sanjeev Mohan

Amazon Web Services provides extensive capabilities to build scalable, end-to-end data management solutions in the cloud. This document helps data and analytics technical professionals select the right combination of solutions and products to build an end-to-end data management platform on AWS.

Key Findings

- AWS is a strong primary strategic cloud provider for a broad range of data and analytics use cases and workloads.
- AWS provides a mature platform with proven stability, scale and performance of services. It has demonstrated over the years that it can operate its data and analytics tools while adhering to published SLAs.
- AWS has a broad range of services and appeals to customers across many industries. The breadth and number of possible combinations of offerings from AWS can be daunting, so keep up with new announcements and updates to get the most from your AWS investment.

Recommendations

Technical professionals designing data management solutions should:

- Choose available native tools from the ecosystem carefully, especially data integration and data ingestion. Each tool provides a pattern that is best fit for a certain set of use cases, with overlap.
- Evaluate requirements with what is currently supported and what is on the roadmap before choosing. Some services like AWS Glue or AWS Lake Formation are new, and additional features are continuously being added.
- Strengthen in-house AWS-specific skills. AWS will let you build and scale systems to the size of the largest organization, but the complexity of the services when used at scale requires significant depth of knowledge, skill sets and experience.

- Use orchestration tools like AWS CloudFormation and AWS Glue, and monitoring services like AWS CloudWatch to enforce DevOps and DataOps principles with focus on automation and CI/CD. Supplement native tools with marketplace or community products when developing an end-to-end data platform to satisfy orchestration, scheduling, deployment and monitoring capabilities.

Table of Contents

List of Tables

List of Figures

Analysis

This document was revised on 22 June 2020. The document you are viewing is the corrected version. For more information, see the Corrections page on gartner.com.

Amazon Web Services (AWS) offers a wide set of data services to implement a comprehensive, end-to-end data management platform. Some of these services are new and some have been battle-tested. AWS cloud platform offers compelling data ingestion, storage, and processing and analytics offerings with which enterprises can innovate for different use cases across a wide spectrum of data workloads.

No single vendor provides an automated, integrated and orchestrated out-of-the-box way to build an end-to-end data management platform. While cloud vendors like AWS give customers an abundance of in-house data processing options — and an even wider array of third-party solutions via cloud marketplaces — it's still up to the customer to connect all the dots.

While cloud usage deepens, customers should ensure they have proper cost management policies in place. Use AWS native tools like Trusted Advisor and Cost Explorer to identify waste and manage costs. Purchase Reserved Instances or use the AWS Spot Market to lower individual resource expenses. An organization's cloud bill can rapidly snowball if resource usage and utilization are left uncontrolled without proper governance.

Architectural Best Practices for Cloud

Organizations that architect their data platforms on the cloud can reap huge benefits by leveraging some of the architectural principles and best practices for cloud computing discussed below:

- Don't build in the cloud, but build for the cloud. Use cloud-native capabilities and treat the cloud as an operating system, rather than just using a few services in an ad hoc manner. Focus on essential capabilities required in the architecture and let that dictate the services needed from the cloud service provider instead of the cloud provider dictating the architecture.
- "Forklifting" an existing data and analytics stack "as is" from the data center to the cloud can result in some costs and higher business agility, yet it may leave organizations with a stagnant solution hosted in a different place. To truly improve your solutions, costs and results, consider your architecture carefully, use managed services where possible, and iterate on your design to gain efficiency and to lower costs.
- Embrace the modern cloud data stack by moving away from managing your own storage and processing clusters, and embracing serverless data offerings like Amazon Athena and others. To face the challenges of exponential future growth and the rapid evolution of business, it is not enough to manage your own open-source software cluster on cloud hardware. Use of fully managed and serverless solutions allow your precious IT resources to focus on solving tough business problems.
- Invest in data orchestration systems like Alluxio, which provide data-driven applications the ability to read and write data without worrying about where and how the data is stored. This is akin to a container orchestration system that provisions compute resources without worrying

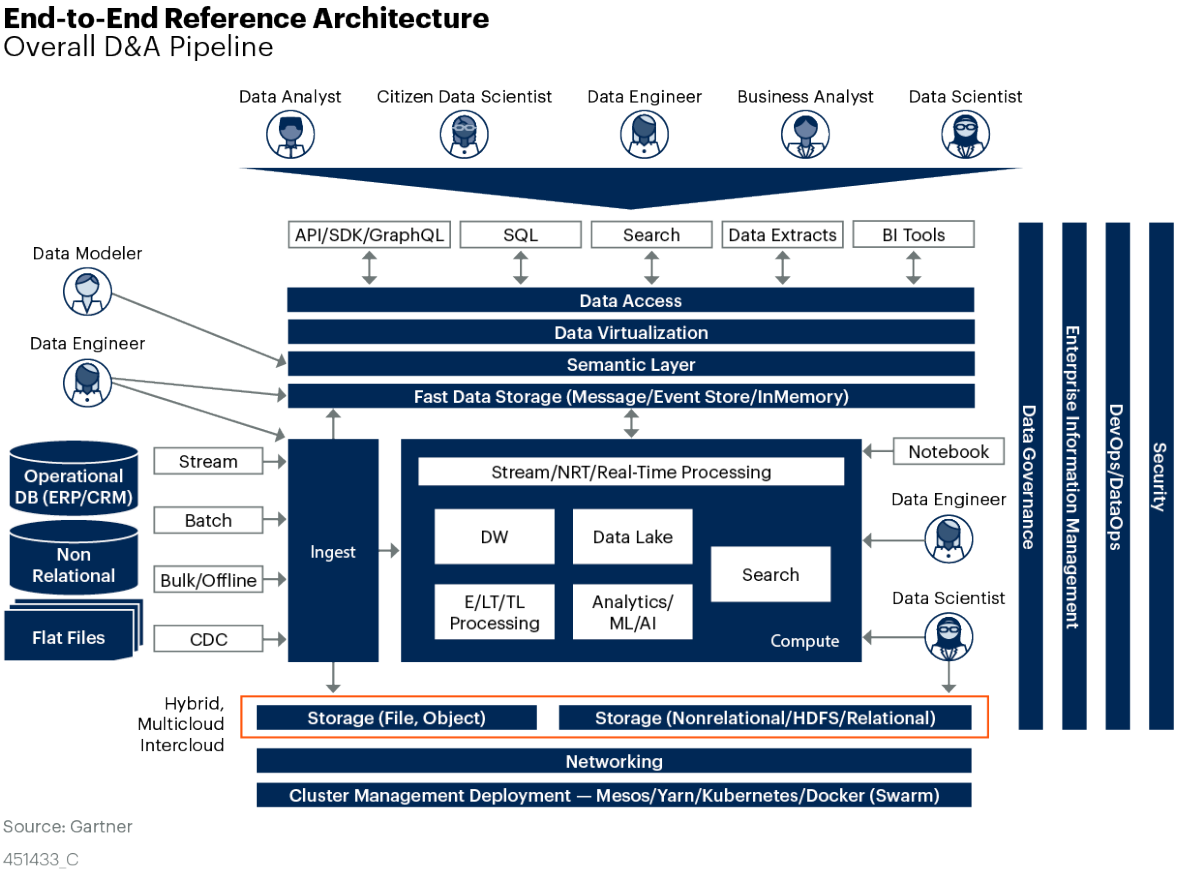
about where and how the compute resource is allocated. Similarly, use tools like AWS Glue, AWS Step Functions, or Apache Airflow to automate the movement and transformation of your data-driven workflows.

- Use continuous monitoring (CM) to monitor resource usage and costs with cloud-native or third-party tools to ensure there are no cost surprises.
- Use the right tool for the right job. Understand data structures, latency, throughput, concurrency and access patterns before choosing the service, whether cloud-native or from the marketplace.
- Manage costs — big data should not equal big cost. Monitor not just resource uptime but costs across the end-to-end data platforms. Tools like billing alerts and cost allocation tags can help. Define and automate policies to terminate inactive resources. Be familiar with the cost controls and features AWS provides for continuous cost monitoring.
- The cloud offers native security services that can help with identity and access management, data protection, threat detection and incident response. Customers can use DevSecOps practices and build automation around security controls, allowing them to focus on both agility and security.
- Adopt cloud-native technologies by adopting an everything-as-code approach from infrastructure to applications and platforms. Use configuration driven orchestration and invest in DataOps- and DevOps-based approaches of building and running data platforms.

The cloud model is great for agility and time to deployment, and works well for both ephemeral and long-running workloads. With options like reserved pricing and savings plans, you can predict the cost structure for your deployments and long-running workloads. For workloads that must reside on-premises due to regulatory or other requirements, customers are now looking to bring cloud to the data, for example, using AWS Outposts to leverage cloud capabilities on-premises. The cloud is not an escape valve. Even with the cloud, it's still an organization's responsibility to put together all the components and make it work in an automated, predictable and scalable manner.

Gartner has developed an end-to-end data and analytics reference architecture, shown in Figure 1.

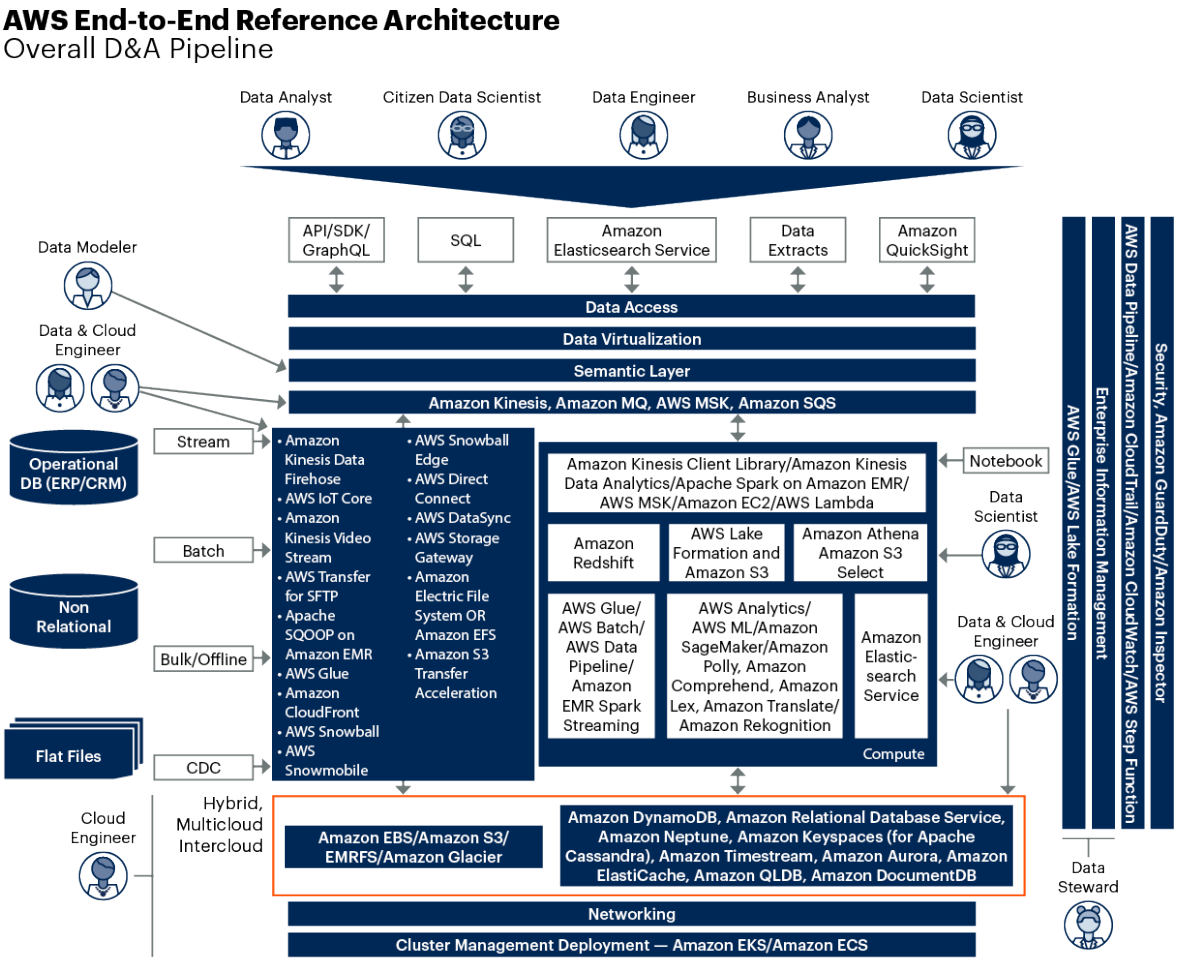
Figure 1. End-to-End Reference Architecture



Source: Gartner
451433_C

AWS offers comprehensive native and third-party services to meet varying end-to-end data management needs. Figure 2 shows sample components of AWS data management and analytical offerings to deliver a data platform. It shows the capabilities of the AWS ecosystem for ingestion, data integration, transformation, data storage, data security, governance, analytics and data science.

Figure 2. AWS End-to-End Reference Architecture



Source: Adapted From Amazon

451433_C

Data Transfer and Ingestion

The ingestion and integration layers are about getting data into the AWS ecosystem. The ingestion layer is responsible for connecting both pull- and push-based data sources and extracting data. With the rapid proliferation of a variety of data sources, organizations want to ingest a wide range of datasets both from internal and external data sources to enrich their data and analytics platform. The ingestion layer thus must be agile, adaptable, flexible and robust to support a wide variety of data sources, data formats and transfer protocols.

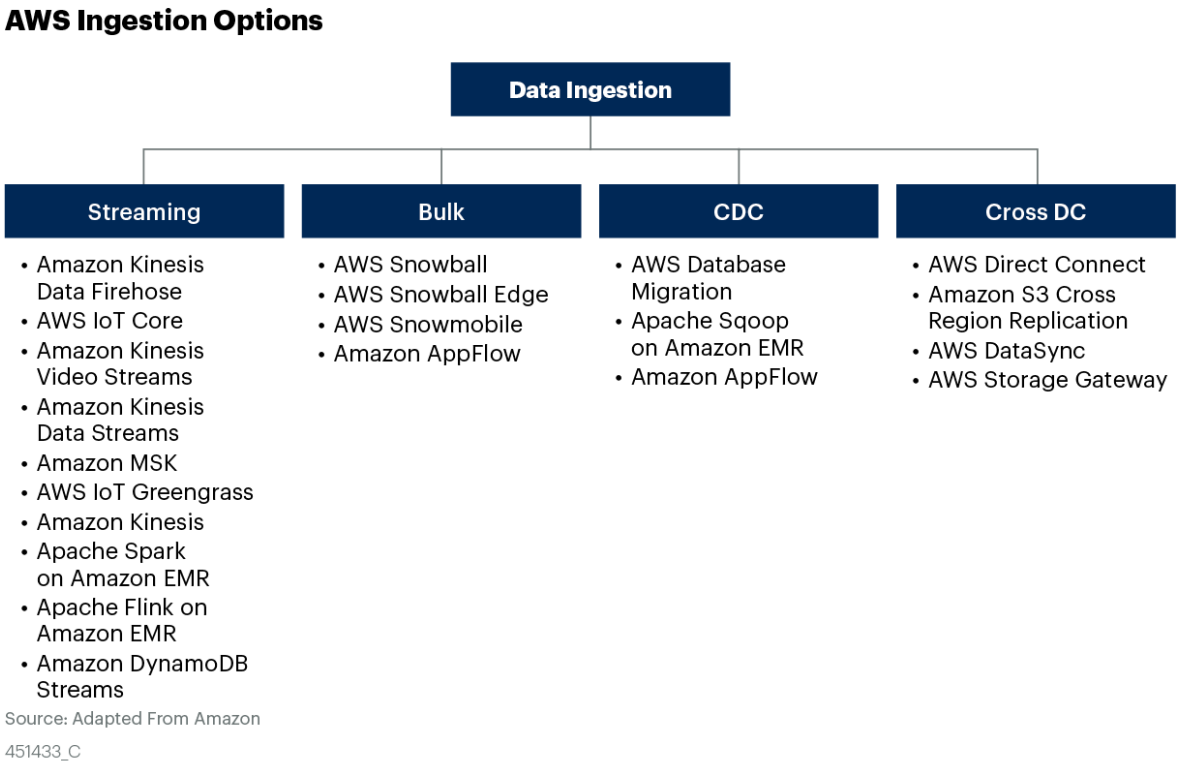
As a first step to cloud migration, data ingestion is becoming more and more critical as organizations move to hybrid, multicloud and intercloud-based deployment models, where data generation happens in one place while the data storage, analytics and insights are generated across the WAN. Moreover, as organizations adopt SaaS applications, they are increasingly looking to

leverage valuable business data that is locked in these applications with the analytics and machine learning services offered by AWS to draw unique business insights and improve operational efficiency.

Data Ingestion Options in AWS

AWS provides a wide variety of services to ingest data from across different data sources. Figure 3 shows the different category of tools in the AWS ecosystem for data ingestion.

Figure 3. AWS Ingestion Options



AWS provides a rich set of native tools for data ingestion and data integration. Some of the tools in each of the categories are discussed below.

Batch and CDC

Batch-based ingestion is the default standard for data ingestion mostly used for data transfer of static data at a given cadence from source to destination. Change data capture (CDC) is an ingestion pattern that enables capturing the deltas (changes) at the source and transmitting the changes to the target. Some of the batch and CDC-based ingestion services in AWS are listed in Table 1.

Table 1. Batch and CDC Ingestion Tools

Service	Description	Use Cases
AWS Transfer for SFTP	■ Managed service with pay-per-use model with autoscaling ■ Provides redundancy across multiple AZs ■ HIPAA and PCI compliant ■ SOC 1, 2 and 3 compliant	■ SFTP transfers into S3
AWS Database Migration Service	■ Migrates and replicates databases and data warehouses to AWS ■ Offers a wide variety of sources and targets.	■ Seed a new DBaaS platform ■ Synchronize destination system with source data source ■ One time or continuous replication from on-premises to cloud ■ Archival purposes
AWS Glue	■ Serverless service provides data transformation, cataloging, and ETL ■ Provides out-of-the-box transformations	■ Building ETL pipelines ■ Automating technical metadata discovery to build internal data catalog
Amazon CloudFront	■ Ingest data using PUT/POST services ■ Provides Amazon S3 multipart upload capabilities ■ Supports SSL, TLS	■ Global content delivery ■ Ingest into Amazon S3, Amazon EC2, Amazon ELB
Apache Sqoop in Amazon EMR	■ Transferring bulk data between Apache Hadoop and structured data stores	■ Batch and CDC-based data ingestion from relational data stores to HDFS and Amazon S3
AWS Batch	■ Fully managed batch scheduler that manages compute and job life cycle	■ Read from third-party or local data stores, transform and write to Amazon S3 ■ Process large volumes of asynchronous jobs ■ Read from storage solution, perform computations, and write back to storage solution
Amazon AppFlow	■ Integration service that enables customers to transfer data bidirectionally between various SaaS applications and AWS services; no coding needed; includes out-	■ Supports CDC data transfers between SaaS and AWS; can transfer data in near rea time (event-based triggers) on a schedule or on click of a button

Service	Description	Use Cases
	of-the-box filtering, validation and transformation capabilities	

Source: Gartner (June 2020)

Integrations With SaaS APIs

SaaS applications such as Salesforce, Workday and Marketo allow users and other applications to access data, business logic or functionality from their back end through application programming interfaces (APIs).

APIs are core to how modern integrations are built. SaaS applications have made hundreds of thousands of APIs publicly available to support a wide range of data transfer patterns, ranging from bulk to near-real-time polling for CDC events. Amazon AppFlow automatically connects to the SaaS application APIs to transfer data or monitor for events. Use Amazon AppFlow to create data flows between SaaS applications and AWS services. Flows can be set to transfer data on a schedule, or triggered by business events such as launching a campaign, converting a lead, or opening a support case. You can also configure data transformation tasks such as masking, truncating, formulas, merging, field mapping, validation and filtering as part of the flow. For use cases that require advanced or custom data transformations, leverage AWS Glue to ETL the data before sending it downstream to other AWS services.

For ingesting events from SaaS applications that have a mechanism to emit events into the AWS ecosystem, leverage Amazon EventBridge. Amazon EventBridge enables developers to build event-driven applications that interact with SaaS applications and AWS services. SaaS applications that have integrated with EventBridge emit events to the customer's event bus, which can then be routed to targets such as Amazon EC2 instances or AWS Lambda functions for processing.

Best Practices

Some recommended best practices for batch data ingestion services are outlined below.

AWS Database Migration Service

- Disable unnecessary triggers, validations and relevant index generation.
- Disable multi-AZ and reduce your automatic backup retention to the minimum on the target until cut-over is completed.
- When migrating from relational databases, use limited LOB (large objects) mode. If tables have few large blobs and many smaller LOBs, migrate using two separate tasks in full LOB and limited LOB mode.

AWS Glue

- Use blueprints that create the necessary workflows, crawlers and jobs for common use cases. Workflows enable orchestration of data workloads by building dependencies between Glue entities, as triggers, crawlers and jobs, and allow tracking status of the nodes in the workflows on the console, making it easier to monitor progress and troubleshoot issues.

Stream

Streaming ingestion involves ingesting data, mostly in real-time or near-real-time feeds from any source (IoT, social media streams, system logging, etc.) and delivering this data to a sink in raw or processed form. Some of the stream-based ingestion services in AWS are given in Table 2.

Table 2. Stream Ingestion Tools

Service	Description	Use Cases
Amazon Kinesis Data Streams	▪ Managed cloud platform for scalable and durable real-time streaming data capture and storage. ▪ Enables dynamic adjustment of throughput of stream based on the volume of data. A single API call can scale from 1 MB/sec to 10 GB/sec. ▪ Enables building real-time stream processing applications using AWS Lambda, Amazon Kinesis Data Analytics, Amazon EMR.	▪ Continuously capture GB/sec of data from sources clickstreams, CDC, transactions, social media feeds, IT logs and location-tracking events. ▪ Automatic capture is supported through over 17 AWS data sources like AWS IoT Core and Amazon CloudWatch logs. ▪ Data collected available in milliseconds to enable real-time analytics as real-time dashboards, anomaly detection, dynamic pricing, performing complex streaming ETL before loading data into other data stores.
Amazon Kinesis Data Firehose	▪ Requires zero admin and autoscales to match data ingest throughput for fully managed streaming. ▪ Can transform, compress, and batch the data before loading. ▪ Provides prebuilt Lambda blueprints — can be used without any change or customization.	▪ Continuously capture GB/sec of data from custom data sources. ▪ Loading streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch (ES) and other data stores.
Amazon Managed Streaming for Apache Kafka (MSK)	▪ Fully compatible with open source Apache Kafka 2.4.1. ▪ Fully managed Apache Kafka, with Apache ZooKeeper managed at no additional cost.	▪ Integrates with fully managed, serverless Apache Flink service through Amazon Kinesis Data Analytics. ▪ Supports multiple security features including VPC network isolation, AWS IAM for control-plane API authorization, encryption at rest, TLS encryption in-transit, TLS based certificate authentication, and supports Apache Kafka Access Control Lists (ACLs) for data-plane authorization.
AWS IoT Core	▪ Managed cloud platform. ▪ Allows connected devices securely interact with cloud applications and other devices.	▪ Capture data from connected devices such as consumer appliances, embedded sensors, and TV set-top boxes.
Amazon Kinesis Video Streams	▪ Securely stream audio and video from connected devices to ingest into AWS.	▪ Streaming ingestion. ▪ Frame analysis of live video streams.
AWS IoT Greengrass	▪ Extends AWS to edge to act locally on data generated using the cloud for management, analytics and storage.	▪ Manage and capture data from connected devices — automation systems, routers, cameras, hubs, gateways.

Service	Description	Use Cases
		■ Devices using FreeRTOS or the AWS IoT Device SDK can be configured to interact with AWS IoT Greengrass. ■ Connect to third-party applications and AWS services out-of-the-box with AWS IoT Greengrass Connectors.

Source: Gartner (June 2020)

Bulk Data Transfer

Ingestion can happen online or offline. When the volume of data is enormous and it is difficult to provision network bandwidth for transfer of such bulk data over the network, organizations resort to using offline techniques to seed the destination systems. Some of the bulk data ingestion services in AWS are given in Table 3.

Table 3. Bulk Ingestion Tools

Service	Description	Use Cases
AWS Snowball	▪ Should be used for transfer 50-80TB capacity range. ▪ Provides end-to-end data encryption. ▪ Provides CLI and S3 SDK interface for integration and automation.	▪ Ingesting static data into and out of S3. ▪ Cloud and data center migration. ▪ Content distribution. ▪ Can be used in conjunction with AWS DataSync for a combination of online and offline data, to accelerate migrations or transfers.
AWS Snowmobile	▪ Should be used for transfers ranging from double-digit petabyte- to exabyte-scale end-to-end data transfer. ▪ Provides data encryption and GPS tracking of the data as it moves from one data center to another.	▪ Offline data transfer for up to exabyte-scale data volumes.
AWS Snowball Edge	▪ Should be used for 15PB or less scale end-to-end data transfer. ▪ Provides interface to S3 and NFS interface and works over 10/25/40 G networking. ▪ Provides end-to-end data encryption and EC2/AMI and AWS IoT Greengrass support for edge compute.	▪ Ingesting static data into and out of S3. ▪ Cloud and data center migration. ▪ Storage and compute in fully or intermittently disconnected, rugged non-data-center environments, enabling local processing.
AWS Snowcone	▪ Should be used for 24TB or less end-to-end data transfer. ▪ Provides interface to NFS interface and works over 1/10 G networking. ▪ Provides end-to-end data encryption and EC2/AMI and AWS IoT Greengrass support for edge compute.	▪ Ingesting static data into and out of Amazon S3 by shipping the device to AWS. ▪ Storage and compute in fully or intermittently disconnected rugged non-data-center environments, enabling local processing ▪ AWS DataSync running on AWS Snowcone enables online data transfers to accelerate migrations or transfers.
AWS DataSync	▪ Automates and accelerates data movement between on-premises and AWS. ▪ Purpose-build transfer protocol to simplify network configuration and optimize data transfer ▪ Provides configurable throughput limits.	▪ Online transfer of active or archive data, including migrations or ongoing workflows. ▪ Transferring data for time sensitive in-cloud analysis. ▪ Ideal for migrating data into Amazon S3, Amazon EFS or FSx for Windows.

Service	Description	Use Cases
	■ Highly optimized for Amazon Elastic File System (EFS) and S3-based data transfers. ■ Automatic recovery from IO errors. ■ Multiple levels of data validation, both at-rest and in-transit ■ PCI, HIPAA, SOC 1/2/3, FedRAMP ■ Also supports batch and CDC ingestion (Table 1) and cross data center ingestion (Table 4).	

Source: Gartner (June 2020)

Cross Data Center Data Ingestion

With the rapid adoption of cloud platforms for data and analytics, organizations need to move the data generated on-premises or managed data center into the cloud on a regular basis. Some of the ingestion services that allow data to be moved from an organization's data center to AWS using a dedicated network connection are given in Table 4.

Table 4. Cross Data Center Ingestion Tools

Service	Description	Use Cases
AWS Direct Connect	■ Reduces costs for bandwidth-heavy outbound workloads by connecting to an AWS Direct Connect router, bypassing ISPs.	■ Private network connections to AWS from DC, office. 1 Gbps/10 Gbps port.
Amazon S3 Transfer Acceleration	■ Fully managed file transfer acceleration using all AWS edge locations. ■ Data not stored on edge cache. ■ AWS makes the transfer optimized by internal data compression.	■ Edge locations for S3 enabled applications. Ideal for long geographic distances.
AWS Storage Gateway	■ Provides applications on-premises access to native storage in S3/Amazon Elastic Block Store (EBS)/Amazon S3 Glacier. ■ Provides low-latency access to frequently used data. ■ File Gateway is used with data lakes, processing, and data distribution workflows, as well as for simple POSIX-compliant or SMB file shares. ■ Used to replace on-premises storage, including for moving on-premises backups to the cloud.	■ Has three gateway types, all of which provide local caching: ■ File Gateway provides on-premises applications with file-based access to S3 storage using SMB or NFS protocols. ■ Volume Gateway provides block-based volumes with EBS Snapshots that can be used to move data for EC2 based applications. ■ Tape Gateway presents a VTL interface to replace traditional tape-based archives, using S3 Standard, S3 Glacier and S3 Glacier Deep Archive storage pools.

Source: Gartner (June 2020)

Best Practices

It can be challenging to make the right choice from the plethora of products and services offered for data ingestion and data transfer in the AWS ecosystem. AWS offers templates for organizations to make the right choices when choosing tools. Implementation, coordination, orchestration and configuration-based integration between the services is left to the end user and the experts.

Tools like AWS Glue offer ingestion as code with multiple capabilities like processing of batch data, development of ETL pipelines, and a native data catalog. You can use Amazon CloudWatch automated monitoring tools to watch AWS Glue and report when something is wrong using Amazon CloudWatch Events, Amazon CloudWatch Logs and AWS CloudTrail. No single unified tool performs monitoring, alerting, notification across all the ingestion services.

Some of the best practices when ingesting data for building a data platform on cloud or on-premises are listed below:

- Before choosing the tools for ingestion, understand the format, type of data, volume, data distributions and delivery guarantees to make the optimal use of limited bandwidth and honoring the SLAs.
- The ingestion layer should not modify and transform incoming data in any way, to make sure that the raw, unprocessed data is always available in the lake for data lineage tracking and reprocessing.
- Focus on ensuring that the data ingestion layer has a pluggable architecture. New types of data sources are added all the time. It's unrealistic to expect that connectors for every data source will be available in the ingestion tool or service that you choose. Make sure that data ingestion layer allows you to add new connector types without significant effort.
- The ingestion layer should be able to handle large volumes of data and be able to scale beyond existing capacities. Choose solutions that will not require completely revamping the ingestion layer as data volume requirements grow. For scalability, enterprises should consider ingestion layers that are serverless and scale automatically for high data volumes.
- Ensure high availability. The ingestion layer should be able to handle failure of individual components, like disk, network, or full virtual machine failures and still be able to deliver data to the data platform with zero or minimal loss.
- Ensure recoverability. This means the ability to recover from a failed execution and restart from the last known point or recover without any data integrity and data loss.
- Ensure observability. The ingestion layer should expose critical metrics, like data throughput and latency, to external monitoring tools. Most of these metrics should be stored in the central metadata repository. Some of the more technical metrics, like memory or CPU or disk utilization, might be exposed to the monitoring tools directly.
- Ensure best practices to manage the schema and structural data drift especially for unstructured or streaming datasets.

Select tools that:

- Maximize performance.
- Use the right tool and networking configuration for efficient data transfer.
- Encrypt data end to end, both in flight and at rest. Use strong ciphers integrated with AWS security services.
- Minimize dependency on the data engineers when onboarding new datasets, preventing bottlenecks. Ensure ability to handle new event types and data schema changes dynamically as soon as they are ingested.

- Automate boilerplate code and configuration generation, ensuring that infrastructure, permissions and deployment setup are abstracted away from users and monitor the data workflow.
- Possess error, exception handling, and retry capabilities to gracefully recover from failed or partially failed ingest jobs and handle piecemeal data transfers as well as large files.
- Handle backpressure and throttling challenges.
- Implement monitoring end to end and specifically at chokepoints and metering at various stages of the ingestion pipeline.

How to Select Data Ingestion or Data Transfer Service

Choosing the right ingestion mechanism depends on use case and nonfunctional requirements like data latency, data volume and data type. Listed below are some guidelines on choosing the ingestion service within the AWS ecosystem:

- Use AWS Direct Connect (DC) when looking for an optimized internet connection between a regional data center and AWS. AWS Direct Connect provides more consistent, predictable network performance guarantees (1 Gbps or 10 Gbps per link).
- Use AWS DataSync for secure, managed, online transfers of TB to PB of active or passive data, if your available network resources can support the volume of data you need to move.
- Use AWS Snowcone or AWS Snowball Edge when looking to migrate TB/PB of data into the cloud and when network bandwidth is too constrained for the project timeline.
- Use Amazon Kinesis Data Firehose when looking to aggregate data streams from multiple devices and an interface with Amazon S3.
- Use Amazon Kinesis Data Streams when looking to capture data from disparate event sources that need custom transformation and preprocessing before sending to Amazon S3 or other data stores within AWS.
- Use AWS Storage Gateway for files when looking to access or store data in Amazon S3 buckets on-premises with a traditional file interface and get local caching. After you migrate data with tools better suited for migration, such as AWS DataSync or AWS Snowball, you can use AWS Storage Gateway for Files to access the data from Amazon S3. You can also use AWS Storage Gateway to move backups or archive processes to AWS with minimal disruption.
- Use AWS Transfer for SFTP when wanting to replace traditional self-managed data transfer servers and infrastructure with a fully managed, cloud integrated, service for Amazon S3.

To choose between Amazon S3 Transfer Acceleration, Amazon CloudFront's PUT/POST and other services, use the following guidelines:

- Amazon S3 Transfer Acceleration uses TCP/IP optimizations to accelerate large object PUTS and GETS to/from S3 over long distances. It routes requests to the nearest AWS Edge location

for faster response to the local application, and then optimizes transfers of data over the AWS backbone.

- If you have objects that are smaller than 1GB or if the dataset is less than 1GB in size, you should consider using Amazon CloudFront's PUT/POST commands for optimal performance.
- Amazon S3 Transfer Acceleration is the best option when your local application can work directly with the S3 API and there is plenty of bandwidth. However, if your data resides in a local file system or NAS, you should look at AWS DataSync for ongoing or one-time transfers over the network. In case of a single large-scale exchange operation when network bandwidth is not available, AWS Snowball will typically be more appropriate.
- When there is a need to move data over long distances using the internet, for instance across countries or continents to Amazon S3, depending on the local interface needed, consider DataSync or S3 Transfer Acceleration. For transfers with the S3 API, S3 Transfer Acceleration is the right choice because it optimizes data transfer with the use of routable network paths and Amazon's backbone network up to 300% compared to normal data transfer speed.

AWS Database Migration Service and AWS Schema Conversion Tool should be used for the three use cases below:

- Modernization, when doing heterogeneous database migrations during modernization of the data tier or data warehouse tier. When moving from commercial databases to open source or commercial to Amazon Aurora, or moving from commercial data warehouses to Amazon Redshift.
- Migration, when migrating business-critical applications to AWS, upgrading to a minor version, or when consolidating shards into Amazon Aurora.
- Replication, when creating cross-region replicas or running analytics on the cloud or keeping different environments (development/QA/production) in sync, even if some of those environments are outside AWS.

To ingest data from sources that are currently not supported by AWS Glue or DMS, organizations can implement and run ingestion code using a serverless AWS Lambda environment. Organizations also have the option of using native database migration tools for the databases that Amazon Relational Database Service (RDS) supports, such as mysqldump, pg_dump/pg_restore, Oracle Data Pump, Oracle Recovery Manager (Oracle RMAN), Microsoft SQL Server backup and restore. They can also use third-party database migration tools from AWS partners. When using multiple services for data ingestion and processing like DMS or AWS Lambda functions, it is best to leverage AWS Step Functions to build workflows that span across multiple services.

Data Storage

Unless one is doing stream processing and streaming analytics, the data needs to be stored post-data ingestion. An important characteristic of a modern data storage system is that it must be scalable, inexpensive, and able to accommodate the variety and volume of data created. Usually, scalable storage in a data center can be a large disk array or network-attached storage. These

provide access to large volumes of storage, but are usually expensive and have a predefined capacity. Flexible and inexpensive storage with on-demand elastic capacity is a lucrative option provided by cloud vendors. The ability to store data in any format is an important foundation of modern data architectures.

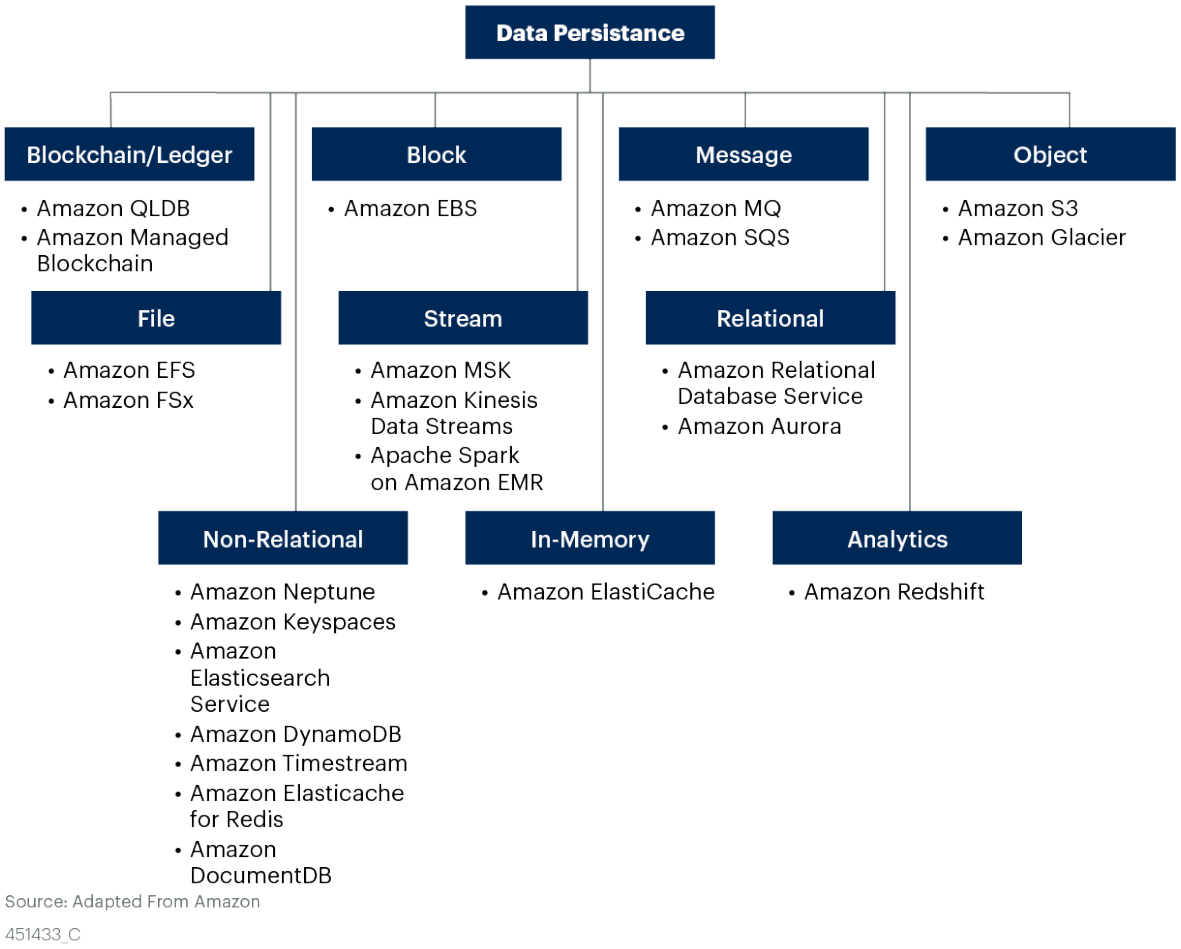
When selecting data stores on a cloud platform:

- Ensure it is fully managed by the cloud provider. This means you don't need to worry about maintenance, software or hardware upgrades, etc.
- Ensure it is elastic. This means cloud vendors will only allocate the amount of storage needed, growing or shrinking the volume as requirements dictate. You no longer need to overprovision storage system capacity in anticipation of future demand.
- Ensure that you only pay for the storage and compute capacity used.
- Ensure you understand the durability of the system being used for data storage and adopt the right best practices to meet your internal standards, including replication, backup, and HA/DR.

Data Storage Options in AWS

Figure 4 below shows the range of storage options in AWS from relational to nonrelational, object, file and message stores.

Figure 4. AWS Storage Options



Source: Adapted From Amazon

451433_C

Structured

Data stored in relational databases is usually structured. Table 5 lists services on AWS for storing structured data.

Table 5. Structured Data Stores

Service	Description	Use Cases
Amazon Relational Database Service (MySQL, Postgres, MariaDB, Oracle and Microsoft SQL Server)	■ Create, manage and scale relational databases in the cloud ■ Management like backups, patches, automatic failure detection, and recovery are taken care by the managed service ■ Provides a standby instance in a separate AZ and up to five read replicas in one click ■ Allows replication to other AWS regions for DR and low-latency global read ■ Uses Amazon EBS storage. Storage can be provisioned with an option of autoscaling configuration	■ Use cases related to relational databases on the cloud
Amazon Aurora	■ MySQL and PostgreSQL-compatible relational database built for the cloud and managed by RDS ■ Meant for GB- to TB-size databases; has 64TB max limit on a database ■ Purpose-built distributed, fault-tolerant, self-healing storage system that autoscales up to 64TB. ■ Provides up to 15 read replicas with shared storage ■ Automatic failover capabilities using multi-AZ configuration (writer and reader deployed in different AZs) ■ Provides availability across three AZs ■ Provides high durability by replicating six copies of the data across three AZs, plus point-in-time recovery, fast database cloning, and continuous backup to Amazon S3 ■ Provides additional features such as fast database cloning, serverless and backtrack, that are not available on RDS engines	■ Ideal for relational database workloads with ■ High concurrent request rates ■ Low latency (ms/sec)

Source: Gartner (June 2020)

Best Practices

Some recommended best practices for structured data stores are:

- Import schema and data in one operation.
- Use native database tools for export and import for schema and data wherever possible.
- Leverage AWS integration with native services such as S3 and AWS Lambda for moving data between AWS services.
- Keep the business logic out of the database.

- File splitting and compression is recommended for databases greater than 1TB.
- Import user accounts/passwords, functions and stored procedures separately.
- Consolidate workloads for better resource management.
- Use role-based access control.
- Manage your database credentials via AWS secrets manager.
- Consider IAM authentication for centrally managing the database users.

Performance best practices:

- Follow engine-specific best practices for database performance (MySQL/MariaDB/Postgres, etc.)
- Configure and test the database with your own application and real-life data instead of synthetic benchmarks, since database workload characteristics vary widely.
- Use Amazon RDS Performance Insights to identify bottlenecks and top SQL.
- Database parameter groups are configured based on the RDS engine and compute class, and allocated storage of the instance is optimal for most cases. Customize it to suit your workload if needed.

Leverage high concurrency:

- Amazon Aurora throughput increases with number of connections. Therefore, architect applications to leverage high concurrency in Aurora.
- Use Amazon RDS Proxy if you need connection pooling, increased application availability, and additional database access control.

Read scaling:

- Amazon Aurora offers read replicas with lower replication lag on the order of tens of milliseconds. Leverage it to scale reads and keep query cache on.
- Use Amazon Aurora Global Database to replicate the data to other regions for DR and low-latency global reads.
- Use Amazon Aurora Serverless if your workload includes infrequent or unpredictable traffic patterns.
- Use Amazon Aurora Auto-Scaling for vertical and horizontal scaling, which would have to be done manually with Amazon RDS.
- For a read-intensive workloads Amazon Aurora is a good fit.
- If you need more flexibility with different instance classes and want to be in step with open-source engines and its features, use RDS.

- If your workload requires high concurrence, high durability and on-demand scaling with enterprise features, leverage Aurora.
- Use Aurora if you're interested in AWS-developed features such as Backtrack, Parallel Query or Aurora Machine Learning.

File-Based Storage

Organizations often have a lot of data stored in flat files across a variety of data formats. Table 6 lists some of the services in AWS for storing file-based data.

Table 6. File-Based Data Stores

Service	Description	Use Cases
Amazon Elastic File System (EFS)	■ Scalable, elastic, cloud-native file system ■ Accessible via the NFS protocol ■ Elastically grows and shrinks as you use the file system ■ Offers traditional file storage organized into directories and subdirectories ■ Can mount EFS onto several EC2 instances at the same time ■ Fully managed'(requires no provisioning) ■ Durable (designed for "eleven nines" durability) ■ Available across three AZs with 99.9% SLA ■ Offers a choice of storage classes (standard and infrequently accessed) to optimize cost/performance ■ Provides up to multiple GB/sec throughput per file system, with up to tens of thousands of IOPS at single-digit-ms latencies using general-purpose mode, and up to hundreds of thousands of IOPS at higher latencies with max I/O mode	■ Useful for SaaS applications ■ Content management systems ■ Lift-and-shift enterprise applications ■ Analytics and machine learning ■ Web serving and content management ■ App testing and development, media and entertainment, database backups ■ Container storage ■ Web serving and content management ■ Data analytics
Amazon FSx for Windows File Server	■ Scalable, shared, fully managed file storage built on Windows Server ■ Accessible via the SMB protocol — supports access from Windows, Linux and Mac OS, and access from a broad variety of compute instances (EC2, WorkSpaces, AppStream 2.0, and VMware Cloud on AWS) ■ Integrates with Microsoft Active Directory for authentication and authorization ■ Provides sub-ms latencies, up to multiple GB/sec throughput, and hundreds of thousands of IOPS per file system ■ Offers a choice of storage types (SSD and HDD) to optimize storage cost vs. performance, and deployment types (single-AZ and multi-AZ) to optimize cost vs. availability, and supports data deduplication and compression to optimize storage costs	■ Home directories, user and departmental shares ■ Business applications ■ Database applications ■ Media and entertainment workflows ■ Content management systems ■ Data analytics
Amazon FSx for Lustre	■ Offers the world's most popular high-performance file system, Lustre, as a fully managed service ■ Offers a POSIX-compliant file system, so you can use your current Linux-based applications without having to make any changes	■ Machine learning ■ High-performance computing (HPC) ■ Media processing and transcoding

Service	Description	Use Cases
	■ Integrates with Amazon S3, making it easy to process your datasets ■ Provides sub-ms latencies, up to hundreds of GB/sec throughput, and millions of IOPS per file system ■ Offers a choice of replication levels (scratch and persistent) to optimize cost for short-term and longer-term data processing	■ Autonomous vehicles ■ Big data and financial analytics ■ Electronic design automation (EDA)

Source: Gartner (June 2020)

Object

Object storage is best for data-intensive applications, for applications that require streaming throughput (GB/sec) rather than high transaction rates (IOPs). Object storage also offers compelling economies for large datasets, so more data can be kept online and available on demand. Table 7 lists some of the services on AWS for object stores.

Table 7. Object-Based Data Stores

Service	Description	Use Cases
Amazon S3	■ Best fit for MB- through PB-size data storage ■ Each object size limited to 5TB ■ Very high durability: eleven nines ■ Support AWS API (SDK, CLI) and third-party tools for integration ■ Available across three AZs	■ Storing, archiving critical data or backup because automatically replicated cross-region, providing maximum availability and durability ■ Image/video and data and files ■ Static datasets ■ Data lakes and big data analytics
Amazon S3 Glacier	■ Object store ■ Best fit for TB- to PB-size datasets ■ Available across three AZs ■ Very high durability ■ Supports S3 and AWS API and third-party tools for integration	■ Meant for cold data ■ Data archival ■ Magnetic tape replacement

Source: Gartner (June 2020)

Best Practices

Some recommended best practices for object-based data stores are:

- Set a life cycle for data to move through different storage classes.

- Use Amazon S3 Standard to store hot data that is being immediately used across different enterprise applications. Data is replicated across a minimum of three AZs.
- Use Amazon S3 Intelligent-Tiering to store data with unknown or changing access patterns. This storage class automatically moves data between two different storage classes for cost optimization. Data is replicated across a minimum of three AZs.
- Use Amazon S3 One Zone-Infrequent Access to hold backup copies or other easily re-creatable data with rapid access requirements. Data is stored in a single AZ.
- Use Amazon S3 Glacier to archive cold data that may need to be retrieved in as little as five minutes at a very low cost as compared to on-premises storage. Data is replicated across a minimum of three AZs.
- Use Amazon S3 Glacier Deep Archive as a long-term data archive at a cost comparable to on-premises tape storage. Data is replicated across a minimum of three AZs.

Amazon S3

- Evaluate the pricing and requirements for S3 and its other storage classes. S3 has four other storage classes on top of S3 Glacier and S3 Glacier Deep Archive: S3 Standard, S3 Standard-Infrequent Access, Amazon S3 One-Zone Infrequent Access and Amazon S3 Intelligent-Tiering. These classes differ in terms of prices and minimum storage duration requirements (some have no minimum requirement).
- Use Amazon S3 for write once, read many times-type workloads.
- Do not use S3 to host OS or databases.
- For high-volume operations, consider naming schemes with more variability at the beginning of the key names for maximum throughput. Latency on S3 operations depends on key names, since prefix similarities become a bottleneck at more than 5,500 read requests per second.
- Use AWS SDK and AWS S3 API mechanisms to optimize S3 downloads and uploads via increased parallelism and concurrency when dealing with large transfers.
- To delete or archive based on object tags, tag objects so that it is easier to apply life cycle policies as well as to search.
- Consider compression schemes for large data that isn't already compressed to mitigate S3 bandwidth and cost constraints.
- Do not use immediate GET or HEAD call, prior to the 200 response, since, this might result in not showing the object. S3 provides read-after-write consistency for PUTS of new objects. Amazon S3 is an eventually consistent storage that returns HTTP 200 response to confirm a successful object creation.
- Ensure that the bucket name is DNS-compliant and does not contain periods.

- Using Amazon S3 Select for retrieval of data can result in lower costs as it provides query-in-place on data stored in S3, without having to download, decompress and process the entire dataset.
- Amazon S3 provides different tiers of storage, all with eleven nines of durability. S3 Standard Storage provides four nines of availability and can sustain the loss of two data centers. To store infrequently accessed data, use S3 Standard-IA or S3 Intelligent-Tiering storage, which have lower storage cost but higher retrieval time. We recommend S3 One Zone-IA for lower-cost, millisecond-access storage.
- Object stores offer tremendous scalability but can be slow depending on a variety of factors. S3 latencies vary from tens to hundreds of milliseconds as compared to the 0.1- to 9-millisecond range with HDFS. Ensure the data platform takes the latency into considerations when building low-latency applications.
- Directory listing is very slow and hence should be used only for noncritical purposes.
- As of this writing, S3 has a default rate limit of 5,500 reads per second and 3,500 writes per second — keep that in mind when launching thousands of MR jobs. Few customers ever reach that limit, and if they do, they can open a ticket to have the limit increased.
- When writing from Spark jobs to S3 directly, it has been observed that the Spark cluster keeps running doing nothing even after the application has completed. This is because each Spark task writes the output to a temporary path and then moves files into the destination folder at the end of the write process. S3 is not a file system and there is no native “move” operation, so what happens under the hood is that it creates a copy of the file in the destination folder and then (after the copy has finished) it deletes the source file. This step could take a long time, and it looks like it isn’t executed in parallel even if you have to move just a few files. Often, it’s quicker to write the application result in HDFS and to add a second step to copy the files from Apache Hadoop Distributed File System (HDFS) to S3 using S3DistCp. For Parquet files, use the EMR File System (EMRFS) S3-optimized committer. For more details, see “Improve Apache Spark Write Performance on Apache Parquet Formats With the EMRFS S3-Optimized Committer.”
- Make sure, when using S3 tagging, that you do not exceed the maximum limit of 10 tags per object and 128 Unicode characters.
- Ensure, when using S3 Select, that the data in S3 is either in CSV, JSON or Parquet format, and only a subset of SQL queries is supported.
- S3 is discussed in more detail in “Key Services Differences Between AWS, Azure and GCP: Compute and Storage.”

Amazon S3 Glacier

- Use Amazon S3 Glacier for archives and backup copies that may be retrieved once or less per year. If used for backups, understand the SLAs.
- Use life cycle policies to set up to move automatically between classes, and even to be deleted permanently at a certain point. This should be automated to move data from Amazon S3

Standard to Amazon S3 Glacier Deep Archive in order to transition from being a hot backup copy to a yearly archive.

- Remember that uploading and downloading many small files onto Glacier might be very expensive and that there is a 32KB storage overhead per file. Aggregating many smaller files into a larger archive when moving to Glacier is recommended. Ensure these overheads are built into the expected SLAs.
- Be aware of per-object costs of archiving S3 data to Glacier. It costs $0.05 per 1,000 requests. If you have large numbers of S3 objects of relatively small size, it will take time to reach a break-even point.
- Use Amazon S3 Glacier only when minimum data storage duration requirements go beyond the limitations around it, as discussed above.

Unstructured and Multistructured

Data stored in nonrelational databases, usually in JSON or XML, is classified as semistructured and exhibits nested and hierarchical organization. Table 8 lists some of the services on AWS for storing unstructured and multistructured data.

Table 8. Unstructured and Multistructured Data Stores

Service	Description	Use Cases
Amazon DocumentDB (with MongoDB compatibility)	■ Fully managed service supports MongoDB 3.6 workloads ■ Store, index, and query JSON-formatted data ■ Ad hoc querying of nested attributes and powerful aggregation capabilities ■ Scales to millions of reads per second with up to 15 read replicas ■ Storage volume replicates data six ways across three AZs ■ Automatic, continuous backups ■ Automatic, continuous incremental backups and point-in-time restore.	■ Content management ■ User profiles ■ Product catalogs ■ Personalization ■ IoT/device management
Amazon Elasticsearch Service	■ Accepts JSON documents, and indexes and stores them in Apache Lucene format ■ Provides very fast search service using reverse indexes with integrated support for analytical functions; item size limited to 2GB	■ Log- and machine-generated analytics ■ Search analytics ■ Full-text search
Amazon DynamoDB and Amazon DynamoDB Accelerator (DAX)	■ Massively scalable, serverless, key value and document data store ■ Single-digit millisecond performance ■ On-demand capacity mode requires no capacity planning ■ Fully managed, multiregion, multimaster ■ Built-in security, with full encryption, backup and restore ■ Replicated across multiple AZs for high availability and durability ■ Handles more than 10 trillion requests per day ■ Supports peaks of more than 20 million requests per second ■ In-memory caching for internet-scale applications via DAX, a fully managed, in-memory cache for DynamoDB that reduces DynamoDB response times	■ Serverless and event-driven applications ■ Mobile back ends ■ Microservices ■ Gaming back ends ■ Shopping carts ■ Session state storage ■ User profiles ■ User transactions ■ Media metadata
Amazon Keyspaces (for	■ Serverless, scalable Apache Cassandra service ■ Only pay for used resources	■ Fast ingest data

Service	Description	Use Cases
Apache Cassandra)	■ Offloads Cassandra management to the service provider ■ Data is encrypted automatically using AWS KMS ■ Replicated across multiple AZs	■ Semi-structured data that needs schema flexibility ■ Build applications that require low latency ■ Build applications using open-source technologies ■ Move Apache Cassandra workloads to the cloud
Amazon Neptune	■ Fully managed graph database service ■ Supports high availability within a region using three AZs ■ Support for ACID transactions with immediate consistency ■ Supports W3C RDF and Apache TinkerPop property graph frameworks ■ Stores up to 64TB of data, which is roughly 100 billion to 200 billion nodes, edges and properties ■ Data is encrypted at rest using AWS KMS and encrypted in flight using TLS 1.2 ■ Eligibility for HIPAA, ISO, PCI/DSS and SOC compliance regimes	■ Graph based use cases using property graphs and RDFs ■ Fraud detection ■ Customer 360 ■ Identity linking ■ Knowledge graph, data lineage ■ Recommendation engine ■ Life sciences ■ Social network ■ Network and infrastructure monitoring in IT
Amazon Timestream	■ Fully managed serverless time series database service for IoT and operational applications ■ Purpose-built for storing time series data, organizes data to support optimal query processing, autoscales based on data volume, and offers data storage tiering to manage data at a lower cost	■ Time series-based use cases ■ High throughput ingestion

Source: Gartner (June 2020)

Best Practices

Some recommended best practices for unstructured and multistructured data stores are:

Amazon DocumentDB (with MongoDB compatibility)

- Deploy clusters with three or more instances in three AZs for a 99.99% availability target
- Use the cluster endpoint in replica set emulation mode when connecting from applications

- Use read replicas in conjunction with driver read preference settings to scale read capacity
- Ensure indexes fit in instance buffer cache (about two-thirds of instance RAM)
- Create alarms for key instance and cluster level metrics in CloudWatch Metrics
- Use role-based access control (RBAC) roles to enforce least-privilege security principles
- Use batch operations for high-volume insert and update workloads
- Configure cluster backup windows to meet your recovery point objective (RPO)

Amazon DynamoDB

- Follow NoSQL data modeling best practices and ensure your DynamoDB table data model matches your expected query patterns. You can use tools like the NoSQL Workbench for DynamoDB to visualize your data model as you design your tables and queries.
- Developers should be conscious of creating unbalanced workloads with Amazon DynamoDB, and follow best practices for designing partition keys. Adaptive capacity is a recent, automatic feature that enables DynamoDB to run unbalanced workloads and helps you reduce cost by managing available table capacity more efficiently.
- Use on-demand capacity mode for a fully automated, hands-off capacity management experience. On-demand is especially good option when traffic patterns are unpredictable, or if you prefer pay-per-request pricing so that you pay only for what you use.
- Leverage the ability to change capacity provisioning mode and adjust capacity provisioning to match your traffic patterns.
- Use Time to Live (TTL) to have DynamoDB automatically expire cold data, at no additional cost.
- As a serverless service, Amazon DynamoDB is designed as a black box with very few user-accessible controls. This approach makes it easy to use, but in production more often than not one needs deeper insight into why data is misbehaving. However, AWS has recently added the visibility and metrics for DynamoDB through CloudWatch and CloudTrail. Amazon CloudWatch Contributor Insights for Amazon DynamoDB (CCID) helps customers understand traffic trends in their applications.

Amazon Neptune

- Deploy at least one read replica to provide for high availability. Use additional read-replicas for scaling write workloads. Make the read replica with the highest failover order the same size as the write master to avoid performance degradation on failover.
- Use CloudWatch metrics such as CPU utilization, freeable memory, and VolumeReadIOPs to help with performance troubleshooting. Alarms and triggers can be used against these metrics to automate scaling during periods of high or low usage.
- Use AWS identity and access management (IAM) users and roles to control access to both Neptune control and data-plane APIs.

- Enable automatic backups and set the backup window to occur at a convenient time based on your RPO.
- When using RDF, logically partition data by storing triples in a named graph. The default named graph is the union of all named graphs.
- For Property Graph, batch multiple vertices (nodes), edges, and properties together within fewer queries when writing data to Neptune in order to optimize write throughput. This also includes using concurrent client threads when performing batch-write operations.
- When loading data into Neptune using the bulk load API for an initial seed, use the largest instance size possible for the write master (db.r5.12xlarge as of this writing) and set the parallelism parameter for the API call to "OVERSUBSCRIBE." Size the write master down to the appropriate size for the workload after the bulk load is completed.

Refer to the following AWS documentation:

- "NoSQL Design for DynamoDB"
- "Read/Write Capacity Mode"
- "Understanding DynamoDB Adaptive Capacity"
- "How Amazon DynamoDB Adaptive Capacity Accommodates Uneven Data Access Patterns (or, Why What You Know About DynamoDB Might Be Outdated)"
- "New Amazon Cloudwatch Contributor Insights for Amazon Dynamodb (Preview) Helps You Identify Frequently Accessed Keys and Database Traffic Trends"
- "Amazon Neptune Developer Resources"
- "AWS Graph Reference Architectures"

Block Storage

Block storage is the oldest and simplest form of data storage. Data is stored in fixed-size chunks called blocks. The granular control that block storage offers makes it an ideal fit for applications that require high performance, such as transactional or database applications. Table 9 lists some of the services in AWS for block-based data storage.

Table 9. Block Data Stores

Service	Description	Use Cases
Amazon Elastic Block Store (Amazon EBS)	■ Persistent block-level data storage for EC2 ■ Stores files in multiple volumes called blocks ■ Flexible offer higher performance than file storage ■ Provides a wide range of choice for configurable IOPS, throughput, and latency profiles ■ Provides flexibility to change volume types and performance profiles without any impact on performance ■ Provides automated backup capabilities to S3 ■ Works best as server disks	■ For throughput and transaction intensive workloads at any scale ■ SSD-backed storage for transactional workloads (IOPS) ■ HDD-backed storage for throughput workloads (MB/sec)

Source: Gartner (June 2020)

Message

Message queues are central to many distributed systems and often provide a backbone for asynchronous processing and communication between. Table 10 lists some of the services on AWS for block-based data.

Table 10. Message Data Stores

Service	Description	Use Cases
Amazon SQS	■ Provides message exchange capabilities through a polling model ■ Decouples sending and receiving components ■ Allows data retention for up to 14 days ■ Provides “at least once” or “exactly once” (FIFO) delivery semantics ■ Message size up to 256KB	■ FIFO queue implementation ■ Message processing with no duplicates ■ Event processing ■ Store messages for later batch processing ■ Asynchronous decoupling of application components
AWS Managed Streaming for Apache Kafka (MKS)	■ Data retention is configurable ■ Provides “at least once,” “at most once” or “exactly once” delivery semantics ■ Flexibility in configuring the message size ■ Provides guaranteed ordering of messages within a partition	■ Stream processing ■ Event processing ■ Streaming analytics ■ Store messages for later batch processing ■ Asynchronous decoupling of application components
Amazon Kinesis Streams	■ Provides maximum data retention for seven days ■ Provides “at least once” delivery semantics ■ Object size is limited to 1MB ■ Provides guaranteed ordering of message within a shard ■ Dynamic scaling of stream capacity to meet real-time, throughput needs ■ Enables replay of messages and fan out to multiple consumers	■ Stream processing ■ Streaming analytics
Amazon Kinesis Data Firehose	■ Pairs Kinesis streams with out-of-the-box Lambda functions to specify transformation and formatting ■ Enables transformation of data into Parquet format before delivery to Amazon S3 ■ “At least once” delivery semantics	■ Load streaming data into data lakes, data stores and analytics tools

Source: Gartner (June 2020)

In-Memory

In-memory data stores optimally support high-throughput and low-latency use cases. This is primarily due to how data is stored (memory versus disk) within the data store itself (retrieving data from memory is orders of magnitude faster than disk). Amazon ElastiCache is fully compatible with two popular In-Memory OSS engines, including Redis and Memcached. Caching use cases such as database query, API caching and session management are commonly attributed to both engines. Redis users also build applications serving a variety of additional use cases due to Redis' support of various advanced data structures and capabilities. Table 11 lists some descriptions and use cases applied to Amazon ElastiCache.

Table 11. Caching Data Stores

Service	Description	Use Cases
Amazon ElastiCache	■ Redis and Memcached protocol-based access (works in lockstep with OSS releases) ■ Can support up to 170TB of data ■ Available across multiple AZs and cross-region (ElastiCache for Redis) ■ ElastiCache delivers higher throughput for Redis leveraging AWS platform enhancements	■ Performance improvement of web applications ■ Session store ■ Real-time ingestion ■ Leaderboards ■ Database query caching ■ Session management ■ API caching ■ Geospatial look-ups ■ Messaging and notification systems ■ Rate limiting applications ■ IOT data ingestion

Source: Gartner (June 2020)

Best Practices

Some recommended best practices for caching data stores are:

Amazon ElastiCache

- Understand the frequency of change of your data and set appropriate time-to-live (TTL) corresponding values to it.
- Select an eviction policy that meets your data needs. Eviction policies apply across the entire dataset within a cluster.

- When possible, group and define clusters based on their purpose. For example, consider a cluster specific to session management vs leaderboards. This allows for specific optimizations including eviction policies to be set for each.
- Select multi-AZ deployed topologies for high-availability requirements.
- Scale out horizontally with cluster mode-enabled Redis when to support larger datasets and reduce the “blast radius” of node failures.
- Scale up or down to different instance types (supported in both ElastiCache Redis topologies) if needed.
- For caching needs, ElastiCache can used to reduce pressure from various back-end systems such as databases, object stores and APIs. Any binary-safe data can be cached with ElastiCache.
- Understand your data access patterns including load and read vs write requests. Select appropriate cluster topologies that map to your needs including replica count and number of shards.
- For production workloads, consider instance types starting from M5 and R5, which feature custom hardware and AWS Nitro optimizations leveraged by ElastiCache for Redis instances. These enhancements provide greater performance when benchmarked against the same instance types using non-ElastiCache-optimized EC2 instances.
- Utilize or upgrade to the latest Redis engines when possible as they have the latest upgrades and optimizations. Upgrading to the latest engine has no read downtime.

Ledger

Ledger databases provide a complete, immutable and transparent history of data and use cryptographic hashing to prove the integrity of the data. Transactions are grouped into blocks and appended to the underlying journal storage, never to be overwritten. Data from the journal is projected into views to provide easy access to current committed state and data history. Ledger databases operate in a centralized architecture where data is owned and controlled by a single entity and trust is centralized. Table 12 lists the ledger services on AWS for immutable data stores.

Table 12. Ledger Data Stores

Service	Description	Use Cases
Amazon Quantum Ledger Database (QLDB)	■ Provides immutability and verifiability without the complexity of blockchain ■ Serverless, fully managed, and highly available ■ Document-style, nonrelational database interface with a SQL-like query language (PartiQL) ■ ACID transactions, serializable isolation level, optimistic concurrency ■ Upcoming streaming feature provides downstream integration	■ Systems of record ■ Asset chain ■ Financial ledger ■ Registrations ■ Healthcare

Source: Gartner (June 2020)

Best Practices

Some recommended best practices for ledger data stores are:

Amazon Quantum Ledger Database (QLDB)

- Use a ledger database instead of a blockchain when data should be owned and controlled by a single, trusted entity.
- Create complex, hierarchical document structures to represent complete business transactions, but separate volatile data elements into separate documents to reduce storage costs incurred by data duplication in document versions.
- Perform targeted reads against indexed, top-level document fields. Avoid queries requiring table scans.
- Write first to QLDB as the system-of-record and stream data out to other purpose-built database engines to support reporting, ad hoc or OLAP-style queries, and high-velocity reads.

How to Select Data Stores in AWS

Before making a storage selection — ask the following questions:

- What is the data structure required for storage? (Fixed, schema-free, key value, graph, etc.)
- How will the data be written and accessed? (Random reads, random writes, sequential reads, sequential writes, REST API, SQL, search.)
- What is the write/read frequency or rate and latency requirements?
- What is the temperature of the data?
- What is the length of time for which the data would be stored?

- What is the cost of data storage?
- What are the durability requirements?
- What are the expected volumes of data during read/write?

Below are some recommendations for choosing across the wide variety of data storage available on AWS:

- Amazon EBS provides persistent block storage for use with EC2 instances. Use EBS if there is a need for a file system abstraction in AWS. EBS has a higher failure rate (0.1-0.2% per year) compared to S3 ("eleven nines" of durability per year), so use regular snapshots.
- Amazon EFS, Amazon FSx for Windows File Server, and Amazon FSx for Lustre are AWS's file services that provide shared file storage for use with AWS Cloud services (EC2, VMware Cloud on AWS, WorkSpaces, etc.) and on-premises resources. All three of these services offer fully managed, highly available, highly durable and secure shared file storage that provides consistent, predictable performance. Consider the following to choose the right file service for your use case:
  - EFS offers a shared, cloud-native file system accessible via NFS that is simple and elastic for general purpose workloads. It provides up to multiple GB/sec throughput per file system, with up to tens of thousands of IOPS at single-digit-ms latencies using general-purpose mode, and up to hundreds of thousands of IOPS at higher latencies with max I/O mode. Use cases include research scientist home directories, web serving and content management and data analytics.
  - FSx for Windows File Server offers fully managed file storage built on Windows Server accessible via SMB. It provides submillisecond latencies and multiple GB/sec throughput and hundreds of thousands of IOPS per file system. Use cases include home directories, business applications, content management systems, media and entertainment workflows, and database applications.
  - FSx for Lustre offers the world's most popular high-performance file system, Lustre, as a fully managed service integrated with Amazon S3 for compute-intensive workloads that require submillisecond latencies and high levels of performance scalability — up to hundreds of GB/sec throughput and millions of IOPS. Use cases include machine learning, high performance computing (HPC), video processing, and big data and financial analytics.
- Amazon EBS and AWS's file services (Amazon EFS, Amazon FSx for Windows File Server, and Amazon FSx for Lustre) are both faster than Amazon S3, with higher IOPS and lower latency. If you have large quantities of data, such as large analytic workloads, AWS's file services (Amazon EFS, Amazon FSx for Windows File Server, and Amazon FSx for Lustre) are best suited for your use case. Data at this scale cannot be stored on a single EC2 instance allowed in EBS, requiring users to break up data and distribute it between EC2 instances. The file services allow concurrent access to thousands of EC2 instances, making it possible to process and analyze large amounts of data seamlessly.
- To store potentially huge objects and to process individual objects at a time, choose S3.

- Use Amazon EFS for applications that require POSIX-compliant shared file systems.
- Use Amazon EFS for applications that require concurrent, consistent access to file systems across multiple AZs.
- Use Amazon EFS if you need a fully managed file service with lowest TCO, instead of building and managing your own shared NFS file systems.
- Use Amazon EFS if your container application requires persistent storage.
- EBS volumes are designed for an annual failure rate (AFR) of between 0.1% and 0.2%. Regular snapshots are recommended for data protection.
- Use Amazon FSx for Lustre when the need is to have a managed file system optimized for compute-intensive workloads. Amazon S3 is cheapest for data storage alone. However, there are various other pricing parameters in S3, including cost per number of requests made, S3 analytics, and data transfer out of S3 per gigabyte. To store potentially huge objects and to process individual objects at a time, choose S3.
- EBS is only available in a particular AZ, while file sharing is possible between AZ on multiple EFS instances. EBS and EFS are both faster than Amazon S3, with high IOPS and lower latency. EBS is scalable up or down with a single API call. Since EBS is cheaper than EFS, use it for database backups and other low-latency interactive applications that require consistent, predictable performance.
- Choose Amazon EFS file workloads whose capacity requirements are unknown. EFS automatically scales up or down based on actual usage, and you pay only for what you use, so there is never any overprovisioning or underprovisioning. Choose EFS Infrequent Access for backup, archival and infrequently accessed files where applications need continued access to the data via the file system, not as an S3 object.
- Use Amazon EFS when there is a need for a scalable file system for use with AWS cloud services like Amazon EC2 and on-premises resources.
- Use AWS Backup for a storage service that allows the user to centralize and automate the backing up of data across EBS, EFS, and storage gateway (volume) services.

Choose Amazon DynamoDB:

- For request processing with high concurrency and strong consistency requiring atomic updates of single items and conditional updates. DynamoDB also supports performing coordinated reads and writes across multiple tables and items using ACID transactions.
- If you need to store small bits of structured data, with minimal latency, and potentially need to process groups of objects in atomic transactions.
- When there is a need for predictable low latency.
- When there is a need for atomic operations.
- For use cases requiring lower response time in the microsecond range. DynamoDB Accelerator (DAX) has fast response times for eventually consistent data, since, DynamoDB response time

is typically in single-digit milliseconds. Use DAX as in-memory cache and for read heavy or bursty workloads — real time bidding, social gaming, etc. Its advantage is that it provides increased throughput and operational cost savings by reducing the need to overprovision read capacity units.

- For creating serverless applications
- Do not use DAX for applications that require strongly consistent reads or those that are write-intensive. DynamoDB and DAX are discussed in more detail in “Assessing the Optimal Data Stores for Modern Architectures” and “Evaluating the Operational Databases on Amazon Web Services.”

Choose Amazon Keyspaces (for Apache Cassandra) Services:

- For existing Cassandra workloads written in CQL that are currently running on self-managed infrastructure and are difficult to scale up and down.
- To use an open-source API with a fully managed and serverless wide-column datastore.
- For Cassandra workloads that require high availability, durable storage and predictable performance at-scale.
- For creating serverless applications.

Choose Amazon ElastiCache:

- To accelerate workload performance whether the need is to reduce latency or increase throughput.
- Amazon ElastiCache is commonly used to complement various back ends and data stores including databases, object stores or APIs, for example, Amazon ElastiCache for Redis is often used for real-time data look-ups, geospatial queries, messaging systems and leaderboards.

Data Processing

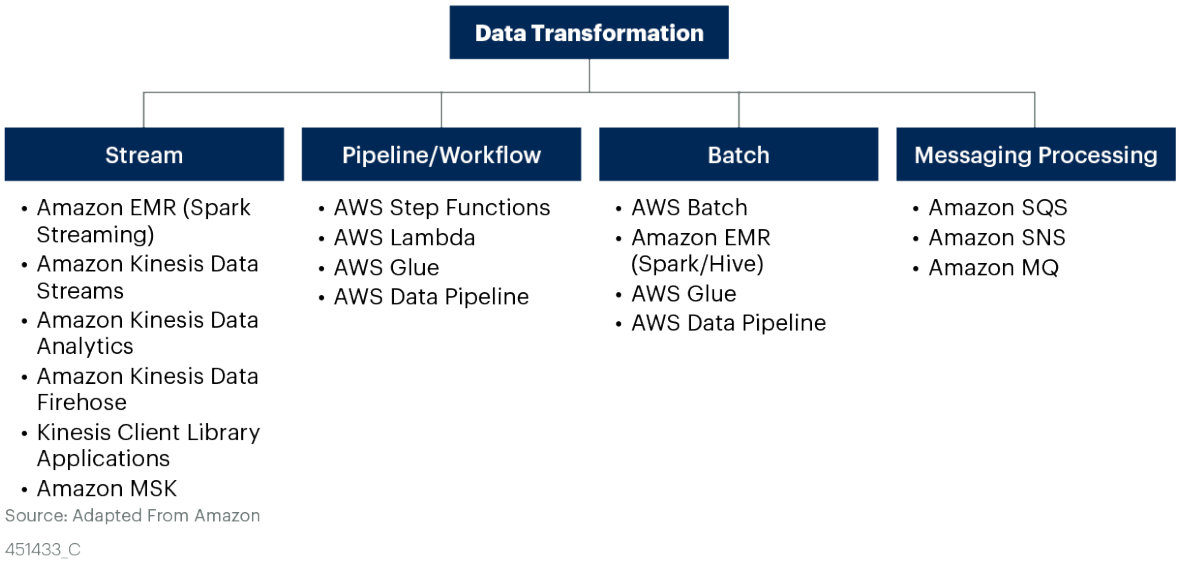
After data is saved to cloud storage, it may need to be processed to make it useful. AWS provides several services based on a wide variety of technologies and frameworks for data processing, apart from traditional data warehouses and SQL engines. Having a data processing framework that can scale to handle large scale datasets and leverage variety of hardware for compute resource makes choosing the right data processing framework a much more challenging proposition.

The AWS ecosystem provides a variety of services across the spectrum of processing workloads, from batch, streaming, messaging and interactive. Some of these frameworks leverage AWS serverless capabilities, while others are managed services.

Data Processing Options in AWS

Figure 5 shows the available services across the different categories of data processing in AWS.

Figure 5. AWS Data Processing Capabilities



Batch Processing

One of the oldest and still the most widely used data processing pattern is batch. Table 13 lists some of the batch processing services in AWS.

Table 13. Batch Processing Options

Service	Description	Use Cases
Amazon EMR	▪ Consists of Spark, Hadoop, Presto, Hive, Pig, Apache Hudi, HBase, Flink, and dozens of services in a bundled and configurable deployment model ▪ EMR supports autoscaling, multimaster for persistent clusters, is fault tolerant for node failures, with native integration with other services including Amazon S3, Amazon Redshift, Amazon Kinesis, Amazon DynamoDB and AWS Glue Catalog	▪ Data processing for batch and ETL, and low-latency interactive queries with Presto
AWS Batch	▪ Dynamically provisions the optimal compute resources needed by the jobs based on their resource requirements ▪ Batch workloads are built as Docker image	▪ ETL, genomics, machine learning ▪ Monte Carlo simulations, media rendering
AWS Glue	▪ Fully managed and serverless service ▪ Define the ETL pipeline in AWS Glue, which generates Apache Spark or Python ETL code ▪ Serverless Data Catalog service that serves as metadata for data in Amazon S3, Amazon Redshift, Amazon DynamoDB, and Amazon RDS databases as well supported on on-premises databases ▪ Supports crawlers to scan S3, Redshift, RDS, etc. data and populate metadata catalog ▪ Supports streaming ETL to consume data from streaming sources likes Amazon Kinesis and Apache Kafka.	▪ ETL, data catalog, data ingestion ▪ Clean, transform and continuously load results into Amazon S3 data lakes, data warehouses, or other data stores ▪ Serverless orchestration: Build and schedule complex DAGs to express Spark jobs, Python jobs, crawlers in a single workflow with built-in logging and monitoring using AWS CloudWatch
AWS Data Pipeline	▪ Data integration web service that is robust and highly available ▪ Create and submit pipelines using Console, CLI, SDK or CloudFormation	▪ Create advanced data processing workflows with ETL jobs that are fault-tolerant, repeatable, and highly available

Source: Gartner (June 2020)

Best Practices

Some recommended best practices for batch processing services are:

Amazon EMR

- Profile the application to figure out runtime characteristics. This gives an understanding of the minimum CPU, memory and storage required. This is critical before trying to optimize spot costs.

- Use newer releases/versions of Amazon EMR to leverage new features and improvements such as the EMR Runtime Environment for Apache Spark.
- Different Amazon EMR releases might have different versions of the open-source applications (e.g., Spark, Presto, HBase). Ensure versions are compatible with the application being developed.
- Once the minimum application requirements are known, instead of resorting to fixed instance types, bid across a variety of instance types, which can give higher chances of getting a spot instance to run an application.
- EMR now provides Spark History Server and Spark logs directly from the console (even for terminated clusters). Consider a log aggregation tool like Kibana for introspection.
- Accidentally suspended or failed jobs can be costly. You can enable clusters to autoterminate on failures as described in "Configuring a Cluster to Auto-Terminate or Continue."
- Use AWS automation tools to switch off EMR clusters that are idle, as described in "Optimize Amazon EMR Costs With Idle Checks and Automatic Resource Termination Using Advanced Amazon CloudWatch Metrics and AWS Lambda."
- With EMR, one pays for EC2 capacity and the service fees, along with the fact that EMR optionally syncs task logs to S3. Be mindful of storage costs and PUT requests to S3, and disable logging if not needed.

AWS Batch

- AWS Batch runs jobs as Docker containers, and workloads need to be containerized.
- Determine whether your jobs should be submitted single, or in an array (array jobs), which are good for functions like parametrics sweeps and Monte Carlo simulations.
- Use Amazon EC2 Spot when possible for interruptible workloads to achieve cost savings. The Spot capacity optimized allocation strategy can launch Spot instances in the deepest spot capacity pools and reallocate capacity as needed in the event of interruption.
- Use Amazon CloudWatch, which integrates with AWS Batch to trigger process events such as data landing in S3 or a daily ingest schedule.
- Use AWS Batch's defaults when possible, particularly services roles and Amazon Machine Image (AMI).
- Use Amazon EC2 launch templates with AWS Batch, which can allow pass-through to EC2 instances for configurations such as user data, without needing a custom AMI.
- AWS Batch tends to bin-pack jobs onto machines when possible; you can control this behavior by specifying instance type in the compute environment.
- For cost or throughput considerations, use AWS Batch's controls in the compute environment to set max, desired, and minimum vCPU allowed.

AWS Glue

- Start with some noncomplex data pipelines with Glue for Batch or stream processing.
- For more complex data transformation workflows, use AWS Glue Workflows. AWS Glue Workflows enable orchestration of data workloads by building dependencies between Glue entities, as triggers, crawlers and jobs. They provide a single pane of glass from which to view all logs in a complex DAG, as well as run, monitor, and troubleshoot end-to-end workflows. Customers that need additional capabilities or integrations with external services can always use Glue's built-in integration with AWS Step Functions, which coordinates serverless workflows across multiple AWS technologies.
- Take advantage of bookmarks, which are very useful for debugging and error handling, and for running incremental data processing jobs and recovering for errors.
- Do capacity planning before launching Glue jobs (data processing units [DPUs]). Default Glue uses the DPU as the unit of processing. One DPU is four vCPU and 16GB of RAM with 50GB of disk. Each DPU runs two executors. For more memory-intensive jobs, Glue offers G1.X (4 vCPU, 16GB of memory, 64GB disk) providing one executor per worker or G.2X, where each worker maps to two DPUs (eight vCPU, 32GB of memory, 128GB disk).
- Apply partitioning, compress and optimal data formats to optimize the transformations and IO bandwidth as recommended for typical Apache Spark workloads.
- Use dynamic frames, as this avoids knowing schema upfront. If you don't want to use dynamic frames, remember that AWS Glue DynamicFrames provides specific features over Apache Spark DataFrames, where each record is self-describing, does not need to conform to a single schema, and comes with out-of-the-box transformation options like ResolveChoice and Relationalize, and out-of-the-box sinks to write to Redshift.
- Use the out-of-the-box transformation library.
- The existing startup times is tens of seconds for Glue Python shell jobs, and is eight to 10 minutes for Glue Spark jobs. This generally depends on the amount of resources requested and availability of resources. Factor this in when launching your applications with strict SLAs. Reduced start times for Spark jobs in AWS Glue, currently in public preview, reduce startup times for Glue Spark jobs to under one minute. (See "Reduced Start Times for Spark Jobs in AWS Glue Preview.")
- Understand that AWS Glue lacks the ability to find new sources as well as provide the compliance workflow before data is made available within the lake. It can only find changes within existing data sources.

AWS Data Pipeline

- Use Amazon EC2 Spot Instances for transient and cost-driven pipelines, and for testing workflows.

- Use Amazon EC2 Reserved Instances for workloads/jobs that run for at least 70% of the time (in a year).
- Data Pipeline-managed resources (resources created by data pipeline) are provisioned and terminated by Data Pipeline. Self-managed resources are managed by customers and won't be provisioned or terminated by DPL.
- For high frequency/concurrent data workflows, EC2 and EMR provisioning times are a bottleneck to workflows. In these situations, increased throughput can be obtained by using Task Runner on existing resources. This bypasses the provisioning step that AWS Data Pipeline-managed resources entail, cutting pipeline times to completion significantly.

Stream Processing

Stream processing is a technology that allows for collection, integration and analysis of data in real time, as data is produced, without requiring data to be stored prior to analysis. Table 14 lists some of the services for stream processing in AWS.

Table 14. Stream Processing Options

Service	Description	Use Cases
Amazon Kinesis Data Streams	▪ Streams are automatically replicated and durably stored across three distinct AZs ▪ Data encrypted in transit and at rest ▪ Pricing model based on number of shards ▪ Dynamically scale number of shards based on incoming throughput ▪ Attach up to 20 simultaneous, real-time consumers to Stream without impacting stream throughput ▪ Pricing model based on number of shards	▪ Stream processing ▪ Streaming analytics ▪ Ingest and store streaming and event-based data
Amazon EMR (Apache Spark or Apache Flink)	▪ Spark structured streaming on EMR with all capabilities of native Spark streaming	▪ Stream processing ▪ Streaming analytics
Amazon Kinesis Data Analytics	▪ Streaming SQL engine and Apache Flink for Java-based stream processing as its two runtime options ▪ Stateful stream processing for real-time analytics and long running computations ▪ Supports advanced time series analytics, window-based functions, machine learning, and more ▪ Underlying compute capacity dynamically scales in response to processing needs ▪ Pay-as-you-go based on Kinesis Processing Unit (KPU)-hour	▪ SQL on streaming data and analytics on data in Kinesis ▪ Apache Flink-based stream processing on data in Amazon Kinesis Data Streams, Amazon MSK, Amazon DynamoDB streams, and custom data stores
AWS Lambda	▪ Run serverless code ▪ Process events from Kinesis in real time ▪ Increase concurrency per shard	▪ Building event-driven, serverless applications
Amazon Managed Streaming for Apache Kafka (Amazon MSK)	▪ Fully managed service to build applications with Apache Kafka ▪ Very easy to set up a new cluster ▪ MSK encrypts data at rest using AWS Key Management Service (KMS) customer master key (CMK) ▪ MSK encrypts data in transit between the brokers of your MSK cluster	▪ Build streaming and real-time event-driven applications ▪ Stream processing ▪ Streaming analytics

Service	Description	Use Cases
AWS Glue streaming ETL	▪ Built on the Apache Spark structured streaming engine ▪ Ingests streams from Amazon Kinesis Data Streams, Apache Kafka and Amazon MSK ▪ Can clean and transform streaming data and load it into Amazon S3 or JDBC data stores	▪ IoT streams ▪ Clickstreams ▪ Network logs

Source: Gartner (June 2020)

Best Practices

Some recommended best practices for stream processing services are:

AWS Lambda

- Minimize deployment module size/footprint. Leverage smaller and faster frameworks and minimize package size. Use only needed SDK modules.
- Use configuration and environment variables for operational behavior change at runtime.
- Put dependency .jar files in a separate/lib directory.
- Improve dependency injection with smaller and simpler inversion of control frameworks that load quickly on startup, like Dagger2.
- Make sure functions invoked by Amazon Simple Queue Service (SQS) don't exceed the visibility timeout.
- AWS Lambda uses Amazon Linux underneath the covers, which means any binary dependencies that will be used inside a Lambda function must first be compiled in the same environment.
- Functions are invoked in ephemeral stateless containers and should never rely on data from a previous invocation.
- AWS Lambda may incur additional overhead latency on the first request, depending on the runtime and function package size. If the first request latency is critical, use Lambda's provisioned concurrency feature to guarantee double-digit millisecond overhead latencies. Keep in mind that response size is critical. If application has a lot of data to be returned from the computations, using Lambda may not be the right choice. AWS Lambda limits any requests and responses to a function to 6MB, so you will need to use an intermediate store like Amazon S3 to pass large amounts of data between your service and an AWS Lambda function.
- Consider memory usage of long-running processes before using an AWS Lambda function, especially when working with large datasets or doing video or image processing. AWS Lambda is not a perfect candidate for a memory-intensive work.

- Be aware of AWS Lambda limitations of 512MB ephemeral disk space and 250MB uncompressed deployment package size.
- AWS Lambda provides a maximum of 3GB for every invocation; if your application has higher requirement, consider AWS Fargate.

Amazon Kinesis Data Streams

- Amazon Kinesis Data Streams offers enhanced fan-out capability that enables 20 real-time consumers to read from the same stream without any reads per second (TPS) limits through a pure stream-based data retrieval.
- Amazon Kinesis Data Streams uses very AWS-specific APIs. Be aware of the potential future costs of migrating away.
- Amazon Kinesis Data Streams' shards by default, when used with RESTful GetRecords API, permit five reads per second, but each retrieve up to 10 MB/sec per read. If data is evenly distributed across many shards, read limit for the stream will remain at five reads per second on aggregate because each consuming application will need to check every single shard for new records. This puts a hard limit on the number of different consuming applications possible per stream for a given maximum read latency. To add more consumers and overcome the limit, use Kinesis Data Streams' SubscribeToShard API, which provides dedicated 2 MB/sec throughput.
- Leverage Amazon Kinesis Scaling Utility, an application offered by AWS Labs, which can monitor metrics via CloudWatch and scale Kinesis streams based on configuration.
- Beware of read throughput, which is 2 MB/sec per shard.
- Amazon Kinesis has a bottleneck in shards per streams. Create larger streams with more shards so that the records are picked up quickly.
- Remember the retention period. Kinesis has a maximum retention period of 168 hours or seven days.

Amazon Kinesis is discussed in more detail in "Hyperscaling Streaming Analytics: Comparing Stream Analytics in the Cloud With Amazon, IBM and Microsoft."

Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Be aware of the following when looking to work with Amazon MSK:

- With Amazon MSK, modifying a running cluster to add or remove broker nodes or change instance type is not supported. These operations require creating a new cluster. Only updating the cluster configuration "update-cluster-configuration" command and increasing EBS storage associated with MSK brokers' "update-broker-storage" can be done without re-creating the cluster.

- The “default” monitoring level is free, and organizations have to pay extra for “per broker” and “per topics per broker.” AWS offers Open Monitoring for Prometheus, which is a free feature that lets customers source JMX metrics from the Amazon MSK cluster.
- There is no managed schema registry, and Amazon MSK lacks the ability to drop in jars in Kafka (for metrics reporter, specific classes, Java agents). Topics cannot be managed from the console.
- Message and retention limits are determined by configuration with default broker limit per customer to 30 brokers. Enterprises need to request a limit increase in the AWS Support Center if they need more than 30 brokers within a cluster.
- Customers can run Kafka Connect and other third-party tooling like schema registries with Amazon MSK, but AWS does not manage the connectors.
- There’s no point-and-click-based backup to S3.

Amazon Kinesis Data Analytics

- The two different runtimes offer very different experiences. The SQL runtime is simple for building many use cases but is limited by the language limitations of SQL. The Apache Flink runtime supports simple to sophisticated applications, but customers must learn the open-source framework.
- For Apache Flink, leave fault tolerance capabilities like checkpoints and savepoints off during development. Turn them on prior going to production.
- Properly test the parallelism of your application prior going to production for both SQL and Apache Flink runtimes. It is key to getting the automatic scaling capabilities to work for you.

Familiarity with Amazon CloudWatch logs, metrics, and insights is required to properly monitor and debug your application. You will need to know these tools well to effectively use the service.

Message Processing

Message processing is a technology that allows for processing of messages in real time. Table 15 lists some of the services for message processing in AWS.

Table 15. Message Processing Options

Service	Description	Use Cases
AWS SQS	■ Fully managed messaging and queuing service ■ Reliably and continuously exchanges messages across a wide variety of data sources ■ Ability to increase message throughput without preprovisioning ■ Pay only for message volume	■ Build decoupled microservices applications ■ Building serverless applications ■ Asynchronous decoupling ■ Batch processing

Source: Gartner (June 2020)

How to Select Data Processing in AWS

This section outlines some criteria to select the data processing services on AWS based on the data processing paradigm. Table 16 outlines some of the distinguishing characteristics when selecting ETL and batch-based data processing on AWS.

Table 16. Comparing ETL Tools on AWS

	AWS Glue	Amazon EMR	AWS Batch
Use Cases	Ingest, transform, catalog	ETL, streaming analytics, interactive analytics, machine learning	Async task that can be encapsulated in a container
Compute	Serverless	Amazon EC2	Amazon EC2 on-demand/spot on Docker
Infrastructure	Serverless	Amazon EC2	Amazon EC2 and Amazon ECS
Pricing	DPUs	Pay a per-second rate for Amazon EC2 and Amazon EMR	Pay for Amazon EC2
Storage Integration	Amazon S3/Amazon DynamoDB/Amazon Redshift	Amazon S3/Amazon DynamoDB/ Amazon Redshift	Amazon S3/Amazon DynamoDB/Amazon Redshift
AWS Service Integration	Step function	Step functions, Glue Data Catalog, CloudFormation, IAM, KMS	Step functions and CloudWatch
Runtime	Spark, Python	Big data frameworks such as Spark, Hive, Presto, HBase, Flink, and Hudi	Any runtime programming language
Limits	Number of concurrent jobs	Number of nodes (EC2 instances)	Queues (20) and compute environments (50)

Source: Gartner (June 2020)

Table 17 provides a high-level comparative overview between AWS Kinesis and Kafka.

Table 17. Comparing Amazon Kinesis Data Streams and Apache Kafka

Amazon Kinesis Data Streams	Apache Kafka
Tightly integrated with AWS ecosystem, API and tools	Open-source compatibility/third-party tooling
Throughput provisioning model	Cluster provisioning model
Configuration options very limited	Highly configurable
Payload size restrictions	Payload size restriction, but more liberal
Retention period cannot be greater than seven days	Configurable retention time
Complexity with Amazon Kinesis Data Streams is without additional components like Amazon Redshift or Amazon DynamoDB; it is unable to look up dimension data or update aggregates	Look-ups can be implemented in Kafka with tools like KSQL and Kafka Streams
All messages are automatically replicated to three AZs	All operationalization with Kafka are within one's own purview
Scaling with Amazon Kinesis Data Streams is much easier — just an API call	Scaling could mean adding brokers to a cluster, then rebalancing partitions
Maximum message size is 1MB	Kafka's message can be bigger

Source: Gartner (June 2020)

Table 18 outlines some of the distinguishing characteristics when selecting Stream and message-processing-based services on AWS.

Table 18. Selecting Stream and Message Processing Services on AWS

Service	Spark Streaming	Kinesis Client Library Application	Amazon Kinesis Data Analytics	SQS Application	Serverless (AWS Lambda)
AWS Managed	Yes (EMR and AWS Glue)	No (EC2 + AutoScaling)	Yes	Yes (AWS Lambda)	Yes (AWS Lambda)
Serverless	No	No	Yes	Yes (AWS Lambda)	Yes
Scale/ Throughput	No limits to the number of nodes	No limits to the number of nodes	▪ No limits on Apache Flink-based applications. ▪ SQL apps have maximum vCPU of 64.	No limits to the number of nodes	No limits
Availability	Single AZ	Multiple AZx	Multiple AZs	Multiple AZs	Multiple AZs
Programming	Java/Scala/ Python	Java, Python, Node.js, Ruby, .NET	▪ SQL runtime supports extensions through AWS Lambda ▪ Apache Flink runtime supports Java, Scala, Python, and SQL	AWS SDK, (Java, .NET, Python, Go, JavaScript, Node.js, PHP, Ruby, C++)	Java, Go, PowerShell, Node.js, C#, Python, Ruby, and custom runtimes for other languages
Use Cases	Multistage processing	Parallel processing of all shards	Stateful stream processing	Simple event-based triggers	Parallel processing of all shards, triggered by batch of records
Reliability	Spark checkpoints	Managed by KCL	Managed by Kinesis Data Analytics	Managed by SQL Visibility Timeouts	Managed by AWS Lambda

Source: Gartner (June 2020)

Data Consumption

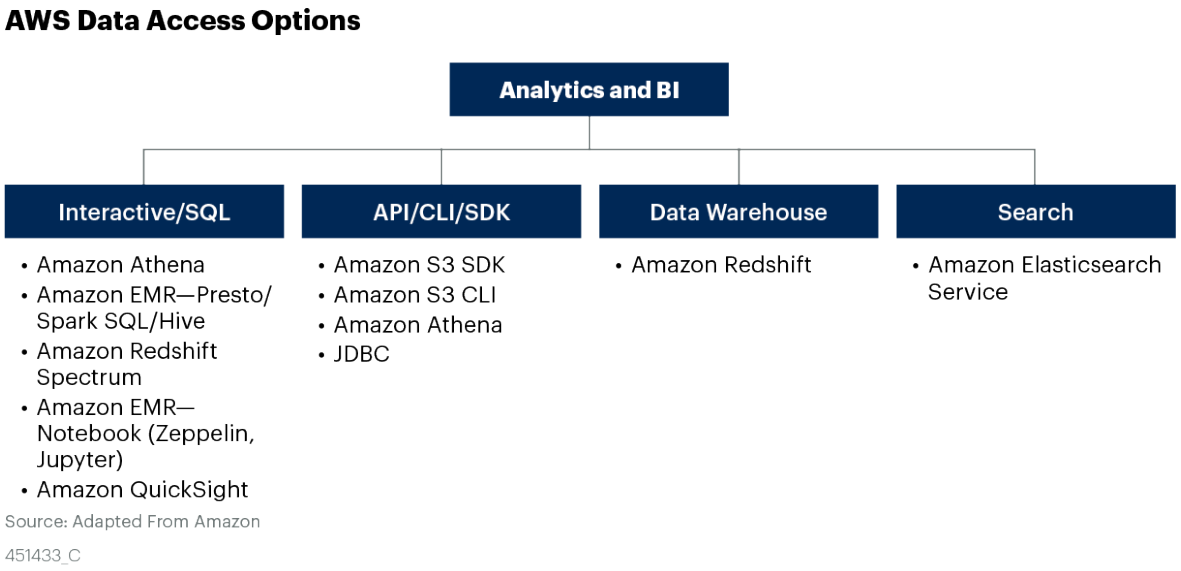
The serving layer delivers the raw data as well as output of processing and analytics to the various data consumers. This layer is responsible for:

- Serving data to consumers who want to access data from raw storage without going through a data warehouse. Data consumers require direct access to the data, typically for data science, data exploration, data discovery and experimentation. This allows them to work with the raw, unprocessed data and moves data experimentation workloads outside of a warehouse to avoid performance impacts. There are multiple ways to provide direct access to the data in the lake. Cloud providers provide a SQL engine that executes queries directly on the files in the cloud storage or in data lakes. For data science experimentation, workload data is often copied from the data platform into sandboxes.
- Data warehouses that provide a data access point for data consumers that require full SQL support and expect data to be presented in a relational format. Such data consumers include dashboards, BI applications, data analysts or power users familiar with SQL. Cloud-based data warehouses are architected to have the following distinguishing characteristics:
  - Scalable and reliable: Work efficiently with large and small data and scale beyond the capacity of a single computer. Should be able to continue to serve data in the face of inevitable failures of individual components.
  - NoOps: Should require little to no tuning or operational maintenance.
  - Elastic cost model: Ability to scale with load. Warehouses experience most of their workloads during massive, precanned burst report generation or during business hours. Cloud cost models should be able to accommodate this variation of spectrum with an efficient cost model.
- Data consumers aren't always humans. For example, results of real-time analytics are rarely consumed by a human. Outputs from real-time analytics and operational systems are consumed by other applications, such as recommendation, notification and alerting systems. Such programmatic data consumers require a dedicated API to consume data from the data stores or outputs using built-in APIs.

Data Access Options in AWS

Figure 6 shows the data access categories and services in AWS.

Figure 6. AWS Data Access Options



Data Warehouse

A data warehouse is a centralized data repository that consolidates vast amounts data from a variety of source systems and organizes them with historical changes. It is invaluable for organizations that require analytical capabilities for improved decision-making and business insights. Table 19 lists the services on AWS for building a data warehouse.

Table 19. Data Warehousing Options

Service	Description	Use Cases
Amazon Redshift	▪ Managed service ▪ Supports columnar storage with data compression and zone maps ▪ Uses internal data catalog as semantic layer ▪ Supports UDFs	▪ Data warehousing ▪ Low latency queries for self-service dashboards

Source: Gartner (June 2020)

Best Practices

Some recommended best practices for data warehousing service like Redshift are:

Amazon Redshift

- Amazon Redshift provides Auto Workload Management (Auto WLM). Customers should use this feature. When workloads have mixed concerns (example: ELT, ad hoc queries and Dashboards), it may make sense to create query priorities, which is a powerful and flexible way to prioritize complex workloads.
- Individual query queues can support concurrency scaling, allowing automatic scaling and addition of transient clusters. This feature is generally used to address an increased number of queries or number of users. Use concurrency scaling to handle spikes in your workload, allowing you to reduce the size of your main cluster. Redshift can spin up to 10 additional clusters to absorb those spikes and will automatically spin them down when no longer needed. Redshift provides 30 free hours (one each day a cluster is operating) per month, so be sure to use them.
- If you see important queries queue up, consider adding query priority roles, enabling both short query acceleration (SQA), and concurrency scaling. If the period of heavy workload is known in advance, in minutes, temporarily increase the cluster with Elastic Resize, either from the console, or schedule it, also from the console.

When considering Redshift optimizations:

- Redshift has powerful capabilities for distributing data (optimizing join performance), sorting data (enabling range restricted scans) and compressing data. When starting out, data distribution is defaulted to AUTO and can be further monitored in the Redshift Advisor (AWS Console). These optimizations can be applied manually with an ALTER TABLE command. Compression is automatically enabled through preset values; optimal storage compression settings can be calculated on an existing table with the ANALYZE COMPRESSION command. Sort keys need to be added on the main predicate column (usually a date or time stamp) for large tables, Redshift Advisor will monitor the workload and surface recommendations.
- Use KEY distribution style when joining tables with the same distribution keys (dist keys). Performance is optimized, as the colocated data avoids unnecessary shuffling when merging. Avoid using distribution keys with low cardinality columns, or where the key may cause skew (example: a large number of null values). Data skew reduces parallelism.
- Avoid running OLTP-type workloads in Amazon Redshift. While queries such as SELECT * and UPDATE do run, these types of operations are better suited for Amazon Aurora.

More details about Redshift's recent changes and advancements are outlined in detail in "Comparing Cloud Data Warehouses: Amazon Redshift and Microsoft Azure SQL Data Warehouse, 2019 Update."

Interactive

Interactive data access is needed when the end user does not necessarily know all the questions to ask ahead of time. As questions are asked, answers by the query engine raise new questions, based on thought process. Table 20 lists the services in AWS for interactive data processing.

Table 20. Interactive Data Processing Options

Service	Description	Use Cases
Amazon Athena	■ Managed serverless service ■ Supports multiple data formats — Avro, Parquet etc. ■ Integrates with Glue catalog ■ Invoke an ML model deployed on Amazon SageMaker	■ SQL querying of data on S3 for use by data analysts, business analysts and data scientists ■ Athena Query Federation (in Public Preview) supports running SQL queries across data stored in relational, nonrelational, object, and custom data sources
Amazon Redshift Spectrum	■ Managed service ■ Supports multiple data formats — Apache Avro, Apache Parquet etc. ■ Integrates with Glue catalog ■ Supports UDFs	■ Query data on S3 using SQL
Amazon QuickSight	■ Managed Business Intelligence Service ■ Integrates with AWS services such as Amazon Redshift, Athena, Aurora, RDS, etc.	■ Interactive exploration of data via direct query for data in databases

Source: Gartner (June 2020)

Best Practices

Some recommended best practices for interactive data processing services are:

Amazon Redshift Spectrum

- Use optimized data formats to improve performance and lower costs, Amazon suggests using columnar data formats such as Apache Parquet. It is common when maintaining very large datasets to use Redshift to transform and clean data and then export that data to S3. Redshift has a powerful UNLOAD command that can maintain partitioned files for efficient access.
- Spectrum supports the following formats Avro, Parquet, TEXTFILE, SequenceFile, RCFile, RegexSerDe, ORC, CSV, Ion and JSON.
- Setting up Amazon Redshift Spectrum requires creating an external schema and tables. Amazon Athena Data Catalog, AWS Glue Data Catalog or Apache Hive metastore, such as Amazon EMR can be used to create an external schema. External tables are read-only and won't allow you to perform insert, update or delete operations.
- Make sure that the data files in Amazon S3 and the Amazon Redshift cluster are in the same AWS region.

- Provide authorization to access external Amazon Athena data catalog, which can be done through an IAM console.
- Ensure that the data architecture team is aware of the limitations around Amazon Redshift Spectrum, which include:
  - Doesn't support nested data types, such as STRUCT, ARRAY and MAP.
  - When using the Athena data catalog, users must be cautious when performing queries as each account is limited to 100 databases, and each database is limited to 100 tables.
  - Maximum of 20,000 partitions per table. This can be increased by contacting AWS support.

Amazon Athena

- Use automatic partitioning of data. This allows you to optimize the amount of data scanned by each query, thus improving performance and reducing the cost for data stored in S3 as you run queries.
- Convert data formats to Parquet. This lowers costs when executing queries because the Parquet files are highly optimized for interactive query services like Athena.
- Ensure that the data architecture team is aware of the limitations around Athena, which include:
  - Athena is not a database and does not support all DML statements like Update and Delete but supports Create Table as Select (CTAS) and INSERT INTO statements to extract, transform, and load (ETL) data into Amazon S3 for data processing.
  - Doesn't support stored procedures or materialized views or user-defined functions (UDFs). Amazon Athena recently announced support (preview) for Amazon Lambda-based UDFs.
  - Amazon has imposed some soft limits on queries and by default allows users five concurrent queries running for each account.
  - Each account has a soft limit of 100 databases, and databases cannot have more than 100 tables. Soft limit defaults for the Glue catalog that serves Athena are quite high — 10,000 data per account and 200,000 tables per database. Customers are expected to raise the limits as per their needs. While Amazon Athena can access data from a different region than the one that initiated the query, a limited number of regions are supported.

Search

Table 21 lists the services on AWS for performing search across data stored within the AWS stores.

Table 21. Search Processing

Service	Description	Use Cases
Amazon Elasticsearch Service	■ Fully managed search service with Kibana ■ Integrates with other AWS services	■ Search-based applications

Source: Gartner (June 2020)

Best Practices

Some recommended best practices for search-based services are:

Amazon Elasticsearch

- Ensure that the slow logs feature is enabled for Amazon Elasticsearch Service (ES) clusters.
- Ensure Elasticsearch nodes are using general-purpose SSD storage instead of provisioned IOPS SSD storage or ephemeral SSDs (I3 instance family) to optimize the service costs.
- Ensure at-rest encryption and encryption-in-transit is enabled for Amazon Elasticsearch Service domains.
- Ensure high availability for Elasticsearch clusters by specifying three availability zones with one or more replicas configured on your indexes.
- Ensure Elasticsearch Service (ES) domains are only accessible to users and applications that require access.
- Ensure ES domains are using dedicated master nodes to increase the production environment stability.
- Ensure Elasticsearch Service (ES) domains are not exposed to everyone.
- Ensure Elasticsearch clusters are using dedicated master nodes to increase the production environment stability.
- Make sure the architecture team is aware of limitations of Elasticsearch, which include limited access to admin cluster APIs. This limits the ability of experienced operators to intervene without a support request if a cluster experiences degraded performance that is not automatically remediated by the managed service.

How to Select Data Access in AWS

This section discusses how organizations should go about selecting data access solutions on AWS:

- Use Amazon Redshift Spectrum when you want to join Amazon S3-based data with data in your data warehouse. This allows low-latency interactive look-ups from your DWH, and in the occasional case when you need additional data, join it with data from S3.
- Amazon Redshift Spectrum queries may run slower than queries in an Amazon Redshift cluster, mainly because of data movement between S3 and the cluster. But if that trade-off is fine and cost is the priority, a combination of Amazon S3/Amazon Redshift Spectrum is a great choice.
- Amazon Redshift Spectrum can optimize access to very large, well-partitioned datasets that can result in minimal scanning and increase cost-effectiveness.
- Amazon Redshift Spectrum offers the best of both worlds. With Spectrum:
  - Continue using analytics applications, with the same queries for Redshift.
  - Leave cold data as-is in S3, and query it via Amazon Redshift, without ETL processing. That includes joining data from the data lake with data in Redshift, using a single query.
  - The pay-for-usage pricing model is very attractive, and the interface is simple to use and intuitive for SQL.
  - With Amazon Redshift RA3 and managed storage, you have the choice of leaving data in Amazon Redshift for datasets that have hot datasets that move in ways that are not obvious (such as the most recent data).
- Organizations using Amazon S3 in search of an ad hoc query service find Athena an ideal solution.
- Amazon Athena integrates with BI tools; however, for an existing Amazon Redshift user, Amazon Redshift Spectrum is a better choice and also integrates with all the major BI tools.
- Amazon Athena has a REST API access. Originally, it was JDBC-centric. (See "Amazon Athena Adds API/CLI, AWS SDK Support, and Audit Logging With AWS CloudTrail.")

There are two main alternatives to running Elasticsearch on AWS:

- Elastic Cloud, a hosted solution on AWS managed and maintained by Elastic. It supports the most recent versions immediately following its release, and provides almost all the visibility needed, and most of the extensibility needed — including log visibility and uploading custom plug-ins. Upgrading to newer versions of ES without any downtime and cluster resizing on the fly are also supported out of the box.
- Running your own Elasticsearch cluster on Amazon EC2 is cheaper and gives full control, visibility and accessibility. Deployment is easy if done using Terraform, and sizing and monitoring are as easy as with a cloud offering.

Data Operationalization and Monitoring

Data operations teams need to understand business needs to efficiently support business process operation. Such teams rely on a wide variety of tools to collect metrics to measure business outcomes. Teams need to have a shared understanding and a bird's-eye view as well as granular

metrics of all end-to-end processes and pipelines across all the workloads running on the system spanning infrastructure, governance, applications and deployment units.

Data Operationalization and Monitoring Options in AWS

Table 22 lists the services in AWS for monitoring and operations.

Table 22. Data Operationalization and Monitoring Options in AWS

Category	Service	Description	Use Cases
Security	Amazon CloudWatch	▪ Monitoring of resources and applications ▪ Set alerts based on metrics and thresholds ▪ Monitoring with automation ▪ Correlation and root cause analysis	▪ Monitor cloud resources and applications running on AWS ▪ Collect, track metrics, monitor log files, set alarms and automatically react to changes in AWS
Security	AWS CloudTrail	▪ Logs every AWS public control plane API usage and events; CloudTrail also logs Data API events for S3 and Lambda	▪ Governance, compliance, operational and risk auditing ▪ Security and compliance Automation
Alerting	Amazon Simple Notification Service (SNS)	▪ AWS CodeCommit, GitHub, and Amazon S3 can publish to a start-build SNS topic when a change happens ▪ Is integrated with Amazon SageMaker build to other systems	▪ Starting builds and for notifications
Configuration	AWS Systems Manager Parameter Store	▪ Durable, centralized, and scalable data store; AWS will store the parameters of training jobs and deployment	▪ Secrets management ▪ Application and other services parameter management ▪ Secure, hierarchical storage for configuration data management
Compliance or Configuration Governance	AWS Config	▪ Fully managed service that provides an AWS resource inventory, configuration history and configuration change notifications to enable security and governance	▪ Use to determine permissions that belonged to a user or group at a specific time ▪ Continuously monitors and compares current configuration to a desired state ▪ Can alert or remediate when noncompliant
Configuration	AWS Service Catalog	▪ Create, manage, monitor and distribute portfolios of approved products to end users	▪ Control which users have access to specific products to enforce compliance with organizational standards and manage product life cycles

Category	Service	Description	Use Cases
Security	AWS Trusted Advisor	▪ To identify common security misconfigurations, suggestions for improving system performance, and underutilized resources	▪ Service limit check ▪ Specific port unrestricted check ▪ IAM and MFA check
Security	Amazon GuardDuty	▪ Monitors AWS account access, service APIs, network and DNS activity for signs of compromise	▪ Integrated threat intelligence such as lists of known malicious IP addresses, anomaly detection, and machine learning to identify activity associated with threats
Security	AWS Security Hub	▪ Comprehensive view of your high-priority security alerts and compliance status across AWS accounts	▪ Aggregates, organizes and prioritizes security alerts, or findings, from multiple AWS services
Security Orchestration	Amazon Macie and AWS Step Function	▪ Sensitive data discovery and preventive control monitoring	▪ Monitors access controls like bucket policies, bucket encryption ▪ Uses techniques including ML to detect sensitive data types, inspect object ACL permissions
Orchestration	AWS Step Functions	▪ Acts like a state machine ▪ Uses AWS Lambda to transform state changing, branching or loop in through state	▪ Coordinate multiple AWS services into serverless workflows to build and update apps
Cost	AWS Budgets	▪ Set custom budgets that alert you when costs or usage exceed	▪ Set reservation utilization or coverage targets and receive alerts when utilization drops below defined thresholds
Cost	AWS Cost Explorer	▪ Interface that lets visualize, understand, and manage AWS costs and usage over time	▪ Create and analyze custom reports to analyze cost and usage data
Resilience	AWS Health	▪ Visibility into the state of AWS resources, services, and account ▪ Supports an API to integrate with other AWS services	▪ Personalized view of the status of the AWS services that power data applications
Performance	AWS X-Ray	▪ End-to-end, cross-service tracing view of requests	▪ Instrument requests to applications

Category	Service	Description	Use Cases
		▪ Creates a map of services used by data applications	

Source: Gartner (June 2020)

Best Practices

Some recommended best practices for operationalization and monitoring are:

- Use custom tags for cost control and accountability.
- Monitor the ingestion process. This is fundamental to ensure that the various consumers in the business have a view of the data they need at all times.
- Use Datadog-like tools to monitor a number of indicators, like throughput per event type, AWS Glue costs, Lambda functions error rates and AWS S3 delivery performance and freshness. Datadog tracks the execution of AWS Glue crawlers and ETL jobs to notify you when something fails and when a defined schedule does not complete successfully.
- Use AWS CloudFormation to create version-controlled templates for infrastructure.
- Set up continuous integration/continuous deployment (CI/CD) pipelines using the AWS developer tools (for example, AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeDeploy and AWS CodeStar).
- Adopt practices that allow you to detect defects early, and fix or work around them safely in production.
- Use the artifacts from AWS CloudFormation and AWS developer tools to share design standards.
- Apply metadata using tags to enable identification of resources for operations activities.
- Use AWS CloudFormation to define dashboards in code instead of clicking through the AWS Management Console.

Data Security, Privacy and Governance

Data governance has become the linchpin in operationalizing analytics. A common refrain in Gartner inquiries is the time organizations take to put machine learning models into production because of lack of trust and knowledge of the data. Data governance has become imperative in successful delivery of analytics.

While guardrails and discipline are needed to manage data, it should not come at the cost of agility and hamper innovation. Another challenge is that cloud data governance is complex, and the tools are not yet mature. Partners extend the native data governance capabilities of the cloud vendor to solve for customer-specific data governance problems. AWS has released some products to address this space that are covered in this section.

The area of governance is very vast. Some of the technical areas it spans include:

- Account and identity governance
- Storage governance
- Cloud cost management
- Data governance
- Data access governance

This document is focused on data governance, which has two aspects:

- **Regulatory.** Handles personally identifiable information (PII) data, which could differ between industries. For example, banking is concerned with PCI, while healthcare must protect protected health information (PHI). Amazon Web Services (AWS) supports governance and compliance processes, including performing assessments, demonstrating compliance and achieving certifications, such as HITRUST and FIPS 140.2-validated services, and a SOC 1 Type 2 accreditation program. AWS provides tools to help with regulations such as the General Data Protection Regulation (GDPR), the Basel Committee for Banking Supervision 239 (BCBS239) and California Consumer Privacy Act (CCPA). HITRUST is an audit and accreditation widely accepted in the healthcare industry and aligned to the Health Insurance Portability and Accountability Act (HIPAA) requirements.
- **Data democratization/data monetization.** Deals with the ability to make data-driven decisions through data that can be trusted.

Data governance dictated by regulatory and compliance reasons has become more important as laws have become stricter and threaten to impose heavy fines for not securing personal data. Organizations are required to demonstrate complete lineage of the analytics workflow for sensitive data. Regulations such as the EU GDPR mandate security and privacy by design. This type of data governance is nonnegotiable, but it is also defensive in nature. It concerns protecting sensitive data to protect users' privacy.

Security and privacy controls need to be in place whenever the access privileges are given to humans or even to downstream applications and programs such as AI. Organizations looking to set up data governance should ensure they are able to answer the following questions:

- **Why is the data being collected in the first place?** The first iteration of data lakes focused on unfettered ingestion of data from internal and external sources, leading to data swamps. Today, organizations link data ingestion to high-level use cases and build data lakes in an agile manner. If your data includes privacy data, then your governance tool must allow storing users' consent. While not all compliance regulations require this yet, it is a good discipline to associate consent so that when new regulations are introduced, you are prepared.
- **What data is being collected and from which sources?** This is needed to ensure that data is being collected from the single source of truth. Ensure your data profiling tool can determine the

file formats and data schemas. Additionally, your tool should be able to classify and "tag" the data. Ascertain the quality of data and integrate with data quality remediation process.

- **Is the data secured while it is transmitted and at rest?** Your governance requirement may require that the source data, data in motion and target data be encrypted. Please note that encrypting data is not enough, as application security is still needed for common threats such as SQL Injection and cross-site scripting.
- **Can private and sensitive data be discovered?** Business users demand self-service capabilities to augment internal data with external sources. So, handling sensitive and private data must be automatic. Your data governance program must be able to support all the needed formats that may be structured or may even be on documents such as Microsoft Word, or PDF or Office 365 emails. The data governance program is heavily reliant on processes, organization, culture, policies and technology. This document is focused on the technology aspects, although other aspects are mandatory to address to achieve a secure and governed modern data and analytics architecture.
- **Is the metadata collected for data ingestion, integration, and transformation and enrichment tasks?** This activity shows more complete data lineage. For example, data governance tools such as Collibra can collect metadata on AWS Glue tasks.
- **How is the data shared?** Many data catalogs today also catalog the end-user reports such as Tableau. They can detect how changes in the sources can impact reports.
- **Who has access, and is the access aligned with internal roles and policies?** The answer to this question helps enable role-based access control (RBAC) to the data. Access control can further be extended to "tags," which is called attribute-based access control (ABAC).
- **What compliances need to be supported?** Some regulations, such as GDPR and CCPA, are generic, while others, like PCI-DSS, pertain to financial services dealing with credit or debit cardholder data when HIPAA compliance is required by healthcare entities handling personal health information.
- **How is the PII data defined?** What is considered private data may differ between organizations and even regulations such as GDPR and CCPA. Organizations also must be cognizant that once users are de-identified, data transformation tasks do not inadvertently reidentify the users.
- **Do I have a workflow to remediate compliance needs?** While this document is focused on technological aspects, align your processes to handle requests to delete or "forget" data.

The second aspect of data governance is to unlock the full potential of all your data assets by providing governed and secure access with complete audit and lineage. While the compliance regulations are concerned with securing private and sensitive data, this aspect facilitates enablement of all enterprise data.

Organizations should have a process to onboard new datasets, but before they are made visible to the end users, they have the right policies enforced. Some of the areas where data governance helps data-driven organizations include:

- **Improved decision making** by identifying and remediating data quality problems on historical and current data.
- **Enabling new use cases** such as revenue growth, customer satisfaction, etc.
- **Higher operational efficiencies** through self-service business data access, automation of data life cycle pipelines and reduced time in performing root cause or impact analyses in case of downtime.
- **Better risk management** by reducing the risks of losing customer trust in data and incurring fines through better audibility.

This section focuses on data governance and not on security infrastructure, which is equally important. A brief overview of AWS capabilities in the other types of governance follows.

AWS Identity and Access Management (IAM) allows organizations to securely manage access to its various services and resources. AWS IAM does not incur any charges because its sole purpose is to enable access to other services. Some of the best practices to govern resources include:

- **Limit the number of principals** (users, roles, compute resources) with privileged permissions using the IAM policies. IAM policy is attached to each user or group of users or roles. These policies are global and apply to all regions. Develop fine-grained policies that consider data sovereignty and localization requirements; for example, create policies that restrict data processing and storage to a specific jurisdiction or geographic region.
- **Identify unused roles** and take measures to reduce overly permissive policies. Features such as “role last used date,” Access Advisor and Access Analyzer can be used to identify areas where customers can reduce the scope of permission and resource policies.
- **Ensure the data locality** governance requirement by defining a policy that ensures that the data stays local to the specified region.
- **Control data access at the network level.** IAM policy can specify which regions are allowed and which are rejected. A good recommendation is to give “least privileges.” Organizations can “allow list” or “block list” IP addresses. In addition, organizations can set up disaster recovery (DR) on their data, preferably in a different region and under a different account.
- **Control data access at the storage level.** IAM policy can allow read-only access to certain S3 buckets and restrict writes or deletes. Another best practice is to turn on versioning on Amazon S3 objects.
- **Control data access at the data security level.** IAM policy can deny unencrypted writes to Amazon S3 objects. Data security policies can also be applied at the application level, e.g., EMR or SageMaker. AWS organization policies can be useful to enforce this kind of practice across a customer’s entire environment.
- **Secure your encryption keys.** AWS provides fully managed key management service (KMS) with features such as key rotation. However, some Gartner clients prefer to manage keys on their own and even have it on-premises.

The IAM policies mentioned above can be encapsulated in AWS CloudFormation templates.

Amazon Macie

AWS provides a security and sensitive data discovery service called Amazon Macie to identify and classify sensitive data such as personal identifiable information, source code and credential material. As this document has stated, Amazon S3 has become the de facto storage layer for raw data in AWS due to the ease of loading and retrieving data. This means that it is easy to load data that may have sensitive data embedded in it. Macie uses ML to reduce false positives and contains built-in regex functions such as a Luhn check for identifying credit card numbers.

The current version of Amazon Macie is optimized for data residing Amazon S3. The input is an S3 bucket or buckets, and the service inspects the contents and produces findings for any sensitive data discovered. For data not natively residing in Amazon S3, a temporary secure Amazon S3 environment can be staged, data can be pushed into an S3 bucket temporarily for Macie to inspect, and results can be reviewed or acted upon using Amazon CloudWatch Events and AWS Lambda. Amazon Macie continues to expand the file formats, the sensitive data types that it can detect, and the native support for data stores within AWS.

Amazon Macie uses the following steps:

- **Identify document type.** Macie has ability to detect sensitive data in various formats, including CSV, JSON, PDF, XML, TXT, Parquet, Avro and Microsoft Office files. The service also supports common compression formats such as GZIP, TAR and ZIP.
- **Classify data.** Macie uses multiple techniques, including regular expression (regex) and ML to identify a growing list of fully managed sensitive data types. It can securely inspect data encrypted with AWS Key Management Service and supports customer-defined regex definitions.
- **Analyze and understand access controls.** Macie does this both at the Amazon S3 bucket and object level by consciously assessing bucket configurations and inspecting object ACLs during data classification.
- **Integrate with DevOps workflows.** Findings are pushed to Amazon CloudWatch Events. Users can write Lambda functions to act, e.g., delete buckets, inform and change resource tagging.

Amazon Macie automates discovery of sensitive data as new data sources are added. Findings are sent to Amazon CloudWatch Events or external tools like Splunk and Atlassian Jira.

AWS CloudTrail enables governance and compliance by auditing what each user does, and allows customers to store the logs in an S3 bucket. As logs can become very big, admins can define a life cycle policy to limit the data retention period. These policies are usually defined by the CISO. AWS CloudTrail continuously monitors all API access. As command line interface (CLI) and AWS SDKs make API calls, these are automatically included. It is enabled by default. The most recent 90 days of account activity history is free of charge and is visible in the event history in the AWS Management Console. CloudTrail also offers auditing of S3 and Lambda data events. Customers can enable this paid feature and store logs in an S3 bucket.

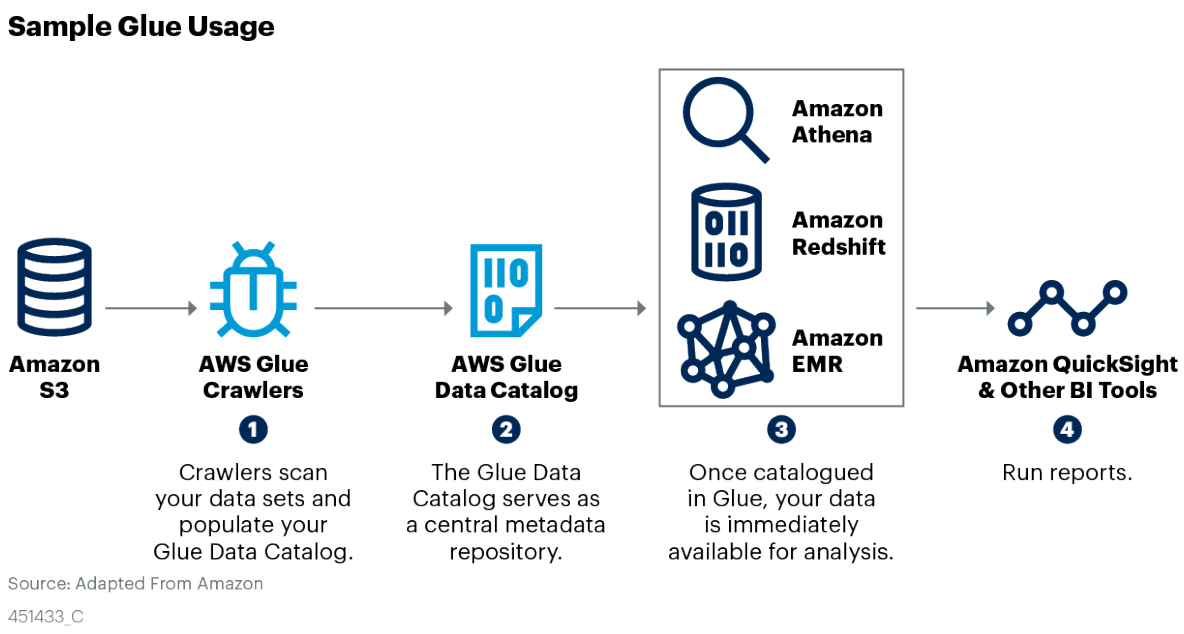
Amazon GuardDuty analyzes AWS CloudTrail events, VPC Flow Logs, DNS logs, and other AWS data sources to continuously detect threats such as malware network activity or unusual or unauthorized access. GuardDuty recently added the ability to monitor threats against S3 resources by analyzing CloudTrail S3 data events. Like the other data sources, GuardDuty obtains access to S3 data events without requiring the customer to enable, and pay for those logs to be delivered by CloudTrail. When S3 data event monitoring is enabled, GuardDuty immediately begins to analyze S3 data events from all of your S3 buckets and monitor them for malicious and suspicious activity such as unusual access patterns or use of credentials from Tor or other known malicious IPs.

AWS Glue

As mentioned earlier in the document, AWS Glue is a fully managed, serverless ETL engine that runs on Apache Spark. It allows users to author ETL jobs in Python and Scala. In addition, AWS Glue is also Amazon's metadata catalog solution. It plays an important role in governing access to the data. The Glue Data Catalog is integrated with Amazon Athena, Redshift and Amazon EMR, Amazon SageMaker and Amazon QuickSight. However, it is not a full-featured data catalog such as Collibra, Alation, IBM Watson Knowledge Catalog or Informatica Enterprise Data Catalog.

Figure 7 below depicts how AWS Glue can help with data security, governance and privacy.

Figure 7. Sample Glue Usage



Source: Adapted From Amazon

451433_C

AWS Glue can be used to process data on Amazon S3 from staging to curation. Its operations include data cleansing, and preparing and optimizing data for querying. All the metadata that is created is stored in the Glue Data Catalog, an Apache Hive-compatible metastore. Once the data is cataloged in the Glue data catalog, it is immediately available for analysis in integrated services like

Amazon Athena, Amazon Redshift, Amazon EMR. Users can now create reports with Amazon QuickSight and other BI tools.

Metadata consists of many types — technical, operational and business. For more information on metadata and overall data governance, see "Building a Comprehensive Data Governance Program." AWS Glue contains crawlers that scan sources and extract technical metadata, which includes table information, file formats, partition keys and schema. AWS Glue allows users to annotate this data with their own tags, e.g., data owner.

In April 2020, AWS launched Serverless Streaming ETL with AWS Glue that adds streaming capabilities to Glue.

Crawlers support structured and semistructured data in AWS native offerings, such as Amazon RDS engines, including Amazon Aurora, Amazon S3 and Amazon DynamoDB. They can also profile Oracle, MySQL, Microsoft SQL Server and PostgreSQL databases that are accessible via JDBC. Glue crawlers support the most popular data formats and sources, including JSON, XML, MySQL and PostgreSQL. Users can add custom classifiers by writing "grok expressions" to support their legacy data formats.

Please note that the Glue data catalog primarily manages technical metadata. While it can annotate the metadata in the data catalog, if your requirements call for a tool to create and manage business metadata, then you will need to explore third-party tools, some of which are already mentioned in this section. The next section, on AWS Lake Formation, extends Glue by providing a unified security model to govern the metadata and data in your data lakes.

AWS Lake Formation

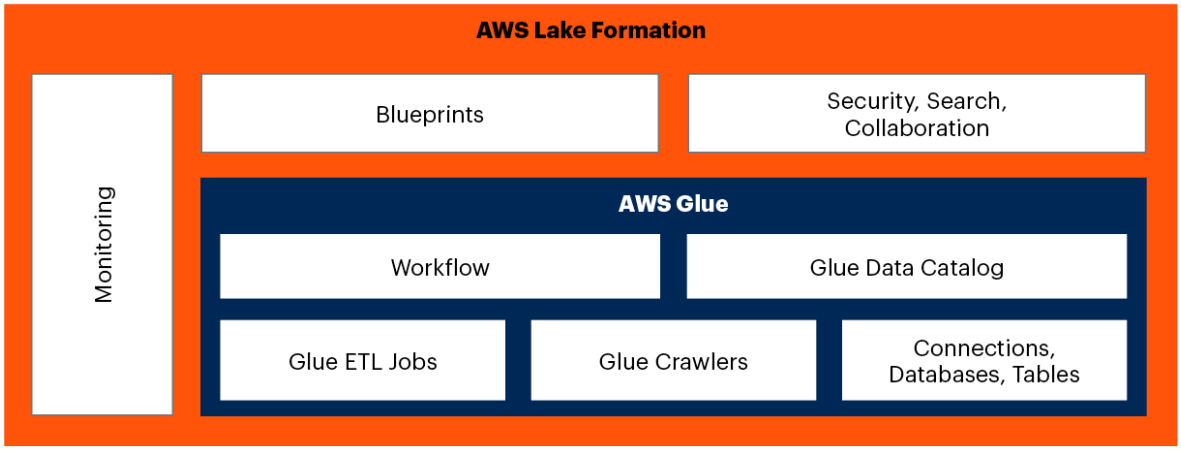
AWS Lake Formation is a fully managed service, built on AWS Glue, to reduce the time and complexity of building a data lake on Amazon S3. It provides a centralized approach for data engineers, data analysts and data scientists to utilize the data lake in secure and governed manner. While AWS Identity and Access Management (IAM) permissions provide coarse-grained security for Glue metadata at the table level, and S3 at the buckets and object level, Lake Formation takes it a level deeper. It provides fine-grained access control on this data, including column level. This allows different types of users to access the same underlying data without duplicating the data with restricted columns removed. Instead, users can now dynamically apply access control policies that enforce their governance requirements.

Lake Formation is designed to make data lakes easier to manage and reduce barrier to entry. It does not rely on complex IAM policies for access control. Instead, Lake Formation permissions are applied to Glue Data Catalog objects (i.e., databases, tables, and columns) and you can define access controls through the AWS Console or SDK/CLI. This reduces error-prone and manual tasks and enables faster ramp-up time for data lakes.

Figure 8 shows the components of the AWS Lake Formation service.

Figure 8. AWS Lake Formation Building Blocks

AWS Lake Formation Builds on AWS Glue



Source: Adapted From Amazon

451433_C

In Figure 8, blue is Amazon Glue and orange is AWS Lake Formation.

As a recap, AWS Glue is a serverless ETL and catalog offering. It consists of the following features:

- **Crawlers.** These infer schema types and formats of the source data. They help ingest source data and collect statistics.
- **An ETL platform.** This uses Apache Spark. As it is fully managed, it provisions the resources needed to run jobs in PySpark, Scala, etc. One of the tasks is to deduplicate data and perform record matching. It uses ML to identify overlaps in very large datasets, which is better than fuzzy matching, which consumes intensive compute resources on large datasets. This provides 90% accuracy in a cost-performant manner.
- **A data catalog.** This is a hive metastore-compatible catalog of databases, schema, tables and columns.
- **A workflow engine.** This orchestrates jobs, monitors and set alerts using AWS Cloud Watch.

AWS Lake Formation is a service built on top of Glue to build and secure data lakes. It offers the following features:

- **Blueprints.** These are built-in templates that ease the process of ingesting data from common data sources. Blueprints create a workflow that is run as a Glue task. Blueprints exist for well-known RDBMSs such as Oracle and MySQL, and for common log formats like AWS CloudTrail, ALB and Amazon Elastic Load Balancing.

- **Simplified permissions management.** Glue and Lake Formation use the same data catalog; they are the same. Lake Formation provides an additional authorization layer to catalog resources, e.g., databases, tables and columns. So instead of just IAM policies governing access to catalog resources, both IAM and Lake Formation policies now govern access. The expected usage is coarse-grained IAM policies and fine-grained Lake Formation policies on the data lake.
- **Consistent access control across analytic services.** Lake Formation permissions are honored by compliant analytics services such as Glue, EMR, Athena, Redshift Spectrum and QuickSight. Support for additional services, including SageMaker and third parties, will be added in future. So you have a single pane of glass to define fine-grained access policies at table, row and column level that are honored across compliant analytic services.
- **ML transforms.** Lake Formation provides more sophisticated transformation jobs than Glue via ML transforms. These transforms are used to match and deduplicate records across diverse datasets.
- **Monitoring.** Workflow and ETL jobs are monitored and audit trails provided that can be analyzed in CloudTrail using Amazon Athena. Lake Formation UI monitors in real time access to the data and permissions.

Lake Formation Steps and How They Overcome Traditional Data Lake Challenges

The process of creating a data lake in AWS can be time-consuming. Lake Formation doesn't change the tasks, but it provides short cuts such as Blueprints and intuitive user interfaces to achieve these tasks faster. These steps are shown in Table 23.

Table 23. Traditional Data Lake Steps Versus Lake Formation Steps

Steps	Traditional Data Lake	Lake Formation
Create S3 buckets, database and tables.	Use AWS Cloud console.	Use Lake Formation UI or the API to register the buckets. Lake Formation will manage these buckets and create databases and tables on your behalf.
Ingest source data. For example, ingest CloudTrail data for one week and load it in Parquet format.	Write PySpark or Scala code in Glue UI Console.	Use Blueprint, which will create Glue workflows on your behalf. These workflows can then be run on demand or scheduled to execute periodically.
Configure access and create bucket security policies.	Use S3 console to create an S3 bucket and object policies in JSON. Hard to manage and update, and needs JSON skills	Use the UI or APIs to GRANT and REVOKE access to data at the table and column level. No complex JSON.
Map S3 locations to data catalog tables/columns.	Manual step in S3 console.	Blueprints, crawlers and workflows manage this for you.
Enforce security policies.	Coarse-grained at object level using cloud identity and access management.	Fine-grained at table and column level. These permissions apply regardless of which analytic engine is used access the data lake (e.g., Athena or Redshift Spectrum).
Data transformation.	Create ETL code to transform and load data into refined zone.	Automatically discover data, infer schema, convert to destination format and partition. Uses ML transform for record matching/deduplication
Create metadata access policies.	IAM policies on Glue databases and tables.	Fine-grained access control, via GRANT and REVOKE commands.
Workflow.	Manual.	Use UI or APIs to schedule tasks or to run on demand.
Monitoring.	Uses CloudWatch.	Real-time monitoring and integrated with Amazon CloudWatch.

Source: Gartner (June 2020)

Strengths

Some strengths of AWS ecosystems for data management platforms include:

- The AWS ecosystem for data management is extremely well-tested with customers across a variety of domains. AWS provides many highly customizable services across areas like ingestion, processing, and integration with diverse features and capabilities.

- The AWS ecosystem includes best-practice out-of-the-box templates and blueprints. Many AWS services are based on open source projects where skills are readily available and to a large extent can ease the high upskilling and learning associated with onboarding.
- AWS holds the edge for organizations with web-scale applications that support a lot of users. When looking for a platform that is feature-rich and scalable, AWS is a good choice and continues to offer new features and updates to attract more customers.

Weaknesses

Some weaknesses of AWS ecosystems for data management platforms include:

- Abundance of features and tools capability overlap, which may complicate usability and selection process. There is a learning curve with AWS, and the complexity of using the tools and making them work can initially slow down organizations that are new to AWS.
- AWS does not include enterprise support by default. It should be purchased as an add-on service.
- No cloud provider has a complete end-to-end data governance suite of tools that can meet business and technical and operational needs. Data catalog tooling in AWS is very rudimentary, and AWS lacks any solutions around data virtualization.

Guidance

- Winning the cloud strategy depends on not just cloud hardware but embracing managed services, serverless, and the entirety of cloud services so that the infrastructure stack functions reliably without managing virtual servers, storage and networking. Simple IaaS alone on cloud is legacy now.
- Leverage AWS native tools for ingestion, processing, data formats, data sources and network boundaries before investing in third-party tools. AWS tools are optimized in terms of cost, bandwidth utilization and speed to work across network boundaries and provide almost all possible solutions required by most organizations to ingest, integrate and process data to the cloud.
- Adopt cloud-native technologies and especially the philosophy of “everything as code” — infrastructure, applications and platforms. Use configuration-driven orchestration by autogenerating configurations, and invest in a DataOps- and DevOps-based approach of building and running data platforms.
- Data governance and data cataloging tools within the AWS ecosystem are still evolving compared to more mature commercial offerings. While metadata management features and fine-grained security controls are available within AWS Glue and AWS Lake Formation respectively, organizations should look to invest in third-party tools, especially for the business data catalog and to track data lineage.

- Keep an eye on the costs. Use the right tools within the AWS ecosystem to monitor costs and budget. Organizations can get carried away by the plethora of choices in the cloud world. This can easily lead to a cluttered architecture and lack of governance in the architecture. Organizations need to ensure they have the right governance in place to manage the diverse services. It is highly recommended that organizations use the tools provided by AWS to monitor usage, resources and costs, and plan their budgets accordingly to prevent paying high costs for suboptimal system usage in the cloud.

Gartner Recommended Reading

“Evaluating the Operational Databases on Amazon Web Services”

“Hyperscaling Streaming Analytics: Comparing Stream Analytics in the Cloud With Amazon, IBM and Microsoft”

“Solution Comparison for AWS, GCP and Microsoft Azure Native Cost Optimization Tools”

“Infrastructure Monitoring With the Native Tools of AWS, Google Cloud Platform and Microsoft Azure”

“Key Services Differences Between AWS, Azure and GCP: Availability and Network”

“Solution Comparison of the IAM Capabilities Within AWS, Azure and GCP”

“Key Services Differences Between AWS, Azure and GCP: Compute and Storage”

GARTNER HEADQUARTERS

Corporate Headquarters
56 Top Gallant Road
Stamford, CT 06902-7700
USA
+1 203 964 0096

Regional Headquarters
AUSTRALIA
BRAZIL
JAPAN
UNITED KINGDOM

For a complete list of worldwide locations,
visit http://www.gartner.com/technology/about.jsp