Best Practices for Building a Data Lake on AWS for Games

AWS Whitepaper

Best Practices for Building a Data Lake on AWS for Games: AWS Whitepaper

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

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Abstract and introduction i
- Abstract 1
- Are you Well-Architected? 1
- Introduction 1

What is a data lake, and how does it benefit game developers? 2
Data lake design patterns and principles 3
- Framework 3
  - 10,000 foot view 3
  - 5000 foot view 3
  - 1000 foot view 4

Lake house architecture 5
Data ingestion 7
- Data collection - third-party service or self-made 7
- Streaming or batch 7
- Client or server 8
- REST API or ingesting directly to a stream 8
- Other sources 9

Data transformation 10
- Event format and changing schema 10
- Transform/analytics engine 10

Data cataloging 13
- Use AWS Glue Data Catalog as a metadata store for your data lake 13
- Other options for data catalog 15

Data lifecycle management 16
Workflow orchestration 18
- AWS Step Functions 18
- Amazon Managed Workflows for Apache Airflow (MWAA) 19

Data security and governance 21
- How to regulate your data (GDPR, CCPA, and COPPA) 21
- Data discovery 22
- Data governance 23

Data visualization 25
Monitoring 26
Cost optimization 27
Further reading 28
Contributors 29
Document revisions 30
Notices 31
AWS glossary 32

Best Practices for Building a Data Lake on AWS for Games

Publication date: **May 11, 2022** (*Document revisions* (p. 30))

Abstract

Many customers run their Digital Games on AWS. Each of them has made unique design choices that allow them to run the fastest, most complex, and visually breathtaking games on AWS. This whitepaper outlines best practices for building a data lake for games on the AWS Cloud, and offers a reference architecture to guide organizations in the delivery of these complex systems.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console (sign-in required), you can review your workloads against these best practices by answering a set of questions for each pillar.

In the Games Industry Lens, we focus on designing, architecting, and deploying your games workloads on AWS.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Introduction

The gaming industry is more competitive than ever before, with thousands of titles published every year, all competing for players' time and attention. Building a successful game in today's market is a challenging endeavor, and it takes an enormous amount of work. Building an integrated data platform to effectively analyze player data helps game developers better understand player behavior and business performance, enable timely and data-driven decision making, and delight players. Today, more and more game developers are building data lakes on AWS to achieve these business outcomes.

This whitepaper provides an in-depth discussion of best practices for building a data lake on AWS for games. It aims to help game developers maximize the value of their player data to achieve better game design, development, monetization insights, and strategies.

What is a data lake, and how does it benefit game developers?

A *data lake* is an integrated and centralized data platform combining data storage and governance, analytics, machine learning (ML), and visualization. It is a secure, durable, and cost-effective cloud-based storage platform that allows you to ingest, transform, analyze, and visualize structured and unstructured data as needed. Using Amazon Simple Storage Service (Amazon S3)-based data lake architecture capabilities, game developers can achieve the following:

- Ingest data from a wide variety of data sources using different combinations of batch, near real-time, and real-time ingestion methods.
- Store ingested data in a cost-effective, reliable, and centralized platform.
- Build a comprehensive data catalog to enable easy access of data assets stored in the data lake.
- Secure and govern all the data and business metadata stored in the data lake.
- Monitor, analyze, and optimize cost and performance.
- Visually map data lineage.
- Transform raw data assets into optimized usable formats and query them in place.
- Easily and securely share processed datasets and results across your organization.
- Use a broad and deep portfolio of data analytics, data science, ML, and visualization tools.
- Quickly integrate current and future third-party data-processing tools.
- Subscribe third-party data in the cloud, load data directly into the data lake, and analyze it with a wide variety of analytics and ML services.

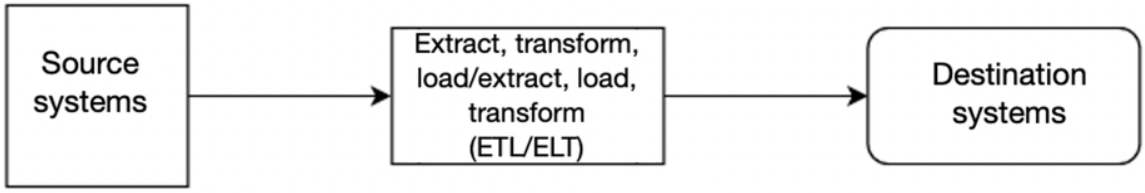
Amazon S3 decouples storage from compute and data processing, and makes it easy to build a multi-tenant environment. The real-time analytics capability that a data lake delivers can help track your most important Key Performance Indicators (KPIs) on performance, engagement, and revenue, and provide a complete view of the player experience and their Lifetime Value (LTV) at any point in their lifecycle. This allows game developers to investigate and understand player behaviors and experiences, make better decisions, and create and deliver new winning games.

Data lake design patterns and principles

Framework

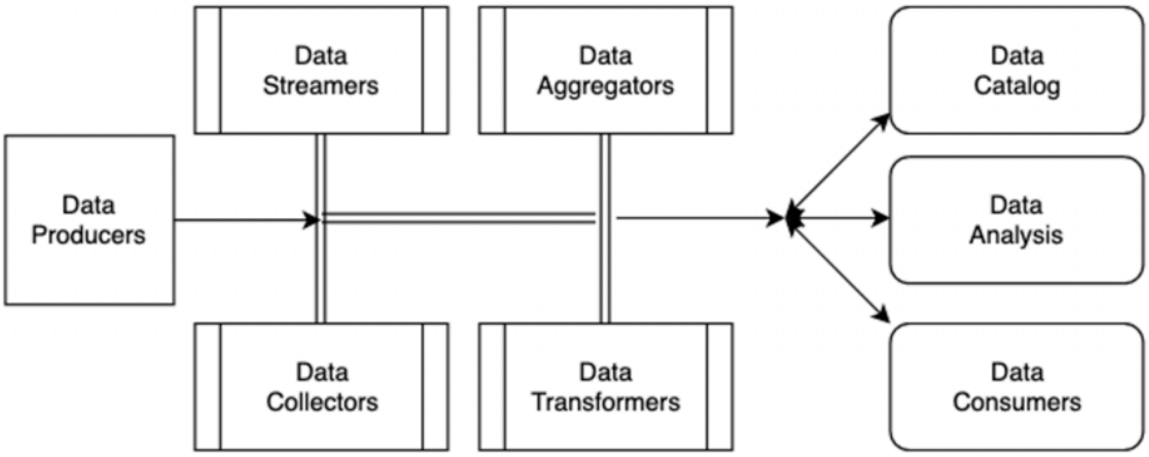
Following is a high-level framework for building a data lake on AWS.

10,000 foot view



This is a 10,000 foot (high level) view of how analytics systems work with source and destination systems.

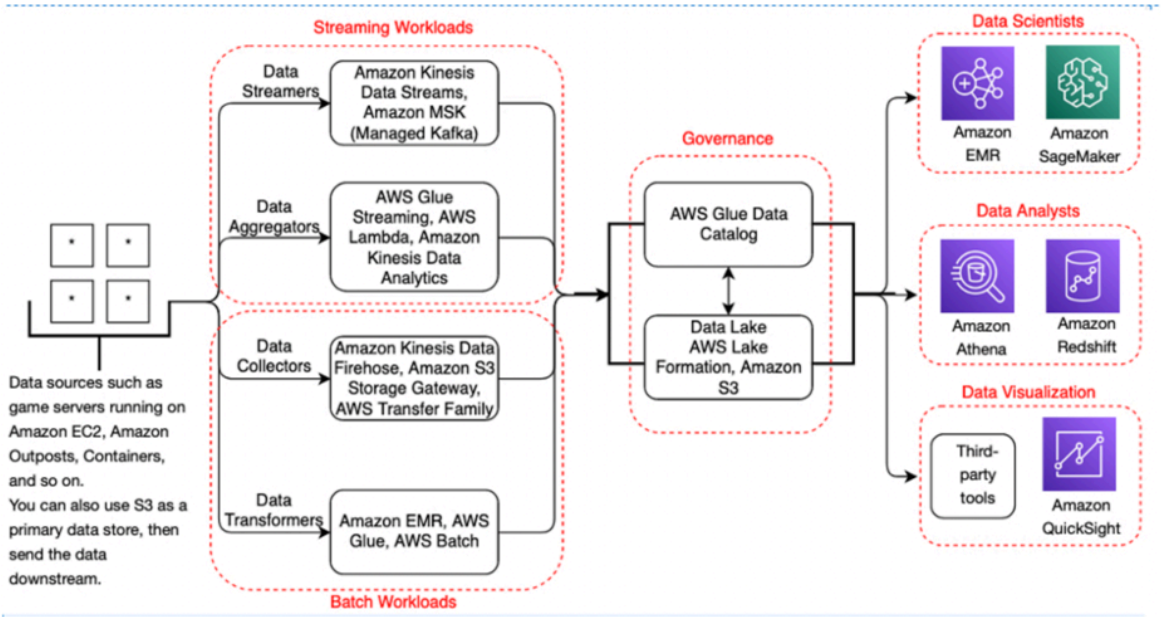
5000 foot view



This is a 5,000 foot (mid-level) view of how analytics systems work with source and destination systems.

Diving deeper into the framework, there are data streamers, data collectors, data aggregators, and data transformers that collect the data from the data producers (sources). Depending on the use-case, data is then consumed for analysis or downstream consumers and cataloged into a data lake for governed access.

1000 foot view



This is a 1,000 foot (detailed) view of how analytics systems work with source and destination systems.

Diving deeper in the framework, Some AWS services are added as an example to show data flow. This layout is a common pattern AWS observed with its customers.

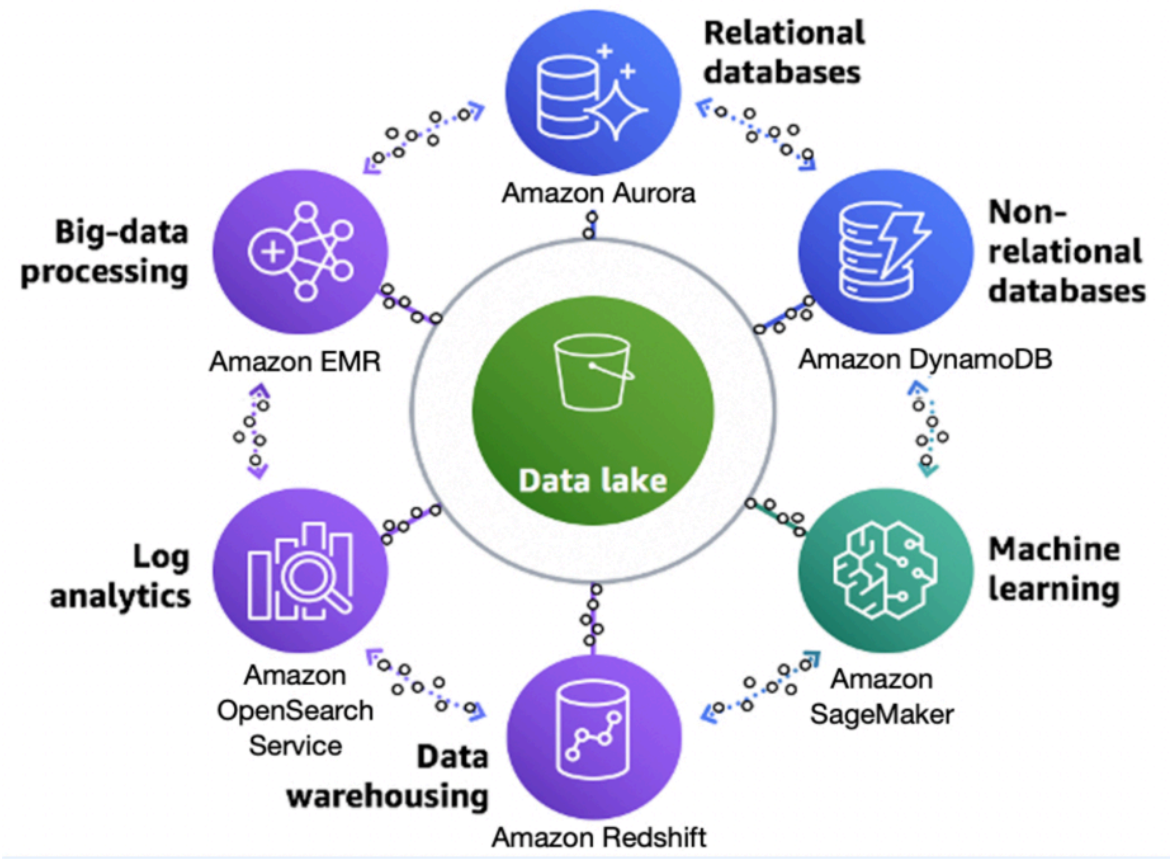
Lake house architecture

Game developers often use data warehouse alongside a data lake. Data warehouse can provide lower latency and better performance of SQL queries working with local data. That's why one of the common use-cases for the data warehouse in games analytics is building daily aggregations to be consumed from business intelligence (BI) solutions. Games can generate a lot of data, even logging activity down to the key stroke. This can result in having to process terabytes of data every day, and the data may reside in, or need to be loaded into, different data stores. These data stores can be a cache such as Amazon ElastiCache for Redis, a relational database such as Amazon Aurora, a NoSQL database such as Amazon DynamoDB, and log data potentially being stored in Amazon S3. The challenge then becomes having to manage that data, and finding ways to get meaningful insights throughout your repositories.

For example, DynamoDB performs well for specific use cases such as reading and writing data with single digit millisecond latency. But it should not be used as a source for your analytical queries, as there is no one tool that is perfect for every job. Having a lake house architecture allows customers to easily move data to and from their data stores in a fast and secure manner. This also allows customers to connect their data lake to their databases and data warehouses using the AWSAWS Glue Data Catalog, which is integrated with many AWS services.

Instead of building a siloed data warehouse, you can use technologies to integrate data lake with it. For example, use Redshift Spectrum to query data directly from the S3 data lake, or the Amazon Redshift COPY command to load data from S3 directly into Amazon Redshift in a parallelized way. Many customers don't want to have to load ten or 20 years' worth of data into their data warehouse when they rarely need to query it. Extending your data warehouse to a data lake is a great option in this case, as you can keep your historical (cold) data in your data lake to help save on cost, and use your data warehouse to query your data lake when necessary.

Refer to Derive Insights from AWS Modern Data for more details, and the Build a Lake House Architecture on AWS blog entry for a deep dive.



AWS lake house architecture

Data ingestion

Game developers collect and process different type of events from various sources. Typical examples include marketing data from the game and third-party services (clicks, installs, impressions) and in-game events. Before you can transform and analyze this data in the data lake, it needs to be ingested into a raw region of the data lake. This chapter discusses different data ingestion mechanisms and design choices.

Data collection — third-party service or self-made

Sometimes game developers develop their own solutions to generate and ingest events. For example, they can develop their own mobile tracker and own the code that generates and ingest events. In this case, they can use the technology stack and data ingestion methods of their choice. However, sometimes game developers rely on partner services to collect data. Examples include mobile attribution services such as AppsFlyer, or third-party mobile trackers such as Amplitude or Google Analytics. An ingest solution depends on data export options that such third-party service provides. The most common is a batch export to an S3 bucket, where you can pick up files from there with a batch extract, transfer, and load (ETL) process.

Streaming or batch

You can stream events or load them in batches. Streaming means that a game or a partner service ingests events as they are generated. A streaming storage service such as Amazon Kinesis Data Streams or Apache Kafka stores these events, and consumers can read them in the order of arrival.

However, ingesting every single event is inefficient - it generates excessive network traffic and might lead to high cost depending on the streaming storage and service used. That's why some degree of batching is often used with streaming to improve performance. For example, events are sent to a stream after every battle in the game, or once in five minutes.

Batch ingest means that a pack of events is accumulated on a server, or in a partner service, and then is uploaded as a single file containing multiple events. You can upload directly into a raw region of your data lake, or upload to a separate bucket and apply some kind of transformation when copying to the data lake. The latter is usually used when importing data from third-party services.

Streaming is a good choice in most cases. Advantages of streaming include:

- Lower risk of losing events if the client crashes, if there is a network outage (which is normal for mobile games), and so on.
- Shorter time to report, you don't need to wait hours for the next batch to have fresh events available in the data lake.
- Real-time analytics capabilities with a streaming framework such as Apache Flink or Spark Streaming.
- Support of multiple independent consumers in parallel. For example, you can save data in the data lake and write it to a time-series database at the same time.
- Cost becomes a factor when we are dealing with large volumes of data (stream or batch). Refer to the pricing page for the respective AWS service to understand the costs associated with per stream, data ingested, data retrievals, and data stored.

Sometimes batch processing is the option. Advantages of batch include:

- The ability to stitch together data from multiple data sources with various data formats.
- The ability to process large or small datasets. Scale cluster size based on dynamic requirements (for example, serverless capability with AWS Glue, Amazon EMR Serverless, and AWS Batch with AWS Fargate).
- Bound data processing time and resources by a business service-level agreement (SLA) for a cost-efficient workload.

Client or server

Depending on the architecture, you might choose to ingest game events from a game client (such as a mobile device), from a game server, or both. Both methods are popular, and often used together serving different use cases. Events such as users' interactions with the game can be streamed from a mobile device, and events such as battle results can be streamed from the server.

When streaming from a mobile device, you can use a ready software development kit (SDK) such as Amazon Pinpoint, or develop your own, that can be re-used between games. Such an SDK would have three main responsibilities:

- Batching events for performance, and providing temporary on-device storage and retries so events are not lost when the device is not connected.
- Ingesting to a stream in cloud or on-premises, including authentication.
- Automatically populating demographics data (operating system family, device model) and automatically collecting basic events (such as session start, session stop, and so on).

Another popular option is to stream from a client is using a third-party SDK that is not connected to a stream, but to a backend provided by a third-party (such as Amplitude or Google Firebase), and export events in batches from their backend. Advantages of such approach include ready-to-use dashboards in the backend service, and easy setup of the SDK and backend. Disadvantages include losing streaming and real-time capabilities.

REST API or ingesting directly to a stream

If you choose to build a custom backend for streaming events, you can integrate your game directly with a streaming storage such as Amazon Kinesis or Apache Kafka using a client SDK, or implement a generic REST API in front of the stream.

Benefits of the REST API include:

- Decoupling of particular streaming storage technology from an interface. You don't need to integrate your client or server with a stream and you don't need AWS SDK on the client for Amazon Kinesis, which might be important for mobile games or platforms such as game consoles.
- Ease of support for existing games that don't support direct streaming - you can build an API for them to mimic a legacy data ingestion solution.
- More authentication and authorization options.

However, such API requires extra development effort and can be more costly than ingesting directly. Often games developers use a hybrid approach - they ingest directly to a stream (Amazon Kinesis Data Streams or Amazon Kinesis Data Firehose) from newer games, and implement a REST API layer for legacy games.

Other sources

Sometimes you might need data from other sources, such as operational databases. You can query them directly through services such as Amazon Athena Federated Queries, or extract data and store it in your data lake using the AWS Database Migration Service (AWS DMS). AWS DMS can be configured many ways. One is to offload data from your databases into your data lake on Amazon S3. This can be done in a number of ways that include full load, full load + change data capture (CDC), and CDC only. Refer to the AWS Database Migration Service Documentation for further details.

Data transformation

Event format and changing schema

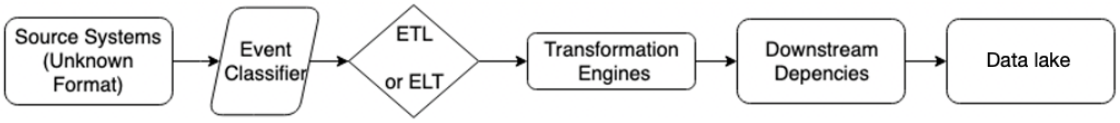
Data at rest can be tuned to one format; however, it is not easy for all source systems to match that. Events generated at the source systems can vary depending on the use case and underlying technology. It is critical to build a data pipeline that can evolve and match with such a high degree of variance. For example, a product analytics workload will typically use a finite number of fields such as userid, timestamp, and productid, whereas a game event workload has fields that are unique to the game, scenario, and device type.

There are two approaches to this high-level scenario:

- AWS has source systems, which are your upstream dependencies, that generate and collect the data. The source systems then use an ETL or extract, load and transform (ELT) mechanism to modify the data. There are transformation engines that are built for large-scale data processing to handle batch and/or streaming data. We then have the downstream dependencies that consume all or part of the data, which eventually makes its way to the data lake.

Data flow (with defined data format) from the source to the data lake

- This is a similar approach as the first, with the key difference being a scenario for an unknown format. For this, you can use a data classifier that can read the data from the data store for known formats, and a custom-classifier to read the data when a new format emerges.



Data flow (with undefined/unknown data format) from the source to the data lake

Transform/analytics engine

Whether you are generating a few gigabytes or are collecting and storing multiple petabytes of user and game data every month, you want to have a few questions answered ahead of time to transform data efficiently and avoid a data swamp. Here are some starter questions:

- How much data is coming in daily and monthly, and what does that growth look like over time?
- What is the format and schema of the data?
- Average file size?
- How many sources?

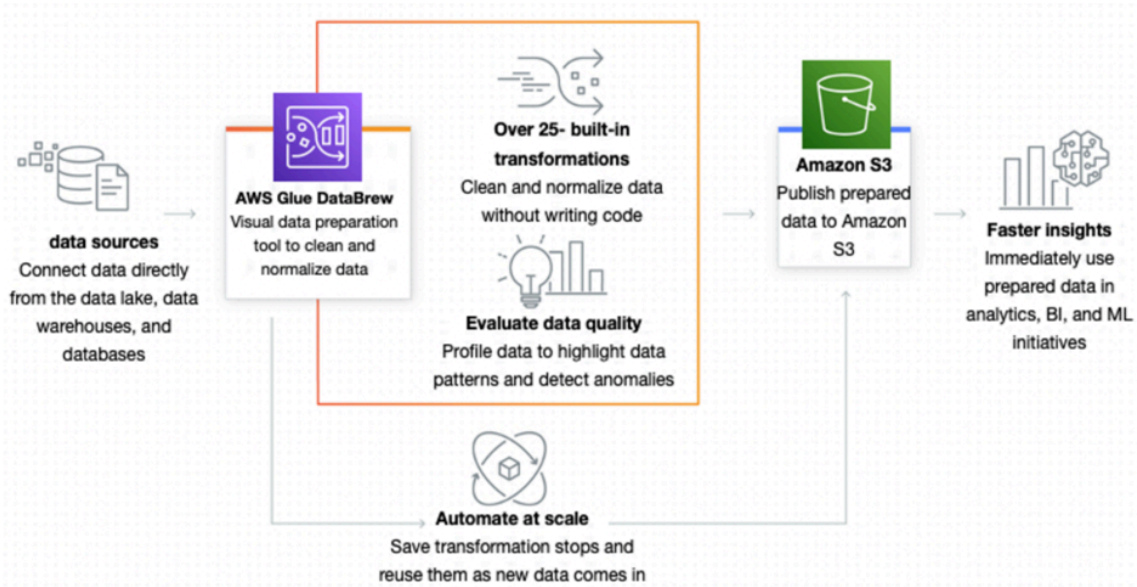
- Who will the stewards of the data be?
- What type of transformations need to be done?
- Who are the consumers?
- How will the data be secured, governed and audited?

This is not a comprehensive list, but should start to give you an idea as to considerations and necessary decisions to make in order to run an efficient and scalable data lake. When talking about extracting, transforming, and loading data, there is not one tool that is the best for every use case. Instead, AWS offers many tools and options based on challenging use cases that customers have asked AWS to help with.

- AWS Glue is a serverless data integration service that makes it easy to discover, prepare, and combine data for analytics, ML, and application development. With AWS Glue, you can do data discovery, transformation, replication, and preparation with many of the different tools AWS Glue offers.
- Streaming ETL jobs - With AWS Glue streaming ETL, you can consume data coming in from Amazon Kinesis Data Streams or Apache Kafka. This is useful when you need to transform and process your data in real-time, and load into downstream services such as databases, or a data warehouse such as Amazon Redshift, which can be used to power a real-time dashboard for user behavior, fraudulent activity, game metrics, and more.
- Batch ETL jobs - You may not always need to transform data in real-time as you would with AWS Glue streaming. Instead, you can implement a batching strategy for your ETL jobs that can be scheduled to run as frequently or infrequently as needed. You can run batch ETL jobs for complex types of aggregations and joins of datasets. Some popular use cases for using batch jobs include:
  - Migrating table data from transactional databases to Amazon S3 for analysis of other downstream services such as Amazon Athena. You can then visualize this data with Amazon QuickSight by using Amazon Athena as the query engine.
  - Loading data from Amazon S3 into Amazon Redshift, where you can also use Amazon Redshift as the query engine for your BI tools.
  - Pre-aggregating and joining data in order to prepare a dataset to be consumed by Amazon SageMaker, the fully managed ML service.

- AWS Glue Elastic Views makes it easy to build materialized views that combine and replicate data across multiple data stores without you having to write custom code. With AWS Glue Elastic Views, you can use familiar Structured Query Language (SQL) to quickly create a virtual table—a materialized view—from multiple different source data stores.

  Replicating and keeping data synced across different data stores is much easier with Elastic Views, as you only need to write an SQL query to start creating the materialized views in your target data stores. You no longer need to spend time developing AWS Lambda functions to read off of streams and then write the records to the target data store, or run AWS Glue jobs every five minutes or so just to keep data fresh.
- AWS Glue DataBrew is a visual data preparation tool that makes it easy for data analysts and data scientists to clean and normalize data to prepare it for analytics and ML. You can choose from over 250 pre-built transformations to automate data preparation tasks, all without the need to write any code. AWS Glue DataBrew is a great option if you want to load your dataset into a visual editor, and then transform your data step-by-step. These steps are saved into a recipe, which is reusable, so you can consistently run and schedule DataBrew jobs with the recipes you create.



How AWS Glue DataBrew helps clean and normalize data

- Amazon EMR (previously called Amazon Elastic MapReduce) is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data. Using these frameworks and related open-source projects, you can process data for analytics purposes and BI workloads. Amazon EMR also lets you transform and move large amounts of data into and out of other AWS data stores and databases, such as Amazon S3 and Amazon DynamoDB. You would want to run an EMR cluster for your ETL operations if your use cases revolve around ETL jobs running 24/7, or you need to manage other tools such as Apache Hadoop, Apache HBase, Presto, Hive, and so on.
- Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run. You can query data stores other than in S3, such as RDS databases, on premises databases, Amazon DynamoDB, ElastiCache for Redis, Amazon Timestream, and more with Athena federated queries. With this functionality, you can combine and move data from many data stores using Athena. Athena is a great tool for ad-hoc data exploration, and benefits most when your data lake is built according to user query patterns.

  A common pattern and good place to start if you aren't sure about what the query patterns are yet is to partition your data as *year*, *month*, and *day*. By doing this, you can use Athena to filter only the partitions you need to access, which will result in a reduction of cost and faster results per query. Athena can also be used as a lightweight ETL tool if you need to process a smaller subset of data through the use of the Create Table as Select (CTAS), where you can format, compress, and partition raw data into an optimized format to be consumed by downstream engines.
- Amazon Redshift is the fastest and most widely used cloud data warehouse. Amazon Redshift is integrated with your data lake and offers up to 3x better price performance than any other data warehouse. Amazon Redshift can help power your data lake when you have longer running, complex queries that need to aggregate large amounts of data. You can move data in and out of Amazon Redshift using integrated tools such as AWS Glue, Amazon EMR, AWS Database Migration Service (AWS DMS), Amazon Kinesis Data Firehose, and more.

Data cataloging

Data catalogs have become a core component and central technology for modern data management and governance. They are essential for data sharing and self-service analytics enablement, increasing the business value of data and analytics. According to the Gartner research report:

“Demand for data catalogs is soaring as organizations continue to struggle with finding, inventorying and analyzing vastly distributed and diverse data assets. Data and analytics leaders must investigate and adopt ML-augmented data catalogs as part of their overall data management solutions strategy."

Successful data catalog implementations empower organizations to continuously improve the speed and quality of data analysis, and better achieve data democratization and productive use of data across the organization. Therefore, choosing the right data catalog technology to implement is an important decision to make while designing a data lake on AWS for games.

A *data catalog* is a collection of metadata, combined with data discovery and management tools, and it provides an inventory of data assets across all your data sources. Data catalog helps data consumers within an organization to discover, understand, and consume data more productively. It helps organizations break down barriers to data lake adoption. A properly maintained data catalog empowers data analysts and users to work in self-service mode to discover trustworthy data quickly, evaluate and make informed decisions for which datasets to use, and perform data preparation and analysis efficiently and with confidence.

Given multiple frameworks, tools and technologies can be employed within a data lake environment for data ingestion, transformation, visualization, and access control. The most efficient way to accomplish this is to maintain a central data catalog and use it across various frameworks such as AWS Glue, Amazon EMR, Amazon Athena, Apache Hadoop, Apache Spark, Hive, Impala, and Presto on AWS. This makes the job of ensuring metadata integrity and applying data governance policies relatively easier.

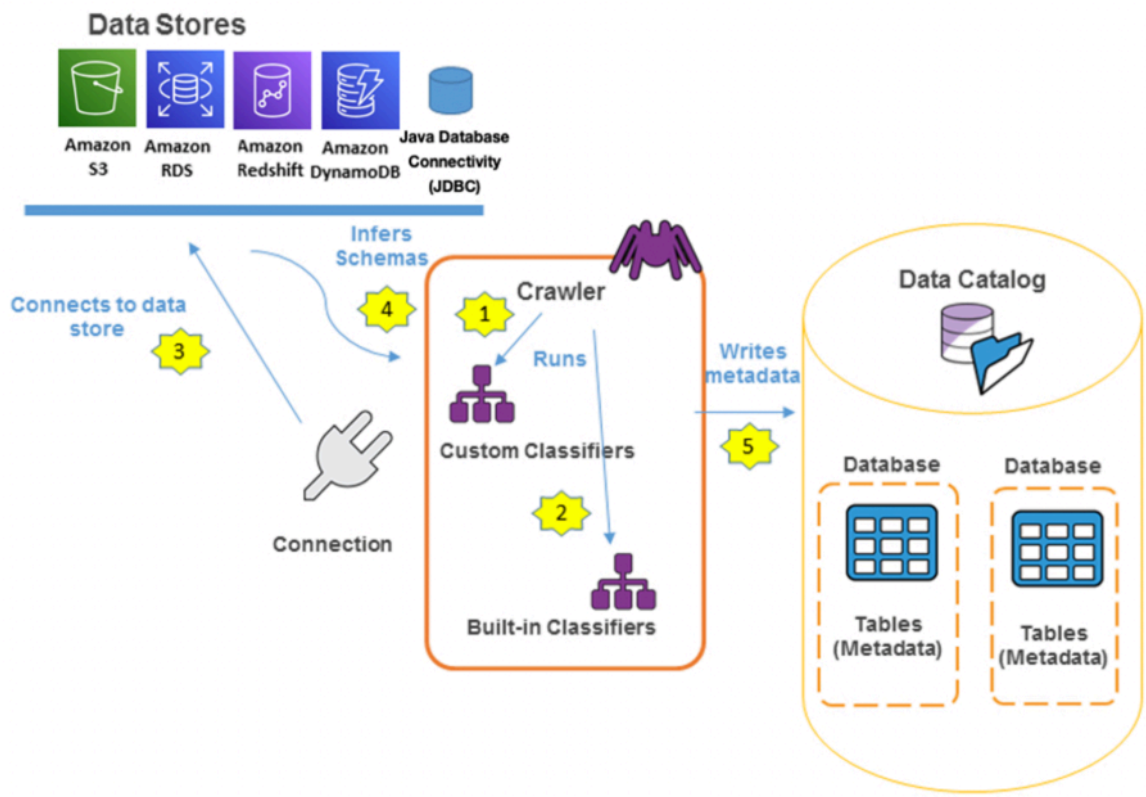
Use AWS Glue Data Catalog as a metadata store for your data lake

The AWS Glue Data Catalog is a fully-managed persistent metadata store allowing you to store, annotate, and share metadata. It provides a unified metadata repository across a variety of data sources and data formats, integrating with Amazon EMR as well as Amazon RDS, Amazon Redshift and Redshift Spectrum, Amazon Athena, AWS Lake Formation, and any application compatible with the Apache Hive metastore.

With AWS Glue Data Catalog, you can store and find metadata, keep track of data in data silos, and use that metadata to query and transform the data. AWS Glue Data Catalog also provides comprehensive audit and governance capabilities with schema change tracking and data access controls, allowing you to audit changes to data schemas. This helps ensure that data is not inappropriately modified or inadvertently shared.

AWS Glue Data Catalog can be extended to meet many of your data cataloging requirements and needs. Sources for AWS Glue Data Catalog tables can include Amazon S3, Amazon Kinesis, Amazon DocumentDB, Amazon DynamoDB, Amazon Redshift, MongoDB, Apache Kafka, Java Database Connectivity (JDBC), and so on. Custom database and table descriptions and table properties can be added to the catalog either manually or through automation. For example, you can add the data owner, data description, and data sensitivity to AWS Glue tables.

The AWS Glue Data Catalog is an index to the location, schema, and runtime metrics of your data. You can use the information in the Data Catalog to create and monitor your ETL jobs. Each AWS account can have one AWS Glue Data Catalog per AWS Region. Information in the Data Catalog is stored as metadata tables, where each table specifies a single data store. Typically, you run a crawler to take inventory of the data in your data stores, but there are other ways to add metadata tables into your Data Catalog. For information about how to use the AWS Glue Data Catalog, refer to Populating the AWS Glue Data Catalog.



How AWS Glue crawlers interact with data stores and other elements to populate the Data Catalog

When configuring the AWS Glue crawler to discover data in Amazon S3, you can choose from a full scan, where all objects in a given path are processed every time the crawler runs, or an incremental scan, where only the objects in a newly added folder are processed.

Full scan is useful when changes to the table are non-deterministic and can affect any object or partition. Incremental crawl is useful when new partitions, or folders, are added to the table. For large, frequently changing tables, the incremental crawling mode can be enhanced to reduce the time it takes the crawler to determine which objects changed.

With the support of Amazon S3 Event Notifications as a source for AWS Glue crawlers, game developers can configure Amazon S3 Event Notifications to be sent to an Amazon Simple Queue Service (Amazon SQS) queue, which the crawler uses to identify the newly added or deleted objects. With each run of the crawler, the SQS queue is inspected for new events, if none are found, the crawler stops. If events are found in the queue, the crawler inspects their respective folders and processes the new objects. This new mode reduces the cost and time a crawler needs to update large and frequently changing tables.

Using Amazon EMR, you can configure Spark SQL to use the AWS Glue Data Catalog as its metastore. AWS recommends this configuration when you require a persistent metastore, or a metastore shared by different clusters, services, applications, or AWS accounts. AWS Glue Data Catalog Client for Apache Hive Metastore is an open-source implementation of the Apache Hive Metastore client on Amazon EMR

clusters that uses the AWS Glue Data Catalog as an external Hive Metastore. It serves as a reference implementation for building a Hive Metastore-compatible client that connects to the AWS Glue Data Catalog. It may be ported to other Hive Metastore-compatible platforms such as other Hadoop and Apache Spark distributions. You can migrate from Apache Hive metastore to AWS Glue Data Catalog. For more information, refer to Migration between the Hive Metastore and the AWS Glue Data Catalog on GitHub.

Because AWS Glue Data Catalog is used by many AWS services as their central metadata repository, you might want to query Data Catalog metadata. To do so, you can use SQL queries in Athena. You can use Athena to query AWS Glue catalog metadata such as databases, tables, partitions, and columns. For more information, refer to Querying AWS Glue Data Catalog.

You can use AWS Identity and Access Management (IAM) policies to control access to the data sources managed by the AWS Glue Data Catalog. These policies allow different groups in your enterprise to safely publish data to the wider organization while protecting sensitive information. IAM policies let you clearly and consistently define which users have access to which data, regardless of its location.

Other options for data catalog

If AWS Glue Data Catalog does not satisfy all of your business and technical requirements for data cataloging purpose, there are other enterprise-grade solutions available on the AWS Marketplace such as Informatica, Collibra, Alation, and unifi for your evaluation. Those solutions can also help you create and maintain a business glossary for mapping to the underlying tables and columns. Some of them provide connectors to integrate with native AWS Cloud services such as AWS Glue Data Catalog, S3, Athena, DynamoDB, and Amazon Redshift.

Additionally, there are also many open-source metadata management solutions for improving the productivity of data consumers and accelerate time to insights. For example, Apache Atla, Amundsen, Metacat, Herd, and Databook.

Data lifecycle management

Most game developers choose to use Amazon S3 as its primary storage platform when they design and build a data lake on AWS. Amazon S3 is designed for 99.999999999% (11 nines) of data durability and offers industry-leading scalability, data availability, security, and performance. Amazon S3 also delivers strong read-after-write consistency automatically, at no cost, and without changes to performance or availability. Using Amazon S3 as your data lake reduces over-provisioning, tears down data silos, and provides unlimited scale.

Amazon S3 storage classes are designed for different use cases, and offer the flexibility and automation to manage your costs. They include:

- S3 Standard for general-purpose storage of frequently accessed data
- S3 Intelligent-Tiering for data with unknown or changing access patterns
- S3 Standard-Infrequent Access and S3 One Zone-Infrequent Access for long-lived, but less frequently accessed data
- S3 Glacier and S3 Glacier Deep Archive for long-term archive and digital preservation
- S3 on Outposts for storing S3 data on-premises if you have data residency requirements that can't be met by an existing AWS Region

This wide range of cost-effective storage classes allows game developers to take advantage of the cost savings without sacrificing performance or making changes to their applications. You can use S3 Storage Class Analysis to analyze storage access patterns to help you decide when to transition the right data to the right storage class to lower cost, and configure an S3 Lifecycle policy to complete transitions and expirations.

If your data has changing or unknown access patterns, you can use S3 Intelligent-Tiering to tier objects based on changing access patterns, and automatically deliver cost savings without operational overhead. You pay a small monthly monitoring and auto-tiering fee. There are no lifecycle fees and retrieval fees. S3 Intelligent-Tiering works by monitoring access patterns, and then moving the objects that have not been accessed in 30 consecutive days to the Infrequent Access tier. Once you have activated one or both of the archive access tiers, S3 Intelligent-Tiering will automatically move objects that haven't been accessed for 90 consecutive days to the Archive Access tier, and then after 180 consecutive days of no access to the Deep Archive Access tier.

You can configure Amazon S3 Lifecycle to manage and optimize your S3 storage cost effectively at scale. An S3 Lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. There are two types of actions:

- Transition actions
- Expiration actions

As the number of objects grow, multi-tenant buckets are created, and the size of the workloads increase, it can become a very complex task to create and manage the many needed lifecycle configuration rules. The use of object tagging can help simplify this process by reducing the number of rules that you need to manage. The purpose of object tagging is to categorize storage, and each tag is a key-value pair. You can add tags to new objects when you upload them, or you can add them to existing objects.

S3 Lifecycle rules contain filters for object tags to specify the objects eligible for the specific lifecycle action, whether it is moving the objects to a more cost-effective storage class, or deleting them after a predefined retention period. Each rule can contain one or a set of object tags. The rule then applies to the subset of objects with the specific tag. You can specify a filter based on multiple tags. For more

implementation details, refer to Simplify your data lifecycle by using object tags with Amazon S3 Lifecycle.

Amazon S3 Storage Lens provides a single view of usage and activity across your Amazon S3 storage. It analyzes storage metrics to deliver contextual recommendations to help you optimize storage costs and apply best practices for protecting your data. You can use S3 Storage Lens to generate summary insights and identify potential cost savings opportunities. For example, you can find objects that could be transitioned to a lower-cost storage class and better utilize S3 storage classes; you can identify and reduce incomplete multipart upload bytes; and you can reduce the number of noncurrent versions retained.

Workflow orchestration

ETL operations are the backbone of a data lake. ETL workflows often involve orchestrating and monitoring the execution of many sequential and parallel data processing tasks. As the volume of data grows, game developers find they need to move quickly to process this data to ensure they make faster, well-informed design and business decisions. To process data at scale, game developers need to elastically provision resources to manage the data coming from increasing diverse sources and often end up building complicated data pipelines.

AWS managed orchestration services such as AWS Step Functions and Amazon Managed Workflows for Apache Airflow (MWAA) are managed workflow orchestration services that help simplify ETL workflow management that involves a diverse set of technologies. These services provide the scalability, reliability, and availability needed to successfully manage your data processing workflows

AWS Step Functions

AWS Step Functions is a serverless orchestration service that lets you combine AWS Lambda functions and other AWS services to build to scalable, distributed applications using state machines. Step Functions is based on state machines and tasks. A *state machine* is a workflow. A *task* is a state in a workflow that represents a single unit of work that another AWS service performs. Each step in a workflow is a *state*. With the Step Functions built-in controls, you examine the state of each step in your workflow to make sure that your data processing job runs as expected.

Step Functions scales horizontally and provides fault-tolerant workflows. You can process data faster using parallel transformations or dynamic parallelism, and it lets you easily retry failed transformations, or choose a specific way to handle errors without the need to manage a complex process. Step Functions manages state, checkpoints, and restarts for you to make sure that your workflows run in order. Step Functions can be integrated with a wide variety of AWS services including:

- AWS Lambda
- AWS Fargate
- AWS Batch
- AWS Glue
- Amazon Elastic Container Service (Amazon ECS)
- Amazon Simple Queue Service (Amazon SQS)
- Amazon Simple Notification Service (Amazon SNS)
- Amazon DynamoDB, and more

Step Functions has two workflow types:

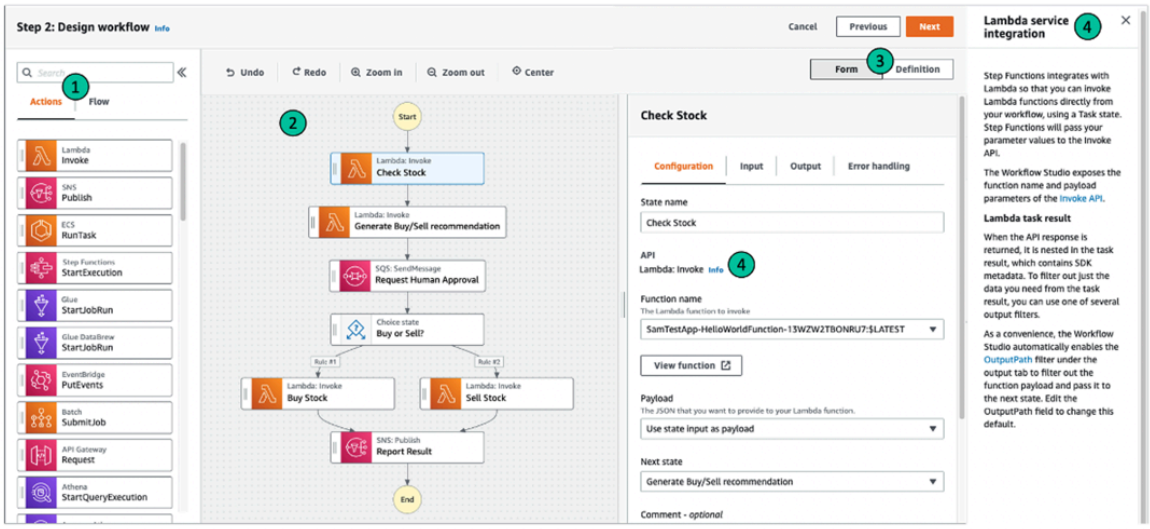
- **Standard Workflows** have exactly-once workflow transformation, and can run for up to one year.
- **Express Workflows** have at-least-once workflow transformation, and can run for up to five minutes.

Standard Workflows are ideal for long-running, auditable workflows, as they show execution history and visual debugging. Express Workflows are ideal for high-event-rate workloads, such as streaming data processing. Your state machine transformations will behave differently, depending on which Type you select. Refer to Standard vs. Express Workflows for details.

Depending on your data processing needs, Step Functions directly integrates with other data processing services provided by AWS, such as AWS Batch for batch processing, Amazon EMR for big data processing,

AWS Glue for data preparation, Athena for data analysis, and AWS Lambda for compute. Run ETL/ELT workflows using Amazon Redshift (Lambda, Amazon Redshift Data API) demonstrates how to use Step Functions and the Amazon Redshift Data API to run an ETL/ELT workflow that loads data into the Amazon Redshift data warehouse. Manage an Amazon EMR Job demonstrates Amazon EMR and AWS Step Functions integration.

Workflow Studio for AWS Step Functions is a low-code visual workflow designer that lets you create serverless workflows by orchestrating AWS services. It makes it easy for game developers to build serverless workflows and empowers game developers to focus on building better gameplay while reducing the time spent writing configuration code for workflow definitions and building data transformations. Use drag-and-drop to create and edit workflows, control how input and output is filtered or transformed for each state, and configure error handling. As you create a workflow, Workflow Studio validates your work and generates code.



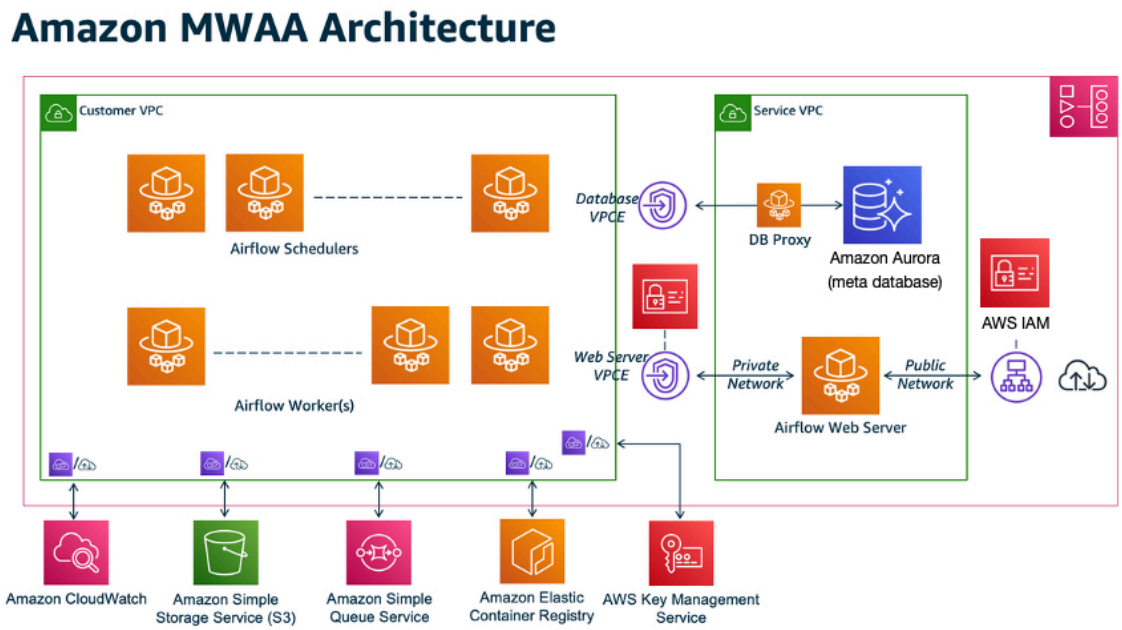
Sample design workflow

Amazon Managed Workflows for Apache Airflow (MWAA)

Apache Airflow is an open-source tool for programmatically authoring, scheduling, and monitoring workflows. Amazon Managed Workflows for Apache Airflow (MWAA) is a managed orchestration service for Apache Airflow that makes it easier to build, operate, and scale end-to-end data pipelines without having to manage the underlying infrastructure for scalability, availability, and security. Amazon MWAA can help reduce operational cost and engineering overhead.

The auto scaling mechanism of MWAA automatically increases the number of Apache Airflow workers in response to queued tasks and disposes of extra workers when there are no more tasks queued or running. You can configure task parallelism, auto scaling, and concurrency settings directly on MWAA console.

Amazon MWAA orchestrates and schedules your workflows by using Directed Acyclic Graphs (DAGs) written in Python. To run DAGs in an Amazon MWAA environment, you copy your files to the Amazon S3, then let Amazon MWAA know where your DAGs and supporting files are located on the Amazon MWAA console. Amazon MWAA takes care of synchronizing the DAGs among workers, schedulers, and the web server.



All of the components contained in the MWAA section appear as a single Amazon MWAA environment in your account

Amazon MWAA supports open-source integrations with Amazon Athena, AWS Batch, Amazon CloudWatch, Amazon DynamoDB, AWS DataSync, Amazon EMR, AWS Fargate, Amazon EKS, Amazon Kinesis Data Firehose, AWS Glue, AWS Lambda, Amazon Redshift, Amazon SQS, Amazon SNS, Amazon SageMaker, and Amazon S3, as well as hundreds of built-in and community-created operators and sensors and third-party tools such as Apache Hadoop, Presto, Hive, and Spark to perform data processing tasks.

Code examples are available for faster integration. For example, Using Amazon MWAA with Amazon EMR demonstrates how to enable an integration using Amazon EMR and Amazon MWAA. Creating a custom plugin with Apache Hive and Hadoop walks you through the steps to create a custom plugin using Apache Hive and Hadoop on an Amazon MWAA environment.

Data security and governance

Security within data lakes is addressed in two areas.

- Security of data at rest
- Security of data in transit

Data at rest — With AWS, you have many options that can potentially meet your encryption needs. Within the data lake framework, data primarily resides in the primary datastore (for example, Amazon S3) and in certain use-cases in secondary datastore (for example, S3 in a non-primary Region, or Outposts). Refer to the latest AWS documentation for the latest details:

- For S3, Protecting data using server-side encryption - Amazon Simple Storage Service
- For Outposts, Data protection in AWS Outposts - AWS Outposts
- For importance of encryption, refer to The importance of encryption and how AWS can help on the AWS Security blog

Data in transit — To protect data in transit, AWS encourages customers to use a multi-level approach. All network traffic between AWS data centers is transparently encrypted at the physical layer. All traffic within a VPC and between peered VPCs across Regions is transparently encrypted at the network layer when using supported Amazon EC2 instance types. At the application layer, customers have a choice about whether and how to use encryption using a protocol like Transport Layer Security (TLS). All AWS service endpoints support TLS to create a secure HTTPS connection to make API requests. Refer to our latest documentation for the latest details:

- Logical separation for Data Encryption
- Implement secure key and certificate management
- Alerting for unintended data access
- Authenticate all network traffic via EC2 or EKS using TLS
- Security Pillar - AWS Well-Architected Framework

How to regulate your data (GDPR, CCPA, and COPPA)

Regulations such as GDPR, CCPA, and COPPA require that operators must implement the *right to forget*. This means there must be a technical ability to delete Personally identifiable information (PII) of specific users by request within a specified number of days. For analytics data stored in multi-terabyte data lakes in read-optimized format such as Apache Parquet, this is often a challenge. There are two reasons for this:

First, you need to scan through all data in the data lake to identify partitions that contain records with the user ID you need.

Second, you cannot delete a single record from a Parquet file or update a single Parquet file within a partition - you'll have to re-calculate and re-write the whole partition.

Both operations are very time- and resource-consuming on a big data lake, because they involve many metadata operations (such as S3 LIST), and they must scan through all data.

There are two main approaches to solve this: avoid storing PII in the data lake, or implement an additional metadata layer to reduce the number of operations and volume of data to scan to search for a specific user ID, and delete the corresponding data.

The safest way to comply with regulatory frameworks is to not store PII in the data lake. Depending on what kind of analytics you require, this may or may not be possible. For example, you can get rid of any forms of user identifiers in your data lake, but this will make any kind of per-user aggregations, often required by analytics, impossible.

A common approach is to replace real user identifiers with surrogate ones, store these mappings in a separate table outside of the data lake, and avoid storing any other kind of user information besides these internal IDs. This way, you can still analyze user behavior without knowing who the user is. The data lake itself doesn't store any PII in this case. It's enough to delete a mapping for a specific user from an external table to make data records in the data lake not relatable to users. Consult your legal department to find out if this is enough to comply with regulatory requirements in your case.

There are also a number of techniques to impersonate data, such as masking, hashing, blurring, modifying numbers by a random percent, and so on. They can be used to impersonate other personally identifiable data, if you choose to store it in a data lake. There are some third-party products such as Dataguise or Collibra (available on the AWS Marketplace) to automate this.

Another approach is to optimize user ID lookup and delete performance, usually by implementing an additional metadata layer. This might be required if your legal department tells you impersonation or surrogate IDs are not enough to comply. You can build your own solution to maintain indexes by user_id, or use an open-data formats such as Apache Hudi in the data lake.

You can build your own solution for indexing. For example, you can maintain an index of files in the data lake by user ID, and update it when new files are added with a Lambda function. Then a delete job can re-write only those files directly. An example of such a solution is described in the How to delete user data in an AWS data lake blog entry.

For an alternative approach which provides additional benefits such as versioning, transactions, and improved performance, use one of the data formats such as Apache Hudi, Delta Lake, or Apache Iceberg for the data lake. Such solutions maintain additional metadata on top of open file formats such as Parquet or AVRO to enable upsert/delete. When using these solutions, you can delete specific user data by ID with an SQL query or a Spark job. This will still be expensive in terms of reads, but it will be much faster on write. Generally, such delete jobs can work in reasonable time, even in big data lakes.

The downside is that you'll need to use analytical engines and ETL tools that support such formats. The degree of support by AWS services varies for different formats. Hudi is probably a good place to start. Amazon Redshift, Athena, AWS Glue, and EMR all support it. Specific feature support can also vary by service. Be sure to check the specific service's documentation.

Data discovery

Data discovery refers to the overall recognition of patterns in data stored on AWS. Services such as Amazon Macie provide a managed data security and data privacy service that uses machine learning and pattern matching to discover and protect your sensitive data in AWS. Use cases for data discovery include:

- Extract value from stored data within the business SLA
- Prevent sensitive data types to be ingested into a data lake

For an example of discovering sensitive data, refer to Use Macie to discover sensitive data as part of automated data pipelines.

For information on Amazon RDS as a data store, refer to Enabling data classification for Amazon RDS database with Macie.

For information on detecting sensitive data in DynamoDB, refer to Detecting sensitive data in DynamoDB with Macie.

Data governance

Data governance refers to the overall management of data assets in terms of the availability, quality, usability, lineage and security of the data in an organization. Data governance largely depends upon business policies and usually covers the following areas:

- Data ownership and accountability
  - Have proper structure in place to determine permissions and roles for data in raw, curated, and processed formats.
  - Ability to monitor all API calls made within AWS, which is critical to audit any suspicious actions in your AWS account.
- Enforcing policies and rules
  - Developing policies based on the organizational structure regarding what the data will be used for, and who can access it.
  - Having automation in place for data sharing, data quality, alerts, and faster access to data for users with proper permissions.
- Technical processes, tools and practices
  - AWS Lake Formation is an integrated data lake service that makes it easy for you to ingest, clean, catalog, transform, and secure your data and make it available for analysis and machine learning. Lake Formation gives you a central console where you can discover data sources, set up transformation jobs to move data to an Amazon S3 data lake, remove duplicates and match records, catalog data for access by analytic tools, configure data access and security policies, and audit and control access from AWS analytic and ML services. Lake Formation offers features that make it easy to govern and administer data in your data lake:
    - Governed tables allow your data lake to support atomicity, consistency, isolation, durability (ACID) transactions, where you can add and delete S3 objects reliably, while protecting the integrity of the data catalog.
    - Row level security provides security at the table, column, and row level, and is managed directly from Lake Formation. You can apply row level security to S3-based tables such as AWS Glue Data Catalog, Amazon Redshift data shares, Governed, Apache Hive, Apache Hudi, and more.
    - AWS Glue Data Catalog is a persistent metadata store and contains information about your data such as format, structure, size, data types, data fields, row count, and more. You can register databases and tables within the AWS Glue Data Catalog, which is integrated with many native AWS services.
    - AWS CloudTrail for auditing and monitoring API calls made within your AWS accounts and is built directly into Lake Formation.

Lake Formation manages all of the tasks shown here, and is integrated with the AWS data stores and services

- AWS Glue DataBrew helps by giving you a visual interface to prepare, profile data, and track lineage in a repeatable, automated fashion. DataBrew can take your raw data that has come in and run the automation recipe you have set for it, like removing columns, formatting some of the values or headers, and then write the processed data to Amazon S3 for downstream consumption of other analytical or machine learning services. AWS Glue DataBrew can also be used for data quality automation and alerts.
- Open-source tools such as Apache Atlas, Deequ, and Apache Ranger are also popular among customers who are comfortable managing and setting up the infrastructure and required configurations, as these tools do not natively integrate with other AWS services like Athena or Glue.

Unlike data management, where the primary concern is usage of data in making sound business decisions, data governance is concerned with how disciplined you are in managing and using the data across the organization. Without data governance and proper mechanisms to maintain it, the data lake risks becoming a data swamp, and being a collection of disconnected data silos.

Data visualization

Amazon QuickSight is a serverless BI tool that allows users to create visualizations and dashboards from your data. QuickSight can connect to many different data stores like Amazon S3, on-premises databases, Salesforce, Amazon RDS, Amazon Redshift, and more. With QuickSight, you can create dashboards for many gaming-specific use cases. QuickSight has ML insights built directly into the platform, where it can potentially detect fraud or an anomaly in player activity. QuickSight can also be used in conjunction with AWS CloudTrail, to help monitor, visualize, and audit your data lake operations.

For more information, refer to Run usage analytics on Amazon QuickSight using AWS CloudTrail and Ingest and visualize streaming data for games.

Monitoring

At a high-level, there are two types of monitoring within your data lake:

- Monitoring your resources
- Monitoring your data quality in those resources

There are a couple of options in this space:

- **Monitoring your AWS resources** — Amazon CloudWatch collects and tracks metrics, monitors logs, sets thresholds, and triggers alarms per your environment. CloudWatch can monitor AWS resources such as Amazon EC2 instances, Amazon S3, Amazon EMR, Amazon Redshift, Amazon DynamoDB, and Amazon Relational Database Service (RDS) database instances, as well as custom metrics generated by other data lake applications and services. CloudWatch provides system-wide visibility into resource utilization, application performance, and operational health. You can use these insights to proactively react to issues and keep your data lake applications and workflows running smoothly.

  For more information, refer to Monitoring and optimizing the data lake environment and the Data Lake on AWS implementation guide.

  AWS Partners such as Datadog, New Relic, and Splunk offer similar capabilities as Amazon CloudWatch, and are available as options to our customers.
- **Monitoring your data quality** — Everyday customers accumulate petabytes of data on AWS. With this constant flow of data, it is a struggle for these customers to maintain the quality of data they manage. Fortunately, AWS has some options in this space. For example, on the data layer, you can use Deequ with AWS Glue, AWS Amplify, and DynamoDB to create a data quality and analysis framework. On the ML side, you can use SageMaker to automatically monitor ML models in a production environment, and get notifications when data quality issues arise.

Cost optimization

Cost is derivative from the usage of the underlying services that contribute to the data lake. There is no additional charge for using features in Lake Formation; however, standard usage rates apply when using services such as AWS Glue, Amazon S3, Amazon EMR, Amazon Athena, Amazon Redshift, Amazon Kinesis, and so on.

A few things to consider regarding cost when building a data lake on AWS include:

- **Cost drivers** — There are three main drivers of cost with AWS: Compute, storage and outbound data transfer. These vary depending on AWS product, pricing model and the Region (Data location).
- **Cost optimization strategies** — Based on the three main drivers of costs, and we can leverage a plethora of cost-efficient solutions on AWS to help meet the right use-case and budget. For example:
  - **Compute: On-Demand Instances vs. Savings Plans vs. Spot Instances vs. Reservations** — Customers can pick and choose the right model, based on workload type and costs. For services such as Amazon Redshift and EMR, it is important to "right-size" your solution to best fit your business needs, and then use Reserved Instances to further reduce costs. Refer to Amazon EC2 pricing.
  - Most AWS analytics services are serverless, which means you pay only for the compute you use and nothing more. When optimizing for cost with serverless services, there are some key considerations:
    - Avoiding re-processing data when possible, and only process what you need. As an example, this can be done by using AWS Glue Job bookmarks when processing data, or targeting only the data you need within a partition when using Athena.
    - Assigning the correct number of resources for data processing.
    - Correct partitioning, compression, storage, and lifecycle policy strategies when storing data in Amazon S3.

- **Storage:**
  - **S3 Tiers vs. EBS vs. EFS vs. Amazon FSx vs. Snow Family**: — You can choose from a variety of storage tools available on AWS. Intelligent options are also available if you prefer to let AWS decide the right S3 tier, depending on your unique usage.
  - **Outbound data transfer** — Customers do not pay for inbound data transfer across all services in all Regions. Data transfer from AWS to the internet is charged per service, with rates specific to the originating Region. Refer to the pricing pages for each service to get more detailed pricing information. Best practices include:
    - Avoid routing traffic over the internet when connecting to AWS services. Use VPC endpoints if available.
    - Consider AWS Direct Connect for connecting to on-premises networks.
    - Avoid cross-Region costs unless your business case requires it.
    - Create a data transfer cost analysis dashboard to be strategic in managing your capacity and usage.

- **Cost calculation tools** — For a more detailed calculation of monthly costs on AWS, refer to the AWS Pricing Calculator. For existing infrastructure on your AWS account, refer to AWS Cost Management Console in your AWS Management Console for a detailed analysis on your usage (sign-in required). Subscribe to your Trusted Advisor reports in your AWS account for cost optimization checks that can save you money (sign-in required). For example, you might have unused resources in your AWS account that can be deleted.

Further reading

For additional information, refer to:

- *Building Big Data Storage Solutions (Data Lakes) for Maximum Flexibility* (AWS whitepaper)
- Deploy data lake ETL jobs using CDK Pipelines (blog post)
- Orchestrate multiple ETL jobs using AWS Step Functions and AWS Lambda (samples on GitHub)
- Orchestrate Apache Spark applications using AWS Step Functions and Apache Livy (blog post)
- Building complex workflows with Amazon MWAA, AWS Step Functions, AWS Glue, and Amazon EMR (blog post)
- Field Notes: How to Build an AWS Glue Workflow using the AWS Cloud Development Kit (AWS CDK) (blog post)
- Setting up automated data quality workflows and alerts using AWS Glue DataBrew and AWS Lambda (blog post)
- Discovering metadata with AWS Lake Formation: Part 1 (blog post)
- Discovering metadata with AWS Lake Formation: Part 2 (blog post)
- Effective data lakes using AWS Lake Formation, Part 1: Getting started with governed tables (blog post)
- AWS Data and Analytics Competency Partners

Contributors

Contributors to this document include:

- Karthik Kumar Odapally, Sr. Solutions Architect for Games, AWS
- Jackie Jiang, Sr. Solutions Architect for Games, AWS
- Eugene Krasikov, Sr. Solutions Architect for Games, AWS
- Michael Hamilton, Analytics Specialist SA, AWS

Document revisions

Date	Description
May 11, 2022	First publication

Notices

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

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

AWS glossary

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