AWS Prescriptive Guidance

Designing a data lake for growth and scale on the AWS Cloud
Table of Contents

Introduction .................................................................................................................. 1
  Targeted business outcomes .................................................................................... 1
Common scaling challenges ....................................................................................... 3
  Initial data lake deployment .................................................................................... 3
  Data consumers increase ......................................................................................... 4
  Data producers increase .......................................................................................... 4
Reference architecture ............................................................................................... 6
Reference architecture components .......................................................................... 7
  Data producers ......................................................................................................... 7
  Centralized catalog .................................................................................................. 9
    AWS Lake Formation .............................................................................................. 9
    AWS Glue .............................................................................................................. 10
  Data consumers ....................................................................................................... 10
Onboarding and granting access ................................................................................ 13
  Onboarding data producers ..................................................................................... 13
  Onboarding data consumers ................................................................................... 13
Grant Select access in a data consumer account ......................................................... 14
FAQ .............................................................................................................................. 16
  Is this data lake reference architecture more applicable to enterprise organizations? .................................................. 16
  Can I still use this reference architecture if my organization only has one data producer? ........................................... 16
  My data lake directly connects one data producer with multiple data consumers. Is this guide's data lake reference architecture still relevant? .............................................................................. 16
  Should my organization follow the onboarding and access granting workflow without making changes to it? .............................................................................................................. 16
Resources .................................................................................................................. 18
Document history ...................................................................................................... 19
Glossary ..................................................................................................................... 20
Modernization terms ................................................................................................. 20
Designing a data lake for growth and scale on the AWS Cloud

Wei Shao, Amazon Web Services (AWS)

October 2021 (document history (p. 19))

Organizations are increasingly designing and building data lakes on the Amazon Web Services (AWS) Cloud as part of their data modernization strategy. Data lakes can be repositories that store your structured and unstructured data at any scale and make it available to a wide range of internal and external users.

However, it can take significant time and effort to ingest data into data lakes that serve your growing business use cases. To help reduce costs and maximize the value that is generated from the data, many organizations plan a one-time ingestion of data into their data lake and then consume this data multiple times. Designing a data lake architecture that scales with data production, sharing, and consumption is critical for delivering value to business stakeholders when your data lake grows.

A scalable data lake architecture provides your organization with a solid foundation to gain value from your data lake while bringing more data into it. By continuously gaining data insights without being slowed down or interrupted because of scalability constraints, a scalable data lake also helps your organization remain competitive.

Typically, a data lake has data producers and data consumers. Data producers create data assets by collecting, processing, and storing data from their data domain. These collective data assets form the content of your data lake. Data producers can choose to selectively share their data assets with the data lake's data consumers.

Data consumers need the data from the data producers to fulfill their business use cases and can also occasionally combine this data with their own data. Data producers and data consumers are typically, but not always, part of your organization. Importantly, they can both be data producers or data consumers at the same time.

A scalable data lake architecture helps you to achieve the following outcomes:

- Onboard data producers at scale without requiring them to maintain the entire data sharing process. This helps data producers onboard their data into the data lake and focus on collecting, processing, and storing data from their data domain.
- Enable data consumers to access data from multiple data producers without increasing your overall costs and management overhead.

This guide describes the common scaling challenges that can occur when organizations grow their data lake, provides a data lake reference architecture, and outlines approaches for onboarding and granting access to data producers and data consumers. The data lake reference architecture in this guide leverages the different features and capabilities provided by AWS Lake Formation. The guide is intended for teams that are responsible for designing data lakes on the AWS Cloud, including enterprise data architects, data platform architects, designers, or data domain leads.

Targeted business outcomes

You should expect the following three outcomes after designing a data lake for growth and scale on the AWS Cloud:
AWS Prescriptive Guidance Designing a data lake for growth and scale on the AWS Cloud
Targeted business outcomes

- Lower data sharing and data consumption overheads across multiple lines of business in your organization.
- A secure and consistent approach that helps your organization include external data producers and share data with them in your data lake.
- Continuously gain data insights without being slowed down or interrupted because of scalability constraints.
Common scaling challenges

A data lake goes through several stages when its data grows after the initial deployment. If you didn’t use scalable architecture to design your data lake, your organization might encounter challenges and can be disadvantaged by the data lake’s growth.

The following sections explain how a typical data lake’s growth can cause scaling challenges.

Initial data lake deployment

The following diagram shows a data lake’s architecture after its initial deployment by Line of business A.

The diagram shows the following components:

- The data producer account collects and processes data, stores the processed data, and prepares it for consumption.
- Data in the data producer account is stored in Amazon Simple Storage Service (Amazon S3) buckets, which can have multiple data layers.
- You can use AWS services for data processing (for example, AWS Glue and Amazon EMR).
- The data producer not only produces and stores data in the data lake but then also needs to decide what data to share with a data consumer and how to share it. AWS Lake Formation manages the data lake in the data producer account, in addition to managing cross-account data sharing from the data producer to the data consumer.
- The data consumer account consumes shared data from the data producer account for specific business use cases.
Data consumers increase

The following diagram shows that more data is brought into the data lake when Line of business A's data grows. The data lake then attracts more data consumers to leverage and gain value from the data.

The diagram shows how an organization generates nearly continuous value from an existing data asset and that this attracts more data consumers. However, when data consumers increase, the data producer only has the following two options to accommodate this growth:

- Manually manage data sharing and access by individual data consumers, which is not a scalable approach.
- Develop an automated or semi-automated process for data sharing and managing data access. Although this could be a scalable option, it requires significant time and effort to design and build because internal and external data consumers have different security control requirements. In the future, additional time and effort would also be required for any solution improvements.

Data producers increase

The following diagram shows the data lake architecture when multiple lines of business join as data producers.
The data lake's architecture becomes increasingly complicated, even with only three data producers and three data consumers.

Each data producer needs to handle data sharing and data access management for multiple data consumers. It is unrealistic to expect all data producers to develop an automated or semi-automated process for data sharing and data access management. Some data producers might choose to not share their data and therefore avoid unaffordable management overhead. Similarly, each data consumer needs to interact with multiple data producers to understand their different data consumption processes. This means that individual data consumers face increasing management overhead for handling different data-sharing patterns.

In many organizations, this data lake causes bottlenecks and cannot grow or scale. This might mean that your organization must redesign and rebuild its data lake to remove the bottleneck, which can cost significant time, resources, and money.
Reference architecture

The following diagram shows this guide's reference architecture for growing and scaling a data lake on the AWS Cloud.

The diagram shows the following components:

1. A data producer layer in different AWS accounts.
2. A data consumer layer in different AWS accounts.
3. A centralized catalog in an AWS account.
4. Although each line of business only has one data producer and one data consumer, the guide's reference architecture supports multiple data producers and data consumers for each line of business. It’s typical to onboard one data producer with one or multiple data consumers that include both data-serving and application types. For more information about this, see the Reference architecture components (p. 7) section of this guide.
5. The centralized catalog is the interface used by data producers and data consumers to share and consume data.

The reference architecture's approach makes it possible to standardize data sharing and consumption, and independently scale data producers and data consumers without growing your management overhead. The reference architecture also enables data production and distribution across different data producers. Any data producer can be part of the data lake, share their data, and contribute to the overall value provided by the data lake.

This approach enables your organization to harvest data value throughout your lines of business and external data owners, without causing a bottleneck by constraining data collection and processing in a single pipeline.
Reference architecture components

The following diagram shows the three components of this guide's reference architecture: data producers, data consumers, and the centralized catalog.

The diagram shows the following components:

1. A data producer account.
2. A centralized catalog.
3. Two data consumer accounts, which include data-serving and application types.

**Note**

The diagram shows only one data producer and two data consumers, but this guide's reference architecture supports multiple data producers and data consumers when the data lake grows.

Understanding each component helps your organization to implement reusable patterns and scale your data lake by independently and consistently onboarding new data producers and data consumers.

The following sections discuss the reference architecture's components:

- **Data producers** (p. 7)
- **Centralized catalog** (p. 9)
- **Data consumers** (p. 10)

**Data producers**

A data producer collects, processes, and stores data from their data domain, in addition to monitoring and ensuring the quality of their data assets. The following diagram shows the data producer account as a component of this guide's reference architecture.
Each data producer has a private Data Catalog managed by AWS Lake Formation in their AWS account that is used by their internal data process. Data producers provide the centralized catalog with selective permissions to their data, which means that Lake Formation in the centralized catalog account can access data that the data producer wants to share.

This means that data producers don't directly interact with data consumers. Instead, the data producer account and its data storage location are completely abstracted and hidden from the data consumer. This approach reduces costs by removing unnecessary overhead for data producers that experience an increase in their data consumers.

A change to the data producer's data location doesn't impact the data consumer if the new data location is registered by the centralized catalog. If the data producer wants to stop sharing a particular piece of data, they can remove the centralized catalog's permissions. This prevents data consumers from accessing the data and removes the need to manually revoke access for each data consumer.

By using public and private data catalogs, data producers can choose what to share with data consumers, while independently managing internal data access through a private data catalog.

The following table describes the two AWS services that data producers use to share data with the centralized catalog.
Centralized catalog

The following diagram shows how the centralized catalog connects data producers and data consumers in the data lake.

The centralized catalog stores and manages the shared data catalog for the data producer accounts. The centralized catalog also hosts the shared data's technical metadata (for example, table name and schema) and is the location where data consumers come to access data.

Data consumers can access data from multiple data producers in the centralized catalog and can then mix this data with their own data for further processing. Using a centralized catalog removes the need for data consumers to directly connect with different data producers and reduces operational overhead.

Because the centralized catalog has visibility into data sharing and data consumption by data producers and consumers, it can be an ideal location to apply your centralized data governance functions (for example, access auditing).

The following sections describe how the centralized catalog uses AWS Lake Formation and AWS Glue.

AWS Lake Formation

AWS Lake Formation helps create databases in an AWS Glue Data Catalog that point to the locations of multiple data producers in your data lake. An AWS Identity and Access Management (IAM) role is
AWS Prescriptive Guidance Designing a data lake for growth and scale on the AWS Cloud

AWS Glue

You must create databases in AWS Glue for each data producer in your centralized catalog. Because the centralized catalog uses AWS Glue to host databases from all data producers, you must make sure that the database name is unique across all data producers and that it reflects the data producer and their type of data. For example, you can use the following database naming structure: `<Data_Producer>`-`<Environment>`-`<Data_Group>`

- `<Data_Producer>` – The data producer's name.
- `<Environment>` – The data lake environment, such as dev for a development environment, sit for a system integration test environment, or prod for a production environment.
- `<Data_Group>` – The name of the data group that is used to separate data from a data producer into logical groups. You can use the source system name, ID, or abbreviation as the name. A database description also helps to describe the content and purpose of the database.

You can use an AWS Glue crawler on the data producer's data to maintain its schema in the centralized catalog's database. If data is regularly created on the same frequency by a data producer, you can use a single AWS Glue crawler. In all other cases, you should use multiple AWS Glue crawlers to accommodate different crawling frequencies. Depending on your business use case, the crawler can either be scheduled for a predefined frequency or initiated by events.

You can also maintain table schema in AWS Glue by calling the AWS Glue API to create or update the schema. Although this can provide flexibility, additional effort is required for code development and maintenance. Make sure that you evaluate the use case and business value and then choose the option that meets your requirements and has the least overhead.

Data consumers

Data consumers consume the data from the data producer after the centralized catalog shares it using AWS Lake Formation. The following diagram shows two data consumers in the data lake.
Data consumers

There are two types of data consumer: **application** and **data-serving**. The following table describes these two types.

| Application type | Data consumers run applications in their own AWS accounts. The applications consume the AWS Identity and Access Management (IAM) roles to access the shared data from a data producer and then process it according to their logic.

Typically, this type of data consumer has prescriptive data requirements to fulfill an application's needs. |
|---|---|
| Data-serving type | Data-serving data consumers are typically meant for individuals (for example, data analysts or data scientists) and applications (for example, a business intelligence application) that don't have their own AWS accounts.

Multiple data-serving data consumers can exist in one organization's data lake. For example, different lines of business might choose to set up their own data-serving data consumers to help users consume data from the data lake. These data consumers have their own IAM role principals configured in their AWS account (for example, IAM roles associated with [AWS IAM Identity Center](https://aws.amazon.com/iam/)) that are used by end users in the data consumer account to access shared data through AWS services (for example, [Amazon Athena](https://aws.amazon.com/athena/)).

Typically, this type of data consumer has wide-ranging and continuously increasing data requirements. |

AWS Lake Formation is the most important AWS service used by a data consumer for cross-account data sharing and accessing the centralized catalog. After databases are shared by the centralized catalog,
Data consumers

the shared resources are available in Lake Formation in the data consumer account. Data access can then be granted to local IAM principals in the data consumer account, with permission from the data producer, if required. The shared data can then be used by AWS services integrated with Lake Formation (for example, Amazon Athena and AWS Glue). You can use the following AWS services to access shared data in the data consumer account:

- **Amazon Athena** is an interactive query service that helps directly analyze data in Amazon Simple Storage Service (Amazon S3) using standard SQL. For more information about Athena and Lake Formation, see [How Athena accesses data registered with Lake Formation](#) in the Amazon Athena documentation.

- **Amazon Redshift Spectrum** helps you to efficiently query and retrieve structured and semi-structured data from files in Amazon S3 without having to load the data into Amazon Redshift tables. For more information about Redshift Spectrum and Lake Formation, see [Using Redshift Spectrum with Lake Formation](#) in the Amazon Redshift documentation.

- **AWS Glue** is a fully managed extract, transform, and load (ETL) service that makes it simple and cost-effective to categorize your data, clean it, enrich it, and move it reliably between different data stores and data streams. An AWS Glue ETL job's associated IAM role can access the data lake data managed by Lake Formation if it has the required access permissions.

- **Amazon EMR** helps run big data frameworks (for example, Apache Hadoop and Apache Spark) to process and analyze large amounts of data. For more information about Amazon EMR and Lake Formation, see [Integrate Amazon EMR with Lake Formation](#) in the Amazon EMR documentation.

- **Amazon QuickSight** is a scalable, serverless, embeddable, and machine learning (ML)-powered business intelligence service that you can use to analyze and visualize data from your data lake. For more information about Amazon QuickSight and Lake Formation, see [Authorizing connections through Lake Formation](#) in the Amazon QuickSight documentation.

- **Amazon SageMaker Data Wrangler (Data Wrangler)** reduces the time it takes to aggregate and prepare data for ML. For more information about Data Wrangler and Lake Formation, see [Prepare ML Data with Amazon SageMaker Data Wrangler](#) in the Amazon SageMaker documentation.
Onboarding and granting access

This guide’s data lake reference architecture helps you to independently scale data producers and data consumers, in addition to defining and establishing a consistent process for onboarding and granting access to those data consumers.

The following sections describe the onboarding process for data producers and data consumers and how to grant access in a data consumer account. This guide uses the named resource method between the centralized catalog and data consumers. The process for the LF-TBAC method is similar but slightly different. We recommend that you evaluate and configure these approaches to meet your organization’s data governance practices and policies.

For more information about these two methods, see the Centralized catalog (p. 9) section of this guide.

Onboarding data producers

The following diagram shows how to onboard a new data producer to your data lake.

The diagram shows the following onboarding process:

1. The data producer selectively provides the centralized catalog with access to its data (for example, an Amazon Simple Storage Service (Amazon S3) bucket and AWS KMS key). Access is provided to the centralized catalog’s AWS Identity and Access Management (IAM) principals to register the data producer’s data lake location in AWS Lake Formation and the IAM principals used to maintain the data producer’s catalog.

2. Register the data producer’s data lake location (for example, an S3 bucket) that uses the centralized catalog’s Lake Formation.

3. Create the database, tables, and table schemas for the new data from the data producer in the AWS Glue Data Catalog.

Onboarding data consumers

The following diagram shows how to onboard a new data consumer to your data lake.
The diagram shows the following onboarding process:

1. The data consumer requests approval to view the data producer's data and specifies the data that it needs to access.
2. The data producer’s data steward reviews the request from the data consumer and evaluates whether to:
   - Share some or all tables in the requested databases. We recommend database-level sharing when there are no data security implications of sharing all tables with the data consumer, which helps avoid the management overhead of table-level sharing.
   - Share at the data consumer's organization, OU, or account level.
3. When approved by the data producer, the required Data Catalog resources are shared with the data consumer in the centralized catalog.
4. Resource links can be created in the data consumer's account by using Lake Formation and then point to the shared Data Catalog resources in the centralized catalog.

After the onboarding process is complete, the data consumer's Lake Formation administrator can see the database catalog resource from the centralized catalog and the resource link. At this stage, no one else in the data consumer's account can access the data producer's data.

Grant Select access in a data consumer account

The following diagram shows the process for granting Select access to shared data resources with a local IAM principal in the data consumer account. The local IAM principal can be the IAM role for individual users or an IAM role that is consumed by specific AWS services.

Note
When the data being shared is of low sensitivity, you can delegate access granting to the data consumer itself without requiring approval from the data producer. This is because trust and sharing are already established between them.
The diagram shows the following process:

1. The individual IAM principal in the data consumer account requests `Select` access to the resource link from the IAM principal in the data consumer account.
2. The data producer’s data steward reviews the request from the data consumer and provides approval if all requirements are met.
3. `Select` access is granted and this allows the IAM principal to consume the requested data.
FAQ

This section provides answers to commonly raised questions about designing a data lake for growth and scale on the AWS Cloud.

Is this data lake reference architecture more applicable to enterprise organizations?

This guide's data lake reference architecture can be applied to data lakes belonging to organizations of any size. The reference architecture standardizes the data exchange interface, lowers the overhead and cost to maintain and grow the data lake, and can be applied to any scale that your organization's data lake grows to.

Can I still use this reference architecture if my organization only has one data producer?

This guide's data lake reference architecture is still relevant and beneficial even if your organization only has one data producer. Without the centralized catalog, your data producer has to handle the growth of data consumers, which adds increasing complexity and overhead. Your data lake is also a long-term asset for your organization and typically organizations add more data producers. For example, you might need an additional data producer to store sensitive data for compliance reasons or because your organization acquires another business unit that has its own data producer.

My data lake directly connects one data producer with multiple data consumers. Is this guide's data lake reference architecture still relevant?

The data lake reference architecture would benefit your organization in the long term. You could use a two-step approach and begin by building the centralized catalog for new data consumers. You could then connect your existing data consumers to the centralized catalog.

Should my organization follow the onboarding and access granting workflow without making changes to it?

No, the main purpose of that section is to illustrate the logical activity blocks required during the onboarding process. All organizations should customize the process and might even have multiple processes, depending on the sensitivity of their data.
Another consideration is that the process flow uses the resource-based sharing approach in AWS Lake Formation. There are other data-sharing methods supported by Lake Formation, such as tag-based sharing, where differences in the process can be tailored for the specific sharing method.
Resources

- [AWS Glue developer guide](#)
- [AWS Lake Formation developer guide](#)
- [Cross-account access in Lake Formation](#)
- [Granting and revoking data catalog permissions in Lake Formation](#)
- [Populating the AWS Glue Data Catalog](#)
## Document history

The following table describes significant changes to this guide. If you want to be notified about future updates, you can subscribe to an [RSS feed](#).

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>— (p. 19)</td>
<td>Initial publication</td>
<td>October 1, 2021</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance glossary

The following are commonly used terms in strategies, guides, and patterns provided by AWS Prescriptive Guidance. To suggest entries, please use the Provide feedback link at the end of the glossary.

Modernization terms

anti-pattern

A frequently used solution for a recurring issue where the solution is counter-productive, ineffective, or less effective than an alternative.

business capability

What a business does to generate value (for example, sales, customer service, or marketing). Microservices architectures and development decisions can be driven by business capabilities. For more information, see the Organized around business capabilities section of the Running containerized microservices on AWS whitepaper.

domain-driven design

An approach to developing a complex software system by connecting its components to evolving domains, or core business goals, that each component serves. This concept was introduced by Eric Evans in his book, Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). For information about how you can use domain-driven design with the strangler fig pattern, see Modernizing Legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

historian modernization

An approach used to modernize and upgrade operational technology (OT) systems to better serve the needs of the manufacturing industry. A historian is a type of database that is used to collect and store data from various sources in a factory.

microservice

A small, independent service that communicates over well-defined APIs and is typically owned by small, self-contained teams. For example, an insurance system might include microservices that map to business capabilities, such as sales or marketing, or subdomains, such as purchasing, claims, or analytics. The benefits of microservices include agility, flexible scaling, easy deployment, reusable code, and resilience. For more information, see Integrating microservices by using AWS serverless services.

microservices architecture

An approach to building an application with independent components that run each application process as a microservice. These microservices communicate through a well-defined interface by using lightweight APIs. Each microservice in this architecture can be updated, deployed, and scaled to meet demand for specific functions of an application. For more information, see Implementing microservices on AWS.

modernization

Transforming an outdated (legacy or monolithic) application and its infrastructure into an agile, elastic, and highly available system in the cloud to reduce costs, gain efficiencies, and take advantage of innovations. For more information, see Strategy for modernizing applications in the AWS Cloud.
modernization readiness assessment

An evaluation that helps determine the modernization readiness of an organization's applications; identifies benefits, risks, and dependencies; and determines how well the organization can support the future state of those applications. The outcome of the assessment is a blueprint of the target architecture, a roadmap that details development phases and milestones for the modernization process, and an action plan for addressing identified gaps. For more information, see Evaluating modernization readiness for applications in the AWS Cloud.

monolithic applications (monoliths)

Applications that run as a single service with tightly coupled processes. Monolithic applications have several drawbacks. If one application feature experiences a spike in demand, the entire architecture must be scaled. Adding or improving a monolithic application's features also becomes more complex when the code base grows. To address these issues, you can use a microservices architecture. For more information, see Decomposing monoliths into microservices.

polyglot persistence

Independently choosing a microservice's data storage technology based on data access patterns and other requirements. If your microservices have the same data storage technology, they can encounter implementation challenges or experience poor performance. Microservices are more easily implemented and achieve better performance and scalability if they use the data store best adapted to their requirements. For more information, see Enabling data persistence in microservices.

split-and-seed model

A pattern for scaling and accelerating modernization projects. As new features and product releases are defined, the core team splits up to create new product teams. This helps scale your organization's capabilities and services, improves developer productivity, and supports rapid innovation. For more information, see Phased approach to modernizing applications in the AWS Cloud.

strangler fig pattern

An approach to modernizing monolithic systems by incrementally rewriting and replacing system functionality until the legacy system can be decommissioned. This pattern uses the analogy of a fig vine that grows into an established tree and eventually overrides and replaces its host. The pattern was introduced by Martin Fowler as a way to manage risk when rewriting monolithic systems. For an example of how to apply this pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

two-pizza team

A small DevOps team that you can feed with two pizzas. A two-pizza team size ensures the best possible opportunity for collaboration in software development. For more information, see the Two-pizza team section of the Introduction to DevOps on AWS whitepaper.