# 15
# Implementing a Data Mesh Strategy

The original definition of a data lake, which first appeared in a blog post by James Dixon in 2010 (see https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/), was as follows:

> *If you think of a datamart as a store of bottled water – cleansed and packaged and structured for easy consumption – the data lake is a large body of water in a more natural state. The contents of the data lake stream in from a source to fill the lake, and various users of the lake can come to examine, dive in, or take samples.*

In his vision of what a data lake would be, Dixon imagined that a data lake would be fed by a single source of data, containing the raw data from a system (so not pre-aggregated like you would have with a traditional data warehouse). He imagined that you may then have multiple data lakes for different source systems, but that these would be somewhat isolated.

Of course, new terms and ideas often seem to take on a life of their own and regularly don't end up looking like the original vision of the creator. And that is true of data lakes, as what happened in the decade between 2010 – 2020 was that many organizations attempted to build centralized data lakes that would contain data from across the organization.

There were of course different implementations and approaches to data lakes, but it was common for an organization to set up a central data engineering team that would become responsible for collecting and processing data into a data lake. Raw data would be collected from across the organization, and a central data engineering team would be responsible for running transformations on the data to further enrich it and to join data across diverse systems. Different lines of business would then make requests to the central data engineering team for new types of transforms or new data sources they wanted to be ingested into the lake.

This approach was common for a long time, but had some significant limitations, as we will discuss in this chapter. In this chapter, we will also introduce a new approach to data lakes that has become popular over the past few years, with a concept known as a data mesh. Specifically, we cover the following topics in this chapter:

- What is a data mesh?
- Challenges that a data mesh approach attempts to resolve
- The organizational and technical challenges of building a data mesh
- AWS services that help enable a data mesh approach
- A sample architecture for a data mesh on AWS
- Hands-on – Implementing a data mesh approach on AWS

Before we get started, review the following *Technical requirements* section, which lists the prerequisites for performing the hands-on activity at the end of this chapter.

# Technical requirements

In the last section of this chapter, we will go through a hands-on exercise that uses Amazon DataZone to implement a basic data mesh approach.

As with the other hands-on activities in this book, if you have access to an administrator user in your AWS account, you should have the permissions needed to complete these activities.

You can access more information about running the exercises in this chapter using the following link: https://github.com/PacktPublishing/Data-Engineering-with-AWS-2nd-edition/tree/main/Chapter15

# What is a data mesh?

The concept of a **data mesh** was introduced around 2019 by Zhamak Dehghani, who at the time was a consultant for a company called ThoughtWorks. The data mesh architecture was built around four principles:

- Domain-oriented, decentralized data ownership
- Data as a product
- Self-service data infrastructure as a platform
- Federated computational governance

Over time, as with data lakes, the term began to mean different things to different people. Some organizations would claim they had implemented a data mesh because they had enabled data sharing between multiple data lakes, while others would go all in with organizational change, in addition to building technology stacks to support a data mesh.

I believe that it is okay for a term to evolve and change, but that does mean that when someone uses a term such as data mesh, you need to ask them exactly what that means to them. If someone defines a data mesh as the ability to share data between multiple data lakes or analytical systems, then that is fine, as long as you understand their limited definition. Other people you speak to may define their data mesh as an organization-wide program that is intended to modify their approach to how data is produced, transformed, and consumed, in a way

that involves both organizational (people, process, culture) changes and technology changes. The second definition is closer to what Dehghani envisioned for a data mesh.

In this chapter, we will dig deeper into the original intention for the data mesh, mostly following the concept as defined by Dehghani. In the original blog posts about the data mesh concept (`https://martinfowler.com/articles/data-monolith-to-mesh.html`), Dehghani made it clear that she was not proposing a data mesh as a technical solution. Rather, she defined the data mesh as an approach for how to organize responsibilities around analytical data, and the fundamental requirements that needed to be in place to gain maximum value for analytical data across an organization. So, as we start to look at the four principles that Dehghani defined for building a data mesh, remember that these are approaches, and not technical solutions or designs. Later in this chapter, we will review potential architectures for building a data mesh on AWS.

Let's dive into the four principles for a data mesh, starting with domain-oriented, decentralized ownership.

## Domain-oriented, decentralized data ownership

This first principle is around organization structure and who is responsible for creating the analytics data that is developed out of the transactional data that runs the business. A core idea with a data mesh, in the way that Dehghani laid it out, is that you no longer attempt to have a central team that collects and processes all the analytical data for an organization. It is important that transactional and analytical data responsibilities and ownership instead be defined in terms of business domains.

If we use an example of a company that streams music (such as Spotify, Amazon Music, or Apple Music), then there may be a business domain responsible for managing users/customers, and a separate do-

main for managing the music catalog (in addition to many other domains for items like partners, playlists, etc.). Based on the modern microservices approach to application development, each of these teams is likely to be responsible for creating APIs for managing their domain (such as creating a new user, or adding a new song to the catalog), and will also own the transactional data that they generate (the user database, or the song catalog database).

With a traditional approach, a centralized data engineering team may have worked with these teams to ingest their data into a central data lake, and the centralized team would also be responsible for performing various ETL-related tasks (cleaning data, ensuring data quality, reacting to schema changes, joining across different datasets, etc.).

However, with a data mesh approach, one of the core principles is that analytical data is now also domain-oriented and decentralized. There is no longer a central team that collects data from across domains in order to process the data for analytics, but rather each domain becomes responsible for not only their transactional data and related APIs, but also for creating an analytical set of data for their domain. For example, the music catalog team creates an analytical dataset that groups all artists in the catalog by the country where their music is published, and the user team creates a cleaned, master list of all users, as well as a dataset showing the number of users per country.

Each team supporting a specific business domain works independently (in other words, analytical data is decentralized), but, as we discuss later, there is a central governance team that helps to ensure standards and controls for data that is part of the data mesh.

Let's now look at the second principle for developing a data mesh, and that is around treating analytics datasets produced by a domain team as a *data product*.

## Data as a product

Along with this new approach of having decentralized, domain-oriented teams creating analytical data, there comes a concept of treating the data that is created as a product. Much as a team may create a software product, in the data mesh world, the team will create a **data product**.

When you create a software product, you are responsible for knowing who your customers are and what they want, and for delivering a high-quality product that is well documented, easily accessible, reliable, regularly updated, etc. You need to apply the same type of thinking to building a data product.

For example, the team that owns the music catalog may also have data related to streams of each song in the catalog. They also know that identifying the top 20 streamed songs for each day, month, and year is useful to other teams in the organization (such as the marketing team). As a result, the music catalog team creates three new tables in the data lake – top daily songs, top monthly songs, and top songs of the year so far.

Other parts of the business rely on having access to this aggregated data on top streaming songs, and it is the responsibility of the music catalog team to ensure that they deliver this data on time. Dehghani outlined some of the attributes, or properties, that a data product should have, including:

- **Discoverability and accessibility** (ability for other teams in the business to discover that the dataset exists and learn how to access it, or request access to the data)
- **Security** (the data should only be accessible to people who have a right to access it, and ideally there should be fine-grained access controls in place so that some people may be able to see all columns, while others will have a limited view)
- **Understandability** (data consumers should be able to understand the data, such as what data is stored in each column and how that

data is defined for the business. Additional documentation may also provide examples of how to use the data, etc.)

Other aspects of data product thinking include agreeing on an **SLA** (**Service Level Agreement**) for your data product, which sets expectations between the data producer and data consumer around both the quality of the data and the timeliness of the data being updated. You should also have a cross-domain governance team that defines some standards for data products, such as agreement on a date format used in all analytical products, or data types for specific fields.

One of the recommendations for implementing this approach for the creation of data products is to create a new role in each domain team for a **data product manager**/owner. Much as each software product has one or more people that are product managers, responsible for understanding customer requirements, setting the roadmap, and delivering a high-quality product, the same can be applied to the data product. Specifically tasking someone with the responsibility of developing and maintaining a data product for the domain is one of the organizational changes that should be implemented when developing a data mesh.

Let's now move on to some of the technical aspects of building a data mesh, as we look at the next principle, which is having *self-service data infrastructure as a platform*.

## Self-service data infrastructure as a platform

One of the concerns about having domain-oriented, decentralized teams is that there could be duplication of effort and expense if each team is responsible for setting up their own data infrastructure. Therefore, a recommendation for organizations that are looking to implement a data mesh is that you create a central team that is responsible for data infrastructure, but not responsible for implementing business logic in data pipelines.

That distinction between the data pipeline infrastructure and the actual data pipelines that implement business logic is important. The central data mesh team is responsible for building out infrastructure and systems that make it easy for data engineers working in a data domain to easily build ETL pipelines that apply business logic to create data products.

Some of the components that the central data mesh team may provide include:

- Scalable storage buckets (such as Amazon S3) for storing data
- Ingestion tooling (such as Amazon DMS, Glue, or third-party tools such as Upsolver)
- Data quality tools (such as Glue DataBrew, Glue Studio Data Quality, or third-party tools such as Deequ)
- Data transformation tools (such as Glue Studio, or third-party solutions such as Databricks)
- A central data catalog for making data products discoverable (such as Amazon DataZone, or third-party tools such as Collibra)
- Automation for routing data access requests to a data owner, and automated sharing of data once a request is approved (such as with Amazon DataZone, or can be built on top of a third-party tool such as Collibra)
- Data warehouse solution for low-latency data access (such as Amazon Redshift, or third-party solutions such as Snowflake)
- Data access control solutions (such as AWS Lake Formation, or third-party tools such as Privacera)
- Data orchestration tools (such as Amazon MWAA, AWS Step Functions, or third-party tools such as Apache Airflow)
- **Continuous Integration (CI)/Continuous Delivery (CD)** infrastructure for deploying pipelines (such as AWS CodePipeline, or third-party tools such as GitHub Actions)

The central data mesh engineering team should provide the infrastructure, as well as documentation and support for using that infra-

structure. The purpose of this team is to enable the domain specific data engineers to easily build high-quality data products, make them discoverable through a central catalog, and ensure they are easily accessible with data sharing (i.e., make sure that other teams can access the data in place, without needing to make a copy of the data they want to access).

Let's now look at the last principle suggested by Dehghani, *federated computational governance*, which addresses the interoperability and governance of data products.

## Federated computational governance

**Federated computational governance** is a complex phrase, but effectively means that all data product owners and the data platform owner are represented in a central group that agrees on governance policies for data in the data mesh, and that governance is enforced/monitored through automation.

Previously, you may have had a central governance team that wrote up standards and governance policies that a central data engineering team would then work to implement; however, this changes with the decentralized data mesh approach. You still need certain standards and governance agreements in order to ensure a good level of interoperability between different data products, and to ensure that corporate governance requirements are met. However, these are now decided by a governance group that is made up of representatives that include data owners and the data platform owner.

The goal is to provide as much independence and flexibility as possible to the individual domain teams, while also ensuring good interoperability between data products. To do this, the data governance team can define certain standards, such as the date format that should be used, and how columns should be named for something like customer ID (for example, ensuring all columns with the customer ID are named

`customer_id`, instead of having different data products use variations such as `cust-id`, `customer_identifier`, `customer_key`, etc.).

Other examples of items that the data governance team may create standards for include items around data quality metrics that must be reported for each data product, and a minimum set of metadata that must be captured for each data product registered in the data mesh catalog. The goals of these standards are to ensure that high-quality, trustworthy data products are easily discoverable in the data mesh, and that the data products can easily be joined through common field names and items like date formats. However, as far as possible, each data domain team should have as much freedom as possible to create data products, with only minimal and necessary standardization and governance policies being applied from the federated governance team.

Finally, as far as possible, the data platform team should create automation to monitor compliance with the standardization and governance rules that the governance teams put in place. This should include the automated monitoring of published data products to verify compliance, along with alerting to notify relevant data product owners and the governance team of data products that are out of compliance.

For example, there should be an automated process that runs data quality checks against published data products, ensuring that data quality requirements are being met. For data products that are not meant to contain any PII (such as those where the PII is obfuscated in some way), there can be an automated process that scans data products, looking for PII, and notifying relevant stakeholders if PII is detected.

In addition to the four principles that we have outlined above, there is another important concept to understand when talking about a data

mesh, and that is the concept of data producers and data consumers, which we discuss next.

## Data producers and consumers

When people talk about a data mesh approach, you will often hear them talk about data producers and data consumers. These are used to identify two distinct personas that work within a data mesh, and how each of them interacts differently with the data mesh.

**Data producers** are teams that publish new data products within the data mesh. Teams within a domain will create analytical data products, from their transactional data, and publish the resulting dataset on the data mesh.

**Data consumers** are other teams within an organization that will search the data catalog for data that they need, and then subscribe to a dataset published by another team. For example, the marketing team may discover a dataset in the data catalog that contains the top 1000 songs that are streamed from the music catalog of a streaming service. They subscribe to the dataset to use this top 1000 songs data in their marketing campaigns.

However, data consumers may also be data producers. For example, if there is a dataset listing the top 1000 songs streamed from the music catalog, the marketing team may join that dataset with a dataset that contains details about artists and create a new dataset that lists the top 10 artists with the most popular songs on the streaming service. For example, in a specific week, Taylor Swift may not have the top-streamed song, but she may have 10 different songs that are in the top 1000 list. As a result, she may be the most popular artist in the list of the top 1000 songs. The marketing team may choose to publish this list of the most popular artists in the top 1000 streaming songs as a new and separate dataset, and publish this to the data mesh, making them both a data consumer and data producer.

Having a better understanding of what is meant by a data mesh, let's now look at some of the items that a data mesh approach helps to fix.

# Challenges that a data mesh approach attempts to resolve

Traditional data lakes and approaches served many organizations well for a long time, but as with everything, there are always new developments and approaches that help drive improvements.

In the previous chapter, we looked at how new table formats (such as Apache Iceberg) introduced new functionality that improved querying and processing data in data lakes. In a similar way, the concepts and approaches introduced by a data mesh help solve some different challenges of traditional data lakes and how data teams are structured.

Let's look at a few of the traditional challenges that a data mesh helps solve.

## Bottlenecks with a centralized data team

While not the case for every data lake, it was common for large enterprises to create a centralized team that would ingest data from transactional systems across the organization and then perform ETL tasks on that data (cleaning the data, joining data from across different sources, etc.). This team would respond to requests from different parts of the business when they needed new data sources ingested or new reports created.

However, this was not always the most efficient way to get the analytics insights that each team needed. Often the central team would be overloaded with new ingestion or transformation requests, and this central team would also often need time to learn about and understand the new data they were ingesting. This was because the central team was taking data from across the organization and could not be

expected to be experts about the data and business logic of every part of the organization.

The central team would also end up having to try and prioritize the different requests coming from across the business, without necessarily having visibility into which request would have the biggest business impact.

One of the big benefits of a decentralized data mesh approach is that data stays within a domain and therefore close to domain experts. If each domain team hires data engineers to create analytical products of their data, those data engineers can become experts in the data for that specific domain. These domain experts will have a good idea of what good data quality looks like, and will more easily identify data quality problems. They will also get to know that part of the business well, and be in the best position to respond to the reporting and visualization requirements for that specific business domain (as well as know which data sources they need and how best to apply business logic transforms to create the required reports).

Finally, with the self-service approach that is core to a data mesh, each business domain can easily discover data that has been generated by other business domains and can directly request access to that data without needing to go through a central team to coordinate everything.

## The “Analytics is not my problem” problem

When you create a central team that is responsible for doing all the data engineering tasks and creating the datasets and reporting that the business requires, there can be friction between this team and the data source owners.

Traditionally, there was strong separation of duties between the engineers responsible for building and running transactional systems and

the engineers responsible for creating analytic reporting for the business. And when data engineers need data from a source system, the owners of the source system may get nervous about how that data is going to be extracted from the source database.

This would sometimes cause the owners of the source system to complain that analytics was not their problem (meaning not their responsibility). And strictly, they were correct – their primary responsibility to the organization is to keep the transactional systems up and running and performing well. Anything that potentially could jeopardize that (such as a data engineering system trying to read the full database) is considered a risk, and a source system owner may fight against accepting that risk. This of course can lead to delays in data engineers being able to build the reports that another part of the business may be requesting.

Therefore, when implementing a data mesh and, as part of that, changing the organizational culture regarding how data is comprehensively viewed and who is responsible for creating it (both transactional and analytical data), much of that friction can be removed.

For this to be successful does require organizational and cultural change (which can be difficult). But the end goal is to have business domains take responsibility for both the transactional data and analytical data for their domain. And if a team building a new transactional data source is aware that their wider team will also be responsible for making the analytics data available, and they involve data engineers from the design phase of the new system, this can significantly smooth the process of ensuring that required data can be efficiently extracted from the transactional system.

## No organization-wide visibility into datasets that are available

When we discussed data products in the previous section, we spoke about how there are various attributes of data products that are core to the data mesh approach, and this included discoverability and accessibility.

With a centralized approach to data lakes, there ends up being a core data engineering team that collects all the data and then reactively responds to requests from across the organization to build new reports or to make specific data available. Often business requests go to that team from business users that don't know what data or existing reports the central team has. The data engineering team may find that they already have the required data, but at times they will need to go to a different part of the business to find the source data and arrange to ingest it (as discussed in the "**Analytics is not my problem**" problem section above).

With traditional data lakes, a lot of activity was reactive. But with a data mesh approach, it is far more of a proactive, self-service approach. Each domain is responsible for creating analytical products for the data that they own, and the data platform team creates automation to enable new data products to be added to a central data catalog. All users across the organization can search the catalog to discover datasets, and from the catalog should have the ability to request access to a specific dataset. Once that is approved by the data owner, there should be automatic data sharing that makes the data available to the data consumer that requested it.

The above are just a few of the challenges that a data mesh approach helps to overcome. Let's now do a deeper dive into the other side of the coin – the organizational and technical challenges of building a data mesh.

## The organizational and technical challenges of building a data mesh

As we discussed at the start of this chapter, a data mesh may mean different things to different people. Some people approach a data mesh implementation as though it were just a technical challenge about improving the sharing and creation of analytical data. But as we have seen, the way that Dehghani proposed a data mesh approach is not about technical solutions to data sharing, but much more about the overall way that an organization approaches analytical data.

In this section, we look at some of the challenges (both organizational and technical) of implementing a data mesh.

### Changing the way that an organization approaches analytical data

While there are technical challenges to building a data mesh, the more difficult part is changing the way that an organization views analytical data, and changing who is responsible for creating analytical data.

While there is no “traditional” way to creating analytical data, it has been very common for this to be the job of a central analytics team. Many companies, even today, still have a data & analytics team that is responsible for ingesting transactional data into a central data store (such as a data warehouse or data lake). Once ingested, this central team is also responsible for cleansing the data (ensuring data quality) and for building ETL pipelines to transform the transactional data into analytics data. A data mesh approach changes this completely.

Let’s take a look at how a data mesh changes things for the centralized data & analytics team.

## Changes for the centralized data & analytics team

With a data mesh approach, the centralized data mesh team now has a different responsibility – they need to build and run a data platform

that domain teams throughout the organization can use for building analytical data products. Where as their customers used to be business users that wanted specific data or reports for analyses, their new customers are now data engineers in different business teams that want a platform that enables them to easily build data products.

Where as previously they were responsible for implementing and running technology that they then used to build ETL pipelines, they are now focused on building a data platform that others can use to build ETL pipelines. They effectively are now responsible for building and running an internal **Software-as-a-Service** (**SaaS**) data platform for internal data engineers.

This is a change in their focus, but they are using existing skills they already had for deploying data-related infrastructure.

As part of the changing role of the data platform team, a new data platform owner role should be created. This role should work with the data domain teams in the different lines of business to learn about their requirements and expectations from the data platform, and in turn develop a data platform development roadmap that is regularly updated. The data platform owner takes on the responsibility of ensuring that the data platform features and functionality meet both the central governance team requirements and the requirements of the individual domains.

The change within the lines of business is more fundamental, as it introduces brand-new responsibilities and skills requirements, as we will discuss next.

## Changes for line of business teams

The more challenging organizational change is making lines of business responsible for generating not just transactional data, but also relevant analytical data products. This means that a line of business now needs to upskill for various roles, as we outline here.

**Data product owners**

Each line of business needs to appoint data product owners/managers that will be responsible for the development of analytical data products. The data product owners need a strong understanding of their specific line of business and must also understand the bigger picture of how other lines of business may want to use their data.

The data product owner is ultimately responsible for the creation of data products, as well as for the quality of those products. They need to work with the owners of transactional data systems to facilitate the ingestion of data from those systems, work with data architects or engineers to confirm ingestion methods and required transformations, and ensure that the finished data product is added to the central data catalog.

They are also ultimately responsible for ensuring the usability of the data that is generated, communicating any schema changes to downstream customers, and taking feedback from other parts of the business on any new requirements. The data product owner also needs to determine how the data should be delivered (such as batch updates to the data lake, making the data available via a streaming source such as Kafka or Kinesis, or loading the data into a data warehouse for access via JDBC).

**Data architects/engineers**

Each line of business also needs to hire data engineers (or repurpose existing data engineers) that are capable of developing ETL pipelines that can use the central data platform to ingest data from line of business transactional systems and transform that data into analytical data products that can be shared with the rest of the organization.

Depending on the size of the line of business, they may choose to separate the role of data architect and data engineer. If separated, the data architect would work with the data product owner to design end-to-

end pipelines from data ingestion to making the data available to consumers, and then work with data engineers to communicate the requirements for the pipeline.

The data engineers will either work closely with the data architect or directly with the data product owner to understand business requirements, enable ingestion of data from transactional systems, implement business logic in ETL pipelines for transforming data, and load the data product into a final destination where it can be consumed (data lake, data warehouse, streaming target, etc.). They will also play a key role in ensuring data quality.

Some teams may already have data architects/engineers; however, for other organizations, this may be a new role for a line of business. In some cases, the line of business may be able to move existing data architects/engineers from the centralized data engineering team into the line of business, reallocating existing resources as part of the organizational restructuring to implement the data mesh. Otherwise, the line of business may need to create new roles to fill the data architect/engineer positions.

**Data stewards**

The other role that is key in a data mesh is that of the **data steward**, a role that is responsible for ensuring that the data products created by a data producer meet the requirements of the central governance team policies. In addition to meeting the central governance policies, the data steward may also create governance policies that are specific to their line of business.

As discussed previously, the central governance team is responsible for implementing the minimum required policies to meet corporate governance standards, and governance standards that all domain producers agree should be common for the data mesh. However, a specific line of business, such as the HR team, may have additional gover-

nance requirements for their data products. The data steward helps define, and enforce, these policies.

Some of the functions that the data steward may be responsible for include ensuring that all published data products for their line of business have required metadata, and that any PII privacy requirements are enforced. The data steward may also take on the responsibility of ensuring that data quality is maintained for all published data products.

Data stewards will work with data product owners, as well as the data architects and engineers, in ensuring that governance and related requirements are met.

Data stewards may also be nominated to join the federated governance team that works with other data stewards across the organization to set minimum data governance requirements.

For any organization that is looking to adopt a data mesh approach to their data, it is critical that they first address the organizational changes that we have discussed here before they look into implementing any type of *technical* solution.

For this to be successful, it is critical that there is executive buy-in to implementing a data mesh approach. Without executive-level sponsorship of the data mesh, it will not be possible to implement the organizational changes that are required. The executive that sponsors the project needs to be able to work with teams across the organization in order to educate them on the business value of moving to a data mesh approach, and to then secure the required buy-in from business leaders.

Once support has been secured, a comprehensive education program can be put in place to educate people from all parts of the organization

on what a data mesh is, how it will be implemented, and what changes can be expected.

The organizational changes required for implementing a data mesh are often the biggest challenge, but there are also significant technical challenges, which we look at in the next section.

### Technical challenges for building a data mesh

While many vendors advertise that they have the technical solution for building a data mesh, the reality is that there are no single solutions that out of the box enable everything that is needed for data mesh implementation at enterprise scale. In large enterprises, there is likely to already be a wide variety of technologies in use, and many different tools will need to be integrated into the new data mesh approach. For smaller organizations though, out-of-the-box solutions may be suitable.

## Integrating existing analytical tools

If you are building a brand-new platform (such as for a new company, or in the event you're looking to modernize legacy on-prem solutions as you move to the cloud), then this is not a significant challenge. You can select a single vendor that supports a data mesh approach and use their suite of products. For example, you could build a data mesh using AWS-native services, or select a vendor such as Databricks or Snowflake, and use their data mesh solutions.

However, large organizations often already have a wide range of established analytical tools in use across the organization. And different parts of the organization may have different toolsets. For example, it's not uncommon to have an organization where one team has built a data lake in AWS, and another team has built a data lake in Azure or **Google Cloud Platform** (**GCP**). Even if all data is in AWS, for example, you may find that different teams use different tools – such as one

team using AWS-native services, while another team uses Databricks, and another team uses Snowflake. Integrating those different tools into a data mesh is certainly possible, but it generally takes a lot of custom development to build the required integrations. We look at one aspect of that integration in the next section, where we discuss the centralized data catalog.

## Centralizing dataset metadata in a single catalog and building automation

A key part of a data mesh is having a centralized catalog where metadata for all data products is published. This enables all data consumers within the organization to go to a central catalog to search for and discover published data products, learn more about the data product (through comprehensive metadata that should be published with the data product), and request access to the data.

For large enterprises, there may already be one or more data catalogs in existence. Your challenge in this environment is to select a single data catalog solution that provides the needed functionality for your data mesh, and then get agreement from all lines of business to use this new catalog for all new data products.

However, you will not want to just “lift and shift” existing catalogs into the new catalog, because the data mesh has specific requirements around governance, required metadata, data quality, etc. Therefore, a line of business may end up continuing to use their existing data catalog for “legacy” datasets, while needing to publish all new data mesh data products into the new catalog. Therefore, this is both a technical and organizational/process change and challenge.

A big focus of the data mesh approach is about creating a self-serve environment for data producers and consumers. This means building

in as much automation of processes as possible. One common place for this automation is in requesting, approving, and provisioning access to data products.

In an ideal world, a data consumer should be able to log in to the central data catalog using their corporate credentials, and from there they should be able to search for relevant data for their use case. The search terms that they provide should be matched against attributes of the data products, such as column names, table descriptions, and additional metadata associated with the data product.

Once a data consumer finds a data product in the catalog, they should be able to learn more about the data product by reviewing the provided metadata, and ideally also viewing the lineage of the data product (which sources were used in the creation of the product, and what transforms and joins were applied). They may also be able to view a sample of the dataset.

If the data consumer decides that a specific data product is well suited to their use case, they should be able to request a subscription to the data product from within the catalog. Some data products may be marked as being available to all users in the organization without additional approval being required, in which case the data consumer should immediately receive access to the data. Access to the data product may be via a JDBC connection, or it could be provided via the data being shared with the consumer (such as the dataset being shared from a source Redshift or Snowflake data warehouse, with a target Redshift or Snowflake data warehouse that the data consumer has access to).

Other datasets may require the data owner or data steward from the publishing domain to approve the request for access. In this case, the data consumer should be able to provide a justification for why they want access, and the request should be routed to the data producer.

Once the data producer approves the request for access, the data should be automatically made available to the data consumer.

If an enterprise has a wide variety of analytic tools, then this will generally require custom development to enable this level of automation. Alternatively, if an organization is using analytical tools from one primary vendor, the integration will be much simpler.

## Compromising on integrations

Because of the complexities of integrating different analytical tools, across different vendors and perhaps even different clouds and on-prem data sources, an organization may compromise on their vision of what a data mesh is. The data mesh is a good approach to making analytical data available across an organization, but it is not always practical to enable every integration to reach the full vision of what a data mesh should be, as defined by Dehghani.

As a result, organizations may implement limited aspects of the data mesh approach, or they may build a new data mesh that operates alongside existing analytical systems. This approach may be viewed by some as failing to really implement a data mesh, but my view is that if you gain at least some of the benefits of the data mesh approach, and you are able to do this in a reasonable time frame and cost, and in a way that brings concrete business benefits, then you should go with that. Ideally, over time, you will mature your data mesh to be more in line with the vision created by Dehghani. This is why at the start of this chapter we discussed how the term data mesh means different things to different people.

## AWS services that help enable a data mesh approach

Most analytics vendors have been adding functionality to their solutions to support a data mesh approach over the past few years. And

while there is currently no single solution that enables a data mesh across a complex selection of analytical tools and hybrid environments, many companies have made good progress in supporting the data mesh approach, at least within their own "ecosystem" of tools.

AWS has supported sharing both S3-and Redshift-based datasets across AWS accounts for a while. And while easy sharing of data across different AWS accounts (and different teams within an organization generally have their own AWS accounts) is a key component of a data mesh architecture, it is only a piece of building a data mesh. Another key component is the ability to centrally catalog data and add rich business metadata for each dataset, which can be done with the **Amazon DataZone** service. In this section, we explore the AWS services that can help build an AWS-based data mesh in more detail.

### Querying data across AWS accounts

In AWS, two of the primary query engines that are used for data consumption are Amazon Athena (for ad-hoc queries of data in an S3 data lake, and federated queries to other data sources) and Amazon Redshift (for low-latency use cases, such as for powering dashboards or visualizations in BI tools). Both tools can access data that has been shared across accounts using AWS Lake Formation.

## Sharing data with AWS Lake Formation

We previously discussed the AWS Lake Formation service (in *Chapter 4*, *Data Governance, Security, and Cataloging*) and how it provides the ability to enforce granular data access controls. Using AWS Lake Formation, in conjunction with the Glue Data Catalog, you can control access to data in your S3-based data lake by granting or revoking permissions for users through the Lake Formation console (or API). This includes the ability to control access at the column or row level.

With Lake Formation, you can also enable the sharing of data between different AWS accounts. As a reminder, in the Glue Data Catalog you have databases, which contain tables, and each table has one or more columns. When you share a database or table from a source AWS account to a target AWS account using Lake Formation, the database tables become visible in the target account's catalog. Once a user in the target account is granted access to the shared database/table, they can query the table using Amazon Athena as though it were locally in the account, even though the data files are still in the source AWS account's S3 bucket. With Lake Formation data sharing, the data is *not copied* to a target account, but rather *made accessible* in the target account.

In a similar way, you can use Lake Formation to manage Redshift data shares. This enables you to configure the sharing of a Redshift table from a data producer's Redshift cluster directly to a data consumer's Redshift cluster. Using Lake Formation, you can share both S3 and Redshift data between separate AWS accounts, as well as share data between different AWS Regions within the same account.

From a data mesh perspective, this enables data producers in one domain to grant access to their data products to data consumers in a different domain. A common pattern would be to have a business data catalog where data consumers can find datasets that they wish to subscribe to. Within the data catalog, the consumer should be able to request access to a dataset, and once approved by the data producer, access should be granted to the dataset. For data products that are S3-or Redshift-based, you can build an automation that uses Lake Formation to share the relevant table/s from the data producer account to the data consumer account.

Once S3-based data has been shared to a data consumer's account, a data consumer can query the data using Amazon Athena, Amazon Redshift Spectrum, or Amazon QuickSight, as though the table were in the account locally. And in a similar way, if a Redshift table has been

shared across accounts using Lake Formation, a data consumer can view the table in Redshift as though it were a local table in their cluster, and immediately start querying it.

In October 2023, AWS launched a new service that provides business data catalog functionality integrated with Lake Formation data sharing, called Amazon DataZone. Let's explore how this service can significantly simplify the process of building a data mesh on AWS for environments that use AWS analytic services.

## Amazon DataZone, a business data catalog with data mesh functionality

Amazon DataZone is a service from AWS that provides built-in data governance features, helping to unlock the power of data across an organization.

While AWS does not specifically market Amazon DataZone as a data mesh solution, the service does include functionality that simplifies building out a data mesh, via integration with AWS Lake Formation. Let's review some of the core concepts and components found in DataZone.

# DataZone concepts

The following are some of the core DataZone concepts that you need to understand before we move on to examine the various components that make up DataZone.

**Domains**

The first concept to understand is that of **domains**, which are a way of organizing data assets, projects, associated AWS accounts, and data sources.

Most commonly, a domain will align with a specific organizational boundary, such as a business unit or specific function. As such, you may have domains for finance, HR, manufacturing, sales, marketing, etc.

**Data sources**

DataZone supports publishing data to the business data catalog from the Glue Data Catalog and Amazon Redshift. While traditionally the Glue Data Catalog is used to catalog Amazon S3-based data lake resources, it can also be used to capture database and table information from other sources, such as relational databases or a Snowflake data warehouse. All sources that are supported by AWS Glue crawlers can also be imported as a DataZone data source.

**Business glossaries**

A business glossary is effectively a dictionary used to define business-related metadata, and to ensure the consistent use of business terms. For example, an organization may have datasets related to their business in different countries around the world. If you allow the free-form capture of metadata associated with a dataset, different people may label countries differently, making it difficult to consistently search the data catalog based on the country that the data applies to. For example, you could have the following metadata applied to different datasets:

- Country: United States
- Country: USA
- Region: U.S.
- Geography: united_states

If the data catalog had the metadata shown above, it would be difficult to search the catalog to find all datasets with data from the United States. Therefore, you can instead create a business glossary called

*Country*, and then create a list of all countries that your organization operates in, as shown below:

- Country
  - United States
  - South Africa
  - New Zealand
  - India
  - United Kingdom

When adding metadata to a dataset, a data producer or data steward can select to apply metadata from the *Country* glossary and select the specific country the dataset applies to. This ensures consistency in metadata, making it easier to find related datasets.

**Metadata forms**

Metadata forms can be used to ensure that datasets for a specific domain contain a consistent set of metadata. A domain data steward creates metadata forms, where they specify metadata that should be captured for datasets. Multiple metadata forms can be created, and then a data steward can select forms to be applied to all datasets in the domain. Optionally, a data publisher can also elect to attach additional metadata forms to datasets that they publish.

A metadata form lists field names for each metadata item, and then a field type. The field type could be a string (allowing for free-form entry), Boolean value, date, integer, or decimal. Alternatively, the field type could be tied to a specific business glossary term. The following is a simple example of items that could be included in a metadata form.

| Field Name | Field Type | Description |
|---|---|---|
| Sales Region | *Country* Business Glossary | The sales region for this dataset |
| Sales Quarter | *Quarter* Business Glossary | The calendar quarter that this dataset covers |
| Dataset Support | String | Provide the email address to use for queries about this dataset |
| Last Audit Date | Date | Provide the date that this dataset was last audited |

*Figure 15.1: Sample items for a metadata form in Amazon DataZone*

Metadata forms are critical to ensuring that all data in the business data catalog has good metadata that can be used to help discover datasets, and to better understand the dataset.

**Associated AWS accounts**

A DataZone domain is deployed to a specific AWS account, and you can then associate additional AWS accounts with the domain.

This enables teams that have their own AWS accounts where they produce or consume data to continue to use those accounts. They can publish data from their account into the DataZone business data catalog, and when they subscribe to a data product from another project in a different account, they can have the data shared into their account for local consumption.

Having reviewed some of the core DataZone concepts, let's now look at the key components of the DataZone service.

# DataZone components

DataZone has a number of components that are used to provide an environment where users can search for and discover data, subscribe to a data product, and publish a data product. In this section, we will review two key DataZone components.

**Data portal**

When you configure Amazon DataZone, a data portal UI is created that is accessible from outside of the AWS console via a unique URL generated for your DataZone domain. The data portal provides the ability to manage your data products and is also a full business data catalog that enables data consumers to easily search for and discover data products.

As part of your data mesh, it is important that all potential data consumers across the domain can easily access the data catalog. By creating the data portal UI outside of the AWS console, it means that you do not need to grant AWS console access to all data producers and consumers in your domain. Instead, you can integrate the data portal UI with your **Single Sign-On** (**SSO**) provider, enabling users to easily log in to the data portal using their corporate credentials.

While DataZone administrators may still need AWS console access for some aspects of managing DataZone, most regular users will only work in the data portal UI. Within the data portal, data producers can do tasks such as add business metadata to their data products, view and approve (or reject) subscription requests from data consumers, etc. And for data consumers, the data portal allows them to search for and discover data products, as well as to request access to a data product directly from the portal.

**Projects and environments**

Projects provide a way to group different resources together, including people, data assets, and analytical tools. When you create a project in the DataZone data portal UI, you are enabling a collaborative environment where a group of users can discover, subscribe to, and consume data registered in the business data catalog, as well as create new data products and publish them as part of a project.

Within a DataZone project, you can create environments, which are collections of configured resources (such as Amazon S3 buckets, AWS Glue databases, or Amazon Athena workgroups). Each environment also has a set of IAM principals that are granted access to use those resources. **DataZone environment profiles** are pre-configured sets of resources and blueprints that are a template for creating a new environment.

Instead of access to data products being granted to individuals, with DataZone, access is granted to a project. As a project administrator, you can then add people to the project, and they can use the access granted to the project to access the data.

Having a better understanding of the Amazon DataZone service, let's look at what an architecture for a data mesh on AWS looks like. In the following section, we will illustrate how DataZone can be a technical enabler of a data mesh approach to managing data.

# A sample architecture for a data mesh on AWS

We have looked at what a data mesh is, some of the organizational and technical challenges of building a data mesh, and finally some of the AWS services that can be used to build a data mesh. Now, in this section, let's bring it all together with a sample architecture for a data mesh on AWS.

## Architecture for a data mesh using AWS-native services

Earlier in this chapter, we discussed how a data mesh is easier to build if starting new, or if just using AWS-native services. In this section, we will look at a sample architecture for when you only use AWS-native services, and in the section after this, we will review an architecture for environments that use analytic tools from other vendors.

The following architecture diagram shows a data mesh that is built using Amazon DataZone, with data in an Amazon S3-based data lake (a similar architecture would support data in Amazon Redshift) and using AWS Lake Formation for data sharing.

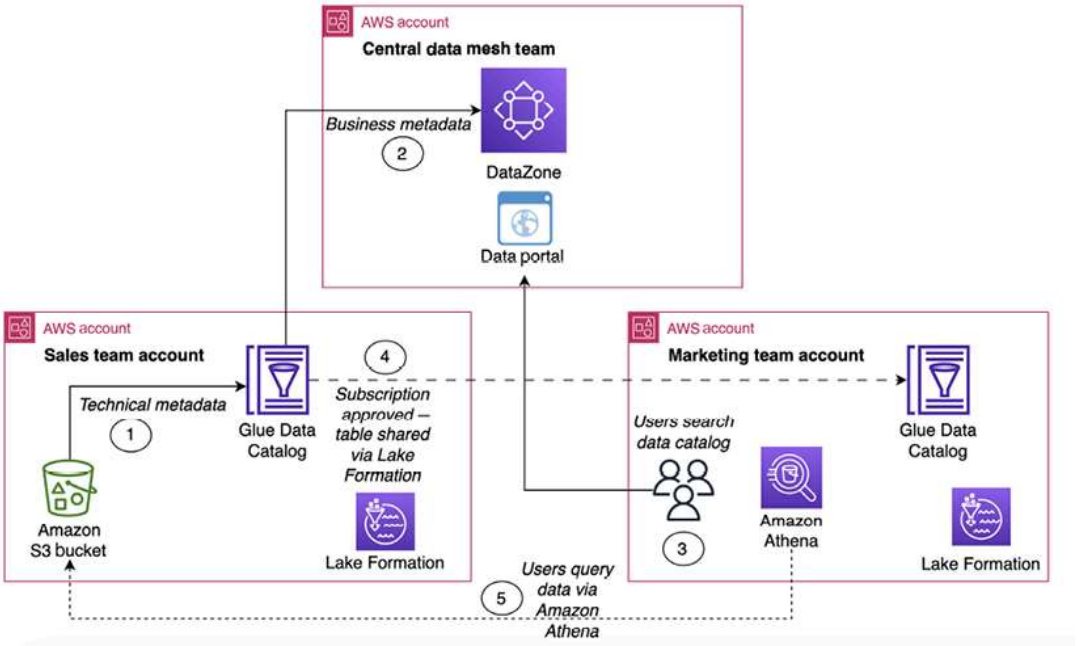


*Figure 15.2: Sample data mesh architecture when using AWS analytic services*

In the preceding diagram, we see how there is an account for the *central data mesh team*, and this is the account where we have created our DataZone domain (and therefore is also the account that hosts the DataZone data portal). This account doesn't host data producers or data consumers but is owned and managed by the central data mesh team. Here they create the DataZone domain and also manage the high-level administrative tasks for DataZone. The central data mesh team will also associate AWS accounts from data producers and consumers with the DataZone domain.

In addition to the central data mesh account that deploys the domain for Amazon DataZone, we also see two other accounts. On the left, we see a data producer account. This account is owned by the *sales* team, and in this account the data engineers on the sales team create data products to be part of the DataZone *sales* project. On the right, we see the account for the *marketing* team, who in this case are consumers of a sales data product.

While this diagram may look somewhat complicated, the architecture is relatively simple. Let's go through each of the numbers in the preceding diagram to explain the process:

1. A data product is created and stored in an Amazon S3-based data lake. A Glue crawler (or other method) is used to register the technical metadata for the data product in the AWS Glue Data Catalog of the sales team's AWS account.
2. A member of the sales team that has been added to a DataZone project adds the data product to the DataZone catalog. They do this by creating a new DataZone data source that points to the database in the AWS Glue Data Catalog. This effectively imports the table metadata from the Glue Data Catalog, and they can then add relevant business metadata via a metadata form and the business glossary.
3. A user in the marketing team (who has been added to a relevant DataZone project for the marketing domain) searches the catalog and discovers the dataset from the sales team. They choose to subscribe to the data product, and a data steward or the data publisher in the sales team approves the subscription.
4. Once the subscription is approved, DataZone automatically uses AWS Lake Formation to share the dataset from the sales account Glue Data Catalog to the marketing team's Glue Data Catalog.
5. The marketing team users can now use the deep-links in the DataZone data portal UI to access Amazon Athena and query the sales team data. When they run a SQL query, Athena accesses the data in the S3 bucket of the sales team, using Lake Formation-provisioned permissions.

This architecture is a good solution for where you are primarily using AWS-native services. In the preceding diagram, we showed an architecture that uses data in Amazon S3, but this architecture would also support Amazon Redshift tables in a similar way.

Let's now review an architecture for an environment where there is a broad range of analytic services, from multiple vendors.

## Architecture for a data mesh using non-AWS analytic services

As we have discussed in this chapter, Amazon DataZone (at the time of writing) supports the automated sharing of data in an Amazon S3-based data lake and data in Amazon Redshift. However, if you have data in other systems (such as Snowflake, or Databricks data cataloged in the Databricks Unity catalog), as well as perhaps on-premises data in other data warehouses (such as Teradata), then a data mesh architecture is more complex.

While most of these data sources could be cataloged in the Glue Data Catalog and then imported to Amazon DataZone from there, DataZone would not support enabling the automated sharing of those data sources across different domains.

Also, if you have more advanced catalog and data governance requirements (such as data lineage, which is not currently supported in Amazon DataZone), then you may want to consider a different business data catalog solution, such as Collibra, Atlan, or Alation.

There are many different ways that you could build out a data mesh architecture in this environment, so the following example is just one potential pattern that you could use, although this is a pattern that is relatively common. In this example, we use just a single analytic service (Snowflake), but later on we will discuss approaches for when you have multiple analytic solutions from various vendors.

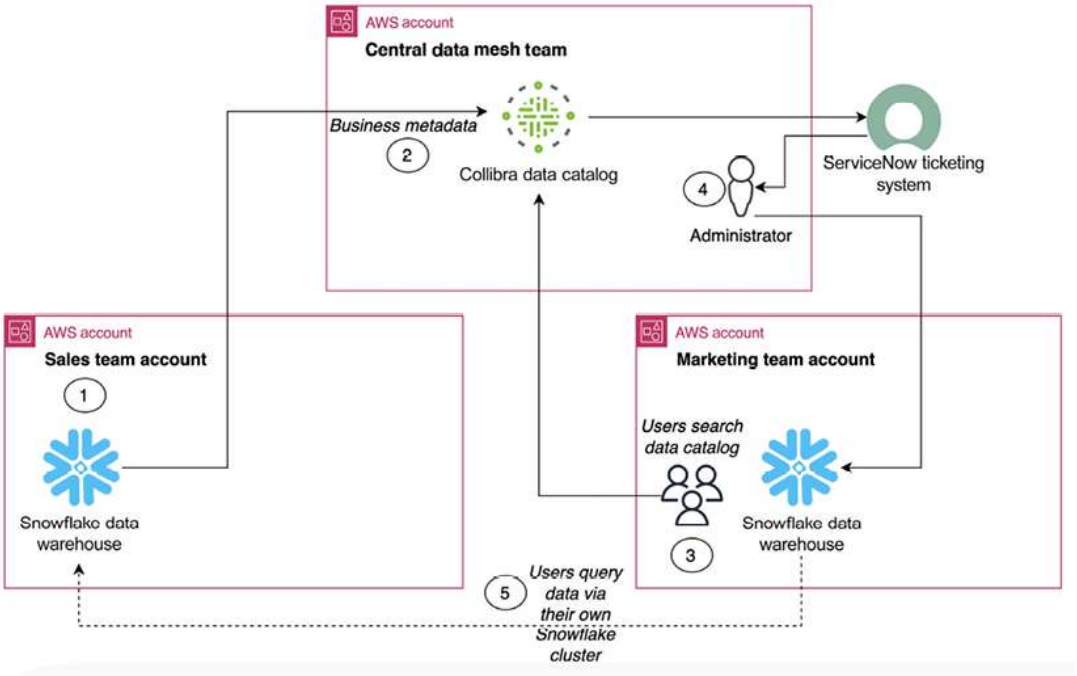


*Figure 15.3: Sample data mesh architecture when not using AWS analytic services*

The preceding diagram shows a simplified architecture where just Snowflake is used as an analytic service, but the same architecture could be expanded to include additional analytic services such as Databricks, an on-premises Teradata data warehouse, Oracle, and others. We will talk about that in a moment, but let's first work through the steps shown in *Figure 15.3*:

1. A data product is created and stored in the Snowflake data warehouse.
2. Using the Collibra JDBC driver for Snowflake, the data producer registers the data product in Collibra and adds the relevant business metadata. Note that we are using Collibra as an example, but this could be replaced with Atlan, Alation, or any other business data catalog of your choosing.
3. A user in the marketing team searches the Collibra data catalog and discovers the dataset from the sales team. They choose to subscribe to the data product, and a data steward or the data publisher in the sales team approves the subscription.

4. Once the subscription is approved, Collibra automatically creates a ticket in ServiceNow (this could be using a custom-built integration, or an integration downloaded from the Collibra marketplace). An administrator is notified of the approved data share via ServiceNow, and they then log in to Snowflake and share the specified dataset from the sales Snowflake cluster to the marketing team's Snowflake cluster.
5. The marketing team users can now use their Snowflake cluster to query the data from the sales team.

Of course, there are multiple variations and different ways that this integration could have been built. Let's discuss a few different options that we could use to modify the above architecture.

# Automating the sharing of data in Snowflake

If Snowflake were the only analytic tool that we were using, then it would make sense to simplify things and remove the step of having an administrator manually do the datashare from the source Snowflake cluster to the target Snowflake cluster.

In the Collibra marketplace, there is an integration that enables Collibra to automatically share the relevant dataset between Snowflake clusters. See **https://marketplace.collibra.com/listings/snowflake-data-share-to-collibra-integration/**

In the architecture shown in *Figure 15.3*, we used the ServiceNow integration, as a typical environment may have many different analytic tools, and there may not be a Collibra integration for each of those tools. Therefore, in a more complex environment, the simplest solution may be to just have Collibra raise a ServiceNow ticket, and then have an administrator do the share manually, as each analytic tool would have a different process for sharing data.

# Using query federation instead of data sharing

In the architecture shown in *Figure 15.3*, we used the built-in functionality of Snowflake to make data in the producer Snowflake cluster (the sales team) available in the consumer Snowflake cluster (the marketing team). To the marketing team, the sales data would appear as though it were a local table, and they could easily use the Snowflake query tool on their cluster to join the sales data with their marketing data.

However, not every analytic tool may support this type of data sharing. For example, if you have on-premises legacy data warehouses, it is unlikely that they would support data sharing to a different cluster. Therefore, an alternate approach is to use query federation to enable users in one team (such as marketing) to be able to easily access and query data from another team (such as sales), without needing to copy the data.

In *Chapter 11*, *Ad-Hoc Queries with Amazon Athena*, we discussed Athena query federation. As a reminder, this functionality enables you to create connectors to other database systems, and then run queries that can join data in the S3-based data lake with data from other database systems. To configure this in Amazon Athena, you need to install and configure the appropriate connector, which does require some relatively advanced technical skills.

As an alternative, there are commercial solutions from vendors that enable the same type of functionality but with an easy-to-use tool. One of the popular products in this space is Starburst. With Starburst, you can easily configure integrations with many popular database and analytical solutions. Starburst then sits between SQL clients and the target database. Users connect their SQL client to Starburst, and Starburst has connectors to various database systems. A data consumer can then

query any of the database systems that they have been granted access to via Starburst.

If we were to incorporate Starburst into the architecture shown in *Figure 15.3*, when a subscription is approved for a data product, an administrator would grant access to the data product in the sales team Snowflake cluster to the marketing team consumers via Starburst. The marketing team could then connect to Starburst to query the sales data.

The benefit of query federation is that it allows a data consumer to subscribe to multiple datasets, all in different analytic tools, and access them all via Starburst. However, it does mean that the data platform team has to license and manage additional infrastructure (such as the Starburst server), and in certain cases there may be increased query latency when querying via query federation.

Having looked at sample architectures, let's now get hands-on with setting up a simple data mesh environment in the AWS console using Amazon DataZone.

# Hands-on – Setting up Amazon DataZone

In the hands-on section of this chapter, we are going to set up and configure the Amazon DataZone service. We will then create a DataZone project, import a data source, add business metadata, and publish a data product. Finally, we will access the DataZone data portal as a data consumer, to search for and subscribe to a data product.

At the time of publication of this book, the Amazon DataZone service has just recently been released. There are sometimes a number of changes made to a service shortly after it becomes Generally Available (GA), so make sure to reference the GitHub page for this chapter to check for any updates related to these hands-on exercises. The GitHub

page is available at https://github.com/PacktPublishing/Data-Engineering-with-AWS-2nd-edition/tree/main/Chapter15

Let's get started.

## Setting up AWS Identity Center

To log in to the **Amazon DataZone data portal**, you can either use your AWS IAM credentials, or you can log in using an identity configured in **AWS Identity Center**. With AWS Identity Center, you can create local users, or you can perform identity federation using SSO by configuring Identity Center to work with your identity provider (such as Azure Active Directory, Okta, Ping Identity, etc.). For this hands-on exercise, we are going to set up AWS Identity Center with local users.

Note that Identity Center is an AWS Organizations-wide service, meaning that to configure Identity Center, you must log in with the credentials of your AWS Organizations management account. If you created a new account for performing the activities in this book, then the instructions provided here will help you get Identity Center configured. However, if you are using a sandbox or other work account that is part of AWS Organizations, then you will either need to work with your cloud management team to configure Identity Center, or you can use local IAM accounts instead of Identity Center:

1. Log in to the **AWS Management Console** and use the top search bar to search for, and open, the **IAM Identity Center** service.
2. Ensure that you are in the Region that you have been using for the exercises in the previous chapters of this book.
3. Within the Identity Center console, click the **Enable** button in the **Enable IAM Identity Center** box. If your account is not currently configured to be part of an AWS organization, you will be prompted to allow Identity Center to create an organization for you. Click on **Create AWS Organization**. Note that you will receive

an email verification request, and you must verify your email address within 24 hours.

4. By default, the **Identity Source** is configured to be the **Identity Center directory,** which is what we will use to create local users in this exercise. Therefore, you do not need to edit the Identity Source, but if you wanted to link Identity Center to your SSO provider, this could be done at this point.
5. Using the left-hand menu, navigate to the **Groups** page, and click on **Create group**.
6. Provide a **Name** for the group, such as **DataZone Users**.
7. Optionally provide a **Description** for this group, and then click on **Create group**.
8. Using the left-hand menu, navigate to the **Users** page, and click on **Add user**.
9. We want to create a user that will be used to manage our DataZone domain. So, for **Username**, set this to `film-catalog-team-admin`. In a production environment, we would use the username of an actual person, but to make it easier to track the different roles that we will log in to DataZone with, we are setting the name to be more descriptive of the role.
10. Provide an email address to be associated with this user. As we have done elsewhere in this book, we can create a unique email address that will go to our primary email by specifying our email address with a PLUS sign and a unique identifier after the name portion of the email. Gmail and Outlook both support this mechanism, but not all email providers may support this, in which case you should first create a new email address with a provider to use for this step. If your provider supports this, and your email was, for example, `gareth-eagar@example.com`, then you could use `gareth-eagar+film-catalog-team-admin@example`.
11. Fill in **First Name** and **Last Name**, and then click on **Next**.
12. On the **Add user to groups** screen, select the **DataZone Users** group and then click **Next**.
13. Review the details, and then click **Add user**.

14. Check your email for a message titled **Invitation to join AWS IAM Identity Center**. Click **Accept Invitation** in that message. In the page that opens, provide a password for your new account. **NOTE**: Identity Center requires passwords to be a minimum of 8 characters, and must contain lowercase, uppercase, numbers, and a special character.
15. **Repeat steps 8 – 14** to add a second user. Set the **Username** for this user to be `marketing-team-admin`.

In the above steps, we enabled AWS Identity Center, created a new user and group, and set the password for our new user. Let's now configure our DataZone domain.

## Enabling and configuring Amazon DataZone

Amazon DataZone domains are used to organize your data assets, users, and associated projects. In this section, we will set up Amazon DataZone by configuring a domain:

1. Log in to the **AWS Management Console** and use the top search bar to search for, and open, the **DataZone** service.
2. Click on **Create domain**.
3. Under **Domain details**, provide a name for the domain. We will be creating a domain that manages data related to our movie streaming business, so let's call our domain `streaming`.
4. Optionally provide a description, and then under **Quick setup** click the checkbox for **Set-up this account for data consumption and publishing**.
   By selecting Quick setup, DataZone will enable default blueprints for working with Data Lake and Data Warehouse data. This will also automate the configuration of required permissions and resources, and create a default DataZone project and environment profiles.

5. Leave all other defaults, scroll to the bottom of the page, and click on **Create domain**.
6. It may take a few minutes for your domain to be created. Once it is created, the **Welcome to your DataZone Domain** screen is displayed. Scroll down this screen, and in the **Summary** section, click on **Update domain to enable IAM Identity Center** (or alternatively, display your list of domains, click on the **streaming** domain, and then click **Edit**).
7. In the **User management** screen, click the **checkbox** for **Enable users in IAM Identity Center**. Leave the default option of **Do not require assignments**, which will allow any Identity Center user to access the DataZone portal. In a production environment, you may want to require assignments, so that only approved users can access the DataZone portal.
8. Scroll down and click on **Update domain**.
9. In the **Data Domain** console, scroll down to the **Summary** section again, and click the **Copy** icon to the left of the **Data portal URL**.

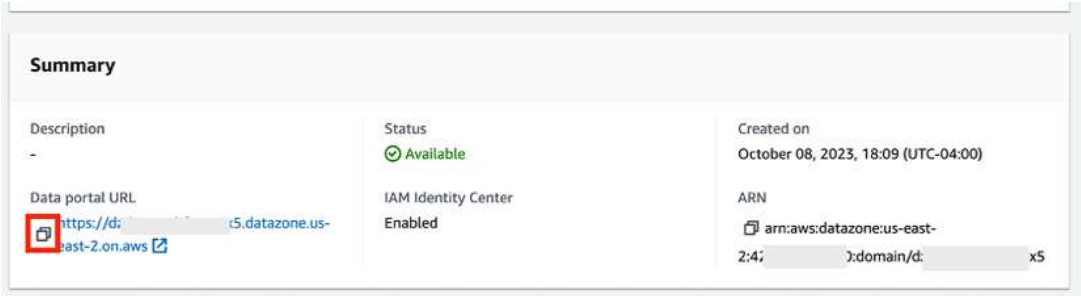


*Figure 15.4: DataZone console showing Data Portal URL*

10. Open a **new private/Incognito browser** window, and paste the **Data Portal URL** in that window. Note that if you do this in your current browser, you will be automatically logged in as your IAM user, but we want to log in as the **movie-catalog-team-admin** user. Therefore, use a private/Incognito window.
11. Log in to the data portal using the username ( `film-catalog-team-admin` ) and password you configured in AWS Identity Center.

12. In the **Welcome to Amazon DataZone** popup, click on **Create New Project**.
13. In the **Create project** popup, enter `Film Catalog Project` for **Name**. Optionally provide a description, and then click **Create**.
14. On the **Your project is ready to use** popup, click on **Create environment**.
15. For the environment **Name**, enter `Film Datasets S3` as this is the project where we will produce datasets related to our catalog of films, using data in Amazon S3.
16. For **Environment profile**, select **DataLakeProfile**. This creates an environment that can be used to work with data in an Amazon S3-based data lake.
17. You can optionally customize the environment to specify the name to use for a Glue database that is used for data that you publish, as well as the name of a Glue database that is used to contain tables that you subscribe to from other environments. You can also optionally specify the name of the Athena workgroup that you want to use. These are optional, so you can skip these and just click **Create environment**. DataZone will now create Glue objects (producer and consumer database), an S3 bucket, and other resources, as well as configure default Lake Formation permissions for the objects. This may take a few minutes to complete.

With the above steps, we have configured our DataZone domain and created a new project and environment that we will use to produce datasets related to our catalog of films. We can now move on to adding a data source to our project.

## Adding a data source to our DataZone project

We are now ready to add one of our existing Glue datasets to DataZone. To do this, we need to configure a data source:

1. In the **DataZone data portal**, make sure you are in the **Film Catalog Project** and on the **Overview** tab.

2. In the **Working with Projects** section, click on **Create data source**.
3. For **Name**, enter `films-CuratedZoneDB` , optionally provide a description, and then for **Data source type**, select **AWS Glue**.
4. For **Select an environment**, chose the **Film Datasets S3** environment from the dropdown.
5. In the **Data Selection** section, enter `curatedzonedb` for **Database name**. This is the name of the AWS Glue database that we have been using throughout this book in order to store our curated datasets.
6. For **Table selection criteria**, leave the default of the wildcard asterisk, which will add all of the tables in our **curatedzonedb** to DataZone. Then click **Next**.
7. For **Publishing settings**, leave the default of **No** for **Publish assets to the catalog**. This will add the tables to the project inventory, and later, after reviewing and adding additional metadata, we can choose to publish the tables to the data catalog.
8. Leave the default setting of having **Automated business name generation** enabled.
9. Click on **Next**.
10. For **Run preference**, select **Run on demand** and then click **Next**.
11. Review the summary of settings, and then click **Create.**
12. On the `films-CuratedZoneDB` summary page, click on **Run** to run the data import process.
13. When the import completes, the three tables in the **curatedzonedb** in our Glue Data Catalog should now be listed, as shown in the following screenshot. Even though we have multiple tables, for this exercise we will focus on the `film_category` table.

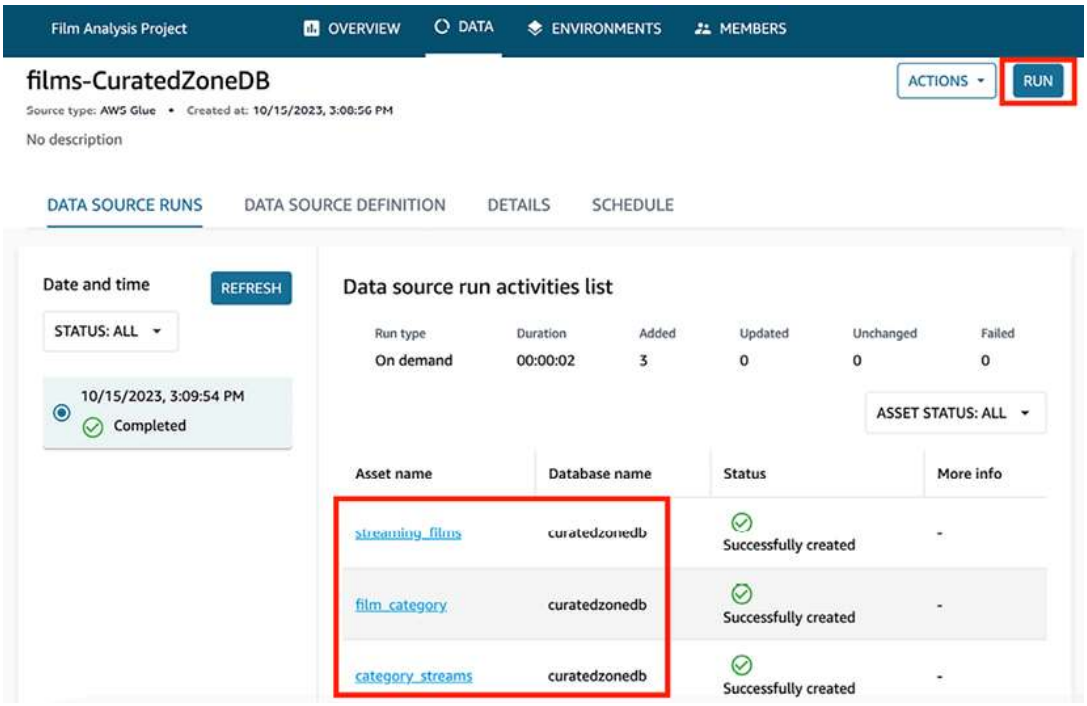


*Figure 15.5: DataZone import from the Glue Data Catalog*

Now that we have imported data into DataZone, we can add additional business metadata and then publish to the data catalog.

## Adding business metadata

We can now review the current metadata for one of the tables we imported and add additional business metadata:

1. In the DataZone data portal, click on the **Data** tab along the top. This provides an overview of data for this project.
2. On the left-hand side menu, click on **Inventory data**, and then click on the `film_category` table. Inventory data contains tables that we have imported into DataZone but that have not yet been published into the DataZone business data catalog.
3. Click on **Schema** to view the schema of this table. For each column, a green icon indicates where automated metadata has been generated. For example, the first column in this table is `category_id`, and DataZone has automatically provided a label of **Category ID** for this column. You could approve each automatic metadata item

individually, but in this exercise, we will accept all the changes by clicking the **ACCEPT ALL** button at the top of the screen, as shown in the following screenshot.

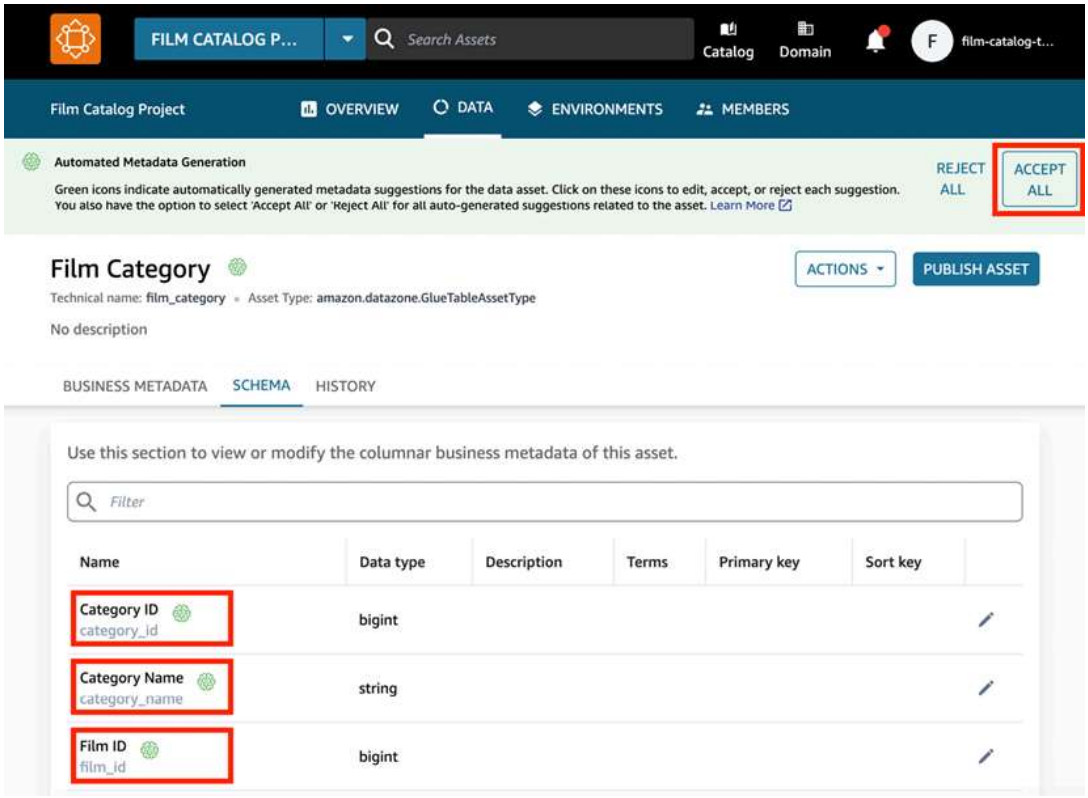


*Figure 15.6: Automated metadata for the Film Category table*

4. Let's make this dataset easier to discover for our business users by adding additional business metadata. At the top right, click on **Actions**, and then select **Edit** from the dropdown.
5. In the **Edit Asset** popup, change the name to `Movie listings with category` . For **description**, provide the following: *This table contains a complete listing of all films in our streaming movie catalog, including category/genre information for each film, as well as a list of special features*.
6. Click on **Update Asset**.
7. Click on the **Schema** tab, then click the **Pencil** icon next to the **Category Name** field. For description, add the following: *This field contains the movie category/genre. Sample categories include Animation, Comedy, Sports, Children, Drama, Action, and more*.

8. Click **Save**.
9. We could add additional metadata for each of the columns, but to save time we are going to publish the data asset to the category as it is now. Click on **Publish asset**, and then in the confirmation popup, click **Publish asset** again.

This table is now added to the business data catalog, and is discoverable by other users of the catalog.

## Creating a project for data analysis

We can now create a new project that will be used by our marketing team to do data analysis:

1. Log out of the DataZone data portal by clicking on your username in the top right and clicking **Log out**.
2. Sign back in to the DataZone data portal with SSO, but sign in as the `marketing-team-admin` user. Note that you may be signed back in as the `film-catalog-team-admin` automatically, but if you are logged out again and you attempt to log in a second time, you should be prompted for your username.
3. Once logged in, click on **Create Project**.
4. For the project name, enter `marketing-team-analysis` . Optionally provide a description, and then click **Create**.
5. In the popup confirming that the project has been created, click on **Create environment**.
6. For the environment **Name**, enter `marketing-team-datalake` , and optionally provide a description.
7. Under **Environment profile**, select **DataLakeProfile** from the dropdown.
8. Leave all other defaults, and then click **Create environment**.

Now that our analysis project is ready, we can search for and subscribe to data.

## Search the data catalog and subscribe to data

In this section, our marketing team searches the business data catalog and subscribes to a dataset:

1. The marketing team wants to analyze the movie catalog in order to identify the most popular genres of films. Use the search bar along the top of the DataZone data portal console and search for `movie genre`.
2. In the results, you should see the `Movie listings with category` table. Note how even though the original table in the Glue Data Catalog was called `films_category`, we were able to find the table based on the business metadata we had added in DataZone. **Click on the table name** to select it.
3. We can now review details of the table, including schema information for the table. We decide that this table will be good for our analysis, so click on **Subscribe**.

---

> Note the warning message on this page, indicating that this dataset is an **unmanaged asset**. With DataZone, if an S3-based table uses Lake Formation for permissions management, or you have a Redshift table, these are considered **managed assets** and DataZone can enable the automatic sharing of these resources between different projects/teams.
>
> Because our `film_category` table has not been configured to use Lake Formation permissions, it is listed as unmanaged. This means we can still request a subscription to the dataset, and the data owner will receive a notification to approve access. However, once they approve the subscription, they will need a process outside of DataZone to grant permissions to the team that requested the data. This could be, for example, working to add IAM-based permissions for the Glue

database, table, and S3 data location for the `film_category` table to the marketing team IAM role.

In addition to modifying the table to use Lake Formation permissions, to configure the table for automatic sharing managed by DataZone, a DataZone administrator would also need to grant a set of permissions to a specific IAM role that is auto created by DataZone to be used by our marketing-team-analysis project. This is beyond the scope of this book, but refer to the following documentation for more information on this topic: https://docs.aws.amazon.com/datazone/latest/userguide/lake-formation-permissions-for-datazone.html

---

4. In the **Subscribe popup**, select the **marketing-team-analysis** project from the drop-down, and then provide a **Reason for request**. For example, this could be *Need access to the movie listings with category table for a new marketing project we are running*.
5. Click on **Subscribe**.

## Approving the subscription request

We now log in to the DataZone portal as the `film-domain-admin` (the owner and publisher of the `film_category` table) so we can approve the request:

1. Log out of the DataZone data portal by clicking on your username in the top right and clicking **Log out**.
2. Sign back in to the DataZone data portal with SSO, but sign in as the **film-catalog-team-admin** user.
3. Click on the **Notifications icon** at the top right (just to the left of your username). You should see a list of alerts, with the most recent

alert being **Subscription request created**. Click on the notification.

4. Review the **Subscription request** popup, which provides details about the requestor ( `marketing-team-admin` ) and the subscriber (the `marketing-team-analysis` project). This page also lists the **Reason for access** that the subscriber provided. Leave a short comment on your decision in the **Decision comment** box, and then click on **Approve** to approve the request.

As mentioned previously, since this dataset is not managed using AWS Lake Formation permissions, the dataset will not have been automatically shared with the marketing team. Therefore, a process would need to be in place to now grant the marketing team access, such as by updating the marketing team IAM policies to grant access. In order to get the full benefits of DataZone automated data sharing between projects and accounts, it would be recommended to ensure that all your S3 data is managed by Lake Formation.

# Summary

In this chapter, we looked at the concept of a data mesh, a relatively new approach to data management. And while this concept has only been around since 2019, it has been rapidly adopted in many organizations.

The data mesh approach moves away from centralizing data analytics within a single team, and instead uses a decentralized approach, moving the responsibility for creating analytic data products closer to the teams that produce, or own, the operational data. In this chapter, we examined several different aspects of a data mesh, including the people, process, and organizational changes required to implement a data mesh, as well as some of the architecture approaches to implementing the technology required to enable a data mesh.

We also explored the core concepts and components of the Amazon DataZone service, which provides data governance functionality, in-

cluding a business data catalog. After looking at potential data mesh architectures within AWS, we then got hands-on with the Amazon DataZone service. In the hands-on exercise, we set up DataZone, imported a dataset from the AWS Glue Data Catalog, and added business metadata to the dataset. We then had a look at how to search the DataZone business data catalog and how to subscribe to a data product.

As we near the end of this book, we will look at some of the key concepts required to build a modern data platform on AWS, which we will cover next.

# Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

https://discord.gg/9s5mHNyECd