5
Architecting Data Engineering Pipelines

Having gained an understanding of data engineering principles, the core concepts, and the available AWS tools, we can now put these together in the form of a data pipeline. A data pipeline is the process that ingests data from multiple sources, optimizes and transforms it, and makes it available to data consumers. An important function of the data engineering role is the ability to design, or architect, these pipelines.

In this chapter, we will cover the following topics:

- Approaching the task of architecting a data pipeline
- Identifying data consumers and understanding their requirements
- Identifying data sources and ingesting data
- Identifying data transformations and optimizations
- Loading data into data marts
- Wrapping up the whiteboarding session
- Hands-on – architecting a sample pipeline

Technical requirements

For the hands-on portion of this lab, we will design a high-level pipeline architecture. You can perform this activity on an actual whiteboard, a piece of paper, or using a free online tool called *diagrams.net*. If you want to make use of this online tool, make sure you can access the tool at **http://diagrams.net**.

You can find the code files of this chapter in the GitHub repository using the following link: https://github.com/PacktPublishing/Data-Engineering-with-AWS-2nd-edition/tree/main/Chapter05.

Approaching the data pipeline architecture

Before we get into the details of the individual components that will go into the architecture, it is helpful to get a 10,000 ft view of what we're trying to do.

A common mistake when starting a new data engineering project is to try and do everything at once, creating a solution that covers all use cases. A better approach is to identify an initial, specific use case and start the project while focusing on that one outcome, but keeping the bigger picture in mind.

This can be a significant challenge, and yet it is really important to get this balance right. While you need to focus on an achievable outcome that can be completed within a reasonable time frame, you also need to ensure that you build within a framework that can be used for future projects. If each business unit tackles the challenge of data analytics independently, with no corporation-wide analytics initiative, it will be difficult to unlock the value of corporation-wide data.

An ideal project will include sponsorship from the highest levels of an organization but will identify a limited-scope project to build an initial framework. This project, when completed, can be used as an internal case study to drive forward additional analytic projects.

In the 1989 film *Field of Dreams*, a farmer (played by Kevin Costner) hears a voice saying

> *"If you build it, he will come."*

Everyone in the town thinks he is crazy when he ends up sacrificing his crops to build a baseball field, but when he does, several long-dead baseball players come to the field to play. In business, a common mantra is the following:

> *"If you build it, they will come."*

This implies that if you build something really good, you will find customers for it. However, this is not a recommended approach for building data analytic solutions.

Some organizations may have hundreds, or even thousands, of data sources, and many of those data sources may be useful for centralized analytics. But that doesn't mean we should attempt to immediately ingest them all into our analytics platform so that we can see how a business may use them. When organizations have taken this approach, embarking on multi-year-long projects to build out large analytic solutions covering many different initiatives, they have often failed.

Rather, once executive sponsorship has been gained and an initial project with limited scope has been identified, the data engineer can begin the process of designing a data pipeline for the project.

Architecting houses and pipelines

If you were to build a new house, you would identify an appropriate piece of land and then contract an architect to work with you to create the plans for the building. The architect would do several things:

- Discuss your requirements with you (how you want to use the home, what materials you would like, how many bedrooms and

bathrooms, and so on).
- Gather information on the land where you will be building (the size of the land, slope, and so on).
- Determine the type of materials that are best suited to build in that environment.

As part of this, the architect may create a rough sketch showing the high-level plan. Once that high-level plan is agreed upon, the architect can gather more detailed information and then create a detailed architectural plan. This plan would include the layout of the rooms, and then where the shower, toilet, lights, and so on would go and, based on that, where the plumbing and electrical lines would run.

For a data engineer creating the architecture for a data pipeline, a similar approach can be used:

- Gather information from project sponsors and data consumers on their requirements. Learn what their objectives are, what types of tools they want to use to consume the data, the required data transformations, and so on.
- Gather information on the available data sources. This may include what systems store the raw data, what format that data is in, who the system and data owner are, and so on.
- Determine what types of tools are available and which may be best suited for these requirements.

A useful way to gather this information is to conduct a whiteboarding session with the relevant stakeholders.

Whiteboarding as an information-gathering tool

Running a whiteboarding session with relevant stakeholders enables a data engineer to develop a high-level plan for the data pipeline, helping to gather the information required to start working on the detailed design. The purpose of the whiteboarding exercise is not to work out

all the technical details and finalize the specific services and tools that will be used. Rather, the purpose is to agree with stakeholders on the overall approach for the pipeline and to gather the information that's required for the detailed design.

In this book, we will use an architectural approach, where we ingest data into an Amazon S3-based data lake. Data is initially ingested into a raw zone, and then we transform and optimize the data using several tools to move the data through different data lake zones. As we covered in *Chapter 2*, *Data Management Architectures for Analytics*, a data lake has multiple zones that the data moves through. Typically, this includes zones such as *raw*, *transformed*, *conformed*, and *enriched*, but a data lake can also include zones such as *staging* and *inference* (for data science purposes). There is no specific number of zones that a data lake requires, as zones should be based on business requirements, but for our whiteboarding session, we will show three zones.

Depending on data consumption requirements, we can then load subsets of the data into various data marts (such as Amazon Redshift, a cloud data warehouse service), making the data available to data consumers via various services.

The following diagram illustrates a high-level overview of the primary components of a typical data pipeline and the approach to developing the high-level pipeline architecture. Note that the diagram below shows data coming in on the left, and data consumers accessing the data on the right, which is a logical layout to understand the pipeline (data must be ingested before it can be consumed). However, when we approach designing the pipeline, we work backward, starting with the requirements of our data consumers, before then looking at data sources, and finally, meeting in the middle by looking at the required data transformations.

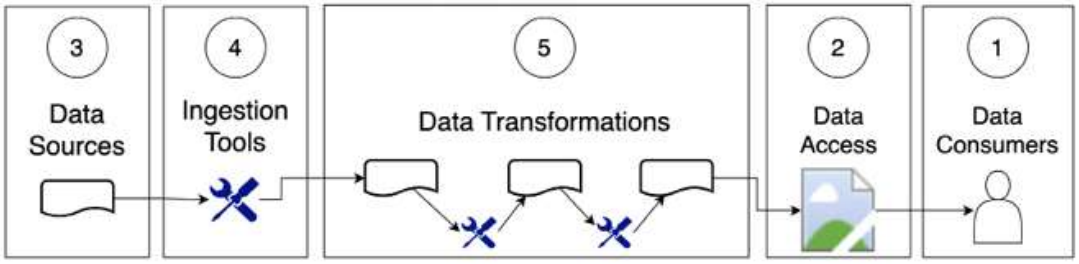


Figure 5.1: High-level overview of a data pipeline architecture

When approaching the design of the pipeline, we can use the following sequence (which is also reflected by the numbers in the preceding diagram):

- Understanding the business objectives, who the data consumers are, and their requirements
- Determining the types of tools that data consumers will use to access the data
- Understanding which potential data sources may be available
- Determining the types of toolsets that will be used to ingest data
- Understanding the required data transformations at a high level to take the raw data and prepare it for data consumers

As you can see, we should always work backward when designing a pipeline – that is, we should start with the data consumers and their requirements, and then work from there to design our pipeline.

Conducting a whiteboarding session

Once an initial project has been identified, the data engineer should bring together relevant stakeholders for a workshop to whiteboard the high-level approach. Ideally, all stakeholders should meet in person, have a whiteboard available, and should plan for a half-day workshop. Stakeholders should include a group of people that can answer the following questions:

- Who is the executive sponsor, and what is the business value and objective of the project?
- Who is going to be working directly with the data (the data consumers)? What types of tools are the data consumers likely to use to access the data?
- What are the relevant raw data sources?
- At a high level, what types of transformations are required to transform and optimize the raw data?

The data engineer needs to understand the business objectives, and not just gather technical information during this workshop. A good place to start is to ask for a business sponsor to provide an overview of current challenges, and to review the expected business outcomes, or objectives, for the project. Also, ask about any existing solutions or related projects, as well as any gaps or issues with those current solutions.

Once the team has a good understanding of the business value, the data engineer can begin whiteboarding to put together the high-level design. We will work backward from our understanding of the business value of the project, which involves learning how the end-state data will be used to provide business value, and who the consumers of the data will be. From there, we can start understanding the raw data sources that will be needed to create the end-state data, and then develop a high-level plan for the types of transformations that may be required.

Let's start by identifying who our data consumers are and understanding their requirements.

Identifying data consumers and understanding their requirements

A typical organization is likely to have multiple different categories, or types, of data consumers. We discussed some of these roles in *Chapter*

1, *An Introduction to Data Engineering*, but let's review them again:

- **Business users**: A business user generally wants to access data via interactive dashboards and other visualization types. For example, a sales manager may want to see a chart showing last week's sales by sales rep, geographic area, or top product categories.
- **Business applications**: In some use cases, the data pipeline that the data engineer builds will be used to power other business applications. For example, Spotify, the streaming music application, provides users with an in-app summary of their listening habits at the end of each year (top songs, top genres, total hours of music streamed, and so on). Read the following Spotify blog post to learn more about how the Spotify data team enabled this: https://engineering.atspotify.com/2020/02/18/spotify-unwrapped-how-we-brought-you-a-decade-of-data/.
- **Data analyst**: A data analyst is often tasked with doing more complex data analysis, digging deeper into large datasets to answer specific questions. For example, across all customers, you may be wondering which products are most popular by different age or socio-economic demographics. Alternatively, you may be wondering what percentage of customers have browsed the company's e-commerce store more than 5 times, for more than 10 minutes at a time, and in the last 2 weeks but have not purchased anything. These users generally use structured query languages such as SQL.
- **Data scientist**: A data scientist is tasked with creating machine learning models that can identify non-obvious patterns in large datasets, or make predictions about future behavior based on historical data. To do this, data scientists need access to large quantities of raw data that they can use to train their machine learning models.

During the whiteboarding workshop, the data engineer should ask questions to understand who the data consumers are for the identified project. As part of this, it is important to also understand the types of tools each data consumer is likely to want to use to access data.

As information is discovered, it can be added to the whiteboard, as illustrated in the following diagram:

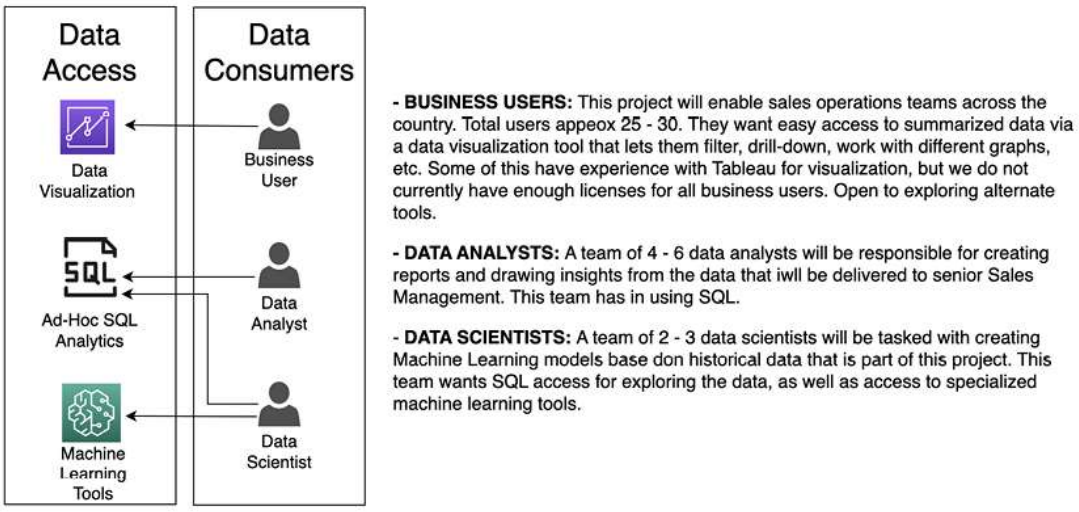


Figure 5.2: Whiteboarding data consumers and data access

In this example, we can see that we have identified three different data consumers – a data analyst team, a data science team, and various business users. We have also identified the following:

- That the data analysts want to use ad hoc SQL queries to access data
- That the data science team wants to use both ad hoc SQL queries and specialized machine learning tools to access data
- That the business users want to use a **Business Intelligence** (**BI**) data visualization tool to access data

It is useful to ask whether there are any existing corporate standard tools that the data consumer must use, but it is not important to finalize the toolsets at this point. For example, we should take note if a team already has experience with Tableau (a common BI application) and whether they want to use it for data visualization reporting. However, if they have not identified a specific toolset to use, that can be finalized at a later stage.

Once we have a good understanding of who the data consumers are for the project, and the types of tools they want to use to work with data, we can move on to the next stage of whiteboarding, which is to examine the available data sources and means to ingest the data.

Identifying data sources and ingesting data

With an understanding of the overall business goals for the project, and having identified our data consumers, we can start exploring the available data sources.

While most data sources will be internal to an organization, some projects may require enriching organization-owned data with other third-party data sources. Today, there are many data marketplaces where diverse datasets can be subscribed to or, in some cases, accessed for free. When discussing data sources, both internal and external datasets should be considered.

The team that has been included in the workshop should include people who understand the data sources required for the project. Some of the information that the data engineer needs to gather about these data sources includes the following:

- Details about the source system containing data (is the data in a database, in files on a server, existing files on Amazon S3, coming from a streaming source, and so on)?
- If this data is internal data, who is the owner of the source system within the business? Who is the owner of the data?
- What frequency does the data need to be ingested on (continuous streaming/replication, loading data every few hours, or loading data once a day)?
- Optionally, discuss some potential tools that could be used for data ingestion.

- What is the raw/ingested format of the data (CSV, JSON, a native database format, and so on)?
- Does the data source contain PII or other data types that are subject to governance controls? If so, what controls need to be put in place to protect the data?

As information is discovered, it can be captured on the whiteboard, as illustrated in the following diagram:

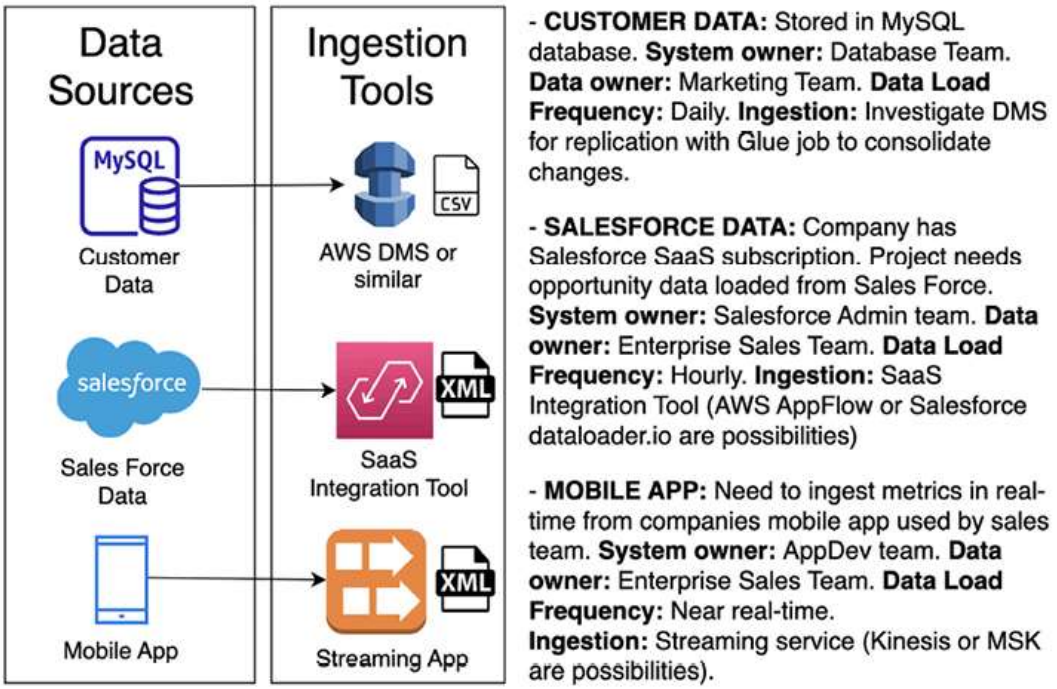


Figure 5.3: Whiteboarding data sources and data ingestion

During the whiteboarding process, additional notes should be captured to provide more context or detail about the requirements. These can be captured directly on the whiteboard or captured separately.

In this example, we have identified three different data sources – customer data from a MySQL database, opportunity information from Salesforce, and near-real-time sales metrics from the organization's mobile application. We have also identified the following:

- The IT team that owns each source system and the business team that owns the data
- The velocity of ingesting the data (how often each data source needs to be ingested)
- Potential services that can be used to ingest the data

When discussing ingestion tools, it may be worthwhile to capture potential tools if you have a good idea of which tool may be suitable. However, the objective of this session is not to come up with a final architecture and decision on all technical components.

Additional sessions (as discussed later in this book) will be used to thoroughly evaluate potential toolsets against requirements and should be done in close consultation with source system owners.

During this whiteboarding session, we have worked backward, first identifying the data consumers, and then the data sources we plan to use. At this point, we can move on to the next phase of whiteboarding, which is to examine some of the data transformations that we plan to use to optimize the data for analytics.

Identifying data transformations and optimizations

In a typical data analytics project, we ingest data from multiple data sources and then perform transforms on those datasets to optimize them for the required analytics.

In *Chapter 7*, *Transforming Data to Optimize for Analytics,* we will do a deeper dive into typical transformations and optimizations, but we will provide a high-level overview of the most common transformations here.

File format optimizations

CSV, XML, JSON, and other types of plaintext files are commonly used to store structured and semi-structured data. These file formats are useful when manually exploring data, but there are much better, binary-based file formats to use for computer-based analytics. A common binary format that is optimized for read-heavy analytics (such as by compressing data and adding in useful metadata to optimize data reads) is the *Apache Parquet* format. A common transformation is to convert plaintext files into an optimized format, such as Apache Parquet.

Data standardization

When building out a pipeline, we often load data from multiple different data sources, and each of those data sources may have different naming conventions that refer to the same item. For example, a field containing someone's birth date may be called *DOB*, *dateOfBirth*, *birth_date*, and so on. The format of the birth date may also be stored as *mm/dd/yy*, *dd/mm/yyyy*, or in a multitude of other formats.

One of the tasks we may want to do when optimizing data for analytics is to standardize column names, types, and formats. By having a corporate-wide analytic program, standard definitions can be created and adopted across all analytic projects in an organization.

Data quality checks

Another aspect of data transformation may be the process of verifying data quality and alerting on any ingested data that does not meet the expected quality standards.

Data partitioning

A common optimization strategy for analytics is to partition data, grouping it at the physical storage layer by a field that is often used in queries. For example, if data is often queried by a date range, then it

can be partitioned by a date field. If storing sales data, for example, all the sales transactions for a specific month would be stored in the same Amazon S3 prefix (which is very much like a directory). When a query is run that selects all the data for a specific day, the analytic engine only needs to read the data in the directory that stores data for the relevant month.

Data denormalization

In traditional relational database systems, data is normalized, meaning that each table contains information on a specific focused topic, and associated, or related, information is contained in a separate table. The tables can then be linked through the use of foreign keys.

In data lakes, combining data from multiple tables into a single table can often improve query performance. Data denormalization takes two (or more) tables and creates a new table, with data from both tables.

Data cataloging

Another important component that we should include in the transformation section of our pipeline architecture is the process of cataloging the dataset. During this process, we ensure that all the datasets in the data lake are added to the technical data catalog, and additional business metadata can be added to the business data catalog.

Now that we have an understanding of the common types of transformations, let's look at how this is applied to our whiteboarding session.

Whiteboarding data transformation

For the whiteboarding session, we do not need to determine all the details of the required transformations, but it is useful to agree on the main transformations for the high-level pipeline design.

Some of the information that the data engineer needs to gather about expected data transformations during the whiteboarding session includes the following:

- Is there an existing set of standardized column name definitions and formats that can be referenced? If not, who will be responsible for creating these standard definitions?
- What additional business metadata should be captured for datasets? For example, data owner, column descriptions, cost allocation tags, data sensitivity, and so on.
- What format should optimized files be stored in? Apache Parquet is a common format, but you need to validate that the tools used by the data consumers can work with files in Apache Parquet format.
- Is there an obvious field that data should be partitioned by?
- Are other required data transformations obvious at this point? For example, if you ingest data from a relational database, should the data be denormalized?
- What data transformation engines/skills does the team have? For example, does the team have experience creating Spark jobs using PySpark?

As information is discovered, it can be captured on the whiteboard, as illustrated in the following diagram:

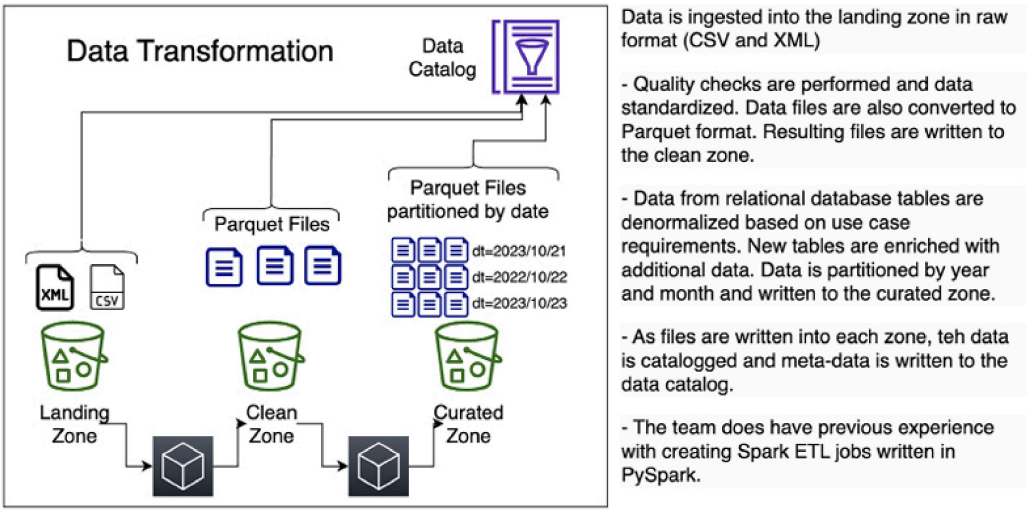


Figure 5.4: Whiteboarding data transformation

In this example, we create a data lake with three zones – the landing zone, the clean zone, and the curated zone (as previously discussed in *Chapter 2*, *Data Management Architectures for Analytics*):

- Raw files are ingested into the landing zone and will be in plaintext formats such as CSV and XML. When the files are ingested, information about the files is captured in the data catalog, along with additional business metadata (data owner, data sensitivity, and so on).
- We haven't identified a specific data transformation engine at this point, but we did capture a note indicating that the team has previous experience in creating Spark ETL jobs using PySpark. This means that AWS Glue may be a good solution for data transformation, but we will do further validation of this at a later stage.
- As part of our pipeline, we will have a process to run data quality checks on the data in the landing zone. If the quality checks pass, we will standardize the data (uniform column names and data types) and convert the files into Apache Parquet format, writing out the new files in the clean zone. Again, we will add the newly written-out files to our data catalog, including relevant business metadata.
- Another piece of our pipeline will now perform additional transformations on the data, as per the specific use case requirements. For example, data from a relational database will be denormalized, and tables can be enriched with additional data. We will write out the transformed data to the curated zone, partitioning the files by date as they are written out. Again, we will add the newly written-out files to our data catalog, including the relevant business metadata.

It's important to remember that the goal of this session is not to work out all the technical details but, rather, to create a high-level overview of the pipeline. In the preceding diagram, we did not specify that AWS

Glue will be the transformation engine. We know that AWS Glue may be a good fit, but it's not important to make that decision now.

We have indicated a potential partitioning strategy based on date, but this is also something that will need further validation. To determine the best partitioning strategy, you need a good understanding of the queries that will be run against the dataset. In this whiteboarding session, it is unlikely that there will be time to get into those details, but after the initial discussion, this appeared to be a good way to partition data, so we have included it.

Having determined transformations for our data, we will move on to the last step of the whiteboarding process, which is determining whether we will require any data marts.

Loading data into data marts

Many tools can work directly with data in the data lake, as we covered in *Chapter 3*, *The AWS Data Engineer's Toolkit*. These include tools for ad hoc SQL queries (Amazon Athena), data processing tools (such as Amazon EMR and AWS Glue), and even specialized machine learning tools (such as Amazon SageMaker).

These tools read data directly from Amazon S3, but there are times when a use case may require much lower latency and higher performance reads of the data. Alternatively, there may be times when the use of highly structured schemas may best meet the analytic requirements of the use case. In these cases, loading data from the data lake into a data mart makes sense.

In analytic environments, a data mart is most often a data warehouse system (such as Amazon Redshift or Snowflake), but it could also be a relational database system (such as Amazon RDS for MySQL), depending on the use case's requirements. In either case, the system will have local storage (often high-speed flash drives) and local compute power, offering the best performance when you need to query across large

datasets and specifically, when queries require joining across many tables.

As part of the whiteboarding session, you should spend some time discussing whether a data mart may be best suited to load a subset of data. For example, if you expect a large number of users to use your BI tool (for data visualizations), you may spend some time discussing which data will be used the most by these teams. You could then include a note in your whiteboarding session, about loading a subset of the data into a data warehouse system and connecting the data visualization tool to the data warehouse.

Wrapping up the whiteboarding session

After completing the whiteboarding session, you should have a high-level overview architecture that illustrates the main components of the pipeline that you plan to build. At this point, there will still be a lot of questions that have been left unanswered, and there will not be a lot of specific details. However, the high-level architecture should be enough to get broad agreement from stakeholders on the proposed plans for the project. It should have also provided you with enough information that you can start on a detailed design and set up follow-up sessions as required.

Some of the information that you should have after the session includes the following:

- A good understanding of who the data consumers for this project will be
- For each category of data consumer, a good idea of what type of tools they would use to access the data (SQL, visualization tools, and so on)
- An understanding of the internal and external data sources that will be used

- For each data source, an understanding of the requirements for data ingestion frequency (daily, hourly, or near-real-time streaming, for example)
- For each data source, a list of who owns the data and the source system containing the data
- A high-level understanding of likely data transformations
- An understanding of whether loading a subset of data into a data warehouse or other data marts may be required

After the session, you should create a final high-level architecture diagram and include notes from the meeting. These notes should be distributed to all participants to request their approval and agreement on moving forward with the project, based on the draft architecture.

Once an agreement has been reached on the high-level approach, additional sessions will be needed with the different teams to capture additional details and fully examine the requirements.

The final high-level architecture diagram, based on the scenario we have looked at in this chapter, may look as follows:

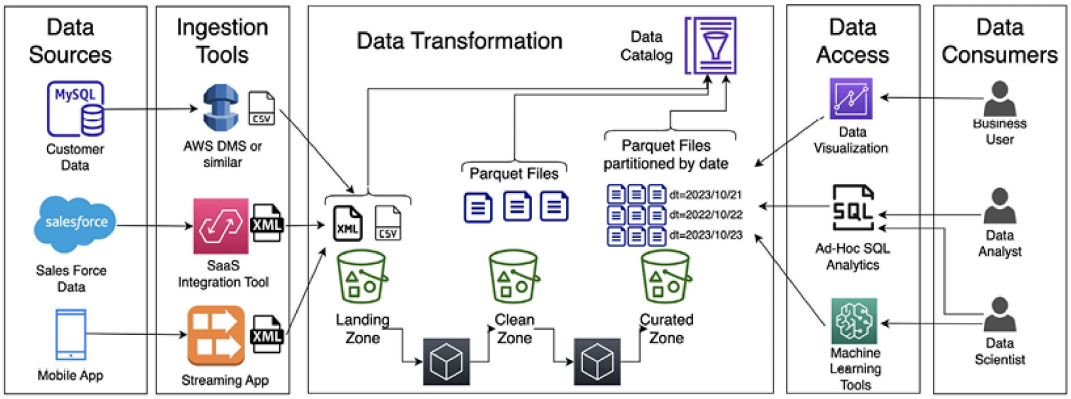
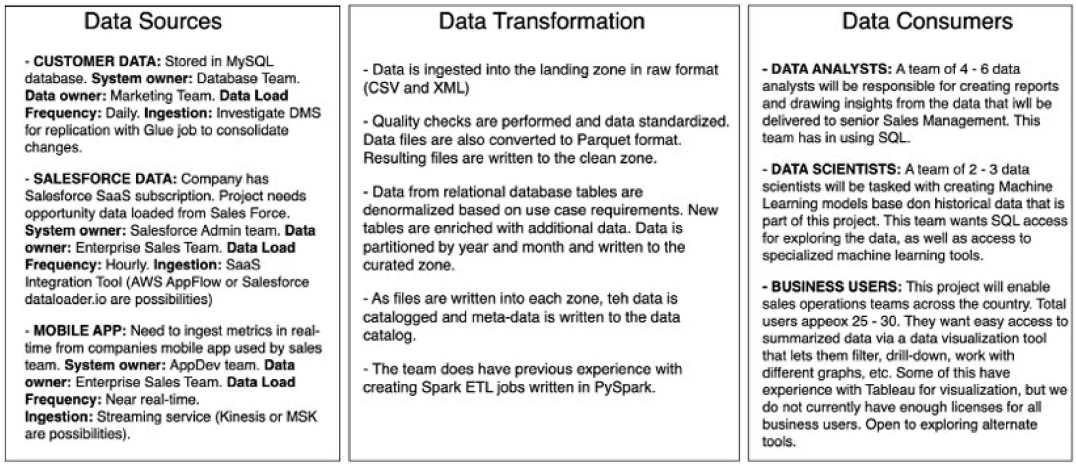


Figure 5.5: High-level architecture whiteboard

In addition to our high-level architecture diagram on the whiteboard, we will also capture associated notes about the various architectural

components during the discussion. The notes that were captured for the scenario we discussed in this chapter may look like this:



Data Sources

- **CUSTOMER DATA:** Stored in MySQL database. **System owner:** Database Team. **Data owner:** Marketing Team. **Data Load Frequency:** Daily. **Ingestion:** Investigate DMS for replication with Glue job to consolidate changes.

- **SALESFORCE DATA:** Company has Salesforce SaaS subscription. Project needs opportunity data loaded from Sales Force. **System owner:** Salesforce Admin team. **Data owner:** Enterprise Sales Team. **Data Load Frequency:** Hourly. **Ingestion:** SaaS Integration Tool (AWS AppFlow or Salesforce dataloader.io are possibilities)

- **MOBILE APP:** Need to ingest metrics in real-time from companies mobile app used by sales team. **System owner:** AppDev team. **Data owner:** Enterprise Sales Team. **Data Load Frequency:** Near real-time. **Ingestion:** Streaming service (Kinesis or MSK are possibilities).

Data Transformation

- Data is ingested into the landing zone in raw format (CSV and XML)

- Quality checks are performed and data standardized. Data files are also converted to Parquet format. Resulting files are written to the clean zone.

- Data from relational database tables are denormalized based on use case requirements. New tables are enriched with additional data. Data is partitioned by year and month and written to the curated zone.

- As files are written into each zone, teh data is catalogged and meta-data is written to the data catalog.

- The team does have previous experience with creating Spark ETL jobs written in PySpark.

Data Consumers

- **DATA ANALYSTS:** A team of 4 - 6 data analysts will be responsible for creating reports and drawing insights from the data that iwll be delivered to senior Sales Management. This team has in using SQL.

- **DATA SCIENTISTS:** A team of 2 - 3 data scientists will be tasked with creating Machine Learning models base don historical data that is part of this project. This team wants SQL access for exploring the data, as well as access to specialized machine learning tools.

- **BUSINESS USERS:** This project will enable sales operations teams across the country. Total users appeox 25 - 30. They want easy access to summarized data via a data visualization tool that lets them filter, drill-down, work with different graphs, etc. Some of this have experience with Tableau for visualization, but we do not currently have enough licenses for all business users. Open to exploring alternate tools.

Figure 5.6: Notes associated with our whiteboarding

Now that you understand the theory of how to conduct a whiteboarding session, it's time to get some practical hands-on experience. This next section provides details about a fictional whiteboarding session and allows you to practice your whiteboarding skills.

Hands-on – architecting a sample pipeline

For the hands-on portion of this chapter, you will review the detailed notes from a whiteboarding session held for the fictional company GP Widgets Inc. As you go through the notes, you should create a whiteboard architecture, either on an actual whiteboard or on a piece of poster board. Alternatively, you can create the whiteboard using a free online design tool, such as the one available at **http://diagrams.net**.

As a starting point for your whiteboarding session, you can use the following template. You can recreate this on your whiteboard or poster board, or you can access the diagrams.net/Draw.IO template for this via the GitHub site of this book at **https://github.com/PacktPublishing/Data-Engineering-with-AWS-**

2nd-edition/blob/main/Chapter05/Data-Engineering-Whiteboard-Template.drawio.

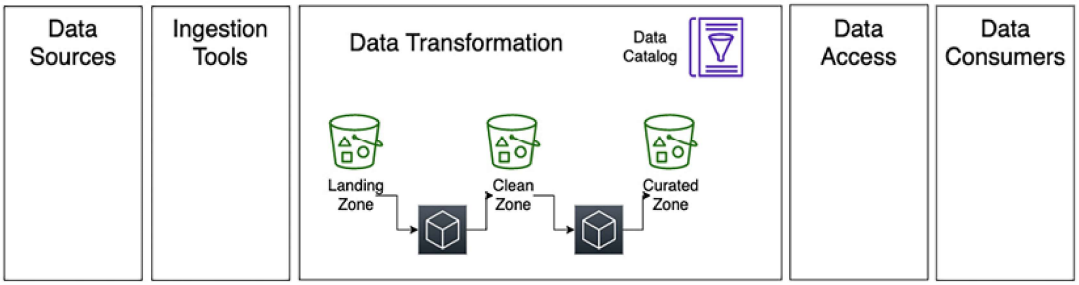


Figure 5.7: Generic whiteboarding template

Note that the three zones included in the template (landing zone, clean zone, and curated zone) are commonly used for data lakes. However, some data lakes may only have two zones, while others may have four or more zones. The number of zones is not a hard-and-fast rule but, rather, is based on the requirements of the use case you are designing for.

As you go through the meeting notes, fill out the relevant sections of the template. In addition to drawing the components and flow for the pipeline, you should also capture notes relating to the whiteboard components, as per the example in *Figure 5.6*. At the end of this chapter, you can compare the whiteboard you have created with the one that the data engineer lead for GP Widgets has created.

By completing this exercise, you will gain hands-on experience in whiteboarding a data pipeline and learn how to identify the key points about data consumers, data sources, and required transformations.

Detailed notes from the project “Bright Light” whiteboarding meeting of GP Widgets, Inc

Here is a list of attendees participating in the meeting:

- Ronna Parish, VP of marketing
- Chris Taylor, VP of enterprise sales
- Terry Winship, data analytics team manager
- James Dadd, data science team manager
- Owen McClave, database team manager
- Natalie Rabinovich, web server team manager
- Shilpa Mehta, data engineer lead

Meeting notes

Shilpa started the meeting by asking everyone to introduce themselves and then provided a summary of the meeting objectives:

- Plan out a high-level architecture for a new project to bring improved analytics to the marketing teams. During the discussion, Shilpa will whiteboard a high-level architecture.
- They reinforced that not all the technical details would be worked out in this meeting, but looking for agreement from all the stakeholders with a high-level approach and design is critical.
- It's already agreed that the project will be built in the AWS cloud.

Shilpa asked Ronna (marketing manager) to provide an overview of the marketing team requirements for project *Bright Light*:

- Project Bright Light is intended *to improve the visibility that the marketing team has into real-time customer behavior*, as well as **longer-term** *customer trends*, through the use of data analytics.
- The marketing team wants to give **marketing specialists** real-time insights into interactions on the company's e-commerce website. Some examples of these **visualizations** include a heatmap to show website activity in different geographic locations, redemptions of coupons by product category, top ad campaign referrals, and spend by zip code in the previous period versus the current period.
- All visualizations should enable a user to select a custom period, filter on custom fields, and drill down from summary information to

detailed information. Data should be *refreshed on at least an hourly basis, but more often would be better*.

The **data analyst team** supporting the marketing department will run more complex investigations of current and historical trends:

- Identify the top 10% of customers over the past *x* days by spend, and identify their top product categories for use in marketing promotions.
- Determine the average time a customer spends on the website, as well as the number of products they browse versus the number of products they purchase.
- Identify the top returned products over the past day/month/week.
- Compare the sales by zip code this month versus sales by zip code in the same month 1 year ago, grouped by product category.
- For each customer, keep a running total of the number of widgets purchased (grouped by category), the average spend per sale, the average spend per month, the number of coupons applied, and the total coupon value.

We have tasked our **data science team** with building a machine learning model that can predict a widget's popularity based on the weather in a specific location:

- Research highlights how weather can impact e-commerce sales and the sales of specific products – for example, the types of widgets that customers are likely to buy on a sunny day compared to a cold and rainy day.
- The marketing team wants to target our advertising campaigns and optimize our ad spend and real-time marketing campaigns, based on the current and forecasted weather in a certain zip code.
- We regularly add new widgets to our catalog, so the model must be updated at least once a day, based on the latest weather and sales information.

- In addition to helping with marketing, the manufacturing and logistics teams have expressed interest in this model to help optimize logistics and inventory for different warehouses around the country, based on 7-day weather forecasts.

James Dadd (data science team manager) spoke about some of the requirements for his team:

- They would need **ad hoc SQL access** to **weather**, **website clickstream logs**, and **sales** data for at least the last year.
- They have determined that a provider on *AWS Data Exchange* offers historical and forecast weather information for all US zip codes. There is a subscription charge for this data, and the marketing team is working on allocating a budget for this. Data would be delivered **daily via AWS Data Exchange in the CSV format**.
- James indicated he had not had a chance to speak with the database and website admin teams about getting access to their data yet.
- The team currently uses **SparkML** for a lot of their machine learning projects, but they are **interested in cloud-based tools** that may help them speed up delivery and collaboration for their machine learning products. They also **use SQL queries to explore datasets**.

Terry Winship (data analytics team manager) indicated that her team specializes in **using SQL** to gain complex insights from data:

- Based on her initial analysis, her team would need access to the **customer, product, returns, and sales databases**, as well as **clickstream data from the web server logs**.
- Her team has experience in working with on-premises data warehouses and databases. She has been reading up about data lakes, and as long as the team can use SQL to query the data, they are open to using different technologies.
- She also has specialists on her team that **can create visualizations** for the marketing team to use. This team primarily has experience

with using **Tableau** for visualizations, but the marketing team **does not have licenses to use Tableau**. There would be a learning curve if a different visualization tool were used, but they are **open to exploring other options**.
- Terry indicated that a **daily update of data from the databases** should be sufficient for what they need, but that they would need **near-real-time streaming for the clickstream web server log files** so that they could provide the most up-to-date reports and visualizations.

Shilpa asked Owen McClave (database team manager) to provide an overview of the database sources that the data science and data analytics teams would need:

- Owen indicated that all their **databases run on-premises** and run **Microsoft SQL Server 2016, Enterprise Edition**.
- Owen said he doesn't know much about AWS and has some concerns about providing administrative access for his databases to the cloud, since he does not believe the cloud is secure. However, he said that, ultimately, it is up to the data owners to approve whether the data can be copied to the cloud. If approved, he will work with the cloud team to enable data syncing in the cloud, so long as there is no negative impact on his databases.
- Chris Taylor (VP of sales) is the data owner of the databases that have been discussed today (customer, product, returns, and sales data).

Shilpa asked Chris Taylor (VP of sales) to provide input on the use of sales data for the project:

- Chris indicated that this analytics project has executive sponsorship from senior management and visibility by the board of directors.
- He indicated that sales data can be stored in the cloud, so long as the security team reviews and approves it.

Shilpa indicated that AWS has a tool called **Database Migration Service**, which can be used to replicate data from a relational database, such as SQL Server 2016, to Amazon S3 cloud storage. She said she would set up a meeting with Owen to discuss the requirements for this tool in more detail at a later point, as there are also various other options.

Shilpa requested that Natalie Rabinovich (web server team manager) provide more information on the web server log files that will be important for this project:

- Natalie indicated that they currently run the e-commerce website on-premises on **Linux servers running Apache HTTP Server**.
- A load balancer is used to distribute traffic between **four different web servers**. Each server stores its clickstream web server logs locally.
- The log files are **plaintext files in the Apache web log format**.
- Shilpa indicated that AWS has an agent called the **Kinesis Agent**, which can be used to read the log files and stream their contents to the AWS cloud. She queried whether it would be possible to install this agent on the Apache web servers.
- Natalie indicated that it should be fine, but they would need more details and to test it in a development environment before installing it on the production servers.
- Shilpa asked who the data owner was. Natalie indicated that the marketing team owns the web server clickstream logs from a business perspective.

Shilpa led a discussion on the potential data transformations that may be required on the data that is ingested for this project:

- Based on the description from James, it appears that data should be made available daily in the CSV format and can be loaded directly into the raw zone of the data lake.

- Using a tool such as Amazon DMS, we can load data from the databases into the raw zone of the data lake in the Parquet format.
- The Kinesis Agent can convert the Apache HTTP Server log files into the JSON format, streaming them to Kinesis Firehose. Firehose can then perform basic validation of the log (using Lambda), convert it into the Parquet format, and write directly to the clean zone.
- Shilpa indicated that an **initial transformation** could perform basic data **quality checks on incoming database data**, add contextual information as new columns (such as ingestion time), and then write the file **to the clean zone** of the data lake.
- Shilpa explained that partitioning datasets helps optimize both the performance and cost of queries. She indicated that partitions should be based on how data is queried and led a discussion on the topic. After some discussion, it was agreed that partitioning the database and weather by day (yyyyy/mm/dd) and web server logs by hour (yyyy/mm/dd/hh) may be a good partitioning strategy, but this would be investigated further and confirmed in future discussions.
- A **second transformation** could then be run against the data in the clean zone, performing tasks such as **denormalizing the relational datasets, joining tables to optimized tables, enriching data with weather data**, or performing any other required business logic. This optimized data would be **written to the curated zone** of the data lake. AWS Glue or Amazon EMR could potentially be used to perform these transformations.
- As each dataset is loaded, and then the transformed data is written out to the next zone, the **data will be added to the Glue Data Catalog**. Once data has been added to the data catalog, authorized users will be able to query the data using SQL queries, enabled by a tool such as Amazon Athena. **Additional metadata could be added** at this point, including items such as data owner, source system, data sensitivity level, and so on.
- Shilpa indicated that she will arrange future meetings with the various teams to discuss the business metadata that must be added.

- Shilpa wrapped up the meeting with a brief overview of the white-board, and she committed to providing a copy of the whiteboard architecture and notes to all attendees for further review and comment. She also indicated that she would schedule additional meetings with smaller groups of people to dive deep into specific aspects of the proposed architecture, providing additional validation.

Shilpa used a whiteboard in the meeting room to sketch out a rough architecture, and then after the meeting, she created the following diagram to show the architecture:

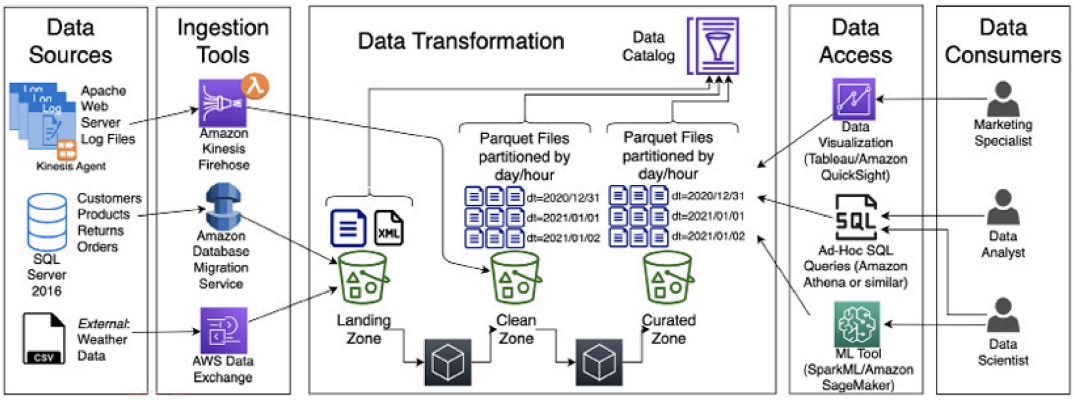


Figure 5.8: Completed whiteboard architecture for project Bright Light

Shilpa also added some notes to go with the whiteboard, sending out both the whiteboard architecture and the notes to the meeting attendees:

Data Sources

- **Apache Web Server Log Files:** From 4 Apache web servers. **System Owner:** Natalie Rabinovich. **Data Owner:** Marketing. **Ingestion:** Could use Kinesis Agent to transform to JSON and send to Kinesis Firehose. Firehose does validation (using Lambda function) and transforms to Parquet format. Could write direct to clean zone, partitioned by day (yyyy/mm/dd).

- **Databases:** Customers, Products, Returns, Orders on SQL Server 2016 Enterprise Edition. **System Owner:** Owen McClave. **Data Owner:** Sales Team. Potentially use Amazon DMS to replicate to Amazon S3 raw zone in Parquet format.

- **Weather Data:** External data source available via subscription. **Data Owner:** Marketing. **Ingestion:** Available from AWS Data Exchange marketplace. Lambda function can load data into Amazon S3 raw zone when available.

Data Transformation

- **Raw Zone:** Database and weather data replicated into raw zone. When files ingested triggers Lambda function to perform data quality checks and then loads into Clean Zone partitioned by yyyy/mm/dd.

- **Clean Zone:** Web server log files loaded directly into clean zone after Kinesis Firehose uses a Lambda function to perform data quality checks. Firehose configured to write to clean zone partitioned by yyyy/mm/dd. Database and weather files loaded from raw zone after data quality checks, and partition by yyyy/mm/dd.

- **Curated Zone:** Database files denormalized, enriched (with weather data potentially), other business logic added. Partitioned by either day (databases, weather) or hour (web server log files)

Data Consumers

- **Marketing Specialists:** Want to use business intelligence (visualization) tool to view up-to-date website analytics (ad-campaign referrals, coupon redemption, heatmap showing activity by geographic location). Refresh on at least hourly basis. Analytics team generally uses Tableau, but marketing team does not have licenses. Open to other BI tools.

- **Data Analysts:** Responsible for creating reports and insights using SQL queries. Database and weather data could be refreshed daily, but they would need web server clickstream log files refreshed at least hourly.

- **Data Scientists:** Need ad-hoc SQL access to databases, weather and web server log files. They currently use SparkML on-premises, but open to new cloud based tools that may make speed up delivery and collaboration for their machine learning products.

Figure 5.9: Completed whiteboard notes for project Bright Light

Compare the whiteboard you created to the whiteboard created by Shilpa, and note the differences. Are there things that Shilpa missed on her whiteboard or notes? Are there things that you missed on your whiteboard or notes?

The exercises in this chapter allowed you to get hands-on with data architecting and whiteboarding. We will wrap up this chapter by providing a summary, and then do a deeper dive into the topics of data ingestion, transformation, and data consumption in the next few chapters.

Summary

In this chapter, we reviewed an approach to developing data engineering pipelines by identifying a limited-scope project, and then whiteboarding a high-level architecture diagram. We looked at how we could have a workshop, in conjunction with relevant stakeholders in an organization, to discuss requirements and plan the initial architecture.

We approached this task by working backward. We started by identifying who the data consumers of the project would be and learning about their requirements. Then, we looked at which data sources could be used to provide the required data and how those data sources could be ingested. We then reviewed, at a high level, some of the data transformations that would be required for the project to optimize the data for analytics.

In the next chapter, we will take a deeper dive into AWS services to ingest batch and streaming data, learning more about how to select the best tool for our data engineering pipeline.

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