1. **Identify data sources**

The first step in building a data lineage is to identify all of the data sources that are used within the organization. This includes structured and unstructured data, and may include data stored in databases like Postgres, data warehouses like Snowflake, brokers like Kafka, and object storage facilities like S3. It's important to understand the different types of data that are being used and where they are coming from.

1. **Ingest data and metadata**

Once the data sources have been identified, the next step is to ingest metadata about the data into the data lineage tool.

Very likely are your company, there will not be a single, consolidated place where all data resides that can be leveraged to derive the data lineage. Rather, instead, you might have to

Our data ingestion approach, in a nutshell, is classified broadly into two buckets — push or pull. Today, we are operating using a pull-heavy model. In this model, we scan system logs and metadata generated by various compute engines to collect corresponding lineage data. For example, we leverage [inviso](https://medium.com/netflix-techblog/inviso-visualizing-hadoop-performance-f834175c6df8) to list pig jobs and then [lipstick](https://medium.com/netflix-techblog/introducing-lipstick-on-a-pache-pig-f17e0a4e0c89) to fetch tables and columns from these pig scripts. For spark compute engine, we leverage spark plan information and for Snowflake, admin tables capture the same information. In addition, we derive lineage information from scheduled ETL jobs by extracting workflow definitions and runtime metadata using [Meson](https://medium.com/netflix-techblog/meson-workflow-orchestration-for-netflix-recommendations-fc932625c1d9) scheduler APIs.

In the push model paradigm, various platform tools such as the data transportation layer, reporting tools, and Presto will publish lineage events to a set of lineage related Kafka topics, therefore, making data ingestion relatively easy to scale improving scalability for the data lineage system.  
  
we started with a generic data model, and a simple metadata ingestion pipeline that pulls the information from various data stores and processes across Shopify. The metadata extractor also builds the dependency graph for our lineage feature. Once processed, the information is stored in Elasticsearch indexes, and GraphQL APIs expose the data via an Apollo client to the Artifact UI.

1. **Construct lineage:**

After the metadata has been ingested, it can be used to construct the data lineage. The data lineage tool will use the metadata to trace the flow of data through the organization and identify how it is transformed and used. This step may involve performing additional data enrichment, for example, pulling cluster and scheduler metadata from other sources.

1. **Make lineage consumable:**

Once the data lineage has been constructed, it needs to be made consumable by the organization. This means that the lineage information needs to be presented in a way that is easy to understand and navigate. This can be achieved by using a variety of visualization and reporting tools.

The data lineage can be visualized in a variety of ways, such as flow diagrams, timelines, and heatmaps. Visualizing the data lineage can help to identify data quality issues, data duplication, and data lineage gaps.

SLA service relies on the job dependencies defined in ETL workflows to alert on potential SLA misses. This service also proactively alerts on any potential delays in few critical reports due to any job delays or failures anywhere upstream to it.

Owners of datasets can use a “notify” button which presents them with information about their downstream consumers, along with a call-to-action and other fields. Users use this functionality to inform their downstream owners (which includes dashboard creators, dataset owners) of upcoming changes in our data warehouse via Slack, thereby allowing consumers of downstream datasets to keep themselves well informed of upcoming changes. This functionality is extensively used in Data Retention program execution, planned dataset deprecation, and many other use-cases.

1. **Evangelize lineage:**

Finally, it's important to evangelize the data lineage within the organization. This means that the data lineage needs to be communicated to the different teams and stakeholders within the organization. This can be done through training sessions, workshops, and presentations. By communicating the data lineage, the organization can ensure that everyone understands the importance of data governance and how to use the data lineage tool.

1. **Continuously Monitor and Improve:**

The data lineage should be continuously monitored and improved. This means keeping the data lineage up to date as new data sources and data flows are added, and ensuring that the data lineage is accurate and complete. This can be done by regularly reviewing the data lineage and making updates as necessary. Additionally, regular audits should be performed to ensure that the data lineage is accurate and that any data quality issues are identified and addressed.

<https://slack.engineering/data-lineage-at-slack/>

<https://shopify.engineering/solving-data-discovery-challenges-shopify>

<https://eng.lyft.com/open-sourcing-amundsen-a-data-discovery-and-metadata-platform-2282bb436234>

<https://netflixtechblog.com/building-and-scaling-data-lineage-at-netflix-to-improve-data-infrastructure-reliability-and-1a52526a7977>

### Off-the-shelf solutions

[@Alation](https://twitter.com/Alation)

[@collibra](https://twitter.com/collibra)

[@AtlanHQ](https://twitter.com/AtlanHQ)

[@Databricks](https://twitter.com/databricks) Unity Catalog