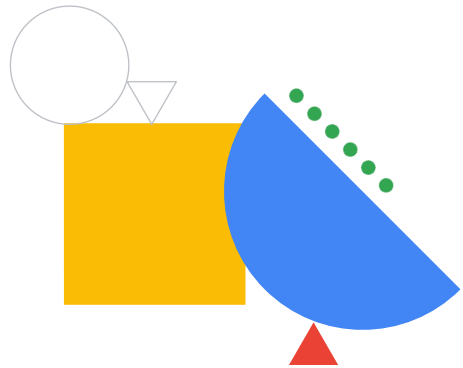


# Data Engineering for Streaming Data

## Module 2

Google Cloud Big Data and Machine Learning Fundamentals





# Introduction

01 Big Data and Machine Learning on Google Cloud ☒

02 Data Engineering for Streaming Data ☒

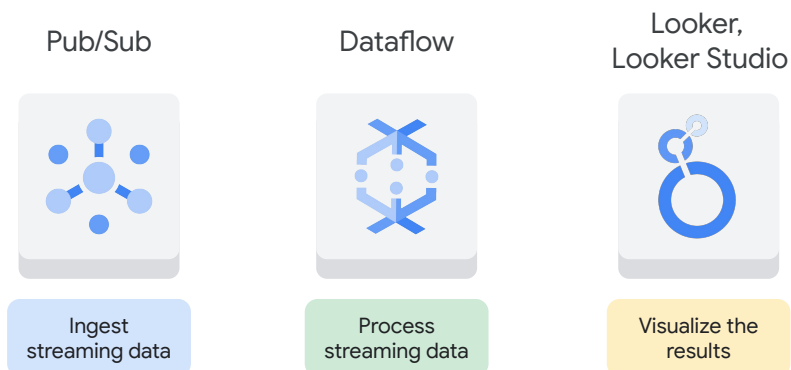
03 Big Data with BigQuery ☐

04 Machine Learning Options on Google Cloud ☐

05 The Machine Learning Workflow with Vertex AI ☐

In the previous module of this course, you learned Google Cloud architecture such as storage and compute. You were also introduced to the data and machine learning products on Google Cloud to support the data-to-AI lifecycle. In the next two modules, you'll explore Google products and technologies in data engineering. In this module, you'll explore data engineering for streaming data with the goal of building a real-time data solution with Google Cloud products and services.

# Build a real-time data solution with Google Cloud



Specifically, this includes how to:

- Ingest streaming data using **Pub/Sub**
- Process the data with **Dataflow**, and
- Visualize the results with **Looker** and **Looker Studio**.

In between data processing with Dataflow and visualization with Looker or Looker Studio, the data is normally saved and analyzed in a data warehouse such as BigQuery. You will learn the details about BigQuery in module 3.

# Agenda

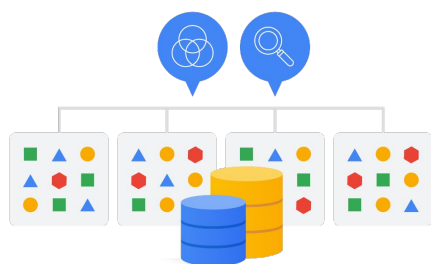


- ✓ Examine big data challenges
- ✓ Learn about message-oriented architecture
- ✓ Design and implement streaming pipelines
- ✓ Visualize data with Looker and Looker Studio
- ✓ Hands-on lab

Coming up in this module, you'll:

- Examine some of the **big data challenges** faced by today's data engineers when setting up and managing pipelines.
- Learn about **message-oriented architecture**. This includes ways to capture streaming messages globally, reliably, and at scale so they can be fed into a pipeline.
- See how to **design streaming pipelines** with Apache Beam, and then **implement** them with Dataflow.
- Explore how to **visualize** data insights on a dashboard with **Looker** and **Looker Studio**.
- Get **hands-on practice** building an end-to-end data pipeline that handles real-time data ingestion with Pub/Sub, processing with Dataflow, and visualization with Looker Studio.

# Batch processing vs. streaming data processing

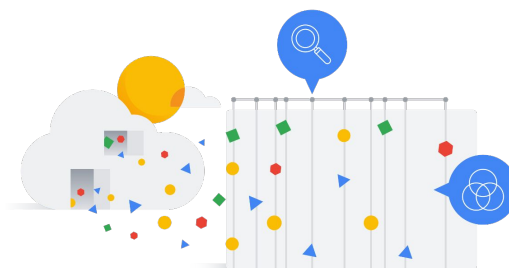


## Batch processing

Processing and analysis happens on a set of stored data.

Payroll system

Billing system



## Streaming data processing

A flow of data records generated by various data sources.

Fraud detection

Intrusion detection

Google Cloud

Before we get too far, let's take a moment to explain what streaming data is, how it differs from batch processing, and why it's important.

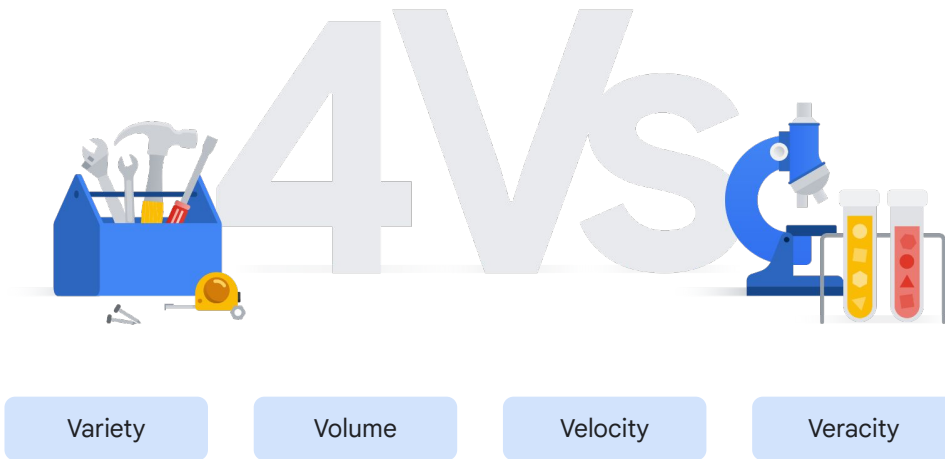
- **Batch processing** is when the processing and analysis happens on a set of stored data. An example is payroll and billing systems that have to be processed on either a weekly or monthly basis.
- **Streaming data** is a flow of data records generated by various data sources. The processing of streaming data happens as the data flows through a system. This results in the analysis and reporting of events as they happen. An example would be fraud detection or intrusion detection.  
Streaming data processing means that the data is analyzed in near-real time and that actions will be taken on the data as quickly as possible.

Modern data processing has progressed from legacy batch processing of data toward working with real-time data streams. An example of this is streaming music and movies. No longer is it necessary to download an entire movie or album to a local device. **Data streams** are a **key** part in the world of big data.



## Big data challenges

## Big Data challenges



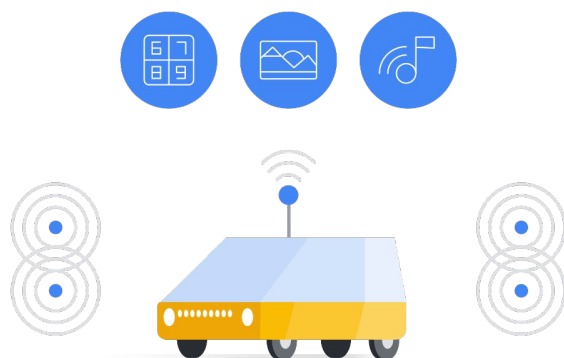
Building scalable and reliable pipelines is a core responsibility of data engineers. However, in modern organizations, data engineers and data scientists are facing four major challenges.

These are collectively known as the [4 Vs](#).

They are variety, volume, velocity, and veracity.



# 1 | Variety

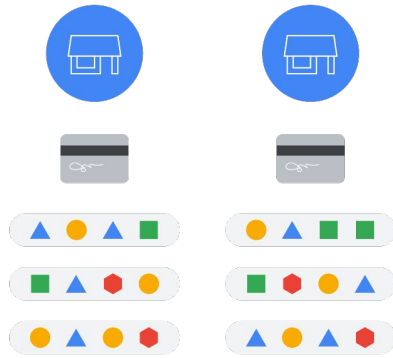


Sensors on roads around the world

Data could come in from a variety of different sources and in various formats.

First, data could come in from a **variety** of different sources and in various formats. Imagine hundreds of thousands of sensors for self-driving cars on roads around the world. The data is returned in various formats such as number, image, or even audio.

# 1 | Variety



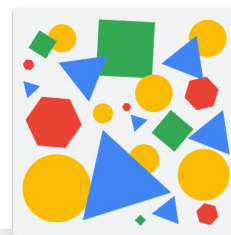
Point-of-sale data

How do we alert downstream systems of new transactions in an organized way with no duplicates?

Now consider point-of-sale data from a thousand different stores. How do we alert our downstream systems of new transactions in an organized way with no duplicates?

## 2 | Volume

Will the pipeline code and infrastructure scale, or will it grind to a halt or even crash?



Volume of data

Next, let's increase the magnitude of the challenge to handle not only an arbitrary variety of input sources, but a **volume** of data that varies from gigabytes to petabytes.

You'll need to know whether your pipeline code and infrastructure can scale with those changes or whether it will grind to a halt or even crash.

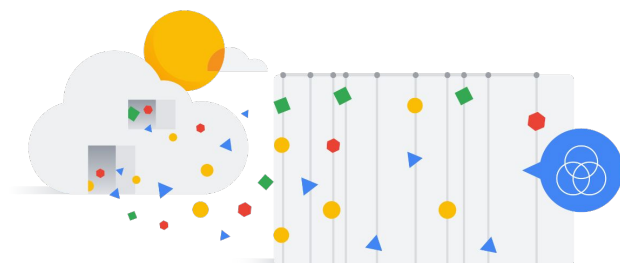
### 3 | Velocity



Data often needs to be processed in near-real time, as soon as it reaches the system.

The third challenge concerns **velocity**. Data often needs to be processed in near-real time, as soon as it reaches the system.

### 3 | Velocity



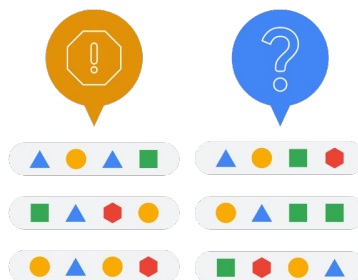
Need a way to handle data that:

- ✓ Arrives late
- ✓ Has bad data in the message
- ✓ Needs to be transformed

You'll probably also need a way to handle data that arrives late, has bad data in the message, or needs to be transformed mid-flight before it is streamed into a data warehouse.

## 4 | Veracity

Gathered data might come in with inconsistencies and uncertainties due different data types and sources.



And the fourth major challenge is **veracity**, which refers to the data quality. Because big data involves a multitude of data dimensions resulting from different data types and sources, there's a possibility that gathered data will come with some inconsistencies and uncertainties.

Challenges like these are common considerations for pipeline developers. By the end of this module, the goal is for you to better understand the tools available to help successfully build a streaming data pipeline and avoid these challenges.



## Message-oriented architecture

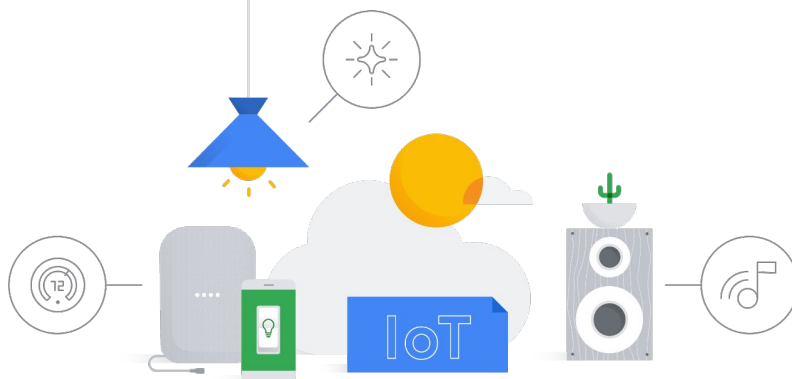
## Data ingestion is when stream data is received



One of the early stages in a data pipeline is data ingestion, which is where large amounts of streaming data are received.



## For example, IoT



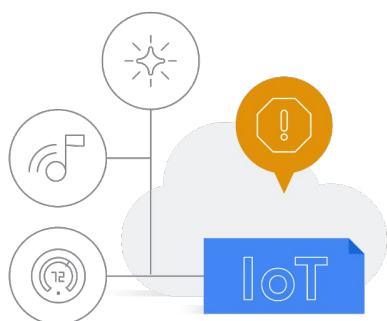
Internet of Things applications

Data, however, may not always come from a single, structured database. Instead, the data might stream from a thousand, or even a million, different events that are all happening asynchronously.

A common example of this is data from IoT, or Internet of Things, applications.

These can include sensors on taxis that send out location data every 30 seconds or temperature sensors around a data center to help optimize heating and cooling.

# Challenges to data ingestion

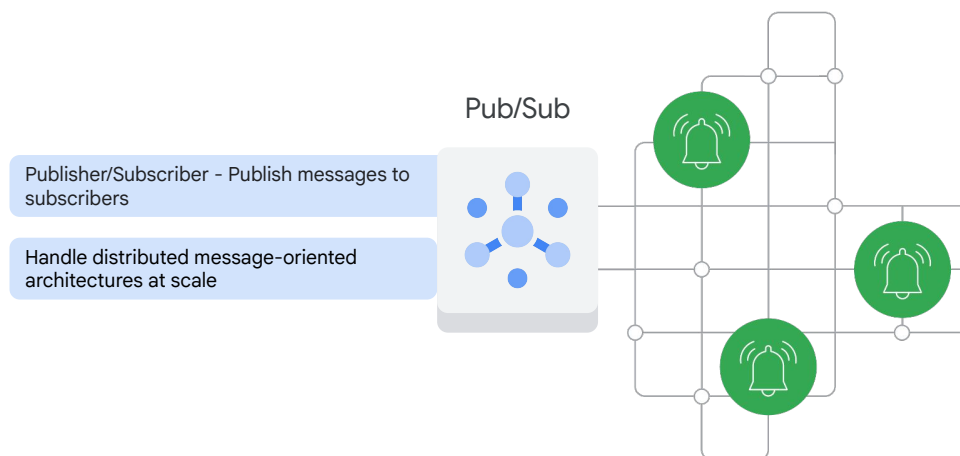


- 01 Data can be streamed from many different methods and devices
- 02 It can be hard to distribute event messages to the right subscribers
- 03 Data can arrive quickly and at high volumes
- 04 Ensuring services are reliable, secure, and perform as expected

These IoT devices present new challenges to data ingestion, which can be summarized in four points:

1. The first is that **data can be streamed from many different methods and devices**, many of which might not talk to each other and might be sending bad or delayed data.
2. The second **is that it can be hard to distribute event messages** to the right subscribers. Event messages are notifications. A method is needed to collect the streaming messages that come from IoT sensors and broadcast them to the subscribers as needed.
3. The third is that **data can arrive quickly and at high volumes**. Services must be able to support this.
4. And the fourth challenge is **ensuring services are reliable, secure, and perform as expected**.

## Pub/Sub: A distributed messaging service

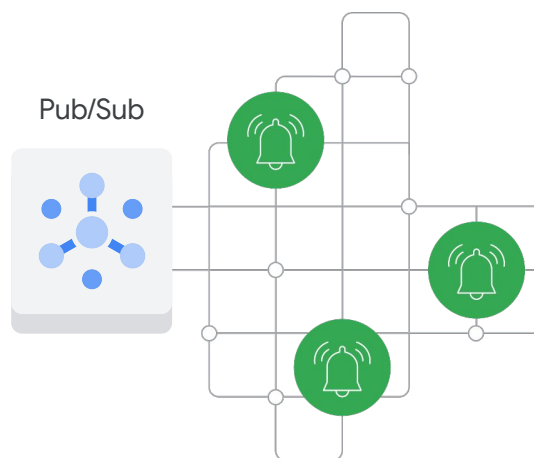


Google Cloud has a tool to handle distributed message-oriented architectures at scale, and that is **Pub/Sub**. The name is short for Publisher/Subscriber, or *publish messages to subscribers*.

Pub/Sub is a distributed messaging service that can receive messages from a variety of device streams such as gaming events, IoT devices, and application streams.

## Pub/Sub features

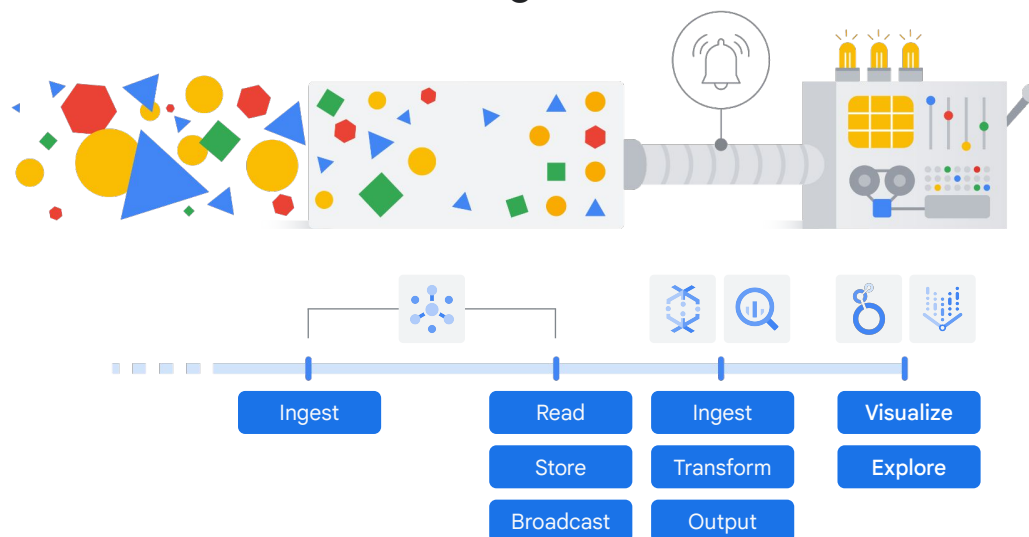
- ✓ Ensures at-least-once delivery
- ✓ No provisioning is required
- ✓ APIs are open
- ✓ Global by default
- ✓ Offers end-to-end encryption



Pub/Sub ensures at-least-once delivery of received messages to subscribing applications, with no provisioning required.

Pub/Sub's APIs are open, the service is global by default, and it offers end-to-end encryption.

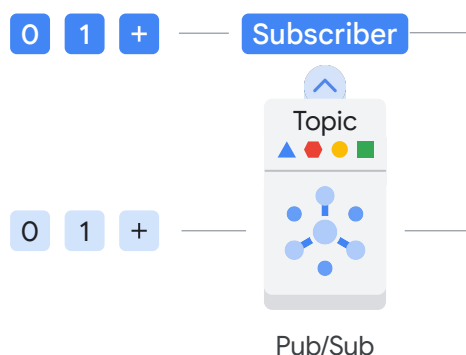
## End-to-end architecture using Pub/Sub



Let's explore the end-to-end architecture using Pub/Sub.

- Upstream source data comes in from devices all over the globe and is ingested into Pub/Sub, which is the first point of contact within the system. Pub/Sub reads, stores, and broadcasts to any subscribers of this data topic that new messages are available.
- As a subscriber of Pub/Sub, Dataflow can ingest and transform those messages in an elastic streaming pipeline and output the results into an analytics data warehouse like BigQuery.
- Finally, you can connect a data visualization tool, like Looker, to visualize and monitor the results of a pipeline, or an AI or ML tool such as Vertex AI to explore the data to uncover business insights or help with predictions.

# Topics



A topic is a central element of Pub/Sub.

Topics and subscribers are **decoupled**.

Like a radio antenna, broadcasting no matter the subscribers.

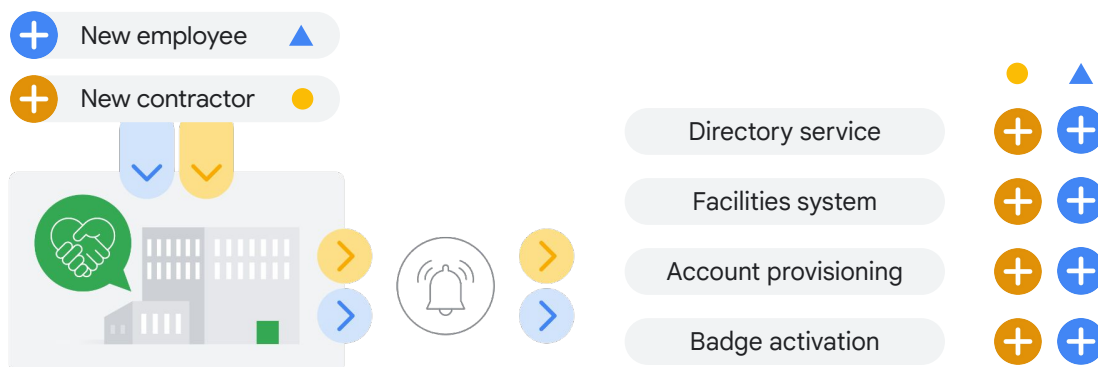
Topics and subscribers can be 0..M to 0..M relationship.

A central element of Pub/Sub is the **topic**.

You can think of a topic like a radio antenna. Whether your radio is playing music or it's turned off, the antenna itself is always there. If music is being broadcast on a frequency that nobody's listening to, the stream of music still exists. Similarly, a publisher can send data to a topic that has no subscriber to receive it. Or a subscriber can be waiting for data from a topic that isn't getting data sent to it, like listening to static from a bad radio frequency. Or you could have a fully operational pipeline where the publisher is sending data to a topic that an application is subscribed to.

That means there can be zero, one, or more publishers, and zero, one or more subscribers related to a topic. And they're completely decoupled, so they're free to break without affecting their counterparts.

## Example: Human resource topics



Human Resources Topic

Google Cloud

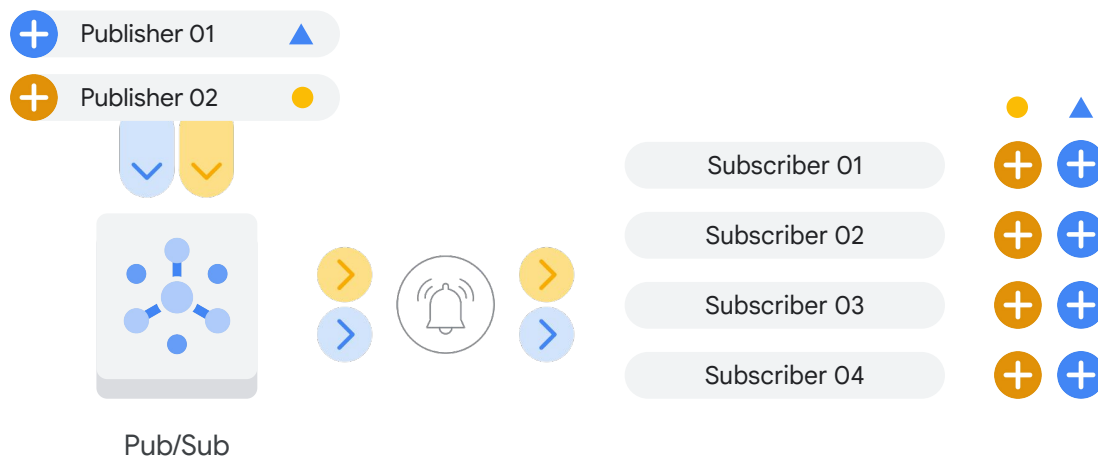
It's helpful to describe this using an example.

Say you've got a human resources topic. A new employee joins your company, and several applications across the company need to be updated. Adding a new employee can be an event that generates a notification to the other applications that are subscribed to the topic, and they'll receive the message about the new employee starting.

Now, let's assume that there are two different types of employees: a full-time employee and a contractor. Both sources of employee data could have no knowledge of the other but still publish their events saying "this employee joined" into the Pub/Sub HR topic.

After Pub/Sub receives the message, downstream applications like the directory service, facilities system, account provisioning, and badge activation systems can all listen and process their own next steps independent of one another.

## Pub/Sub ingests streaming data



Pub/Sub is a good solution to buffer changes for lightly coupled architectures, like this one, that have many different publishers and subscribers.

Pub/Sub supports many different inputs and outputs, and you can even publish a Pub/Sub event from one topic to another.

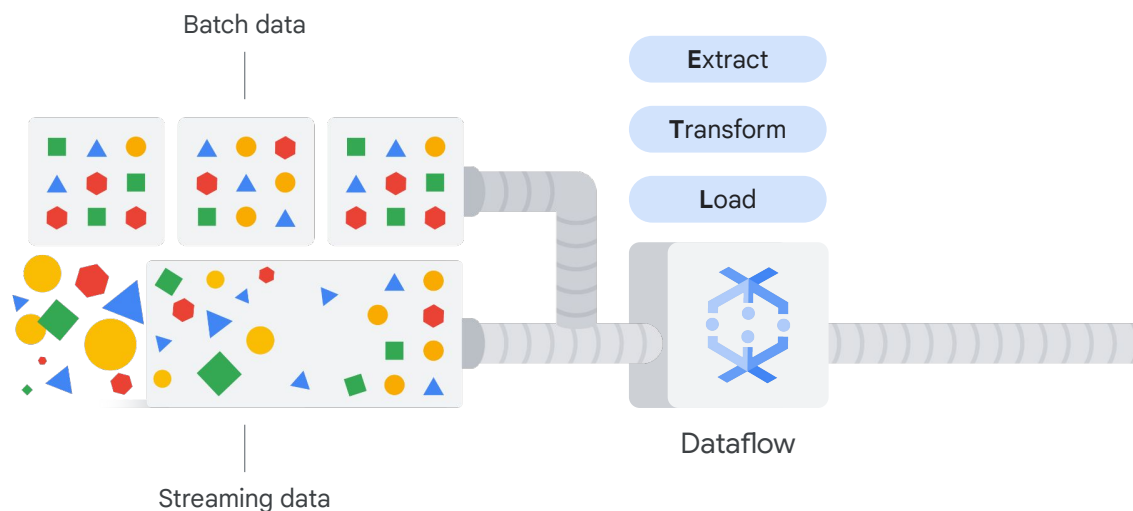
The next task is to get these messages reliably into our data warehouse, and we'll need a pipeline that can match Pub/Sub's scale and elasticity to do it.





## Designing streaming pipelines with Apache Beam

## Dataflow creates a pipeline to process data

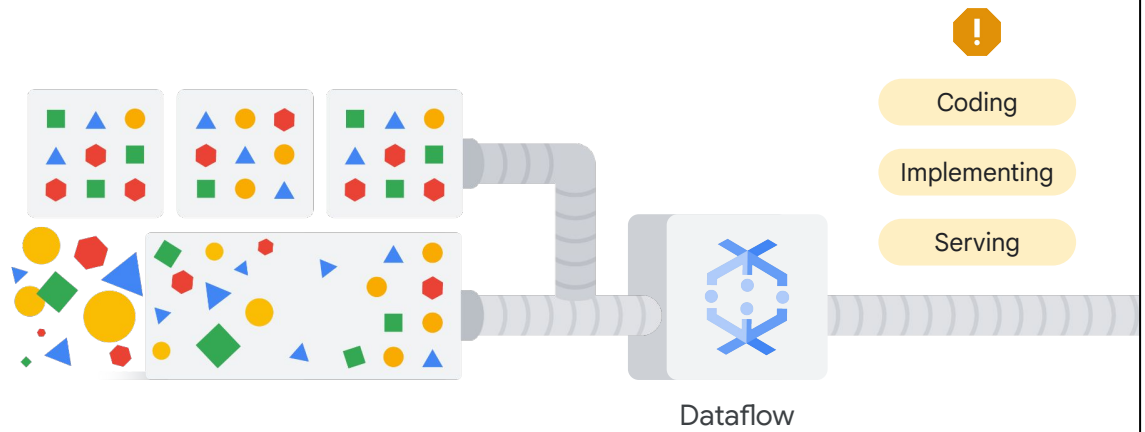


Google Cloud

After messages have been captured from the streaming input sources, you need a way to pipe that data into a data warehouse for analysis. This is where **Dataflow** comes in.

Dataflow creates a pipeline to process both streaming data and batch data. "Process" in this case refers to the steps to extract, transform, and load data, or ETL.

# Pipeline challenges



When building a data pipeline, data engineers often encounter challenges related to coding the pipeline design and implementing and serving the pipeline at scale.

## Questions to consider during pipeline design

**01**

Will the pipeline code be compatible with both batch and streaming data, or will it need to be refactored?

**02**

Will the pipeline code SDK have all the transformations, mid-flight aggregations, and windowing?

**03**

Will the pipeline code SDK be able to handle late data?

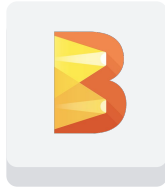
**04**

Are there existing templates or solutions that should be referenced?

During the pipeline design phase, there are a few questions to consider:

- Will the pipeline code be compatible with both batch and streaming data, or will it need to be refactored?
- Will the pipeline code software development kit, or SDK, being used have all the transformations, mid-flight aggregations and windowing? and
- Be able to handle late data?
- Are there existing templates or solutions that should be referenced?

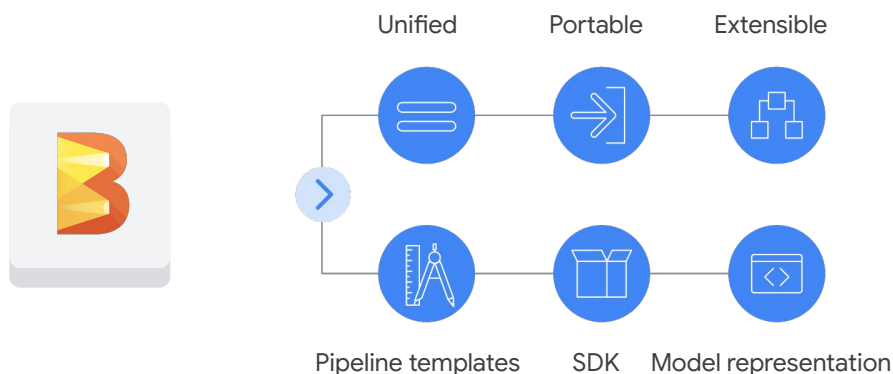
# Apache Beam



- ✓ Solution for pipeline design
- ✓ Open source
- ✓ Unified programming model
- ✓ Defines data processing pipelines
- ✓ Executes data processing pipelines

A popular solution for pipeline design is **Apache Beam**. It's an open source, unified programming model to define and execute data processing pipelines, including ETL, batch, and stream processing.

# Apache Beam features

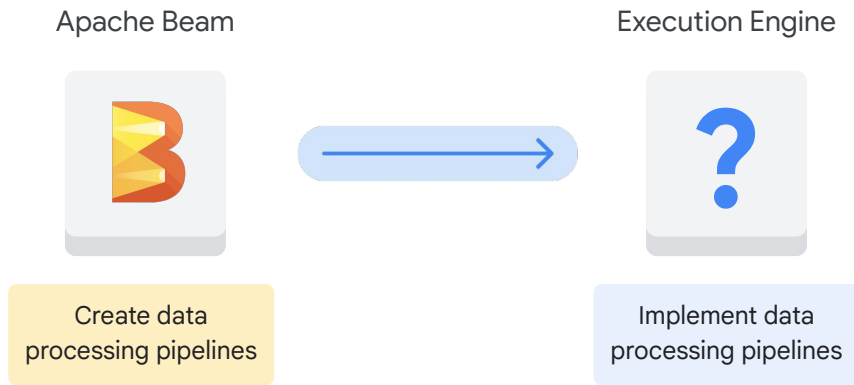


- Apache Beam is **unified**, which means it uses a single programming model for both batch and streaming data.
- It's **portable**, which means it can work on multiple execution environments, like Dataflow and Apache Spark, among others.
- And it's **extensible**, which means it allows you to write and share your own connectors and transformation libraries.
- Apache Beam provides **pipeline templates**, so you don't need to build a pipeline from nothing. And it can write pipelines in Java, Python, or Go.
- The Apache Beam software development kit, or **SDK**, is a collection of software development tools in one installable package. It provides a variety of libraries for transformations and data connectors to sources and sinks.
- Apache Beam creates a **model representation** from your code that is portable across many runners. Runners pass off your model for execution on a variety of different possible engines, with Dataflow being a popular choice.



## Implementing streaming pipelines on Cloud Dataflow

## Identify an extension engine to implement pipelines



As you just saw, Apache Beam can be used to create data processing pipelines. The next step is to identify an execution engine to implement those pipelines.





01 How much maintenance overhead is involved?

02 Is the infrastructure reliable?

03 How is the pipeline scaling handled?

04 How can the pipeline be monitored?

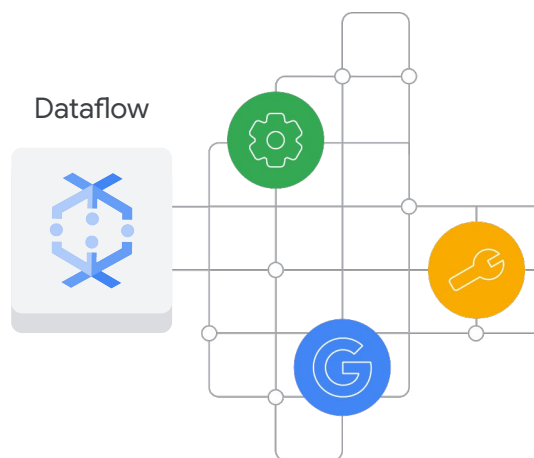
05 Is the pipeline locked in to a specific service provider?

When choosing an execution engine for your pipeline code, it might be helpful to consider the following questions:

1. How much maintenance overhead is involved?
2. Is the infrastructure reliable?
3. How is the pipeline scaling handled?
4. How can the pipeline be monitored?
5. Is the pipeline locked in to a specific service provider?

## Dataflow executes Apache Beam pipelines

- ✓ Handles infrastructure setup
- ✓ Handles maintenance
- ✓ Built on Google infrastructure
- ✓ Reliable auto scaling
- ✓ Meet data pipeline demands

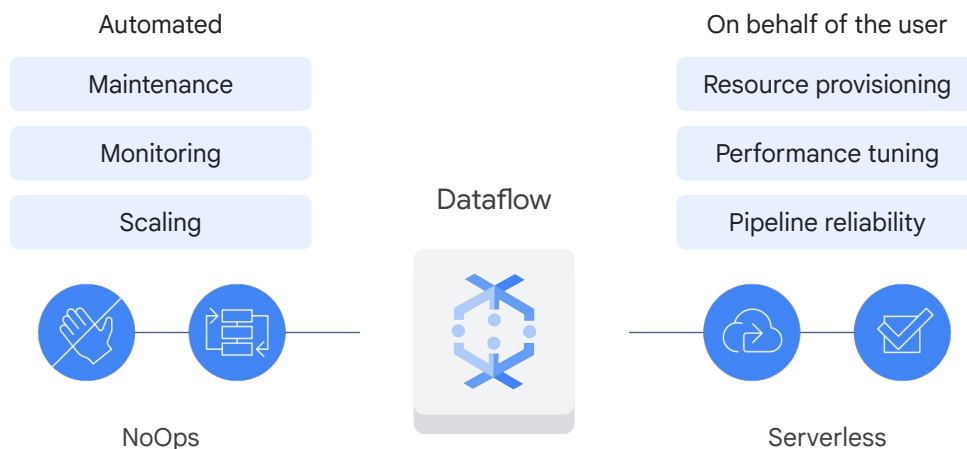


This brings us to **Dataflow**. Dataflow is a fully managed service for executing Apache Beam pipelines within the Google Cloud ecosystem.

Dataflow handles much of the complexity relating to infrastructure setup and maintenance and is built on Google's infrastructure.

This allows for reliable auto scaling to meet data pipeline demands.

# Dataflow is NoOps and serverless



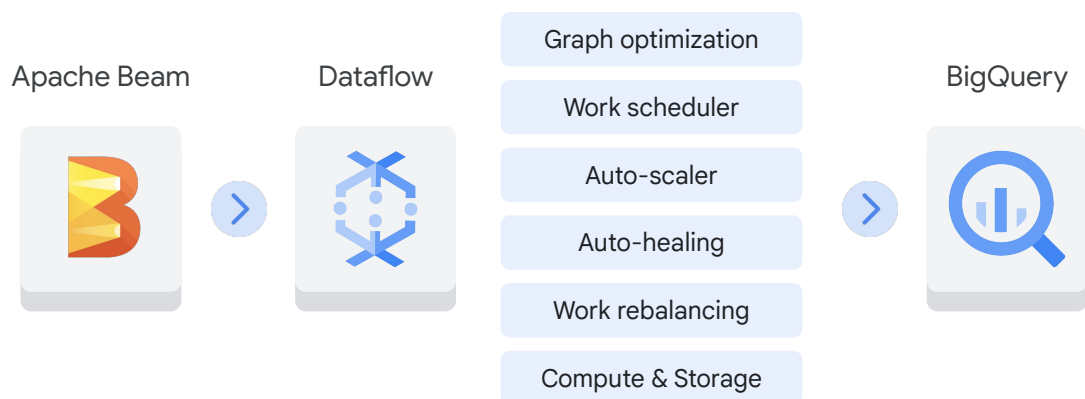
Google Cloud

Dataflow is NoOps (No Operations) and serverless.

- A **NoOps** environment is one that doesn't require management from an operations team, because maintenance, monitoring, and scaling are automated.
- **Serverless** computing is a cloud computing execution model. This is when Google Cloud, for example, manages infrastructure tasks on behalf of the users. This includes tasks like resource provisioning, performance tuning, and ensuring pipeline reliability.

Using a serverless and NoOps solution like Dataflow means that you can spend more time analyzing the insights from your datasets and less time provisioning resources to ensure that your pipeline will successfully complete its next cycles. It's designed to be low maintenance.

## Dataflow tasks



Let's explore the tasks Dataflow performs when a job is received.

- It starts by optimizing a pipeline model's execution graph to remove any inefficiencies.
- Next, it schedules out distributed work to new workers and scales as needed.
- After that, it auto-heals any worker faults.
- From there, it automatically rebalances efforts to most efficiently use its workers.
- And finally, it outputs data to produce a result. BigQuery is one of many options that data can be outputted to. You'll get some more practice using BigQuery later in this course.

By design, you don't need to monitor all of the compute and storage resources that Dataflow manages, to fit the demand of a streaming data pipeline.

# Dataflow templates



## Streaming

- Pub/Sub to BigQuery
- Pub/Sub to Cloud Storage
- Datastream to BigQuery
- Pub/Sub to MongoDB

## Batch

- BigQuery to Cloud Storage
- Bigtable to Cloud Storage
- Cloud Storage to BigQuery
- Cloud Spanner to Cloud Storage

## Utility

- Bulk compression of Cloud Storage files
- Firestore bulk delete
- File format conversion

Google Cloud

Even experienced Java or Python developers will benefit from using Dataflow templates, which cover common use cases across Google Cloud products.

The list of templates is continuously growing. They can be broken down into three categories: **streaming templates**, **batch templates**, and **utility templates**.

**Streaming templates** are for processing continuous, or real-time, data.

For example:

- Pub/Sub to BigQuery
- Pub/Sub to Cloud Storage
- Datastream to BigQuery
- Pub/Sub to MongoDB

**Batch templates** are for processing bulk data, or batch load data.

For example:

- BigQuery to Cloud Storage
- Bigtable to Cloud Storage
- Cloud Storage to BigQuery
- Cloud Spanner to Cloud Storage

Finally, **utility templates** address activities related to bulk compression, deletion, and conversion.

For a complete list of templates, please refer to the reading list.



## Visualization with Looker

## Data visualization tools

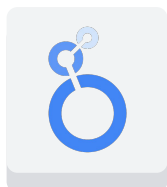


Telling a good story with data through a dashboard can be critical to the success of a data pipeline, because data that is difficult to interpret or draw insights from might be useless. After data is in BigQuery, a lot of skill and effort can still be required to uncover insights.

To help create an environment where stakeholders can easily interact with and visualize data, Google Cloud offers two solutions: **Looker** and **Looker Studio**.



## SQL dialects supported by Looker



Looker

BigQuery

Amazon Redshift

PostgreSQL

Oracle

Teradata

Apache Spark

Sap Hana

Impala

Amazon Aurora

Druid

Cloud Spanner

Presto

Azure synapse analytics

Snowflake

Hive

MySQL

Microsoft SQL Server

Actian

Vertica

Amazon Athena

SigleStore

And many more

Let's explore both of them, starting with **Looker**.

Looker supports BigQuery, and more than 60 different SQL databases.

# Looker Modeling Language



Defines a semantic modeling layer on top of databases.



Frees a data engineer from interacting with individual databases.

LookML

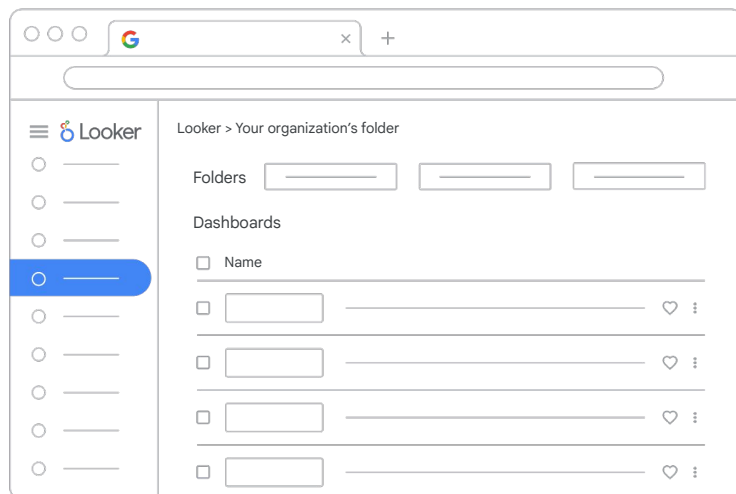


Google Cloud

It allows developers to define a semantic modeling layer on top of databases using **Looker Modeling Language**, or **LookML**.

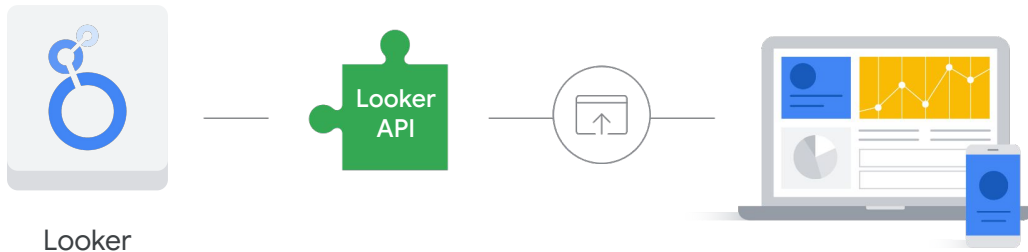
LookML defines logic and permissions independent from a specific database or a SQL language, which frees a data engineer from interacting with individual databases to focus more on business logic across an organization.

## The Looker platform is 100% web-based



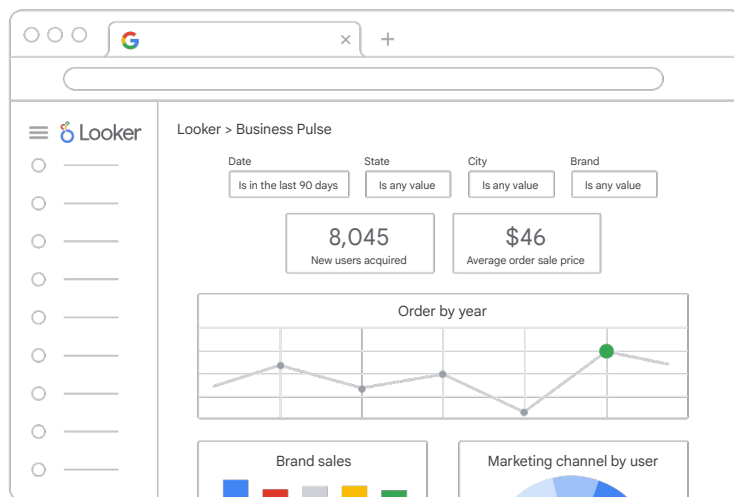
The Looker platform is 100% web-based, which makes it easy to integrate into existing workflows and share with multiple teams at an organization.

# Looker API



There is also a Looker API, which can be used to embed Looker reports in other applications.

## The Business Pulse dashboard



Google Cloud

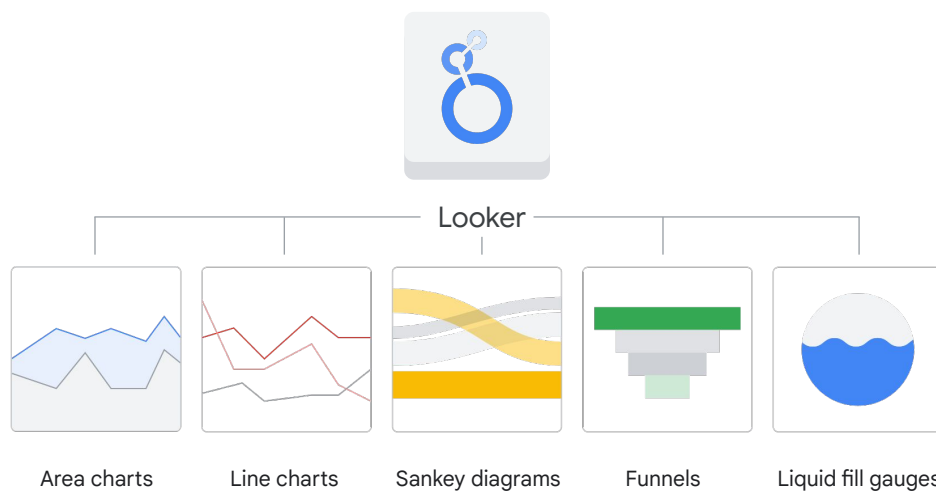
Let's explore some of Looker's features, starting with dashboards.

Dashboards, like the Business Pulse dashboard, for example, can visualize data in a way that makes insights easy to understand.

For a sales organization, it shows figures that many might want to see at the start of the week, like the number of new users acquired, monthly sales trends, and even the number of year-to-date orders. Information like this can help align teams, identify customer frustrations, and maybe even uncover lost revenue.

Based on the metrics that are important to *your* business, you can create Looker dashboards that provide straightforward presentations to help you and your colleagues quickly see a high-level business status.

## Looker visualization options

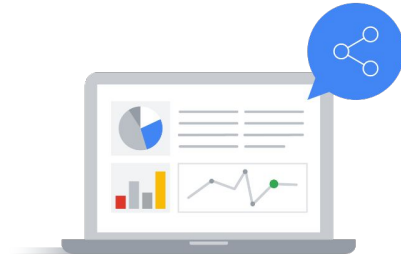


Looker has multiple data visualization options, including area charts, line charts, Sankey diagrams, funnels, and liquid fill gauges.

## Sharing Looker dashboards

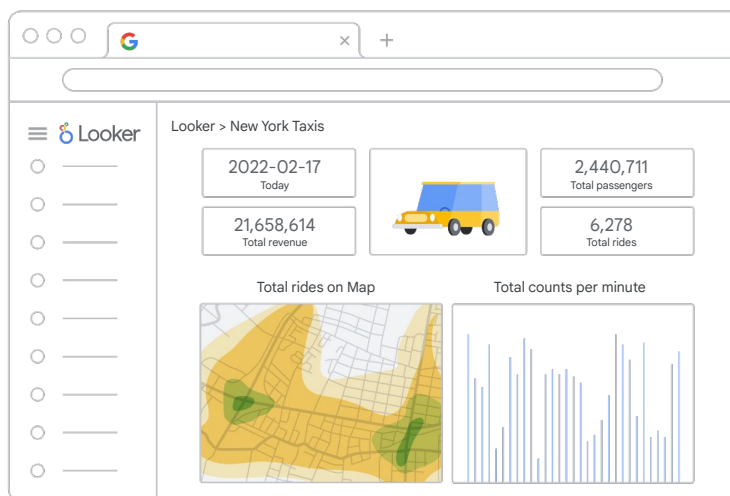
Schedule delivery through storage services such as:

- ✓ Google Drive
- ✓ Slack
- ✓ Dropbox



To share a dashboard with your team, you schedule delivery through storage services like Google Drive, Slack, and Dropbox.

## An example of Looker dashboard: New York taxis



Google Cloud

Let's explore another Looker dashboard, this time one that monitors key metrics related to New York City taxis over a period of time.

This dashboard displays:

- Total revenue
- Total numbers of passengers, and
- Total number of rides

Looker displays this information through a timeseries to help monitor metrics over time. Looker also lets you plot data on a map to see ride distribution, busy areas, and peak hours.

The purpose of these features is to help you draw insights to make business decisions.

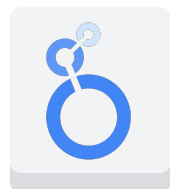
For more training on Looker, please refer to [cloud.google.com/training](https://cloud.google.com/training).





## Visualization with Looker Studio

# Looker Studio



Looker Studio

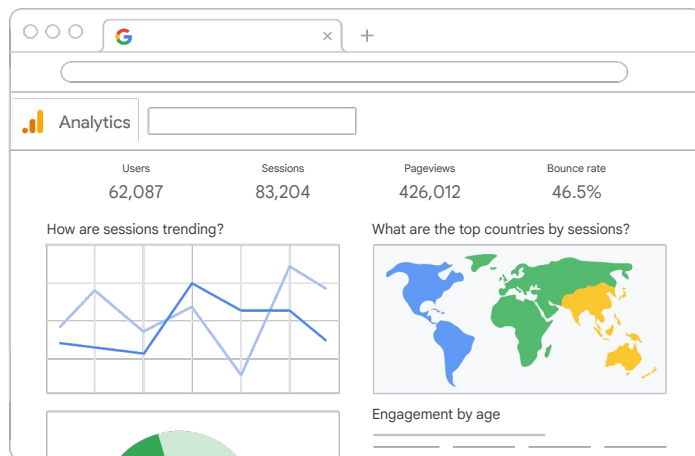


Integrated into  
BigQuery

Another popular data visualization tool offered by Google is **Looker Studio**.

Looker Studio is **integrated into BigQuery**, which makes data visualization possible with just a few clicks. This means that leveraging Looker Studio doesn't require support from an administrator to establish a data connection, which is a requirement with Looker.

# Integrated in Google products and applications



Google Analytics



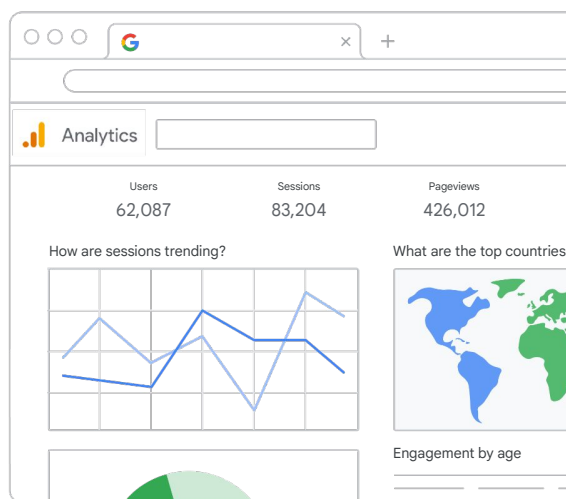
Billing dashboard

Looker Studio dashboards are widely used across many Google products and applications.

- For example, Looker Studio is integrated into **Google Analytics** to help visualize, in this case, a summary of a marketing website. This dashboard visualizes the total number of visitors through a map, compares month-over-month trends, and even displays visitor distribution by age.
- Another Looker Studio integration is the Google Cloud **billing dashboard**. You might be familiar with this from your account. Maybe you've already used it to monitor spending, for example.

# Three steps to create a Looker Studio dashboard

- 1 Choose a template
- 2 Link the dashboard to a data source
- 3 Explore your dashboard



You'll soon have hands-on practice with Looker Studio, but in preparation for the lab, let's explore the three steps needed to create a Looker Studio dashboard.

First, **choose a template**. You can start with either a pre-built template or a blank report.

Second, **link the dashboard to a data source**. This might come from BigQuery, a local file, or a Google application like Google Sheets or Google Analytics—or a combination of any of these sources.

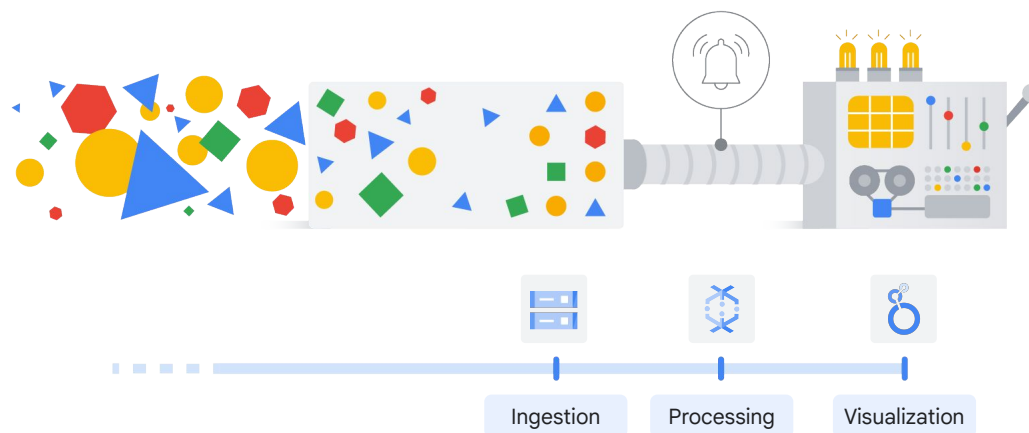
And third, **explore your dashboard!**



## Lab:

### Creating a Streaming Data Pipeline for a Real-time Dashboard with Dataflow

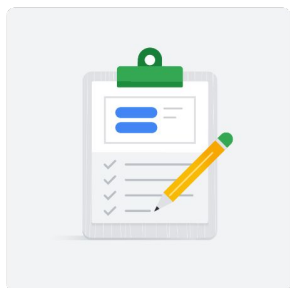
## Hands-on lab: Build a streaming data pipeline



Now it's time for hands-on practice with some of the tools you learned about in this module of the course.

In the lab that follows, you'll build a streaming data pipeline to monitor sensor data, and then visualize the dataset through a dashboard.

## You'll practice



### Hands-on lab

- 1 Creating a Dataflow job from a template
- 2 Streaming a Dataflow pipeline into BigQuery
- 3 Monitoring a Dataflow pipeline in BigQuery
- 4 Analyzing results with SQL
- 5 Visualizing key metrics in Looker Studio

### You'll practice:

- Creating a Dataflow job from a pre-existing template
- Streaming and monitoring a Dataflow pipeline into BigQuery
- Analyzing results with SQL, and
- Visualizing key metrics in Looker Studio.

Please note that, though you will use some SQL commands in this lab, the lab doesn't actually require strong SQL knowledge. We'll explore BigQuery in more detail later in this course.



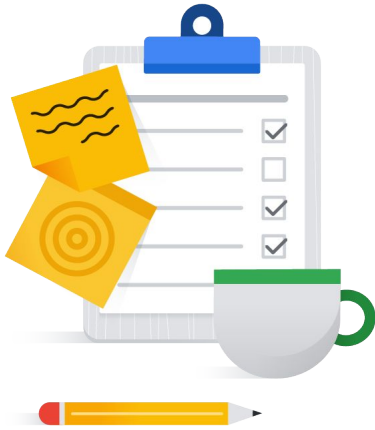
## Summary

Well done completing the lab on building a data pipeline for streaming data!

Before you move on in the course, let's do a quick recap.



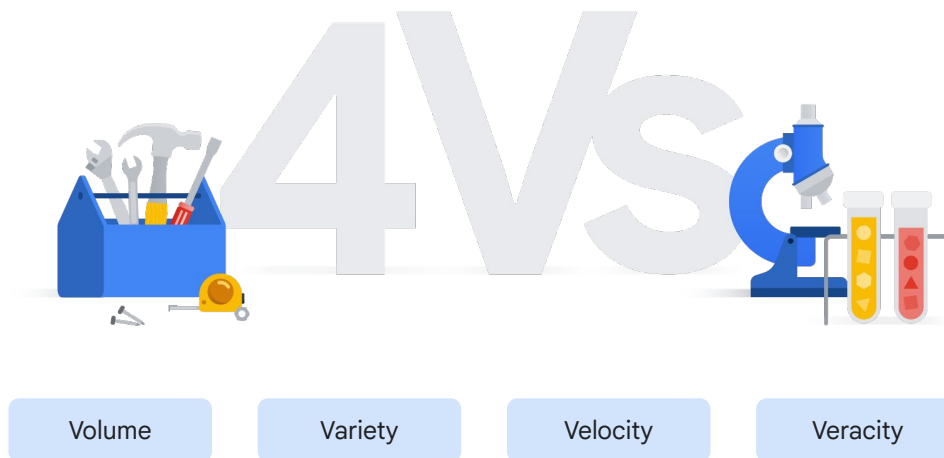
## Summary



Explored the streaming data workflow—  
from ingestion to visualization

In this module, you explored the streaming data workflow, from ingestion to visualization.

## Four big data challenges

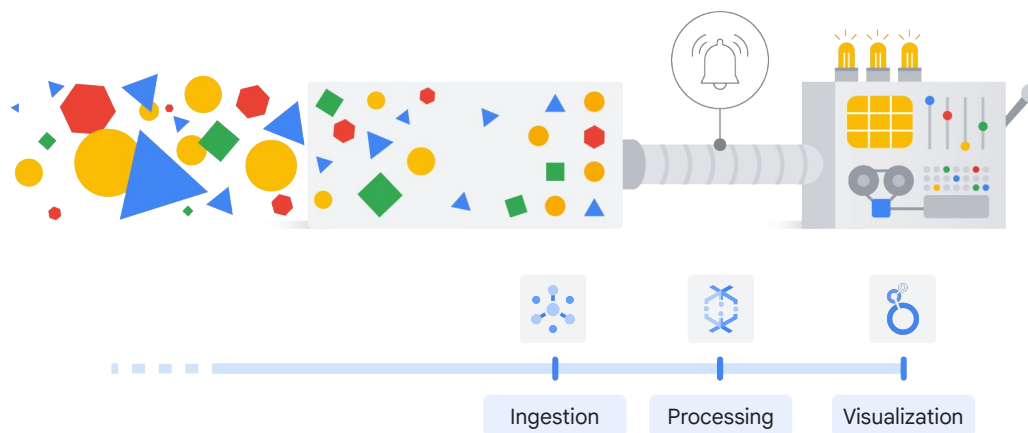


You started by learning about the four common **big data challenges**, or the 4Vs:

1. Volume (data size)
2. Variety (data format)
3. Velocity (data speed), and
4. Veracity (data accuracy).

These challenges can be especially common when dealing with streaming data.

# The streaming data pipeline



You then learned how the streaming data workflow provided by Google can help address these challenges.

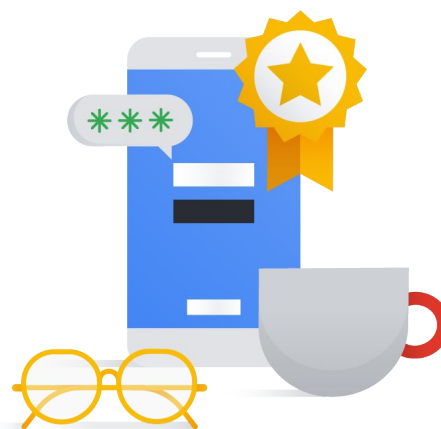
You started with **Pub/Sub**, which can be used to ingest a large volume of IoT data from diverse resources in various formats.

After that, you explored **Dataflow**, a serverless, NoOps service, to process the data. 'Process' here refers to ETL (extract, transform, and load).

And finally, you were introduced to two Google visualization tools, **Looker** and **Looker Studio**.

Head to [Kahoot.it](https://kahoot.it) on  
your phone to join  
the first quiz!

 Kahoot!



Now let's take a few moments to check in with how you're digesting this information with a quiz. We'll be using Kahoot, so you'll need to take out a mobile device if you have one or open up a new tab in your web browser and head to **kahoot.it**. I'll tell you a code to type in just a minute.

You'll be presented with multiple choice questions and will be asked to choose the answer from a list. The person who answers the questions correctly the quickest will earn the most points.

**DO:** Click the link on the slide (or [here](#)) to launch this Kahoot quiz. Select the option to **Continue as a guest**, then select **Classic mode**. Learners should follow the instructions on screen. [Click here](#) to watch an overview of how Kahoot works.



## Quiz



# Question #1

## Question

Select the correct streaming data workflow.

- A. Ingest the streaming data, process the data, and visualize the results.
- B. Visualize the data, process the data, and ingest the streaming data.
- C. Ingest the streaming data, visualize the data, and process the data.
- D. Process the data, visualize the data, and ingest the data.

## Question #1

### Answer

Select the correct streaming data workflow.

- A. Ingest the streaming data, process the data, and visualize the results.
- B. Visualize the data, process the data, and ingest the streaming data.
- C. Ingest the streaming data, visualize the data, and process the data.
- D. Process the data, visualize the data, and ingest the data.



## Question #2

### Question

Choose the distributed messaging service that is designed to ingest messages from multiple device streams.

- A. Looker
- B. Pub/Sub
- C. Looker Studio
- D. Apache Beam



## Question #2

### Answer

Choose the distributed messaging service that is designed to ingest messages from multiple device streams.

- A. Looker
- B. Pub/Sub
- C. Looker Studio
- D. Apache Beam



## Question #3

### Question

Which Google Cloud product acts as an execution engine to process and implement data processing pipelines?

- A. Apache Beam
- B. Looker
- C. Looker Studio
- D. Dataflow

## Question #3

### Answer

Which Google Cloud product acts as an execution engine to process and implement data processing pipelines?

- A. Apache Beam
- B. Looker
- C. Looker Studio
- D. Dataflow



## Question #4

### Question

Processing data in near-real time, as soon as it reaches the system, is represented by which type of data challenge?

- A. Variety
- B. Volume
- C. Velocity
- D. Veracity

## Question #4

### Answer

Processing data in near-real time, as soon as it reaches the system, is represented by which type of data challenge?

- A. Variety
- B. Volume
- C. Velocity
- D. Veracity



## Question #5

### Question

Data quality like accuracy, consistency, completeness, and timeliness are represented by which type of data challenge?

- A. Variety
- B. Volume
- C. Velocity
- D. Veracity

## Question #5

### Answer

Data quality like accuracy, consistency, completeness, and timeliness are represented by which type of data challenge?

- A. Variety
- B. Volume
- C. Velocity
- D. Veracity

