BigQuery Data Ingestion

Google Cloud

Welcome to the "BigQuery Data Ingestion" module. This module summarizes the primary options and best practices for ingesting data into BigQuery, including batch data loading, streaming ingestion, and queries to external data sources. Drawing upon your knowledge of Redshift, this module also provides a high-level overview of the similarities and differences in data ingestion options between Redshift and BigQuery to help you start reading and loading your data into BigQuery.

Let's jump in!

BigQuery Data Ingestion

Upon completion of this module, you will be able to:

1. List the data ingestion options for BigQuery.
2. Identify how the data ingestion options for Redshift differ from BigQuery.
3. Define best practices for ingesting data into BigQuery.
4. Load data into BigQuery.

Lesson 2 of 16

Lesson introduction

In this lesson, you will examine how data is ingested into Redshift and BigQuery, and the similarities and differences between the two data warehouses.

Let's get started!

Lesson 3 of 16

Key points about data ingestion

Let's start with some important similarities.

Click the flash cards to learn more.

Querying data in other locations

Redshift and BigQuery both have the ability to query data located in other locations within their cloud platform.

Ingesting batch data

Redshift and BigQuery both can ingest batch data natively.

There are also some differences to consider.

Redshift primarily ingests data using the COPY command. BigQuery uses its Storage Write API as a unified ingestion tool for both streaming and batch data.

Click the + icon to expand and learn more.

Streaming data

- Redshift uses Kinesis or managed Kafka to ingest streaming data into materialized views.
- BigQuery provides the Storage Write API for streaming records directly into BigQuery tables.

Other data locations within their cloud

- Redshift can use COPY to ingest data from locations inside AWS.
- BigQuery provides several options for querying data stored in other locations within Google Cloud, including Cloud Storage, Cloud Bigtable, Cloud Spanner, Cloud SQL, and Google Drive.

Multi-cloud ingestion

- Redshift lets you ingest data from resources in AWS.
- BigQuery Transfer Service provides the ability to ingest data from other clouds and data warehouses including Amazon S3, Amazon Redshift, and Teradata; BigQuery Omni provides the ability to query data in other clouds including Amazon S3 and Azure Blob Storage.

BigQuery and Redshift

Here's a summary of the options for data ingestion in BigQuery and Redshift.

Data ingestion topic	BigQuery	Redshift
Native data loading options	Batch loading Streaming ingestion Storage Write API: batch loading and streaming ingestion BigQuery Data Transfer Service: scheduled batch loading including Google Software as a Service (SaaS) apps, external cloud storage providers, and external data warehouses Datastream: change data capture and replication from relational database sources to BigQuery	Loading data Loading data best practices Streaming ingestion

Data ingestion topic	BigQuery	Redshift
Query external data	Query external data using: • BigQuery Omni (multi-cloud) • BigLake tables (Cloud Storage, Amazon S3, Azure Blob Storage) • Federated queries (Cloud Spanner, Cloud SQL) • External tables (Cloud Bigtable, Cloud Storage, Google Drive) • Object tables (unstructured data in Cloud Storage)	Query external data with federated queries Query external data with Redshift Spectrum

Lesson introduction

In this lesson, you will explore data ingestion options for BigQuery including batch ingestion, streaming ingestion, query materialization, and the BigQuery Data Transfer Service.

Let's get started!

Summary of data ingestion options and tools

There are four key data ingestion options. The method that you use to load the data into BigQuery depends on data source and the level of transformation required to convert the raw data to the final format you need.

Click the + icon to expand and learn more.

Batch ingestion

With batch loading, you load the source data into a BigQuery table in a single batch operation. For example, the data source could be a CSV file, an external database, or a set of log files. Traditional extract, transform, and load (ETL) jobs fall into this category.

- Multiple file formats are supported.
- Snapshot-based arrival. All data arrives at once, or not at all.

Streaming ingestion

With streaming, you continually send smaller batches of data in real time, so the data is available for querying as it arrives.

- Continuous ingestion from many sources (web/mobile apps, point of sale, supply chain).

- Immediate query availability from buffer.

Query materialization

You can use SQL to generate data and store the results in BigQuery.

- Use data manipulation language (DML) statements to perform bulk inserts into an existing table or store query results in a new table.
- Use a CREATE TABLE ... AS statement to create a new table from a query result.
- Run a query and save the results to a table. You can append the results to an existing table or write to a new table.

Query external data sources

BigQuery also provides many options for querying external data sources on both Google cloud and other cloud providers.

- BigQuery Omni (multi cloud)
- BigLake tables (Cloud Storage, Amazon S3, Azure Blob Storage)
- Federated queries (Cloud Spanner, Cloud SQL)
- External tables (Cloud Bigtable, Cloud Storage, Google Drive)
- Object tables (unstructured data in Cloud Storage)

Frameworks for loading data into BigQuery

There are several frameworks for loading data into BigQuery that begin with extracting data and end with either loading or transforming data. Depending on the amount and type of data transformation that you want to do, you may decide to transform the data before loading it, or vice versa.

Click the + icon to expand and learn more.

Extract and load (EL)

EL is appropriate when the source data can be imported "as is" to a destination system, so no transformation is needed. An example use case for EL is when the source and destination tables have the same table structure.

You can use EL to run batch loads or schedule periodic loads to BigQuery. Some options for EL include scheduled queries and the BigQuery Data Transfer Service. You can also use Cloud Composer or Cloud Functions to trigger EL workflows into BigQuery such as when a new source file is added to Cloud Storage.

Extract, load, and transform (ELT)

ELT is appropriate when the source data is not in the desired final format but the amount of transformation required is low and does not reduce the amount of data.

In this case, you can use ELT to load the data directly into BigQuery, transform the data using SQL, and write it to a new table.

Extract, transform, and load (ETL) –

ETL is appropriate when the source data needs to be significantly transformed or when transforming the source data would greatly reduce the amount of data to be loaded to the final destination.

A common workflow for ETL to BigQuery is to design Dataflow pipelines to transform data from a source, such as Cloud Storage, and then write it into BigQuery after the transformation is completed.

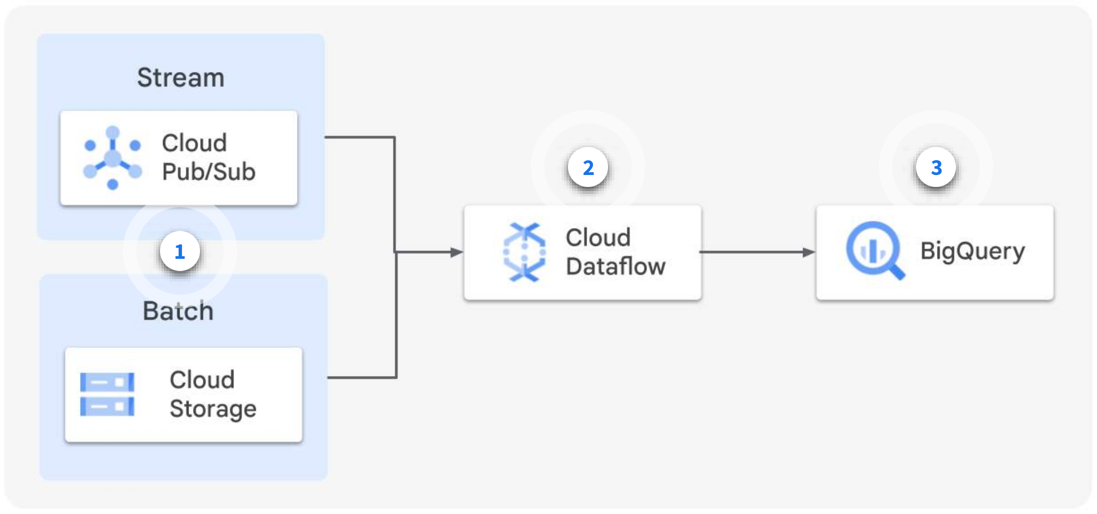
Build ETL pipelines for batch and streaming data ingestion using Dataflow

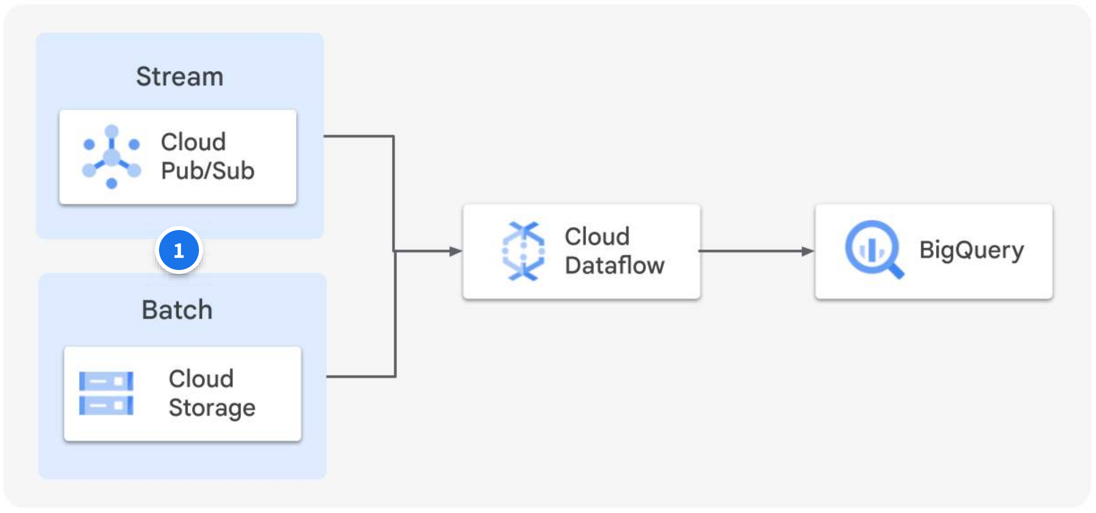
ETL pipelines in Dataflow are particularly useful for the following use cases:

1. The raw data needs to be quality-controlled, transformed, or enriched before being loaded into BigQuery.
2. The transformations are difficult to do in SQL.
3. The data loading has to happen continuously (i.e. if the use case requires streaming).
4. You want to integrate with continuous integration / continuous delivery (CI/CD) systems and perform unit testing on all components.

Here is an example of the Google Cloud architecture for an ETL workflow using Dataflow.

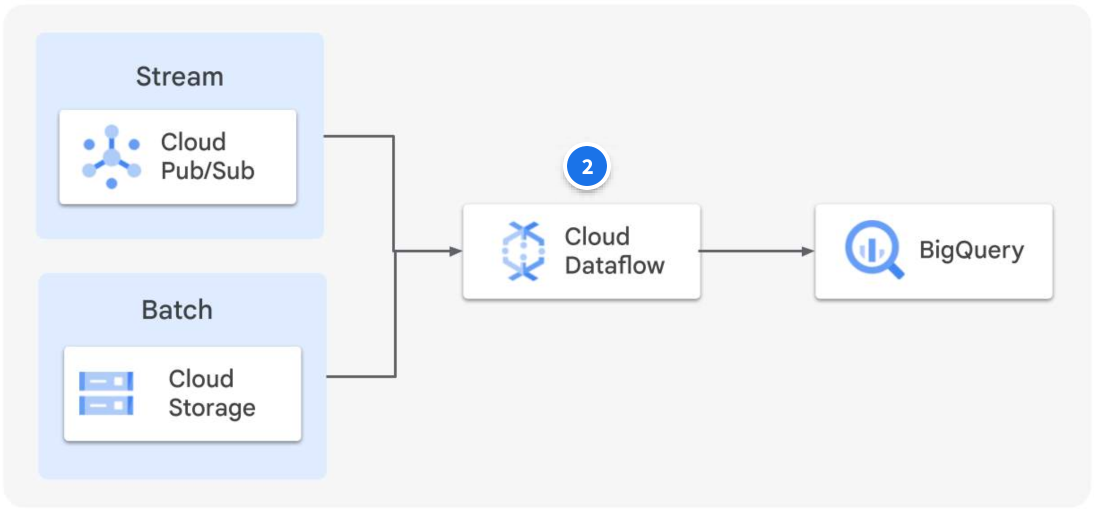
Click the numbers in sequence to learn more.


Stream
Cloud Pub/Sub
1
Batch
Cloud Storage
2
Cloud Dataflow
3
BigQuery



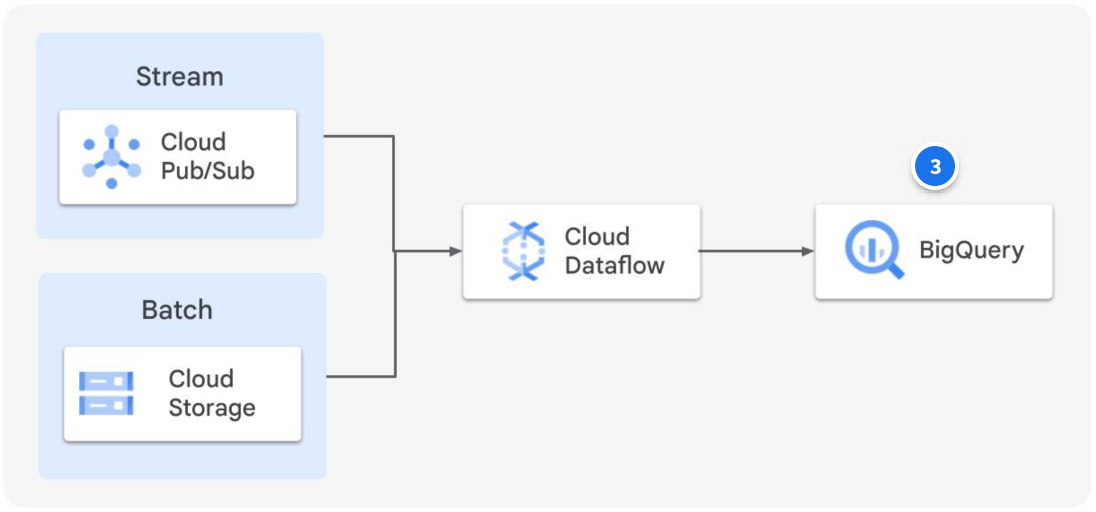
Extract

Extract data from sources such as Pub/Sub for streaming data or Cloud Storage for batch data loading.



Transform

Design and run Dataflow pipelines to clean and transform data.



Load

End your Dataflow pipeline with a step to write transformed data into BigQuery.

Lesson 8 of 16

Batch ingestion

Options for batch loading in BigQuery

Batch loading can be done as a one-time operation or on a recurring schedule. Options for batch loading in BigQuery include the following.

Click the + icon to expand and learn more.

Load jobs

Use a load job to ingest data from Cloud Storage or from a local file. The records can be in Avro, CSV, JSON, ORC, or Parquet format.

SQL

Use the LOAD DATA SQL statement to load data from one or more files into a new or existing table. You can use the LOAD DATA statement to load Avro, CSV, JSON, ORC, or Parquet files.

BigQuery Data Transfer Service

Use the BigQuery Data Transfer Service to automate loading data from Google Software as a Service (SaaS) apps or from third-party applications and services.

BigQuery Storage Write API —

Use the Storage Write API to batch-process records and commit them in a single atomic operation. If the commit operation fails, you can safely retry the operation.

Other managed services —

Use other managed services to export data from an external data store and import it into BigQuery. For example, you can load data from Firestore exports.

Advantages of batch loading

There are many advantages associated with batch ingestion into BigQuery.

- Batch ingestion in BigQuery is free.
- As opposed to traditional systems, batch ingestion doesn’t consume query capacity.
- You can load petabytes per day.
- The load follows the Atomic Consistent Isolated Durable (ACID) pattern of data management semantics.

- After a batch load, you can keep the data updated with streaming APIs for real-time insights.

Required permissions

To load data into BigQuery, you need identity and access management (IAM) permissions to run a load job and load data into BigQuery tables and partitions. If you are loading data from Cloud Storage, you also need IAM permissions to access the bucket that contains your data.

BigQuery Data Transfer Service

The BigQuery Data Transfer Service automates data movement into BigQuery on a scheduled, managed basis. Your analytics team can lay the foundation for a BigQuery data warehouse without writing a single line of code.

After you configure a data transfer, the BigQuery Data Transfer Service automatically loads data into BigQuery on a regular basis. You can also initiate data backfills to recover from any outages or gaps.

You can access the BigQuery Data Transfer Service using the:

- Google Cloud console
- bq command-line tool
- BigQuery Data Transfer Service API

Supported data sources

The BigQuery Data Transfer Service is made to simplify and automate data movement from other sources (inside and outside of Google Cloud) to BigQuery on a scheduled and managed basis without having to introduce other orchestration and ETL tools. Let's explore some examples.

Click the + icon to expand and learn more.

Google Software as a Service (SaaS) apps —

- Campaign Manager
- Cloud Storage
- Google Ad Manager
- Google Ads
- Google Merchant Center (beta)
- Google Play
- Search Ads 360 (beta)
- YouTube Channel reports
- YouTube Content Owner reports

External cloud sources —

External cloud storage providers:

- Amazon S3

Data warehouses:

- Teradata
- Amazon Redshift

BigQuery Storage Write API

The BigQuery Storage Write API is a unified data-ingestion API for BigQuery. It combines streaming ingestion and batch loading into a single high-performance API. You can use the Storage Write API to stream records into BigQuery in real time or to batch process an arbitrarily large number of records and commit them in a single atomic operation.

Here are the numerous advantages of using the Storage Write API.

Exactly-once delivery semantics

The Storage Write API supports exactly-once semantics through the use of stream offsets. Unlike the tabledata.insertAll method, the Storage Write API never writes two messages that have the same offset within a stream, if the client provides stream offsets when appending records.

Stream-level transactions

You can write data to a stream and commit the data as a single transaction. If the commit operation fails, you can safely retry the operation.

Transactions across streams

Multiple workers can create their own streams to process data independently. When all the workers have finished, you can commit all of the streams as a transaction.

Efficient protocol

The Storage Write API is more efficient than the older insertAll method because it uses gRPC streaming rather than REST over HTTP. The Storage Write API also supports binary formats in the form of protocol buffers, which are a more efficient wire format than JSON. Write requests are asynchronous with guaranteed ordering.

Schema update detection

If the underlying table schema changes while the client is streaming, then the Storage Write API notifies the client. The client can decide whether to reconnect using the updated schema, or continue to write to the existing connection.

Lower cost

The Storage Write API has a significantly lower cost than the older insertAll streaming API. In addition, you can ingest up to 2 TB per month for free.

Streaming ingestion

Options for streaming data into BigQuery

In streaming scenarios, data arrives continuously and should be available in real-time for reads with minimal latency. Options for streaming in BigQuery include the following.

Click the + icon to expand and learn more.

Storage Write API

Use the Storage Write API for high-throughput streaming ingestion with exactly-once delivery semantics.

Dataflow

Use Dataflow with the Apache Beam SDK to set up a streaming pipeline that writes to BigQuery.

BigQuery Connector for SAP

Use the BigQuery Connector for SAP for near real time replication of SAP data directly into BigQuery.

Streaming ingestion using BigQuery Storage Write API

What are some common use cases for streaming ingestion using the Storage Write API?

Click the + icon to expand and learn more.

Real-time dashboards and queries

Streaming data into BigQuery can enable near real-time analysis and visualization of your data. You can use SQL in BigQuery to analyze, aggregate, and summarize your data, and then use supported business intelligence (BI) options such as Looker Studio to generate and share dashboards with your stakeholders.

Event logging

Streaming data into BigQuery can also support high volume event logging such as event tracking. For example, if you have a mobile application that tracks events, your application or mobile servers could independently record user interactions or system errors and stream them into BigQuery. You could then analyze this data to determine overall trends, such as areas of high interaction or problems, and monitor error conditions in real-time.

The Storage Write API is a flexible tool that can be used for both batch and streaming data. When using the Storage Write API for streaming workloads, consider what guarantees you need:

- If your application only needs at-least-once semantics and can accept the possibility of duplicate records appearing in the destination table, then use the default stream.
- If you need exactly-once semantics, then create one or more streams in committed type and use stream offsets to guarantee exactly-once writes.

In committed type, data written to the stream is available for query as soon as the server acknowledges the write request. The default stream also uses committed type, but does not provide exactly-once guarantees.

Required permissions

To use the Storage Write API, you must have *bigquery.tables.updateData* permissions.

The following predefined Identity and Access Management (IAM) roles include *bigquery.tables.updateData* permissions:

- **bigquery.dataEditor**
- **bigquery.dataOwner**
- **bigquery.admin**

References

- Learn more about how BigQuery can load both batch and streaming data using a single API from the documentation titled "Batch load and stream data with BigQuery Storage Write API."
- Learn more about the BigQuery Data Transfer Service from the documentation titled "What is BigQuery Data Transfer Service?"
- Learn more about auto-detection of schemas when loading new data or querying external data sources from the documentation titled "Using schema auto-detection."
- Learn more about federated queries from the documentation titled "Introduction to federated queries."
- Learn more about quotas and limits for BigQuery load jobs from the documentation titled "Load jobs."

Lesson introduction

In this lesson, you will explore core considerations for ingesting data into BigQuery, including best practices for batch and streaming data ingestion workflows.

Let's get started!

General best practices for data ingestion

Best practices for working with data formats

BigQuery supports a wide-variety of file formats for data ingestion. Here are the commonly used data formats and considerations for loading each into BigQuery.

Click the tabs to learn more.

AVRO	PARQUET/ORC	CSV/JSON

Avro is a binary row-based format which can be split and read in parallel by multiple slots. This makes the Avro file format an ideal choice when optimizing for load speed.

The Avro binary format is the preferred format for loading both compressed and uncompressed data into BigQuery. While compressed Avro files are not supported, compressed data blocks are supported for loading into BigQuery.

AVRO | PARQUET/ORC | CSV/JSON

Like Avro, Parquet and ORC are binary formats. While they do not load as quickly as Avro, they are still good options for optimizing load speed for both uncompressed and compressed data.

Note that compressed Parquet files are not supported, but compressed data blocks are supported. Similarly, compressed ORC files are not supported, but compressed file footer and stripes are supported.

AVRO | PARQUET/ORC | CSV/JSON

For CSV and JSON files, BigQuery can load uncompressed files significantly faster than compressed files because uncompressed files can be read in parallel.

If loading speed is important to your workflow and you have a lot of bandwidth to load your data, it is recommended to leave your CSV and JSON files uncompressed when loading into BigQuery.

Best Practices for Extract, Load, and Transform (ELT) and Extract, Transform, and Load (ETL)

Here are some high-level best practices for EL, ELT and ETL workflows in BigQuery.

1. Consider your use case carefully when deciding between EL, ELT, or ETL. In some cases, you may not actually need to load all of the data in BigQuery, and in other cases, it may be more efficient to load all data into BigQuery first before transforming it with SQL.

2. Remove redundant data storages. BigQuery leverages federated queries to Google Cloud Storage to load and transform data in a single-step.

3. Load data into raw and staging tables before publishing to reporting tables. Raw tables contain the daily or full extract and can be truncated or appended. Staging tables contain a history of all changes for change-data-capture. Reporting tables contain data in a format suitable for analytical queries.

4. DML statements or views can be used between staging and reporting tables. Use MERGE statements to perform transformations, such as expiring old records and marking a new one as active.

5. You can use Dataflow for streaming pipelines and to speed up large complex batch jobs. Try Google-provided Dataflow templates Dataflow Templates and modify the open-source pipeline for your more complex needs.

Best practices for the Storage Write API

Choosing between streaming and batch options

Since the Storage Write API can be used for both batch and streaming data ingestion, the best practices for using this API begin with identifying which type of data ingestion works best for your use case, and then with understanding how Storage Write API is leveraged for each scenario.

If you need to ingest and analyze data in near real time, consider streaming the data. With streaming, the data is available for querying as soon as each record arrives. For example, if you need to report on frequently updated data, it's often better to stream a change log and use a view to obtain the latest results. Another scenario is data that arrives infrequently or in response to an event. In that case, consider using Dataflow to stream the data or use Cloud Functions to call the Storage Write API in response to a trigger.

If your source data changes slowly or you don't need continuously updated results, consider using a load job. For example, if you use the data to run a daily or hourly report, batch load jobs can be less expensive and can use fewer system resources.

Rather than streaming or batch loading data into BigQuery, another good option for transactional data is to use federated queries. For example, you can use Cloud SQL as your online transaction processing (OLTP) database and use federated queries to join the data in BigQuery.

BigQuery Storage Write API best practices

Regardless of whether you are ingesting batch or streaming data, here are some best practices that will help you to fully leverage the capabilities of the Storage Write API.

Click the + icon to expand and learn more.

Limit the rate of stream creation —

Before creating a stream, consider whether you can use the default stream. For streaming scenarios, the default stream has fewer quota limitations and can scale better than using application-created streams. If you use an application-created stream, then make sure to utilize the maximum throughput on each stream before creating additional streams. For example, use asynchronous writes.

Limit the number of concurrent connections —

For best performance, use one connection for as many data writes as possible. Don't use one connection for just a single write, or open and close streams for many small writes.

Manage stream offsets to achieve exactly-once semantics —

The Storage Write API only allows writes to the current end of the stream, which moves as data is appended. The current position in the stream is specified as an offset from the start of the stream. When you write to an application-created stream, you can specify the stream offset to achieve exactly-once write semantics.

Evaluate whether you need exactly-one semantics

Before using stream offsets, consider whether you need exactly-once semantics. For example, if your upstream data pipeline only guarantees at-least-once writes, or if you can easily detect duplicates after data ingestion, then you might not require exactly-once writes. In that case, using the default stream is recommended, as this does not require keeping track of row offsets.

Do not block on AppendRows calls

The AppendRows method is asynchronous. You can send a series of writes without blocking on a response for each write individually. The response messages on the bidirectional connection arrive in the same order as the requests were enqueued. For the highest throughput, call AppendRows without blocking to wait on the response.

Handle schema updates

If you want to send new fields in the payload, you should first update the table schema in BigQuery. The Storage Write API detects schema changes after a short time, on the order of minutes. When the Storage Write API detects the schema change, the AppendRowsResponse response message contains a TableSchema object that describes the new schema.

References

- Learn more about all BigQuery data loading options from the documentation titled "Introduction to loading data."
- Learn more about considerations for BigQuery data ingestion options from the documentation titled "Choosing a data ingestion method."
- Learn more about loading normalized data to BigQuery from the documentation titled "Loading denormalized, nested, and repeated data."
- Learn about other options for working with data without loading it into BigQuery from the documentation titled "Alternatives to loading data."