

Google Cloud Platform

Transforming, Staging, and Loading Data

BigQuery for Data Analysts

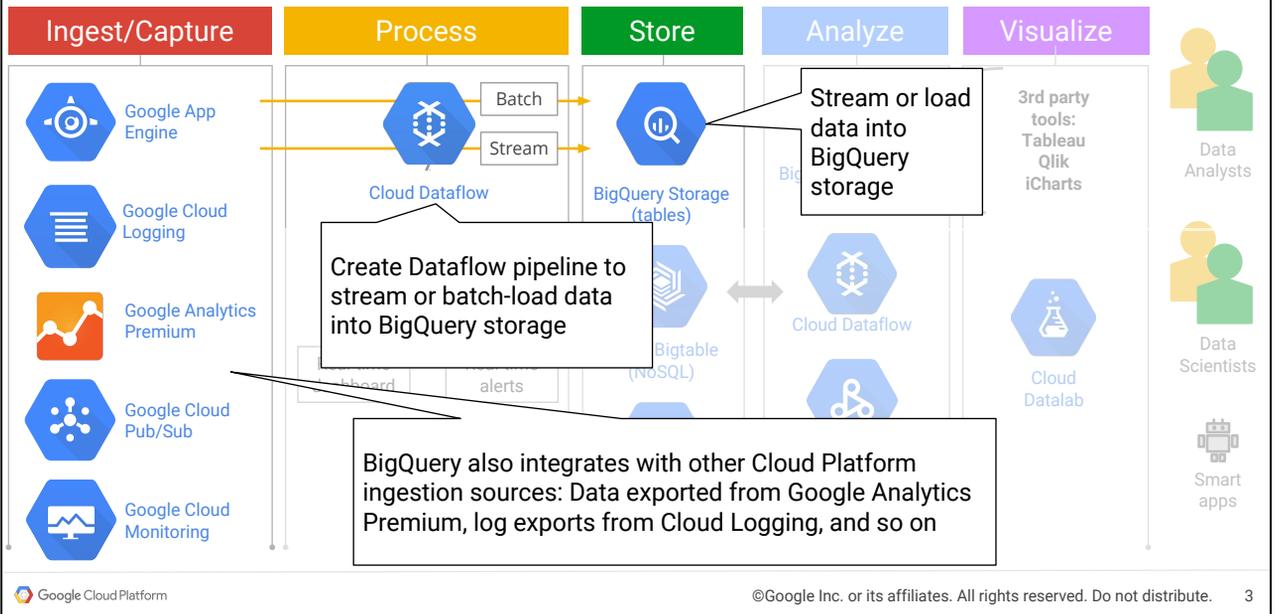
V1.2

Approximate timing: 60 minutes

Agenda

- 1 Ingesting/Capturing Data
- 2 Processing/Transforming Data
- 3 Storing Data
- 4 Federated Queries
- 5 Quiz & Lab

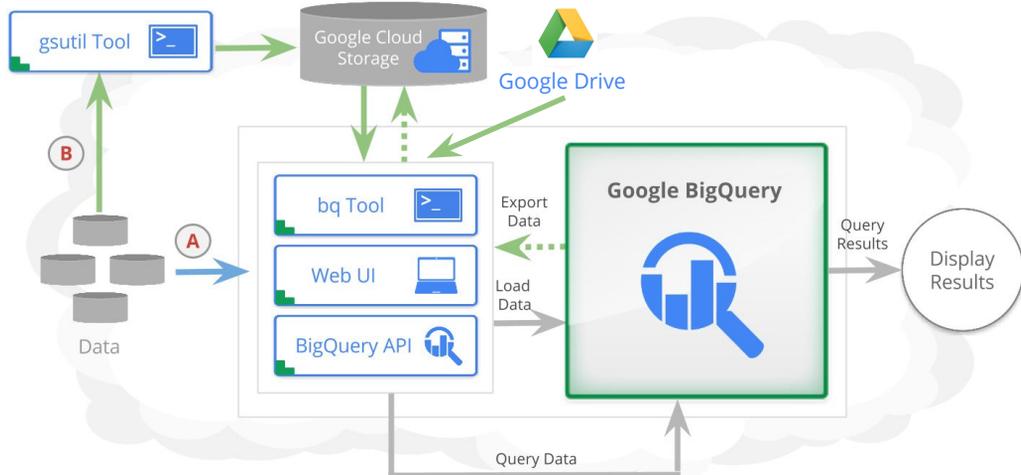
Big Data Lifecycle



Notes:
This is a typical end-to-end big data workflow.

Ingesting Data - BigQuery (1 of 5)

- Load data into BigQuery using CLI, web UI, or API



Notes:

BigQuery CLI:

- Good for uploading large data files, scheduling data file uploads
- Create table, define schema, and load data with one command
- Can also run queries from the command line
 - Interactively or batch mode queries
 - Automating scripts using scripting language
- Run the bq command-line tool from:
 - A Compute Engine instance
 - Cloud Shell
 - Client machine (requires installing the [Google Cloud SDK](https://cloud.google.com/sdk/))

Syntax for loading data via CLI:

```
bq load [--source_format=NEWLINE_DELIMITED_JSON|CSV]  
destination_table data_source_uri table_schema
```

More details: <https://developers.google.com/bigquery/bq-command-line-tool>

Ingesting Data - BigQuery (2 of 5)

- Load data directly into BigQuery storage using streaming insert or a load job
- Load jobs support:
 - Google Cloud Storage
 - Standard, Durable Reduced Availability, or Nearline (reduced performance)
 - Google Drive (JSON, CSV, AVRO, Sheets (first tab only))
 - CSV, JSON, AVRO file uploads (slower)
- Can also use partner-provided tools/services

Ingesting Data - BigQuery (3 of 5)

- Load jobs support four data sources:
 - Objects in Google Cloud Storage
 - Data sent with the job or streaming insert
 - A Google Cloud Datastore backup
 - Data in Google Drive
- Data can be:
 - Loaded into a new table
 - Appended to a table
 - Used to overwrite a table

Ingesting Data - BigQuery (4 of 5)

- BigQuery can ingest both compressed (GZIP) and uncompressed files
 - Highly parallel load operations allow uncompressed files to load significantly faster than compressed files
 - Uncompressed files are larger
 - Possible bandwidth limitations
 - More costly to store
- Weigh compression options based on use case

Ingesting Data - BigQuery (5 of 5)

- File size limits:

File Type	Compressed	Uncompressed
CSV	4 GB	<ul style="list-style-type: none">● With newlines in strings: 4 GB● Without newlines in strings: 5 TB
JSON (newline delimited)	4 GB	5 TB
AVRO	Compressed files not supported; compressed data blocks are	5 TB (2 MB for the file header)

Appending and Reloading Data

- Use CLI or API to append data to an existing table
 - Use `writeDisposition` flag value of `WRITE_APPEND` to append data to end of table
 - Useful for batch load of additional data to reference tables
- BigQuery does not currently support deletes
 - Use the CLI or the API to truncate and reload data
 - Use `writeDisposition` flag value of `WRITE_TRUNCATE`
 - `WRITE_TRUNCATE` will automatically replace the data in the table with load job results

Streaming Data into BigQuery (1 of 2)

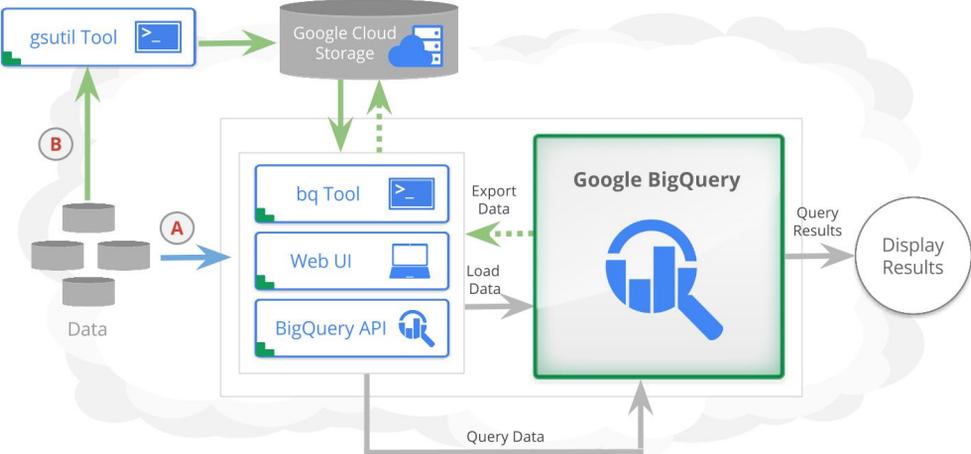
- Stream data into BigQuery using the Streaming API
 - Using `tabledata().insertAll()` method
- Query data without the delay of running a load job
- Subject to streaming quotas
- For data consistency supply `insertId` for each inserted row
 - De-duplication is done on a best effort basis, and can be affected by network errors or internal errors within BigQuery

Streaming Data into BigQuery (2 of 2)

- Use cases:
 - High-volume event logging
 - Near real-time dashboards and queries
- No warm-up delay - Query your data within a few seconds of the first streaming insertion
 - Up to 90 minutes until available for copy, export operations
- Templates allow automatic table creation
 - Split table into smaller tables without adding client-side code

Ingesting Data - Cloud Storage (1 of 3)

- Load data into Cloud Storage using CLI (gsutil) or Cloud Console



Ingesting Data - Cloud Storage (2 of 3)

- Streaming transfers using `gsutil` or boto library
 - Based on HTTP chunked transfer encoding
- Cloud Storage Transfer Service
 - Transfers data from a *data source* to a *data sink*
 - *Data source* can be:
 - Amazon Simple Storage Service (Amazon S3) bucket
 - HTTP/HTTPS location
 - Google Cloud Storage bucket
 - *Data sink* (destination) is always Cloud Storage bucket

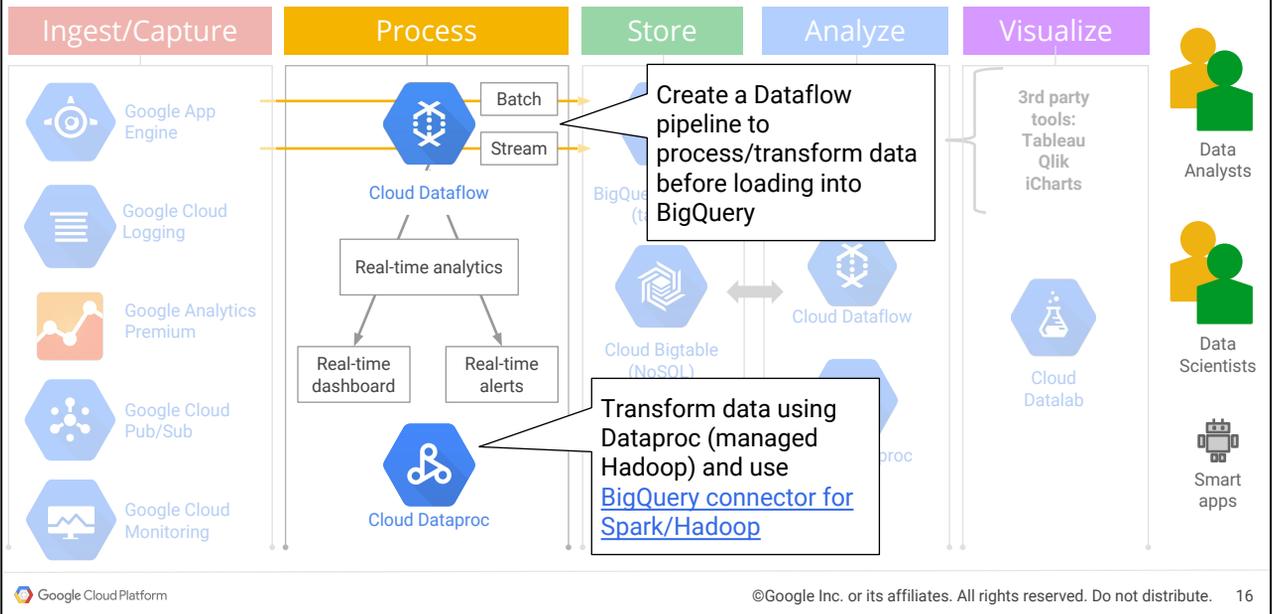
Ingesting Data - Cloud Storage (3 of 3)

- Upload using secure connection via VPN or Google Cloud Interconnect
 - Carrier Interconnect, Direct Peering, or Cloud VPN
- Offline Media Import/Export
 - Send physical media (hard disk drives, tapes, USB flash drives) to third party service provider who uploads data to Cloud Storage on your behalf
 - Helpful for slow, unreliable, or expensive Internet connections
- Resumable uploads using XML API

Agenda

- 1 Ingesting/Capturing Data
- 2 Processing/Transforming Data
- 3 Storing Data
- 4 Federated Queries
- 5 Quiz & Lab

Big Data Lifecycle



Notes:

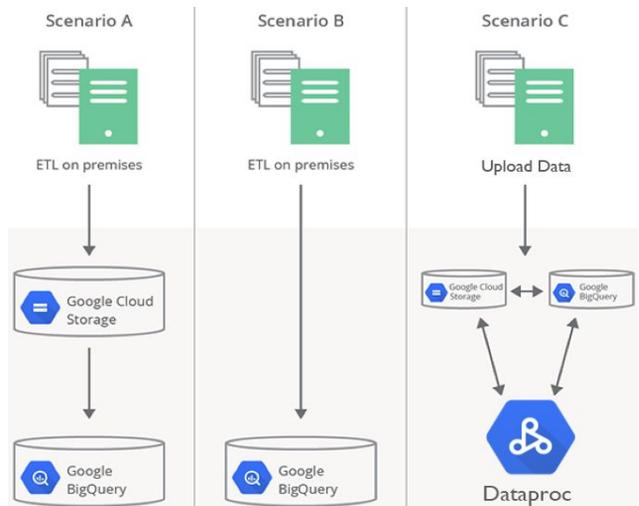
For the transformation stage, you can choose to use an on-premise ETL tool. The problem with using on-premise ETL tool handling data analysis project that may be spikey. For example, if there is more traffic during the holiday season, then you need to provision the resources to meet the peak demand. And for the non-peak seasons, the ETL tool is under-utilized.

When times comes to analyze the data, obviously Hadoop and Spark are very useful there whether you use them directly or you use third party tools on top of them - Either open source or commercial tools.

Or you can use BigQuery which is Google's fully-managed data analytics service in the cloud with unlimited storage. BigQuery offers interactive analysis on multi-terabyte datasets.

Processing/Transforming Data

- Before loading data, process/transform according to schema, data model
- ETL can be on-premises or cloud-based
 - Use Google Cloud Dataproc (Hadoop/Spark, NoOps managed service)
 - Use [third-party ETL tools](#)



Schema Design Considerations

- Denormalized versus relational
 - Denormalized yields better performance with some duplication
 - JOINS on relational data are performant but can be slower than queries on denormalized data
- Flat versus nested and repeated
- Table partitioning
 - Single table for all data
 - Partition data by some range of values (date) - covered later
 - Table decorators (partition, snapshot) - covered later

Notes:

Table Partitioning

One of the design decisions that needs to be made before loading large data sets is whether or not the table should be split up into multiple tables. The following are some reasons for partitioning the tables:

- Static versus Changing Data Management - BigQuery is optimized for read performance and it does not support updates to data that has already been loaded. By separating data that is more likely to change from data that will not, updates can be achieved by simply deleting a table and re-populating it with the updated data, avoiding the need to reload the entire dataset.
- Cost Optimization - Queries are charged by the amount of data processed. It is recommended to partition the table according to the querying pattern to optimize cost. For example, if analysis is mostly focused on the past quarter's performance, it may make sense to shard the tables by quarter so only data from the quarter of interest is processed. Similarly, if analysis is mostly done on a per customer basis, partition the table using customer as the dimension.
- Table unions can be used to query across partitioned tables.
- Usage Patterns - Data can be loaded into smaller tables for ad-hoc analytics and later moved to another aggregate table for durable

- analytics when ad-hoc analytics is no longer required.
- Data Isolation and Security Management - BigQuery's data access is controlled at the project and dataset level. Data should be partitioned into different datasets for security and isolation.

Flat Schema or Nested/Repeated Schema

Some entity relationships are more naturally expressed as a hierarchy. An order transaction and line items is a perfect example. BigQuery supports nested/repeated fields using the JSON format. Alternatively, hierarchical relationships can be flattened first before loading into BigQuery. Once the data is flattened, then it can be loaded either in a CSV or JSON format. Here are the considerations when deciding between the two schemas:

- Nested/Repeated schema reduces duplications.
- JSON file size is much larger than CSV.
- Nested/Repeated fields require the use of special SQL syntax - FLATTEN and WITHIN, which may not be supported by third-party visualization tools.

Data Format Considerations (1 of 3)

- BigQuery supports three data formats: CSV, JSON, AVRO
- AVRO:
 - Open source format that bundles serialized data and schema in same file
 - Supports flat data and nested/repeated fields
 - Nested/repeated data useful for expressing hierarchical data, reduces duplication when denormalizing data
 - Loads faster in BigQuery if data contains embedded newlines
 - Limited to 16 MB block size

Data Format Considerations (2 of 3)

- JSON (newline-delimited):
 - Supports flat data and nested/repeated fields
 - Nested/repeated data useful for expressing hierarchical data, reduces duplication when denormalizing data
 - Loads faster in BigQuery if data contains embedded newlines
 - One JSON object, including any nested/repeated fields, must appear on each line
 - Supports UTF-8 encoding
 - Limited to 2 MB row size

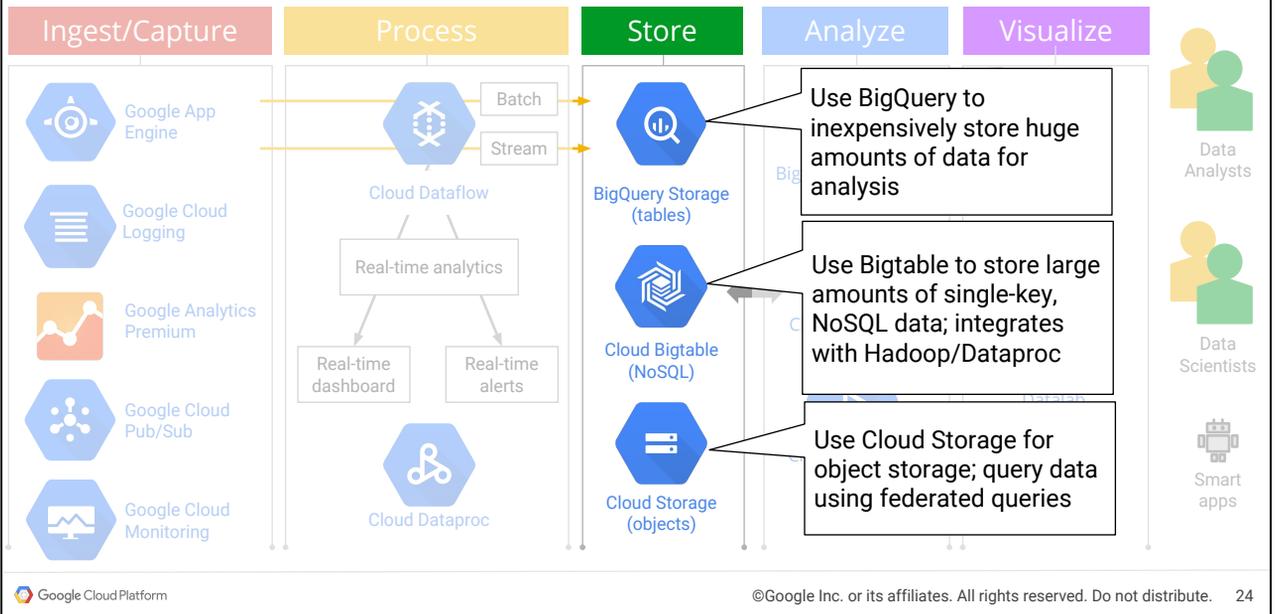
Data Format Considerations (3 of 3)

- CSV:
 - Supports flat data only
 - No nested/repeated data
 - Supports UTF-8 encoding and ISO-8859-1 encoding
 - If loading ISO-8859-1 encoded data, specify `configuration.load.encoding` property when creating load job
 - Limited to 2 MB row and cell size

Agenda

- 1 Ingesting/Capturing Data
- 2 Processing/Transforming Data
- 3 Storing Data
- 4 Federated Queries
- 5 Quiz & Lab

Big Data Lifecycle



Notes:

Google's storage offerings range from Blob, NoSQL and SQL storage depending on what you are trying to do, and it's easy to mix and match.

- For just pure data and blobs, Cloud Storage can deliver what you need. Google has 3 tiers of storage options from warm to cold.
- Recently introduced, Google Cloud Bigtable offers companies a fast, fully managed, infinitely scalable NoSQL database service ideal for web, mobile and IoT applications.

Long-Term Storage in BigQuery

- Automatic discount for data stored longer than 90 days
 - If table data modified, 90-day counter resets
- No need to delete or archive old data
 - Equivalent to cost of Cloud Storage Nearline
- If preferred, store data in Cloud Storage and load into BigQuery when needed
 - No charge for loading/exporting data
 - Loading from Cloud Storage Nearline reduces performance
 - Data in Cloud Storage available to other services

Alternatives to Loading Data (1 of 2)

- **Shared datasets**

- You can run queries against shared datasets without loading the data first

- **Federated data sources**

- Create a table based on an external data source - discussed later

- **Stackdriver log files**

- Export log files (such as App Engine request, application logs) into BigQuery

Alternatives to Loading Data (2 of 2)

- **Public datasets**

- Hosted by Google and partners on BigQuery
- Users can query datasets without worrying about hosting costs
- Data partners can use BigQuery to publish large datasets
- Free 1TB of queries per month

- Learn more at:

cloud.google.com/bigquery/public-data

Agenda

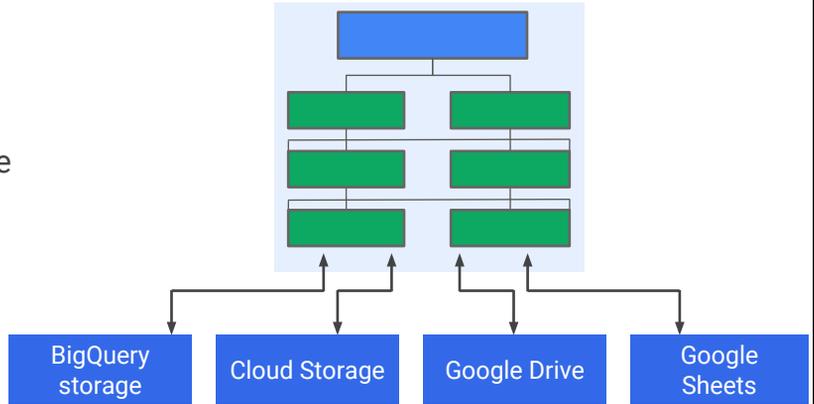
- 1 Ingesting/Capturing Data
- 2 Processing/Transforming Data
- 3 Storing Data
- 4 Federated Queries
- 5 Quiz & Lab

Federated Queries (1 of 3)

- Query data without loading it first

```
SELECT ... FROM bigquery_table bq JOIN gcs_table gcs ON ...  
JOIN gdrive_table gd ON ... JOIN sheets_table sh ON ...
```

- Read data from:
 - Google Drive
 - Google Sheets
 - Google Cloud Storage
- Read data in Avro format
- Automatic schema detection



Federated Queries (2 of 3)

- Query using BigQuery web UI or CLI
- Can load and clean data in one pass
- If working with small dataset that frequently changes, can avoid having to load the latest data before joining with other tables
- BigQuery does not guarantee data consistency for federated data sources
- Impacts query performance

Notes:

For more information on using federated data sources, see:

<https://cloud.google.com/bigquery/federated-data-sources>.

Federated Queries (3 of 3)

- Provide table definition file and URI path
 - Can also use schema auto-detect for CSV and JSON
- Federated query types:
 - *Temporary* – use query-scoped table
 - *Permanent* – create multi-use federated table
- Limits for federated queries are same as for load jobs

Notes:

To directly query a federated data source, provide the Cloud Storage URI paths to your data. For CSV and JSON files, you can also include the table schema using a table definition file. If you omit the schema, BigQuery attempts to automatically detect the schema. For Avro and Cloud Datastore backup files, BigQuery reads the schema from the data source. You have two options for when to specify the URI and schema:

- At table creation time. This option lets you create a permanent, federated table that can be shared with others.
- At query time, using a query-scoped, temporary table. This option is useful for one-time, ad-hoc queries over remote data, or for ETL processes.

Schema Auto-Detect

- For CSV or JSON files, if schema not specified, BigQuery uses schema auto-detect for federated queries
 - Random file selected and scanned (up to 100 rows) to use as representative sample
 - Data types are assigned based on sample values
- Available in CLI and API only
- To see detected schema, use `bq show` command or web UI

Notes:

For CSV and JSON files, you can include the table schema. If you omit the schema, BigQuery attempts to automatically detect the schema. For Avro and Cloud Datastore backup files, BigQuery reads the schema from the data source.

When BigQuery detects schemas, it might, on rare occasions, change a field name to make it compatible with BigQuery SQL syntax. To see the detected schema, use the `bq show` command. Alternately, you can use the BigQuery web UI to see the schema.

If you specify a CSV or JSON file without including a schema description, BigQuery makes a best-effort attempt to automatically infer the schema. Automatic detection is currently available in the command-line tool and the BigQuery API.

BigQuery starts the inference process by selecting a random file in the data set and scanning up to 100 rows of the file to use as a representative sample. BigQuery then examines each field and attempts to assign a data type to that field based on the values in that sample.

Other automatic detection details include:

Compression

BigQuery recognizes gzip-compatible file compression when opening a file.

CSV Delimiter

BigQuery detects the following delimiters:

- commas (',')
- pipes ('|')
- tabs ('\t')

CSV Header

BigQuery infers headers by comparing the first row of the file with other rows in the data set. If the first line contains only strings, and the other lines do not, BigQuery assumes that the first row is a header row.

CSV Quoted new lines

BigQuery detects quoted new line characters within a CSV field and does not interpret the quoted new line character as a row boundary.

Timestamps

BigQuery detects a wide array of timestamp formats, including, but not limited to:

- yyyy-mm-dd in any order
- yyyy-mm-dd hh:mm:ss
- yyyy-mm-dd hh:mm:ss.sss

Other timestamp details:

- Date separators can be "-", "/", or "."
- A time zone can be appended using an offset or name

Agenda

- 1 Ingesting/Capturing Data
- 2 Processing/Transforming Data
- 3 Storing Data
- 4 Federated Queries
- 5 Quiz & Lab

Module Review (1 of 2)

What data formats does BigQuery support?
(select **3** of the available options)

- JSON
- XML
- CSV
- JPG
- AVRO

Module Review (2 of 2)

Which one of the following is true?

- BigQuery federated data sources can include Cloud Storage objects and Bigtable tables
- Compressed files load faster than uncompressed files
- BigQuery supports common CRUD (create, read, update, delete) operations on data
- Load jobs support loading data from Cloud Storage Nearline buckets

Labs

- Part 1: Load data and create federated queries
- Part 2: Export and analyze App Engine logs
- Part 3: Using federated queries and user defined functions

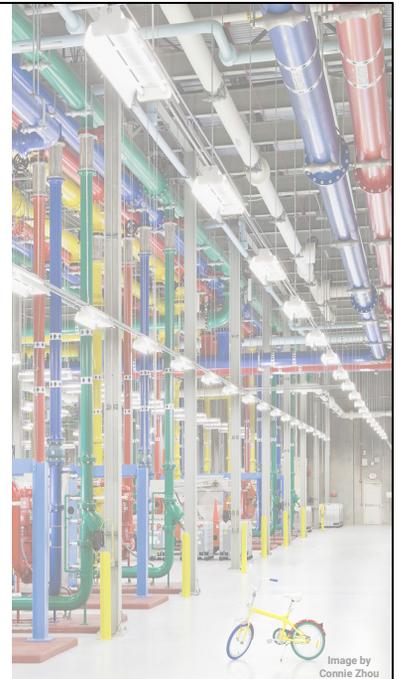


Image by
Cennie Zhou

Resources

- Loading data into BigQuery
<https://cloud.google.com/bigquery/loading-data-into-bigquery>
- Preparing data for BigQuery
<https://cloud.google.com/bigquery/preparing-data-for-bigquery>
- Streaming data into BigQuery
<https://cloud.google.com/bigquery/streaming-data-into-bigquery>
- Release Notes
<https://cloud.google.com/bigquery/release-notes>
- Analyze Google Cloud Storage access logs
<https://cloud.google.com/storage/docs/access-logs>

Module Review Answers (1 of 2)

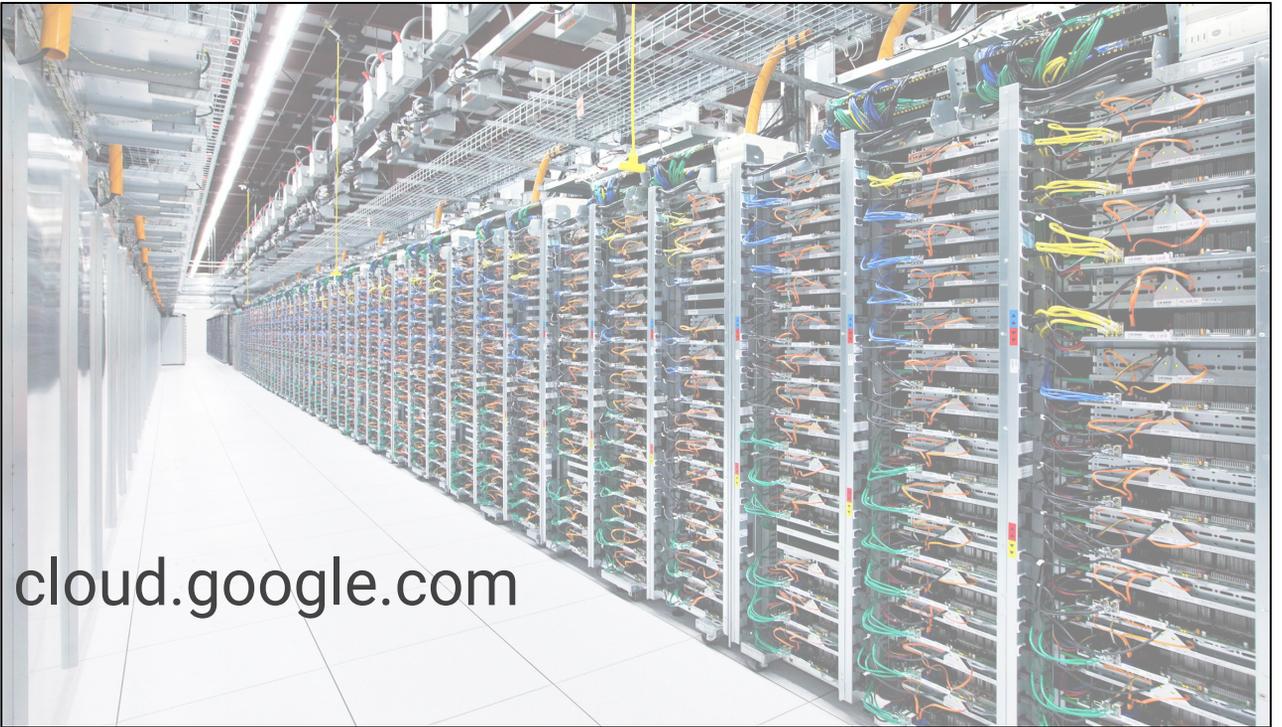
What data formats does BigQuery support?
(select **3** of the available options)

- JSON
- XML
- CSV
- JPG
- AVRO

Module Review Answers (2 of 2)

Which one of the following is true?

- BigQuery federated data sources can include Cloud Storage objects and Bigtable tables
- Compressed files load faster than uncompressed files
- BigQuery supports common CRUD (create, read, update, delete) operations on data
- Load jobs support loading data from Cloud Storage Nearline buckets



cloud.google.com