



Load Data into BigQuery

Next, we'll talk about how to load new data into BigQuery.

The method you use to load data depends on how much transformation is needed

EL

Extract and Load

ELT

Extract, Load, and Transform

ETL

Extract, Transform, and Load

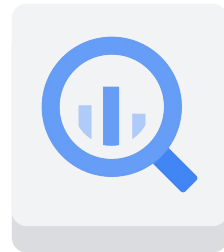
Recall from an earlier module that the method you use to load data depends on how much transformation is needed.

- **E-L, or Extract and Load**, is used when data is imported as-is where the source and target have the same schema.
- **E-L-T, or Extract, Load, Transform**, is used when raw data will be loaded directly into the target and transformed there.
- **E-T-L, or Extract, Transform, Load**, is used when transformation occurs in an intermediate service before it is loaded into the target.

If the data is usable in its original form, just load it



BigQuery
Data Transfer
Service



BigQuery

You might say that the simplest case is E-L. If the data is usable in its original form, there's no need for transformation. Just load it.

Batch load supports different file formats

- CSV
- NEWLINE_DELIMITED_JSON
- AVRO
- DATASTORE_BACKUP
- PARQUET
- ORC

You can batch load data into BigQuery. In addition to CSV, you can also use data files with delimiters other than commas by using the `field_delimiter` flag.

BigQuery supports loading gzip compressed files. However, loading compressed files isn't as fast as loading uncompressed files. For time-sensitive scenarios or scenarios in which transferring uncompressed files to Cloud Storage is bandwidth- or time-constrained, conduct a quick loading test to see which alternative works best.

Because load jobs are asynchronous, you don't need to maintain a client connection while the job is being executed. More importantly, load jobs don't affect your other BigQuery resources.

A load job creates a destination table if one doesn't already exist. BigQuery determines the data schema as follows:

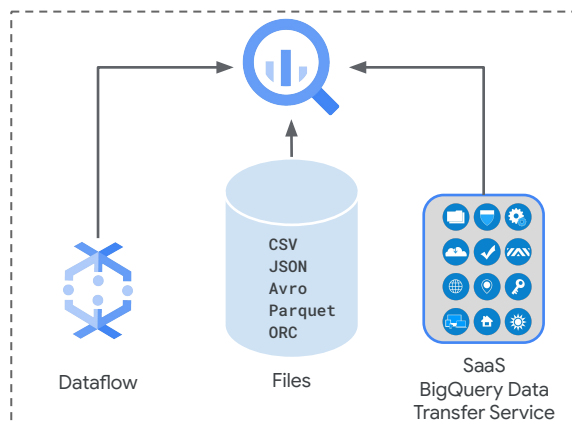
- If your data is in Avro format, which is self-describing, BigQuery can determine the schema directly.
- If the data is in JSON or CSV format, BigQuery can auto-detect the schema, but manual verification is recommended.

You can specify a schema explicitly by passing the schema as an argument to the load job. Ongoing load jobs can append to the same table using the same procedure as the initial load, but do not require the schema to be passed with each job.

If your CSV files always contain a header row that should be ignored after the initial load and table creation, you can use the `skip_leading_rows` flag to ignore the row. For details, see the documentation on BigQuery load flags.

BigQuery sets daily limits on the number and size of load jobs that you can perform per project and per table. In addition, BigQuery sets limits on the sizes of individual load files and records. You can launch load jobs through the BigQuery web UI. To automate the process, you can set up Cloud Functions to listen to a Cloud Storage event that is associated with new files arriving in a given bucket and launch a BigQuery load job.

Most common is loading data into BigQuery tables (batch, periodic)



Loading data into BigQuery tables (batch, periodic) offers the best performance.

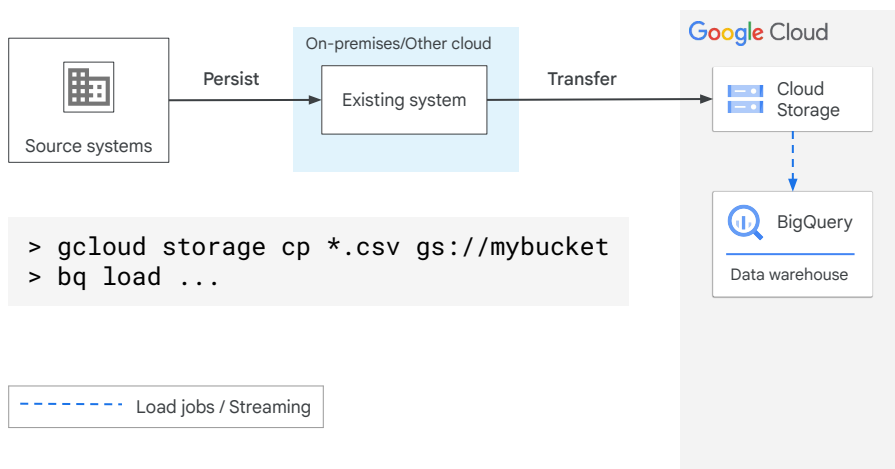
BigQuery can import data stored in the JSON file format, as long as it is newline delimited. It can also import files in Avro, Parquet, and ORC format. The most common import is with CSV files, which are the bridge between BigQuery and spreadsheets.

BigQuery can also directly import Firestore and Datastore export files.

Another way that BigQuery can import data is through the API. Basically, any place where you can get code to run, can theoretically insert data into BigQuery tables. You could use the API from a Compute Engine instance, a container on Kubernetes, App Engine, or from Cloud Functions. However, you would have to recreate the data processing foundation in these cases. In practice, the API is mainly used from either Dataproc or Dataflow.

The BigQuery Data Transfer Service provides connectors and pre-built BigQuery load jobs that perform the transformations necessary to load report data from various services directly into BigQuery.

Loading data through Cloud Storage



Cloud Storage can be useful in the E-L process. You can transfer files to Cloud Storage in the schema that is native to the existing on-premises data storage and then load those files into BigQuery.

Automate the execution of queries based on a schedule

New scheduled query

Details and schedule

Name for scheduled query

daily_schedule


Schedule options

Repeats


Daily

☒ Start now ☐ Schedule start time

☒ End never ☐ Schedule end time

 This schedule will run Every day at 17:46 Europe/London

Destination for query results

 A destination table is required to save scheduled query options.

Project name

qwiklabs-gcp-c710de4945fa7b9c

Dataset name

champions_workshop_models


Table name

scheduled_query_output

Destination table write preference

☒ Append to table ☐ Overwrite table

Notification options

☐ Send email notifications 

Schedule

Cancel

Google Cloud

It is a common practice to automate execution of queries based on a schedule or event and cache the results for later consumption.

You can schedule queries to run on a recurring basis. Scheduled queries must be written in standard SQL, which can include Data Definition Language and Data Manipulation Language statements. The query string and destination table can be parameterized, allowing you to organize query results by date and time.

[\[https://cloud.google.com/bigquery/docs/scheduling-queries\]](https://cloud.google.com/bigquery/docs/scheduling-queries)

BigQuery addresses backup and disaster recovery at the service level (time travel)

```
CREATE OR REPLACE TABLE ch10eu.restored_cycle_stations AS
SELECT
*
FROM bigquery-public-data.london_bicycles.cycle_stations
FOR SYSTEM_TIME AS OF
TIMESTAMP_SUB(CURRENT_TIMESTAMP(), INTERVAL 24 HOUR)
```



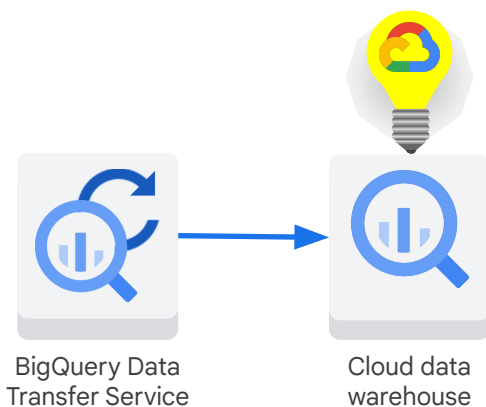
Query a point-in-time snapshot (up to 7 days).
Use it to create a backup.

```
NOW=$(date +%s)
SNAPSHOT=$(echo "($NOW - 120)*1000" | bc)
bq --location=EU cp \
ch10eu.restored_cycle_stations@$SNAPSHOT \
ch10eu.restored_table
```

Restore a 2-min old copy.
(Deleted tables are flushed after 2 days or if recreated with the same name).

- By maintaining a complete 7-day history of changes against your tables, BigQuery allows you to query a point-in-time snapshot of your data. You can easily revert changes without having to request a recovery from backups. This slide shows how to do a SELECT query to query the table as of 24 hours ago. Because this is a SELECT query, you can do more than just restore a table. You can join against some other table or correct the value of individual columns.
- You can also do this using the BigQuery command-line tool as shown in the second snippet. Here, we're restoring data as of 120 seconds ago. You can recover a deleted table only if another table with the same ID in the dataset has not been created. In particular, this means you cannot recover a deleted table if it is being streamed to. Chances are that the streaming pipeline would have already created an empty table and started pushing rows into it. Also be careful using "CREATE OR REPLACE TABLE" because this makes the table irrecoverable.

BigQuery Data Transfer Service helps you build and manage your data warehouse



EL

- Managed service
- Automatic transfers
- Scheduled
- Data staging
- Data processing
- Data backfills

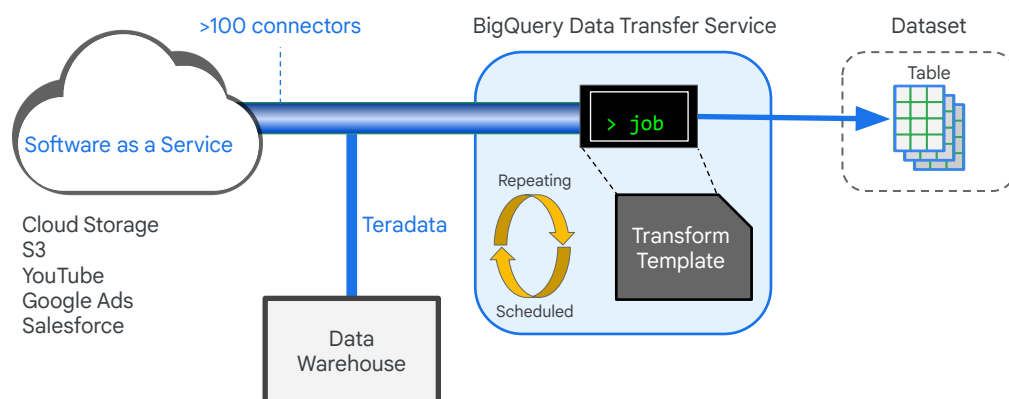
Google Cloud

BigQuery is a managed service, so you don't have the overhead of operating, maintaining or securing the system. A typical Data Warehouse system requires a lot of code for coordination and interfacing. You can get BigQuery Data Transfer Service running without coding. The core of BigQuery Data Transfer Service is scheduled and automatic transfers of data from wherever it is located (in your data center, on other clouds, in SaaS services) in to BigQuery.

Transferring the data is only the first part of building a data warehouse. If you were assembling your own system, you would need to stage the data so that it can be cleaned (data quality), and transformed (E-L-T, extract, load, transform), and processed (put into its final and stable form). A common issue with Data Warehouse systems is late arriving data. For example, a cash register closes late and does not report its daily receipts during the scheduled transfer period. To complete the data, you would need to detect that not all of the data was received, and then request the missing data to fill in the gap. This is called "data backfill" and it is one of the automatic processes provided by BigQuery Data Transfer Service.

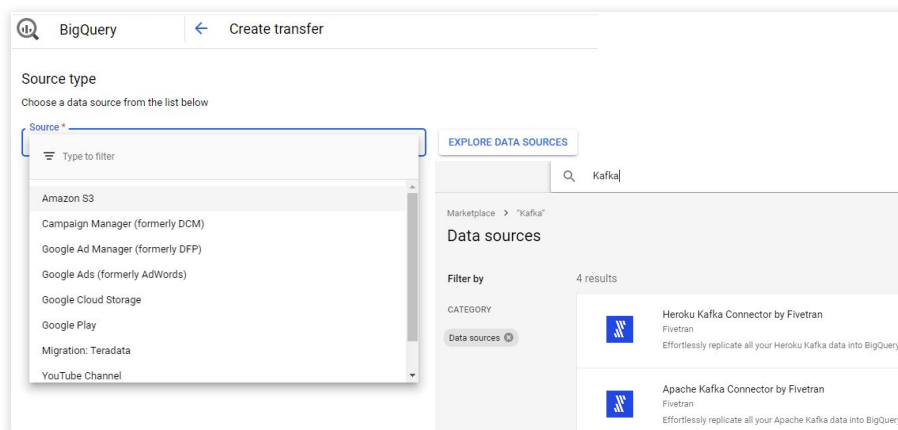
"Backfilling data" means adding missing past data to make a dataset complete with no gaps and to keep all analytic processes working as expected.

BigQuery Data Transfer Service provides SaaS connectors



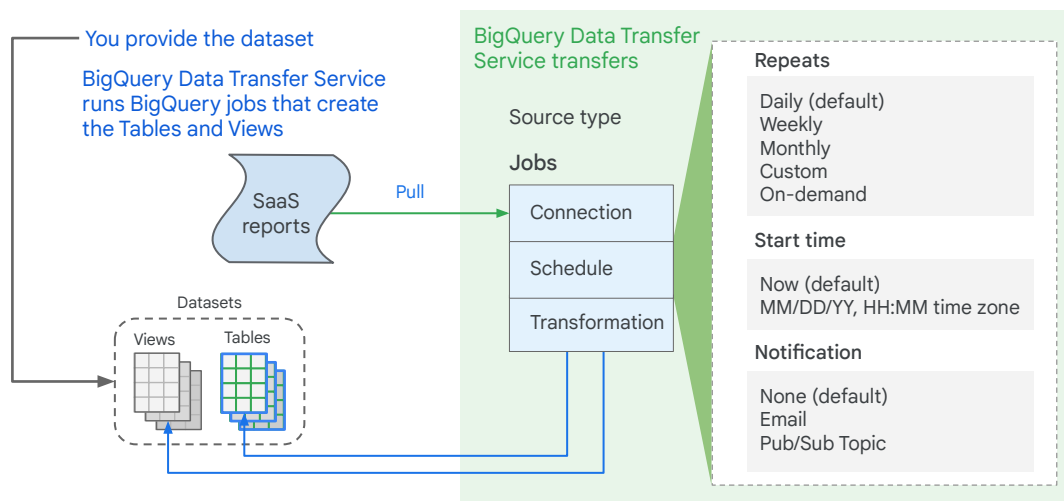
Use the data transfer service for repeated, periodic, scheduled imports of data directly from Software as a Service systems into tables in BigQuery. The BigQuery Data Transfer Service provides connectors, transformation templates, and the scheduling. The connectors establish secure communications with the source service and collect standard data, exports, and reports. This information is transformed within BigQuery. The transformations can be quite complicated, resulting in from 25 to 60 tables. And the transfer can be scheduled to repeat as frequently as once a day.

BigQuery Data Transfer Service supports 100+ SaaS applications



The BigQuery Data Transfer Service can also be used to efficiently move data between regions.

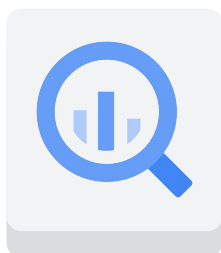
How BigQuery Data Transfer Service transfers work



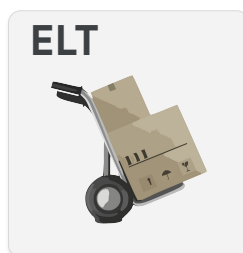
Notice that you don't need Cloud Storage buckets. BigQuery Data Transfer Service runs BigQuery jobs that transform reports from SaaS sources into BigQuery Tables and Views.

Google offers several connectors, including Campaign Manager, Cloud Storage, Amazon S3, Google Ad Manager, Google Ads, Google Play transfers, YouTube channel, YouTube content owner, Teradata migration, and over 100 other connectors through partners.

If the data requires simple transformations, such as scaling, maybe it can be handled in SQL

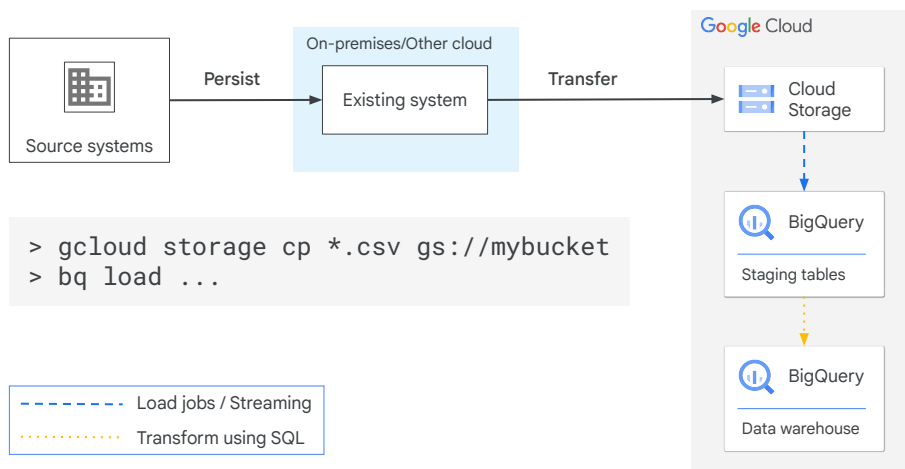


BigQuery



Keep in mind if your data transformations are simple enough, you may be able to do them with just SQL.

Loading and transforming data in BigQuery



If Cloud Storage is part of your workflow, you can load files from Cloud Storage into staging tables in BigQuery first, and then transform the data into the ideal schema for BigQuery by using BigQuery SQL commands.

Modify table data with standard DML statements

INSERT, **UPDATE**, **DELETE**, MERGE records into tables

```
UPDATE table_A
SET
  y = table_B.y,
  z = table_B.z + 1
FROM table_B
WHERE table_A.x = table_B.x
  AND table_A.y IS NULL;
```

```
INSERT INTO table VALUES (1,2,3), (4,5,6), (7,8,9);
```

```
DELETE FROM table WHERE TRUE;
```

BigQuery supports standard DML statements such as insert, update, delete, and merge. There are no limits on D-M-L statements.

However you should not treat BigQuery as an O-L-T-P system. The underlying infrastructure is not structured to perform optimally as an O-L-T-P. There are other more appropriate products on Google Cloud for such workloads.

[<https://cloud.google.com/bigquery/docs/managing-table-schemas>]

Create new tables from data with SQL DDL

```
SELECT
  *
FROM
  movielens.movies_raw
WHERE
  movieId < 5;
```

	movieId	title	genres
0	3	Grumpier Old Men (1995)	Comedy Romance
1	4	Waiting to Exhale (1995)	Comedy Drama Romance
2	2	Jumanji (1995)	Adventure Children Fantasy
3	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy

```
CREATE OR REPLACE TABLE
movielens.movies AS
SELECT
  * REPLACE(SPLIT(genres,
    "|") AS genres)
FROM
  MovieLens.movies_raw;
-- Execute multiple statements.
SELECT * FROM movielens.movies;
```

	movieId	title	genres
0	4	Waiting to Exhale (1995)	[Comedy, Drama, Romance]
1	3	Grumpier Old Men (1995)	[Comedy, Romance]
2	2	Jumanji (1995)	[Adventure, Children, Fantasy]
3	1	Toy Story (1995)	[Adventure, Animation, Children, Comedy, Fantasy]

Question: What's the difference between CREATE OR REPLACE TABLE and CREATE TABLE IF NOT EXISTS ? When would you use each?

BigQuery also supports DDL statements like CREATE OR REPLACE TABLE. In the example on this slide, the replace statement is used to transform a string of genres into an ARRAY. We'll cover ARRAYS in greater detail later in the course.

Custom transformations? BigQuery supports user-defined functions in SQL, JavaScript, and scripting

The screenshot shows the BigQuery Query Editor interface. On the left, the query editor contains the following SQL code:

```

1 CREATE TEMP FUNCTION multiplyInputs(x FLOAT64, y FLOAT64)
2 RETURNS FLOAT64
3 LANGUAGE js AS """
4   return x*y;
5   """;
6 WITH numbers AS
7   (SELECT 1 AS x, 5 as y
8    UNION ALL
9    SELECT 2 AS x, 10 as y
10   UNION ALL
11   SELECT 3 as x, 15 as y)
12 SELECT x, y, multiplyInputs(x, y) as product
13 FROM numbers;

```

On the right, a panel shows the query results. The status bar indicates "Query complete (2.5 sec elapsed, 0 B processed)". The "Results" tab is selected, showing a table with 3 rows and 4 columns: Row, x, y, and product.

Row	x	y	product
1	1	5	5.0
2	2	10	20.0
3	3	15	45.0

At the bottom of the query editor, there are buttons for "Run", "Save query", "Save view", "Schedule query", and "More".

Google Cloud

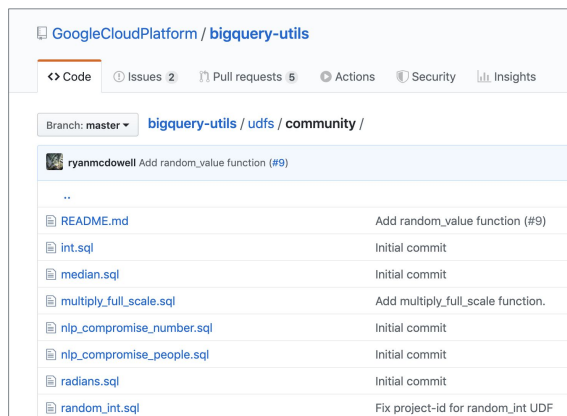
Lastly, what if your transformations went beyond what functions were currently available in BigQuery? Well, you can create your own!

BigQuery supports user-defined functions, or UDF. A UDF enables you to create a function using another SQL expression or an external programming language. JavaScript is currently the only external language supported. We strongly suggest you use Standard SQL though, because BigQuery can optimize the execution of SQL much better than it can for JavaScript.

UDFs allow you to extend the built-in SQL functions. UDFs take a list of values, which can be ARRAYs or STRUCTs, and return a single value, which can also be an ARRAY or STRUCT. UDFs written in JavaScript can include external resources, such as encryption or other libraries.

Previously, UDFs were temporary functions only. This meant you could only use them for the current query or command-line session. Now we have permanent functions, scripts, and procedures in beta but they might even be generally available by the time you are seeing this. Please check the documentation.

You can persist and share your UDF objects with other team members or publically



The BigQuery team has a public GitHub repo for common User Defined Functions

<https://github.com/GoogleCloudPlatform/bigquery-utils/tree/master/udfs/community>

When you create a UDF, BigQuery persists it and stores it as an object in your database. What this means is you can share your UDFs with other team members or even publically if you wanted to. The BigQuery team has a public GitHub repo for common User Defined Functions at the link you see here.

[\[https://cloud.google.com/blog/products/data-analytics/new-persistent-user-defined-functions-increased-concurrency-limits-gis-and-encryption-functions-and-more\]](https://cloud.google.com/blog/products/data-analytics/new-persistent-user-defined-functions-increased-concurrency-limits-gis-and-encryption-functions-and-more)