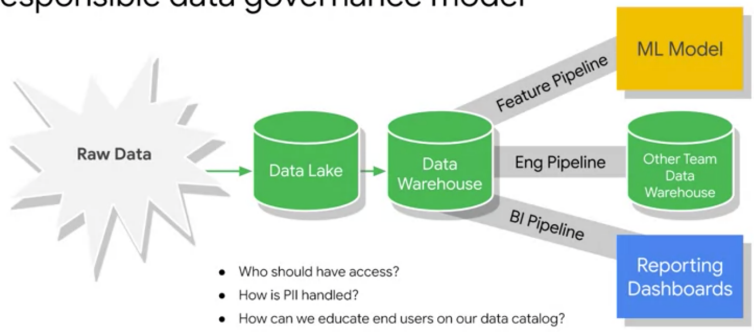
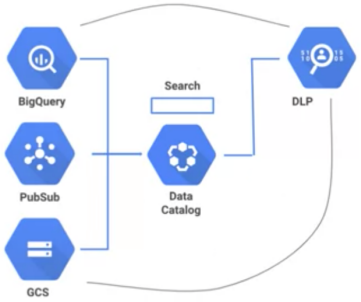
A data engineer manages data access and governance

Data engineering must set and communicate a responsible data governance model



Cloud Data Catalog is a managed data discovery + Data Loss Prevention API for guarding PII



Data Catalog

Simplify data discovery at any scale:
Fully managed metadata management service with no infrastructure to set up or manage

Unified view of all datasets:
Central and secure data catalog across Google Cloud with metadata capture and tagging

Data governance foundation:
Security compliance with access level controls along with Cloud Data Loss Prevention integration for handling sensitive data

Cloud SQL is fully managed SQL Server, Postgres, or mySQL for your Relational Database (transactional RDBMS)

- Automatic encryption
- 30TB storage capacity
- 60,000 IOPS (read/write per second)
- Auto-scale and auto backup

Why not use simply use Cloud SQL for reporting workflows?

RDBMS are optimized for data from a single source and high-throughput writes vs. high-read data warehouses

Cloud SQL

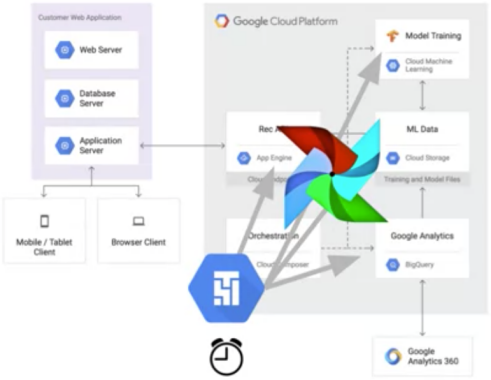
You will likely need and encounter both a database and data warehouse in your final architecture

BigQuery

- Scales to GB and TB
- Ideal for back-end database applications
- Record based storage

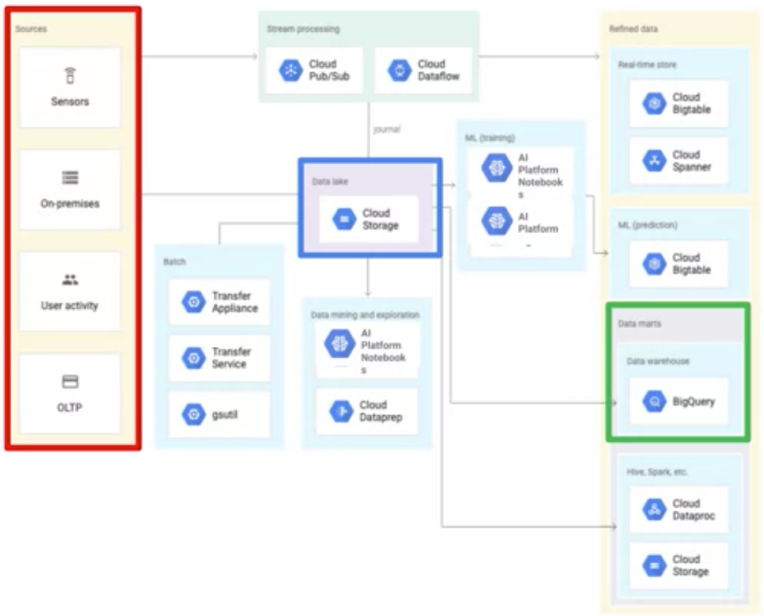
- Scales to PB
- Easily connect to external data sources for ingestion
- Column based storage

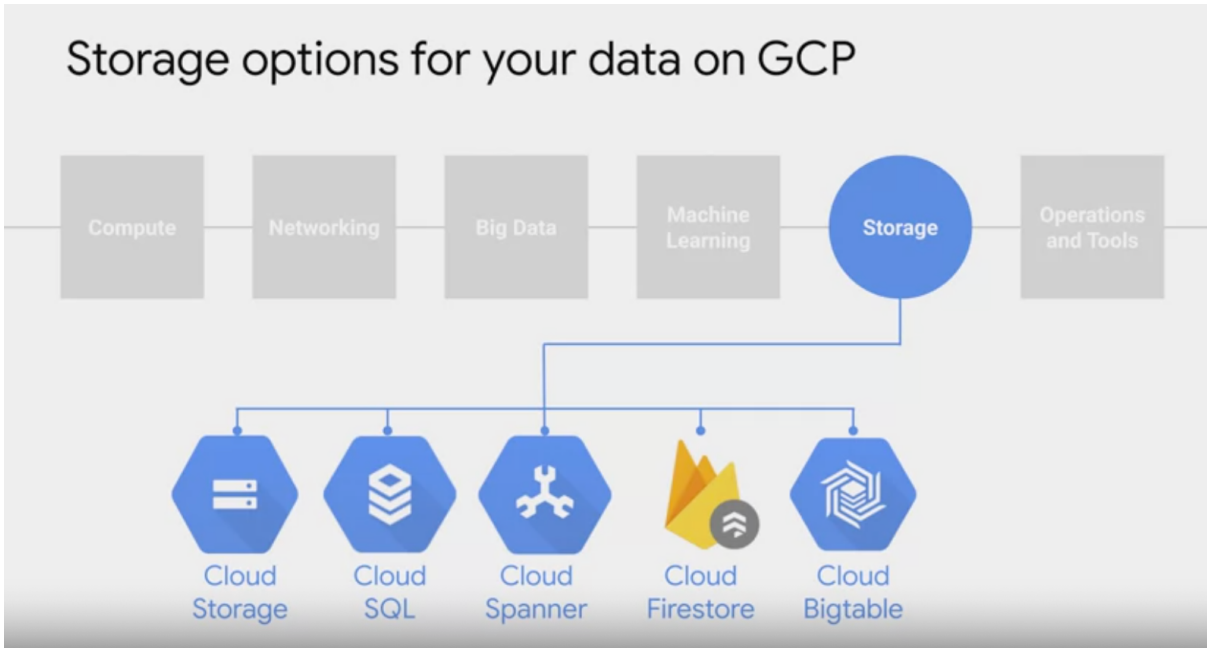
Cloud Composer (managed Apache Airflow) is used to orchestrate production workflows



Concept Review:

Data sources feed into a **Data Lake** and are processed into your **Data Warehouse** for analysis





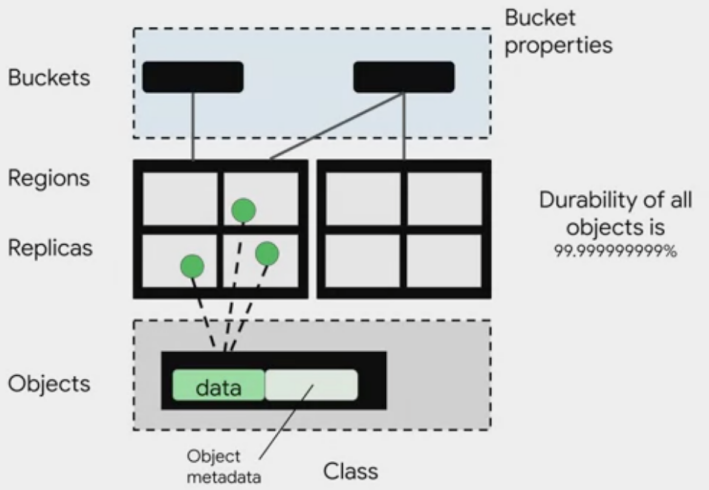
How does Cloud Storage work?

Single global namespace simplifies locating buckets and objects

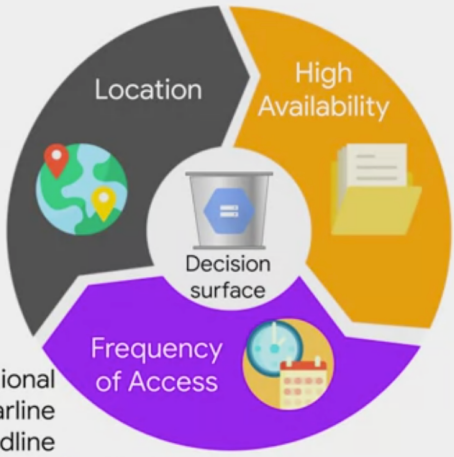
Location to control latency

Durability and availability

Long object names simulate structure



Bucket properties depend on your requirements



europe-north1
asia-south1
eu
asia
nam4 (us-central1, us-east1)A

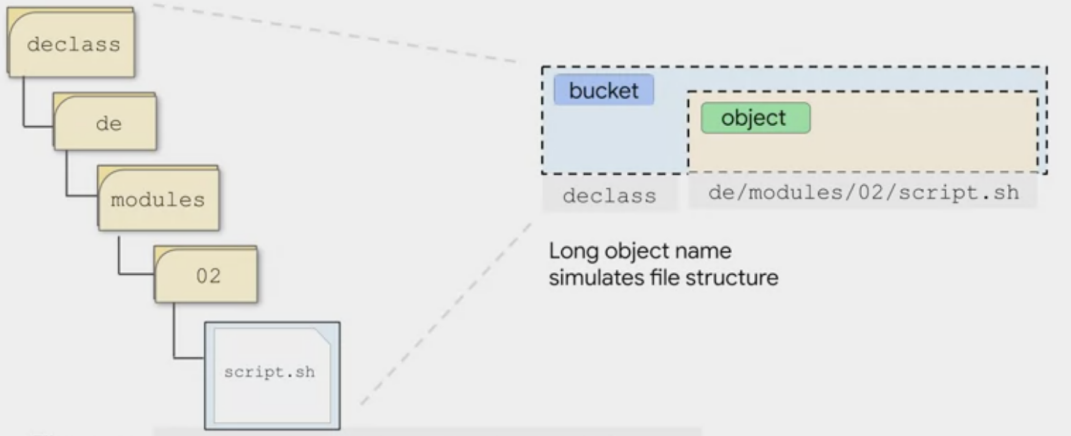
single-region
multi-region
dual-region

https://cloud.google.com/storage/docs/storage-classes

Regional/Multi-Regional
Nearline
Coldline

https://cloud.google.com/storage/docs/locations#location-dr

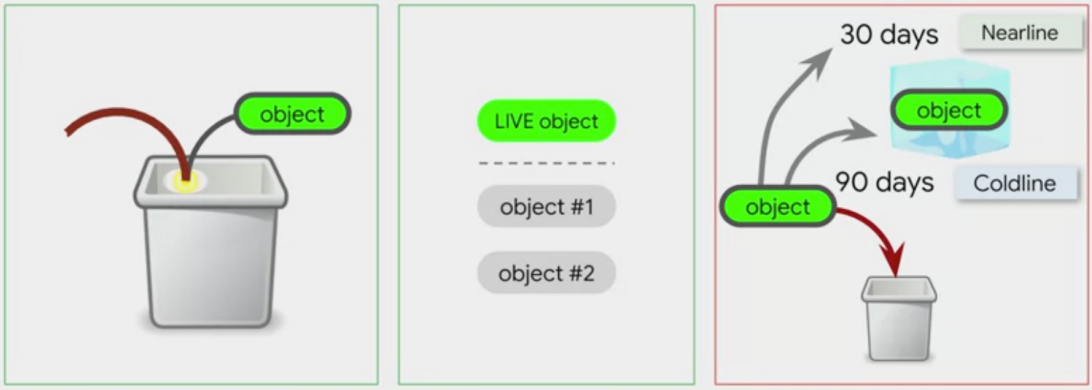
Cloud Storage simulates a file system



File access `gs://declass/de/modules/02/script.sh`

Web access `https://storage.cloud.google.com/declass/de/modules/02/script.sh`

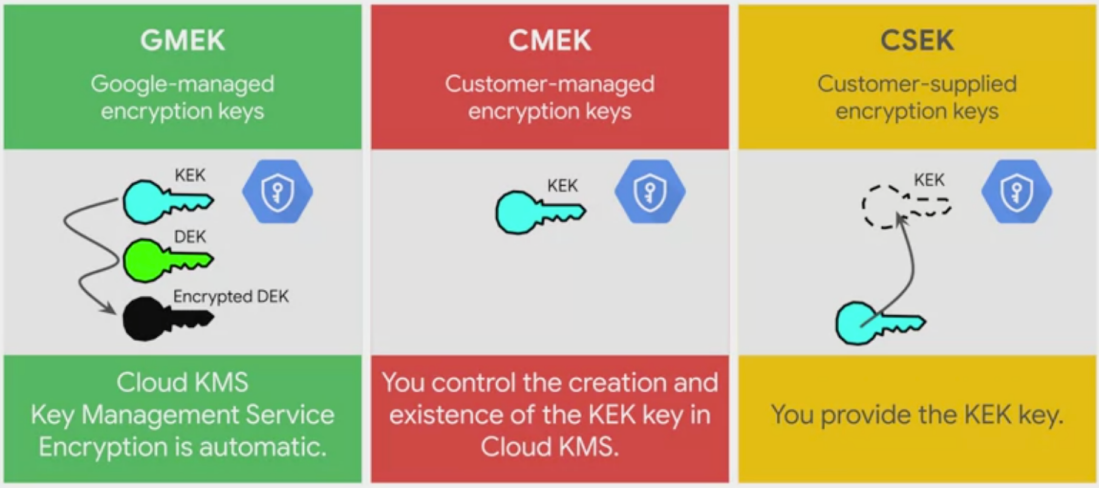
Cloud Storage has many object management features

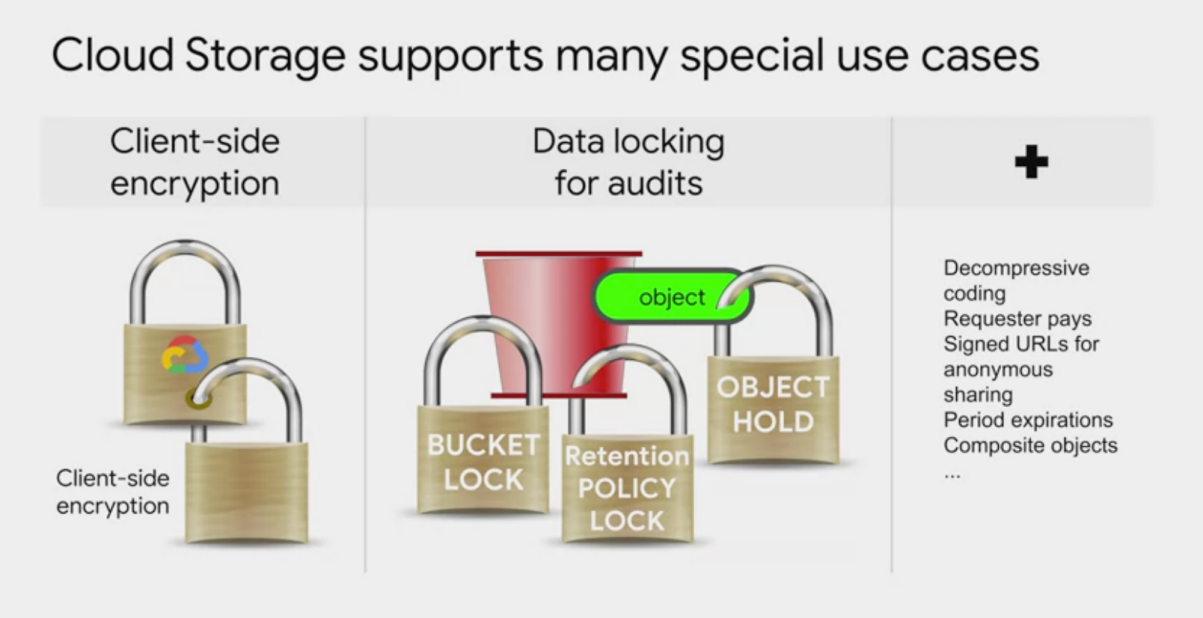


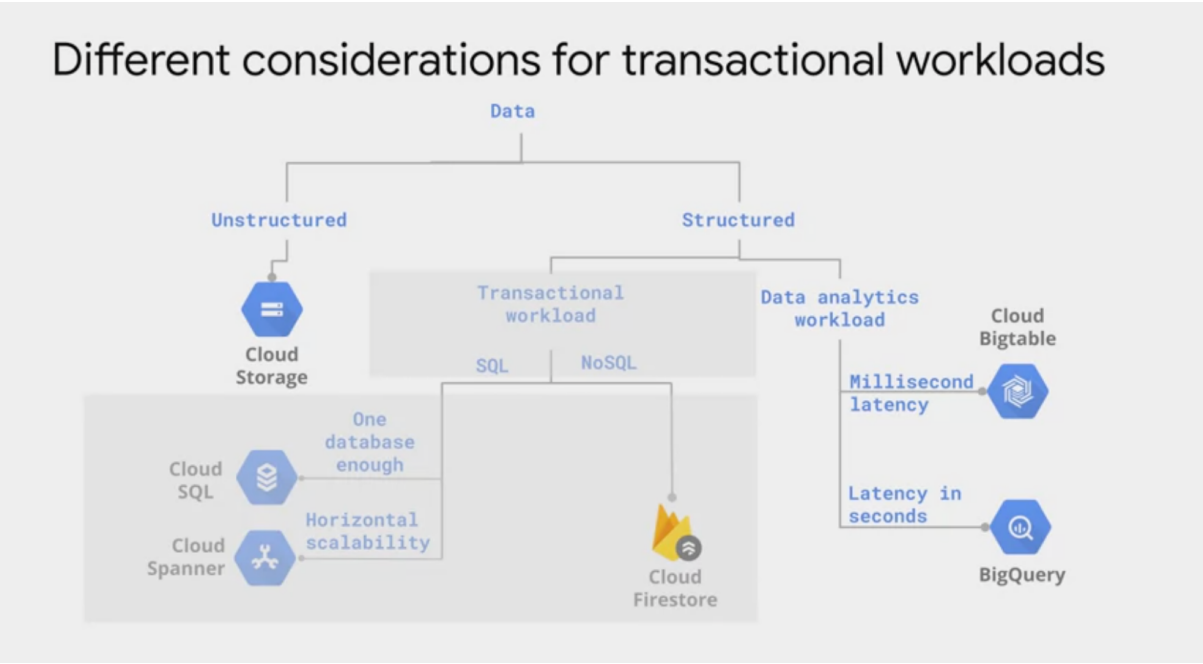
Retention Policy | Versioning | Lifecycle Management

Controlling access with Cloud IAM and access lists

Data encryption options for many requirements

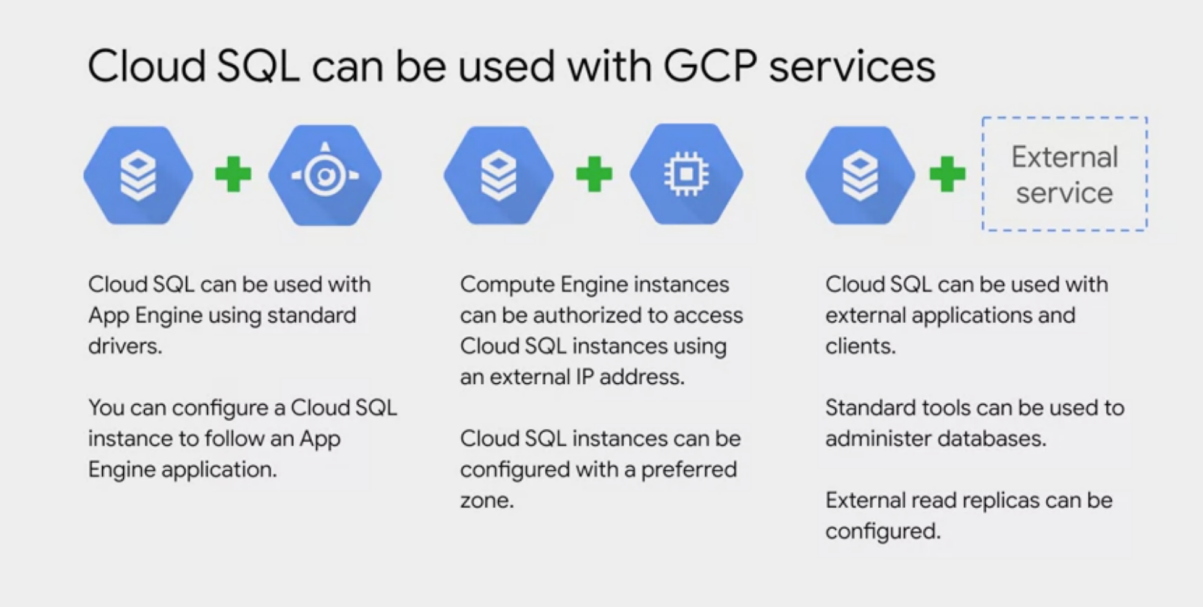






On Google Cloud, a data warehouse option that's very popular tends to be BigQuery.
There is a limit to the size of the data that can be loaded directly into BigQuery. This is because your network might affect that bottleneck.
Rather than load the data directly into BigQuery, it can be much more convenient to first stage it and load it into Google Cloud Storage is that data lake, and then load Google Cloud Storage through a pipeline into BigQuery as your data warehouse.
This is because Google Cloud Storage supports a multithreaded resumable loads of data.
If you're using gsutil, you can literally just provide a -m option there.
Loading data from Cloud Storage will also be much faster because of the high throughput it offers.

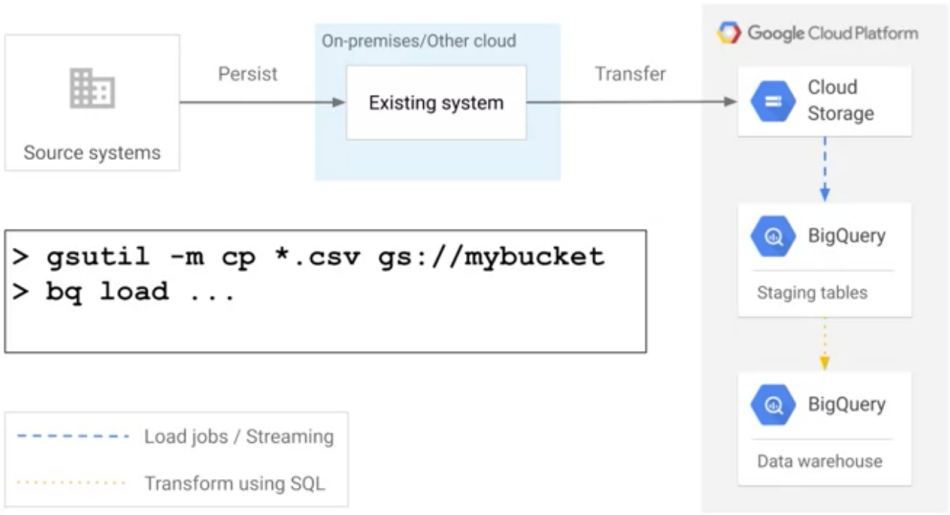
A feature that has recently been released is the ability for BigQuery to query data files that live in Google Cloud Storage, directly query them without having to first load those files into BigQuery zone native storage. This is called a **federated query** or creating an external data source connection.
A super popular use case is if you have your data files in a variety of formats like CSV, Avro, newly supported you can do Parquet or Apache ORC, and then query them directly into BigQuery.



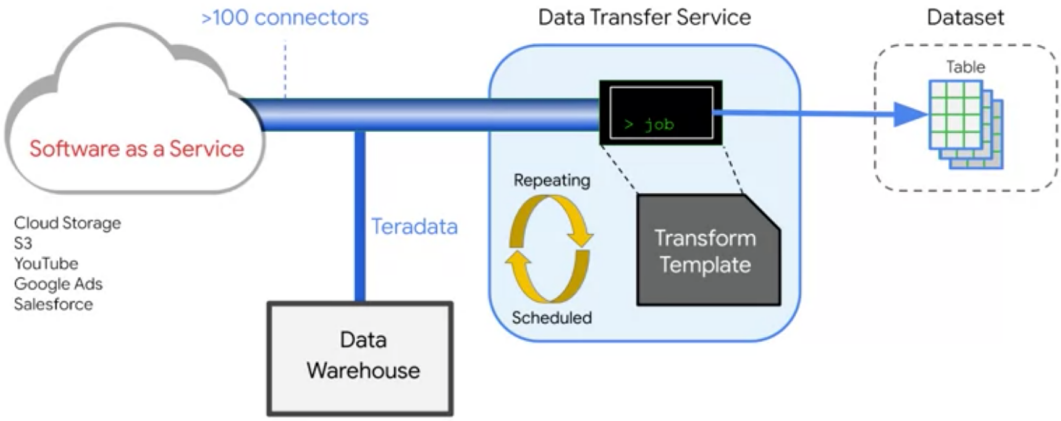
We kept saying that cloud SQL is fully managed. You might also seen us use the word serverless to describe like **BigQuery**.

For example, what's the difference between fully managed and serverless?
By fully managed, we mean that the service runs on a hardware that you can control, you can SSH into a Cloud SQL instance for example. That said, Google does help you manage that instance by automating backups and setting over those failover instances that we just talked about. Serverless is the next step up, you can treat a serverless product that's just like an API that you're calling, sure you pay for using the product, but you don't have to worry about or manage any of the servers, for all you know, it could be completely manual behind the scenes, you just get the benefits of the output. BigQuery is serverless, as is cloud pub/sub for asynchronous messaging and data flow for parallel data processing, you can think of cloud storage as being serverless as well. Sure cloud storage uses discs, but you never actually have to interact with any of the hardware.

Loading data through Cloud Storage



Data Transfer Service provides SaaS connectors



Introduction to ARRAYs and STRUCTs

Store complex data with nested fields (ARRAYS)

Row	order_id	service_type	payment_method	event.status	event.time	pickup.latitude	pickup.longitude	destination.latitude	destination.longitude	total_distance_km	pricing_type
151	FD-5117	GO_FOOD	GOPAY	CREATED	2018-12-31 04:44:02.545210 UTC	-7.75105	110.410561	-7.7430367	110.4046433	1.56	regular
				COMPLETED	2018-12-31 05:06:27.897769 UTC						
				PICKED_UP	2018-12-31 04:48:25.945331 UTC						
				DRIVER_FOUND	2018-12-31 04:44:06.869010 UTC						
152	FD-6834	GO_FOOD	CASH	PICKED_UP	2018-12-31 12:49:52.518880 UTC	1.121272	104.049739	1.1368655	104.03322	4.84	surge
				DRIVER_FOUND	2018-12-31 12:40:14.214843 UTC						
				COMPLETED	2018-12-31 13:04:00.291780 UTC						
				CREATED	2018-12-31 12:40:13.431094 UTC						
153	FD-6293	GO_FOOD	PARTIAL_PAYMENT	PICKED_UP	2018-12-31 04:33:11.856445 UTC	-7.9657554	112.6247491	-7.9384084	112.6227862	4.68	regular
				COMPLETED	2018-12-31 04:56:05.885521 UTC						
				CREATED	2018-12-31 04:16:24.356539 UTC						
				DRIVER_FOUND	2018-12-31 04:16:25.643766 UTC						
154	FD-7817	GO_FOOD	CASH	COMPLETED	2018-12-31 09:14:44.897136 UTC	-6.353915	106.247312	-6.368896	106.25787	3.51	regular
				PICKED_UP	2018-12-31 09:01:11.471274 UTC						
				CREATED	2018-12-31 08:40:31.821796 UTC						
				DRIVER_FOUND	2018-12-31 08:40:32.910319 UTC						

Table JSON

First < Prev Rows 151 - 154 of 1137 Next > Last

Practice reading the new schema

- Practice reading the new schema
- Spot the STRUCTS
- Type RECORD = STRUCTS

booking

Schema Details Preview

Field name	Type	Mode
order_id	STRING	NULLABLE
service_type	STRING	NULLABLE
payment_method	STRING	NULLABLE
event	RECORD	REPEATED
event. status	STRING	NULLABLE
event. time	TIMESTAMP	NULLABLE
pickup	RECORD	NULLABLE
pickup. latitude	FLOAT	NULLABLE
pickup. longitude	FLOAT	NULLABLE
destination	RECORD	NULLABLE
destination. latitude	FLOAT	NULLABLE
destination. longitude	FLOAT	NULLABLE
total_distance_km	FLOAT	NULLABLE
pricing_type	STRING	NULLABLE
duration	RECORD	NULLABLE
duration. booking_to_dispatch	FLOAT	NULLABLE
duration. booking_to_pickup	FLOAT	NULLABLE

Events

Pick-ups

Destination

Duration

Practice reading the new schema

- Practice reading the new schema
- Spot the ARRAYS
- REPEATED = ARRAY

booking

Schema | Details | Preview

Field name	Type	Mode
order_id	STRING	NULLABLE
service_type	STRING	NULLABLE
payment_method	STRING	NULLABLE
event	RECORD	REPEATED
event. status	STRING	NULLABLE
event. time	TIMESTAMP	NULLABLE
pickup	RECORD	NULLABLE

Events

Row	order_id	service_type	payment_method	event.status	event.time
151	FD-5117	GO_FOOD	GOPAY	CREATED	2018-12-31 04:44:02.545210 UTC
				COMPLETED	2018-12-31 05:06:27.897769 UTC

Status and Time are ARRAYs within the Event STRUCT

Recap

- STRUCTS (RECORD)
- ARRAYS (REPEATED)
- ARRAYS can be part of regular fields or STRUCTS
- A single table can have many STRUCTS

booking

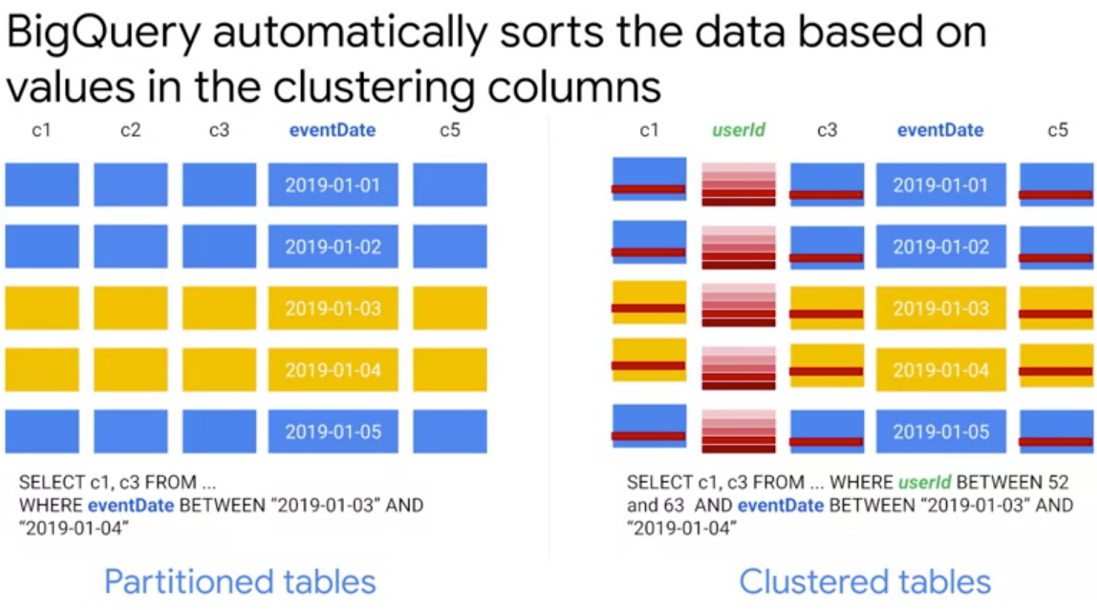
Schema | Details | Preview

Field name	Type	Mode
order_id	STRING	NULLABLE
service_type	STRING	NULLABLE
payment_method	STRING	NULLABLE
event	RECORD	REPEATED
event. status	STRING	NULLABLE
event. time	TIMESTAMP	NULLABLE
pickup	RECORD	NULLABLE
pickup. latitude	FLOAT	NULLABLE
pickup. longitude	FLOAT	NULLABLE
destination	RECORD	NULLABLE
destination. latitude	FLOAT	NULLABLE
destination. longitude	FLOAT	NULLABLE
total_distance_km	FLOAT	NULLABLE
pricing_type	STRING	NULLABLE
duration	RECORD	NULLABLE
duration. booking_to_dispatch	FLOAT	NULLABLE
duration. booking_to_pickup	FLOAT	NULLABLE

You may have wondered why the field alias `hit.page.pageTitle` looks like three fields in one separated by periods. Just as ARRAY values give you the flexibility to *go deep* into the granularity of your fields, another data type allows you to *go wide* in your schema by grouping related fields together. That SQL data type is the STRUCT data type.
The easiest way to think about a STRUCT is to consider it conceptually like a separate table that is already pre-joined into your main table.

A STRUCT can have:

- one or many fields in it
- the same or different data types for each field
- it's own alias



BigQuery supports three ways of partitioning tables

Ingestion time

```
bq query --destination_table mydataset.mytable
--time_partitioning_type=DAY
...
```

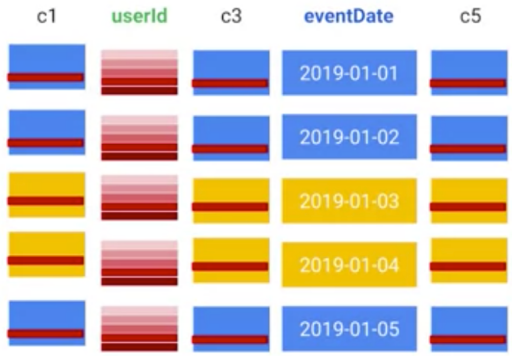
Any column that is of type DATE or TIMESTAMP

```
bq mk --table --schema a:STRING,tm:TIMESTAMP
--time_partitioning_field tm
```

Integer-typed column

```
bq mk --table --schema "customer_id:integer,value:integer"
--range_partitioning=customer_id,0,100,10 my_dataset.my_table
```

Set up clustering at table creation time




```
CREATE TABLE mydataset.myclusteredtable
 (
   c1 NUMERIC,
   userId STRING,
   c3 STRING,
   eventDate TIMESTAMP,
   C5 GEOGRAPHY
  )
 PARTITION BY DATE(eventDate)
 CLUSTER BY userId
 OPTIONS (
   partition_expiration_days=3,
   description="cluster")
 AS SELECT * FROM mydataset.myothertable
```