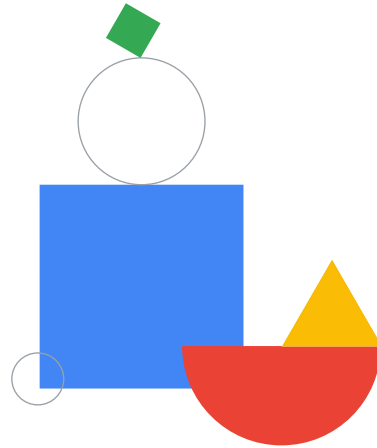


Building a Data Lake



Welcome to the module on building a data lake.

Agenda

Introduction to Data Lakes

Data Storage and ETL Options on Google Cloud

Building a Data Lake Using Cloud Storage

Securing Cloud Storage

Storing All Sorts of Data Types

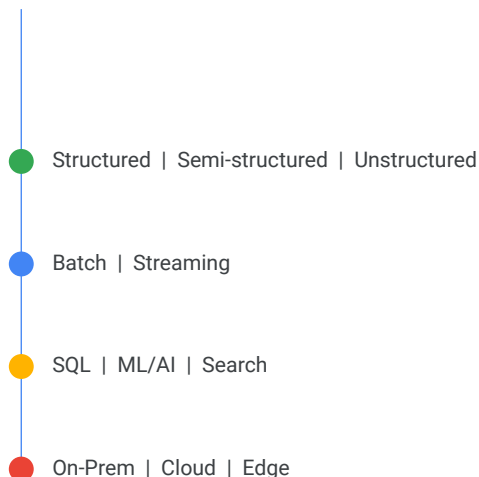
Cloud SQL as a Relational Data Lake

Lab: Loading Taxi Data Into Cloud SQL

Let's start with a discussion about what data lakes are where they fit in as a critical component of your overall data engineering ecosystem.

What is a data lake?

A scalable and secure data platform that allows enterprises to **ingest**, **store**, **process**, and **analyze** any type or volume of information.



What is a data lake after all?

It's a fairly broad term, but it generally describes a place where you can securely store various types of data of all scales, for processing and analytics.

Data lakes are typically used to drive data analytics, data science and ML workloads, or batch and streaming data pipelines.

Data lakes will accept all types of data.

Finally, data lakes are portable, on-premise or in cloud.

Components of a Data Engineering ecosystem

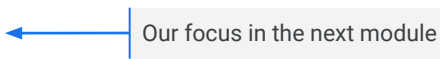
- Data sources
- Data sinks
 - Central Data Lake repository ← Our focus in this module
 - Data Warehouse
- Data pipelines (batch and streaming)
- High-level orchestration workflows

Here's where Data Lakes fit into the overall data engineering ecosystem for your team.

You have to start with some originating system or systems that are the source of all your data -- those are your Data Sources. Then, as a data engineer, you need to build reliable ways of retrieving and storing that data -- those are your data sinks. The first line of defense in an enterprise data environment is your data lake. Again it's the central "give me whatever data at whatever volume, variety of formats, and velocity you got" I can take it. We'll cover a the key considerations and options for building a data lake in this module.

<https://cloud.google.com/solutions/data-lake/>

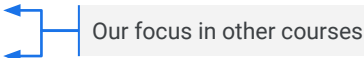
Components of a Data Engineering ecosystem

- Data sources
 - Data sinks
 - Central Data Lake repository
 - **Data Warehouse**
 - Data pipelines (batch and streaming)
 - High-level orchestration workflows
- 

Once your data is off the source systems and inside your environment -- generally a ton of cleanup and processing is required to massage that data into a useful format for the business. It will then end up in your Data Warehouse. That's our focus for the next module.

<https://cloud.google.com/solutions/data-lake/>

Components of a Data Engineering ecosystem

- Data sources
 - Data sinks
 - Central Data Lake repository
 - Data Warehouse
 - Data pipelines (batch and streaming)
 - High-level orchestration workflows
- 
- Our focus in other courses

What actually performs the cleanup and processing of data? Those are your data pipelines! They are responsible for doing the transformations and processing on your data at scale and bring your entire system to life with fresh newly processed data for analysis.

An additional abstraction layer above your pipelines is what I will call your entire workflow. You will often need to coordinate efforts between many different components at a regular or event-driven cadence. While your data pipeline may process data from your lake to your warehouse, your orchestration workflow may be the one responsible for kicking off that data pipeline in the first place when it noticed that there was new raw data available from a source.

<https://cloud.google.com/solutions/data-lake/>

Data Engineering is like Civil Engineering



- 1 Raw Materials need to be brought to the job site (into the Data Lake).

Before we move into what cloud products can fit what roles, I want to leave you with an analogy that has helped me disambiguate these components.

Picture yourself in the world of civil engineering for a moment. You're tasked with building an amazing skyscraper in a downtown city. Before you break ground, you need to ensure you have all the **raw materials** that you're going to need to achieve your end objective. Sure -- some materials could be sourced later in the project but let's keep this example simple. The act of bringing the steel, the concrete, the water, the wood, the sand, the glass from wherever source elsewhere in the city onto your construction site is analogous to data coming from source systems and into your lake.

Transform raw materials into a useful form



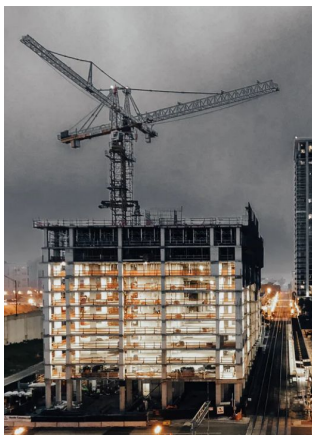
1 Raw Materials need to be brought to the job site (into the Data Lake).

2 Materials need to be cut and transformed for purpose and stored (pipelines to data sinks).



Great! Now you have all these raw materials but you can't use them as-is to build your building. You need to cut the wood and metal, measure and format the glass before it is suited for the purpose of building the building. The end-result, the cut glass, shaped metal, that is the formatted data that is stored in your data warehouse. It's ready to be used to directly add value to your business -- which in our analogy is building the building. How did you transform these raw materials into useful pieces? On a construction site that's the job of the worker. As you'll see later when we talk about Data Pipelines, the individual unit behind the scenes is literally called a worker (which is just a VM) and it takes some small piece of data and transforms it for you.

The new building is the new insight, ML model, etc.



- 1 Raw Materials need to be brought to the job site (into the Data Lake).
- 2 Materials need to be cut and transformed for purpose and stored (pipelines to data sinks).
- 3 The actual building is the new insight or ML model etc.

What about the building itself? That's whatever end-goal or goals you have for this engineering project. In the data engineering world, the shiny new building could be a brand new analytical insight that wasn't possible before, or a machine learning model, or whatever else you want to achieve now that you have the cleaned data available.

An orchestrator governs all aspects of the workflow



- 1 Raw Materials need to be brought to the job site (into the Data Lake).
- 2 Materials need to be cut and transformed for purpose and stored (pipelines to data sinks).
- 3 The actual building is the new insight or ML model etc.
- 4 The supervisor directs all aspects and teams on the project (workflow orchestration).

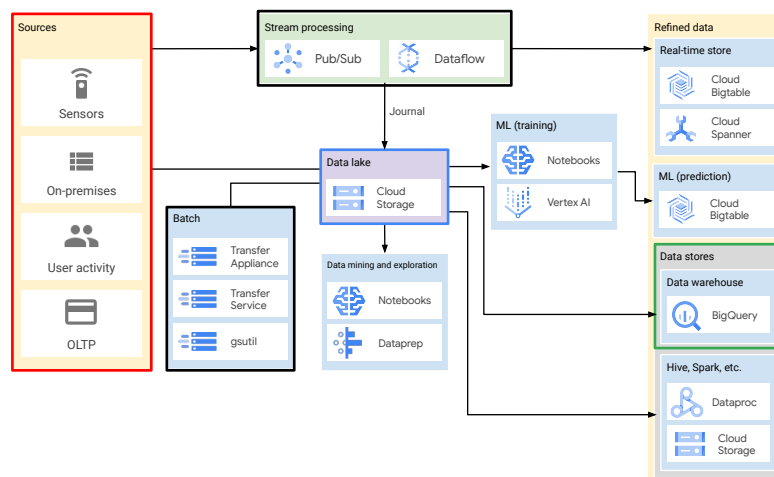
Google Cloud

The last piece of the analogy is the orchestration layer. On a construction site you have a manager or a supervisor that directs when work is to be done and any dependencies. They could say “once the new metal gets here, send it to this area of the site for cutting and shaping, and then alert this other team that it’s available for building”. In the data engineering world that’s your orchestration layer or overall workflow. So you might say “Everytime a new piece of CSV data drops into this Cloud Storage bucket I want you to automatically pass it to our data pipeline for processing AND once it’s done processing, I want you -- the pipeline -- to stream it into the data warehouse AND once it’s in the data warehouse, I will notify the machine learning model that new cleaned training data is available for training and direct it to start training a new model version.” Can you see the graph of actions building? What if one step fails? What if you want to run that every day? You’re beginning to see the need for an orchestrator which, in our solutioning, will be Apache Airflow running on Cloud Composer later.

<https://cloud.google.com/solutions/data-lake/>

Example architecture

1. **Data sources**
2. **Data Lake**
3. **Data Pipelines**
4. **Data Warehouse**
5. Used for ML and analytics workloads



Google Cloud

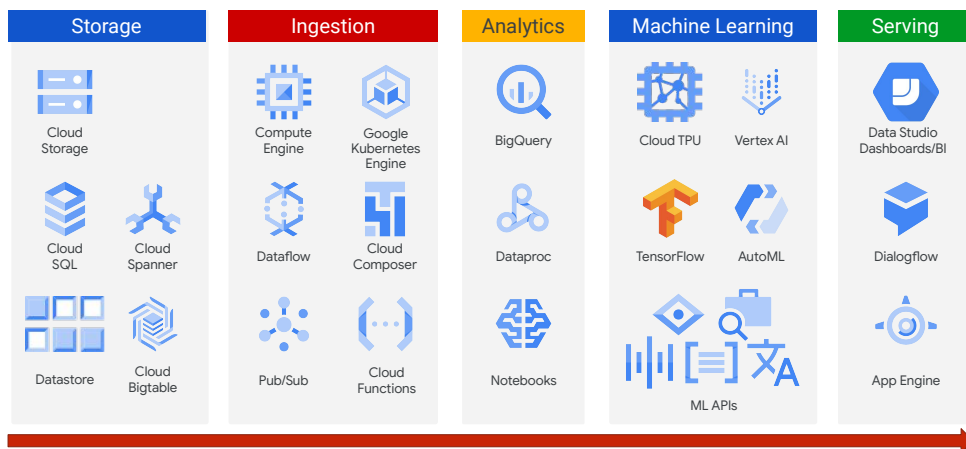
Let's bring back one example solution architecture diagram that you saw earlier in the course.

The data lake here is Google Cloud Storage buckets right in the center of the diagram. It's your consolidated location for raw data that is durable and highly available. In this example our Data Lake is a Google Cloud Storage but that doesn't mean Google Cloud Storage is your only option for Data Lakes. I'll say it again, Cloud Storage is one of a few good options to serve as a Data Lake -- in other examples we will look at, BigQuery may be your Data Lake AND your Data Warehouse and you don't use Cloud Storage buckets at all. This is why it's so important to first understand what you want to do first and then finding which solutions best meet your needs.

Regardless about which cloud tools and technologies you use -- your data lake generally serves as that single consolidated place for all your raw data. I like to think of it as a durable staging area. The data may end up in many other places like a transformation pipeline that cleans it up, moves it to the warehouse, and then it's read by a machine learning model BUT it all starts with getting that data into your Lake first.

Let's do a quick overview on some of the core Google Cloud Big Data products that you need to know as a data engineer and will practice later in your labs.

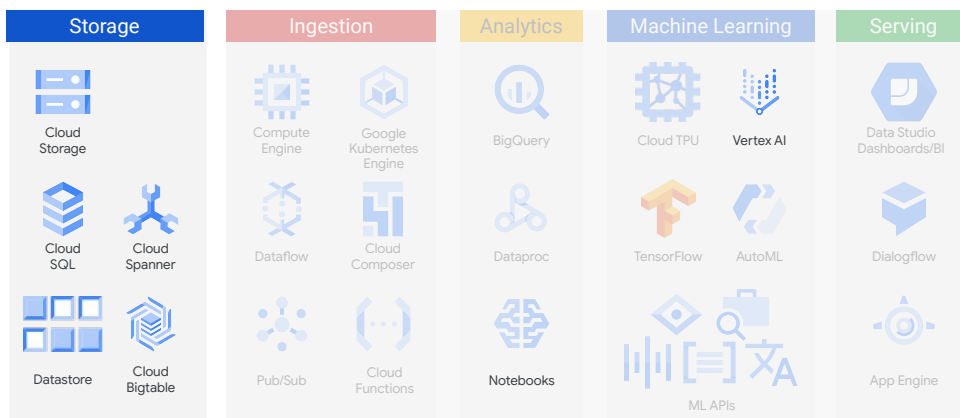
The suite of big data products on Google Cloud



Google Cloud

Here is the complete list of big data and ML products organized by where you would likely find them in a typical data processing workload from storing the data on the left, to ingesting it into your cloud-native tools for analysis, training machine learning models, and serving up insights.

You will build scalable, durable, Data Lakes with Google Cloud storage solutions



In this Data Lake module, we will focus on two of the foundational Storage products which will make up your Data Lake: Google Cloud Storage and Cloud SQL for your relational data. Later in the course you will practice with Cloud Bigtable as well when you do high-throughput streaming pipelines.

You may have been surprised as I was when I first started learning Google Cloud to not see BigQuery in the storage column. Generally BigQuery is used as a Data Warehouse -- what's the core difference between a Data Lake and Data Warehouse then?

Data lake versus data warehouse

A data lake is a capture of every aspect of your business operation.

The data is stored in its natural/raw format, usually as object blobs or files.

- Retain all data in its native format
- Support all data types and all users
- Adapt to changes easily
- Tends to be application-specific

A data lake is essentially the place where you captured every aspect of your business operations. Because you want to capture every aspect you tend to store the data in its natural raw format the format in which it is thrown out by your application so you may have a log file and the log files stored as is... in a data lake. You can basically store anything that you want and because you want to store it all.....you tend to store these things as object blobs or files.

The advantage of the data lake's flexibility as a central collection point is also the problem. With a data lake, the data format is very much driven by the application that writes the data and it is in whatever format it is. The advantage of a data lake is that whenever the application gets upgraded it can start writing the new data immediately because it's just a capture of whatever raw data exists. So how do you take this flexible and large amount of raw data and do something useful with it?

Data lake versus data warehouse

In contrast, a data warehouse typically has the following characteristics:

- Typically loaded only after a use case is defined.
- Processed/organized/transformed.
- Provide faster insights.
- Current/historical data for reporting.
- Tends to have consistent schema shared across applications.

Enter the Data Warehouse.

On the other hand a data warehouse is much more thoughtful. You might load the data into a data warehouse only after you have a schema defined and the use case identified. you might take the raw data that exists in a data lake, and transform it, organize it, process it, clean it up and then store into a data warehouse. why are you getting the data warehouse? Maybe because the data in the data warehouse is used to generate charts reports dashboards Etc. The idea is that because the schema is consistent and shared across all of the applications someone can go ahead and analyze the data and derive Insight from it much faster. So a data warehouse tends to be structured and semi-structured data that is organized and placed in a format that makes it conducive for querying and analysis.

Agenda

Introduction to Data Lakes

Data Storage and ETL Options on
Google Cloud

Building a Data Lake Using
Cloud Storage

Securing Cloud Storage

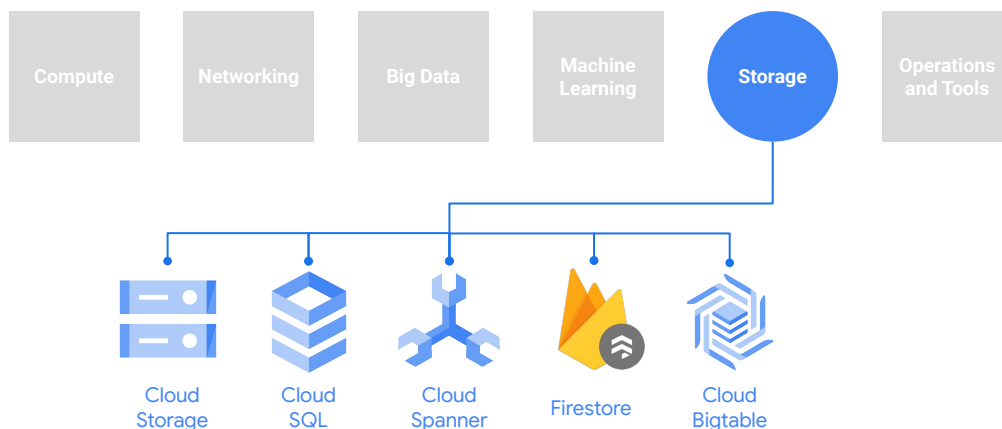
Storing All Sorts of Data Types

Cloud SQL as a Relational
Data Lake

Lab: Loading Taxi Data Into
Cloud SQL

Next let's discuss your data storage and Extract Transform and Load options on Google Cloud.

Storage options for your data on Google Cloud



Your options on Google Cloud for building a Data Lake are your storage solutions you saw earlier. You've got Cloud Storage as a great catch-all, Cloud SQL and Cloud Spanner for Relational Data, and Firestore and Cloud Bigtable for NoSQL data. Choosing which option or options to use depends heavily on your use case and what you're trying to build. In this chapter we will focus on cloud storage and cloud sql but you will see the NoSQL options like Cloud Bigtable later on in the course when we talk about very high-throughput streaming.

So how do you decide on which path to take for your lake?

The path your data takes to get to the cloud depends on

- Where your data is now.

The final destination for where your data lands on the cloud and the paths that you take to get your data to the cloud depends on where your data is now.

The path your data takes to get to the cloud depends on

- Where your data is now.
- How big your data is.

How big your data is (this is the Volume component of the 3 Vs of Big Data).

The path your data takes to get to the cloud depends on

- Where your data is now.
- How big your data is.
- Where it has to go.

And ultimately **where does it have to go**. In architecture diagrams the ending point for the data is called a data sink. A common data sink after a data lake is your data warehouse.

The path your data takes to get to the cloud depends on

- Where your data is now.
- How big your data is.
- Where it has to go.
- How much transformation is needed.

Don't forget a critical thing to consider is how much processing and transformation your data needs before it is useful to your business.

Now you may ask, do I do the processing before I load it into my Data Lake or afterward before it gets shipped off somewhere else? Let's talk about these patterns.

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



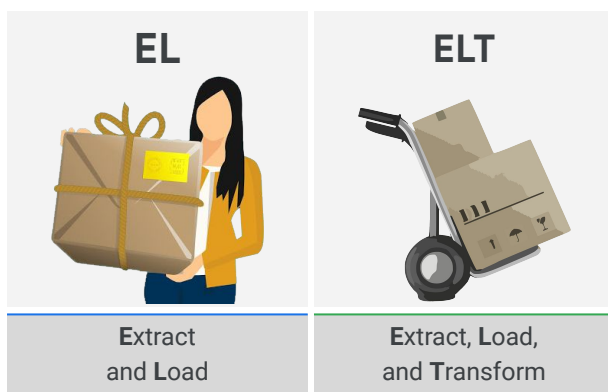
The method that you use to load the data into the cloud depends on how much transformation is needed from the raw data that you have to the final format if you want it in. In this chapter, we will look at some of the considerations for the final format that you want it in.

The simplest case might be that you have a data in a format that is readily ingested by the cloud product that you want to store it in.

Let's say for example you have your data in Avro format and you want to store the data in BigQuery because your use case fits what BigQuery is good at. Then what we do is simply EL. Extract and Load. BigQuery will directly load Avro files. **EL is Extract and Load (EL)**. This refers to when data can be imported "as is" into a system. Examples include importing data from a database, where the source and the target have the same schema.

One of the features that makes BigQuery unique is that -- as you saw before with the federated query example -- you may end up not even loading the data into BigQuery and still can query off of it. Avro, ORC, and Parquet files are all now supported for federated querying.

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

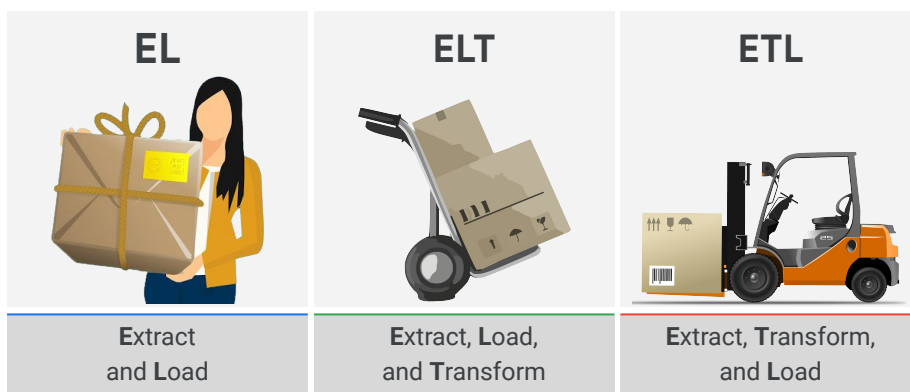


So what's an ELT or Extract, Load, and TRANSFORM the data?

That's when the data as loaded into the cloud product is not the final format you want it in. You may want to clean it up. Or maybe you want to transform the data in some way, such as perhaps you want to correct it. In other words you would extract from your on-premise system, load into the cloud product, and then do the transformation.

That's an extract load and transform, or ELT (ELT). You tend to do this when the amount of transformation that's needed is not very high and the transformation will not greatly reduce the amount of data that you have. ELT allows raw data to be loaded directly into the target and transformed there. For example, in BigQuery, you could use SQL to transform the data and write a new table.

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



The third option is extract transform and load or ETL (ETL). This is the case when you want to extract the data, apply a bunch of processing to it, and then load it into the cloud product. This is usually what you pick when this transformation is essential or if this transformation greatly reduces the data size, so by transforming the data before loading it into the cloud you might be able to greatly reduce the network bandwidth that you need. If you have your data in some binary proprietary format, and you need to convert it before loading, then you need ETL as well.

Extract, Transform, Load (ETL) is a data integration process in which transformation takes place in an intermediate service before it is loaded into the target. For example, the data might be transformed in a data pipeline like Dataflow before being loaded into BigQuery.

Agenda

Introduction to Data Lakes

Data Storage and ETL Options on Google Cloud

Building a Data Lake Using Cloud Storage

Securing Cloud Storage

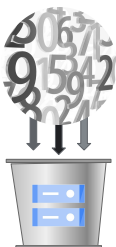
Storing All Sorts of Data Types

Cloud SQL as a Relational Data Lake

Lab: Loading Taxi Data Into Cloud SQL

Cloud Storage is the essential storage service for working with data, especially unstructured data in the cloud. Let's do a deep dive into why Google Cloud storage is a popular choice to serve as a data lake.

Cloud Storage



Qualities that Cloud Storage contributes to data engineering solutions:

- ✓ Persistence
- ✓ Durability
- ✓ Strong consistency
- ✓ Availability
- ✓ High throughput



Cloud
Storage

Google Cloud

Data in Cloud Storage persists beyond the lifetime of VMs or clusters (i.e., it is persistent). It is also relatively inexpensive compared to the cost of compute.

So, for example, you might find it more advantageous to cache the results of previous computations in cloud storage.

Or if you don't need an application running all the time, you might find it helpful to save the state of your application into cloud storage, and shut down the machine it is running on when you don't need it.

Cloud Storage is an object store, so it just stores and retrieves binary objects without regard to what data is contained in the objects. However, to some extent, it also provides file system compatibility and can make objects look like and work as if they were files, so you can copy files in and out of it.

- Data stored in Cloud Storage will basically stay there forever (i.e, it is durable), but is available instantly -- it is strongly consistent.
- You can share data globally, but it is encrypted and completely controlled and private if you want it to be.
- It is a global service, and you can reach the data from anywhere (i.e, it offers global availability). But the data can also be kept in a single geographic location if you need that.
- Data is served up with moderate latency and high throughput.

As a Data Engineer, you need to understand how Cloud Storage accomplishes these apparently contradictory qualities, and when and how to employ them in your solutions.

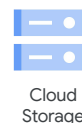
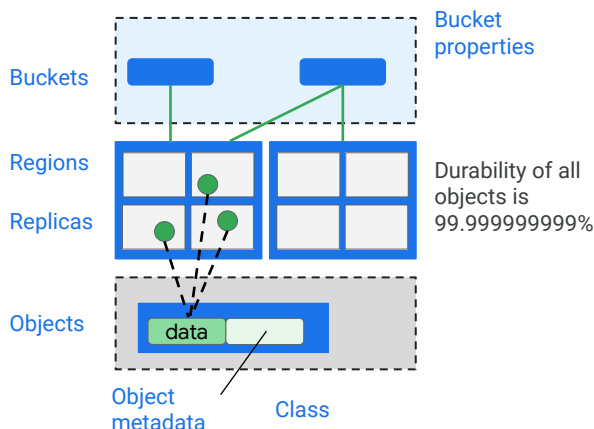
How does Cloud Storage work?

Single global namespace simplifies locating buckets and objects

Location to control latency

Durability and availability

Long object names simulate structure



Google Cloud

A lot of Cloud Storage's amazing properties have to do with the fact it is an object store. And other features are built on top of that base.

The two main entities in Cloud Storage are buckets and objects. Buckets are containers for objects. And objects exist inside of buckets and not apart from them. So buckets are containers for data.

Buckets are identified in a single global unique namespace. So that means, once a name is given to a bucket, it cannot be used by anyone else unless and until that bucket is deleted and the name is released. Having a global namespace for buckets simplifies locating any particular bucket. When a bucket is created it is associated with a particular region or with multiple regions. Choosing a region close to where the data will be processed will reduce latency, and if you are processing the data using cloud services within the region, it will save you on network egress charges.

When an object is stored, Cloud Storage **replicates the object**. It monitors the replicas, and if one of them is lost or corrupted, it replaces it with a fresh copy. This is how Cloud Storage gets many 9s of durability. For a multi-region bucket, the objects are replicated across regions, and for a single region bucket, the objects are replicated across zones.

In any case, when the object is retrieved, it is served up from the closest replica to the requester. And that is how low latency occurs. Multiple requesters could be retrieving the objects at the same time from different replicas. And that is how high throughput is

achieved.

Finally, the objects are stored with metadata. Metadata is information about the object. Additional Cloud Storage features use the metadata for purposes such as access control, compression, encryption, and lifecycle management. For example, Cloud Storage knows when an object was stored, and it can be set to automatically delete after a period of time. This feature uses the object metadata to determine when to delete the object.

Overview of storage classes

	Standard	Nearline	Coldline	Archive
Use case	“Hot” data and/or stored for only brief periods of time like data-intensive computations	Infrequently accessed data like data backup, long-tail multimedia content, and data archiving	Infrequently accessed data that you read or modify at most once a quarter	Data archiving, online backup, and disaster recovery
Minimum storage duration	None	30 days	90 days	365 days
Retrieval cost	None	\$0.01 per GB	\$0.02 per GB	\$0.05 per GB
Availability SLA	99.95% (multi/dual) 99.90% (region)	99.90% (multi/dual) 99.00% (region)		None
Durability	99.999999999%			

Cloud Storage has four storage classes: Standard, Nearline, Coldline and Archive and each of those storage classes provide 3 location types:

- A multi-region is a large geographic area, such as the United States, that contains two or more geographic places.
- A dual-region is a specific pair of regions, such as Finland and the Netherlands.
- A region is a specific geographic place, such as London.

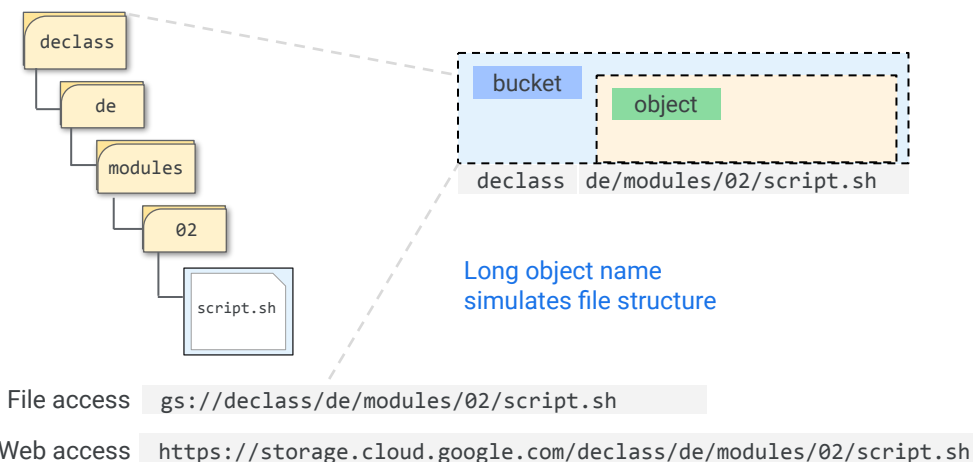
Objects stored in a multi-region or dual-region are geo-redundant. Now, let's go over each storage class:

- **Standard** Storage is best for data that is frequently accessed ("hot" data) and/or stored for only brief periods of time. This is the most expensive storage class but it has no minimum storage duration and no retrieval cost. When used in a:
 - region, Standard Storage is appropriate for storing data in the same location as Google Kubernetes Engine clusters or Compute Engine instances that use the data. Co-locating your resources maximizes the performance for data-intensive computations and can reduce network charges.
 - dual-region, you still get optimized performance when accessing Google Cloud products that are located in one of the associated regions, but you also get the improved availability that comes from storing data in geographically separate locations.
 - multi-region, Standard Storage is appropriate for storing data that is

- accessed around the world, such as serving website content, streaming videos, executing interactive workloads, or serving data supporting mobile and gaming applications.
- **Nearline** Storage is a low-cost, highly durable storage service for storing infrequently accessed data like data backup, long-tail multimedia content, and data archiving. Nearline Storage is a better choice than Standard Storage in scenarios where slightly lower availability, a 30-day minimum storage duration, and costs for data access are acceptable trade-offs for lowered at-rest storage costs.
- **Coldline** Storage is a very-low-cost, highly durable storage service for storing infrequently accessed data. Coldline Storage is a better choice than Standard Storage or Nearline Storage in scenarios where slightly lower availability, a 90-day minimum storage duration, and higher costs for data access are acceptable trade-offs for lowered at-rest storage costs.
- **Archive** Storage is the lowest-cost, highly durable storage service for data archiving, online backup, and disaster recovery. Unlike the "coldest" storage services offered by other Cloud providers, your data is available within milliseconds, not hours or days. Unlike other Cloud Storage storage classes, Archive Storage has no availability SLA, though the typical availability is comparable to Nearline Storage and Coldline Storage. Archive Storage also has higher costs for data access and operations, as well as a 365-day minimum storage duration. Archive Storage is the best choice for data that you plan to access less than once a year.

Let's focus on durability and availability. All of these storage classes have 11 nines of durability, but what does that mean? Does that mean you have access to your files at all times? No, what that means is you won't lose data. You may not be able to access the data which is like going to your bank and saying well my money is in there, it's 11 nines durable. But when the bank is closed we don't have access to it which is the availability that differs between the storage classes and the location type.

Cloud Storage simulates a file system



Google Cloud

Cloud Storage uses the bucket name and object name to simulate a file system. This is how it works. The bucket name is the first term in the URI. A forward slash is appended to it, and then it is concatenated with the object name. The object name allows the forward slash character as a valid character in the name. The very long object name with forward slash characters in it looks like a file system path, even though it is just a single name.

In the example shown, the bucket name is "**declass**". The object name is "**de/modules/02/script.sh**". The forward slashes are just characters in the name. If this path were in a file system, it would appear as shown on the left, with a set of nested directories, beginning with "declass".

Now for all practical purposes it works like a file system. But there are some differences. For example, imagine that you wanted to move all the files in the 02 directory to the 03 directory inside the modules directory. In a file system you would have actual directory structures and you would simply modify the filesystem metadata, so that the entire move is atomic. But in an object store simulating a file system, you would have to search through all the objects contained in the bucket for names that had "02" in the right position in the name. Then you would have to edit each object name and rename them using 03. This would produce apparently the same result; moving the files between directories. However, instead of working with a dozen files in a directory, the system had to search over possibly thousands of objects in the bucket to locate the ones with the right names and change each of them. So the performance characteristics are different. It might take longer to move a dozen

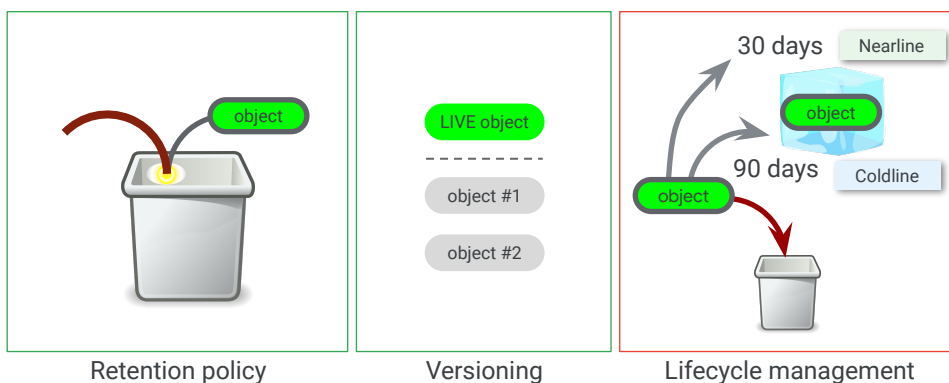
objects from directory 02 to directory 03 depending on how many other objects are stored in the bucket. During the move, there will be list inconsistency, with some files in the old directory and some in the new directory.

A best practice is to avoid the use of sensitive information as part of bucket names because bucket names are in a global namespace. Bucket names, not the data in the buckets. The data in the buckets can be kept private if you need it to be.

Cloud Storage can be accessed using a file access method. That allows you, for example, to use a copy command from a local file directly to Cloud Storage. Use the tool `gsutil` to do this.

Cloud Storage can also be accessed over the web. The site (<https://storage.cloud.google.com>) uses TLS (HTTPS) to transport your data which protects credentials as well as data in transit.

Cloud Storage has many object management features



Cloud Storage has many object management features. For example, you can set a retention policy on all objects in the bucket. For example, the objects should be expired after 30 days.

You can also use versioning, so that multiple versions of an object are tracked and available if necessary. You might even set up lifecycle management, to automatically move objects that haven't been accessed in 30 days to Nearline and after 90 days to Coldline.

Let's take a look at how you can manage these object lifecycles a bit more programmatically to help optimize your use of cloud storage and save on storage costs.

Agenda

Introduction to Data Lakes

Data Storage and ETL Options on Google Cloud

Building a Data Lake Using Cloud Storage

[Securing Cloud Storage](#)

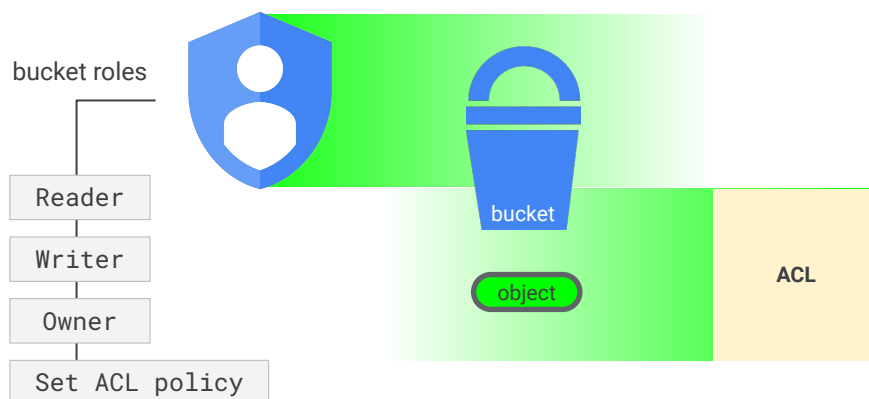
Storing All Sorts of Data Types

Cloud SQL as a Relational Data Lake

Lab: Loading Taxi Data Into Cloud SQL

Securing your Data Lake running on Cloud Storage is of paramount importance. We will discuss the key security features you need to know as a data engineer to control access to your objects.

Controlling access with Cloud IAM and access lists



Google Cloud

Cloud Storage implements two completely separate but overlapping methods of controlling access to objects; Cloud IAM policy and Access Control Lists. Cloud IAM is standard across the Google Cloud. It is set at the bucket level and applies uniform access rules to all objects within a bucket. Access Control Lists can be applied at the bucket level or on individual objects. So it provides more fine-grained access control

The Cloud IAM controls are as you would expect. Cloud IAM provides Project roles and Bucket roles. Including, Bucket Reader, Bucket Writer, and Bucket Owner. The ability to create or change Access Control Lists is an IAM Bucket role. And the ability to create and delete buckets, and to set IAM policy, is a Project level role. Custom roles are available. Project level Viewer, Editor, and Owner roles make users members of special internal groups that give them access by being members of Bucket roles. See the online documentation for details.

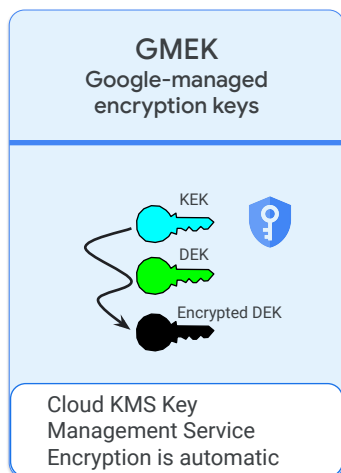
When you create a bucket, you are offered the option of disabling access lists and only using Cloud IAM. Access Lists are currently enabled by default. This choice used to be immutable. But now you can disable access lists even if they were in force previously.

As an example, you might give some bob@example.com Reader access to a bucket through IAM, and also give them Write access to a specific file in that bucket through Access Control Lists.

You can give such permissions to service accounts associated with individual

applications as well.

Data encryption options for many requirements



Google Cloud

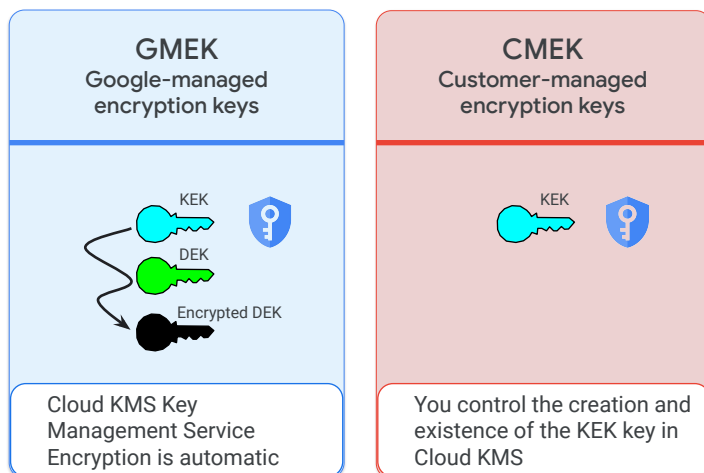
All data in Google Cloud is encrypted at rest and in transit. There is no way to turn the encryption off.

The encryption is done by Google using encryption keys that we manage -- **Google Managed Encryption Keys or GMEK**.

We use two-levels of encryption: First the data is encrypted using a data encryption key. Then, the data encryption key itself is encrypted using a key encryption key or KEK.

The KEKs are automatically rotated on a schedule and the current KEK stored in Cloud KMS, Key Management Service. You don't have to do anything. This is automatic behavior.

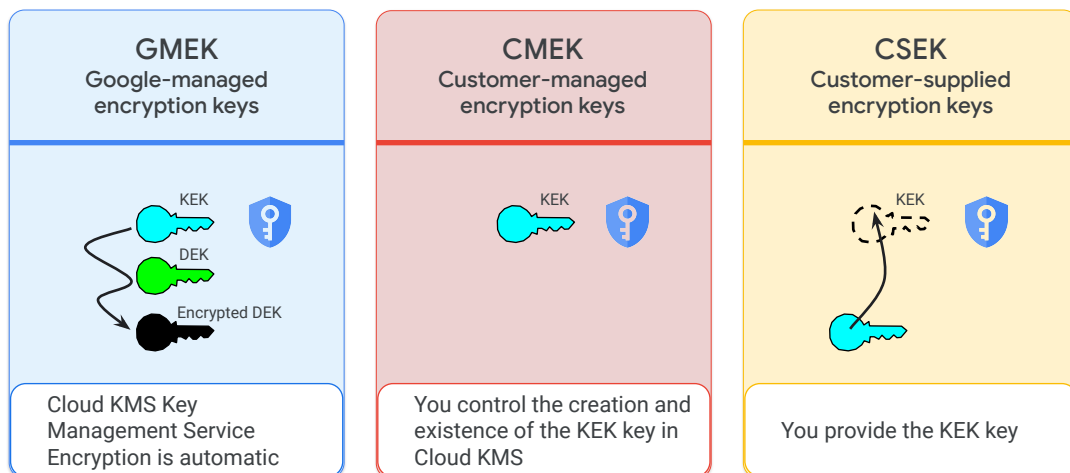
Data encryption options for many requirements



If you want to manage the KEK yourself, you can. Instead of Google managing the encryption key, you can control the creation and existence of the KEK that is used.

This is called **Customer Managed Encryption Keys or CMEK**.

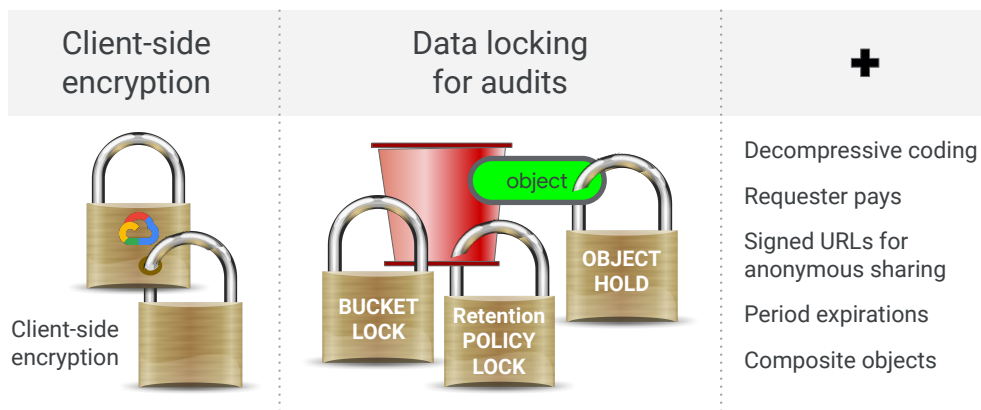
Data encryption options for many requirements



You can avoid Cloud KMS completely and supply your own encryption and rotation mechanism. This is called **CSEK or customer-managed encryption keys**.

Which data encryption option you use depends on business, legal, and regulatory requirements. Please talk to your company's legal counsel.

Cloud Storage supports many special use cases



Google Cloud

The fourth encryption option is Client-Side Encryption. Client-Side Encryption simply means that you have encrypted the data before it is uploaded and have to decrypt the data yourself before it is used. Cloud Storage still performs GMEK, CMEK, or CSEK encryption on the object. It has no knowledge of the extra layer of encryption you may have added.

Cloud Storage supports logging of data access, and these logs are immutable. In addition to Cloud Audit Logs and Cloud Storage Access Logs, there are various holds and locks that you can place on the data itself. For audit purposes, you can place a hold on an object, and all operations that could change or delete the object are suspended until the hold is released. You can also lock a bucket, and no changes or deletions can occur until the lock is released. Finally, there is the locked retention policy previously discussed. And it continues to remain in effect and prevent deletion whether a Bucket Lock or Object Hold are in force or not. Data locking is different from encryption. Where encryption prevents someone from understanding the data, locking prevents them from modifying the data.

There are a whole host of special use cases supported by Cloud Storage. For example, decompressive coding ...

- By default, the data you upload is the same data you get back from Cloud Storage. This includes gzip archives, which, usually are returned as gzip archives. However, if you tag an object properly in metadata, you can cause Cloud Storage to decompress the file as it is being served. Benefits of the

- smaller compressed file are faster upload and lower storage costs compared with the uncompressed files.
- You can set up a bucket to be requester pays. Normally, if data is accessed from a different region, you will have to pay network egress charges. But you can make the requester pay, so that you pay only for data storage.
- You can create a signed URL to anonymously share an object in Cloud Storage, and even have the URL expire after a period of time.
- It is possible upload an object in pieces, and create a composite object without having to concatenate the pieces after upload.

There are a lot of useful features in Cloud Storage, but we have to move on!

Agenda

Introduction to Data Lakes

Data Storage and ETL Options on
Google Cloud

Building a Data Lake Using
Cloud Storage

Securing Cloud Storage

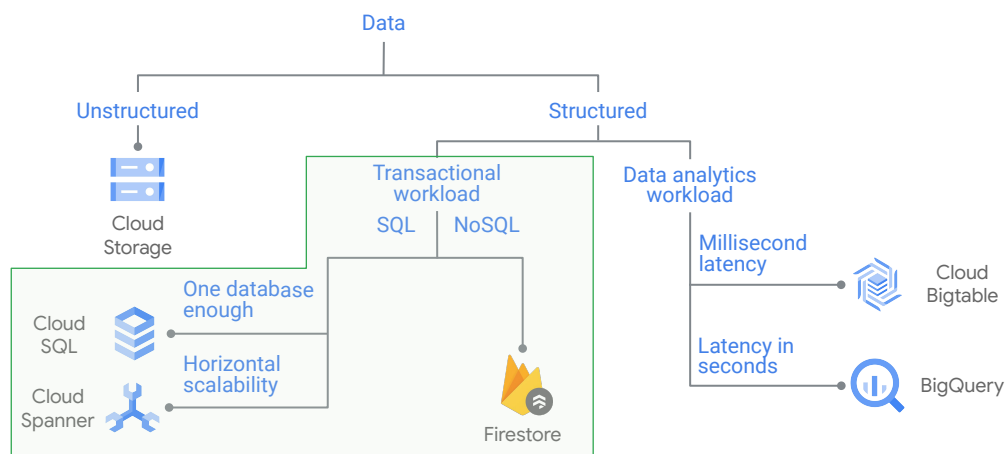
Storing All Sorts of Data Types

Cloud SQL as a Relational
Data Lake

Lab: Loading Taxi Data Into
Cloud SQL

As I highlighted before, Cloud Storage is not your only choice when it comes to storing data on Google Cloud.

Different considerations for transactional workloads



You do not want to use Cloud Storage for transactional workloads. Even though the latency of Cloud Storage is low, it is not low enough to support high-frequency writes. For transactional workloads, use Cloud SQL or Firestore depending on whether you want to use SQL or No-SQL.

You also don't want to use Cloud Storage for analytics on structured data. If you do that, you will spend a significant amount of compute parsing data -- it is better to use Cloud Bigtable or BigQuery for analytics workloads on structured data, depending on the latency required.

Transactional versus analytical

	Transactional	Analytical
Source of data	Operational data; OLTPs are the original source of the data	Consolidation data; OLAP data comes from the various OLTP databases
Purpose of data	Control and run fundamental business tasks	Help with planning, problem solving, and decision support
What the data shows	Reveals snapshot of ongoing business processes	Multi-dimensional views of various kinds of business activities
Inserts and updates	Short and fast inserts and updates initiated by end users	Periodic long-running batch jobs refresh the data
Queries	Relatively standardized and simple queries returning relatively few records	Often complex queries involving aggregations
Processing speed	Typically very fast	Depends on amount of data involved; improve query speed with indexes
Space requirements	Can be relatively small if historical data is archived	Larger, more indexes than OLTP

We keep talking about transactional vs. analytics workloads. What exactly do we mean?

Transactional workloads are ones where you require fast inserts and updates. You want to maintain a snapshot, a current state of the system. The tradeoff is that queries tend to be relatively simple and tend to affect only a few records.

For example, in a banking system, depositing your salary to your account is a transaction. It updates the balance field. The bank is doing online transaction processing or OLTP.

Transactional versus analytical

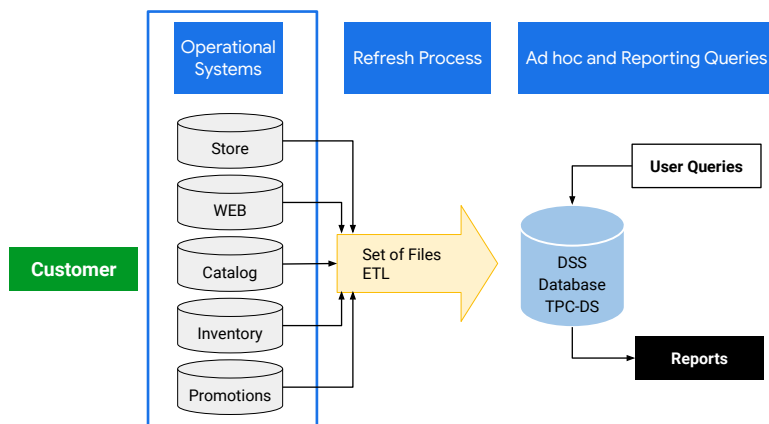
	Transactional	Analytical
Source of data	Operational data; OLTPs are the original source of the data	Consolidation data; OLAP data comes from the various OLTP databases
Purpose of data	Control and run fundamental business tasks	Help with planning, problem solving, and decision support
What the data shows	Reveals snapshot of ongoing business processes	Multi-dimensional views of various kinds of business activities
Inserts and updates	Short and fast inserts and updates initiated by end users	Periodic long-running batch jobs refresh the data
Queries	Relatively standardized and simple queries returning relatively few records	Often complex queries involving aggregations
Processing speed	Typically very fast	Depends on amount of data involved; improve query speed with indexes
Space requirements	Can be relatively small if historical data is archived	Larger, more indexes than OLTP

An analytics workload, on the other hand, tends to read the entire dataset and is often used for planning or decision support. The data might come from a transaction processing system, but it is often consolidated from many OLTP systems.

For example, a bank regulator might require us to provide a report of every customer who transferred more than \$10,000 to an overseas account. They might ask the bank to include customers who try to transfer the \$10,000 in smaller chunks over the period of a week. A report like this will require scanning through a significantly large dataset and require a complex query that involves aggregating over moving time windows.

This is an example of an online analytical processing or OLAP workload.

Transactional systems are 80% writes and 20% reads

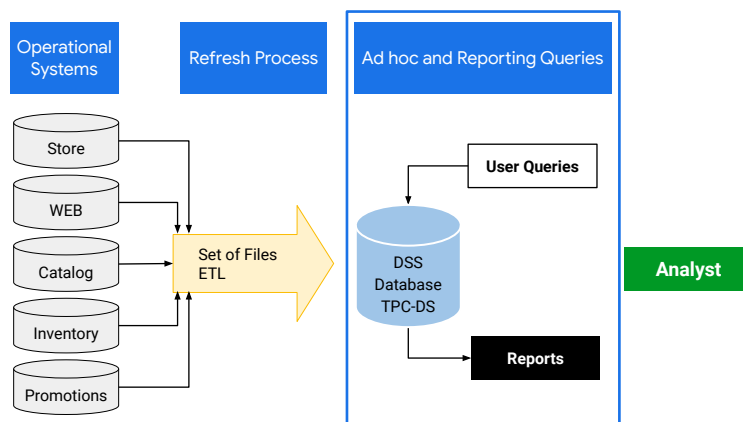


The reason we treat these use cases differently is that transactional systems are heavily write.

These tend to be operational systems. For example, a retailer's catalog data will require updating every time the retailer adds a new item or changes the price.

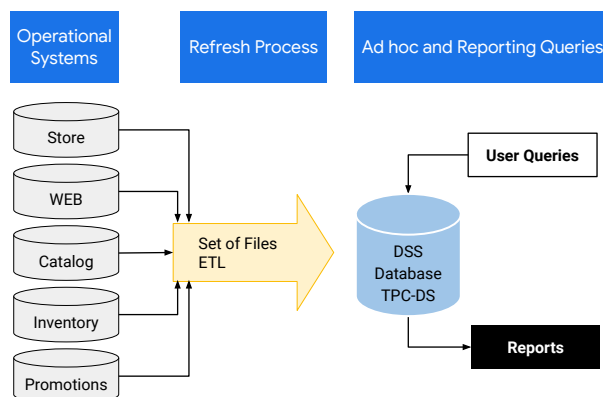
The inventory data will be need to be updated every time the retailer sells an item. This is because the catalog and inventory systems have to maintain an up-to-the-moment snapshot of the business.

Analytical systems are 20% writes and 80% reads



Analytical systems can be periodically populated from the operational systems. We could use this once a day to generate a report of items in our catalog whose sales are increasing, but whose inventory levels are low. Such a report will have to read a bunch of data, but not have to write much. OLAP systems are read-focused.

Data engineers build the pipelines between the systems

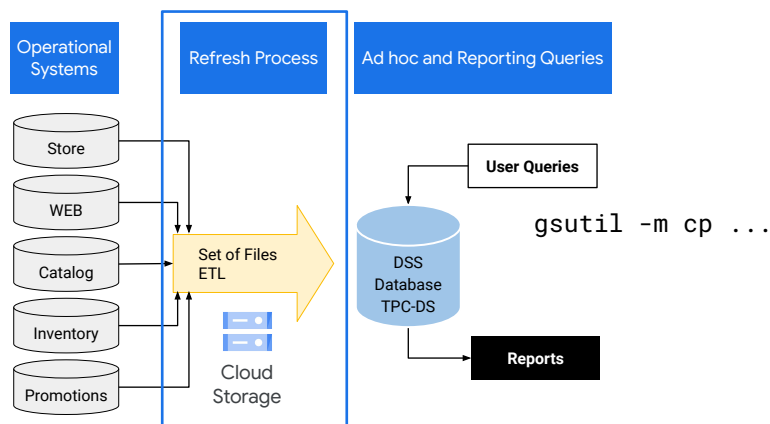


Recall what we said. Analytical systems can be periodically populated from the operational systems.

Data engineers build the pipelines to populate the OLAP system from the OLTP system.

One simple way might be to export the database as a file and load it into the data warehouse. This is what we called EL.

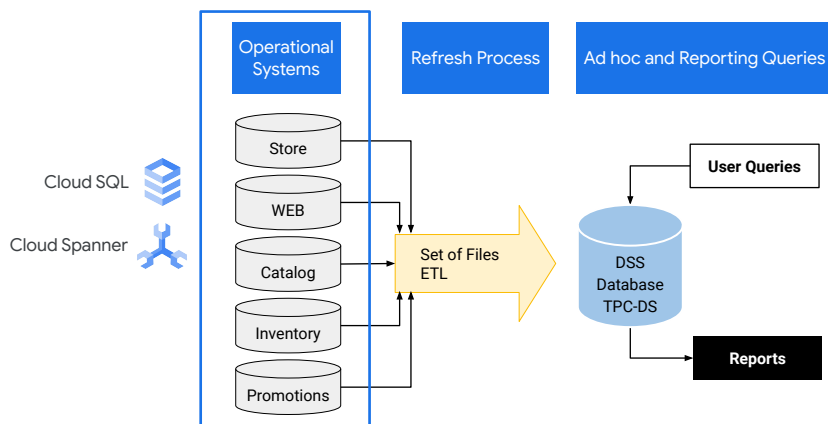
Use Cloud Storage for scalable staging of raw data



On Google Cloud, the data warehouse tends to be BigQuery. There is a limit to the size of the data that you can directly load to BigQuery. This is because your network might be a bottleneck.

Rather than load the data directly into BigQuery, it can be much more convenient to first load it to Cloud Storage and load from Cloud Storage to BigQuery. This is because Cloud Storage supports multi-threaded, resumable loads. Just provide the `-m` option to `gsutil`. Loading from Cloud Storage will also be faster because of the high throughput it offers.

Choose from cloud relational databases for transactional workloads



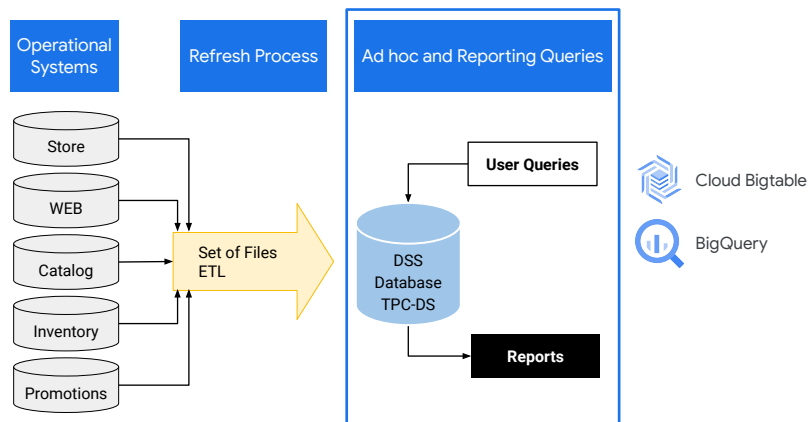
Getting back to the discussion on transactional workloads, you've got a couple options for relational databases. The default choice here is Cloud SQL, but if you require a globally distributed database, then use Cloud Spanner.

You'd want a globally distributed database if your database will see updated from applications running in different geographic regions. The True Time capability of Spanner is very appealing for this kind of use case.

Another reason you might choose Spanner is if your database is too big to fit into a single Cloud SQL instance. If our database is many gigabytes, you need a distributed database. The scalability of Spanner is very appealing for this use case.

Other than that, you'd use Cloud SQL because it is more cost-effective.

Choose from cloud data warehouses for analytic workloads



For analytics workloads, the default choice is BigQuery.

However, if you require high-throughput inserts, more than millions of rows per second, or if you require low latency, on the order of milliseconds, use Cloud Bigtable.

Other than that, you'd use BigQuery because it is more cost-effective.

Agenda

Introduction to Data Lakes

Data Storage and ETL Options on Google Cloud

Building a Data Lake Using Cloud Storage

Securing Cloud Storage

Storing All Sorts of Data Types

Cloud SQL as a Relational Data Lake

Lab: Loading Taxi Data Into Cloud SQL

Cloud SQL, we said, is the default choice for OLTP (or online transaction processing) workloads on Google Cloud. Let's take a quick look.

Cloud SQL is a fully managed database service that makes it easy to set up and administer your relational databases in the cloud



Google Cloud

Cloud SQL is an easy-to-use service that delivers fully managed relational databases. Cloud SQL lets you hand off to Google the mundane, but necessary and often time-consuming tasks—like applying patches and updates, managing backups, and configuring replications—so you can put your focus on building great applications.

Cloud SQL is our managed service for third party RDBMSs. It supports MySQL, PostgreSQL, and Microsoft SQL Server and additional RDBMSs will be added over time.

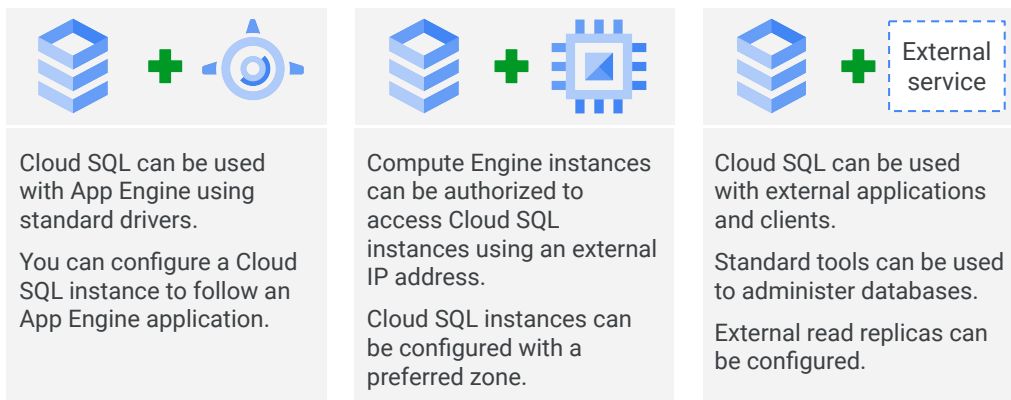
What this means is that we provide a compute engine instance that has MySQL already installed. We'll manage the instance on your behalf.

We'll do backups, we'll do security updates, and update the minor versions of the software so that you don't have to worry about that.

In other words, Google Cloud manages the MySQL database to be point where you can treat it as a service.

We even do DBA-like things. You can tell us to add a failover replica for your database. We'll manage it for you, and you'll have a 99.95% availability SLA.

Cloud SQL can be used with other Google Cloud services



Another benefit of Cloud SQL instances is that they are accessible by other Google Cloud services and even external services. You can use Cloud SQL with App Engine using standard drivers like Connector/J for Java or MySQLdb for Python.

You can authorize Compute Engine instances to access Cloud SQL instances and configure the Cloud SQL instance to be in the same zone as your virtual machine.

Cloud SQL also supports other applications and tools that you might be used to, like SQL Workbench, Toad and other external applications using standard MySQL drivers.

Backup, recovery, scaling, and security is managed for you

- Google security
- Managed backups
- Vertical scaling (read and write)
- Horizontal scaling (read)
- Automatic replication



Google Cloud

One of the advantages of Google managing your database is that you get the benefits of **Google security**. Cloud SQL customer data is encrypted when on Google's internal networks and when stored in database tables, temporary files, and backups. Every Cloud SQL instance includes a network firewall, allowing you to control network access to your database instance by granting access.

Cloud SQL is easy to use: it doesn't require any software installation or maintenance. And **Google manages the backups**. Cloud SQL takes care of securely storing your backed-up data and makes it easy for you to restore from a backup and perform a point-in-time recovery to a specific state of an instance. Cloud SQL retains up to 7 backups for each instance, which is included in the cost of your instance.

You can **vertically scale Cloud SQL** -- just increase your machine size. Scale up to 64 processor cores and more than 100 GB of RAM.

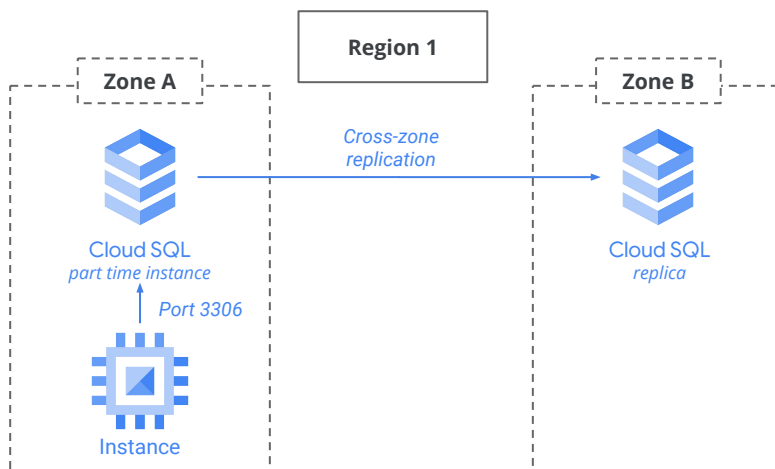
Horizontally, you can quickly scale out with read replicas. Google Cloud SQL supports three read replica scenarios:

- Cloud SQL instances replicating from a Cloud SQL primary instance
Replicas are other instances in the same project and location as the primary instance.
- Cloud SQL instances replicating from an external primary instance
The primary instance is external to Google Cloud SQL. For example, it can be outside the Google network or in a Google Compute Engine instance. You can use this for backing up an on-prem MySQL database.

- External MySQL instances replicating from a Cloud SQL primary instance
External replicas are in hosting environments, outside of Cloud SQL.

If you need horizontal read-write scaling, consider Cloud Spanner.

Cloud SQL replication



Google Cloud

For the special case of failover, Cloud SQL supports this.

Cloud SQL instances can be configured with a failover replica in a different zone in the same region. Then, Cloud SQL data is replicated across zones within a region for durability. In the unlikely event of a data center outage, a Cloud SQL instance will automatically become available in another zone. All changes made to data on the primary are replicated to failover.

If the primary instance's zone has an outage, Cloud SQL automatically fails over to the replica. If the primary has issues not caused by a zone outage, failover doesn't occur. You can, however, initiate failover manually

A few caveats:

1. Note that the failover replica is charged as a separate instance
2. When a zonal outage occurs and your primary fails over to your failover replica, any existing connections to the instance are closed. However, your application can reconnect using the same connection string or IP address; you do not need to update your application after a failover.
3. After the failover, the replica becomes the primary, and Cloud SQL automatically creates a new failover replica in another zone. If you located your Cloud SQL instance to be near other resources, such as a Compute Engine instance, you can relocate your Cloud SQL instance back to its original zone when the zone becomes available. Otherwise, there is no need to relocate your instance after a failover.

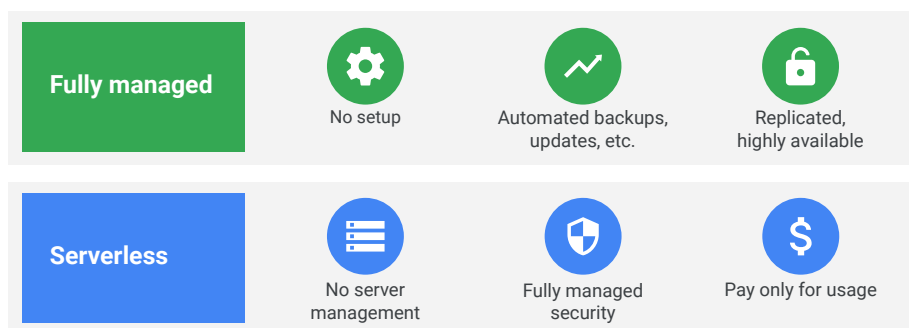
You can use the failover replica as a read replica, to offload read operations from the

primary.

For more details on failover replicas, see

<https://cloud.google.com/sql/docs/mysql/high-availability>

Fully managed versus serverless

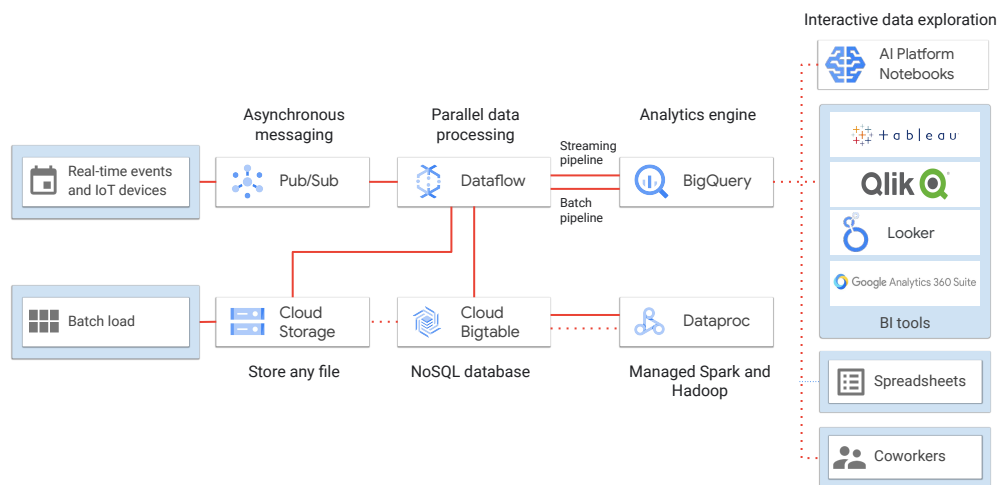


We kept saying that Cloud SQL is fully managed. We have also used the word serverless to describe BigQuery, for example. What's the difference?

By **fully managed**, we mean that the service runs on hardware that you can control. You can ssh into a Cloud SQL instance, for example. That said, Google helps you manage the instance. For example, by automating backups and setting up failover instances, etc.

Serverless is the next step up. You can treat a serverless product as just an API that you are calling. You pay for using the product, but don't have to manage any servers.

Modern serverless data management architecture



Google Cloud

BigQuery is serverless. As is Pub/Sub for asynchronous messaging and Dataflow for parallel data processing. You can think of Cloud Storage as being serverless as well. Sure, Cloud Storage uses disks, but you never actually interact with the hardware. One of the unique things about Google Cloud is that you can build a data processing pipeline of well-designed components all of which are fully serverless.

Dataproc, on the other hand, is fully managed. It helps you run Spark and Hadoop workloads without having to worry about setup.

Given the choice between doing a brand-new project on BigQuery or Dataflow which are serverless and Dataproc which is fully managed, which one should you choose?

All other things being equal, choose the serverless product.

Agenda

Introduction to Data Lakes

Data Storage and ETL Options on Google Cloud

Building a Data Lake Using Cloud Storage

Securing Cloud Storage

Storing All Sorts of Data Types

Cloud SQL as a Relational Data Lake

[Lab: Loading Taxi Data Into Cloud SQL](#)

Lab Intro

Loading Taxi Data into Cloud SQL



In this lab, you will practice creating a data lake for your relational data with Cloud SQL. You will:

- Create a Cloud SQL instance which can hold multiple databases.
- Create a new Cloud SQL database.
- Import text data into Cloud SQL.
- And finally check the data for integrity.