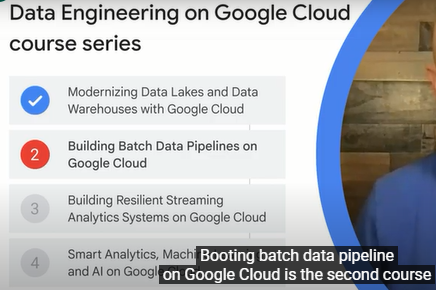
**Building Batch Data Pipelines on Google Cloud.**

**Course Introduction.**

Welcome to Building Batch Data Pipelines on Google Cloud.

I'm Damon, and I'm a technical curriculum creator at Google.

Booting batch data pipeline on Google Cloud is the second course of the Data Engineering on Google Cloud core series.

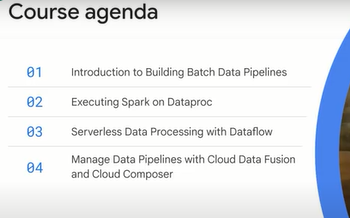


In the previous course, we talked about data lakes and data warehouses as storage options for your data.

In this course, we'll concentrate on how to build batch data pipelines to get data into storage using Extract Load, Extract Transform Load, and Extract Load Transform routines.

We'll cover several technologies for data transformation on Google Cloud.

Including BigQuery, Executing Spark on Dataproc, serverless data processing with Dataflow, and leveraging Google Cloud in pipelines with Cloud Data Fusion and Cloud Composer.



**Module Introduction.**

What are batch pipelines?

These are pipelines that process a bounded amount of data and then exit.

For example, you might have a batch pipeline that runs once a day.

It takes all the credit, debit, and money transfer transactions over that day, balances the books, and writes out the reconciled data to the data warehouse.

If you are going to write such a pipeline to balance the books, should you use EL, ELT, or ETL?

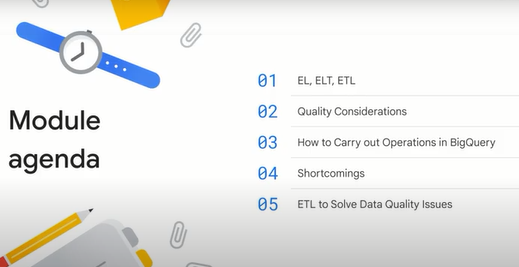
EL, remember, is extract and load.

ELT loads the data as is and then transforms on the fly.

ETL extracts the data, transforms it, then loads it into a data warehouse.

Deciding which to use depends on the kinds of transformations you need and data quality considerations.

We will look at how to build EL and ELT pipelines in BigQuery, the circumstances when EL and ELT are not appropriate, and why you might want to use ETL.



**EL, ELT, ETL.**

Let's start with a quick recap of EL, ELT and ETL.

EL is Extract and Load.

This refers to when data can be imported as is into a system.

ELT or Extract, Load and Transform, allows raw data to be loaded directly into the target and transformed whenever it is needed.

For example, you might provide access to the raw data through a view that determines whether the user wants all transactions or only reconciled ones.

ETL or Extract, Transform and Load, 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 Dataflow before being loaded into BigQuery.

When would you use EL?

The bottom line is that you should use EL only if the data is already clean and correct.

Perhaps you have log files in Cloud Storage.

You can extract data from files on Cloud Storage and load it into BigQuery's native storage.

This is a simple REST API call.

You can trigger this pipeline from Cloud Composer, Cloud Functions or via a scheduled query.

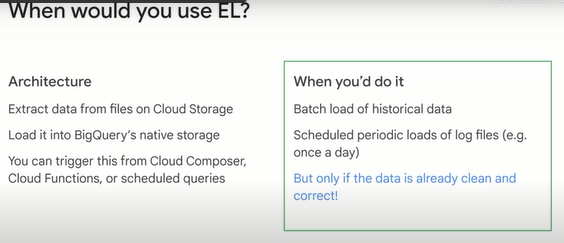
You might even set it to work in micro batches, not quite streaming but near real time.

Whenever a new file hits Cloud Storage, the Cloud Function runs, and the function invokes a BigQuery job.

The data transfer service in BigQuery will also work here.

Use EL for batch loading historical data or to do scheduled loads of log files.

But let me emphasize, use EL only if the data is already clean and correct.



ELT starts with EL, so the loading is the same and could work the same way.

File hits Cloud Storage, function invokes BigQuery load, table appended to.

The big difference is what happens next.

The table might be stored in a private dataset and everyone accesses the data through a view which imposes data integrity checks.

Or maybe you have a job that runs a SQL query with a destination table.

This way, transformed data is stored in a table that everyone accesses.

When do you use ELT?

One common case is when you don't know what kind of transformations are needed to make the data usable.

For example, let's say someone uploads a new image.

You invoke the Vision API and back comes a long JSON message about all kinds of things in the image, text in the image, whether there's a landmark, a logo, what objects.

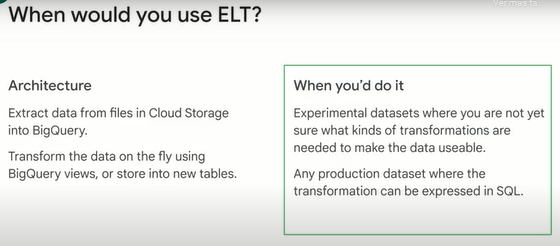
What will an analyst need in the future?

You don't know, so you store the raw JSON as is.

Later, if someone wants to count the number of times a specific company's logos are in this set of images, they can extract logos from the JSON and then count them.

Of course, this works only if the transformation that's needed can be expressed in SQL.

In the case of the Vision API, the result is JSON and BigQuery SQL has support for JSON parsing, so ELT will work in this case.



**Quality considerations.**

Now that we have looked at EL and ELT, let's look at some of the transformations you might want to do and how they can be done in BigQuery.

To keep things precise, let's assume that our data processing needs all revolve around quality improvements.

What are some of the quality related reasons why we might want to process data?

The top row are characteristics of information.

Information can be valid, accurate, complete, consistent and or uniform.

These terms are defined in the science of logic, each is independent.

For example, data can be complete without being consistent, it can be valid without being uniform.

There are formal definitions for each of these terms that you can look up online.

But the main practical reason for seeking them is shown in the second row, the problems they present in data analysis.

It is one thing to seek each of the five badges for your data to have objectively good data quality.

However, it is another thing when poor quality data interferes with data analysis and leads to incorrect business decisions.

So the reason to spend time, energy, and resources detecting and resolving quality issues is that it can affect a business outcome.

Thus, if data does not conform to your business rules, you have a problem of validity.

For example, let's say that you sell movie tickets, and each ticket costs $10.

If you have a $7 transaction, then you have a validity problem.

Similarly, accuracy problems are due to data not conforming to objective truth.

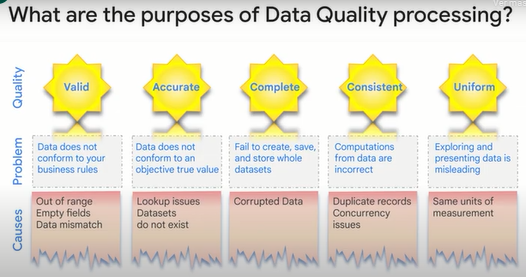
Completeness has to do with failing to process everything.

Consistency problems are if two different operations ought to be the same but yield different results, and because you don't know what to trust, you can't derive insights from the data.

Uniformity is when data values of the same column in different rows mean different things.

The main causes of these problems are listed in the third row.

We will explore methods of detecting each of these issues in data.



Now you have found the problems, what do you do about them?

ELT and BigQuery can often help fix many data quality issues.

Here is an example.

Imagine you plan to analyze data but there are duplicate records making it seem like one kind of event is more common, when in fact this is just a data quality issue.

You cannot derive insights from the data until the duplicates are removed.

So do you need a transformation step to remove the duplicates before you store the data?

Maybe, but a simpler solution exists, to count unique records.

You do, of course, have count distinct in BigQuery and you can use that instead.

Similarly, a problem like data being out of range can be solved in BigQuery without an intermediate transformation step.

Invalid data can be filtered out using a BigQuery view and everyone can access the view rather than the raw data.

**How to carry out operations in BigQuery.**

In this lesson we will look at various quality issues and talk through some BigQuery capabilities that can help you address those quality problems.

We can use views to filter out rows that have quality issues.

For example, remove quantities less than zero using a where clause.

After you do a group by, you can discard groups whose total number of records is less than 10 using the having clause.

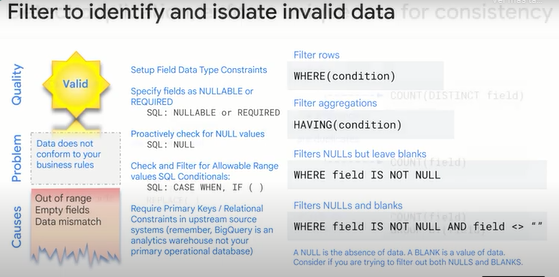
Think carefully about how you wish to treat nulls and blanks.

A null is the absence of data.

A blank is an empty string.

Consider if you were trying to filter out both nulls and blanks or only nulls or only blanks.

You can easily count non-null values using count if, and use the if statement to avoid using specific values in computations.



Consistency problems are often due to duplicates.

You expect that something is unique and it isn't, so things like totals are wrong.

Count provides the number of rows in a table that contain a non-null value.

Count distinct provides the number of unique values.

If they're different, then it means that you have duplicate values.

Similarly, if you do a group by and any group contains more than one row, then you know you have two or more occurrences of that value.

Another reason that you might have consistency problems is if extra characters have been added to the fields.

For example, you may be getting time stamps, some of which may include a time zone, or you have strings that are padded.

Use string functions to clean such data before passing it on.

For accuracy, test data against known good values.

For example, if you have an order, you could compute the subtotal from the quantity ordered and item price and make sure the math is accurate.

Similarly, you can check if a value that has been inserted belongs to a conical list of acceptable values.

You can do that with a SQL in.

For completeness, identify any missing values and either filter out or replace them with something reasonable.

If the missing value is null, SQL provides functions like NULLIF, COUNTIF, COALESCE, etcetera, to filter missing values out of calculations.

You might be able to do a union from another source to account for missing months of data.

The automatic process of detecting data drops and requesting data items to fill in the gaps is called backfilling.

It is a feature of some data transfer services.

When loading data, verify file integrity with checksum values, hash, MD5.

What happens if you were storing some value in centimeters and suddenly you start getting the value in millimeters?

Your data warehouse will end up with nonuniform data.

You have to safeguard against this.

Use SQL CAST to avoid issues with data types changing within a table.

Use the SQL format function to clearly indicate units, and in general document them very clearly.

I hope that what you are coming away with is the idea that BigQuery's SQL is very powerful, and you can take advantage of this.

**Shortcomings.**

In the previous lesson, we showed you some of the ways in which you can use SQL in an ELT pipeline to safeguard against quality issues.

The point is that you don't always need ETL.

ELT might be an option even if you need transformation.

However, there are situations where ELT won't be enough.

In that case, ETL might be what you need to do.

What are the kinds of situations where it is appropriate?

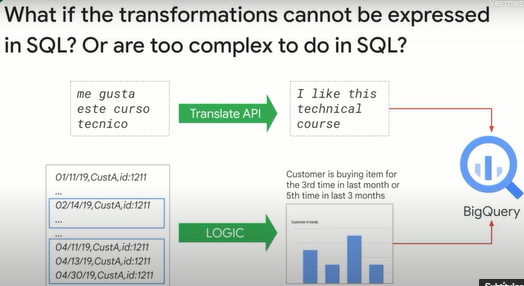
The first example, translating Spanish to English,requires calling an external API.

This cannot be done in SQL.

The second example, looking at a stream of customer actions over a time window, is rather complex.

You can do it with windowed aggregations, but it is far simpler with programmatic logic.

So if the transformations cannot be expressed in SQL or are too complex to do in SQL, you might want to transform the data before loading it into BigQuery.



The reference architecture for Google Cloud suggests Dataflow as an ETL tool.

We recommend that you build ETL pipelines in Dataflow and land the data in BigQuery.

The architecture looks like this.

Extract data from Pub/Sub, Cloud Storage, Cloud Spanner, Cloud SQL, etcetera.

Transform the data using Dataflow, and have the Dataflow pipeline write to BigQuery.

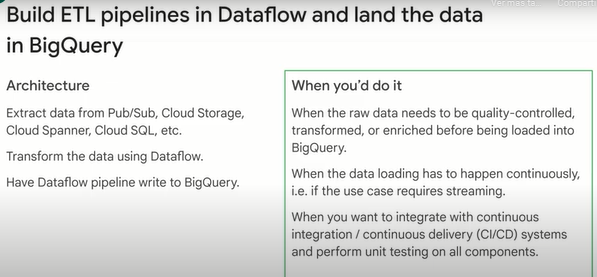
When would you do this?

When the raw data needs to be quality controlled, transformed, or enriched before being loaded into BigQuery and the transforms are difficult to do in SQL.

When the data loading has to happen continuously, i.e., if the use case requires streaming, Dataflow supports streaming.

We'll look at streaming in more detail in the next course.

And when you want to integrate with continuous integration, continuous delivery, CI/CD systems and perform unit testing on all components, it's easy to schedule the launch of a Dataflow pipeline.



Dataflow is not the only option you have on Google Cloud if you want to do ETL.

In this course, we will look at several data processing and transformation services that Google Cloud provides: Dataflow, Dataproc and Data Fusion.

Dataproc and Dataflow can be used for more complex ETL pipelines.

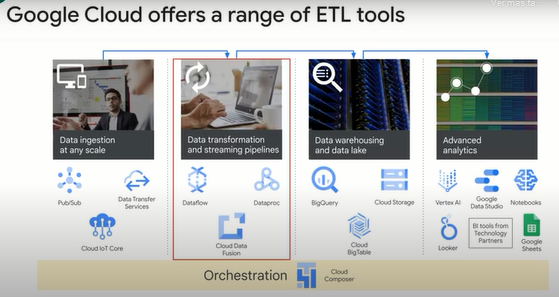
Dataproc is based on Apache Hadoop and requires significant Hadoop expertise to leverage directly.

Data Fusion provides a simple graphical interface to build ETL pipelines that can then be easily deployed at scale to Dataproc clusters.

Dataflow is a fully managed, serverless data processing service based on Apache Beam that supports both batch and streaming data processing pipelines.

While significant Apache Beam expertise is desirable in order to leverage the full power of Dataflow, Google also provides Quick Start templates for Dataflow to allow you to rapidly deploy a number of useful data pipelines.

You can use any of these three products to carry out data transformation and then store the data in a data lake or data warehouse to support advanced analytics.



**ETL to solve data quality issues.**

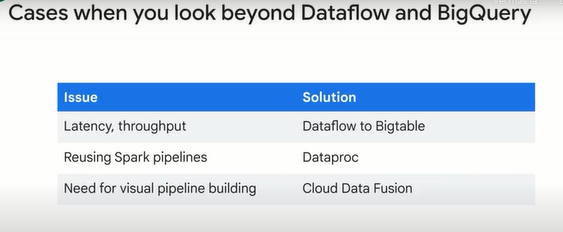
So now let's look at using ETL to solve data quality issues. Unless you have specific needs, we recommend that you use Dataflow and BigQuery. What are a few needs that cannot be met easily with Dataflow and BigQuery? Low latency and high throughput. BigQuery queries are subject to a latency

on the order of a few hundred milliseconds, and you can stream on the order of a million rows per second into a BigQuery table. This used to be 100,000 rows, but recently it got raised to 1 million per project if you can live with not having best-effort deduplication.

The typical latency number quoted for BigQuery is on the order of a second, but with BI Engine, it is possible to get latency on the order of 100 milliseconds. You should always check the documentation and the solutions pages for the latest values. If your latency and throughput considerations are more stringent,

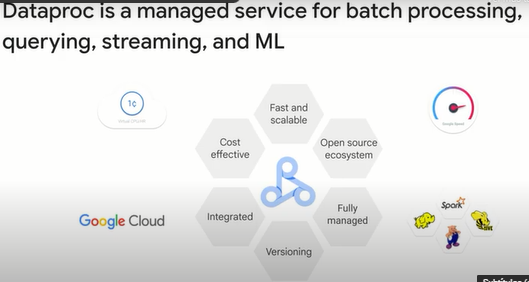
then Cloud Bigtable might be a better sync for your data processing pipelines. Reusing Spark pipelines: Maybe you already have a significant investment in Hadoop and Spark. In that case, you might be a lot more productive in a familiar technology. Use Spark if that's what you know really well.

Need for visual pipeline building: Dataflow required you to code data pipelines in Java or Python. If you want to have data analysts and nontechnical users create data pipelines, use Cloud Data Fusion. They can drag and drop and visually build pipelines. We look at all of these options briefly now and in greater detail in the remainder of this course.



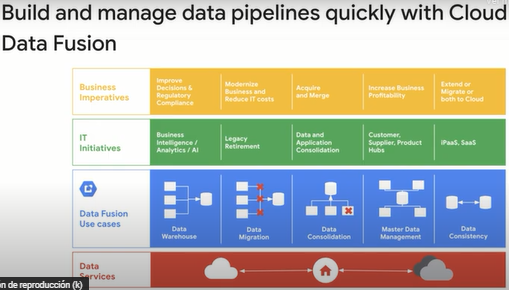
Dataproc is a managed service for batch processing, querying, streaming and machine learning. It is a service for Hadoop workloads and is quite cost effective when taking into consideration eliminating the tasks related to running Hadoop on bare metal and taking on all of the related maintenance activities.

It also has a few powerful features like autoscaling and out-of-the-box integration with Google Cloud products like BigQuery.



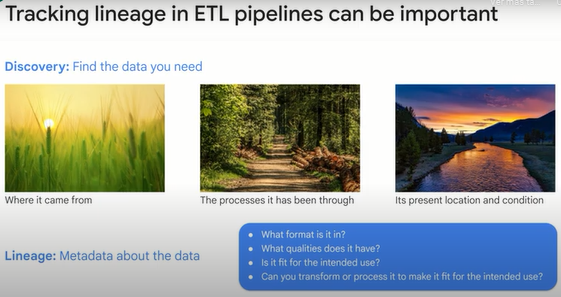
Cloud Data Fusion is a fully managed cloud-native enterprise data integration service for quickly building and managing data pipelines. You can use it to populate a data warehouse, but you can also use it for transformations and cleanup

and ensuring data consistency. Users who can be in nonprogramming roles can build visual pipelines to address business imperatives like regulatory compliance without having having to wait for an IT team to write a dataflow pipeline. Data Fusion also has a flexible API so IT staff create scripts to automate execution.



Regardless of which ETL uses Dataflow, Dataproc or Data Fusion, there are some crucial aspects to keep in mind. First, maintaining data lineage is important. What do we mean by lineage? Where the data came from, what processes it has been through and what condition it is in are all lineage.

If you have the lineage, you know for what kinds of uses the data is suited. If you find the data gives odd results, you can check the lineage to find out if there is a cause that can be corrected. Lineage also helps with trust and regulatory compliance. The other crosscutting (transversal) concern is that you need to keep metadata around. You need a way to track the lineage of data in your organization for discovery and identification of suitability for uses. On Google Cloud, Data Catalog provides discoverability, but you have to do your bit by adding labels.

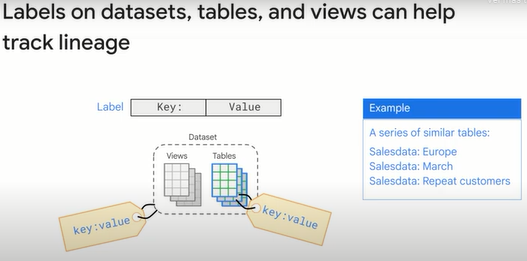


A label is a key value pair that helps you organize your resources.

In BigQuery, you can attach labels to datasets, tables and views.

Labels are useful for managing complex resources because you can filter them based on their labels.

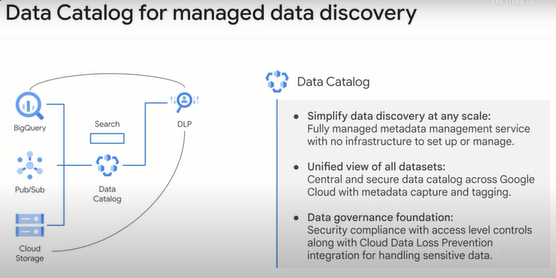
Labels are a first step towards a Data Catalog. Among the things that labels help with is cloud billing. If you attach labels to Compute Engine instances and to buckets and to data flow pipelines, then you have a way to get a fine-grained look at your cloud bill because the information about labels is forwarded to the billing system, and so you can break down your billing charges by label.



Data Catalog is a fully managed and highly scalable data discovery and metadata management service. It is serverless and requires no infrastructure to set up or manage. It provides access-level controls and honors source ACLs for read, write and search for the data assets, giving you enterprise-grade access control. Think of Data Catalog as a metadata as a service. It provides metadata management service for cataloging data assets via custom APIs and the UI thereby (de este modo) providing a unified view of data wherever it is. It supports schematized tags, for example, enum, bool, daytime and not just simple text tags, providing organizations rich and organization business metadata. If offers unified data discovery of all data assets spread across multiple projects and systems. It comes with a simple and easy-to-use search UI to quickly and easily find data assets powered by the same Google search technology that supports Gmail and Drive. As a central catalog, it provides a flexible and powerful cataloging system for capturing both technical metadata automatically

as well as business metadata tags in a structured format. One of the great things about the data discovery is that it integrates with the cloud data loss prevention API. You can use it to discover and classify sensitive data, providing intelligence and helping to simplify the process of governing your data.

Data Catalog empowers users to annotate business metadata in a collaborative manner and provides the foundation for data governance. Specifically, Data Catalog makes all of the metadata about your datasets available to search for your users regardless of where the data is stored. Using Data Catalog, you can group datasets together with tags, flag certain columns as containing sensitive data, etcetera. Why is this useful? If you have many different sets with many different tables to which different users have different access levels, the Data Catalog provides a single unified user experience for discovering those datasets quickly. No more hunting for specific table names in the data basis which may not be accessible by all users.



**Quizz.**

Which of the following is the ideal use case for Extract and Load (EL)

**Scheduled periodic loads of log files (e.g. once a day)**

When the raw data needs to be quality-controlled, transformed, or enriched before being loaded into BigQuery

When the data loading has to happen continuously

When you want to integrate with continuous integration / continuous delivery (CI/CD) systems and perform unit testing on all components.