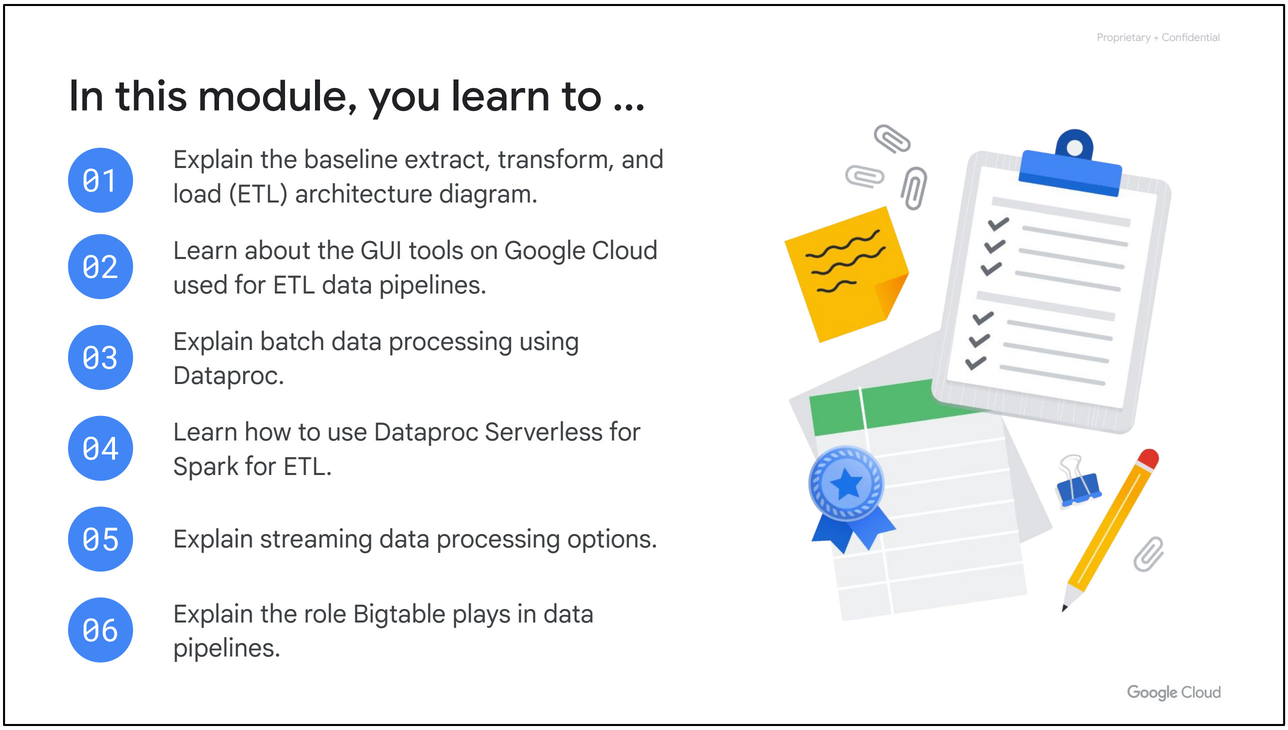
The Extract, Transform, and Load Data Pipeline Pattern



In this module, you learn to ...

01 Explain the baseline extract, transform, and load (ETL) architecture diagram.

02 Learn about the GUI tools on Google Cloud used for ETL data pipelines.

03 Explain batch data processing using Dataproc.

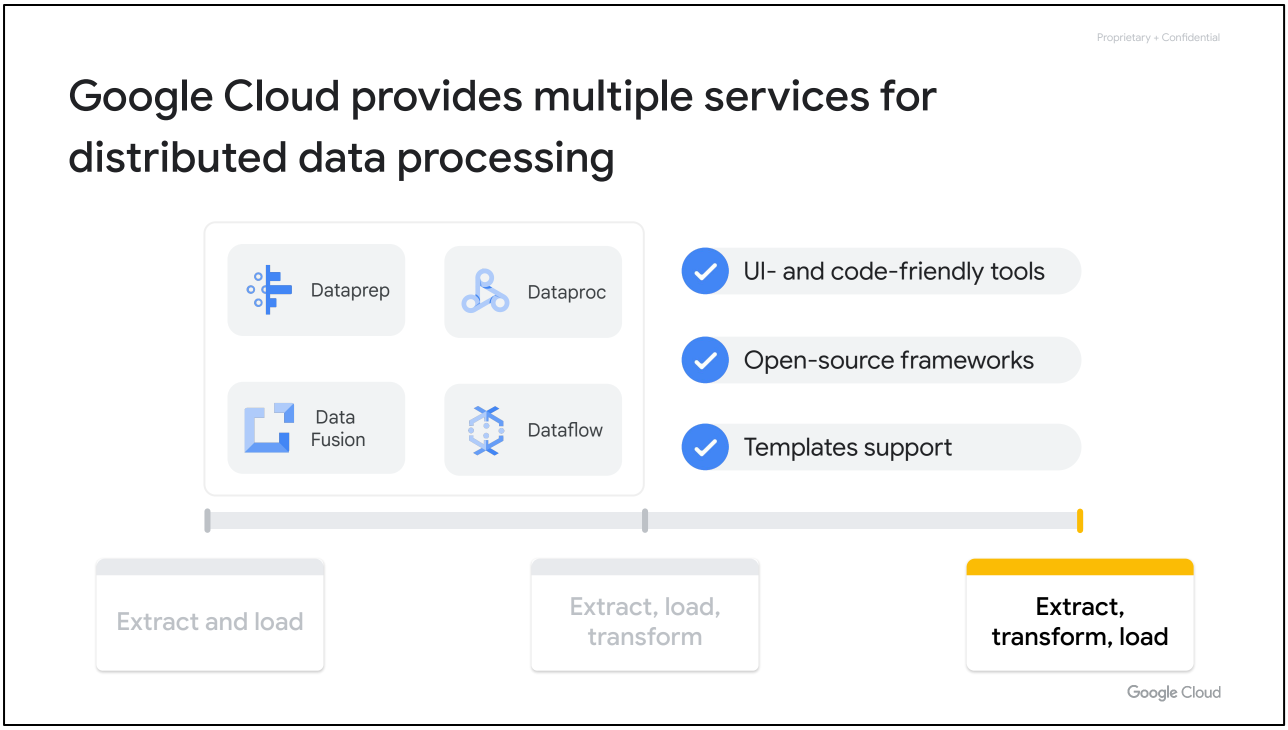
04 Learn how to use Dataproc Serverless for Spark for ETL.

05 Explain streaming data processing options.

06 Explain the role Bigtable plays in data pipelines.

Google Cloud

In this module, first, you review the baseline extract, transform, and load architecture diagram. Second, you look at the GUI tools on Google Cloud used for ETL data pipelines. Third, you review batch data processing using Dataproc. Then, you examine using Dataproc Serverless for Spark for ETL. Next, you review streaming data processing options on Google Cloud. Finally, you examine the role Bigtable plays in data pipelines.



Proprietary + Confidential

Google Cloud provides multiple services for distributed data processing

Dataprep

Dataproc

Data Fusion

Dataflow

✓ UI- and code-friendly tools

✓ Open-source frameworks

✓ Templates support

Extract and load

Extract, load, transform

Extract, transform, load

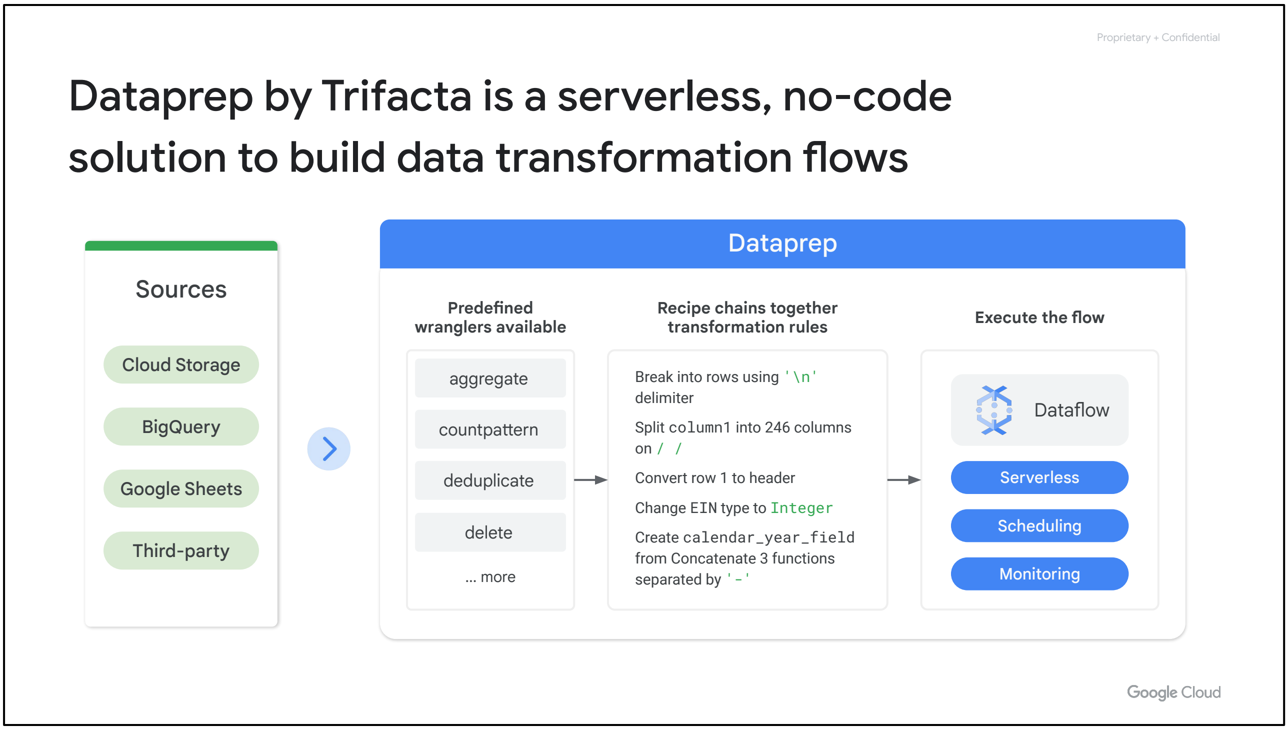
Google Cloud

The extract, transform, and load data pipeline pattern focuses on data being adjusted or transformed prior to being loaded into BigQuery. Google Cloud offers a variety of services to handle distributed data processing.

For developers who prefer visual interfaces, there are user-friendly tools like Dataprep and Data Fusion.

If you favor open-source frameworks, Dataproc and Dataflow are an option.

And to streamline your workflows, template support is provided across various stages of data processing, from extraction and loading to full-fledged transformation.



Proprietary + Confidential

Dataprep by Trifacta is a serverless, no-code solution to build data transformation flows

Sources

- Cloud Storage
- BigQuery
- Google Sheets
- Third-party

Dataprep

Predefined wranglers available

- aggregate
- countpattern
- deduplicate
- delete
- ... more

Recipe chains together transformation rules

Break into rows using `'\n'` delimiter

Split `column1` into 246 columns on `/ /`

Convert row 1 to header

Change `EIN` type to `Integer`

Create `calendar_year_field` from Concatenate 3 functions separated by `'-'`

Execute the flow

Dataflow

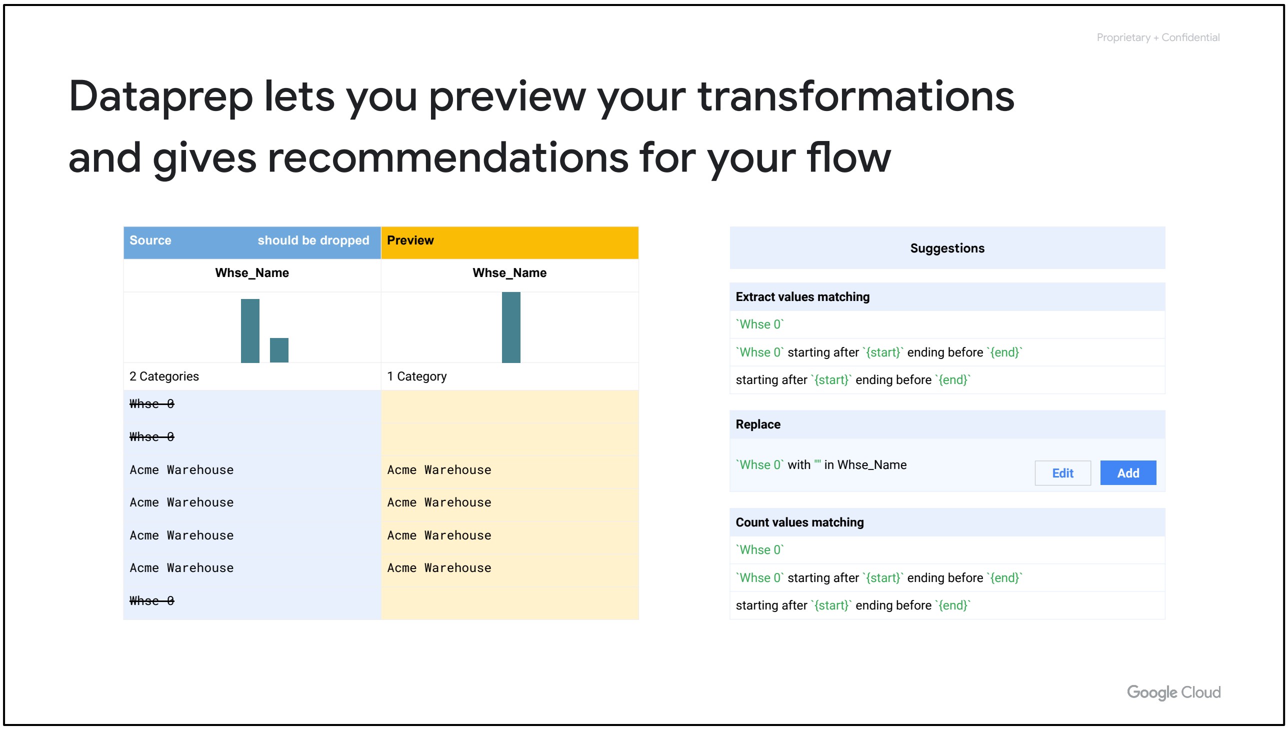
- Serverless
- Scheduling
- Monitoring

Google Cloud

Dataprep by Trifacta simplifies data transformation by offering a serverless, no-code solution.

It connects to various data sources, provides pre-built transformation functions, and allows users to chain these functions together into recipes.

The resulting transformation flows can be executed seamlessly with Dataflow, while also providing capabilities for scheduling and monitoring.



Dataprep lets you preview your transformations and gives recommendations for your flow

Source should be dropped	Preview
Whse_Name	Whse_Name
2 Categories	1 Category
~~Whse 0~~	
~~Whse 0~~	
Acme Warehouse	Acme Warehouse
Acme Warehouse	Acme Warehouse
Acme Warehouse	Acme Warehouse
Acme Warehouse	Acme Warehouse
~~Whse 0~~	

Suggestions

Extract values matching

`Whse 0`

`Whse 0` starting after `{start}` ending before `{end}`

starting after `{start}` ending before `{end}`

Replace

`Whse 0` with "" in Whse_Name

Count values matching

`Whse 0`

`Whse 0` starting after `{start}` ending before `{end}`

starting after `{start}` ending before `{end}`

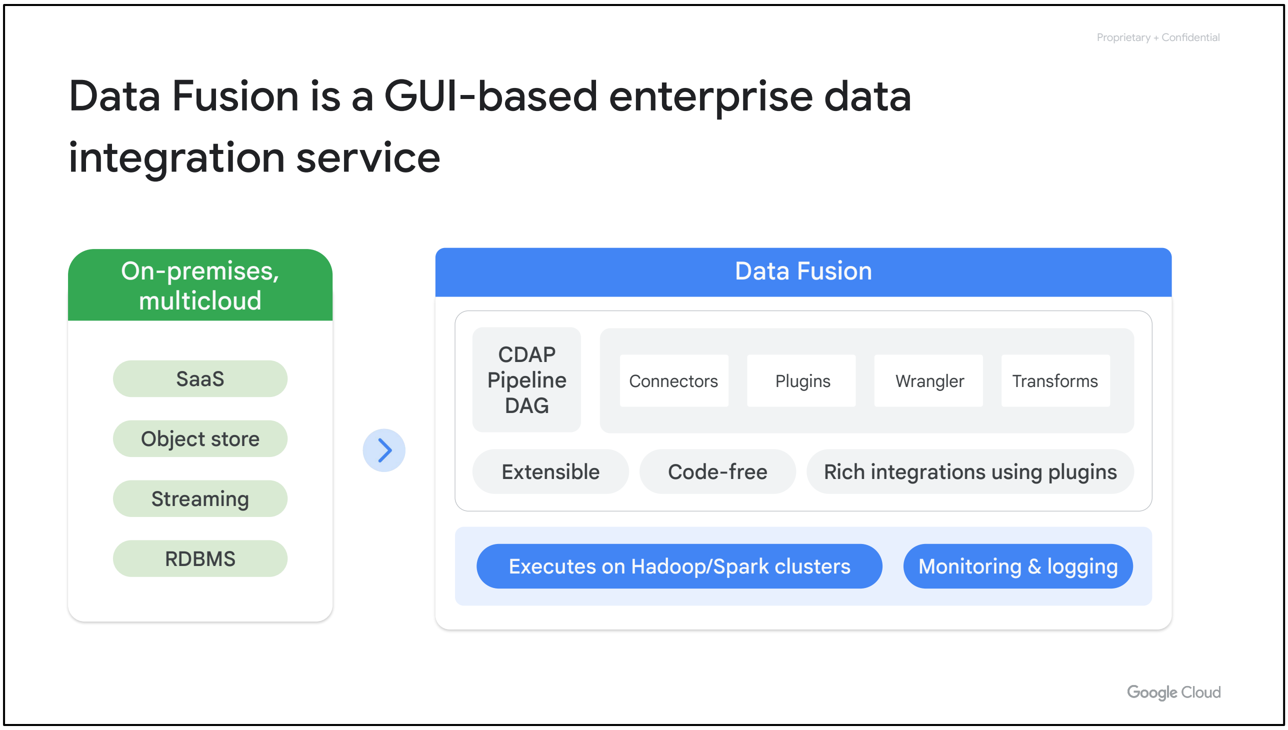
Google Cloud

Dataprep provides a visual interface where you can see the impact of your data transformations before applying them.

It also offers intelligent suggestions to help you refine your data cleaning and preparation process, like extracting specific values or replacing patterns within your data.

References:

https://help.alteryx.com/Dataprep/en/product-overview.html##



Proprietary + Confidential

Data Fusion is a GUI-based enterprise data integration service

On-premises, multicloud

- SaaS
- Object store
- Streaming
- RDBMS

Data Fusion

- CDAP Pipeline DAG
- Connectors
- Plugins
- Wrangler
- Transforms
- Extensible
- Code-free
- Rich integrations using plugins
- Executes on Hadoop/Spark clusters
- Monitoring & logging

Google Cloud

Data Fusion is a user-friendly, GUI-based tool designed for enterprise data integration.

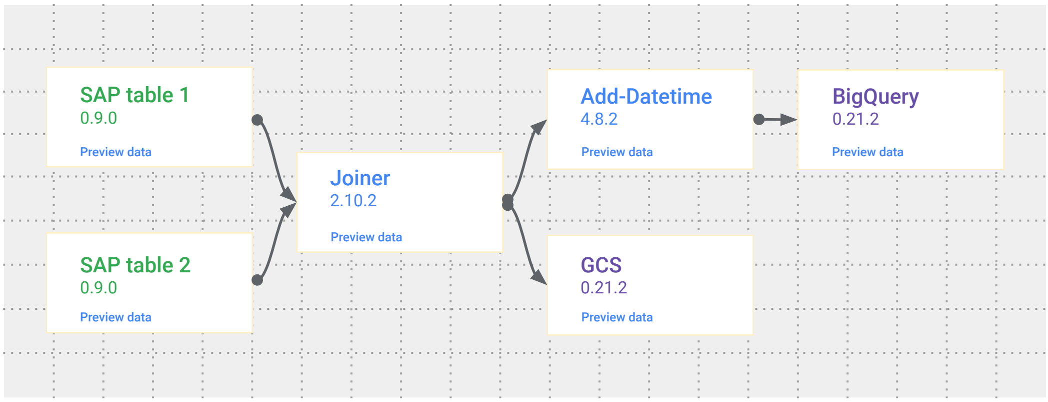
Data Fusion seamlessly connects to various data sources, both on-premises and in the cloud.

You can build data pipelines without coding using its drag-and-drop interface and pre-built transformations.

The platform is extensible, allowing for custom plugins, and it executes on powerful Hadoop/Spark clusters for efficient processing.

References:
Huge set of plugins available to integrate data from everywhere:
https://cloud.google.com/data-fusion/plugins?hl=en

Create and deploy data integration pipelines in Data Fusion Studio



SAP table 1
0.9.0
Preview data

SAP table 2
0.9.0
Preview data

Joiner
2.10.2
Preview data

Add-Datetime
4.8.2
Preview data

GCS
0.21.2
Preview data

BigQuery
0.21.2
Preview data

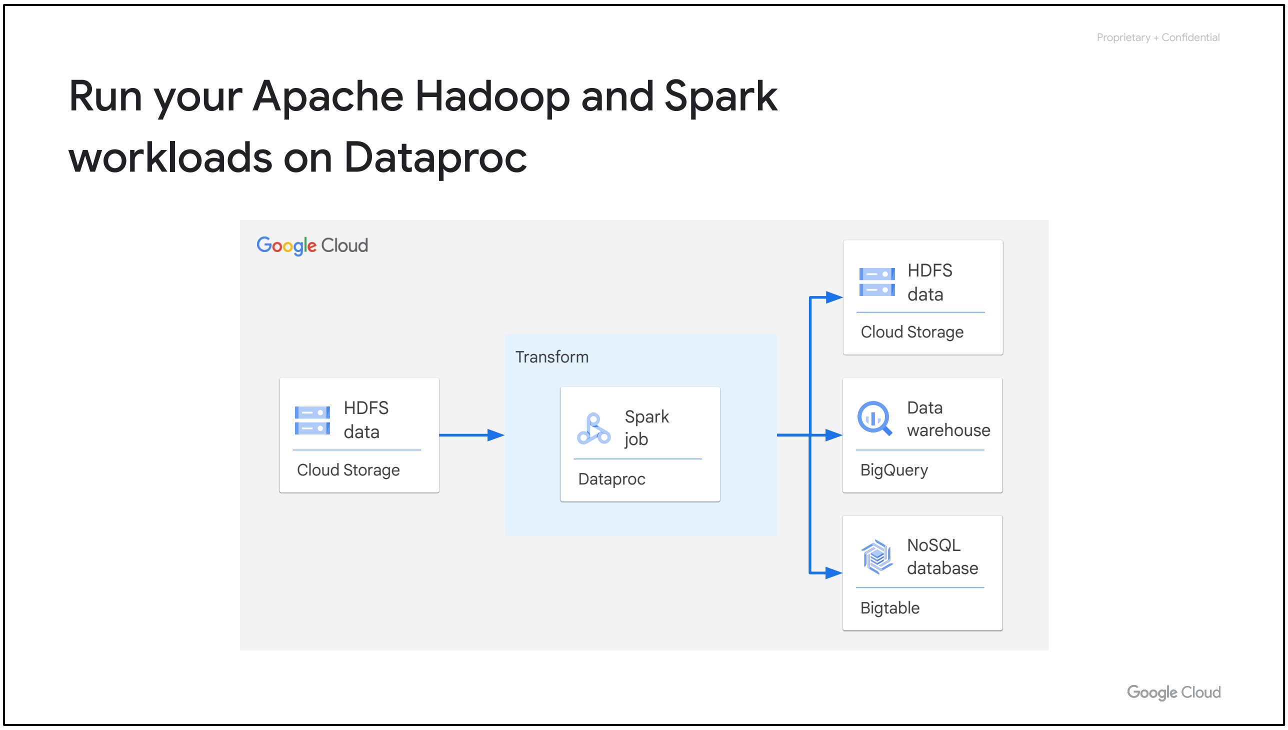
Data Fusion Studio easily allows the creation of data pipelines with visual tools.

The example pipeline shows two SAP tables being used as data sources. The two tables are joined together and then one outbound leg is written to a Cloud Storage bucket. The other leg undergoes an "Add-Datetime" transformation and is outputted to BigQuery.

The pipeline also highlights the ability to preview data at different stages.

References:

reference for this integration pipeline (SAP>BQ):
https://blog.searce.com/sap-to-bigquery-data-ingestion-using-data-fusion-part-2-dde072d2cf6f



Run your Apache Hadoop and Spark workloads on Dataproc

Google Cloud

HDFS data

Cloud Storage

Transform

Spark job

Dataproc

HDFS data

Cloud Storage

Data warehouse

BigQuery

NoSQL database

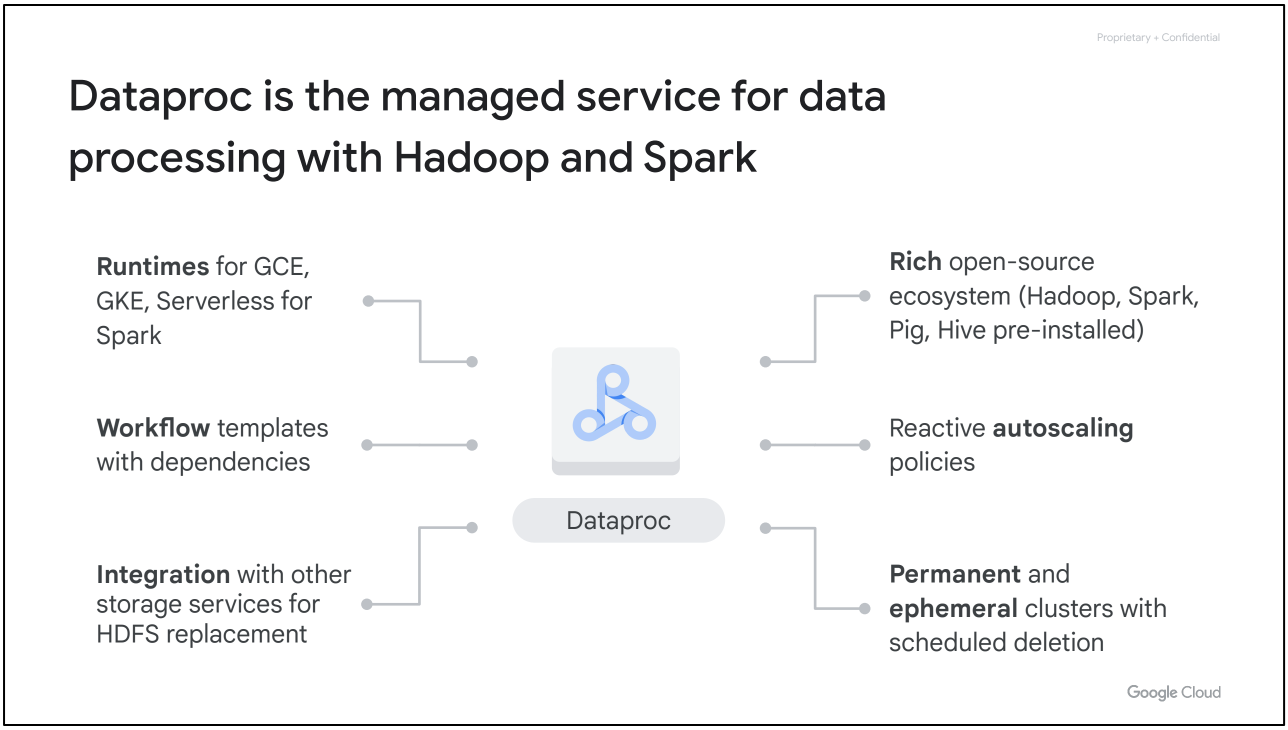
Bigtable

Google Cloud

Dataproc allows you to seamlessly run your Apache Hadoop and Spark workloads on Google Cloud.

You can leverage HDFS data stored on Cloud Storage, and use Dataproc to perform transformations with Spark jobs.

The results can then be easily stored in various destinations like Cloud Storage, BigQuery, or NoSQL databases like Bigtable, all within the Google Cloud ecosystem.



Proprietary + Confidential

Dataproc is the managed service for data processing with Hadoop and Spark

Runtimes for GCE, GKE, Serverless for Spark

Workflow templates with dependencies

Integration with other storage services for HDFS replacement

Dataproc

Rich open-source ecosystem (Hadoop, Spark, Pig, Hive pre-installed)

Reactive **autoscaling** policies

Permanent and **ephemeral** clusters with scheduled deletion

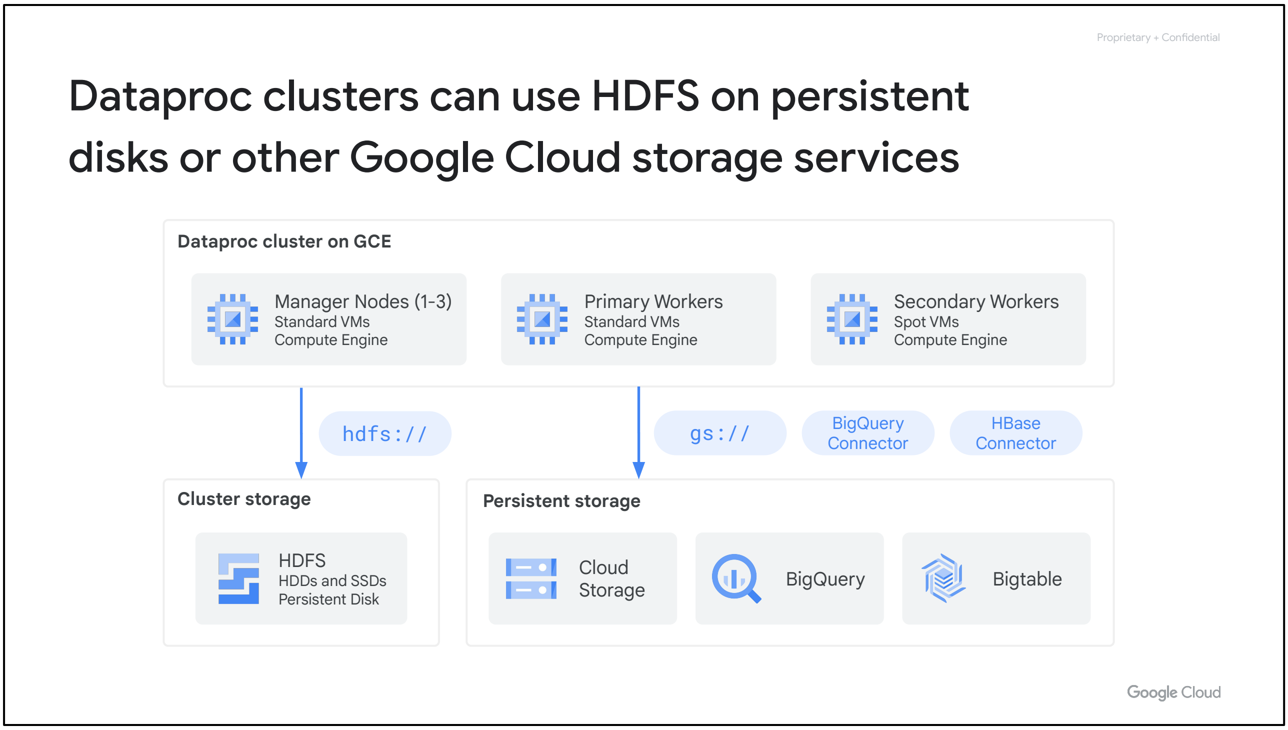
Google Cloud

Dataproc is Google Cloud's managed service for data processing using Hadoop and Spark.

It offers flexibility with runtimes on GCE, GKE, and Serverless Spark, and provides a rich open-source ecosystem.

Dataproc simplifies cluster management with workflow templates, autoscaling, and the option for both permanent and ephemeral clusters.

It also integrates seamlessly with other Google Cloud storage services, eliminating the need for disk-based HDFS.



Dataproc clusters can use HDFS on persistent disks or other Google Cloud storage services

Dataproc cluster on GCE

Manager Nodes (1-3)
Standard VMs
Compute Engine

Primary Workers
Standard VMs
Compute Engine

Secondary Workers
Spot VMs
Compute Engine

hdfs://

gs://

BigQuery Connector

HBase Connector

Cluster storage

HDFS
HDDs and SSDs
Persistent Disk

Persistent storage

Cloud Storage

BigQuery

Bigtable

Dataproc clusters on Google Compute Engine offer flexible storage options.

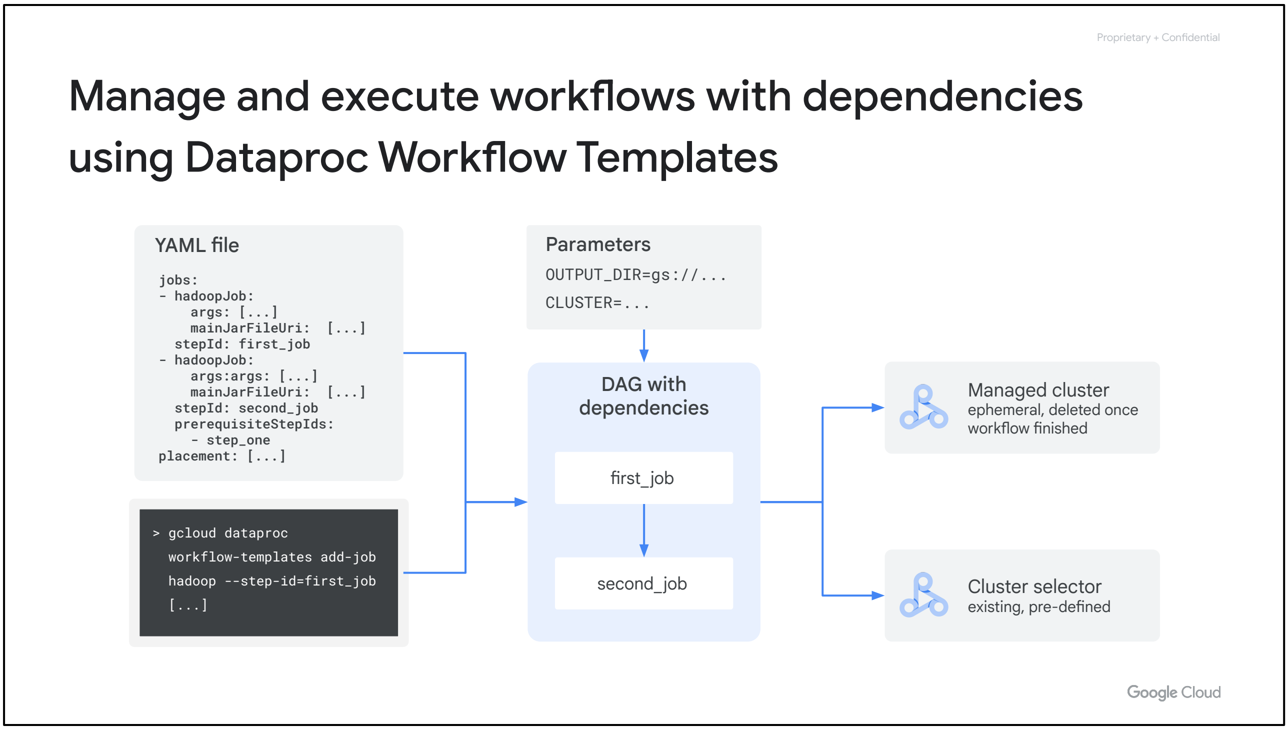
Clusters can utilize HDFS on persistent disks for cluster storage, or leverage other Google Cloud storage services like Cloud Storage for persistent data.

Additionally, Dataproc integrates with BigQuery and Bigtable using connectors, enabling seamless interaction with these data stores.

This setup allows users to choose the most suitable storage solution for their specific needs while taking advantage of Dataproc's processing capabilities.

Instructor notes:

- Connectors (i.e. gs:// > Cloud Storage connector) pre-installed on cluster, so replacing hdfs:// with gs:// works in most cases in the code to point to HDFS on Cloud Storage
- HDFS on Cloud Storage enables ephemeral clusters, since we don’t have to keep the cluster up and running due to HDFS on persistent disk
- ephemeral clusters reduce cost, since they just live as long as the submitted job runs



Manage and execute workflows with dependencies using Dataproc Workflow Templates

YAML file

```
jobs:
- hadoopJob:
    args: [...]
    mainJarFileUri:  [...]
  stepId: first_job
- hadoopJob:
    args:args: [...]
    mainJarFileUri:  [...]
  stepId: second_job
  prerequisiteStepIds:
    - step_one
placement: [...]
```

```
> gcloud dataproc
  workflow-templates add-job
  hadoop --step-id=first_job
  [...]
```

Parameters

```
OUTPUT_DIR=gs://...
CLUSTER=...
```

DAG with dependencies

first_job

second_job

Managed cluster
ephemeral, deleted once workflow finished

Cluster selector
existing, pre-defined

Dataproc Workflow Templates allow you to define and manage complex data processing workflows with dependencies between jobs.

You can specify these workflows in a YAML file, providing details about the jobs like Hadoop or Spark, their order of execution, and any required parameters.

These templates can then be submitted to Google Cloud using the 'gcloud' command-line tool, where they will be executed on either a managed, ephemeral cluster or an existing, pre-defined cluster.

References:

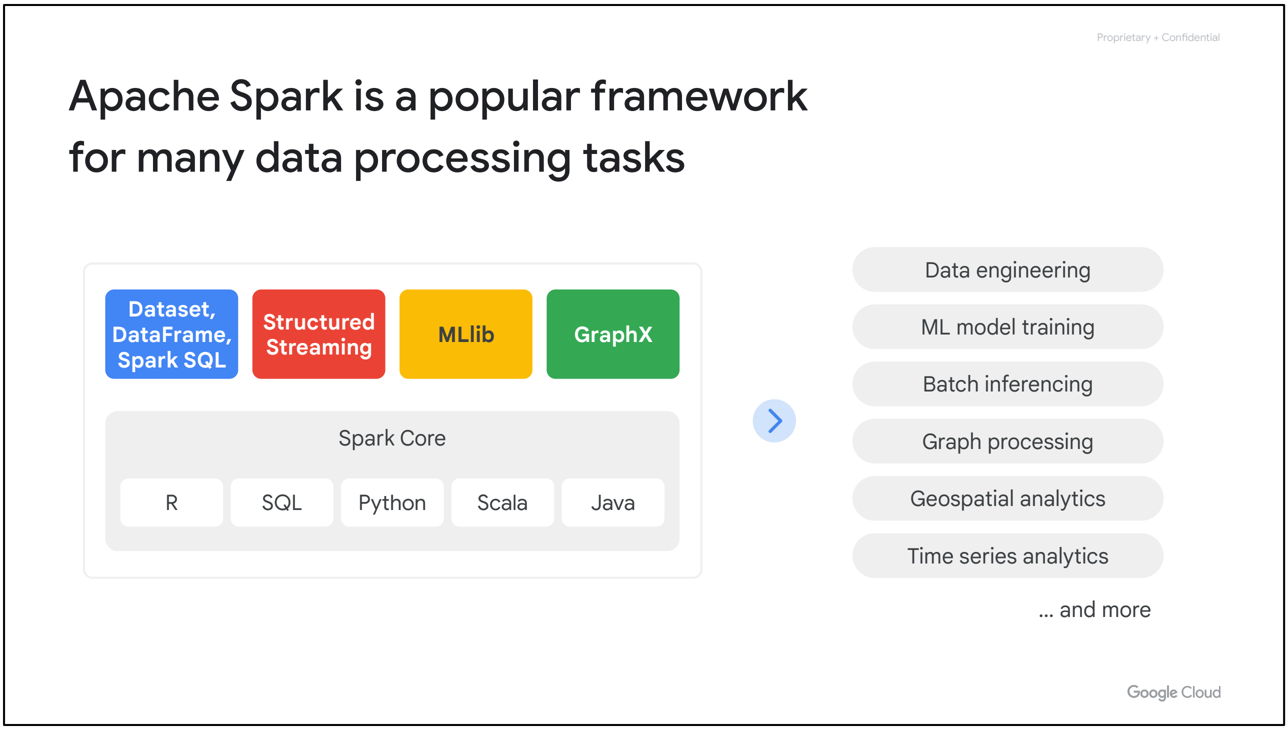
Two ways to create workflow templates:
YAML: https://cloud.google.com/dataproc/docs/concepts/workflows/use-yamls
gcloud: https://cloud.google.com/dataproc/docs/concepts/workflows/use-workflows

Parameters:
https://cloud.google.com/dataproc/docs/concepts/workflows/workflow-parameters

Instructor notes:

Flow:

- create template using YAML or command line, here you outline all the steps (jobs) and their dependencies (prerequisiteStepIds)
- run template, you can pass in parameters so that you don’t have to edit the workflow again, e.g. different output directory
- Dataproc workflow template service creates a DAG with all the steps (jobs) and dependencies
- jobs are submitted either to managed or existing cluster, this is set during workflow template creation (placement)



Apache Spark is a popular framework for many data processing tasks

Dataset, DataFrame, Spark SQL

Structured Streaming

MLlib

GraphX

Spark Core

R

SQL

Python

Scala

Java

Data engineering

ML model training

Batch inferencing

Graph processing

Geospatial analytics

Time series analytics

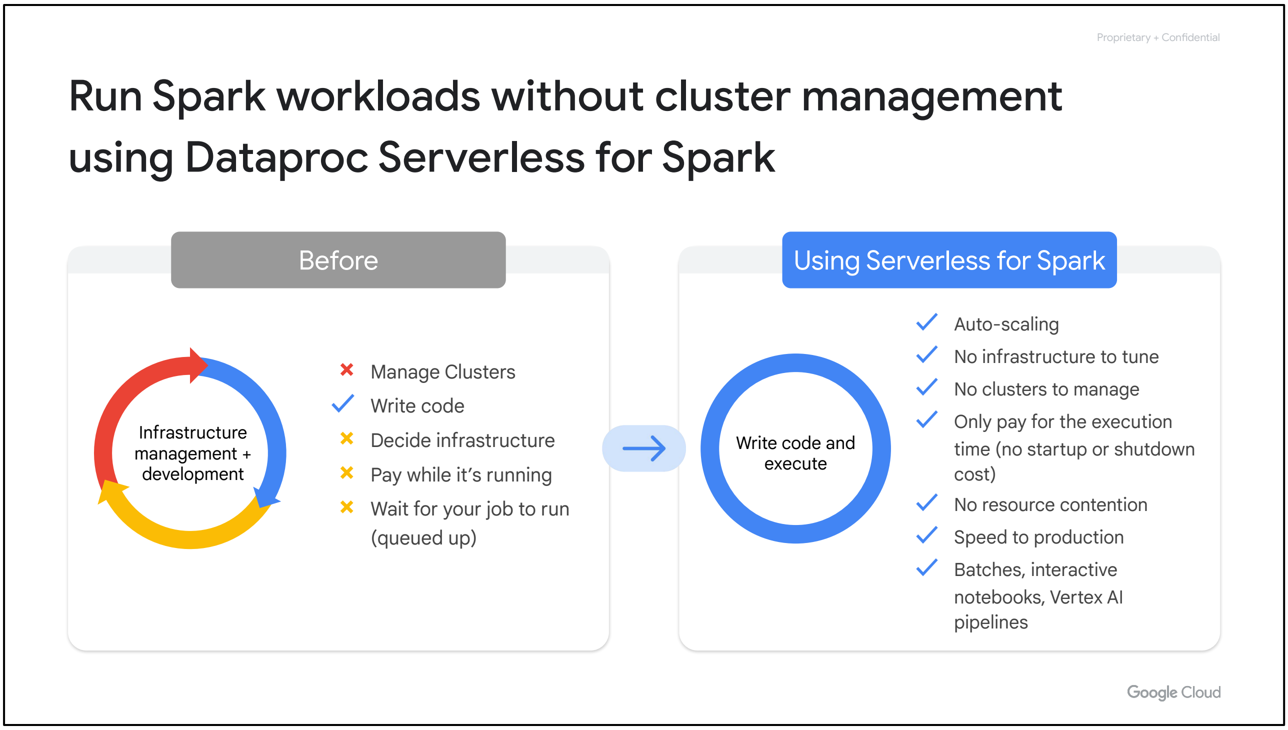
... and more

Google Cloud

Apache Spark is a versatile framework for data processing, offering various capabilities through its components like Spark SQL for structured data, Spark Streaming for real-time data, MLlib for machine learning, and GraphX for graph processing.

Spark supports multiple languages including R, SQL, Python, Scala, and Java, making it accessible to a wide range of users.

With these features, Spark excels in tasks like data engineering, machine learning, analytics, and many more.



Proprietary + Confidential

Run Spark workloads without cluster management using Dataproc Serverless for Spark

Before

Infrastructure management + development

- ✗ Manage Clusters
- ✓ Write code
- ✗ Decide infrastructure
- ✗ Pay while it's running
- ✗ Wait for your job to run (queued up)

→

Using Serverless for Spark

Write code and execute

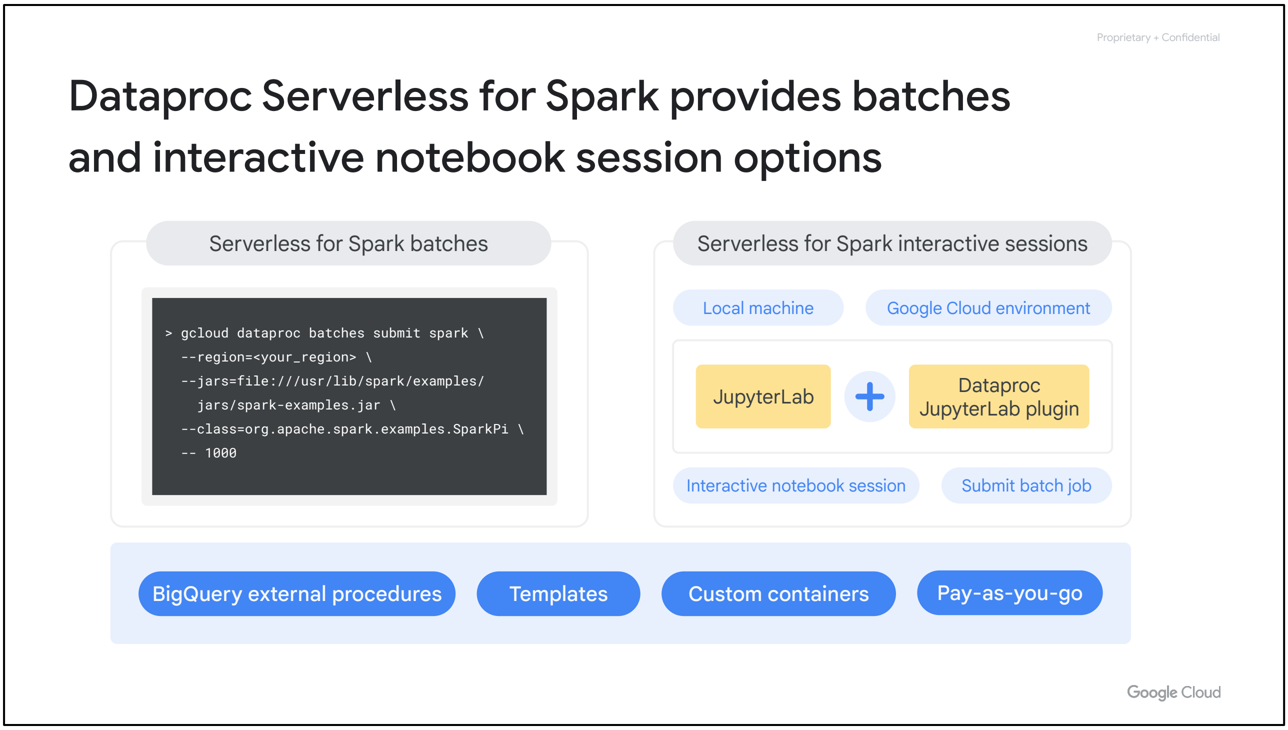
- ✓ Auto-scaling
- ✓ No infrastructure to tune
- ✓ No clusters to manage
- ✓ Only pay for the execution time (no startup or shutdown cost)
- ✓ No resource contention
- ✓ Speed to production
- ✓ Batches, interactive notebooks, Vertex AI pipelines

Google Cloud

Dataproc Serverless for Spark simplifies Spark workload execution by eliminating cluster management.

It offers automatic scaling, cost efficiency with pay-per-execution pricing, faster deployment, and no resource contention.

Users can focus solely on writing and executing their code, making it ideal for various Spark use cases like batch processing, interactive notebooks, and Vertex AI pipelines.



Dataproc Serverless for Spark provides batches and interactive notebook session options

Serverless for Spark batches

```
> gcloud dataproc batches submit spark \
  --region=<your_region> \
  --jars=file:///usr/lib/spark/examples/
    jars/spark-examples.jar \
  --class=org.apache.spark.examples.SparkPi \
  -- 1000
```

Serverless for Spark interactive sessions

Local machine | Google Cloud environment

JupyterLab + Dataproc JupyterLab plugin

Interactive notebook session | Submit batch job

BigQuery external procedures | Templates | Custom containers | Pay-as-you-go

Google Cloud

Dataproc Serverless for Spark offers two main execution modes: Serverless for batches and Serverless for interactive notebook sessions.

Batches are submitted using the 'gcloud' command-line tool and are ideal for automated or scheduled jobs.

Interactive sessions leverage JupyterLab, either locally or within the Google Cloud environment, for interactive development and exploration.

The platform also supports features like BigQuery external procedures, templates, custom containers, and a pay-as-you-go pricing model.

References:

https://cloud.google.com/dataproc-serverless/docs/quickstarts/spark-batch
https://cloud.google.com/dataproc-serverless/docs/quickstarts/jupyterlab-sessions

Instructor notes:

batches command on the slide explained:

- this class is pre-installed as an example in Spark libraries to compute the

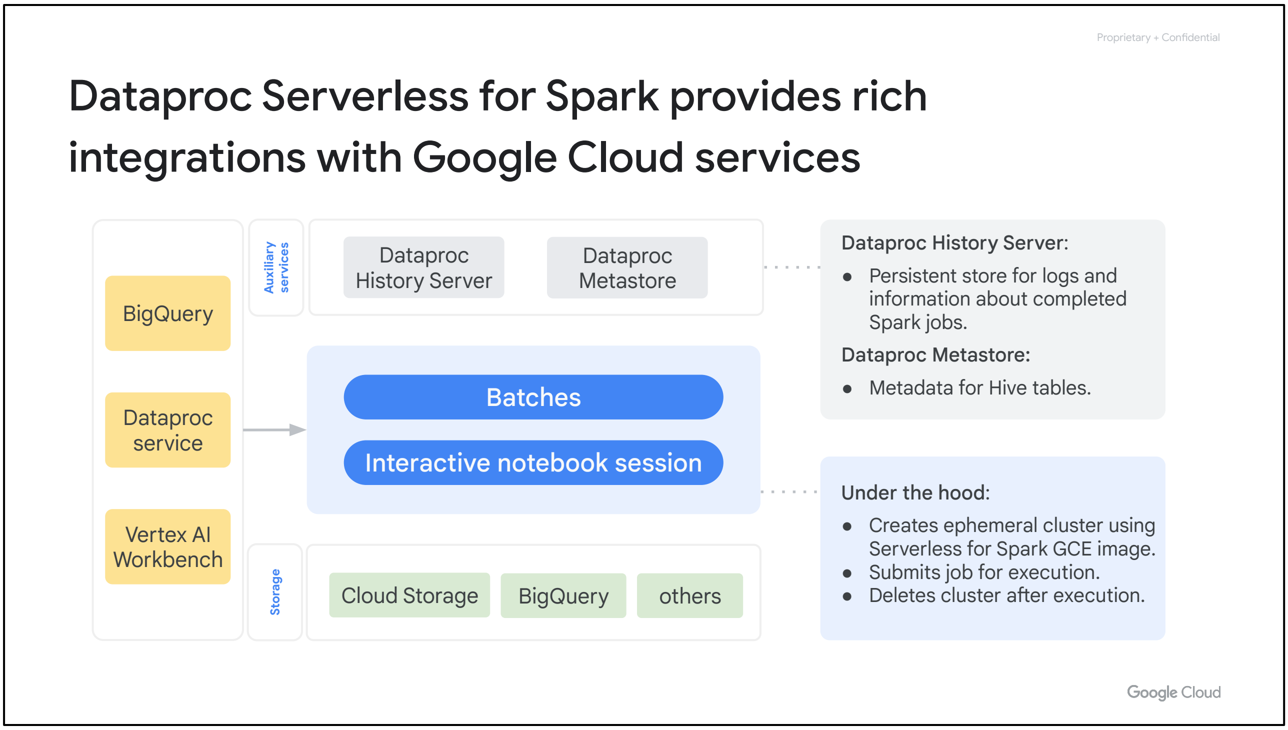
- approximate value of pi
- -- 1000 is an argument for this class (1000 represents the number of samples to use for the Monte Carlo approximation of Pi. The higher this number, the more accurate the approximation will be, but it will also take longer to compute.)

BigQuery external procedures:

- BigQuery can execute PySpark code (as discussed in M3)

Templates:

- quickly set up command Spark batch workloads without writing code (e.g. Cloud Storage to BigQuery)
- https://cloud.google.com/dataproc-serverless/docs/templates/templates



Proprietary + Confidential

Dataproc Serverless for Spark provides rich integrations with Google Cloud services

BigQuery

Dataproc service

Vertex AI Workbench

Auxiliary services

Dataproc History Server

Dataproc Metastore

Batches

Interactive notebook session

Storage

Cloud Storage

BigQuery

others

Dataproc History Server:

- Persistent store for logs and information about completed Spark jobs.

Dataproc Metastore:

- Metadata for Hive tables.

Under the hood:

- Creates ephemeral cluster using Serverless for Spark GCE image.
- Submits job for execution.
- Deletes cluster after execution.

Google Cloud

Dataproc Serverless for Spark seamlessly integrates with various Google Cloud services, enhancing its functionality and usability.

It leverages Dataproc History Server and Dataproc Metastore for persistent storage and metadata management.

It interacts with BigQuery for data warehousing and analytics, and with Vertex AI Workbench for machine learning tasks.

Additionally, it utilizes Cloud Storage and other storage services for data storage and retrieval. Behind the scenes, it creates and manages ephemeral clusters for efficient job execution.

Instructor Notes:

Main value for Interactive:

- before, users had to spin up a cluster themselves (if they didn’t have a long-running one to use this for)
- then they code and develop and experiment with the Notebook
- then maybe they forget to shut down the cluster after everything is done
- now they can just open the Notebook with the plugin installed and off they go

BigQuery:

- external procedures for Spark; run code directly from BigQuery, schedule stored procedures
- via External Connection which then executes code on Dataproc Serverless for Spark

Vertex AI Workbench:

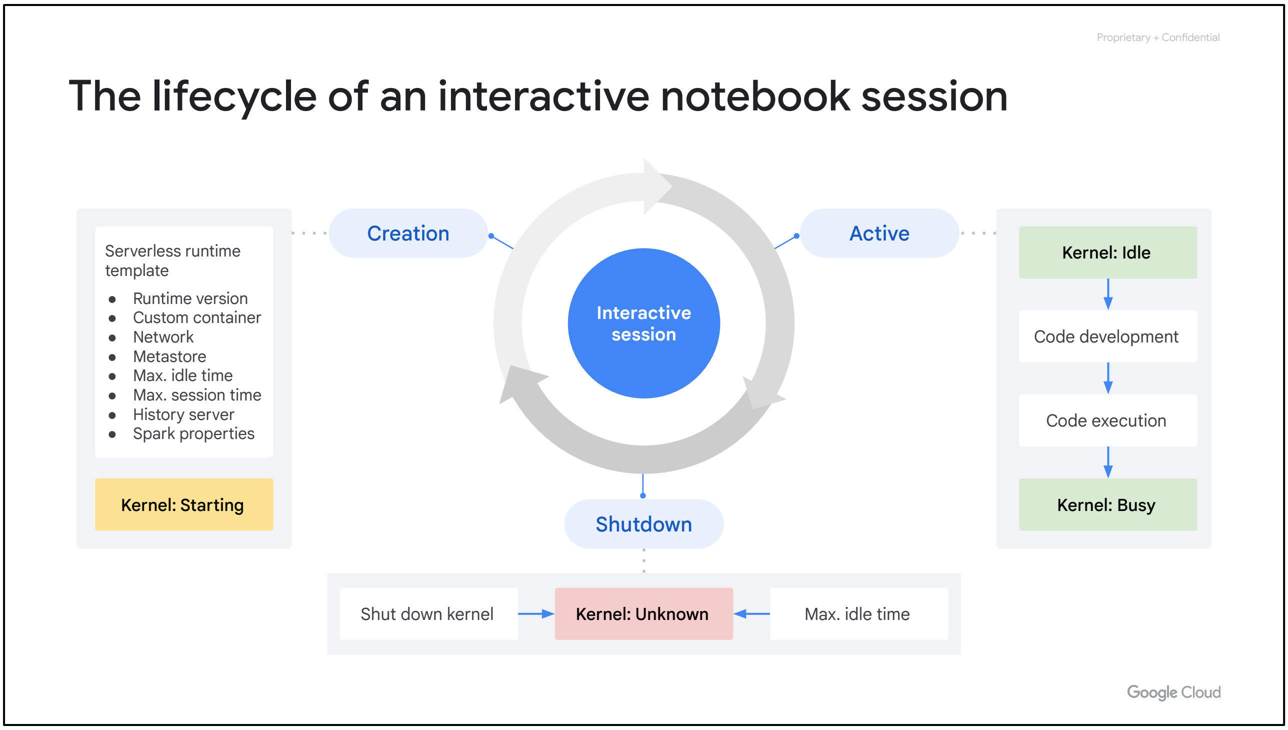
- execute Spark code through notebook executors that are integrated with Vertex AI pipelines
- Vertex AI Workbench instances have the Dataproc JupyterLab plugin preinstalled, as of version M113 and later.
- https://cloud.google.com/vertex-ai/docs/workbench/instances/create-dataproc-enabled

Dataproc History Server:

Job History and Debugging: Dataproc History Server provides a persistent store for logs and information about completed Spark jobs. This allows you to analyze job performance, track resource usage, and debug issues even after the serverless cluster has been torn down.
Long-Term Analysis: If you need to retain job information for auditing, compliance, or long-term performance analysis, the History Server offers a central location to store and access this data.

Dataproc Metastore:

Shared Metadata: If your Spark jobs interact with structured data stored in Hive tables, a Metastore is necessary to manage the schema and metadata for these tables.
Data Sharing: A central Metastore enables multiple Dataproc Serverless clusters and other data processing tools to share access to Hive tables and their metadata.
Metadata Persistence: The Metastore ensures that table definitions and schema information are preserved even if serverless clusters are created and destroyed on demand.



Proprietary + Confidential

The lifecycle of an interactive notebook session

Serverless runtime template

- Runtime version
- Custom container
- Network
- Metastore
- Max. idle time
- Max. session time
- History server
- Spark properties

Kernel: Starting

Creation

Interactive session

Active

Kernel: Idle

Code development

Code execution

Kernel: Busy

Shutdown

Shut down kernel → Kernel: Unknown ← Max. idle time

Google Cloud

The lifecycle of an interactive notebook session begins with its creation, where various configurations like runtime version and network settings are defined.

Once active, the session allows for code development and execution, with the kernel transitioning between idle and busy states.

The session eventually reaches a shutdown phase, either manually triggered or due to inactivity, leading to the kernel being shut down and its state becoming unknown.

References:

https://cloud.google.com/dataproc-serverless/docs/quickstarts/jupyterlab-sessions

Instructor notes:

Main point:

- as long as Kernel is idle or busy, you pay for the Spark workload

Creation:

- create Dataproc Serverless runtime template or choose existing one

- To open a new Serverless Spark notebook, click a runtime template. It takes about a minute for the remote Spark kernel to start. After the kernel starts, you can start coding.

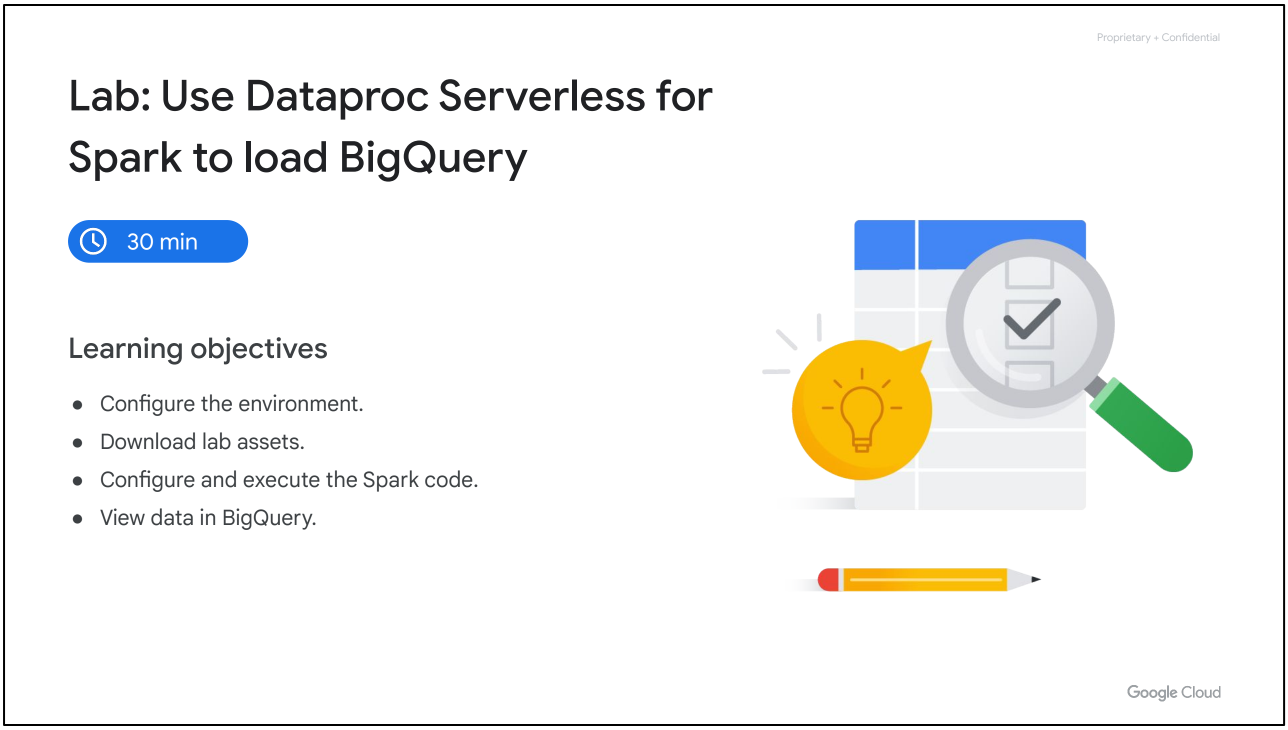
Active:

- When session creation is complete and the notebook kernel is ready to use, the kernel status changes from Unknown to Idle.
- To run your code on Serverless Spark, run a code cell in your notebook.

Shutdown:

- After creating and using a notebook, you can terminate the notebook session by clicking Shut Down Kernel from the Kernel tab.
- If you don't terminate the session, Dataproc terminates the session when the session idle timer expires. You can configure the session idle time in the runtime template configuration. The default session idle time is one hour.

You can optionally show a demo here.
Demo steps are in the guide here:
https://docs.google.com/document/d/1hLoeHfGAnyp6tyjH3LtHsVg2rB_PhGYme47-BLJ5KqQ/edit?resourcekey=0-92Txnxw9r6ZzTPll4pGEzw&tab=t.0#heading=h.5nu6fv47r8rx



Lab: Use Dataproc Serverless for Spark to load BigQuery

30 min

Learning objectives

- Configure the environment.
- Download lab assets.
- Configure and execute the Spark code.
- View data in BigQuery.

In this lab, you use Dataproc Serverless for Spark to load BigQuery.

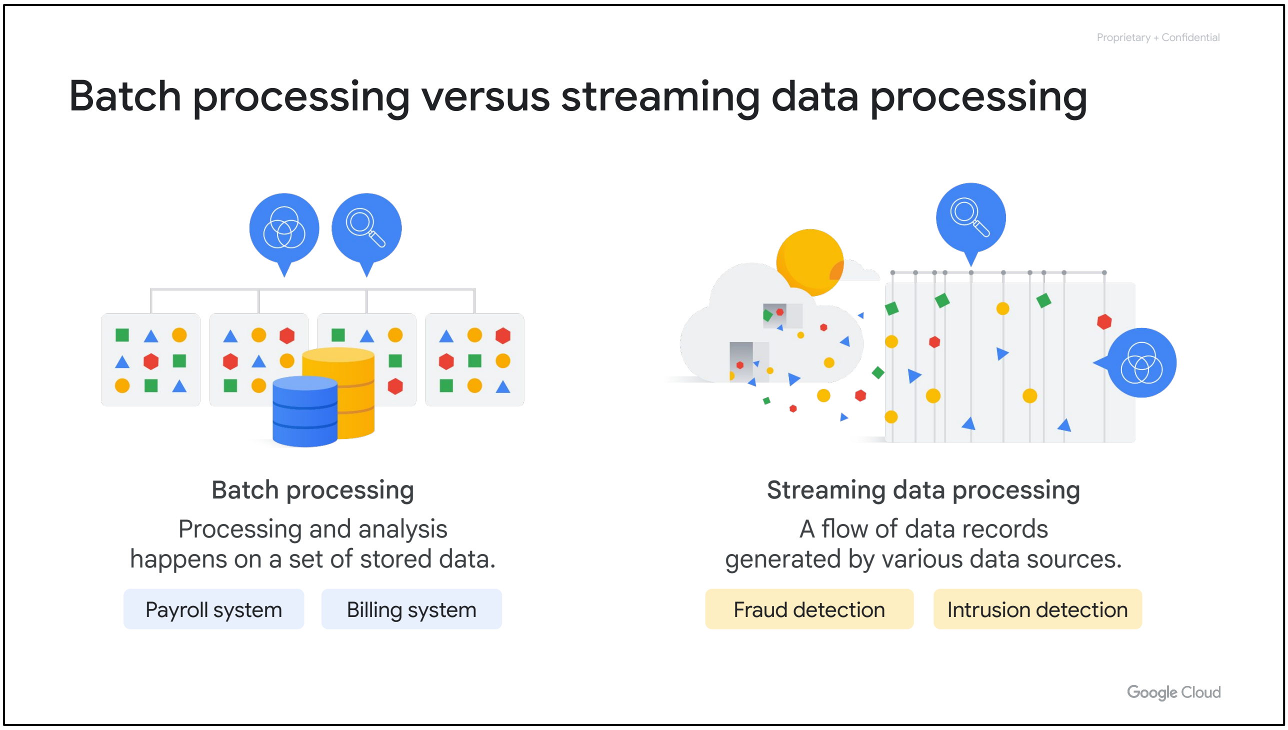
First, you configure the environment.

Next you download lab assets.

You then configure and execute the Spark code.

Finally, you view the data in BigQuery.

reference: https://www.cloudskillsboost.google/authoring/labs/31670



Proprietary + Confidential

Batch processing versus streaming data processing

Batch processing

Processing and analysis
happens on a set of stored data.

Payroll system

Billing system

Streaming data processing

A flow of data records
generated by various data sources.

Fraud detection

Intrusion detection

Google Cloud

Batch processing involves analyzing a fixed set of stored data, suitable for tasks like payroll or billing systems.

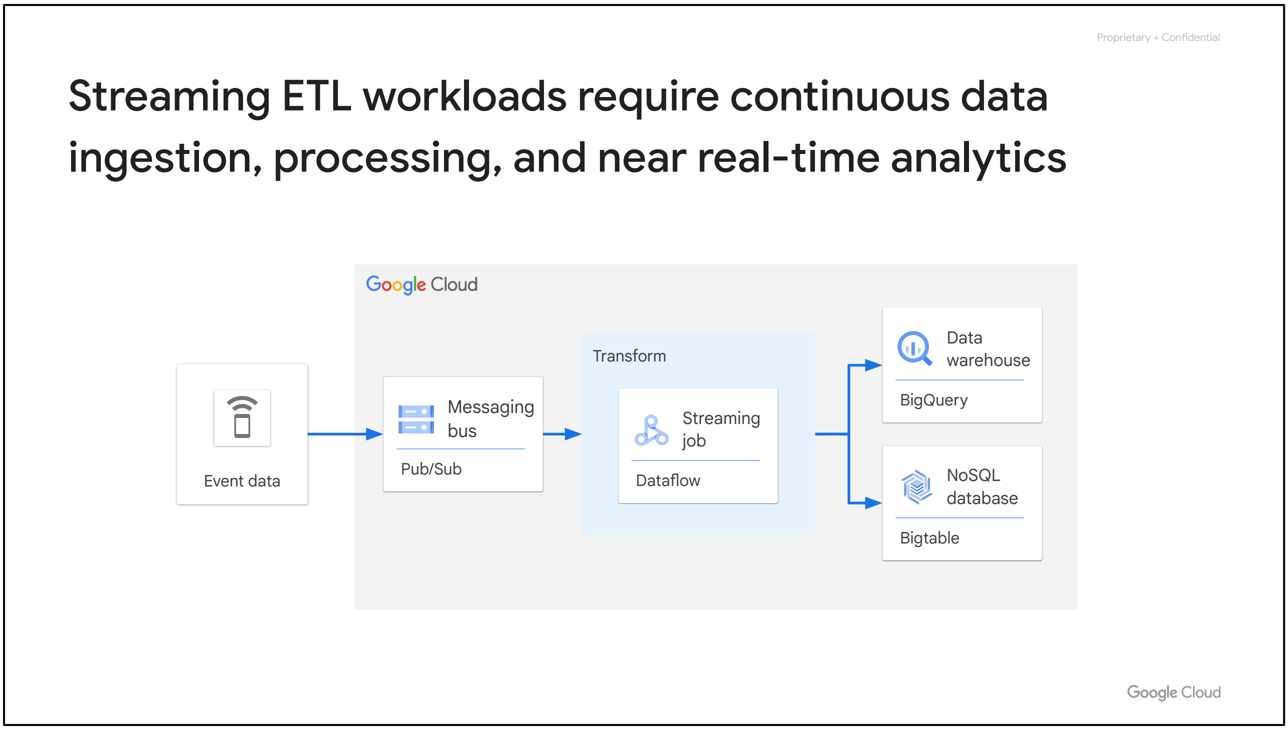
On the other hand, streaming data processing handles a continuous flow of data from various sources, making it ideal for real-time applications like fraud or intrusion detection.

Instructor notes:

- **Batch processing** is when the processing and analysis happens on a set of stored data. An example is payroll and billing systems that have to be processed on either a weekly or monthly basis.
- **Streaming data** is a flow of data records generated by various data sources. The processing of streaming data happens as the data flows through a system. This results in the analysis and reporting of events as they happen. An example would be fraud detection or intrusion detection.

Streaming data processing means that the data is analyzed in near-real time and that actions will be taken on the data as quickly as possible.

Modern data processing has progressed from legacy batch processing of data toward working with real-time data streams. An example of this is streaming music and movies. No longer is it necessary to download an entire movie or album to a local device. **Data streams** are a **key** part in the world of big data.



Proprietary + Confidential

Streaming ETL workloads require continuous data ingestion, processing, and near real-time analytics

Google Cloud

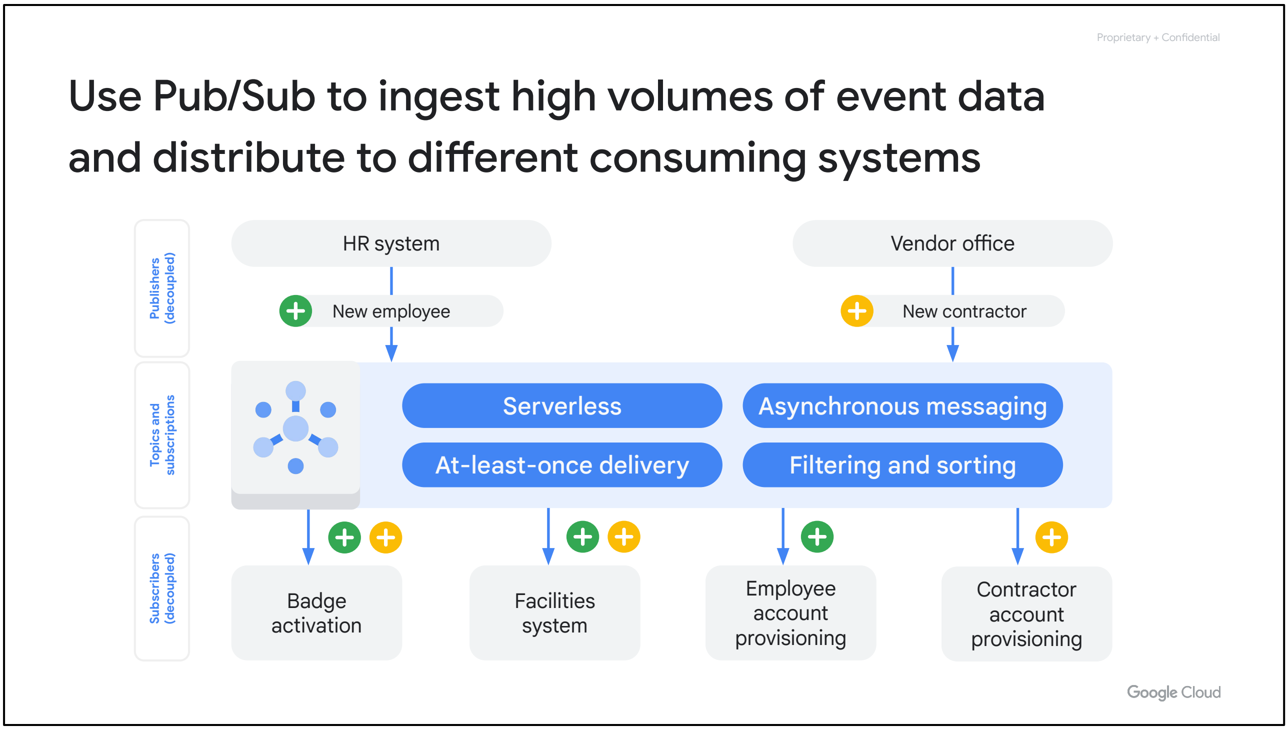
Streaming ETL workloads on Google Cloud involve the continuous ingestion of event data, often through Pub/Sub.

This data is often processed in real-time using Dataflow, allowing for transformations and enrichment.

Finally, the processed data is loaded into various destinations like BigQuery for analytics, enabling near real-time insights, or Bigtable for NoSQL storage.

Instructor notes:

- Mention that common streaming data sinks are BigQuery and Bigtable, both support streaming ingest
- BigQuery has second latency, enabling near-real time analytics
- Bigtable has milliseconds latency, enabling real time analytics



Use Pub/Sub to ingest high volumes of event data and distribute to different consuming systems

Publishers (decoupled)

HR system

New employee

Vendor office

New contractor

Topics and subscriptions

Serverless

Asynchronous messaging

At-least-once delivery

Filtering and sorting

Subscribers (decoupled)

Badge activation

Facilities system

Employee account provisioning

Contractor account provisioning

Pub/Sub can efficiently manage high volumes of event data.

Pub/Sub acts as a central hub, receiving events like "New employee" or "New contractor" from various sources.

Pub/Sub then distributes these events to relevant systems like badge activation, facilities, and account provisioning, ensuring reliable delivery and enabling decoupled, asynchronous communication between systems.

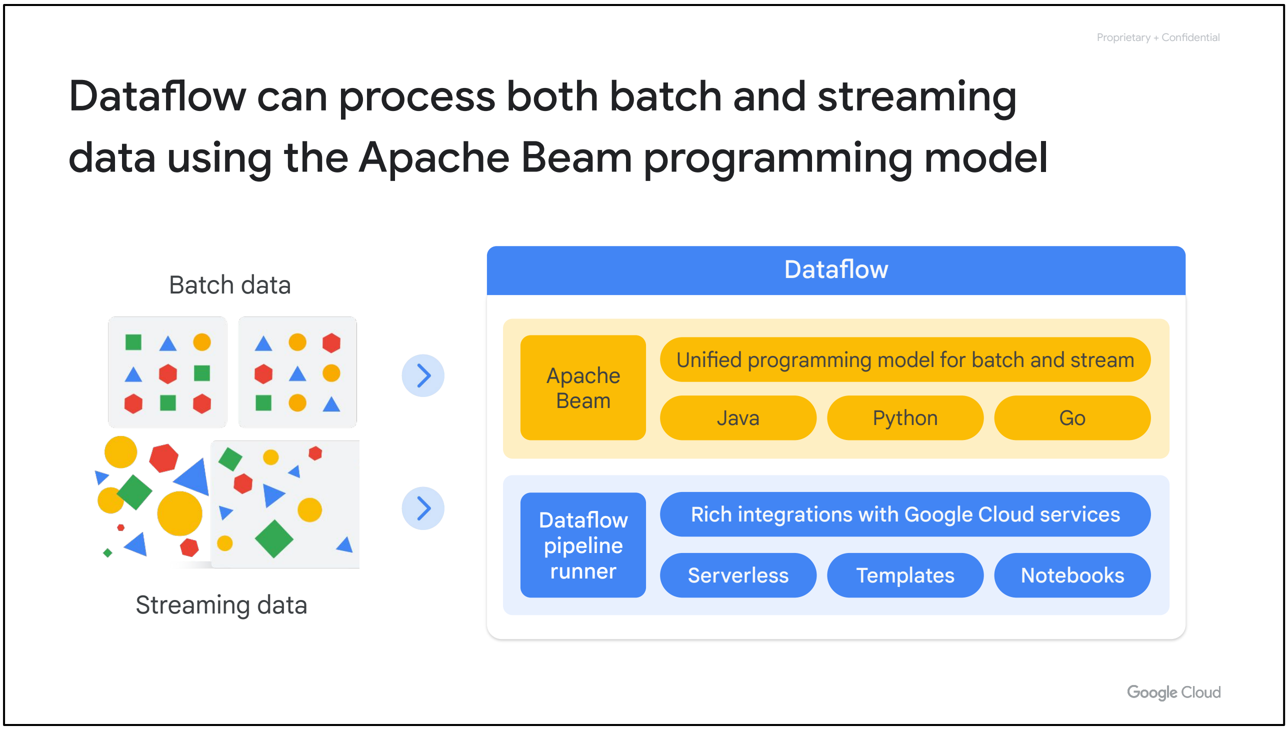
Instructor notes:

Figure shows an example of how different systems would consume messages from two different publishers (HR system, Vendor office > both with new hires).

Main takeaways:

- Pub/Sub uses publish/subscribe with topics to store the messages and subscriptions to deliver messages to consuming applications
- messages can all be delivered to the consuming application, or just specific ones
  - badge activation and facilities system have two subscriptions (one for the HR system topic, one for the Vendor office topic)

    - employee account provisioning and contractor account provisioning have subscriptions for just one of them
- publishing and subscribing applications are decoupled



Proprietary + Confidential

Dataflow can process both batch and streaming data using the Apache Beam programming model

Batch data

Streaming data

Dataflow

Apache Beam

Unified programming model for batch and stream

Java

Python

Go

Dataflow pipeline runner

Rich integrations with Google Cloud services

Serverless

Templates

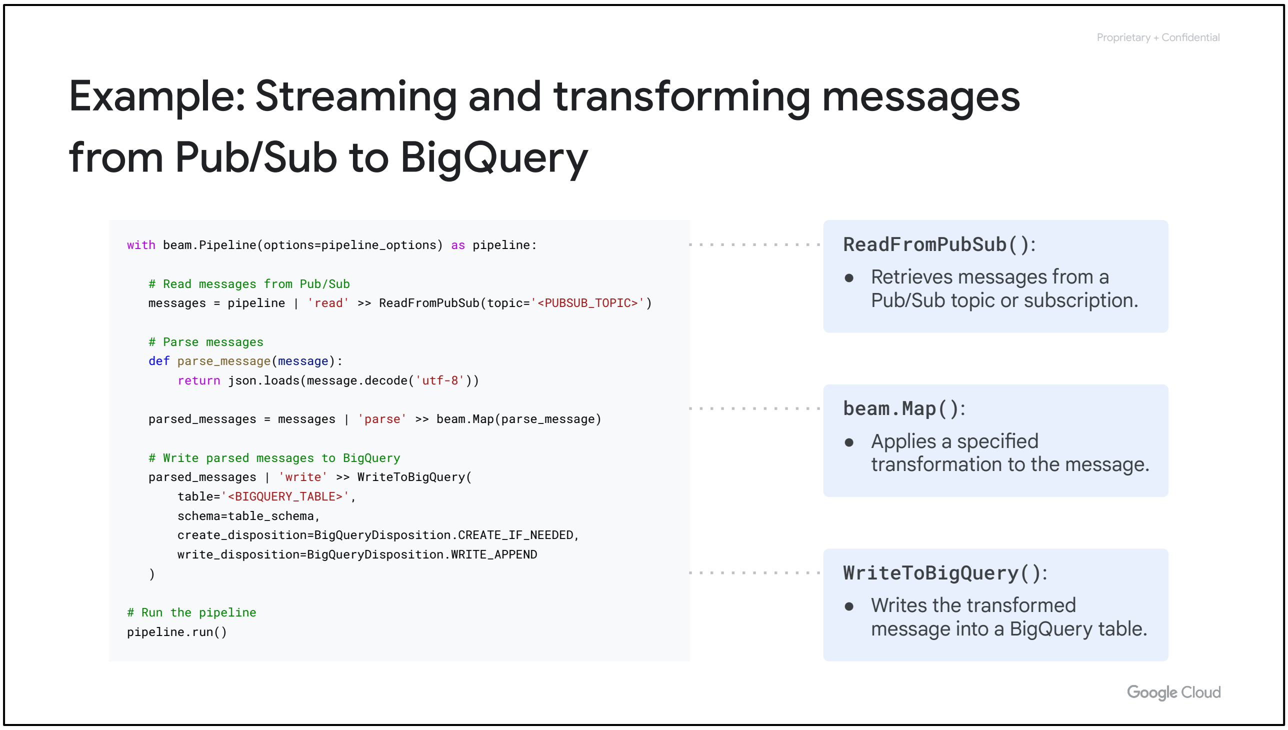
Notebooks

Google Cloud

Dataflow leverages the Apache Beam programming framework to efficiently process both batch and streaming data.

This unified approach simplifies development, allowing you to use languages like Java, Python, or Go.

Dataflow seamlessly integrates with other Google Cloud services and offers features like a pipeline runner, serverless execution, templates, and notebooks for a streamlined experience.



Proprietary + Confidential

Example: Streaming and transforming messages from Pub/Sub to BigQuery

```
with beam.Pipeline(options=pipeline_options) as pipeline:

    # Read messages from Pub/Sub
    messages = pipeline | 'read' >> ReadFromPubSub(topic='<PUBSUB_TOPIC>')

    # Parse messages
    def parse_message(message):
        return json.loads(message.decode('utf-8'))

    parsed_messages = messages | 'parse' >> beam.Map(parse_message)

    # Write parsed messages to BigQuery
    parsed_messages | 'write' >> WriteToBigQuery(
        table='<BIGQUERY_TABLE>',
        schema=table_schema,
        create_disposition=BigQueryDisposition.CREATE_IF_NEEDED,
        write_disposition=BigQueryDisposition.WRITE_APPEND
    )

# Run the pipeline
pipeline.run()
```

`ReadFromPubSub()`:

- Retrieves messages from a Pub/Sub topic or subscription.

`beam.Map()`:

- Applies a specified transformation to the message.

`WriteToBigQuery()`:

- Writes the transformed message into a BigQuery table.

Google Cloud

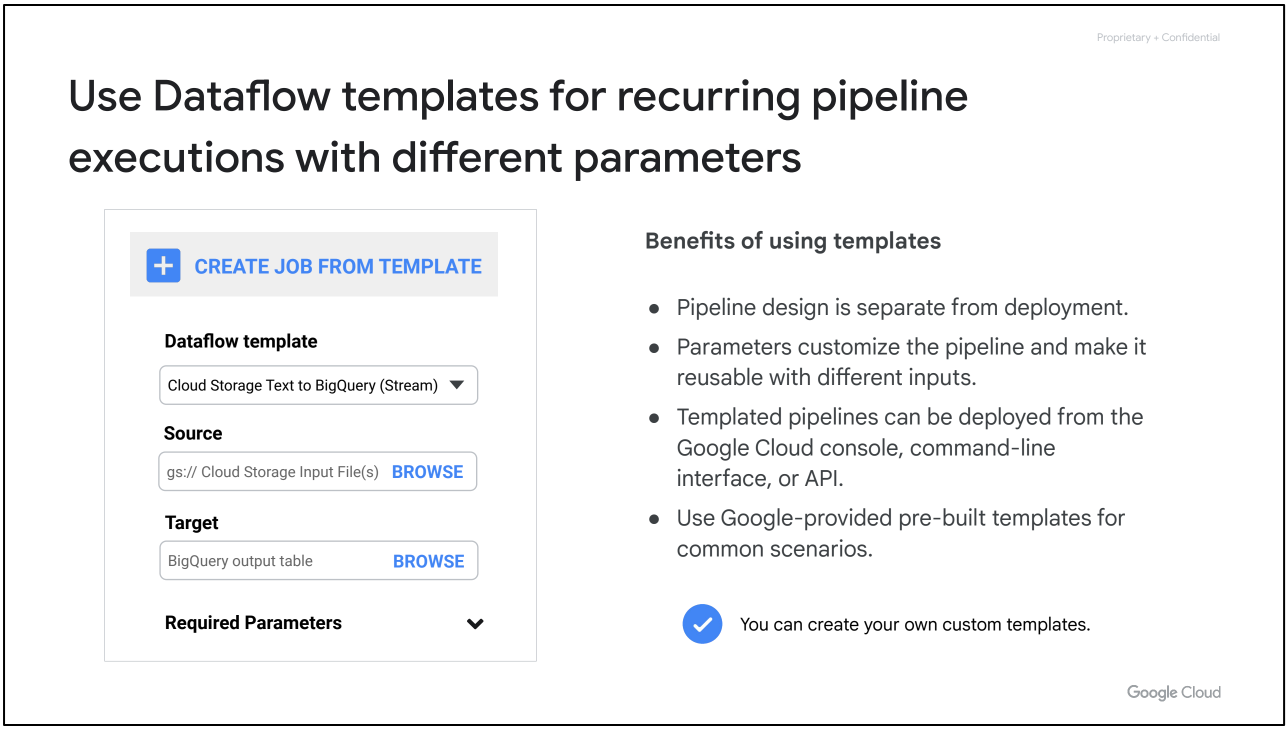
This code example demonstrates how to use Apache Beam to stream messages from Pub/Sub, transform them using a parsing function, and then write the results into BigQuery.

The “`ReadFromPubSub`” function retrieves messages, “`Beam.Map()`” applies the parsing transformation, and “`WriteToBigQuery`” loads the transformed data into a specified BigQuery table, creating the table if necessary and appending new data to it.

Instructor notes:

(additional info) Difference between reading from Pub/Sub Topic vs. Subscription:

- Topic:
    - create temporary new subscription, which is unique per pipeline
    - retrieves just new data that comes in once subscription was created
- Subscription:
    - uses existing subscription
    - retrieves also past data the subscription already had collected



Proprietary + Confidential

Use Dataflow templates for recurring pipeline executions with different parameters

+ CREATE JOB FROM TEMPLATE

Dataflow template

Cloud Storage Text to BigQuery (Stream) ▼

Source

gs:// Cloud Storage Input File(s) **BROWSE**

Target

BigQuery output table **BROWSE**

Required Parameters ⌄

Benefits of using templates

- Pipeline design is separate from deployment.
- Parameters customize the pipeline and make it reusable with different inputs.
- Templated pipelines can be deployed from the Google Cloud console, command-line interface, or API.
- Use Google-provided pre-built templates for common scenarios.

✓ You can create your own custom templates.

Google Cloud

Dataflow templates allow you to create reusable pipelines for recurring tasks.

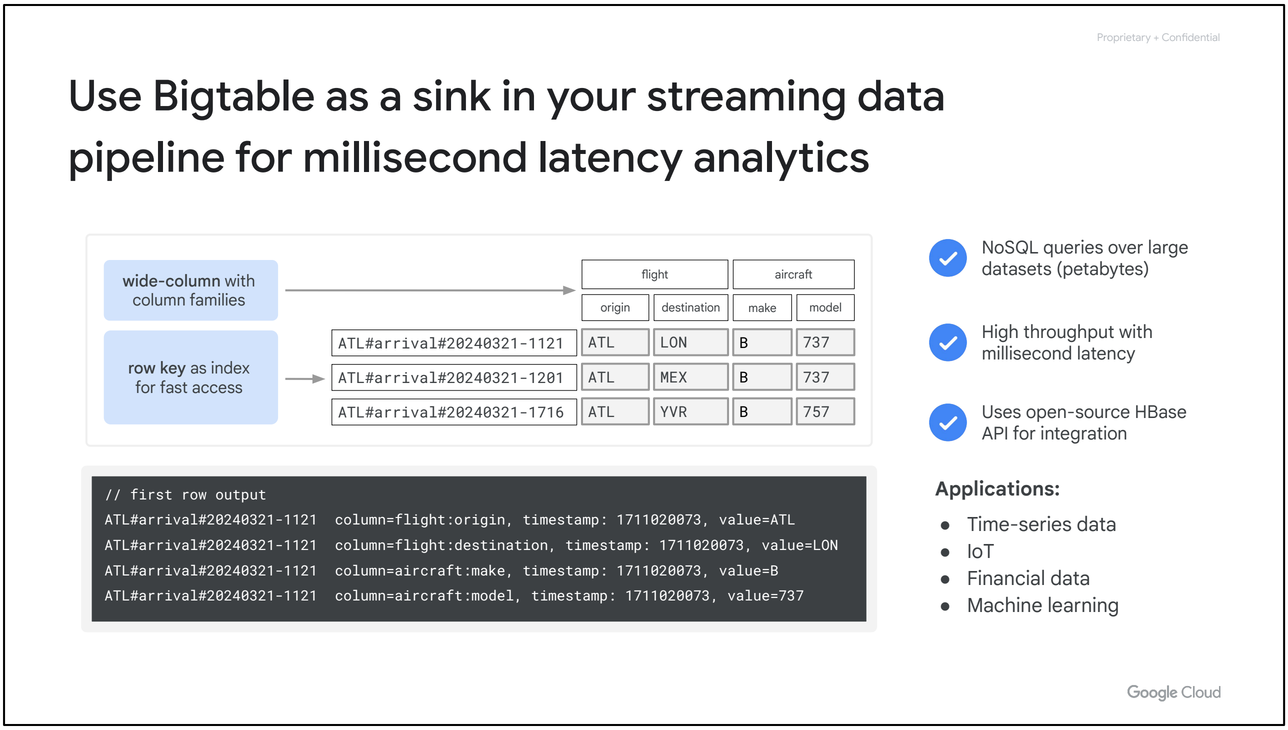
You can separate the pipeline design from its deployment, making it easier to manage and update.

By using parameters, you can customize the pipeline for different inputs, increasing its versatility.

These templated pipelines can be easily deployed through various methods, and Google provides pre-built templates for common scenarios.

References:

https://cloud.google.com/dataflow/docs/concepts/dataflow-templates
https://cloud.google.com/dataflow/docs/guides/templates/provided/text-to-bigquery-stream (used in lab)



Use Bigtable as a sink in your streaming data pipeline for millisecond latency analytics

wide-column with column families

row key as index for fast access

	flight		aircraft	
	origin	destination	make	model
ATL#arrival#20240321-1121	ATL	LON	B	737
ATL#arrival#20240321-1201	ATL	MEX	B	737
ATL#arrival#20240321-1716	ATL	YVR	B	757

```
// first row output
ATL#arrival#20240321-1121  column=flight:origin, timestamp: 1711020073, value=ATL
ATL#arrival#20240321-1121  column=flight:destination, timestamp: 1711020073, value=LON
ATL#arrival#20240321-1121  column=aircraft:make, timestamp: 1711020073, value=B
ATL#arrival#20240321-1121  column=aircraft:model, timestamp: 1711020073, value=737
```

✓ NoSQL queries over large datasets (petabytes)

✓ High throughput with millisecond latency

✓ Uses open-source HBase API for integration

Applications:

- Time-series data
- IoT
- Financial data
- Machine learning

Google Cloud

Bigtable is an excellent choice for handling streaming data pipelines that require millisecond-level latency analytics.

Bigtable utilizes a wide-column data model with column families, allowing for flexible schema design.

Row keys serve as efficient indexes for quick data access.

Bigtable's high throughput and low latency capabilities make it suitable for applications like time-series data, IoT, financial data, and machine learning, especially when dealing with large datasets.

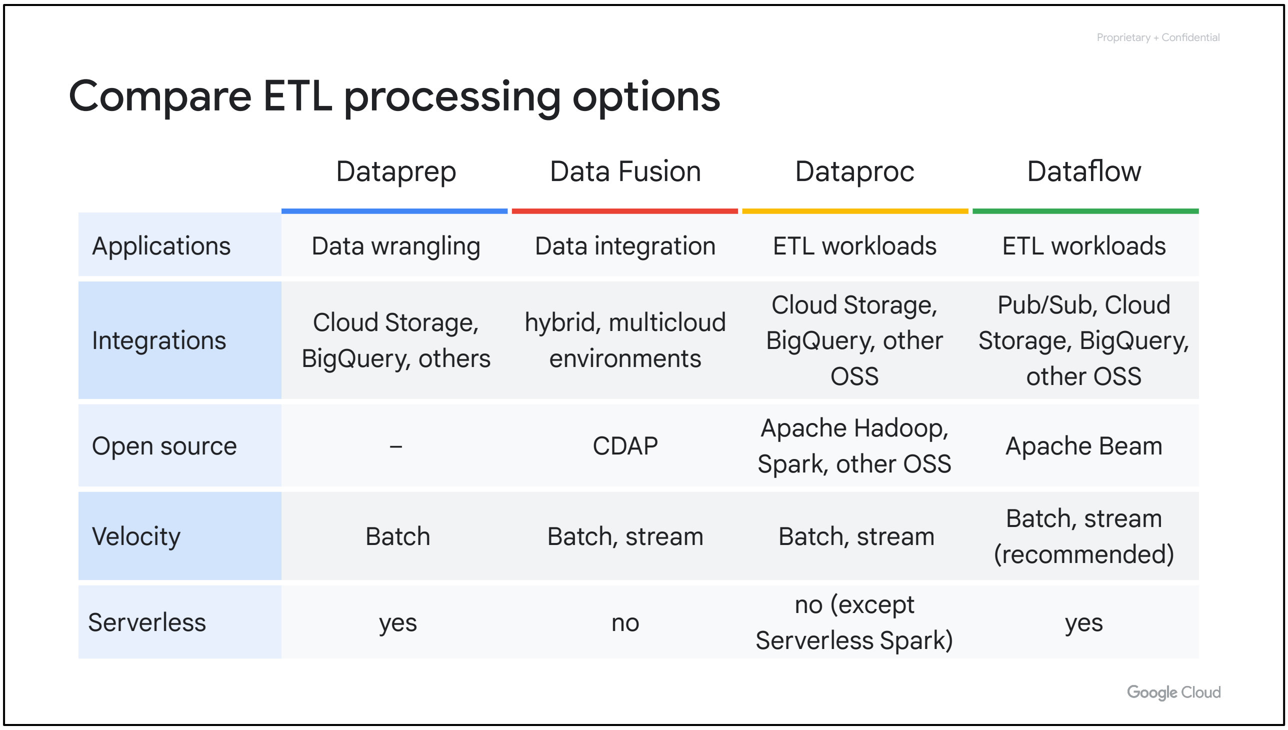
References:

Bigtable column and row limits:
https://cloud.google.com/bigtable/quotas#limits-data-size

Instructor notes:

- figure should give an example of how data is represented and stored in

- Bigtable, using 2 column families (flight and aircraft) with two columns each
- “first row output” shows how Bigtable would return the first row for a query



Proprietary + Confidential

Compare ETL processing options

	Dataprep	Data Fusion	Dataproc	Dataflow
Applications	Data wrangling	Data integration	ETL workloads	ETL workloads
Integrations	Cloud Storage, BigQuery, others	hybrid, multicloud environments	Cloud Storage, BigQuery, other OSS	Pub/Sub, Cloud Storage, BigQuery, other OSS
Open source	–	CDAP	Apache Hadoop, Spark, other OSS	Apache Beam
Velocity	Batch	Batch, stream	Batch, stream	Batch, stream (recommended)
Serverless	yes	no	no (except Serverless Spark)	yes

Google Cloud

In summary, Google Cloud provides various services for ETL processing.

Dataprep is ideal for data wrangling tasks and offers a serverless option.

Data Fusion excels at data integration, particularly in hybrid and multi-cloud environments, utilizing the open-source CDAP framework.

Dataproc handles ETL workloads with support for Hadoop, Spark, and other open-source tools, with Serverless Spark as a serverless option.

Lastly, Dataflow, built on Apache Beam, is recommended for both batch and streaming ETL workloads and provides a serverless architecture.

References:

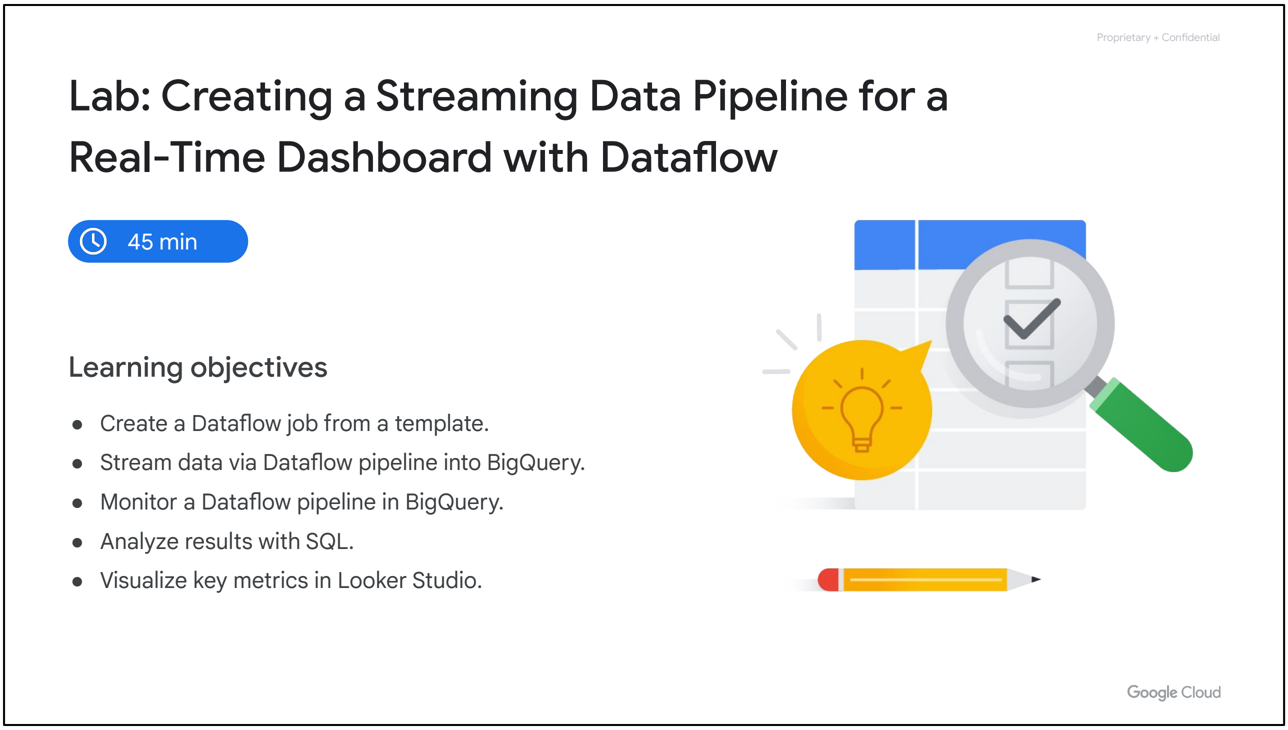
Dataprep connection types:
https://help.alteryx.com/Dataprep/en/platform/connections/connection-types.html

Instructor notes:

Velocity:

- Data Fusion running on Dataproc and Dataproc itself can do streaming, but we

- recommend Dataflow for streaming, since it has more capabilities like windowing and handling of late data



Proprietary + Confidential

Lab: Creating a Streaming Data Pipeline for a Real-Time Dashboard with Dataflow

45 min

Learning objectives

- Create a Dataflow job from a template.
- Stream data via Dataflow pipeline into BigQuery.
- Monitor a Dataflow pipeline in BigQuery.
- Analyze results with SQL.
- Visualize key metrics in Looker Studio.

Google Cloud

In this lab, you create a streaming data pipeline for a real-time dashboard with Dataflow.

You create a Dataflow job from a template.

You then monitor a pipeline loading data into BigQuery.

After that, you examine the data loaded using SQL.

Finally, you visualize key metrics using Looker Studio.

reference: https://www.cloudskillsboost.google/catalog_lab/1796

Google Cloud