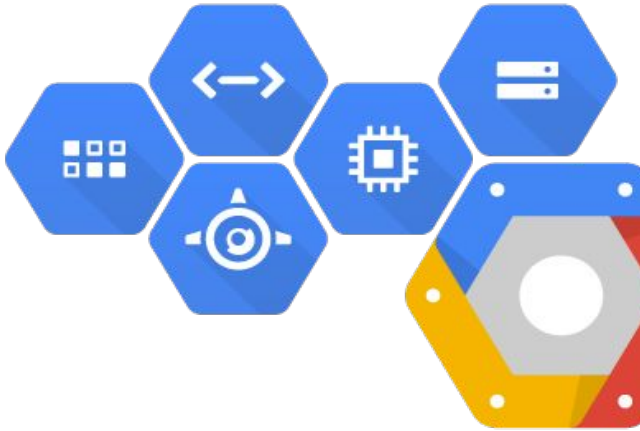
Google Cloud

Google Cloud Analytics 2-Day Training - Big Data

May 2018



Objectives

This 2-day training is designed to provide knowledge to deliver GCP's Big Data and Machine Learning capabilities. Attendees will leave the training with the following:

- Knowledge of how to deliver GCP Big Data solutions for our clients
- Deeper understanding of GCP's Big Data services and how they integrate with each other
- Knowledge of how GCP enables the Big Data Reference Architecture
- Understanding of technical design considerations for GCP's data services
- Understanding of how to enable a data pipeline on GCP

It is assumed that the attendees have GCP foundational knowledge for Big Data and Machine Learning

Agenda - Day 1 Big Data

Sections	Discussion Topics	Time
Setting the Stage	Why are we learning about the GCP platform? What are our business drivers for GCP? Demo of a GCP enabled solution	**8:00 - 9:00 AM**
Data Ingestion and Storage in GCP	What are the architectural features of GCP's data ingestion and storage utilities and how do they influence design decisions? **Break** What are the technical characteristics of GCP's data storage options and use cases of each?	**9:00 - 11:00 PM**
Transforming and Processing Data in GCP	How does GCP enable a no-ops data processing pipeline? What are the different data processing architectures in GCP and the benefits of each?	**11:00 - 12:00 PM**
Lunch		

Agenda - Day 1 Big Data *(cont…)*

Sections	Discussion Topics	Time
Transforming and Processing Data in GCP	How does GCP enable a no-ops data processing pipeline?	**1:00 - 2:00 PM**
	What are the different data processing architectures in GCP and the benefits of each?	
	What technical characteristics of the data processing tools GCP?	
Migrating Your Data Warehouse to Google BigQuery	What are key factors to consider when migrating the data warehouse to BigQuery?	**2:00 - 4:00 PM**
	How should data be architected for storage and querying in BigQuery?	
	Break	
	How does a team plan and prioritize workloads for migration into BigQuery?	
	What are post-migration features to enable?	
Designing Your Conceptual Architecture	How do GCP's Big Data offerings map to the big data reference architecture?	**4:00 - 5:00 PM**
	What are pathways to move to GCP?	
	What are key factors to consider when designing your future state architecture?	

Setting the Stage

Establishing Data as Competitive Currency

In today's landscape, companies are seeking new opportunities hidden within untapped data. To do so, they are increasingly experiencing a need for next-gen technology

1. **Untapped Data Within the Organization**
2. **Nontraditional Unstructured Data**
3. **Data in the Deep Web**

C&IP

Vision and speech enabled shopping assistants

Energy & Resources

Predictive maintenance for gas pipelines

FSI

Personalized finance and investment advisors

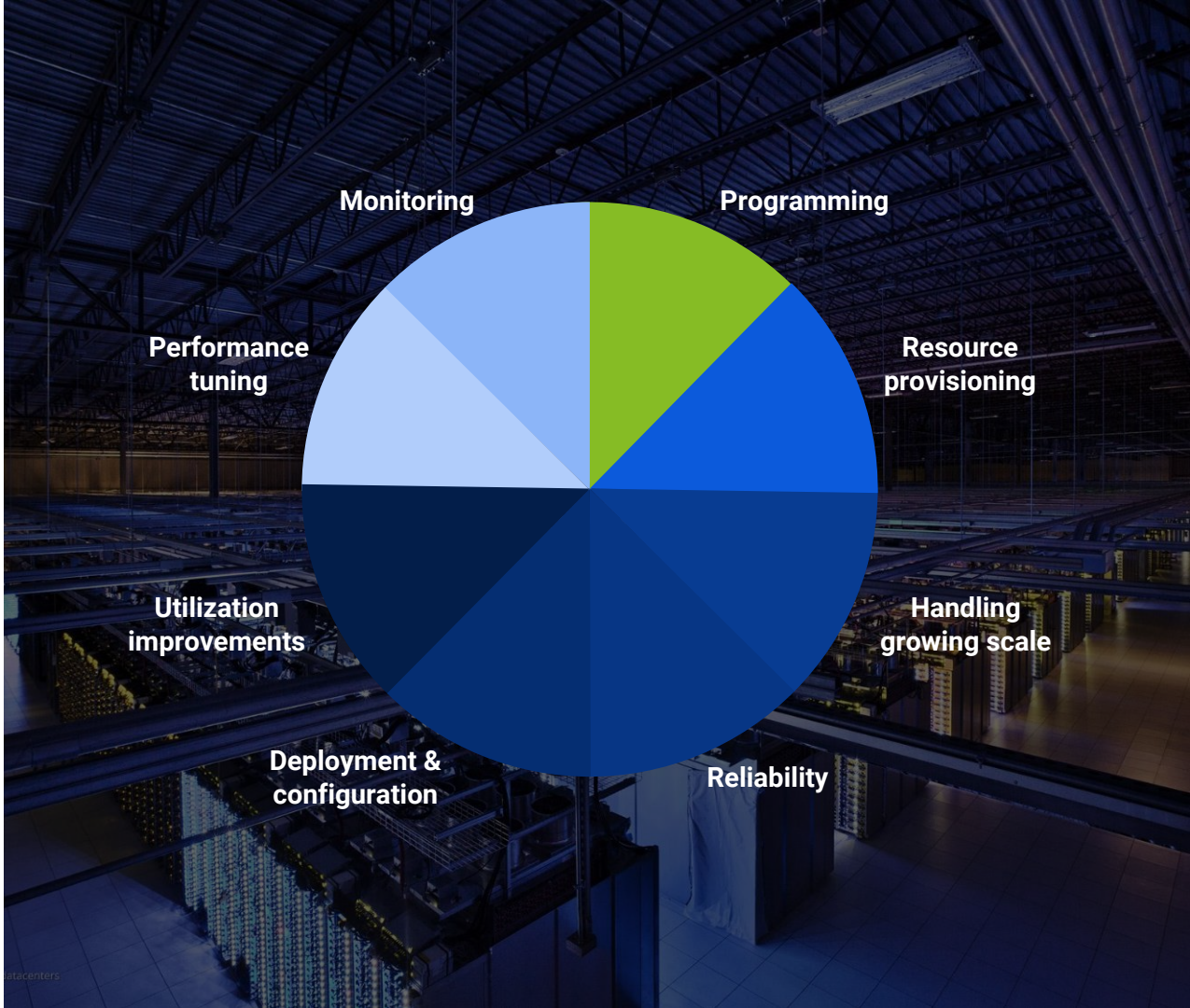
LSHC

AI enabled health plan recommenders

TMT

Virtual network engineers

Typical Big Data Jobs

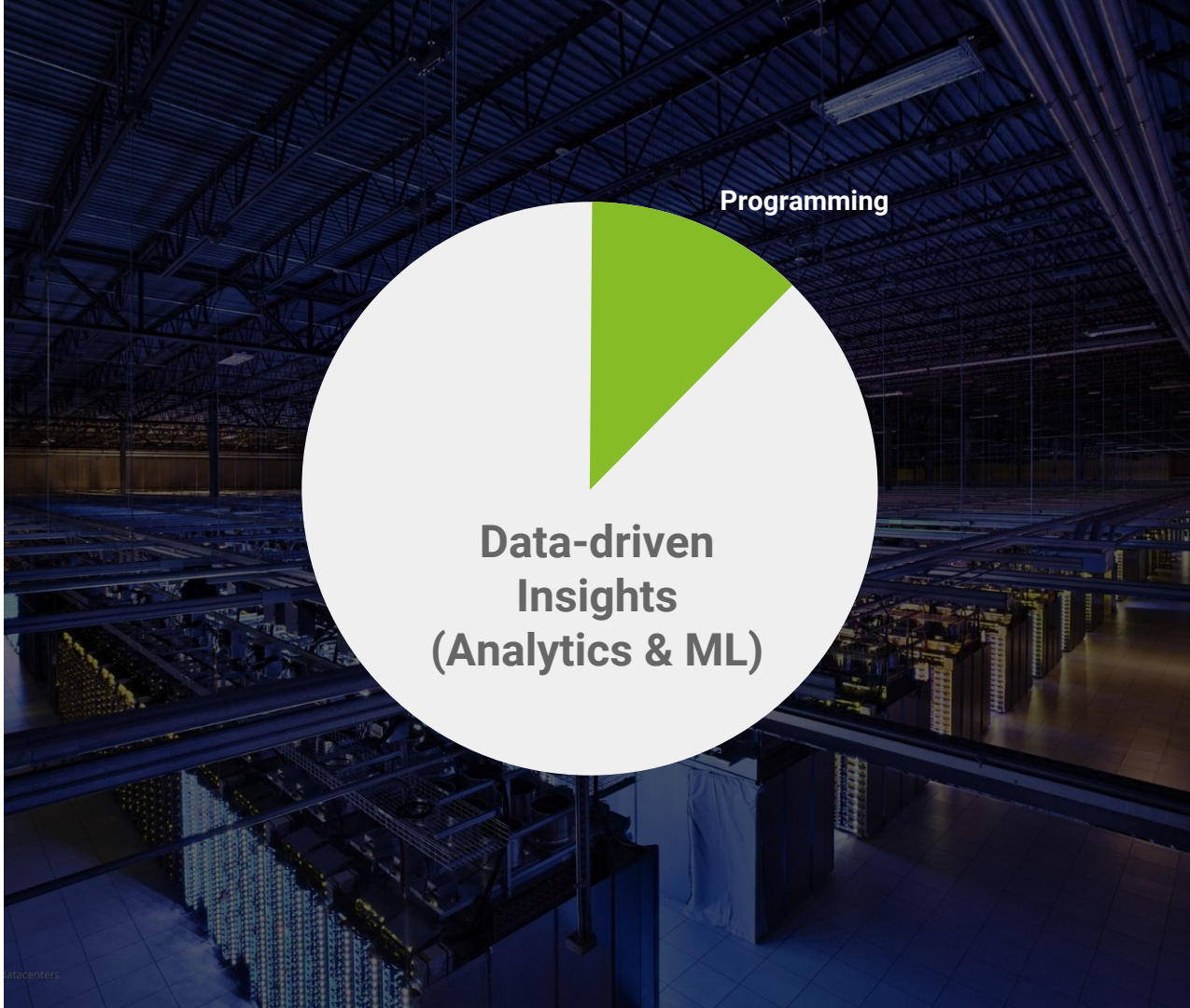


Big Data with Google

Focus on insights.

Not infrastructure.

From batch to real-time.



Big Data Services on GCP

Phases of the Data Lifecycle

Ingest

The first stage is to pull in the raw data, such as streaming data from devices, on-premises batch data, application logs, or mobile-app user events and analytics.

Store

After the data has been retrieved, it needs to be stored in a format that is durable and can be easily accessed.

Process & Analyze

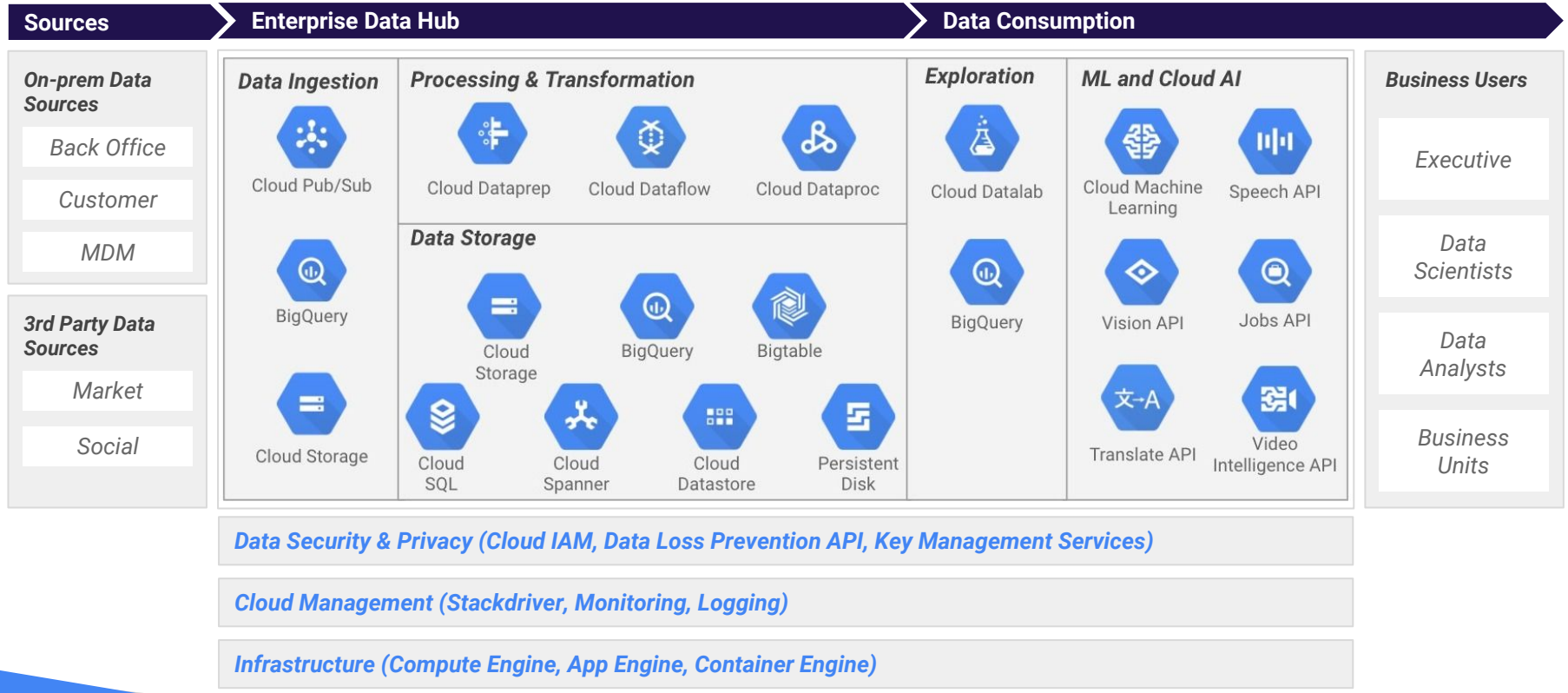
In this stage, the data is transformed from raw form into actionable information.

Explore & Visualize

The final stage is to convert the results of the analysis into a format that is easy to draw insights from and to share with colleagues and peers.

Big Data Reference Architecture on GCP

No-ops big data services on GCP



Google Cloud

A common configuration: acquire data

Events, metrics, etc.

Cloud Pub/Sub **(Data streaming service)**

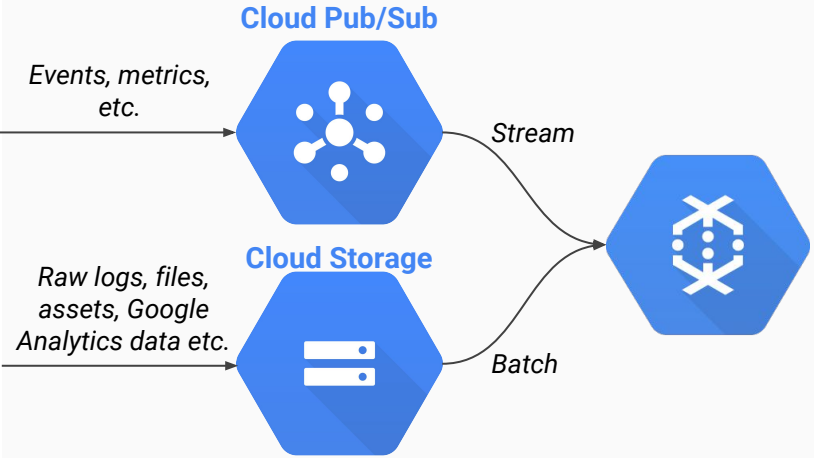
Reliable, many-to-many, asynchronous messaging

Raw logs, files, assets, Google Analytics data etc.

Cloud Storage **(BLOB storage → data lake)**

Powerful, simple and cost-effective object storage

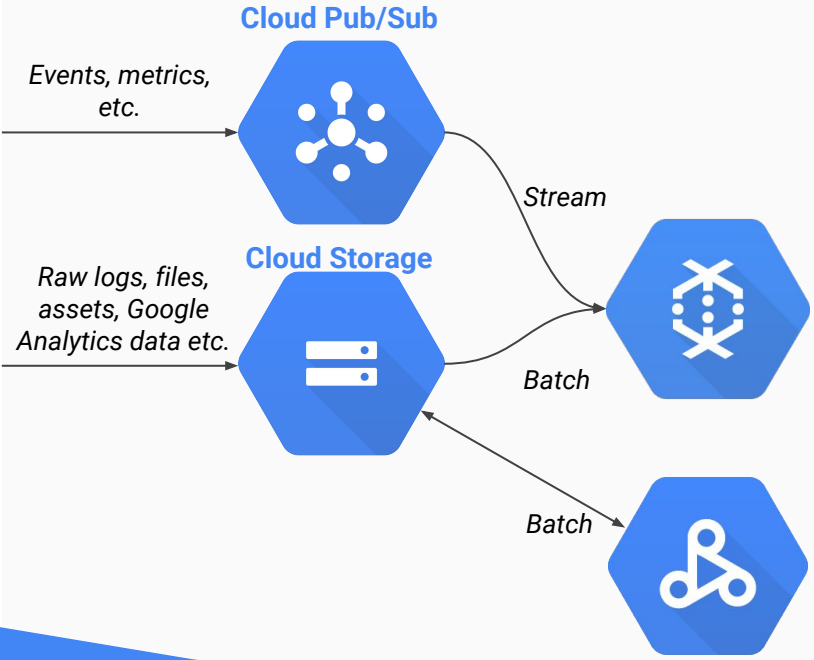
A common configuration: process and transform data



Cloud Dataflow **(ETL/ELT transformation)**

Data processing engine for batch and stream processing

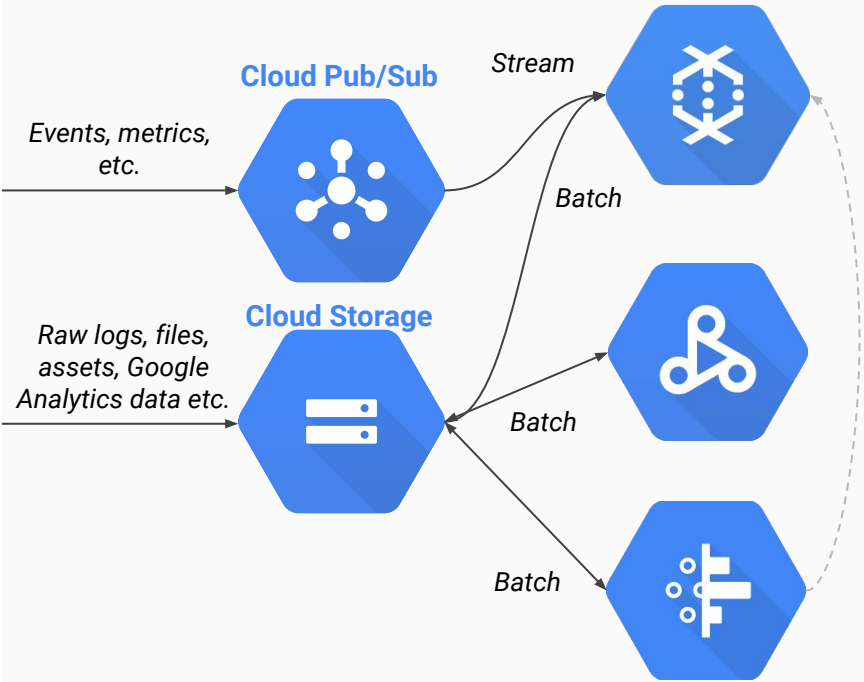
A common configuration: process and transform data



Cloud Dataflow (ETL/ELT pipeline)
Data processing engine for batch and stream processing

Cloud Dataproc (Access to open-source stack)
Managed Spark and Hadoop

A common configuration: process and transform

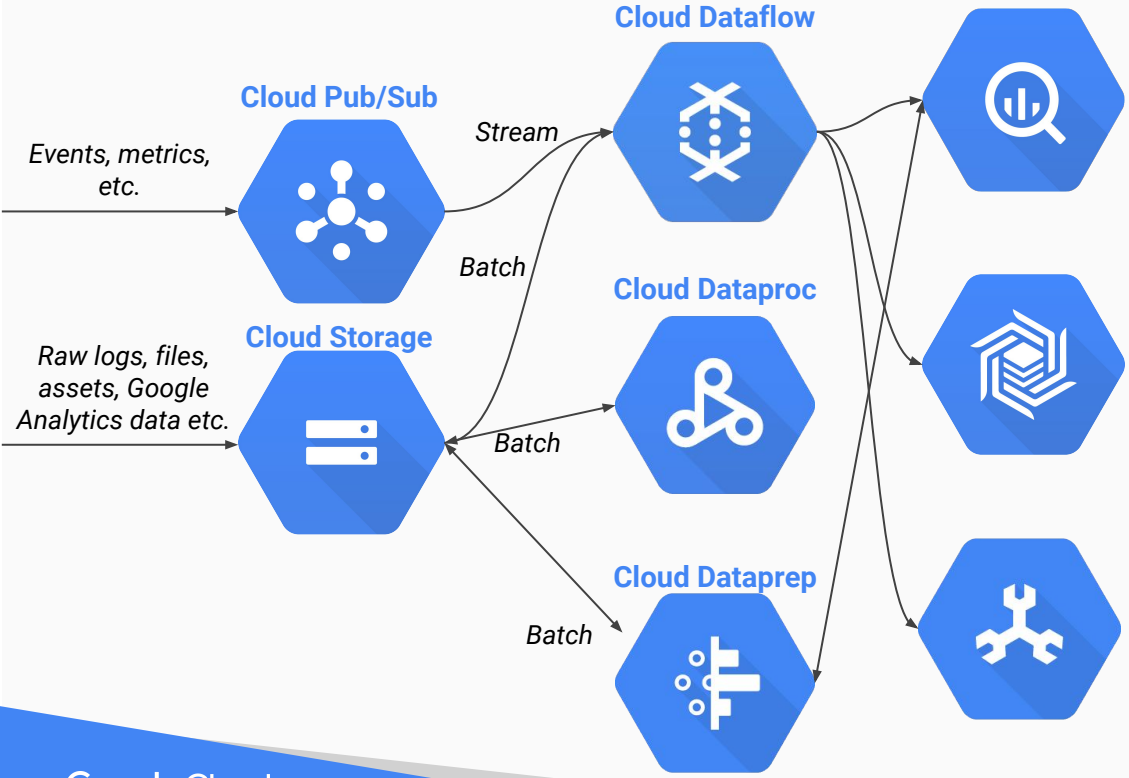


Cloud Dataflow (ETL/ELT pipeline)
Data processing engine for batch and stream processing

Cloud Dataproc (Open-source stack)
Managed Spark and Hadoop

Cloud Dataprep (UI based data preparation)
Data service for visually preparing data for analysis

A common configuration: analyze and store



BigQuery

Extremely fast and cheap on-demand analytics engine

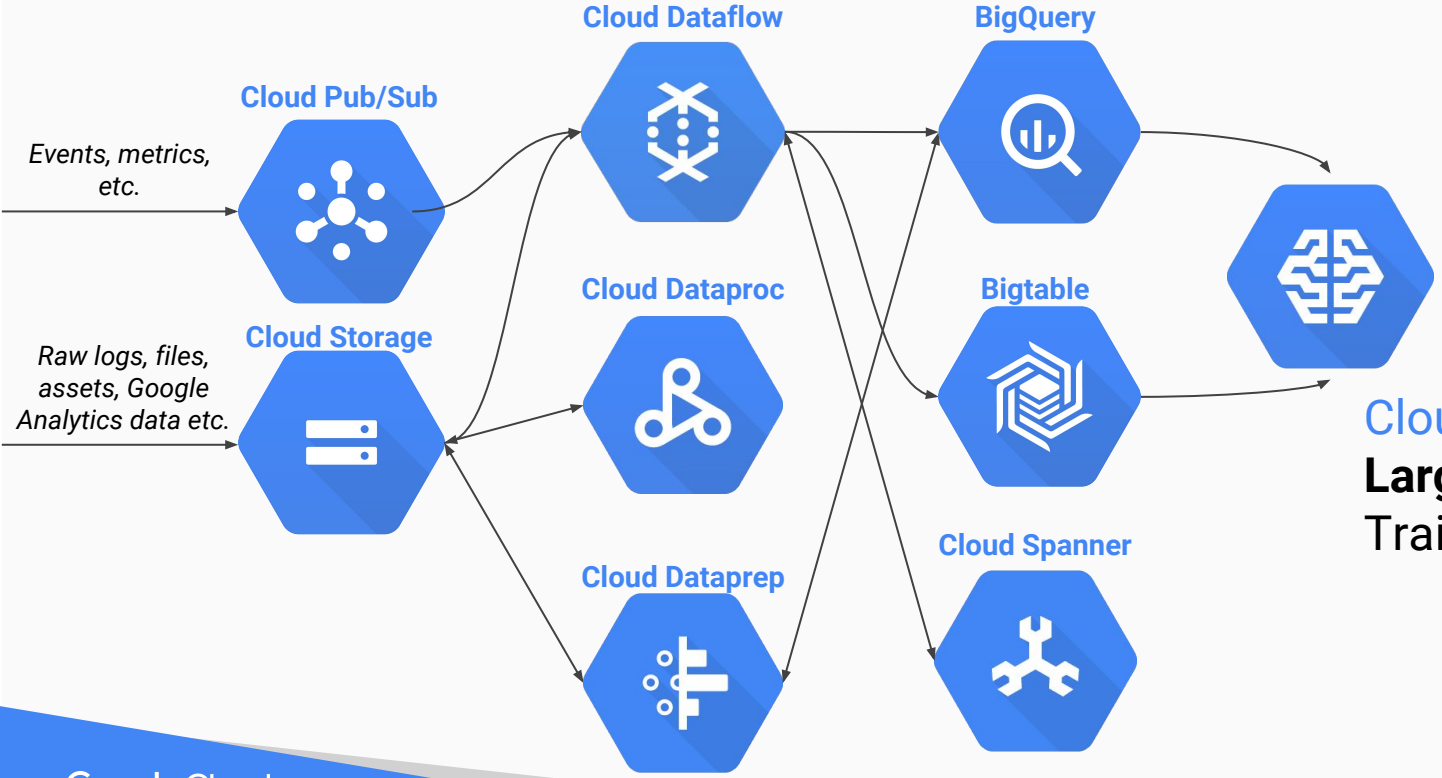
Bigtable

High performance NoSQL database for large workloads

Cloud Spanner

Horizontally scalable, consistent, relational database

A common configuration: learn and recommend

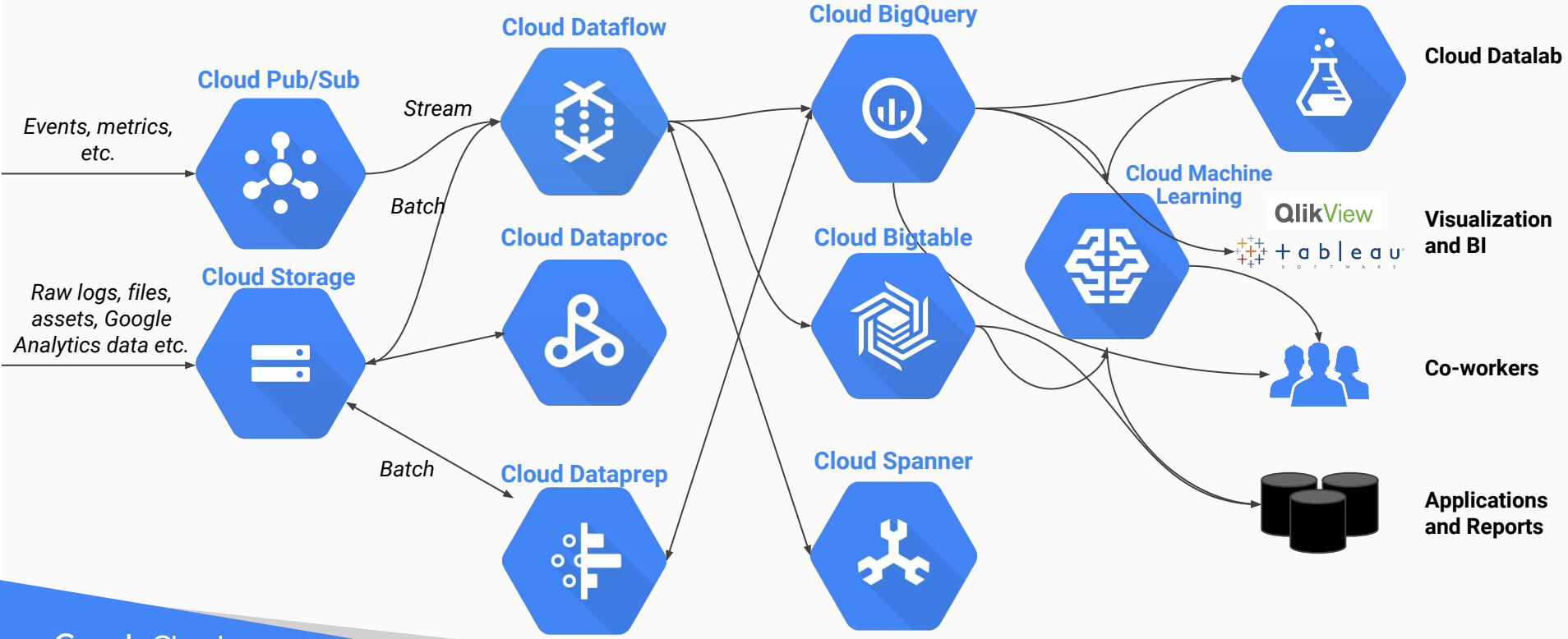


Cloud Machine Learning

Large scale

Train your own models

A common configuration: learn and recommend



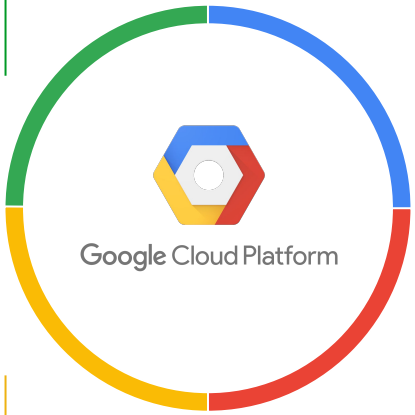
An End to End Solution

Client Example: ACME

ACME wants to develop an end to end marketing analytics solution on GCP with the following objectives:

Holistic view of customers enabled by integration of data in silos

Capability to study customers in real time and engage them in contextually relevant ways



Adoption of a customer data platform to enable deep dive analysis into customer behaviors

Integration with machine learning capabilities to predict customer demand and provide recommendations

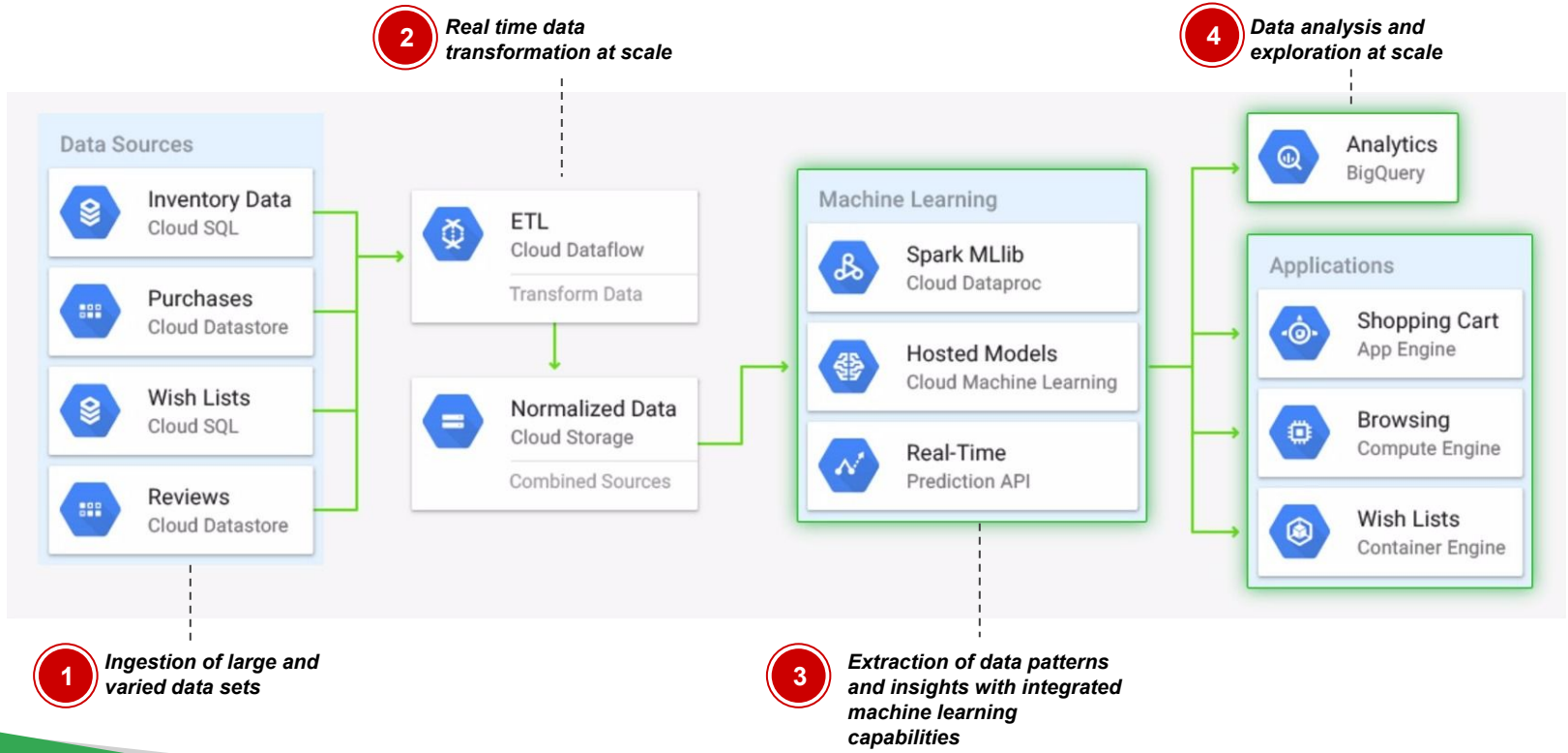
Activity - 10 minutes
Design a No-Ops Solution on GCP

Given ACME's objective, we need to design the target architecture for a solution that harnesses data. In order to do this, we need to consider the following:

1. What type of data sources does ACME need?
2. What are the phases of the data lifecycle for this use case
3. What GCP services does ACME need across the data lifecycle?
4. What type of workflows does ACME need to support (batch or streaming)?

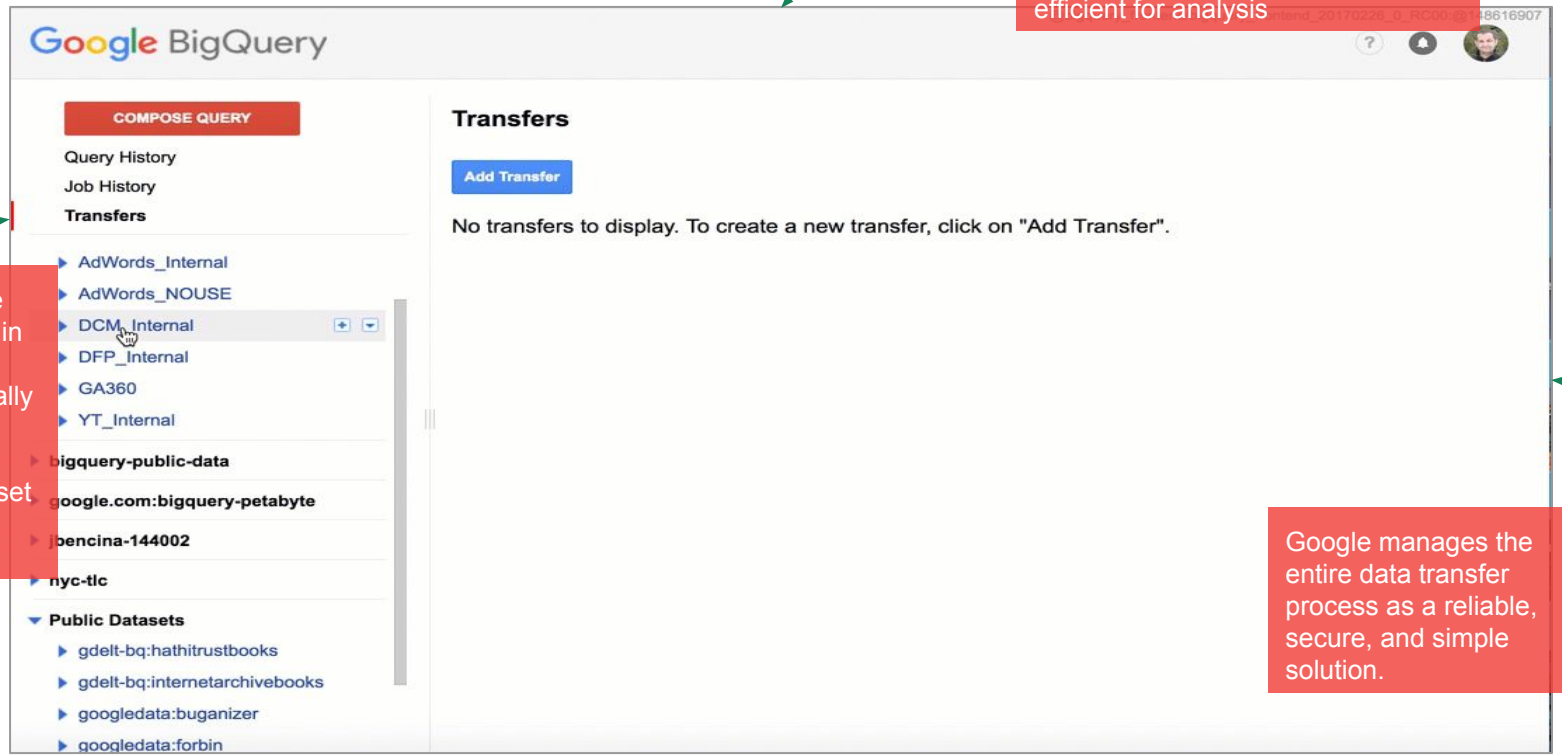
Leveraging a No-ops Architecture on GCP

A sample solution from ingestion of data sources to integration of insights with live processes



1 Ingesting Raw Data into BigQuery (using BQTS)

Data is organized in date-partitioned tables which is efficient for analysis

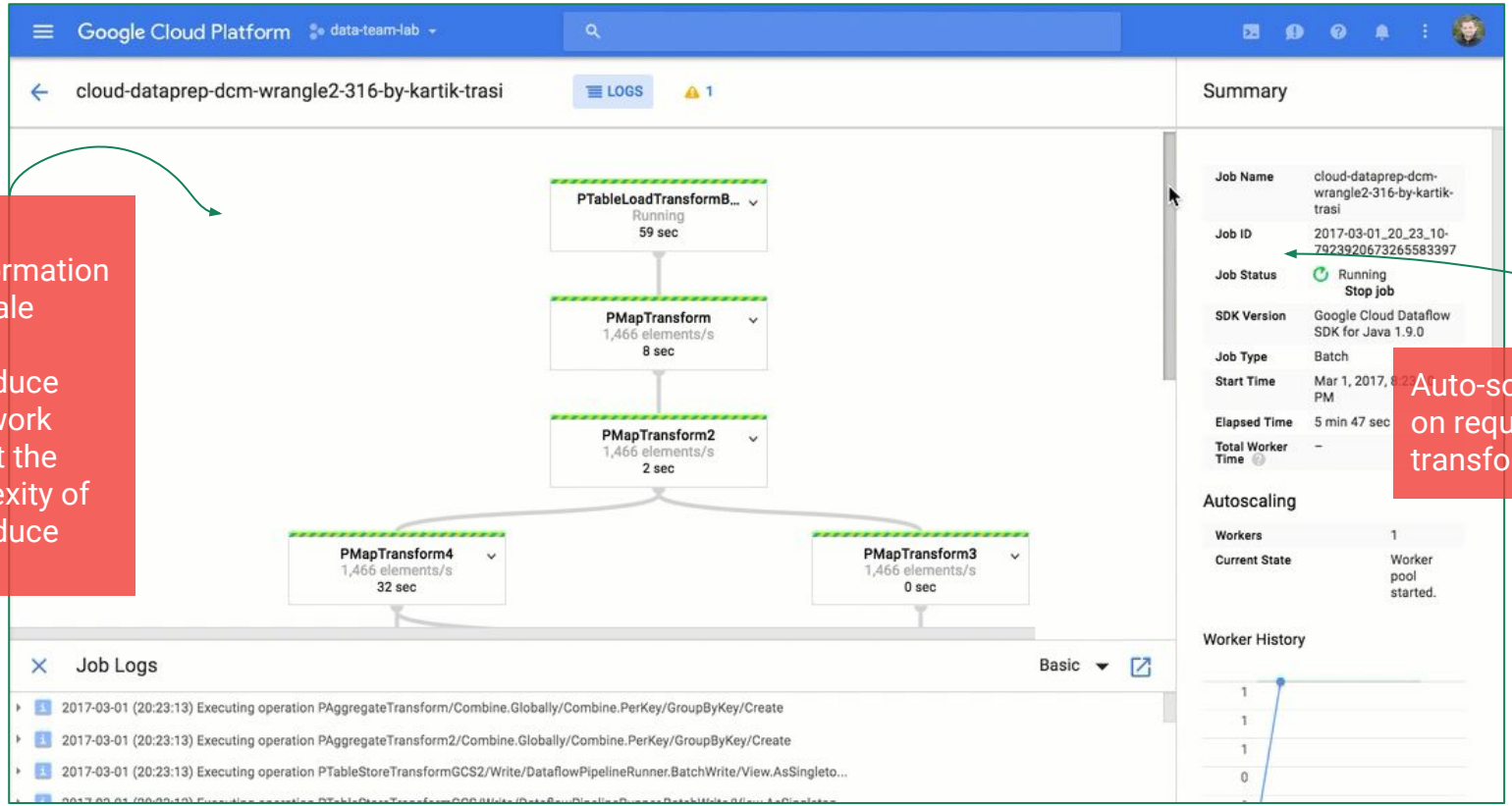


Tables are populated in Datasets automatically once data transfer service is set up

Google manages the entire data transfer process as a reliable, secure, and simple solution.

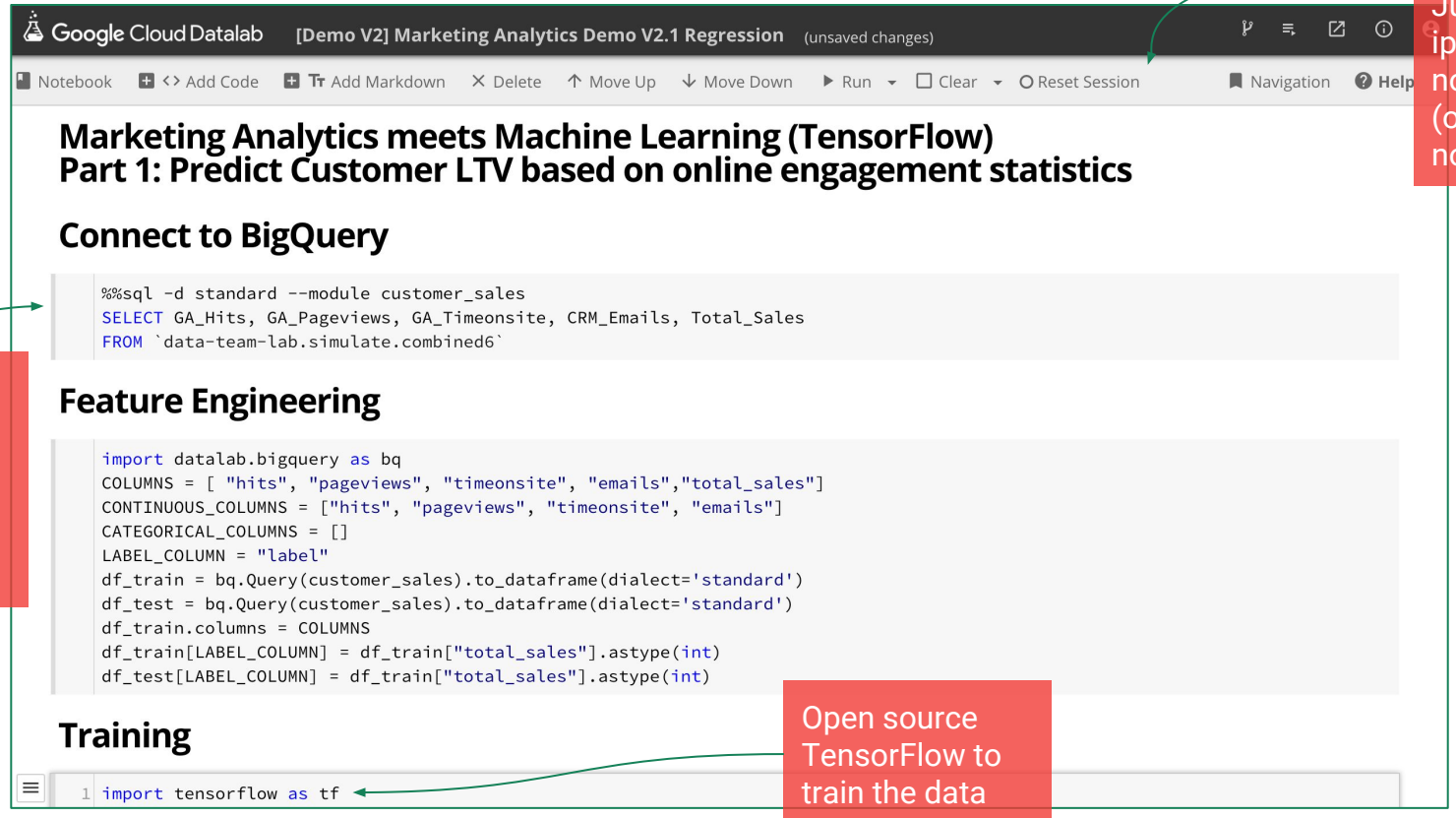
2 Transforming Data in a Dataflow pipeline

Deploy transformations at scale using mapreduce framework without the complexity of mapreduce



Auto-scales based on requirement to transform data

3 Using ML to predict likelihood of customer purchases

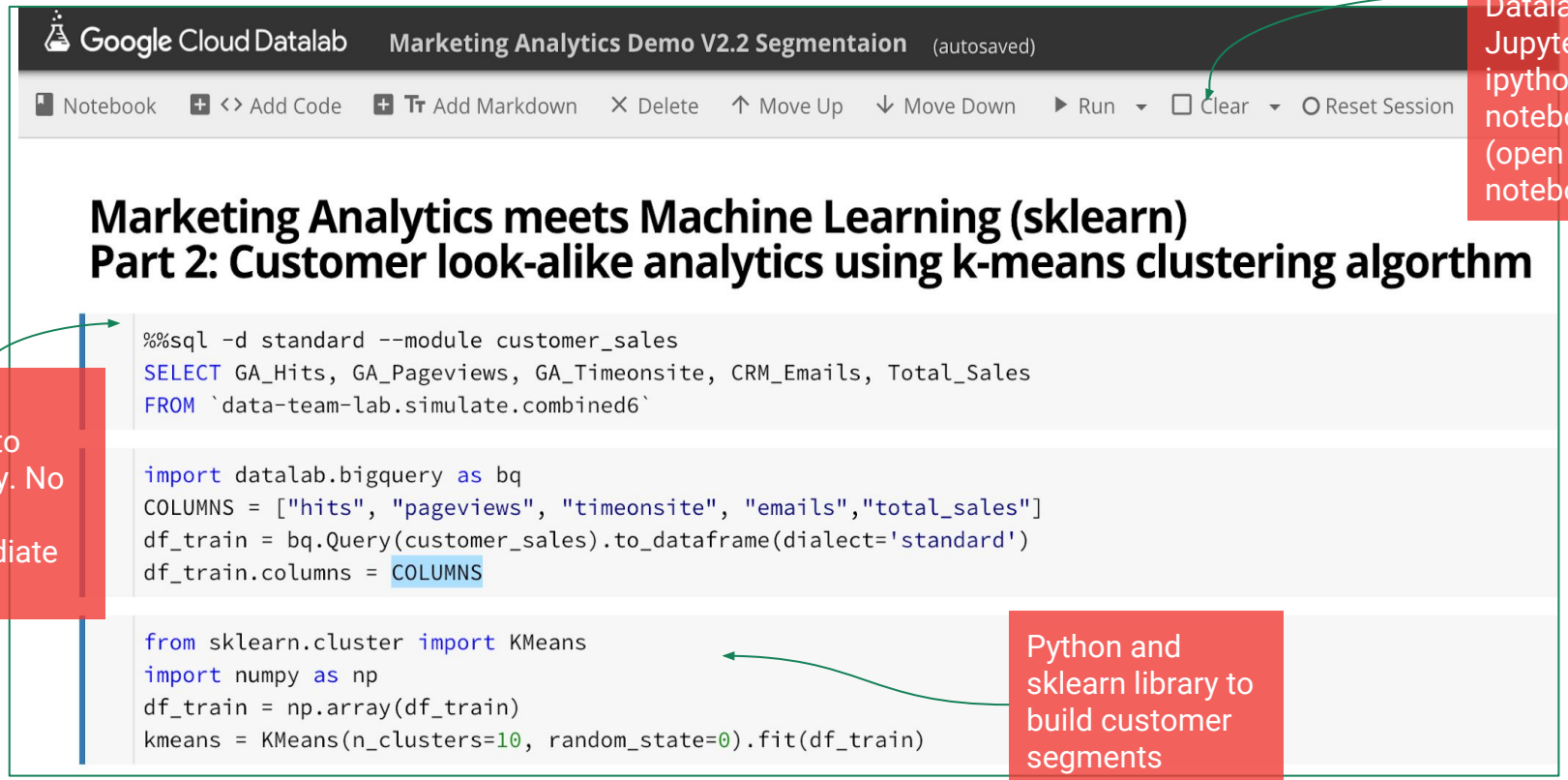


Datalab = Jupyter / ipython notebook (open source) notebook

Connect directly to BigQuery. No export / intermediate steps!

Open source TensorFlow to train the data

3 Customer Segmentation using ML



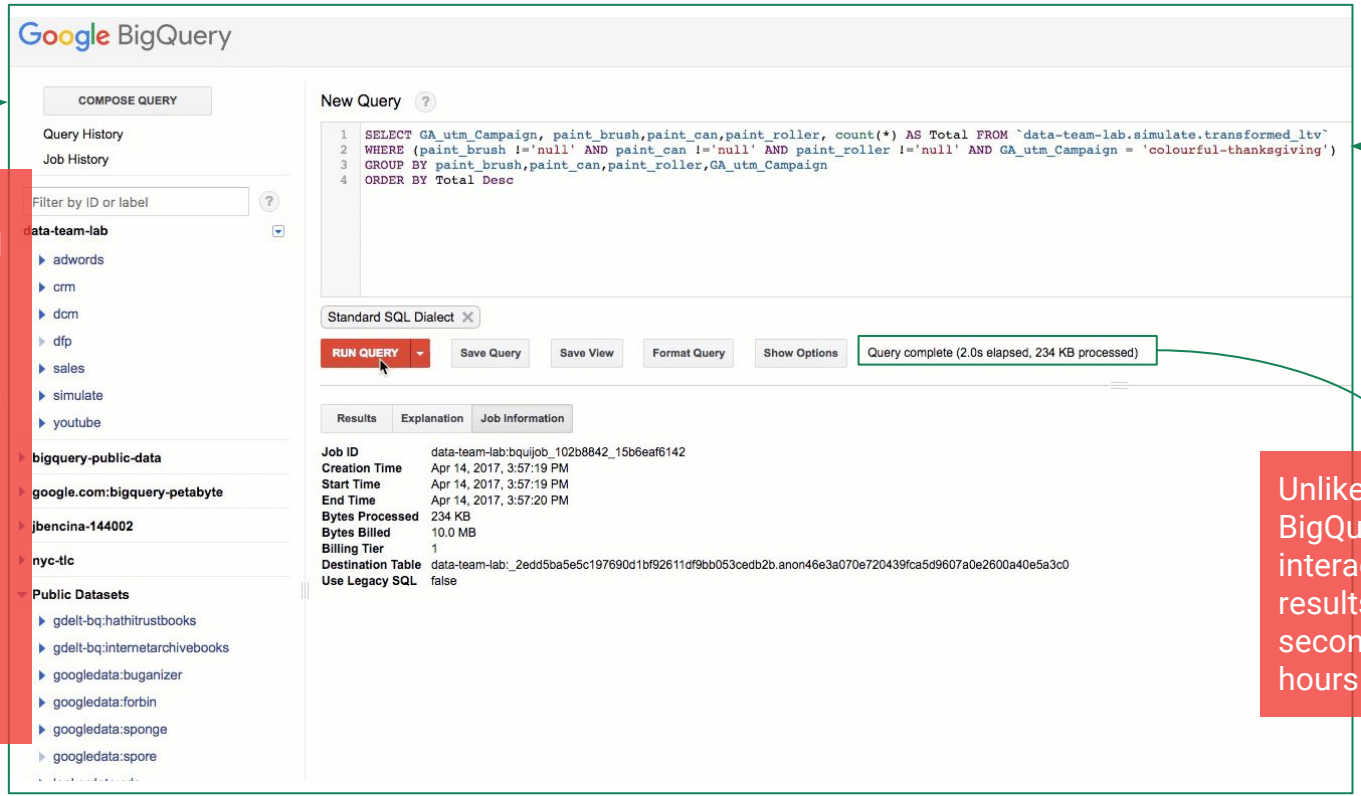
Datalab = Jupyter / ipython notebook (open source) notebook

Connect directly to BigQuery. No export / intermediate steps!

Python and sklearn library to build customer segments

4 Analyzing Data and Insights in BigQuery at Scale

Unlike on-prem/hadoop, BigQuery is Fully managed, no ops, highly scalable enterprise data warehouse on Google Cloud



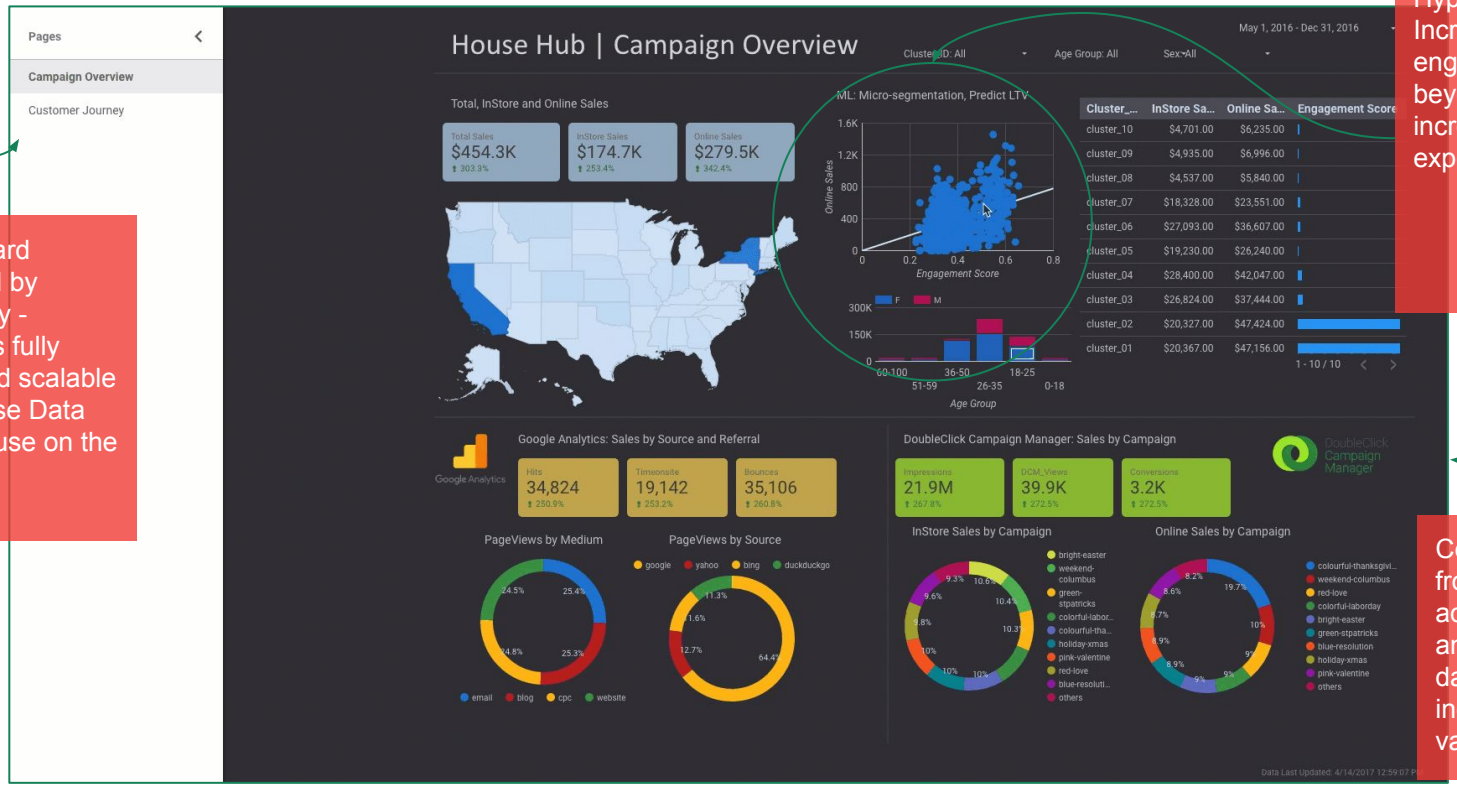
Analyse data using standard SQL

Unlike hadoop, BigQuery is interactive and results in seconds not hours / days

4 Data Visualization in Data Studio

Who are my most valuable customers?

Dashboard powered by BigQuery - Google's fully managed scalable Enterprise Data Warehouse on the cloud



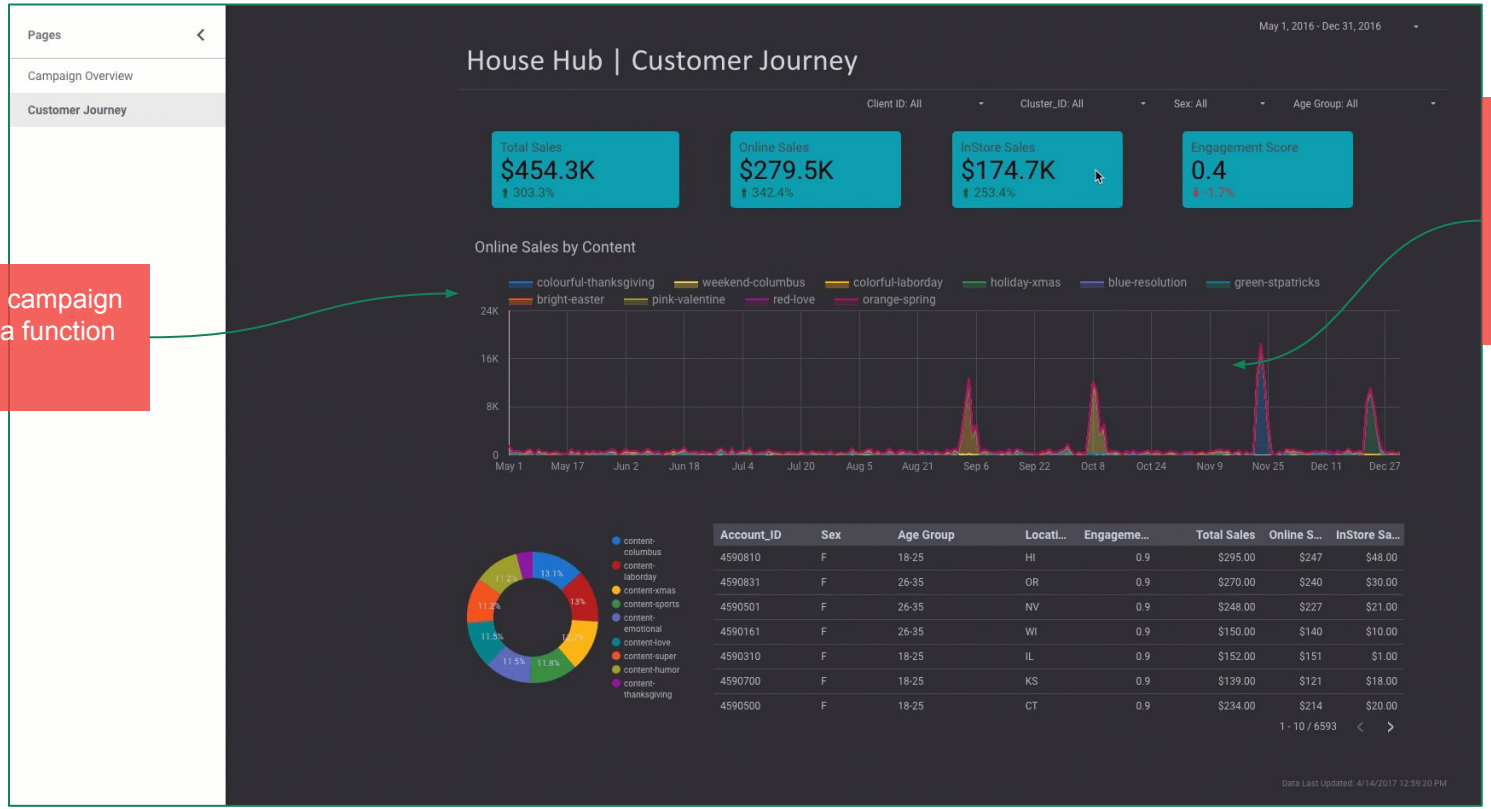
Hypothesis # Increasing brand engagement beyond a threshold increase sales exponentially?

Combining data from multiple ads sources and in-house data sources increases the value

4 Data Visualization in Data Studio

How is my customer engaging with my brand over time?

Analyse campaign data as a function of time



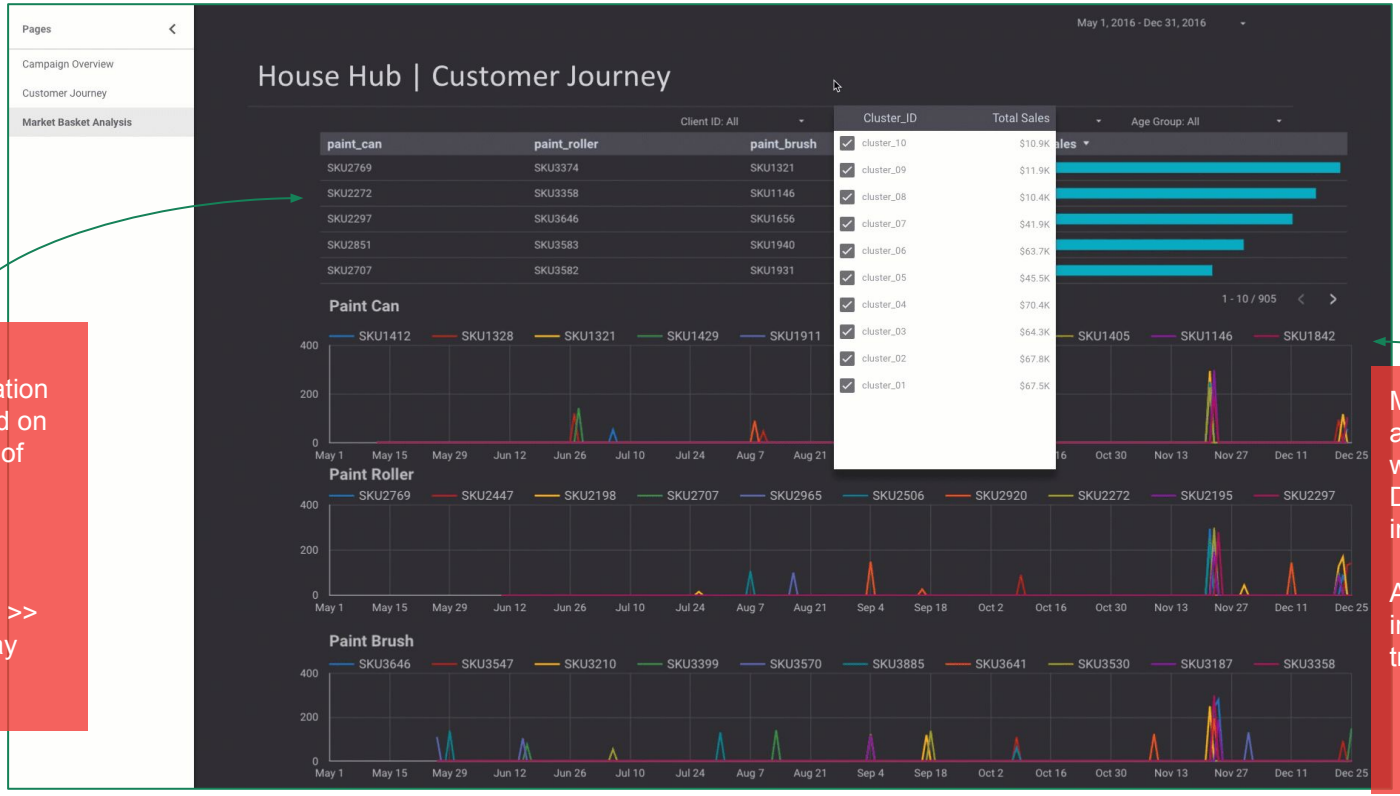
Validate your hypothesis by analyzing campaign performance data

4 Data Visualization in Data Studio

Market Basket Analysis & Recommendation Engine

Build a recommendation engine based on combination of products purchased

If customer purchased X >> customer may also like Y



Market Basket analysis made way using Dataprep earlier in the process.

Aggregations run in BigQuery on transformed data

Section 1 Section 2 Section 3 Section 4

Data Ingestion and Storage in GCP

Data Ingestion and Storage in GCP

The following components are foundational elements for moving and storing data in Google Cloud Platform

1. Ingesting Batch Data
2. Ingesting Streaming or Application Data
3. Determining Your Ingestion Architecture
4. Cloud Storage Architecture
5. Differentiating GCP Storage Options

Data Ingestion Services on GCP

- gsutil
- Cloud Storage Transfer Service
- Transfer Appliance
- Cloud Pub/Sub
- BigQuery Transfer Service
- Stackdriver Logging

*Partner tools

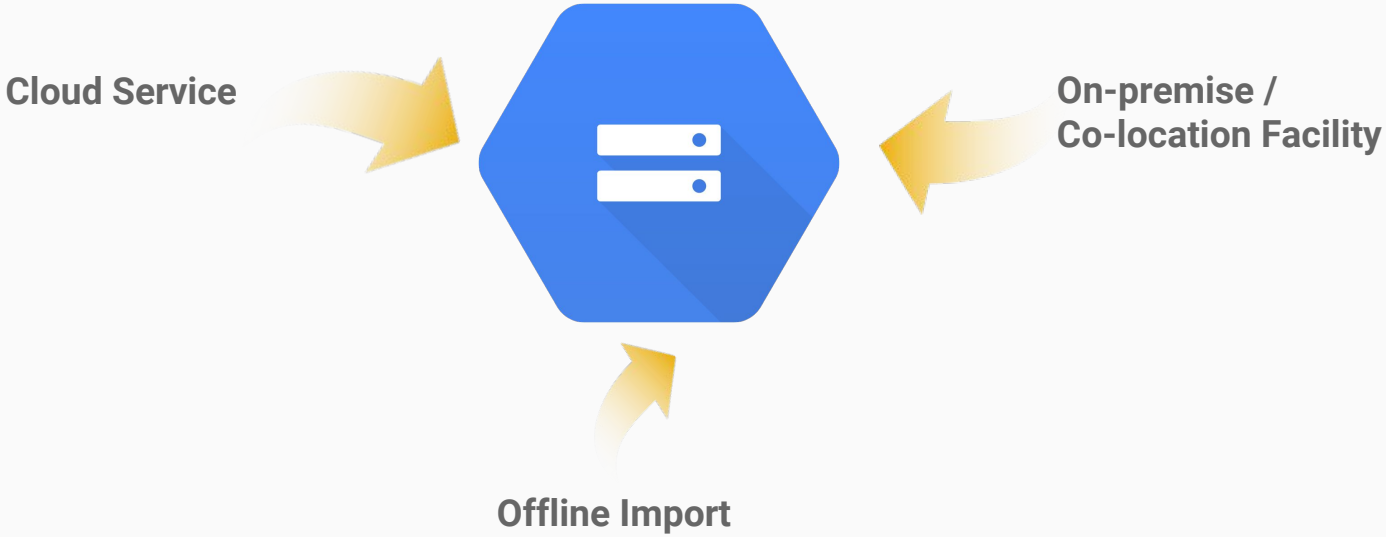
What workloads are you supporting?

Batch	Applications	Streaming
gsutil Cloud Storage Transfer Service BigQuery Transfer Service Transfer Appliance *Partner Tooling	Cloud Pub/Sub Stackdriver Logging	Cloud Pub/Sub
Use App Engine for orchestration of continuous batch transfer jobs	*App Engine, Compute Engine, Container Engine host applications that enable access to GCP services*	*Use Pub/Sub to connect to cloud or Saas hosted data sources for real time data capture*

Ingesting batch data

Cloud Storage as the Landing Stage

Whenever you move data into GCP in batch workloads, the recommended landing stage is Cloud Storage



gsutil - Ingestion of On-Prem Data

- Allows access to GCS from command line
- Use for GCS bucket and object management tasks

Technical Capabilities

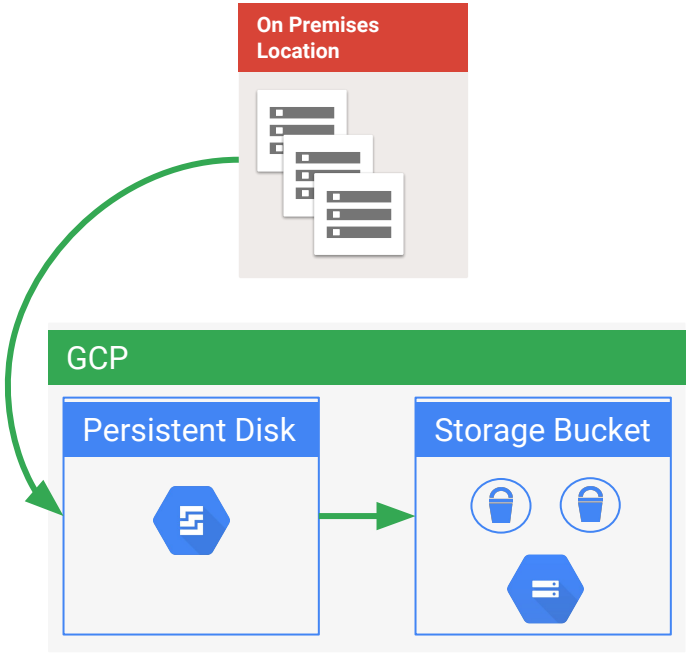
- Resumable uploads
- Multi-thread/process parallel uploads and downloads
- MD5/CRC32 checksums by default
- Retry/resume logic plumbed through every single call

Use Case

- On-premises data movement
- Ingesting publicly readable data

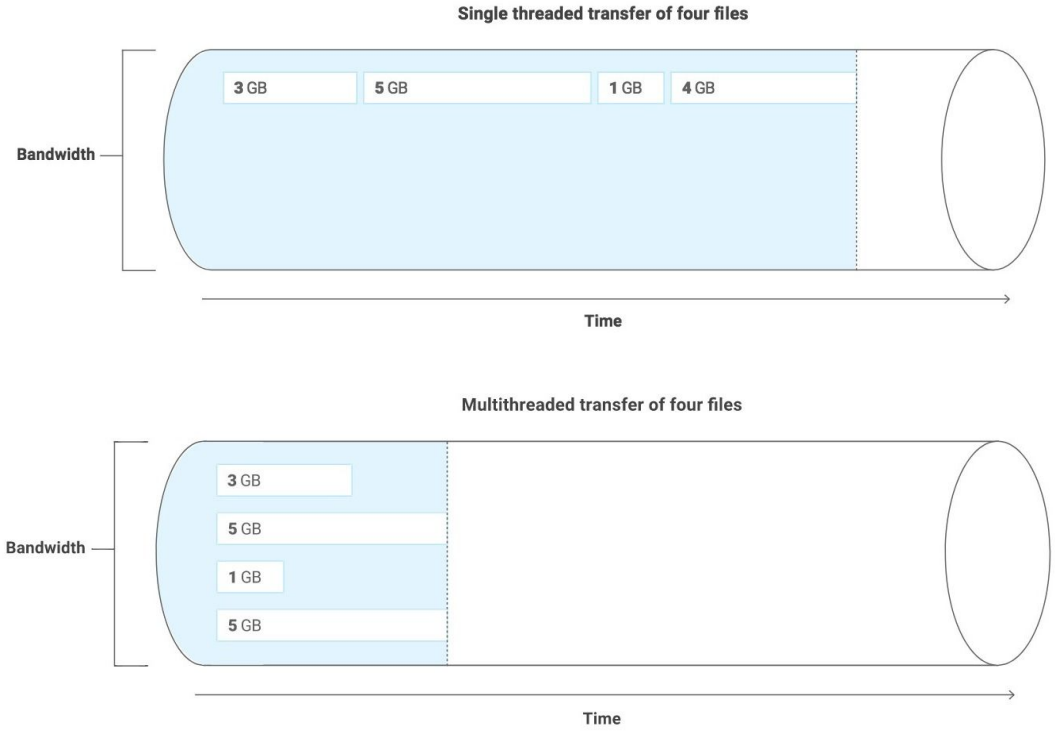
Ingestion Data Flow:

- Data Source → Persistent Disk (via command line) → Cloud Storage Bucket




```
Sample command:
gsutil -m cp *.csv gs://$BUCKET/flights/raw
```

gsutil - Multi-threading



Leverage multi threading to maximize network bandwidth and enable quicker ingestion

Cloud Transfer Service - Importing Online Data

- Quickly import online data into GCP
- *Data sink* (the destination) is always a GCS bucket
- *Data source* can be an Amazon S3 bucket, an HTTP/HTTPS location, or a GCS bucket

Technical Capabilities

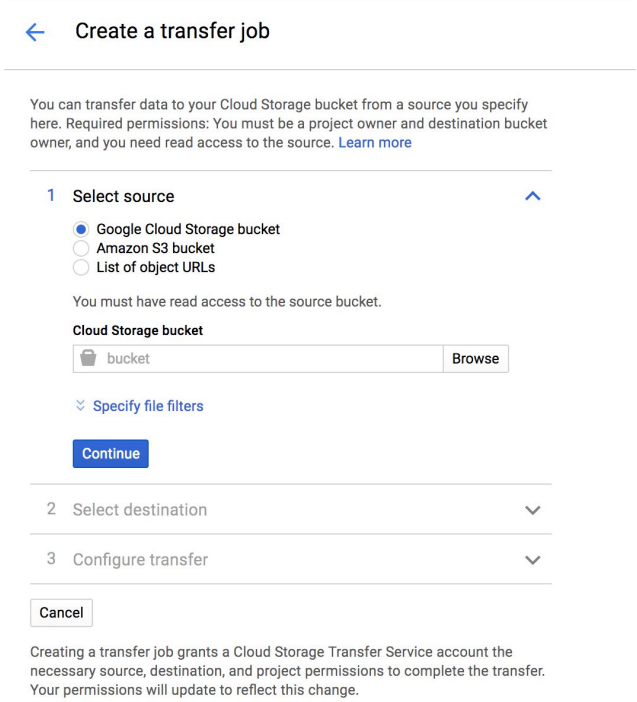
- Use Google API Client library of your choice
- Programatically create and manage jobs via Rest APIs
- Selectively transfer files from data source(s)
- Schedule periodic syncs with data source

Use Cases

- Online data movement
- Moving larger datasets (>10TB)

Ingestion Data Flow

S3 or HTTP/HTTPS → Cloud Storage Bucket
Cloud Storage Bucket → Cloud Storage Bucket
S3 → GCS → BigQuery



Using the Google Transfer Appliance

- Third party picks up data from your data center
- Send drives to third party for transmission to GCP

Use Case

- Large data sets with poor network bandwidth
- No network to speak of

North American vendor Is Iron Mountain

- $85 per drive + $0.03/GB
- >200 drives handled since launch

Demo: Batch Ingestion into GCP

The Bureau of Transportation Statistics (BTS) stores yearly flights data for flights in the United States in their data center. The data has every instance of a flight in the US each year, plus key data points about each flight. Data can be downloaded manually using a form online. We want to do the following:

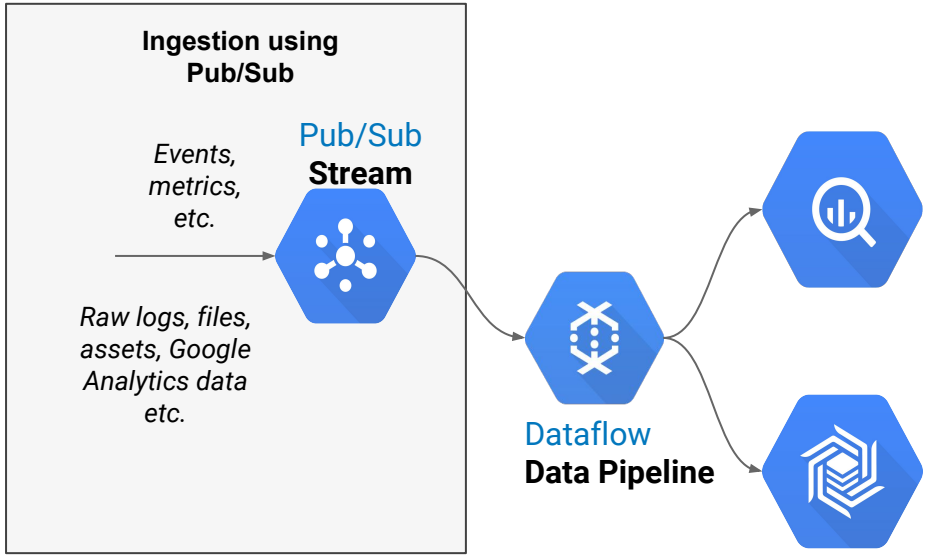
1. Programatically ingest the data for 2015 into GCP
2. Unzip the ingested file to csv
3. Perform minor data cleansing to remove quotes and commas in the csv files
4. Load the cleansed data in Cloud Storage

What Ingestion tool can we use to do this?

Source: https://github.com/GoogleCloudPlatform/data-science-on-gcp.git

Ingesting Streaming or Application Data

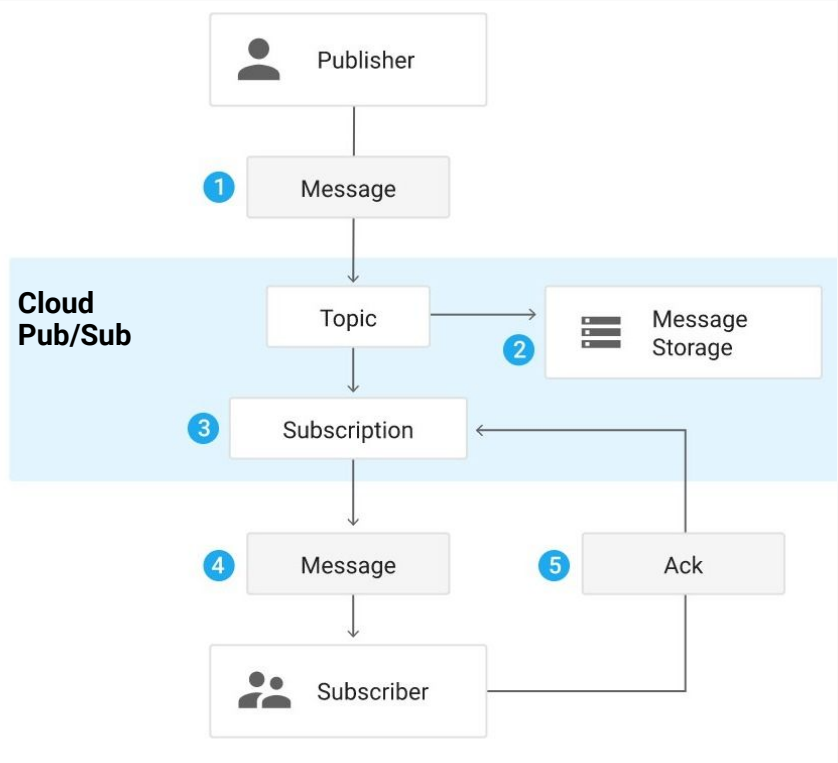
Ingesting Streaming Data Using Pub/Sub



BigQuery
Extremely fast
and cheap on-demand analytics engine

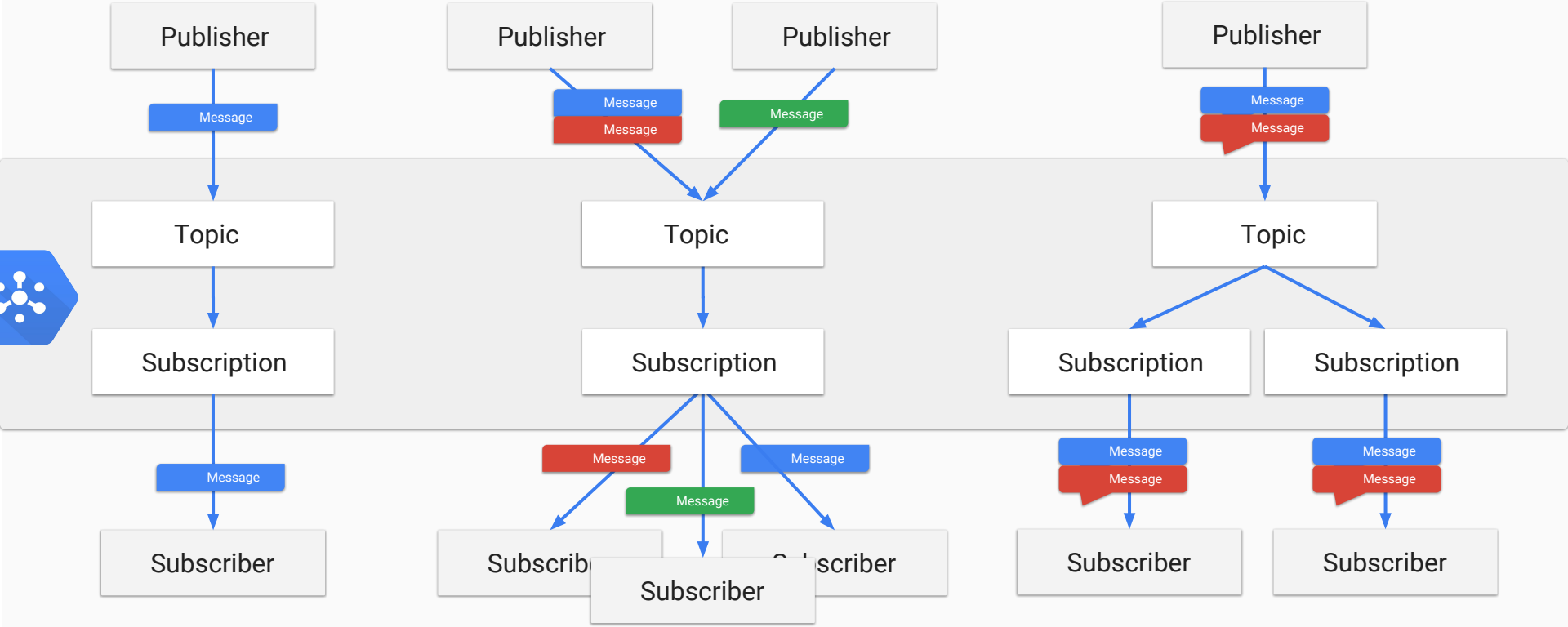
Bigtable
High performance
NoSQL database for large workloads

Pub/Sub Concepts and Data Flow



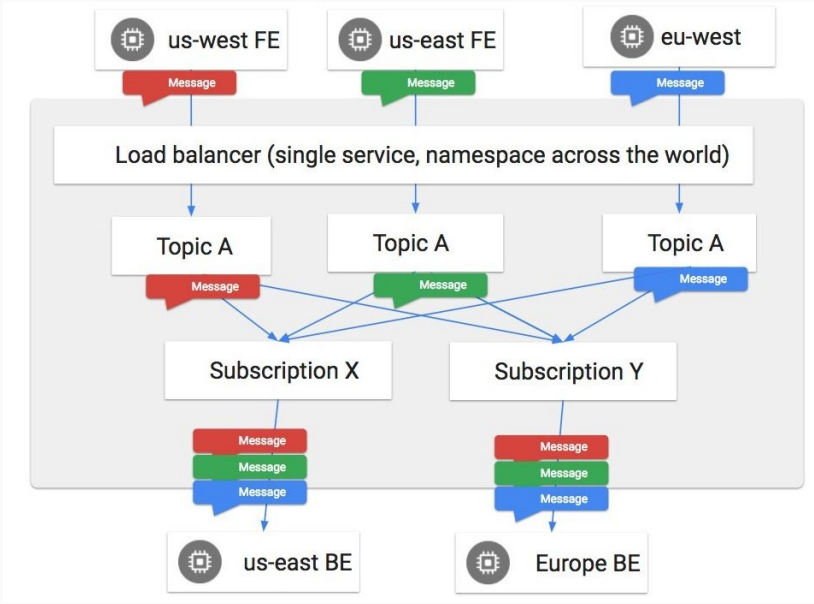
1. **Publisher** application sends message to a Pub/Sub **Topic**
2. Pub/Sub persists message in a **message store** until delivery and acknowledgement
3. Pub/Sub sends the message to its respective **Subscription**
4. **Subscriber** application receives pending messages from its subscription
5. Subscriber sends **message acknowledgement** to Subscription and the message is deleted from the message store

Setting up Ingestion Patterns for Streaming Data

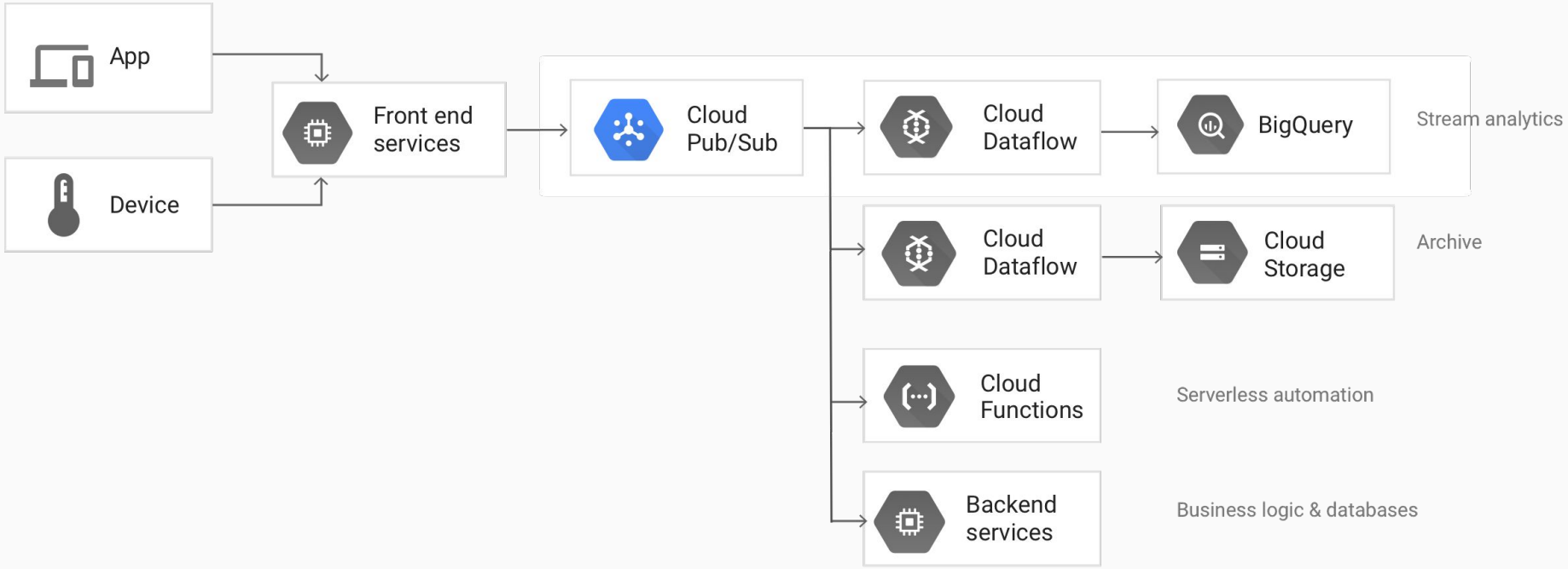


Pub/Sub Technical Characteristics

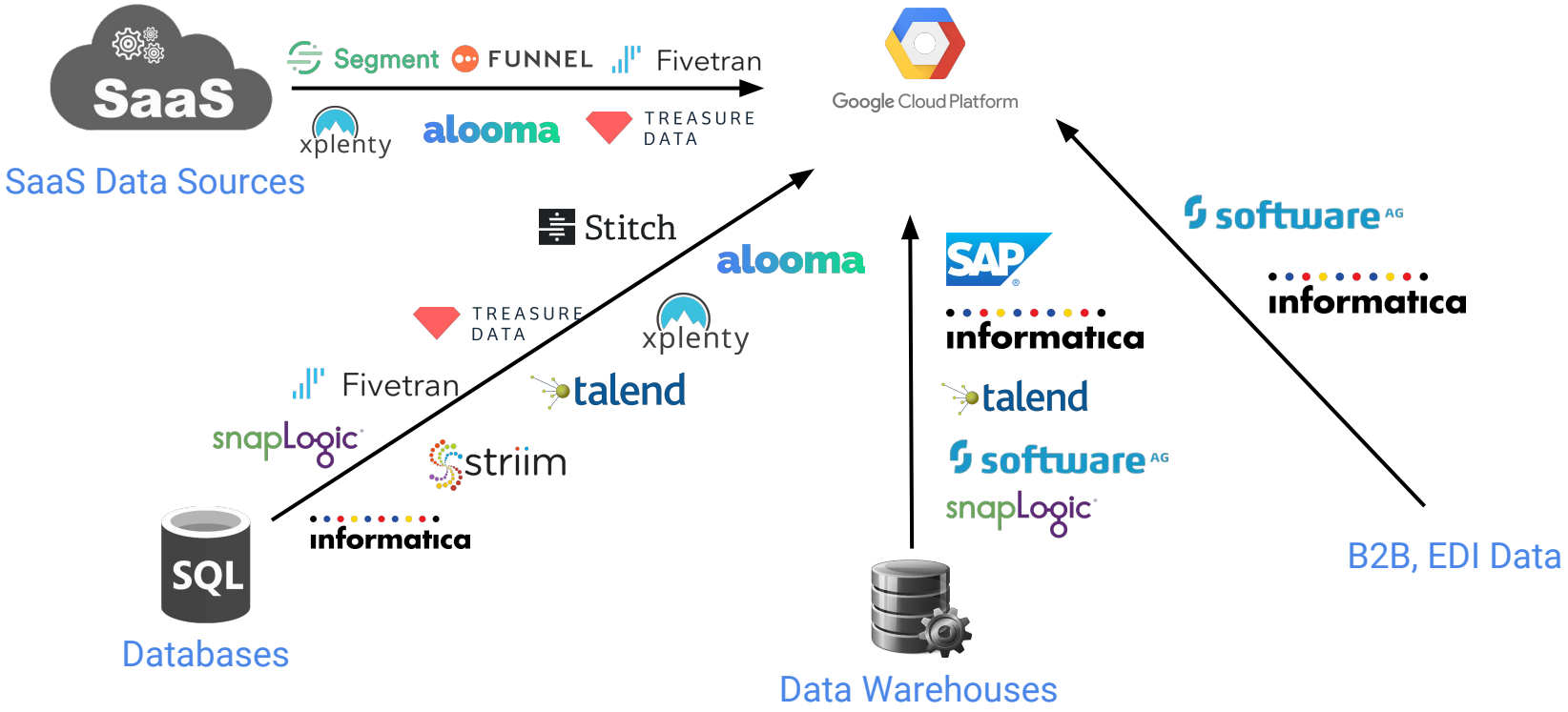
- Global service
- Data encryption in transit
- At least once delivery
- No order guarantees
- Fast: <150 ms 99.9% of publish, <1 s 99.9% end-to-end
- Scalable from 1 KB/s to 100 GB/s, with consistent performance.
- Durable: 3x3 sync data replication before ACK, for 7 days



Enabling Event Streaming Services with Pub/Sub



GCP Data Integration Partners

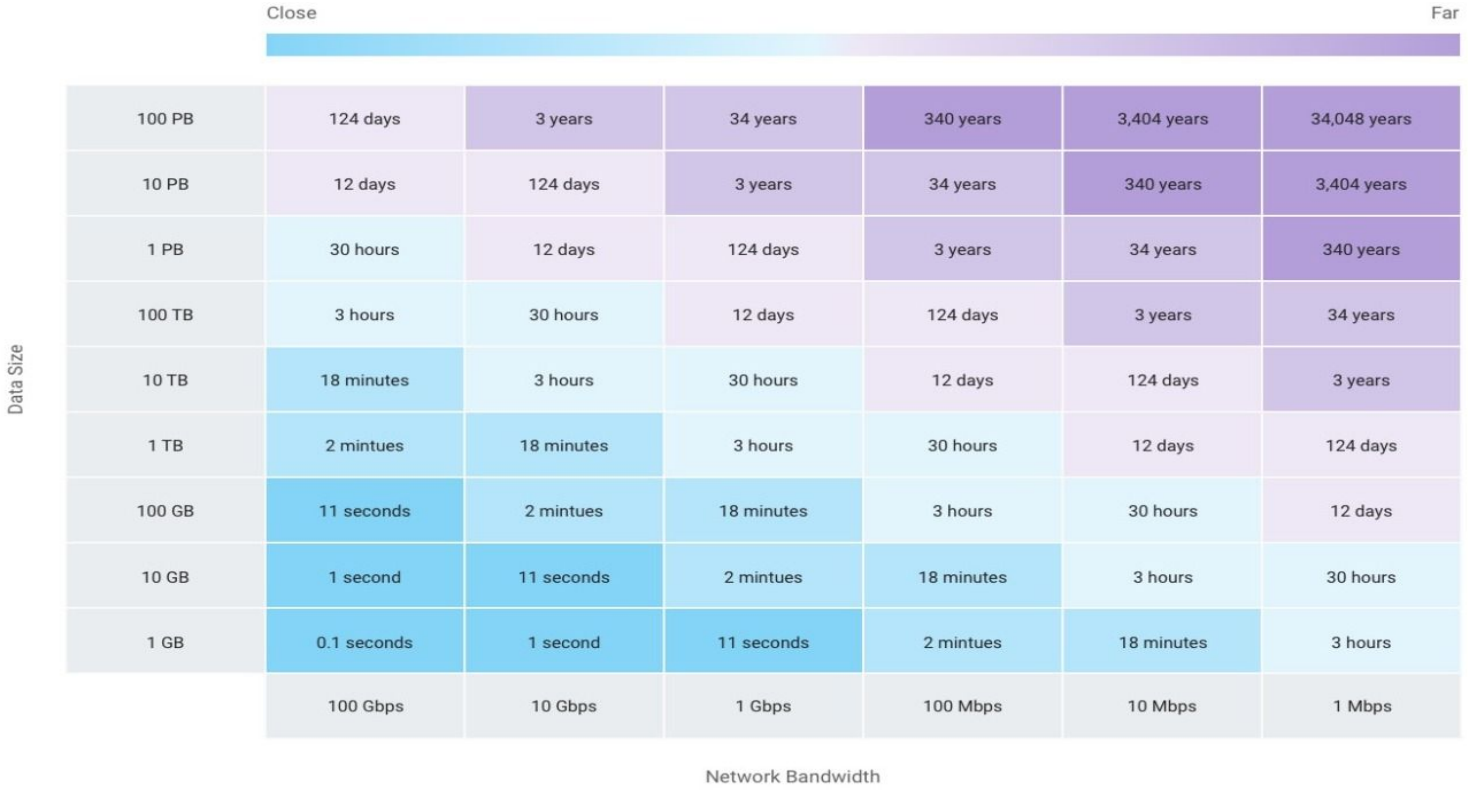


A Summary of Data Ingestion Options

	Use Cases	Considerations
gsutil & GCP Console	• Command-line tools to manage GCP storage platforms such as GCS	• Recommended for 10TB or less • Very good for on-premise moves • Limited by network bandwidth
Cloud Storage Transfer	• Quick import of online data	• Good for 10TB+ or PB scale data • Monitoring for over 100M objects • 24 hour / day scheduler
Google Transfer Appliance	• Send your hard drives to third-party service for transmission to GCP	• N. American vendor is Iron Mountain $85 per drive + $0.03/GB • 200 drives handled since launch
Direct Ingestion to BigQuery	• Simplify ingestion process by ingesting directly into BigQuery	• Avoid slot contention • Moderate size reference data loads
Pub/Sub	• Asynchronous messaging that decouples senders and receivers	• Messages are delivered in near real time, with preference given to delivering older messages first
Partners	• Leverage existing relationships with partners	• ETL partners • Data replication partners

Determining Your Ingestion Architecture

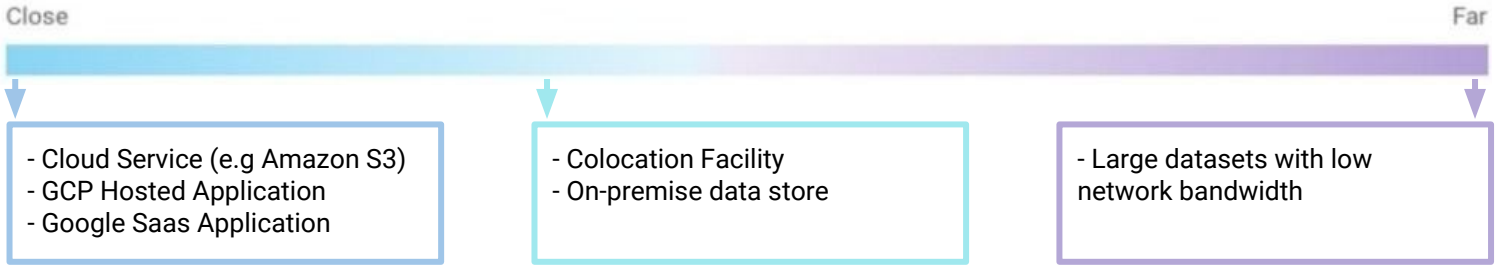
How Accessible Is the Data?



Data Size	100 Gbps	10 Gbps	1 Gbps	100 Mbps	10 Mbps	1 Mbps
100 PB	124 days	3 years	34 years	340 years	3,404 years	34,048 years
10 PB	12 days	124 days	3 years	34 years	340 years	3,404 years
1 PB	30 hours	12 days	124 days	3 years	34 years	340 years
100 TB	3 hours	30 hours	12 days	124 days	3 years	34 years
10 TB	18 minutes	3 hours	30 hours	12 days	124 days	3 years
1 TB	2 mintues	18 minutes	3 hours	30 hours	12 days	124 days
100 GB	11 seconds	2 mintues	18 minutes	3 hours	30 hours	12 days
10 GB	1 second	11 seconds	2 mintues	18 minutes	3 hours	30 hours
1 GB	0.1 seconds	1 second	11 seconds	2 mintues	18 minutes	3 hours

Google Cloud

Data Accessibility by Source

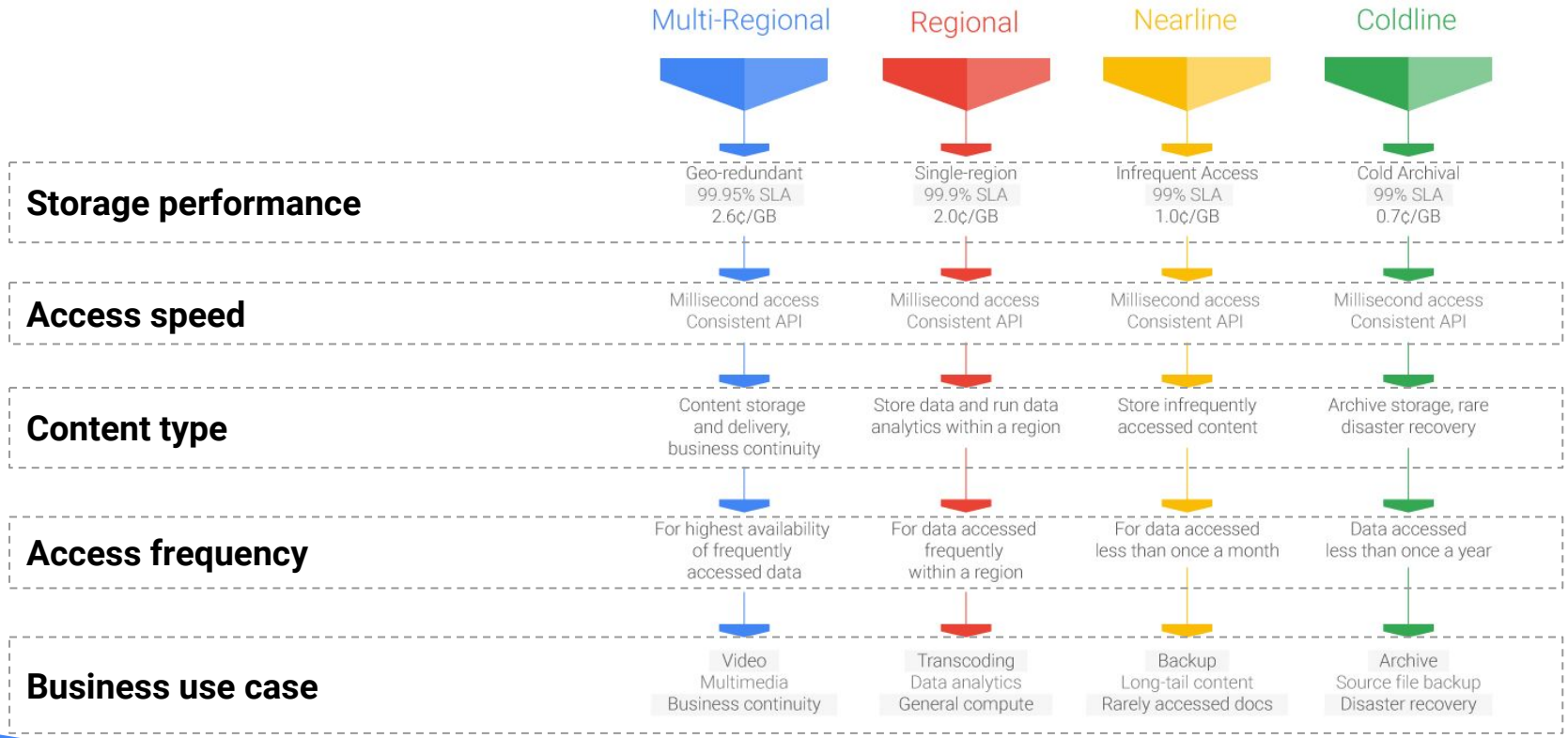


	Close		Far
Data Sources	- Cloud Service (e.g Amazon S3) - GCP Hosted Application - Google Saas Application	- Colocation Facility - On-premise data store	- Large datasets with low network bandwidth
Ingestion Service Options	Cloud Storage Transfer Service BigQuery Transfer Service Cloud Pub/Sub Stackdriver Logging	gsutil Cloud Pub/Sub	Transfer Appliance

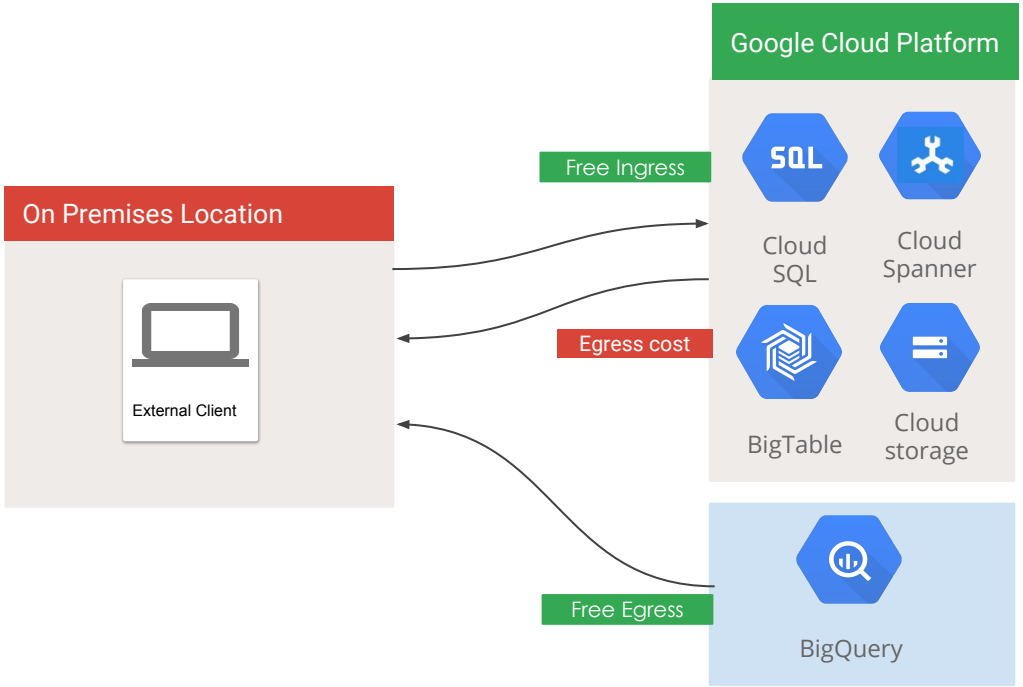
Cloud Storage Architecture

Regions and Zones

Selecting the Right Storage for Your Use Case



Google Cloud Platform: Ingress/Egress

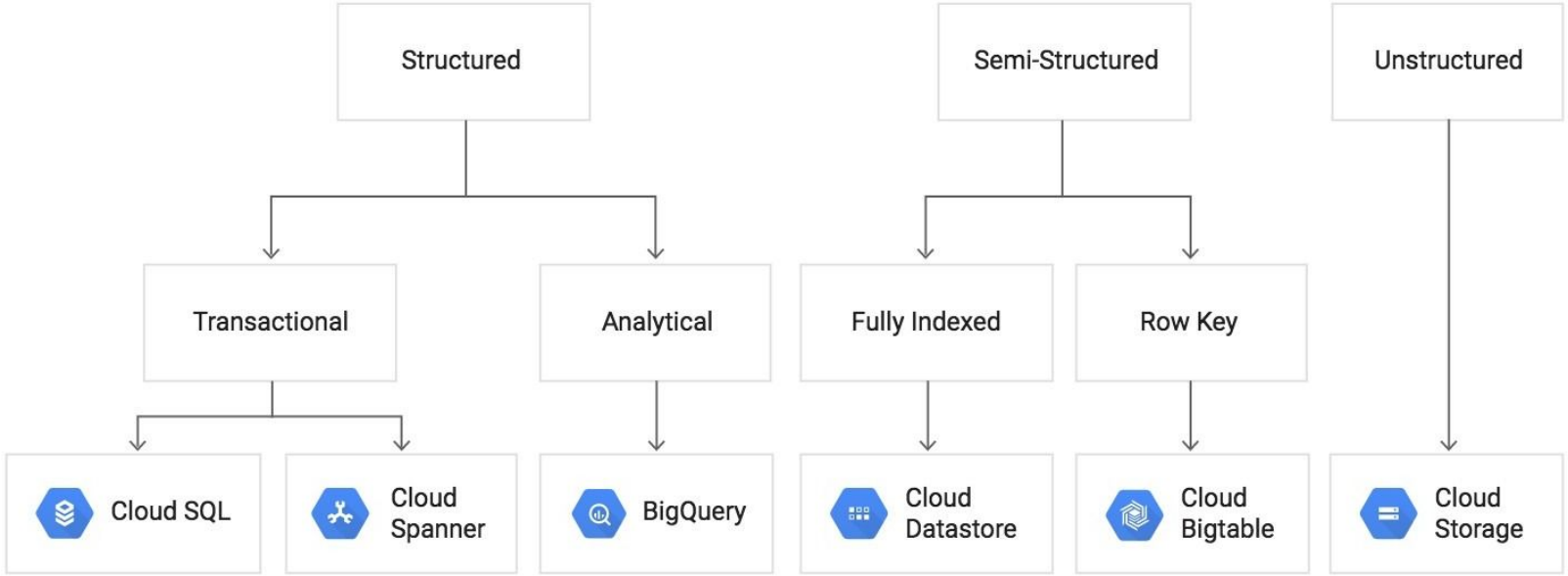


Google Cloud Storage Architecture Considerations

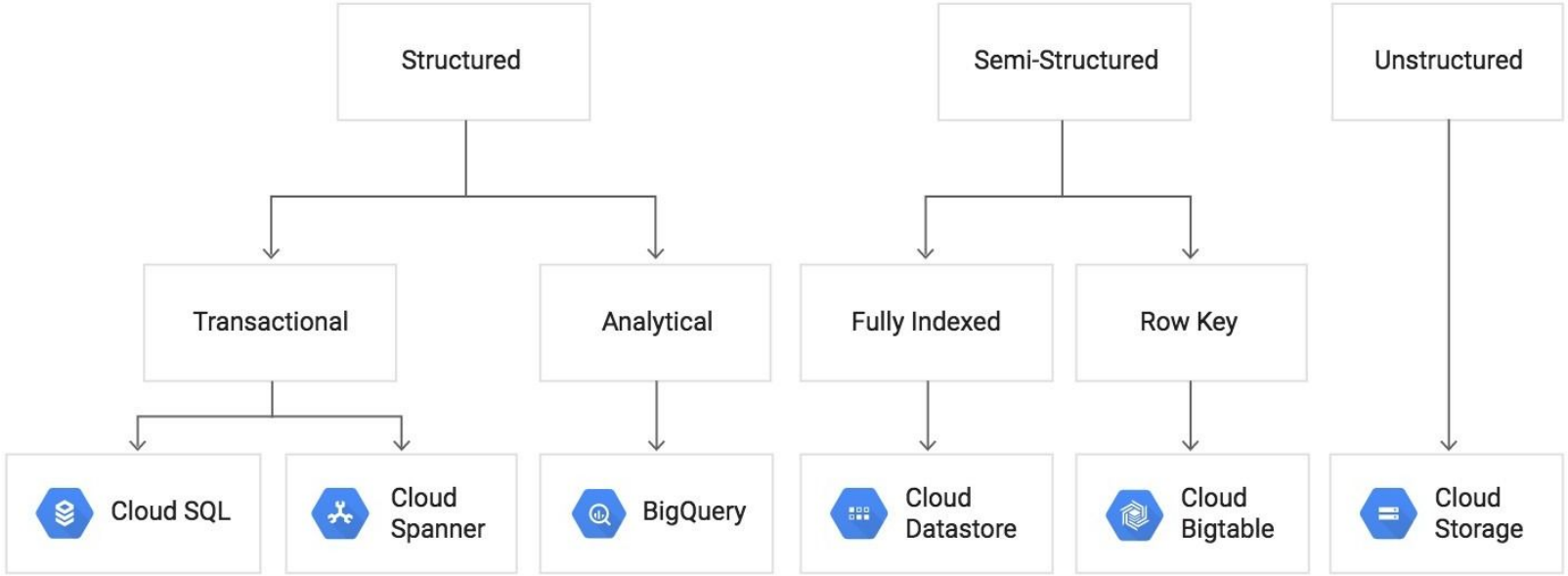
	Description
GCS Tier	Selecting the correct GCS tier is important. This easily fixed if you select the wrong one, but you could incur early deletion charges.
Bucket Naming	GCS uses a global bucket namespace. Bucket names must be unique. Pre-create all buckets as necessary. Do not depend on bucket creation or deletion in critical path of application.
Managing Multiple Buckets	If you need lots of buckets, use GUIDs, put retry logic in your code to handle collisions, and keep a list to cross-reference bucket names
Request Rate	GCS has no upper bound on request rate, but if you plan to receive 1000 write requests per second, your code should ramp up the request rate

Differentiating GCP Storage Options

A Guide for Selecting Your Storage Option



A Guide for Selecting Your Storage Option



OLAP vs OLTP: Which Fits My Use Case?

	OLTP	OLAP
	OnLine Transaction Processing	OnLine Analytical Processing
Data Source	Operational	Historical
Focus	Updating/Retrieve	Reporting
Queries	Simple	Complex
Read Speed	Fast	Slow
Google Cloud Platform Products	 Cloud SQL, Cloud Datastore, Cloud Spanner, BigTable	 BigQuery, Cloud storage

Differentiating Storage Options in GCP

	Product	Simple Description	Like	Good For	Bad For
	**CloudSQL**	**Well-understood VM-based RDBMS**	MySQL, Amazon RDS, Amazon Aurora	Web frameworks, existing applications	Scaling, analytics, heavy writes
	**Datastore**	**Scalable store for structured serve**	MongoDB, CouchDB	GAE apps, structured pure-serve use cases	Relational or analytic data
	**Bigtable**	**High-volume, low-latency database**	HBase, Cassandra, DynamoDB	“Flat”, heavy read/write, or analytical data	High structure or transactional data
	**GCS**	**Binary/object store**	S3	Large or rarely accessed unstructured data	Structured data, building fast apps
	**BigQuery**	**Auto-scaling analytic data warehouse**	Redshift, Impala, Hive	Interactive analysis of static datasets	Building fast apps
	**Spanner**	**Relational DB service**	CockroachDb	Low latency transactional systems	Analytic data

Google Cloud

Technical Considerations for Design Decisions

	Product	Interface	R/W Latency	Typical Size	Storage Type
	**CloudSQL**	SQL	Low (ms)	< 10TB	Relational
	**Datastore**	Proprietary / NoSQL	Medium (10s of ms)	< 200TB	Document
	**Bigtable**	HBase API	Low (ms)	2TB-10PB	Key/Value
	**Cloud Storage**	REST/WebUI	Medium (100s of ms)	Any	Object
	**BigQuery**	REST / SQL / WebUI	High (s)	Any	Columnar
	**Spanner**	SQL	Low (ms)	Any	Relational

Section 1 Section 2 Section 3 Section 4

Transforming and Processing Data in GCP

Transforming and Processing Data in GCP

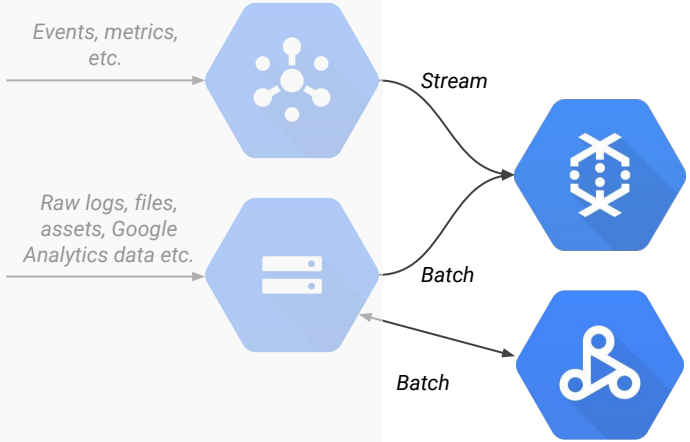
Transforming and processing data in GCP requires understanding of the following building blocks

1. Data Processing Services in GCP
2. Deep Dive into Cloud Dataflow
3. Deep Dive into Cloud Dataproc
4. Migrating to Cloud Dataproc
5. Architectures Over the Course of Migration

Data Processing Services in GCP

Differentiating Between Services

Processing—Data from source systems is cleansed, normalized, and processed across multiple machines, and stored in analytical systems.



Cloud Dataflow
Data processing engine for batch and stream processing

Cloud Dataproc
Managed Spark and Hadoop

Key Characteristics of Dataflow and Dataproc

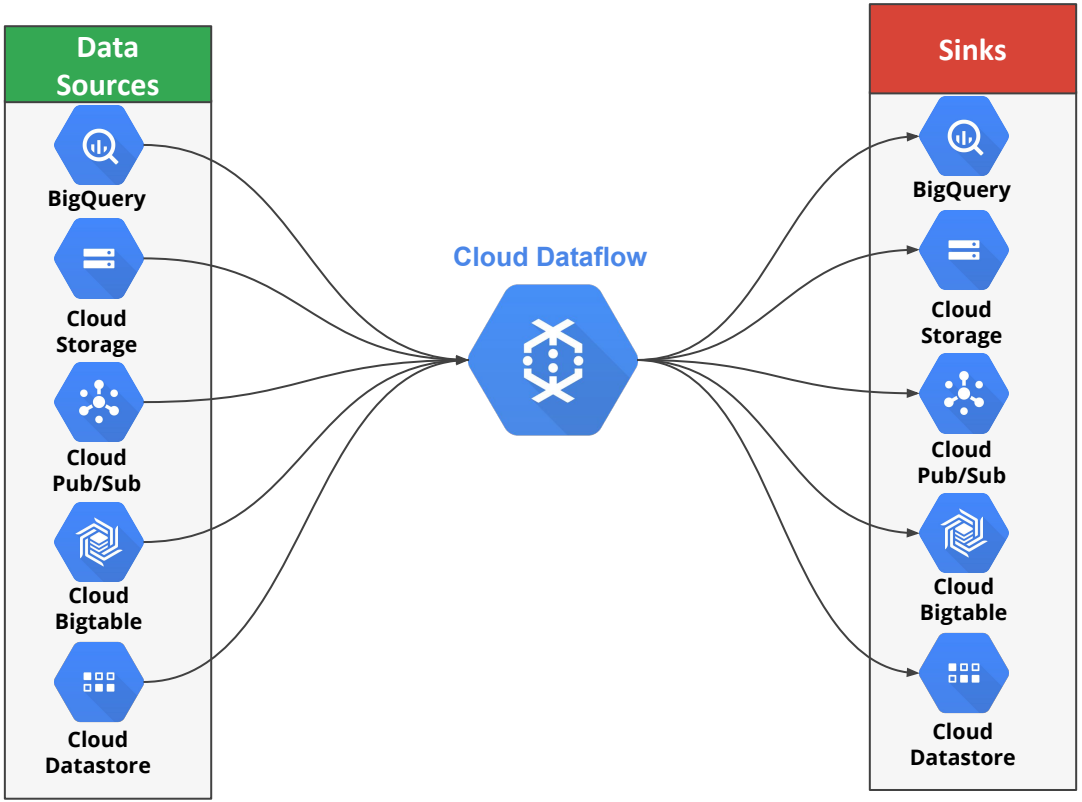
	 Cloud Dataflow	 Cloud Dataproc
Recommended For	New data processing pipelines, unified batch and streaming	Existing Hadoop/Spark applications, machine learning/data science ecosystem, large-batch jobs, preemptible VMs
Use Cases	Advertisers/contents, streaming video providers, real-time inventory	Data analysis, do development on open-source stack, pay-per usage
Fully-Managed	Yes	No
Expertise	Apache Beam	Hadoop, Hive, Pig, Apache Big Data ecosystem
Auto-Scaling	Yes	No
Interface	Cloud SDK, Cloud Console, Dataflow API	Cloud SDK, Cloud Console, SSH
Ephemeral	No	Yes

Deep Dive into Cloud Dataflow

What is Cloud Dataflow?

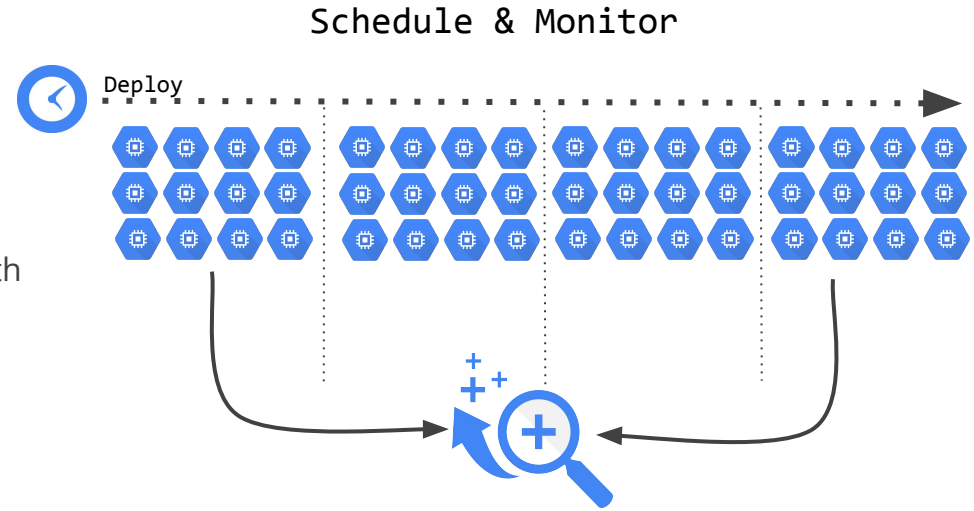
- Unified batch and streaming processing
- Fully Managed, No-Ops data processing
- Open source programming model  beam
- Intelligently scales to millions of QPS

Data Sources and Sinks for Dataflow



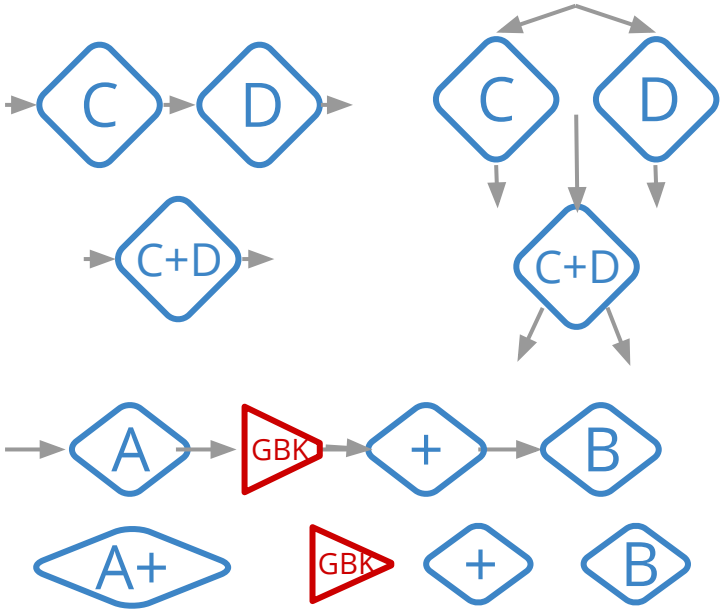
Why Customers Value Dataflow (technical perspective)

1. **Fully-managed and auto-configured**
2. Auto graph-optimized for best execution path
3. Autoscaling mid-job
4. Dynamic Work Rebalancing mid-job



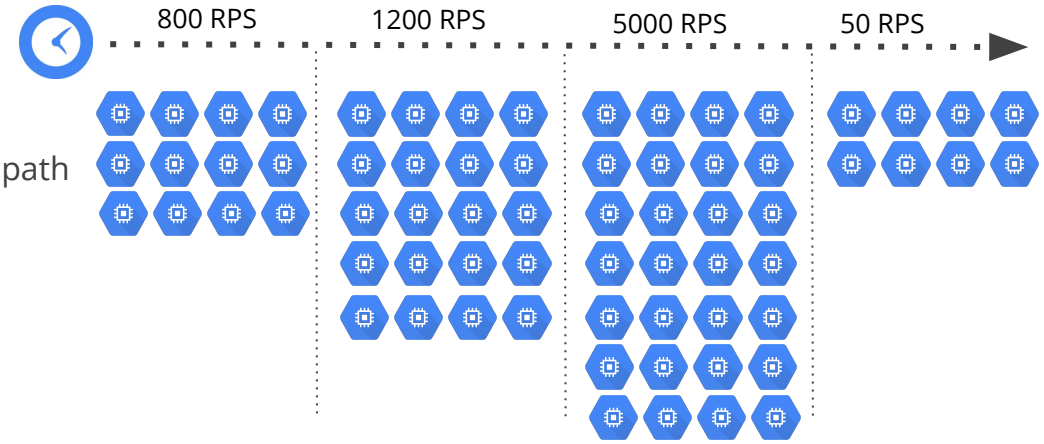
Why Customers Value Dataflow (technical perspective)

1. Fully-managed and auto-configured
2. **Auto graph-optimized for best execution path**
3. Autoscaling mid-job
4. Dynamic Work Rebalancing mid-job



Why Customers Value Dataflow (technical perspective)

1. Fully-managed and auto-configured
2. Auto graph-optimized for best execution path
3. **Autoscaling mid-job**
4. Dynamic Work Rebalancing mid-job



*means 100% cluster utilization by definition

Why Customers Value Dataflow (technical perspective)

1. Fully-managed and auto-configured
2. Auto graph-optimized for best execution path
3. Autoscaling mid-job
4. **Dynamic Work Rebalancing mid-job**

Example 1

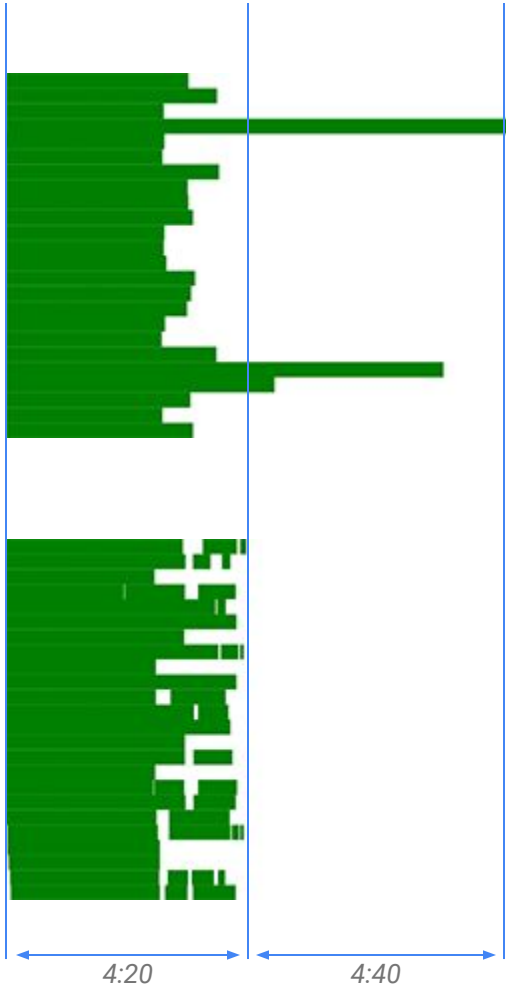
1 ParDo

Skewed data distribution

24 workers

Without load balancing: 9 min

With load balancing: ~4 min



Example 2

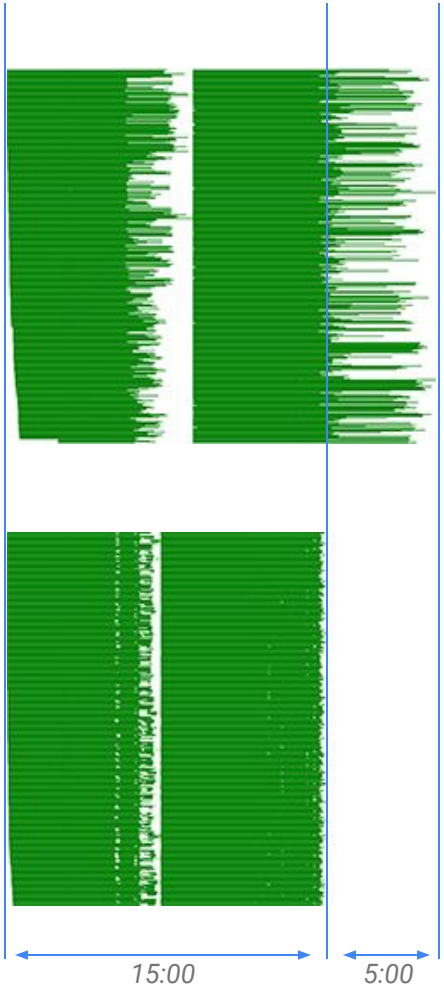
ParDo/GBK/ParDo

Uniform data distribution

400 workers

Without load balancing: 20 min

With load balancing: 15 min



Dataflow Programming Model

3 basic concepts for building pipelines using Apache Beam

1 PCollections

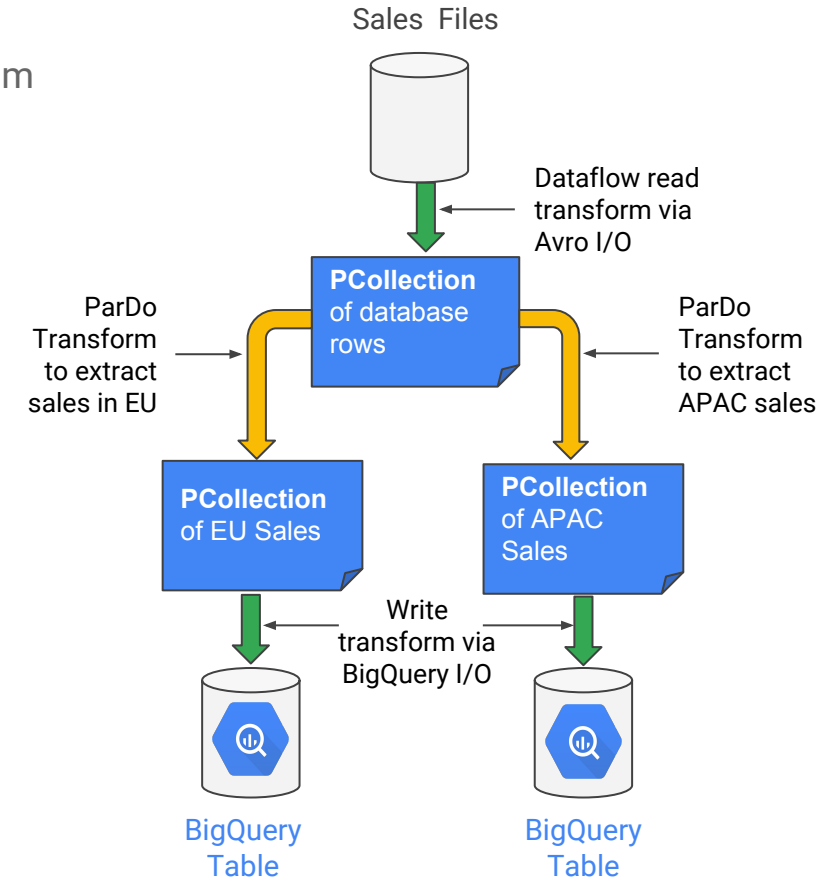
- Object abstraction that represents a potentially distributed, multi-element data set (ie. pipeline data)

2 Transformations

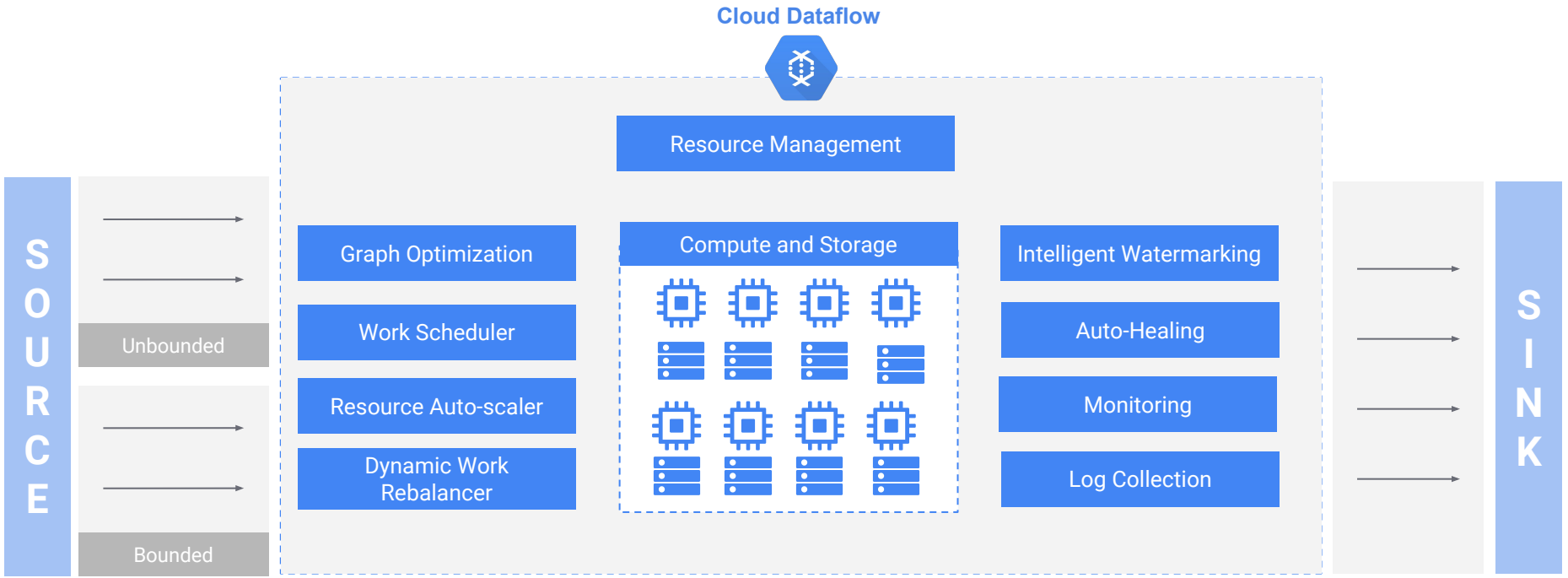
- Operations that shape data in your pipeline and provide a generic processing framework.
- Take PCollections input and output new PCollections

3 Pipeline I/O

- Capability for data read and write transforms for multiple storage and file types



Cloud Dataflow: Under the Hood



Streaming Data Using Pub/Sub Dataflow

Original Dataflow code

```
Pipeline p = Pipeline.create();
    p.begin()
        .apply(TextIO.Read.from("gs://…"))
        .apply(ParDo.of(new ExtractTags()))
        .apply(Count.create())
        .apply(ParDo.of(new ExpandPrefixes())
        .apply(Top.largestPerKey(3))
        .apply(TextIO.Write.to("gs://…"));
    p.run();
```

Modified Dataflow code

```
Pipeline p = Pipeline.create();
    p.begin()
.apply(PubsubIO.Read.from("projects/<project_id>/topics/<
topic_name>"))
        .apply(ParDo.of(new ExtractTags()))
        .apply(Count.create())
        .apply(ParDo.of(new ExpandPrefixes())
        .apply(Top.largestPerKey(3))
.apply(PubsubIO.Write.to("projects/<project_id>/topics/<t
opic_name>"))
    p.run();
```

Dataflow Design Considerations

Key Considerations for Designing Your Pipeline

Below are key design considerations for building pipelines on GCP using Dataflow

1. Dataflow is generally the best way to build big data pipelines on GCP

2. The dataflow programming model handles both batch and streaming manner which allows you to use the same code base for batch and stream processing

3. The code for the dataflow pipeline will be written in Apache Beam in either Python or Java

4. The code base is portable which allows the capability to execute it in multiple environments

5. To execute your code, Beam submits your pipeline to a runner. The ‘DataflowRunner’ enables executing the pipeline on the Dataflow Managed Service

Things to Think About as You Design Your Pipeline

What are you computing?

Where in event time are you processing computations?

When in processing times are results emitted?

Impacts the transforms you use in your pipeline

- Element-wise (e.g - ParDo, Filter
- Aggregations (e.g - Groupby Key, Combine Key)
- Composites (e.g - MapElements + Sum.IntegersPerKey)

Things to Think About as You Design Your Pipeline

What are you computing?

Where in event time are you processing computations?

When in processing times are results emitted?

Impacts the windowing methods you use

- Fixed
- Sliding
- Sessions

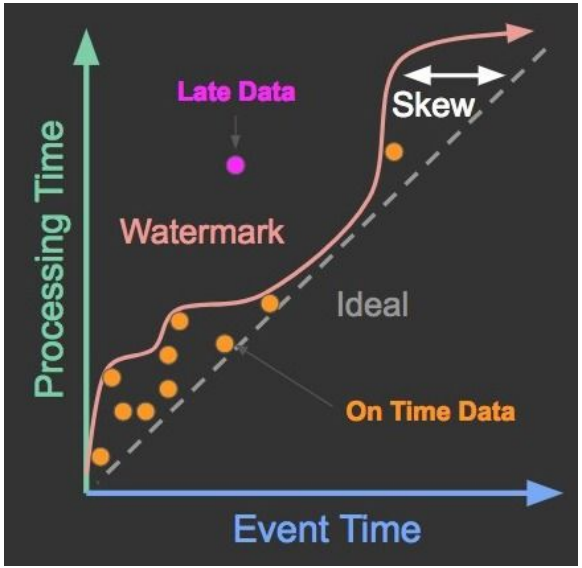
Things to Think About as You Design Your Pipeline

What are you computing?

Where in event time are you processing computations?

When in processing times are results emitted?

Impacts how you use triggers and watermarks to determine when to emit stream processing



Things to Think About as You Design Your Pipeline

What are you computing?

Where in event time are you processing computations?

When in processing times are results emitted?

Impacts how you use triggers and watermarks to determine when to emit stream processing

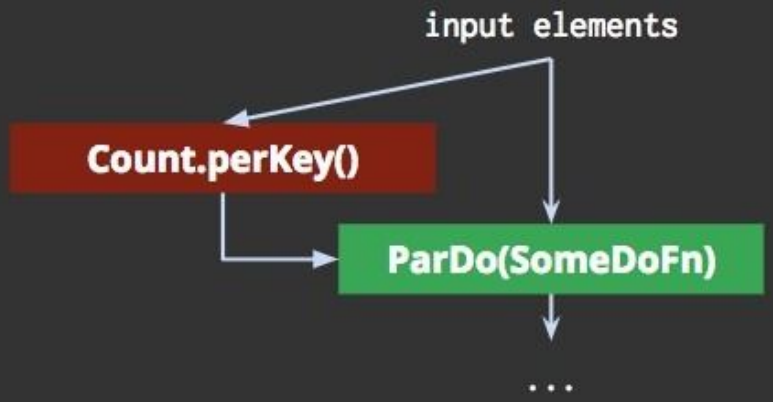
```
PCollection<KV<String, Integer>>
scores = input

.apply(Window.into(FixedWindows.o
f(Minutes(2))

.triggering(AfterWatermark.pastEn
dOfWindow()))
    .apply(Sum.integersPerKey());
```

Side inputs

ParDos can receive extra inputs “on the side”

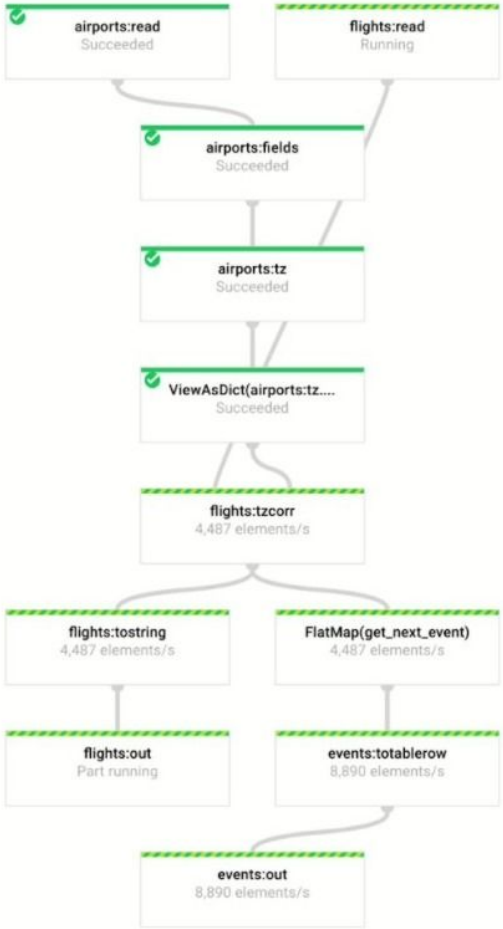
For example broadcast the count of elements to the processing of each element

Side inputs are computed (and accessed) per-window



Demo: Dataflow Pipeline

Our goal enable a use case where we want to make decisions in real time based on expected arrival times. We will need to make some transformations to our datasets



Google Cloud

Deep Dive into Dataproc

The Apache Ecosystem

Use Cases for Cloud Dataproc

1. Big Data processing for Hadoop and Spark
2. Large hosted processing of clusters across verticals
3. Web commerce processing / analysis
4. Genomics data processing

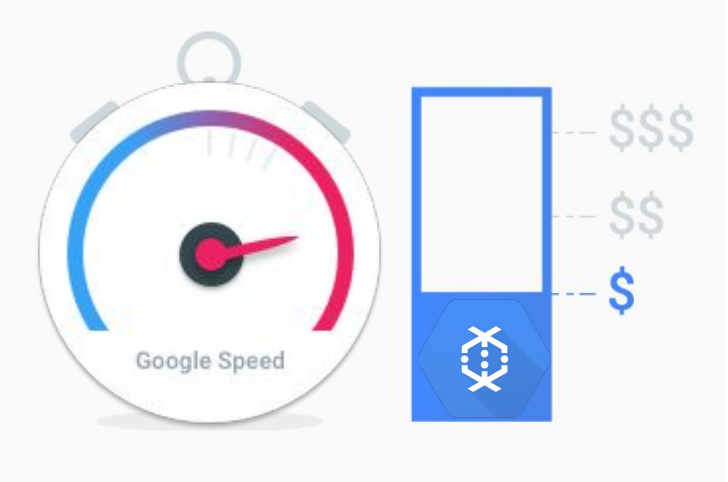
Ephemerality

Use and pay for things only when you need them.

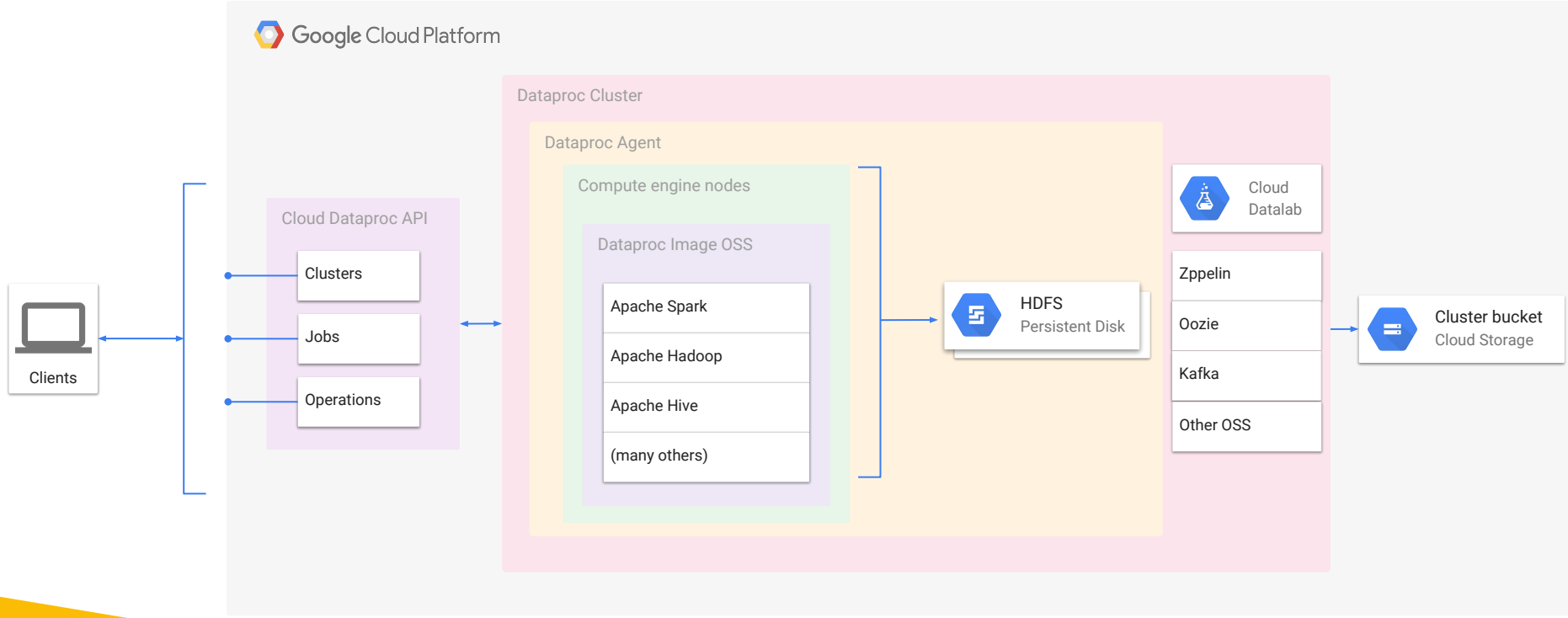
On-demand (ephemeral) Spark and Hadoop service.

Only run clusters when you need them.

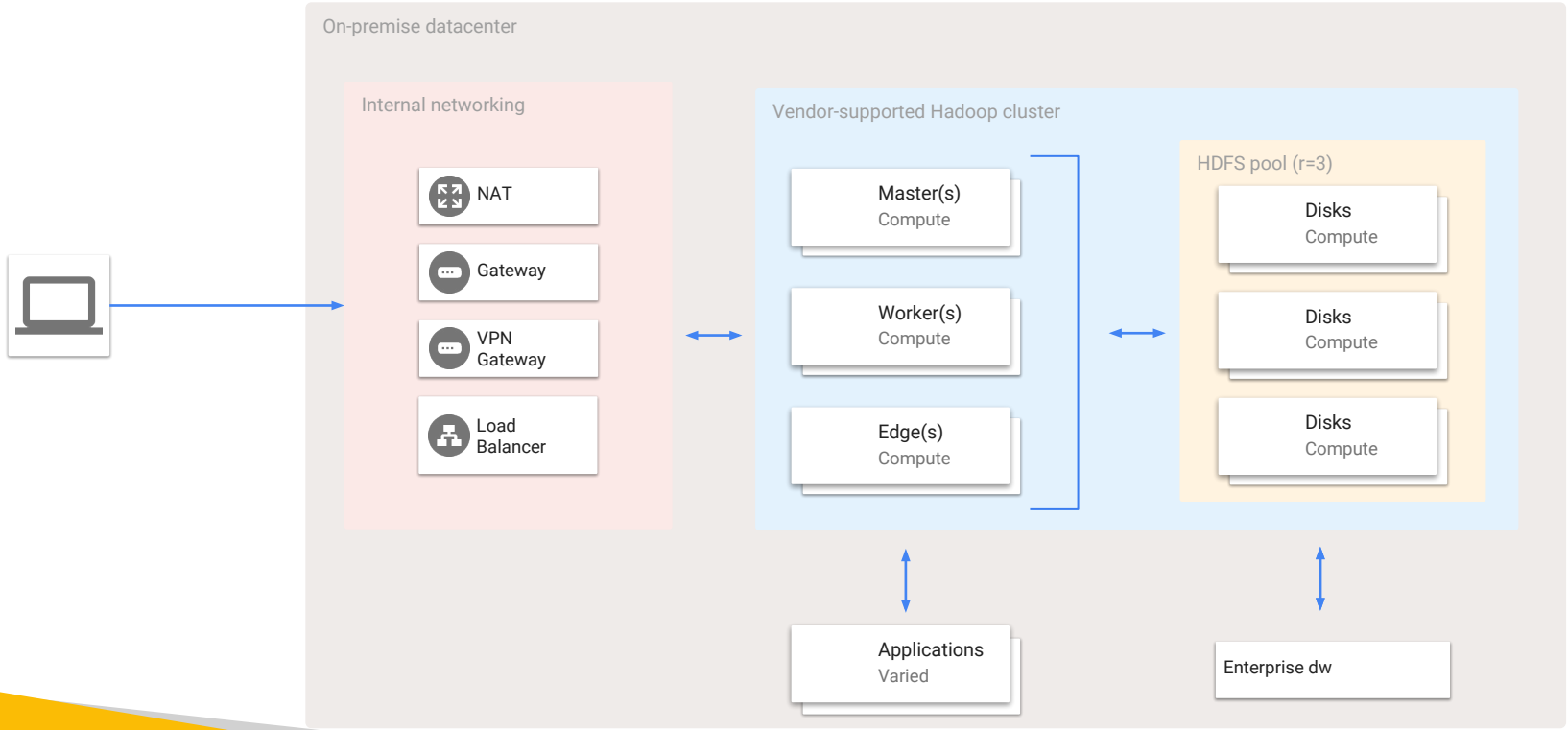
Clusters are set it and forget it.



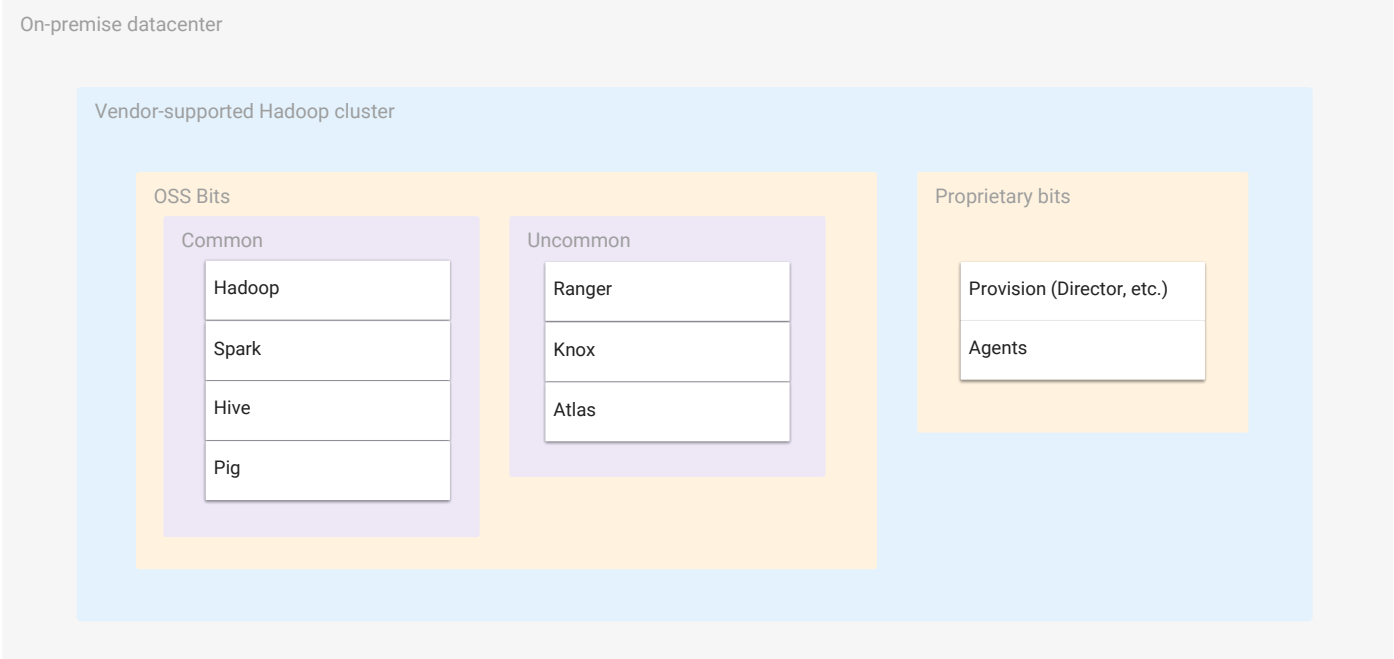
Cloud Dataproc - **under the hood**



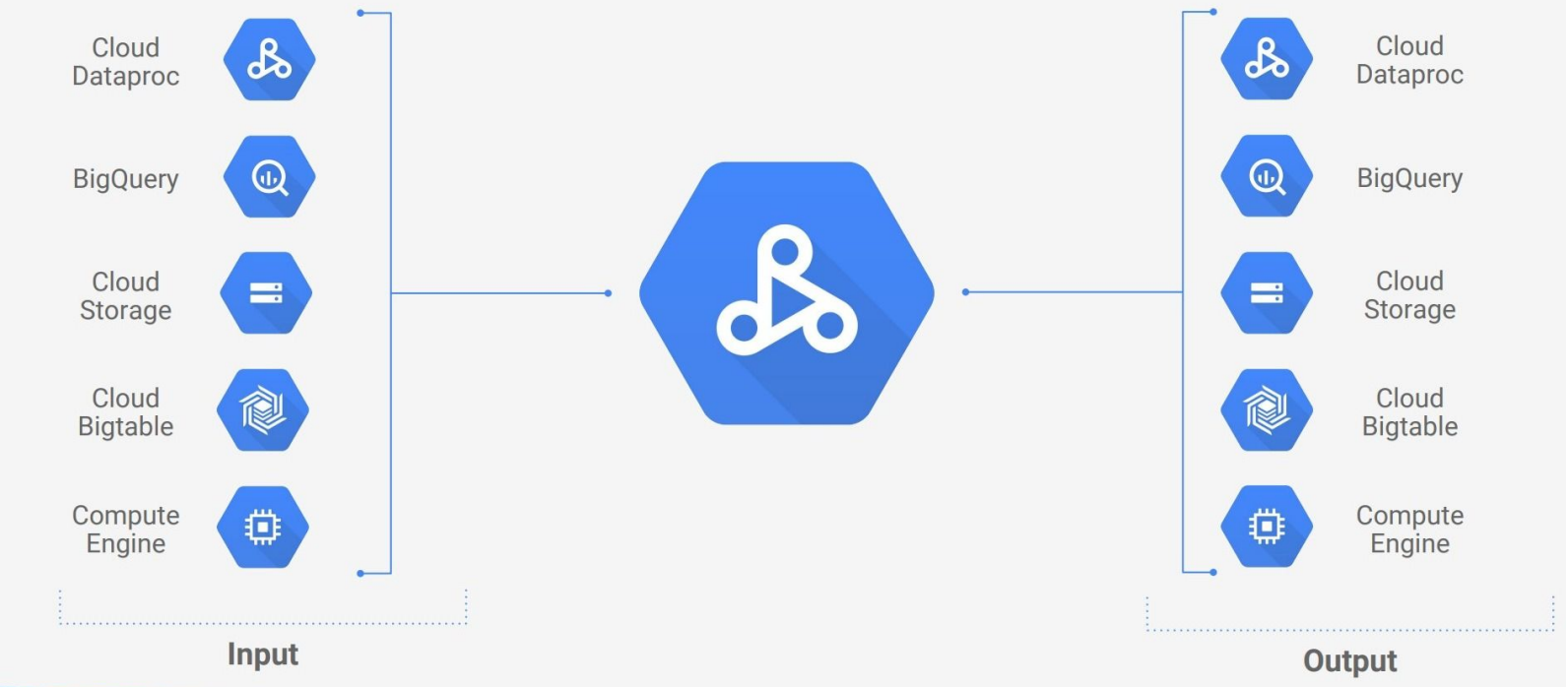
A typical on-premise cluster - **hardware**



A typical on-premise cluster - **software**



Connecting OSS to Cloud Platform



Notice services that are not listed here (Cloud PubSub | Cloud Dataflow)

Google Cloud

Mapping to Dataproc

On premise	GCP
Crunch	Apache Beam
Flume	Cloud Dataflow
HBase	Cloud Bigtable
HCatalog	**Cloud Dataproc + Cloud SQL**
Hive	**Cloud Dataproc**
Hue	**Cloud Dataproc,** Datalab
Impala	BigQuery, Cloud Dataproc
Mahout	**Cloud Dataproc**
Oozie	**Cloud Dataproc**
Parquet, Avro	**Cloud Dataproc**
Pig	**Cloud Dataproc**
Sentry, Ranger, Knox	Cloud IAM**
Spark	**Cloud Dataproc**
Sqoop	**Cloud Dataproc**
Tez	**Cloud Dataproc**
ZooKeeper	**Cloud Dataproc**

Demo: Running a Data processing job on Dataproc

Migrating to Cloud Dataproc

Lift and shift work to Cloud Dataproc

Copy data to GCS

Copy your data to Google Cloud Storage (GCS) by installing the connector or by copying manually

Update file prefix

Update the file location prefix in your scripts from `hdfs:// to gs://` to access your data in GCS

Use Cloud Dataproc

Create a Cloud Dataproc cluster and run your job on the cluster against the data you copied to GCS. Done

Moving to an ephemeral model

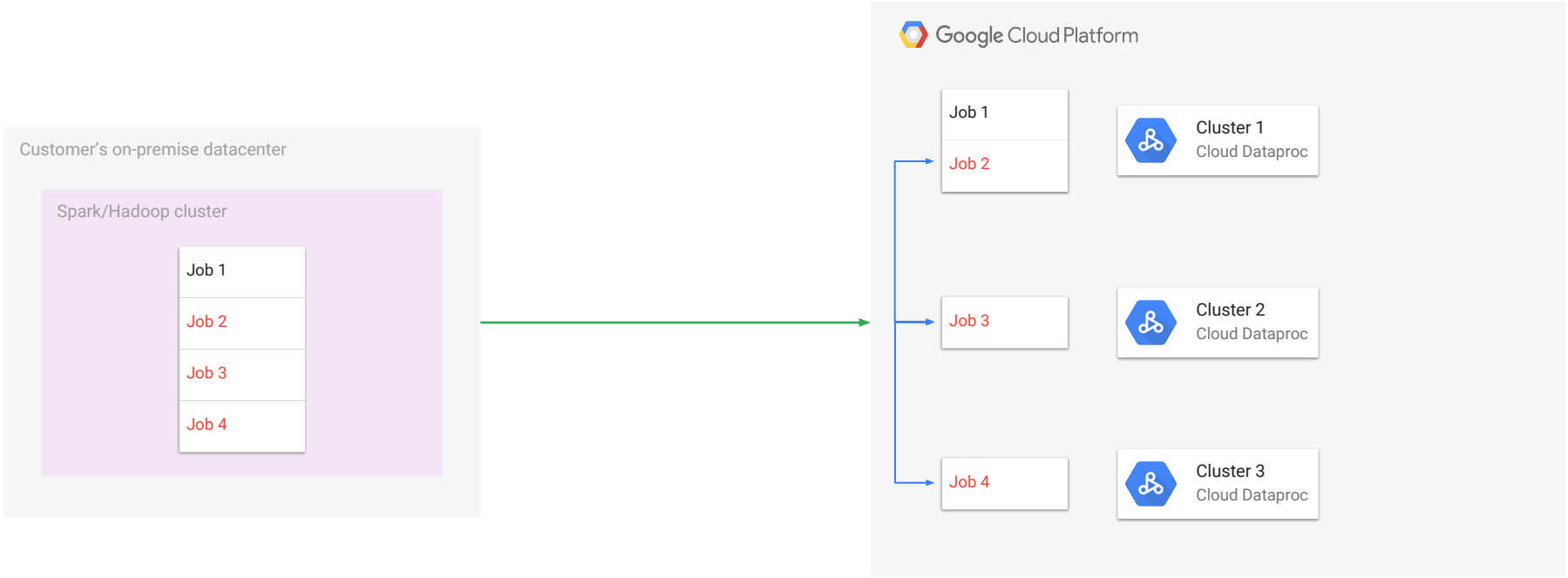
To get the most from Cloud Dataproc, customers need to move to an “ephemeral” model only using clusters when they need them.

This can be scary because a persistent cluster is comfortable.

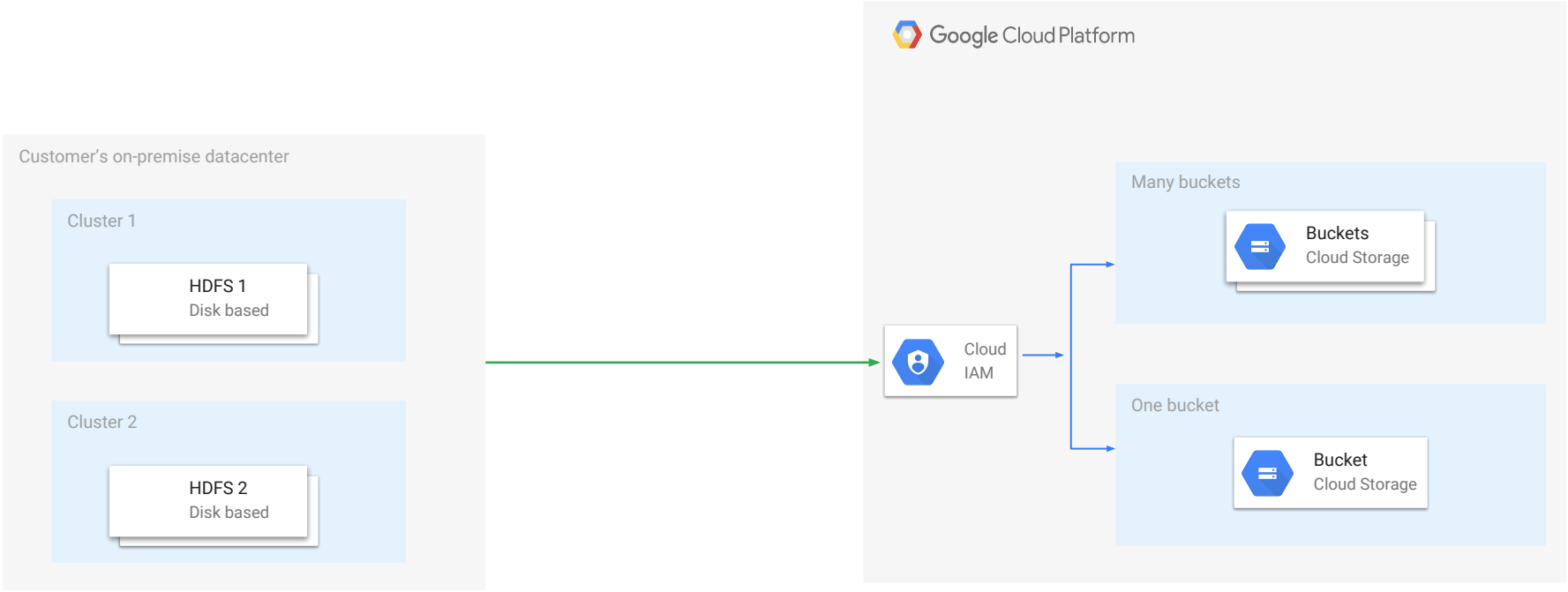
With GCS data persistence and fast boot of Cloud Dataproc, however, a persistent cluster is a waste of resources.

If a persistent cluster is needed, make it small. Clusters can be re-sized anytime.

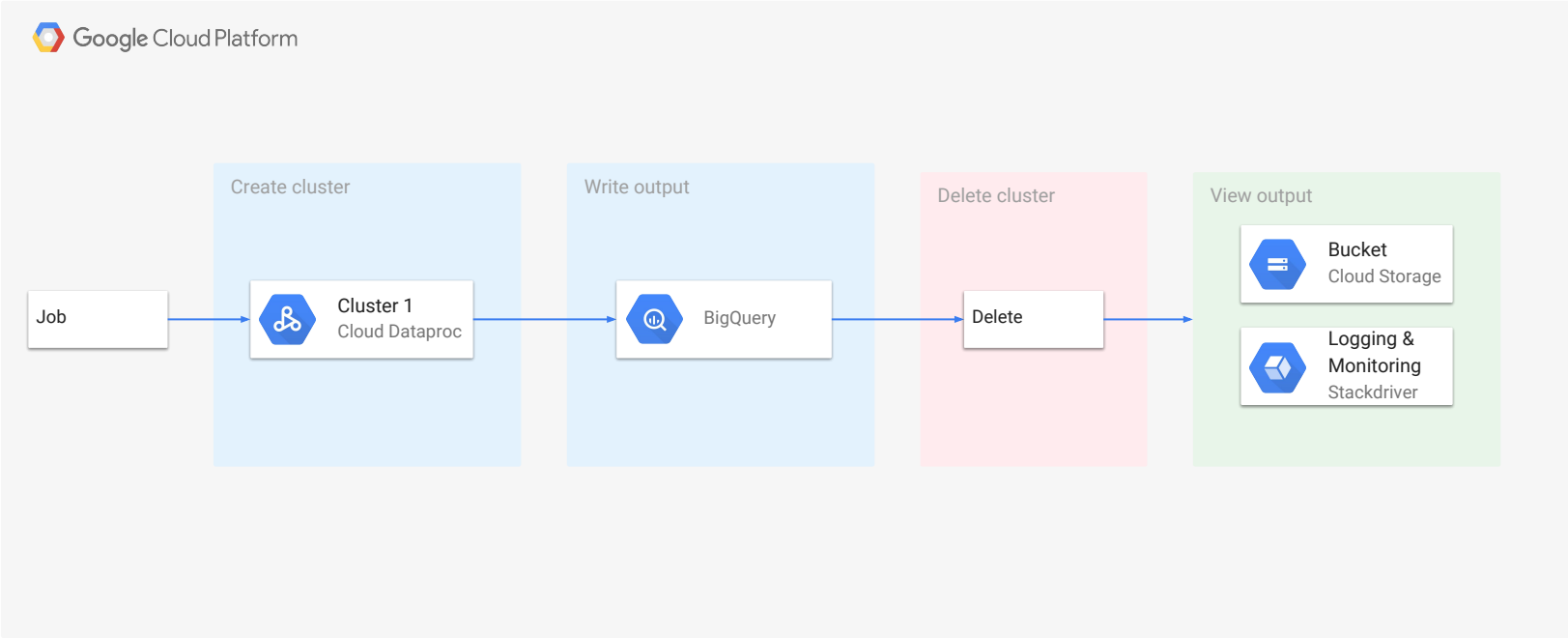
Recommendation - Split clusters and jobs



Recommendation - Use Google Cloud Storage

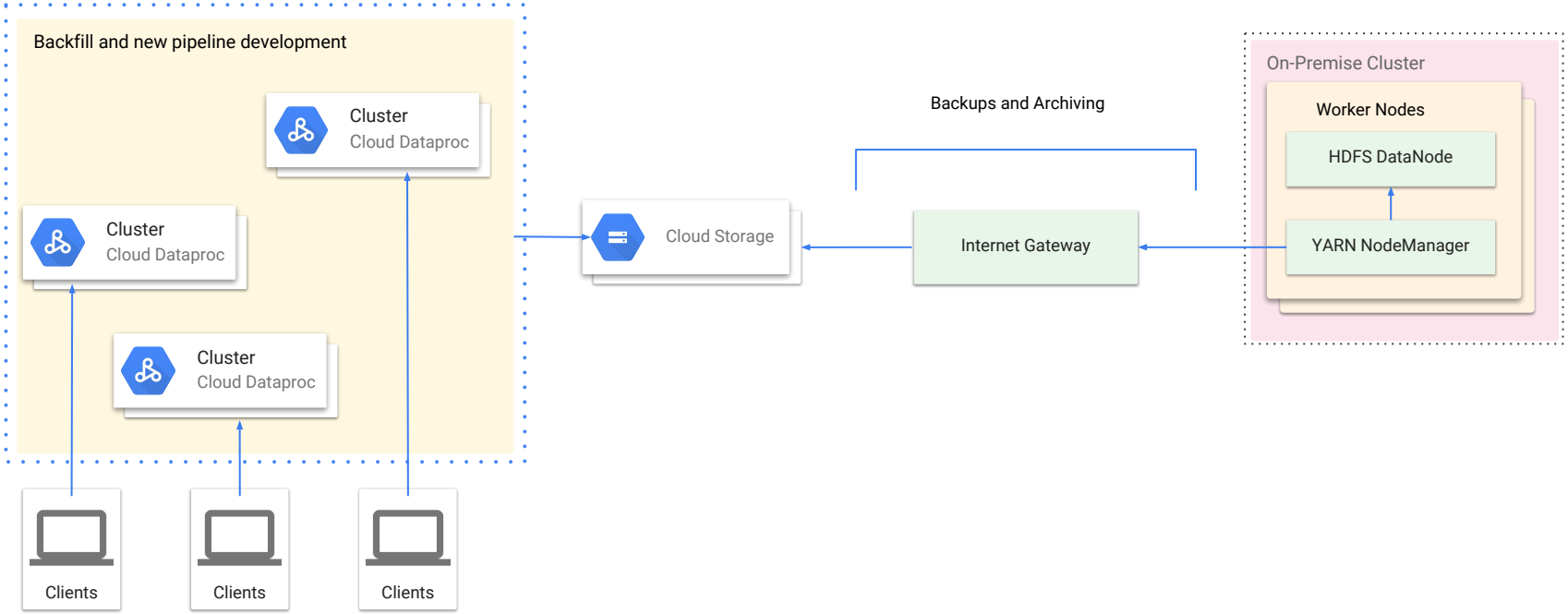


Recommendation - Create and delete often

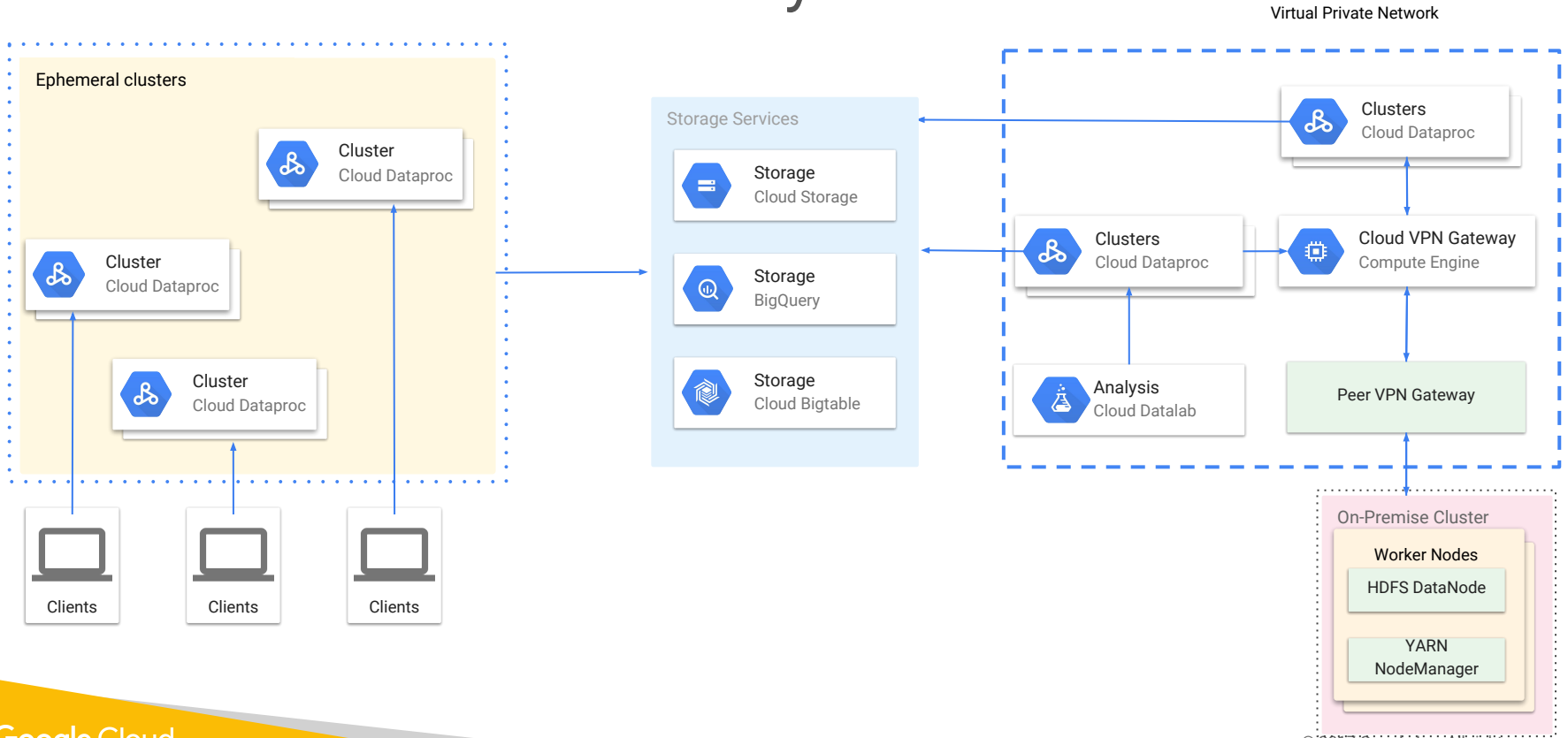


Architectures over the course of migration

Reference architecture - burst workloads in cloud

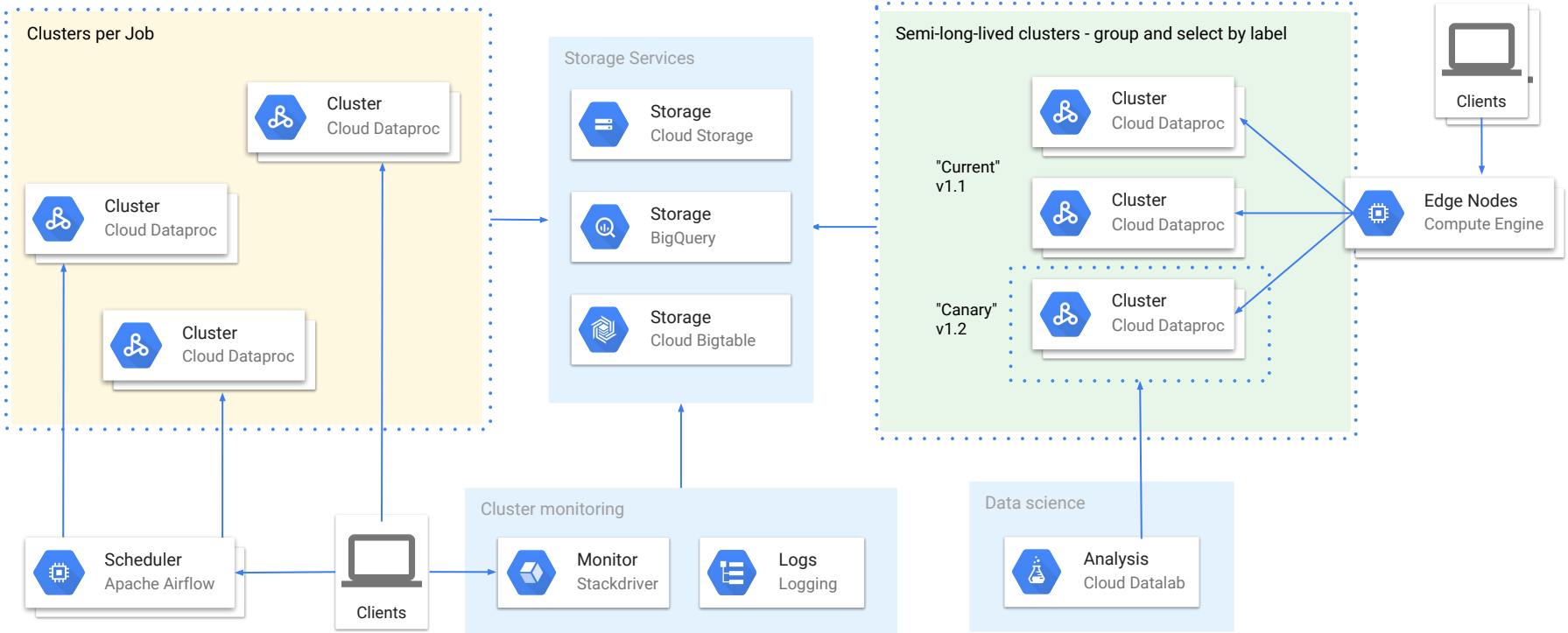


Reference architecture - Hybrid cloud



Google Cloud

Reference architecture - Cloud native



Section 1 Section 2 **Section 3** Section 4

Migrating Your Data Warehouse to BigQuery

Migrating Your Data Warehouse to BigQuery

Building blocks for migrating a data warehouse to BigQuery are as follows:

1. BigQuery Under the Hood
2. A Framework for Migrating Your EDW to BigQuery
3. BigQuery Best Practices
4. Utilities for Optimizing in BigQuery

BigQuery Fundamentals

The Business Case for BigQuery

There are primary motivators for clients to adopt BigQuery which presents a departure from the challenges of the current big data architecture

1. BigQuery is a fully managed, no-operations data warehouse. The concept of hardware is completely abstracted away from the user.

2. BigQuery eliminates the need to forecast storage and compute resources in advance. All the resources are allocated dynamically based on client usage.

3. BigQuery provides a unique ‘pay as you go’ model for the EDW, allowing clients to move away from a CAPEX-based model.

4. BigQuery charges separately for data storage and query processing enabling an optimal cost model, unlike solutions where processing capacity is allocated (and charged) as a function of allocated storage.

5. BigQuery enables extremely fast analytics on a petabyte scale through its unique architecture and capabilities.

BigQuery Under the Hood

BigQuery leverages technologies such as Borg, Colossus, Jupiter, and Dremel to enable lightning fast analytics

Dremel - The Execution Engine

- Turns each query into an execution tree that manages data reading and computation
- Allocates slots as needed for workload

Colossus - Distributed Storage

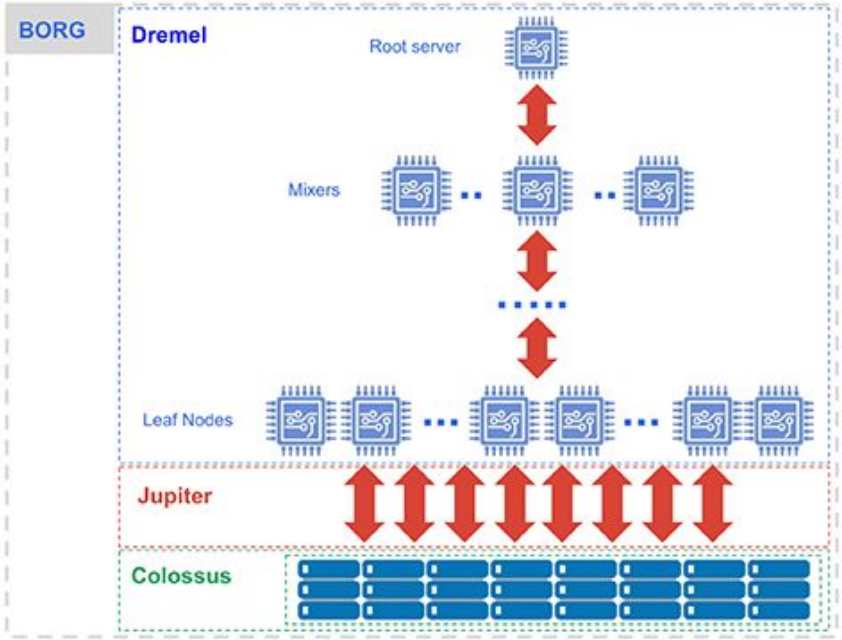
- Latest generation distributed file system
- One Colossus cluster can give every user thousands of dedicated disks at a time
- Innate compression algorithm stores large data optimally

Borg - Compute

- Large-scale cluster management system.
- Assigns server resources to jobs
- Routes around data center clashes to ensure high availability

Jupiter - Network

- Delivers 1 Petabit/sec of total bisection bandwidth



Interrupting Your Existing Data Schema

BigQuery's architectural design requires interrupting your existing data schema to leverage its fast analytical processing capabilities

BigQuery Design Features

Columnar Data Access	Data is stored in a columnar storage fashion which makes possible to achieve very high compression ratio and scan throughput
Distributed Processing	The underlying architecture forms a massively parallel distributed tree for pushing down a query to the tree and then aggregating the results from the leaves at a blazingly fast speed.

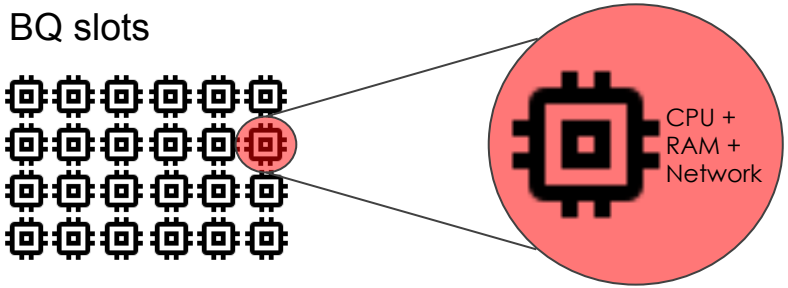
Resulting Impacts for Migration

1. **Transformation of dimensional data**
2. **Adoption of data flattening techniques**
3. **Data partitioning to optimize data access**

BigQuery Slots

What are BigQuery slots?

Under the hood, analytics throughput is measured in BigQuery slots. A BigQuery slot is a unit of computational capacity required to execute SQL queries. BigQuery automatically calculates how many slots are required by each query, depending on query size and complexity.



Demo

Let’s explore data in BigQuery

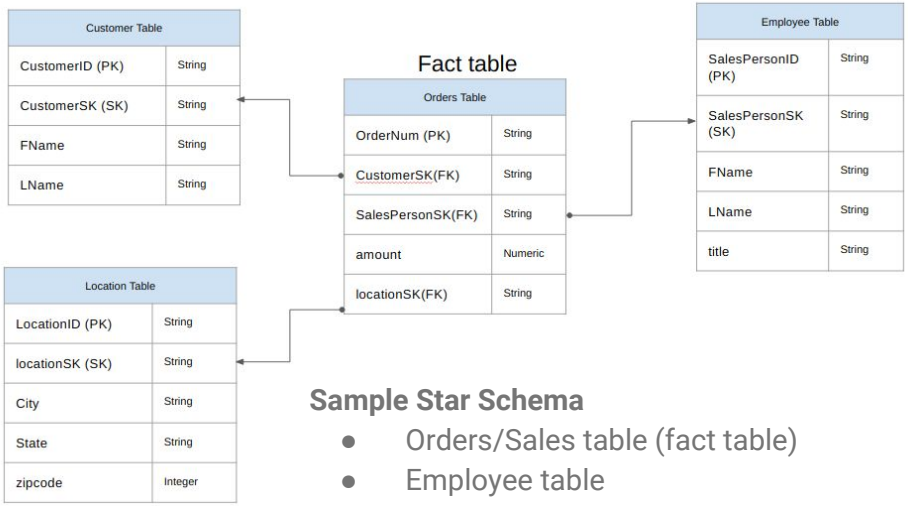
Data Modeling for BigQuery

A few reminders...

- BigQuery does not use or support indexes.
- For each query, BigQuery executes a full-column scan.
- BigQuery performance and query costs are based on the amount of data scanned.
- Storage is cheap!
- BigQuery supports nested and repeated columns.

Transforming the simple star schema

Sample Star Schema



Sample Star Schema

- Orders/Sales table (fact table)
- Employee table
- Location table
- Customer table

Resulting Subject Matter Tables

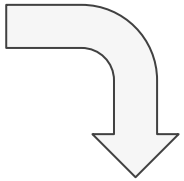
OrderNum	CustomerID	SalesPersonID	amount	Location
O-1	1234	12	234.22	18
O-2	4567	1	192.10	27
O-3		12	14.66	18
O-4	4567	4	182.00	26

SalesPersonID	FName	LName	title
1	Alex	Smith	Sales Associate
4	Lisa	Doe	Sales Associate
12	John	Doe	Sales Associate

CustomerID	FName	LNAME
1234	Amanda	Lee
4567	Matt	Ryan

Transforming the simple star schema

```
#standardSQL
    SELECT
        orders.ordernum,
        orders.customerID,
        customer.fname,
        customer.lname,
        orders.salespersonID,
        employee.fname,
        employee.lname,
        employee.title,
        orders.amount,
        orders.location,
        location.city,
        location.state,
        location.zipcode
    FROM orders
    LEFT OUTER JOIN customer
        ON customer.customerID = orders.customerID
    LEFT OUTER JOIN employee
        ON employee.salespersonID = orders.salespersonI
    LEFT OUTER JOIN location
        ON location.locationID = orders.locationID
```



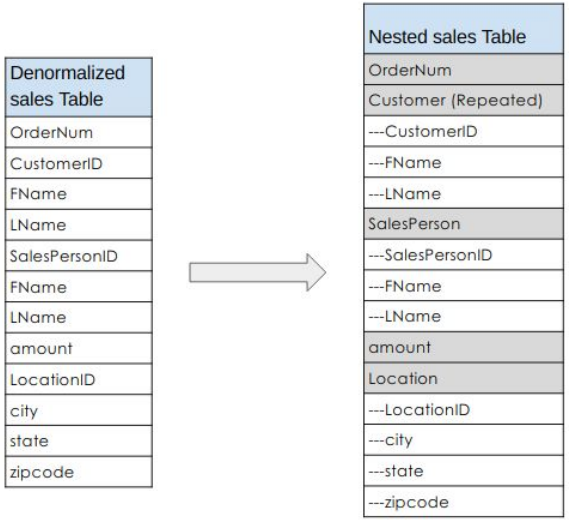
The Big Fat Table

OrderNum	CustomerID	FName	LName	SalesPersonID	FName	LName	amount	Location	city	state	zipcode
O-1	1234	Amanda	Lee	12	John	Doe	234.22	18	Bronx	NY	10452
O-2	4567	Matt	Ryan	1	Alex	Smith	192.10	27	Chicago	IL	60613
O-3				12	John	Doe	14.66	18	Bronx	NY	10452
O-4	4567	Matt	Ryan	4	Lisa	Doe	182.00	26	Mountain View	CA	90210

Leveraging Nested and Repeated Fields

Nested and repeated fields come handy in reducing data footprint within BigQuery, and thereby allowing for faster analytical processing

- **Allow for a reduced number of columns and/or rows during insert and select operations.**
- **Nested structures allow for the grouping of logical fields into a single field**
- **Users can perform `SELECT`, `INSERT`, `UPDATE`, `DELETE` operations on any individual fields**
- **Support one-to-many relationships with the remaining fields on the record.**



Our example in the nested and repeated format:

- CustomerID, FName, LName nested into a new field called **Customer**
- SalesPersonID, FName, LName nested into a new field called **Salesperson**
- LocationID, city,state,zipcode nested into a new field called **Location**

Guidelines for schema design

- Denormalize a dimension table that is larger than 1 GB, unless you see strong evidence that data manipulation, UPDATE and DELETE operation, costs outweigh the benefits of optimal queries.
- Keep a dimension table that is **smaller than 1 GB** normalized, unless the table rarely goes through UPDATE and DELETE operations.
- Take full advantage of nested and repeated fields in denormalized tables.

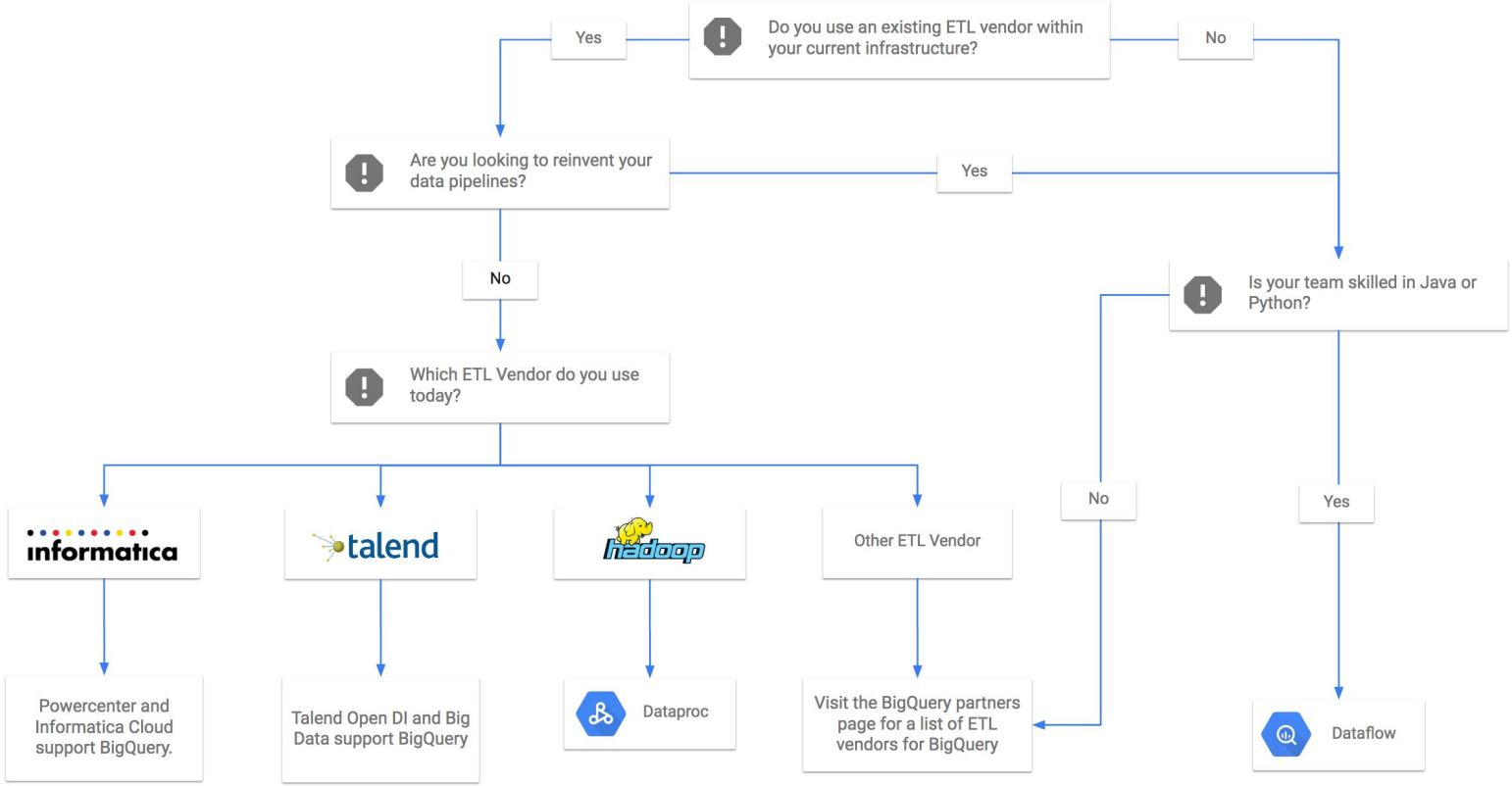
Considerations for Data Transformation

In many migration cases, data transformation should take place before loading data into BigQuery due to some data type constraints

Below are options to transform data before it lands in BigQuery. Consider the cases when each option is best

Where	Recommended for
Dataflow data pipelines that perform sets of actions	Creation of pipelines that incorporate: ● Complex transformations ● Multiple dependencies ● ¨Fan-in and fan-out¨ ● IT shops staffed with Java or Python developers
BigQuery console	Minor transformation tasks with the following characteristics: ● Managing joins and partitions ● Do not consume slots needed for business-critical queries

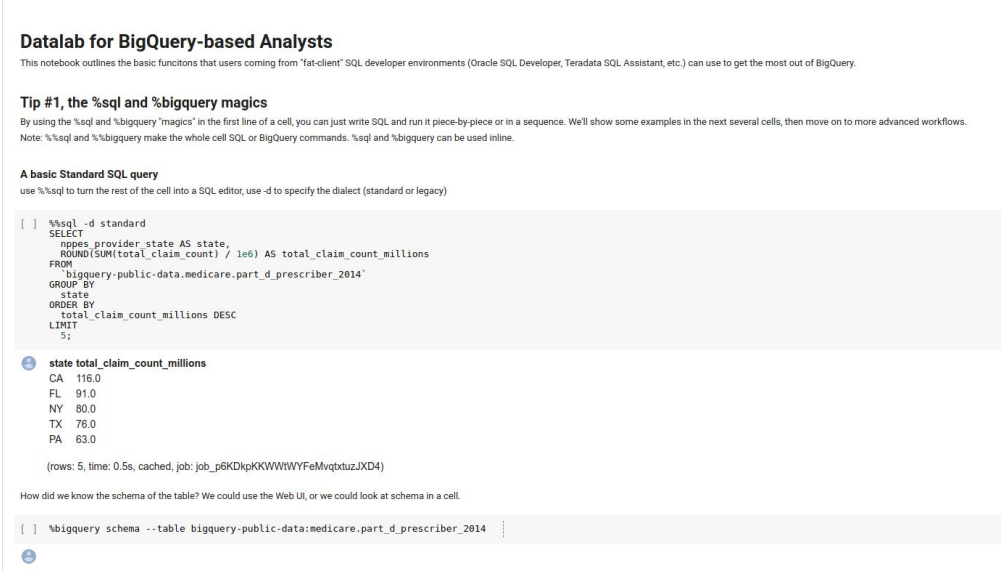
Knowing When to Use Partner Tooling



Rethinking for BigQuery

Replacing Multi-Statement Queries

- Use the WITH-clause to create virtual tables which can be referenced in subsequent SELECT statements.
- Datalab notebooks can replace parameterized multi-statement explore queries.



Datalab for BigQuery-based Analysts

This notebook outlines the basic funcitons that users coming from "fat-client" SQL developer environments (Oracle SQL Developer, Teradata SQL Assistant, etc.) can use to get the most out of BigQuery.

Tip #1, the %sql and %bigquery magics

By using the %sql and %bigquery "magics" in the first line of a cell, you can just write SQL and run it piece-by-piece or in a sequence. We'll show some examples in the next several cells, then move on to more advanced workflows.
Note: %%sql and %%bigquery make the whole cell SQL or BigQuery commands. %sql and %bigquery can be used inline.

A basic Standard SQL query

use %%sql to turn the rest of the cell into a SQL editor, use -d to specify the dialect (standard or legacy)

```
[ ]  %%sql -d standard
     SELECT
       nppes_provider_state AS state,
       ROUND(SUM(total_claim_count) / 1e6) AS total_claim_count_millions
     FROM
       `bigquery-public-data.medicare.part_d_prescriber_2014`
     GROUP BY
       state
     ORDER BY
       total_claim_count_millions DESC
     LIMIT
       5;
```

state	total_claim_count_millions
CA	116.0
FL	91.0
NY	80.0
TX	76.0
PA	63.0

(rows: 5, time: 0.5s, cached, job: job_p6KDkpKKWWtWYFeMvqtxtuzJXD4)

How did we know the schema of the table? We could use the Web UI, or we could look at schema in a cell.

```
[ ]  %bigquery schema --table bigquery-public-data:medicare.part_d_prescriber_2014
```

Google Cloud

Partition Your Tables

- Table partitioning can optimize performance & cost by reducing the amount of data to be scanned via partition elimination
- Use partition decorators to load into the correct partition based on the as-of-date of the data
- Utilize Dataflow's support for Dynamic Destinations to dynamically route data into the correct partition
- This should be solved in the near future with custom date-field partitioning (currently beta) and clustering (H1 2018)

Slowly Changing Dimensions

- Prefer SCD Type 2 Changes
  - Rely on transaction date instead of start and end date to minimize the need to go back and retroactively
  - Flatten records in a view or materialized table to provide only the latest records

Google Cloud

Real-time Updates with Denormalization #1

Handle real-time updates using separate table for the current day's updates and mask the underlying historical table through a view. Periodically materialize the updates into the historical table.

customer_id	customer_first_name	customer_last_name	...
1	Jane	Bar	

Pending Customer Updates

order_id	customer_id	customer_first_name	customer_last_name	...
42	1	Jane	Foo	

Denormalized Fact Table

```
#standardSQL
SELECT
  f.order_id as order_id,
  f.customer_id as customer_id,
  IFNULL(
    u.customer_first_name,
    f.customer_first_name) as customer_first_name,
  IFNULL(
    u.customer_last_name,
    f.customer_last_name) as customer_last_name
FROM fact_table f
LEFT OUTER JOIN pending_customer_updates u
ON f.customer_id = u.customer_id
```

Reporting View

Google Cloud

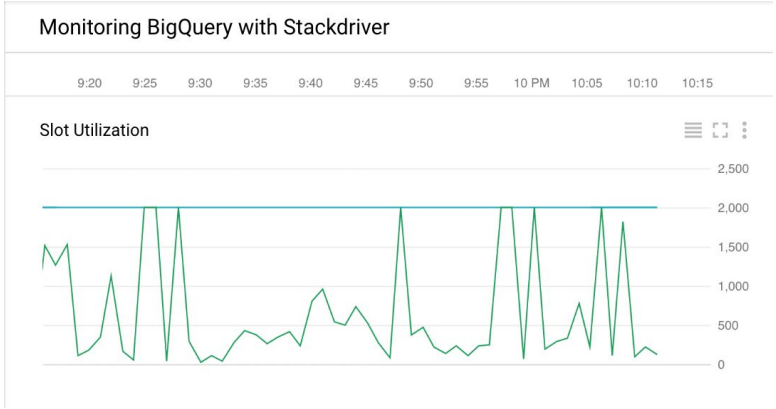
Real-time Updates with Denormalization #2

- Use a combination Dataflow & Bigtable to handle the denormalization
  - Bigtable holds the current transactional state of every entity
  - On a Type 1 change to a record contained in the denormalization, lookup and re-emit all past records for the entity
  - Periodically remove outdated records

Google Cloud

Utilities for Optimizing in BigQuery

BigQuery Monitoring

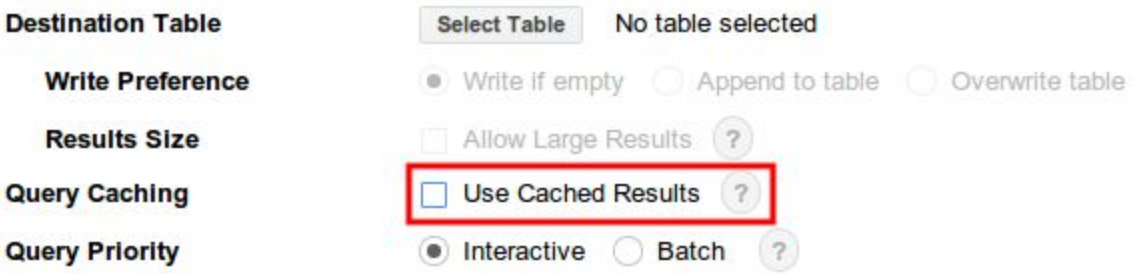


Available for all BigQuery customers whether they are using on-demand or flat-rate

Fully interactive GUI. Customers can create custom dashboards displaying up to 13 BigQuery metrics including:

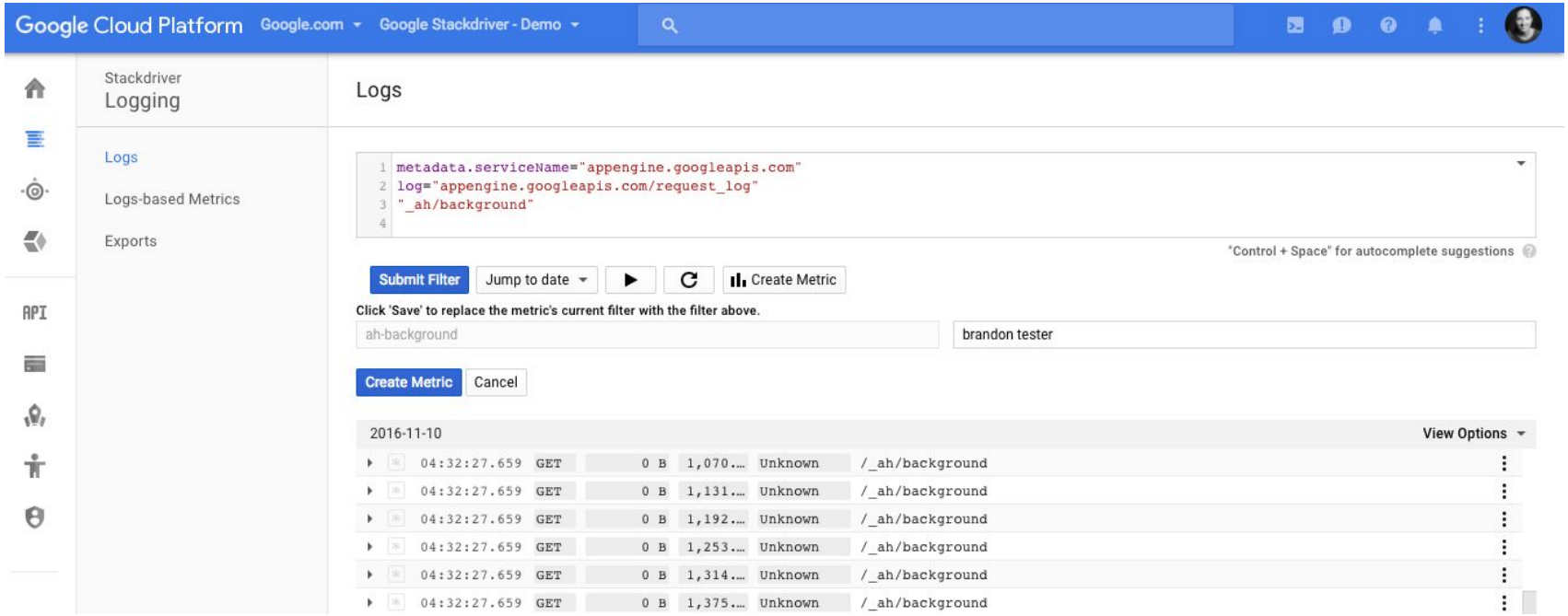
- Slots Available/Utilized
- Queries in Flight
- Uploaded Bytes (not shown)
- Stored Bytes (not shown)

Query Caching



- BigQuery writes all query results to a table.
- The table is either explicitly identified by the user (a destination table), or it is a temporary, cached results table
- Temporary, cached results tables are maintained per-user, per-project.
- The use of cached results does not incur any charges and the results are returned immediately.

Audit Logs



Section 1 Section 2 Section 3 **Section 4**

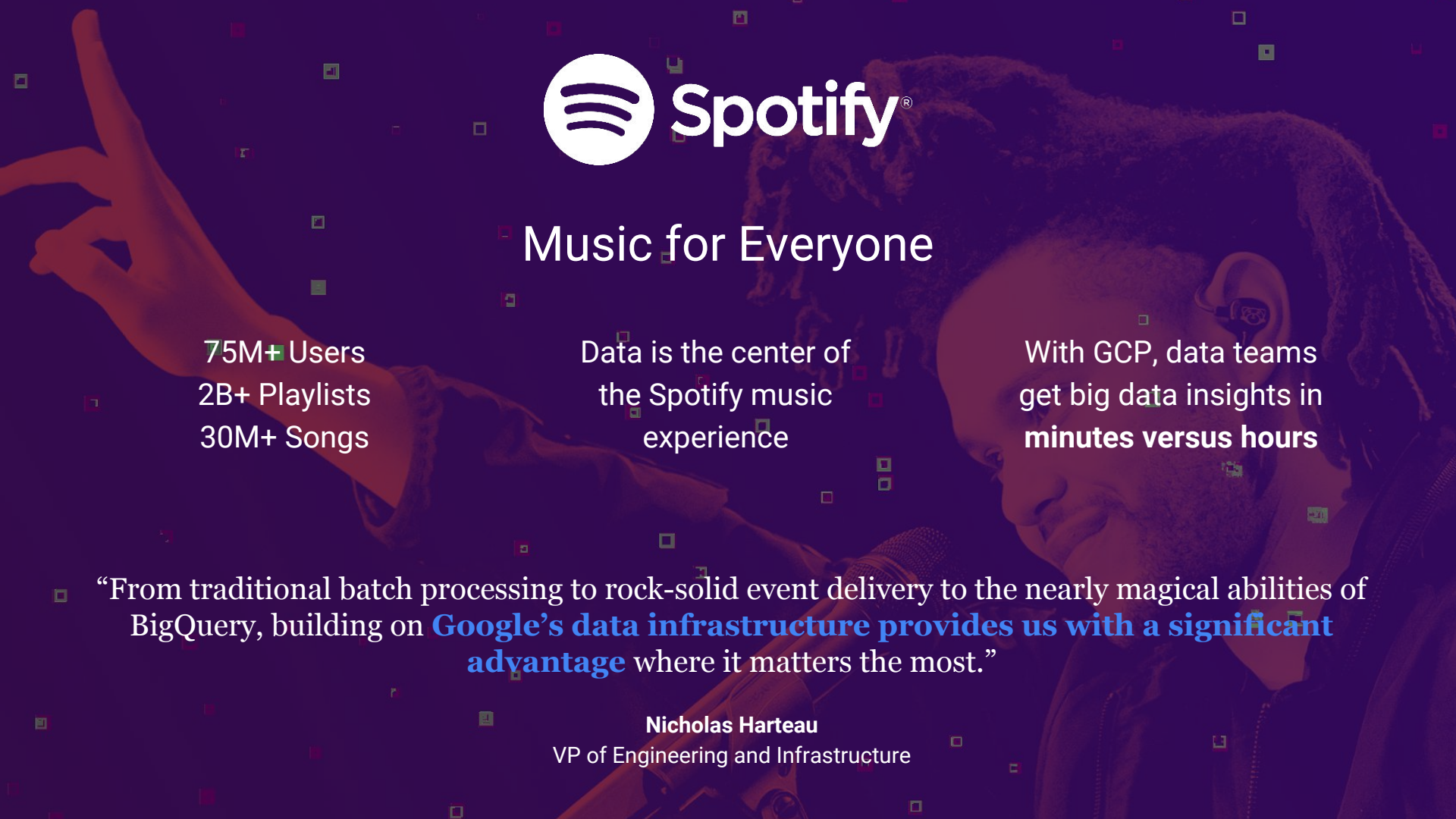
Designing Your Conceptual Architecture

Designing Your Conceptual Architecture

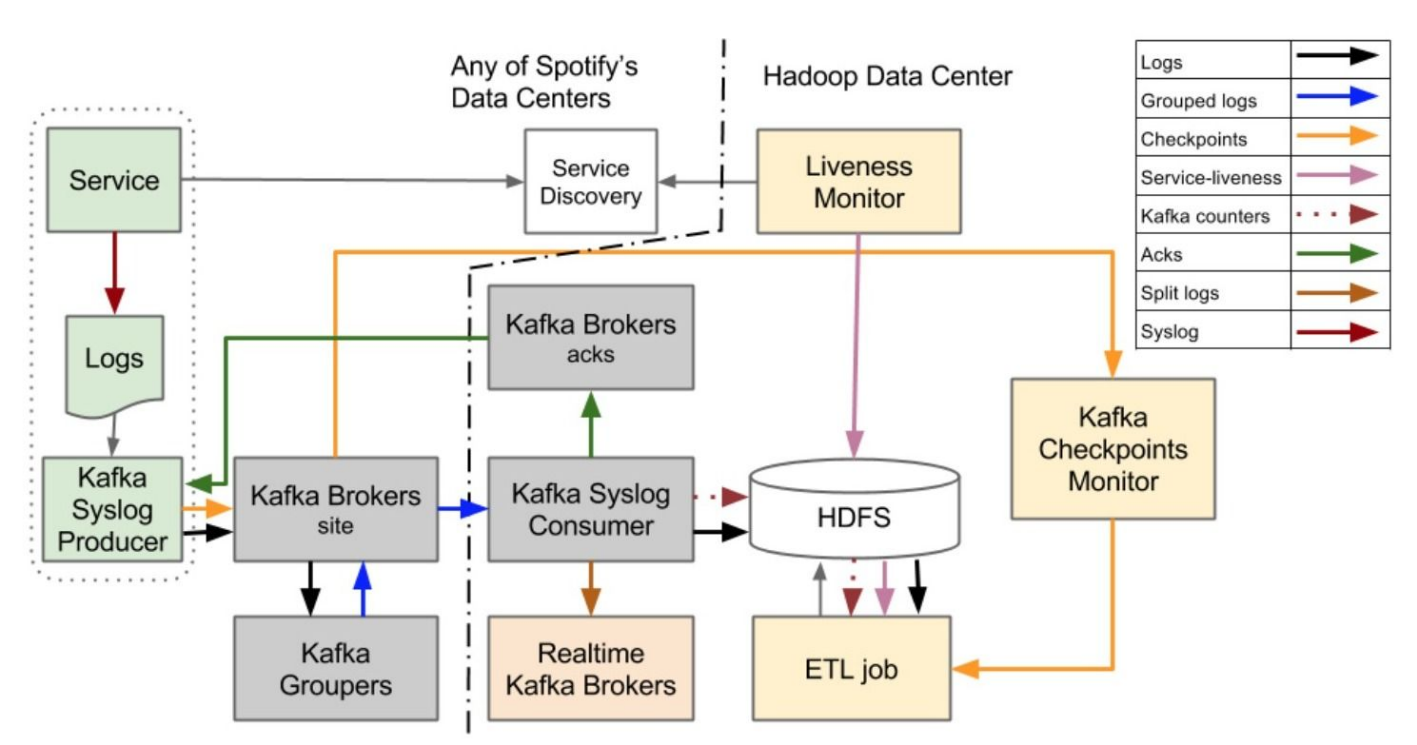
Key topics pertaining to this section include:

1. Walkthrough of a Customer Use Case
2. Enabling Migration to GCP
3. Pathways to Move Data to GCP

Walk Through of a Customer Use Case


Spotify®
Music for Everyone
75M+ Users
2B+ Playlists
30M+ Songs
Data is the center of the Spotify music experience
With GCP, data teams get big data insights in **minutes versus hours**
“From traditional batch processing to rock-solid event delivery to the nearly magical abilities of BigQuery, building on **Google’s data infrastructure provides us with a significant advantage** where it matters the most.”
Nicholas Harteau
VP of Engineering and Infrastructure

Spotify: Journey to GCP



Spotify: Journey to GCP

What happens when Spotify tests sending 2M events to Pub/Sub from one regional location

Figure 3. Number of successful requests per second to Pub/Sub from all data centers

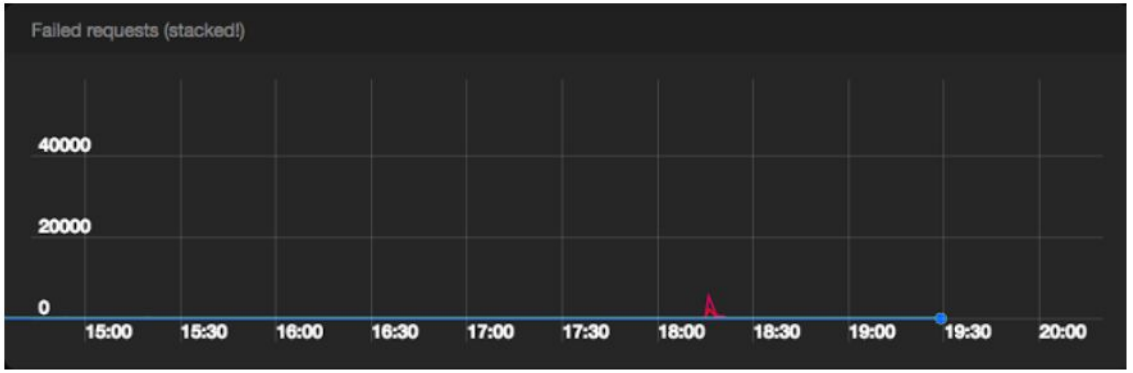
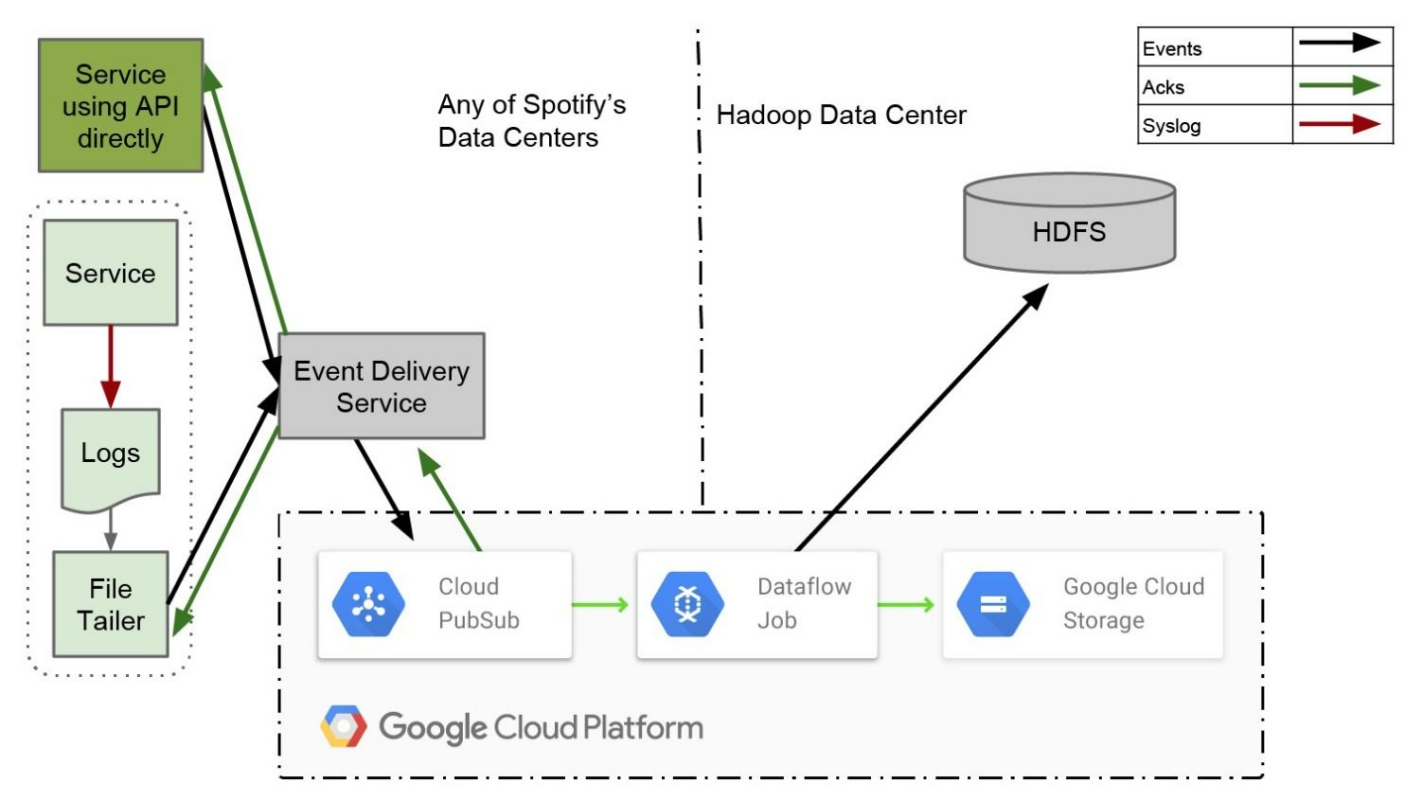


Figure 4. Number of failed requests per second to Pub/Sub from all data centers

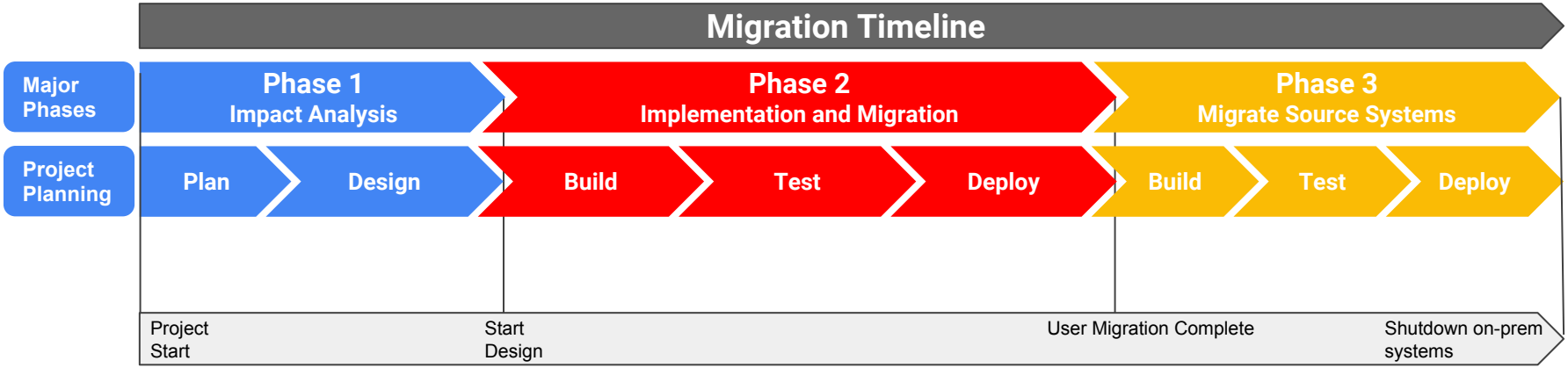
Spotify: Journey to GCP



Enabling Migration to GCP

Anatomy of Big Data Projects on GCP

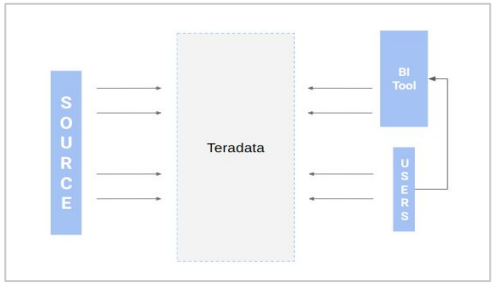
A multi-phase approach for migrating from on-prem systems to Google Cloud Platform

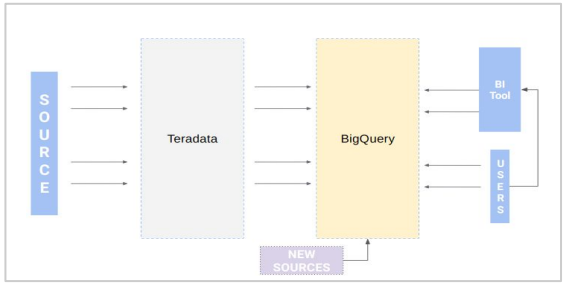


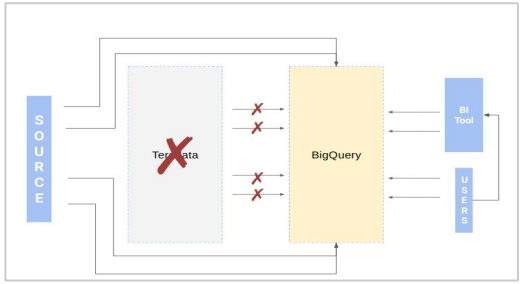
Defining the Migration Phases

In some cases, you will need to define the phases of migration for the specific client context and the related architecture for each stage

Sample Teradata to BigQuery Migration







Phase 1 - Impact Analysis

- Requirement gathering
- Users impact assessment
- Application inventory assessment
- Data sizing and inventory assessment

- **Key Outcome: Understanding of migration workload**

Phase 2 - Implementation and Migration

- Design ETL pipelines to move data from on-prem to GCP
- Build, test, and deploy pipelines
- Connect BI tools to GCP

- **Key Outcome: Data in GCP storage location mirrors data in on-prem systems**

Phase 3 - Migrating Source Systems

- Source systems configured to integrate with GCP
- Complete shutdown of on-prem system

- **Key Outcome: Migration of source systems with limited impact to users in production**

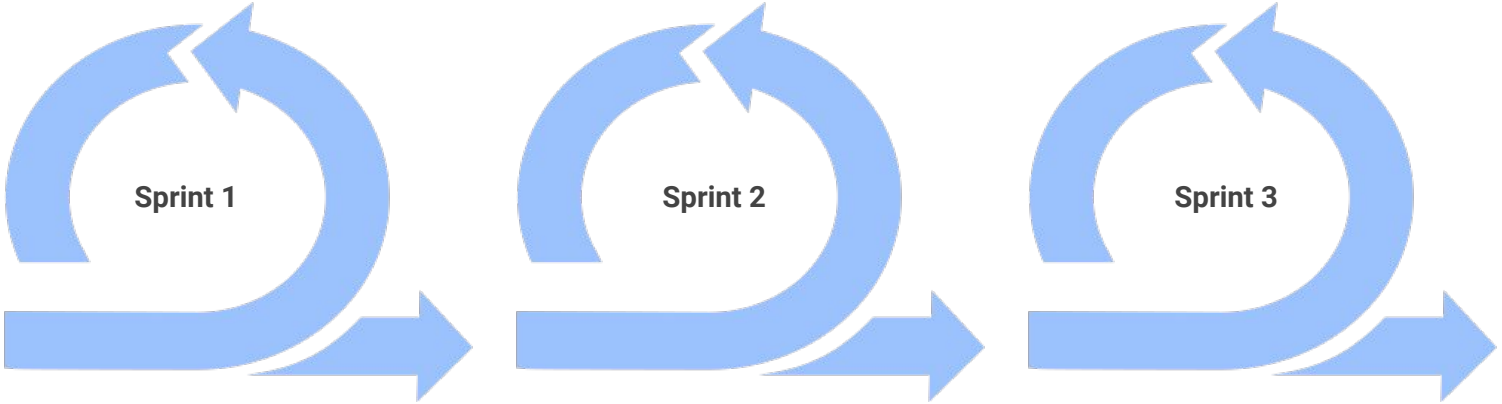
Determining the Migration Architecture

There are multiple pathways to move to GCP; therefore, determining the data migration architecture requires key considerations of the following components and their decision impacts

Migration Components	Decision Impacts
Schema Migration	Design approach to recreate required data structures on the cloud platform
Data Migration	Incorporation of utilities to facilitate seamless data movement from system to system
Data Transformation Requirements	Design of workflows to systematically shape data for insights (e.g: using Cloud Dataflow, Cloud Dataproc)
Real Time vs. Batch Workloads	Design of pipelines to facilitate transformation of streaming or scheduled data movement
OLTP vs. OLAP Requirements	Determination of GCP storage utilities to adopt in the migration architecture
Script Migration	Reengineering of scripts that facilitate data movement from source to consuming systems within the organization

Achieving Quick Wins to Build Momentum

Leverage sprints to achieve quick wins in enabling client migration to GCP



Quick Wins:

- Identify MVP
- Provision a subset of users
- Design smaller ETL pipeline

Quick Wins:

- Transfer MVP data to GCP storage option (e.g.: Cloud Storage, BigQuery)
- Implement simple ETL pipeline

Quick Wins

- Optimize MVP architecture
- Collect lessons learned

Reference Architectures by Use Case

Activity - 10 minutes
Migrating to GCP

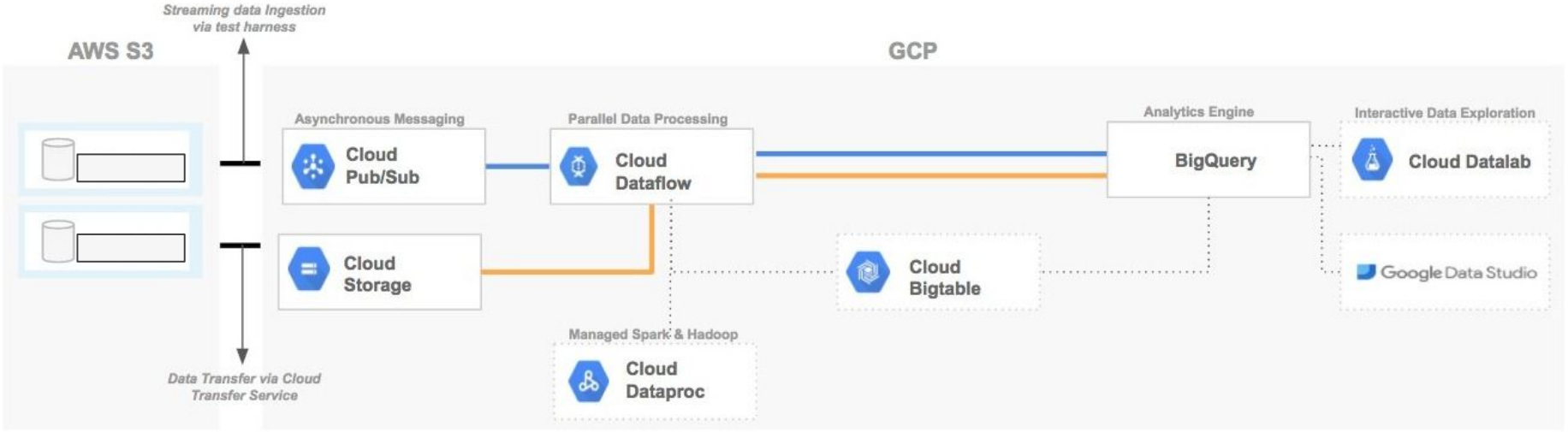
What approach can a client take to migrate data at the petabyte scale to GCP in the following scenario and why?

> XYZ, a tax and financial services corporation, wants to migrate data from its on-prem systems to GCP. To provision data within the organization, the company must ingest and process data in both batch and streaming workloads from multiple data sources.

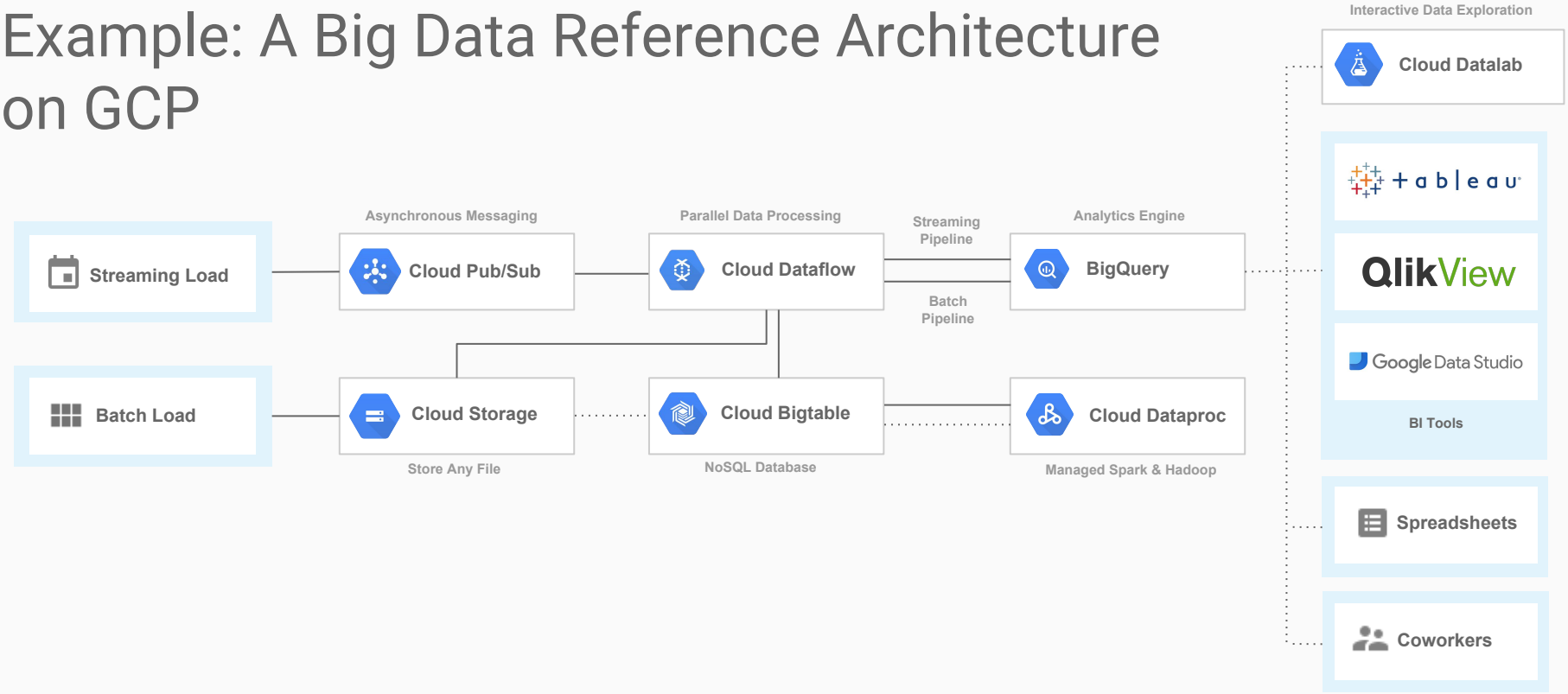
- Incoming data are customer transactions that will need to be transformed
- 30% of XYZ's incoming data is mutable, meaning they need to be updated (*in these cases, an update on a previous record has to occur*)

Activity: What architecture can XYZ adopt to move and process its data within GCP?
In your groups, whiteboard a potential architecture for XYZ

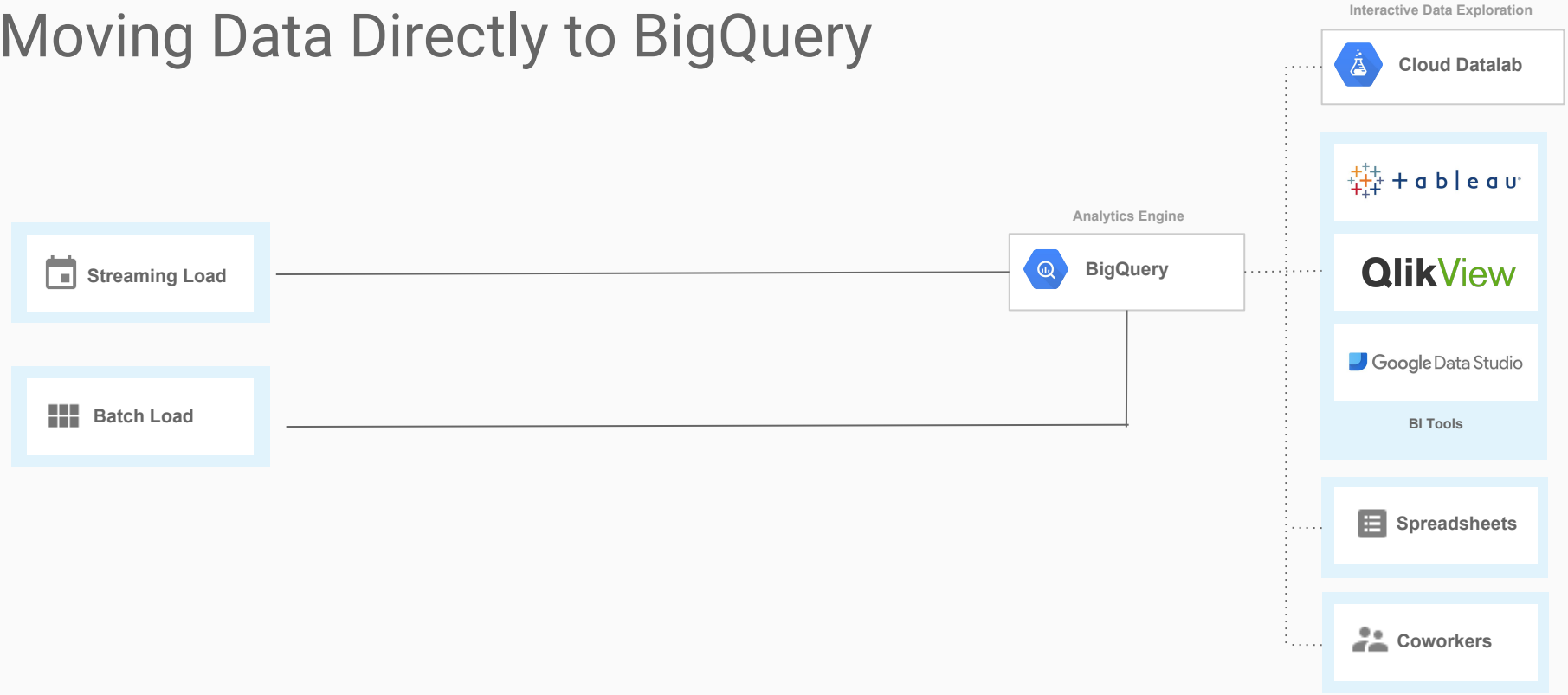
Activity - 10 minutes
Migrating to GCP - Sample Architecture



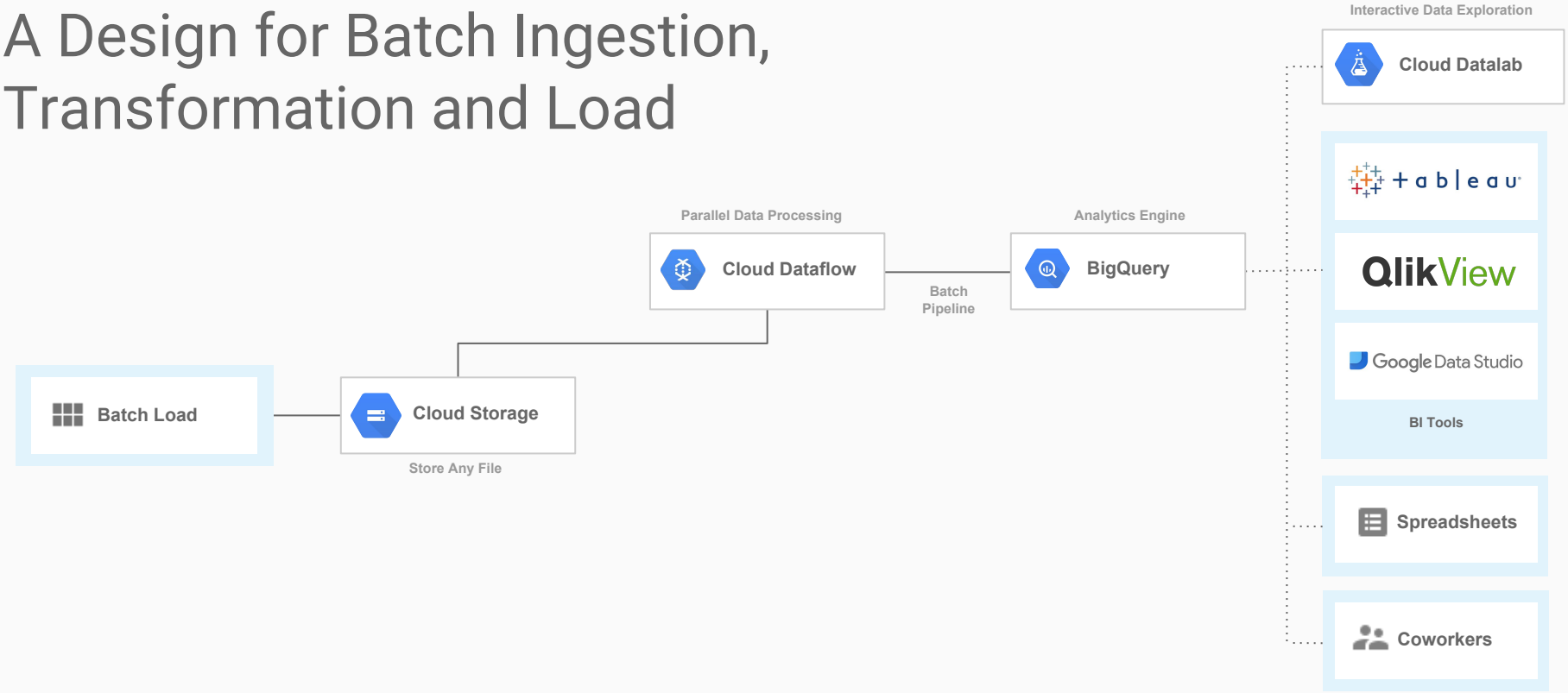
Example: A Big Data Reference Architecture on GCP



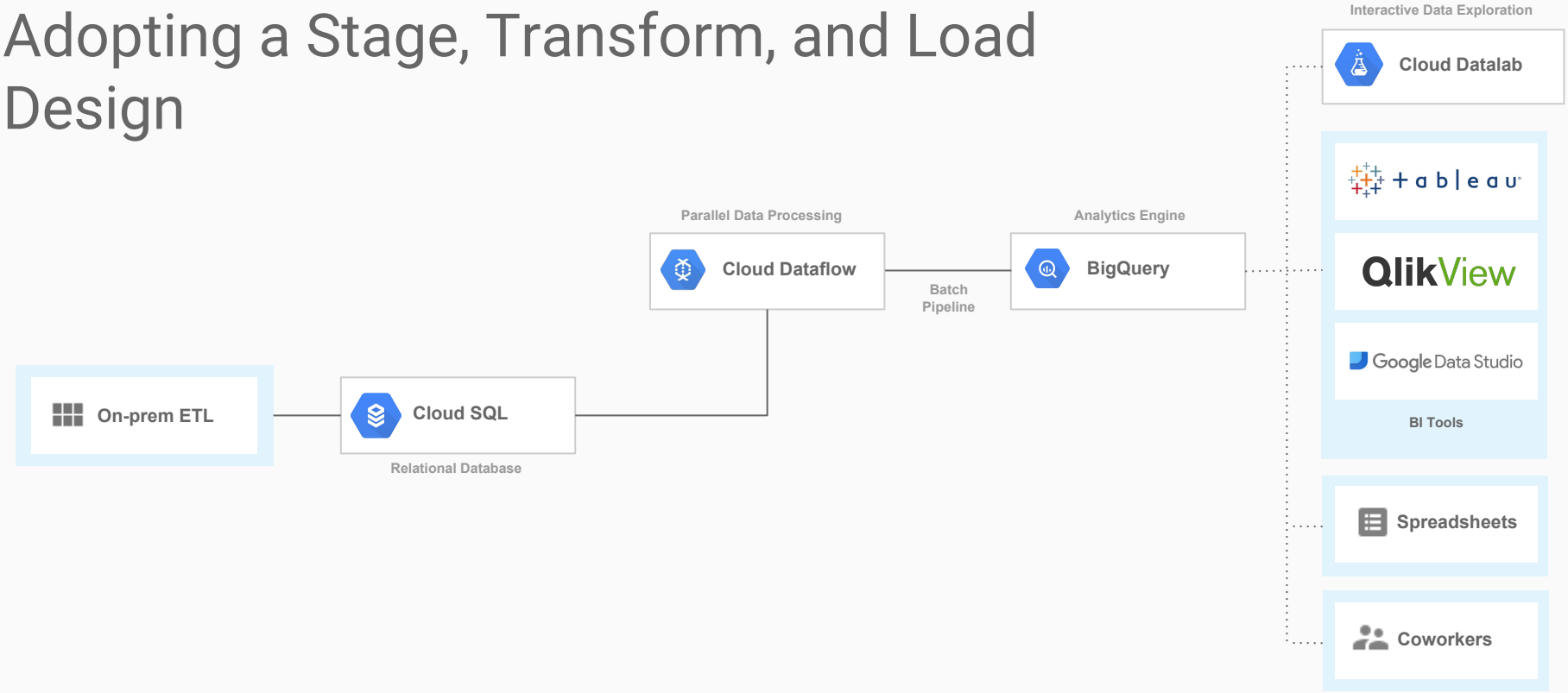
Moving Data Directly to BigQuery



A Design for Batch Ingestion, Transformation and Load

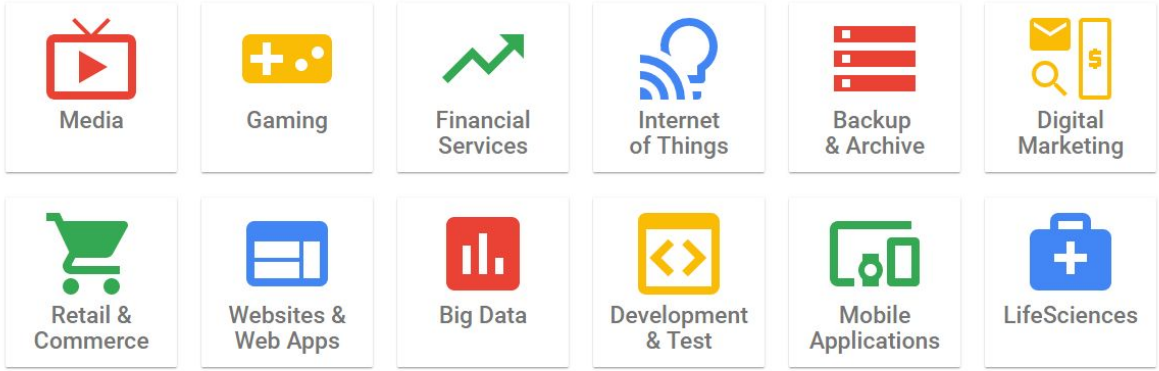


Adopting a Stage, Transform, and Load Design



More Industry Specific Architectures

Treehouse beta*



Link: gcp.solutions

Good Learning Resources

Google Cloud Documentation
A Framework for Migrating Your Data Warehouse to BigQuery
BigQuery for Data Warehouse Practitioners
Data Science on GCP by Valliappa Lakshmanan

Thank You!