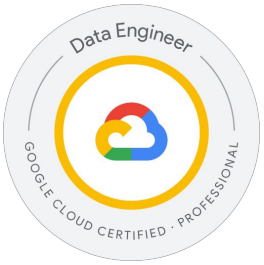
Google Cloud

Partner Certification Academy



Professional Data Engineer

The information in this presentation is classified:

Google confidential & proprietary

⚠ This presentation is shared with you under NDA.

- Do **not** record or take screenshots of this presentation.
- Do **not** share or otherwise distribute the information in this presentation with anyone **inside** or **outside** of your organization.

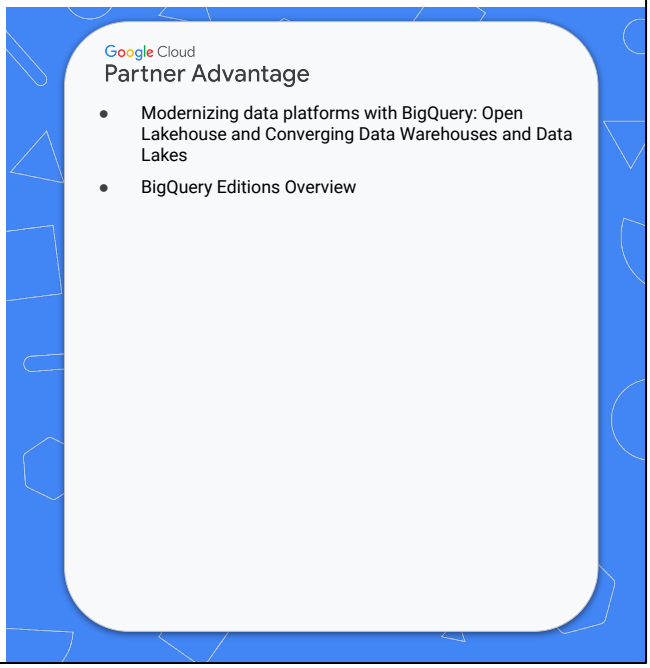
Thank you!

Google Cloud

Source Materials

Some of this program's content has been sourced from the following resources:

- Partner Advantage



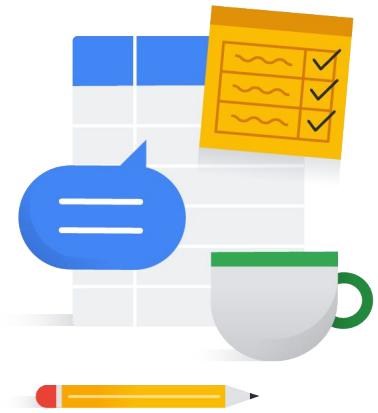
Google Cloud

Session logistics

- When you have a question, please:
  - Click the Raise hand button in Google Meet.
  - Or add your question to the Q&A section of Google Meet.
  - Please note that answers may be deferred until the end of the session.
- These slides are available in the Student Lecture section of your Qwiklabs classroom.
- The session is **not recorded.**
- Google Meet does not have persistent chat.
  - If you get disconnected, you will lose the chat history.
  - Please copy any important URLs to a local text file as they appear in the chat.

Program issues or concerns?

- Problems with **accessing** Cloud Skills Boost for Partners
  - partner-training@google.com
- Problems with **a lab** (locked out, etc.)
  - support@qwiklabs.com
- Problems with accessing Partner Advantage
  - https://support.google.com/googlecloud/topic/9198654



- Problems with accessing **Cloud Skills Boost for Partners**
  - partner-training@google.com
- Problems with a **lab** (locked out, etc.)
  - support@qwiklab.com
- Problems with accessing **Partner Advantage**
  - https://support.google.com/googlecloud/topic/9198654

This Week's Recommended Activities

1. Continue to review the exam guide
2. Labs and Quests for this week:
   a. **Course:** Serverless Data Processing with Dataflow: Foundations
   b. **Course:** Serverless Data Processing with Dataflow: Develop Pipelines
3. Review the recommended materials shared during this workshop
4. Labs and Quests for next week: (or review the content):
   a. **Course:** Serverless Data Processing with Dataflow: Operations
   b. **Quest:** Perform Foundational Data, ML and AI Tasks - Skill Badge

Week 1
Course: Google Cloud Big Data and Machine Learning Fundamentals
https://partner.cloudskillsboost.google/course_templates/3

Quest: Create and Manage Cloud Resources (this is an introductory quest) - Skill Badge
https://partner.cloudskillsboost.google/quests/120

Course: Modernizing Data Lakes & Data Warehouses with Google Cloud
https://partner.cloudskillsboost.google/course_templates/54

Week 2
Course: Building Batch Data Pipelines on Google Cloud
https://partner.cloudskillsboost.google/course_templates/53

Week 3
Course: Building Resilient Streaming Analytics Systems on GCP
https://partner.cloudskillsboost.google/course_templates/52

Course: Smart Analytics, Machine Learning, and AI on GCP
https://partner.cloudskillsboost.google/course_templates/55

Week 4
Course: Serverless Data Processing with Dataflow: Foundations

https://partner.cloudskillsboost.google/course_templates/218

Course: Serverless Data Processing with Dataflow: Develop Pipelines
https://partner.cloudskillsboost.google/course_templates/229

Week 5
Course: Serverless Data Processing with Dataflow: Operations
https://partner.cloudskillsboost.google/course_templates/264

Quest: Perform Foundational Data, ML and AI Tasks - Skill Badge
https://partner.cloudskillsboost.google/quests/117

Week 6
Lab: Optimizing BigQuery for Cost and Performance v1.5
https://partner.cloudskillsboost.google/focuses/18091?parent=catalog

Quest: Build and Optimize Data Warehouses with BigQuery - Skill Badge
https://partner.cloudskillsboost.google/quests/147

Lab: ETL Processing on Google Cloud Using Dataflow and BigQuery
https://partner.cloudskillsboost.google/focuses/11581?parent=catalog

Quest: Engineer data in Google Cloud - Skill Badge
https://partner.cloudskillsboost.google/quests/132

Week 7

Course: Preparing for the Google Cloud Professional Data Engineer Exam
https://partner.cloudskillsboost.google/course_templates/72

Practice: Professional Data Engineer Sample Questions
https://cloud.google.com/certification/practice-exam/data-engineer

Module Agenda

01 Dataflow Streaming

02 Dataflow Templates

03 BigQuery Slots

04 Information Schemas

05 OMNI

06 BI Engine

07 Cost Optimization

08 Partitioning and Clustering

Dataflow Streaming

Using Pub/Sub with Dataflow

- Pub/Sub can be utilized within a Dataflow pipeline
  - PubSub.IO class can read or write to a Pub/Sub topic
- Allows for streaming data flows in real time
  - Pipeline is constantly being sent messages for processing
- Can set up windows to process groups of messages by time
- When using Pub/Sub, data sets are unbounded
  - They continue to grow over time
  - Aggregate operations must be run periodically

Also refer to:
https://cloud.google.com/dataflow/docs/concepts/beam-programming-model

Pub/Sub, like other Google Cloud services, is designed to work directly with other services, including Dataflow. The PubSub.IO class can read or write directly to a Pub/Sub topic.

Pub/Sub can be used to stream data to Dataflow in real time. It can continuously send messages for processing.

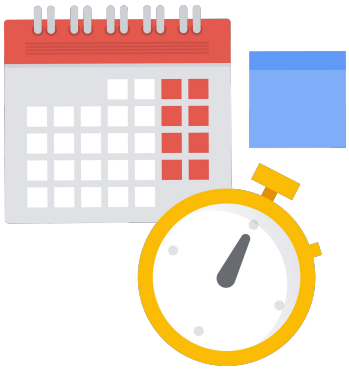
Dataflow can set up windows to process groups of streamed data based on message time.

Pub/Sub sends data to Dataflow that is unbounded.

The data will continue to grow over time, aggregate operations must periodically be run.

Event Time vs. Process Time

- Event time is when something actually occurred
  - The time an order is placed for example
  - This is when the message is published to Pub/Sub
- Process time is when the system observes the event
  - When the subscriber receives the Pub/Sub message
- Obviously, process time is always after event time
  - Usually, this is a short period of time
  - Sometimes, a system problem will delay the process time
- The difference between event and process times can vary significantly
  - Event occurs in a mobile application when the user is on an airplane



Also refer to:
https://cloud.google.com/dataflow/docs/concepts/beam-programming-model

Two types of time we need to understand with Pub/Sub and Dataflow is event time vs. process time.

Event time is when something actually occurred. For example, the time an order is placed. This is when the message is published to Pub/Sub.

Process time is when the system observes the event. For example when a subscriber receives a Pub/Sub message. This would be considered the event time.

In most cases, process time is always earlier than event time, but usually this is a short timeframe. Sometimes however, a system

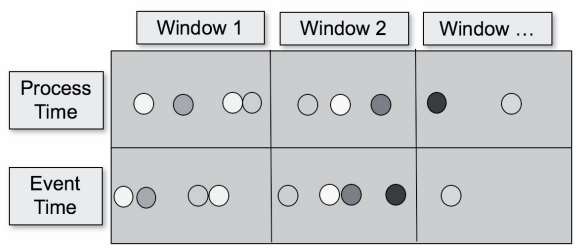
problem can delay the process time.

The difference between event and process time can be significant.

For example, if I am building a mobile game application and tracking user movements and somebody is running the game on their mobile phone on a plane, the time when those events are seen may be hours later.

Windowing

- Groups data into chunks that aggregate functions are run against
  - Often based on time, but can also be based on session
  - Time-based windows can use either event or process time
- Sometimes the process time will fall into a different window than the event



Also refer to:
https://cloud.google.com/dataflow/docs/concepts/beam-programming-model

With windowing, we group data into chunks, then run aggregate functions against those chunks.

In most cases, windowing is based on time, but it can also be based on session.

Time-based windows can be used either with event or processing time.

Sometimes the process time will fall into different windows for that event. In the example, we have the purple data in two windows due to a delay in the processing time.

What can we do about this?

Windowing in Dataflow

- Dataflow has built-in support for three types of windows
  - Shuffles arriving messages into the correct window based on event time
- Fixed time windows are the simplest
  - Each window is given an interval
- Sliding time windows have an interval and a period
  - The interval defines how long a window collects data for
  - The period defines how often a new window starts
- Session windows define windows on areas of concentrated data
  - Uses a key to combine data into groups
  - Like sessions in a web application

Also refer to:
https://cloud.google.com/dataflow/docs/concepts/beam-programming-model

Dataflow supports three different types of windows when processing aggregate data.

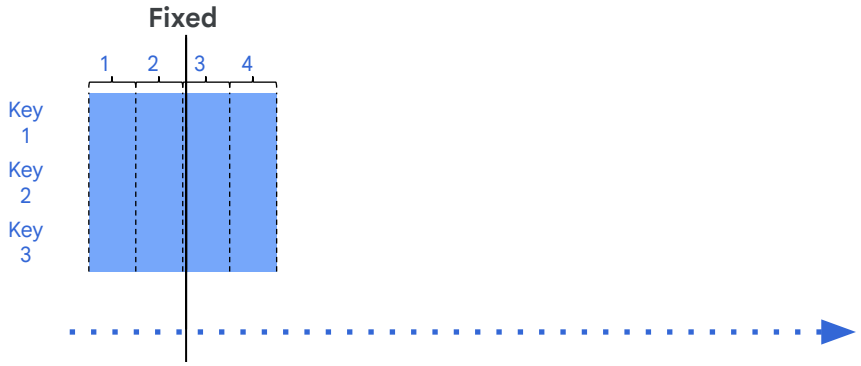
Dataflow can shuffle arriving messages into windows based on event time.

Fixed time is the simplest to use. Each Window is given an interval.

Sliding windows have an interval and a period. The interval defines how long the window collects data for and the period defines how often a new window starts.

Session windows define windows on areas of concentrated data that use a key to combine data into groups, like sessions in a web application.

Three kinds of windows fit most circumstances

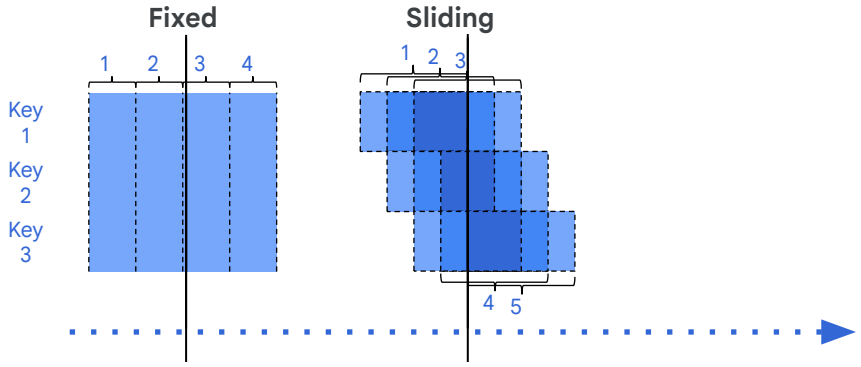


Windowing divides data into time-based finite chunks

Also refer to:
https://cloud.google.com/dataflow/docs/concepts/beam-programming-model

Fixed are those that are divided into time slices, for example, hourly, daily, monthly. Fixed time windows consist of consistent non-overlapping intervals.

Three kinds of windows fit most circumstances



Windowing divides data into time-based finite chunks

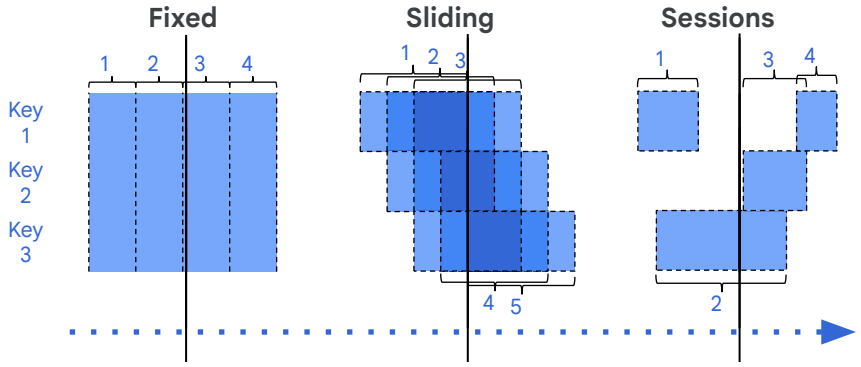
Often required when doing aggregations over unbounded data

Also refer to:
https://cloud.google.com/dataflow/docs/concepts/beam-programming-model

Sliding windows are those or example; the last 24 hours worth of data, every hour), and session-based windows that capture bursts of user activity.

Sliding time windows can overlap, for example, in a running average.
Session windows are defined by a minimum gap duration and the timing is triggered by another element.

Three kinds of windows fit most circumstances



Windowing divides data into time-based finite chunks

Often required when doing aggregations over unbounded data

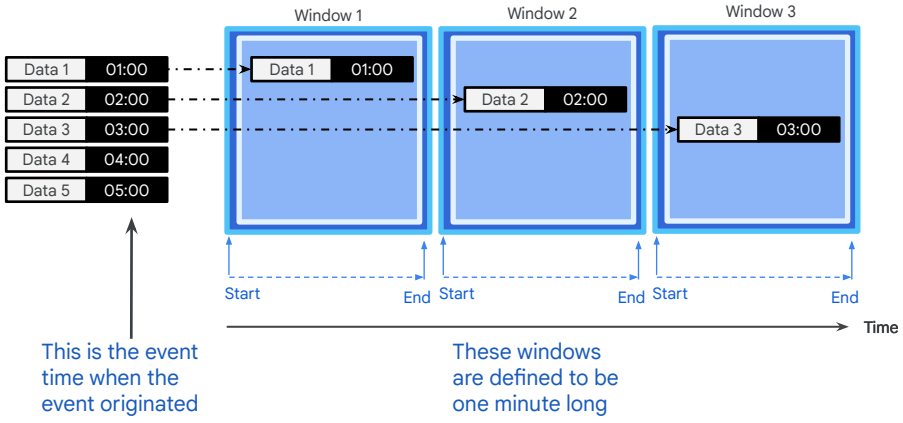
Also refer to:
https://cloud.google.com/dataflow/docs/concepts/beam-programming-model

Sliding windows are those or example; the last 24 hours worth of data, every hour), and session-based windows that capture bursts of user activity.

You could implement window-based processing on bounded data based on the date-timestamp of elements. However, it is necessary when performing aggregation processing on streaming data.

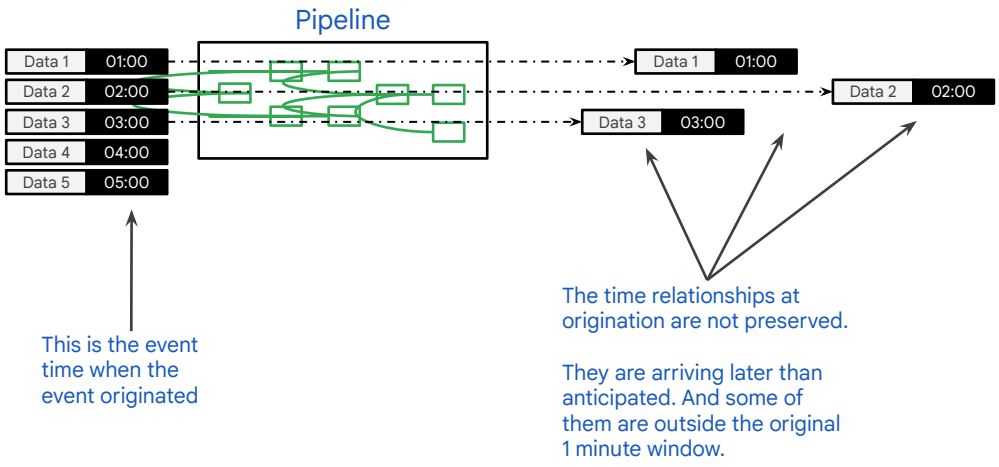
Sliding time windows can overlap, for example, in a running average.
Session windows are defined by a minimum gap duration and the timing is triggered by another element.

Windowing by time if there is no latency



How does windowing work? All things being equal, this is how windowing ought to work. If there was no latency, if we had in an ideal world, if everything was instantaneous, then these fixed time windows would just flush at the close of the window. At the very microsecond at which becomes 8:05:00, a five minute window terminates, and flushes all of the data. This is only IF there is not latency.

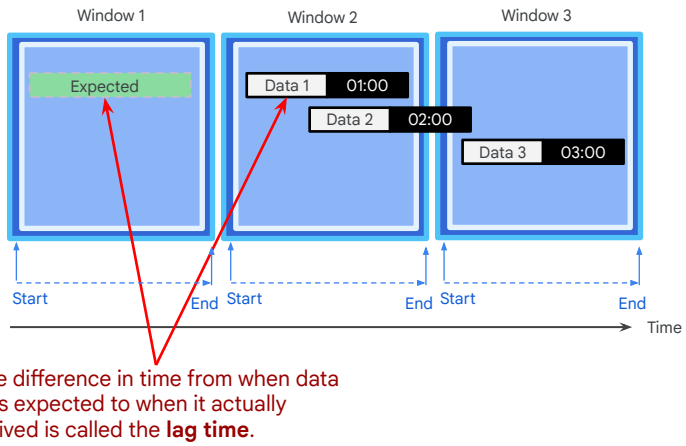
Pipeline processing can introduce latency



But in the real world, latency happens. We have network delays, system backlogs, processing delays, Pub/Sub latency, etc., So, when do we want to close the window? Should we wait a little bit longer than 8:05, maybe a few more seconds?

How should Cloud Dataflow deal with this situation?

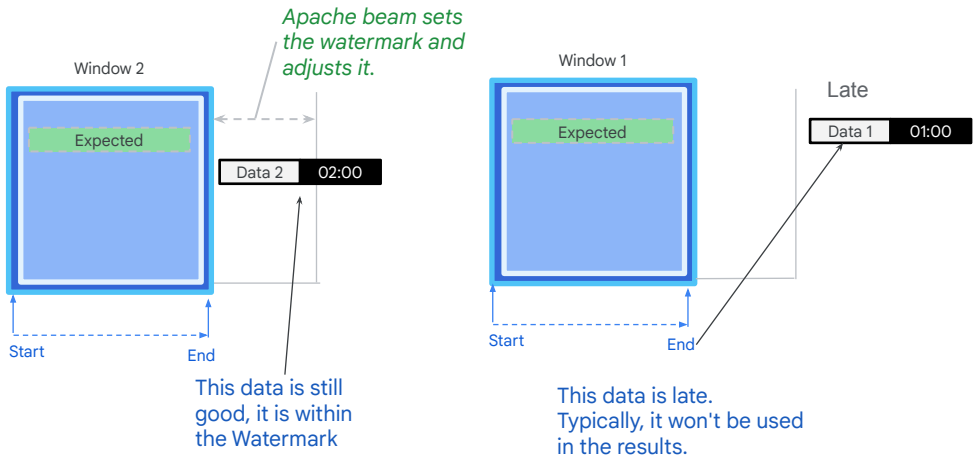
The data could be a little past the window or a lot. Data 2 is a little outside of Window 2. Data 1 is completely outside of Window 1.



The difference in time from when data was expected to when it actually arrived is called the **lag time**.

This is what we call the watermark and Dataflow keeps track of it automatically. Basically, it is going to keep track of the lag time, and it is able to do this, for example, if you are using the Pub/Sub connector, because it knows the time of the oldest, unprocessed message in Pub/Sub. And then it knows the latest message it has processed through the dataflow. It, then, takes this difference and that is the lag time.

Watermarks provide flexibility for a little lag time

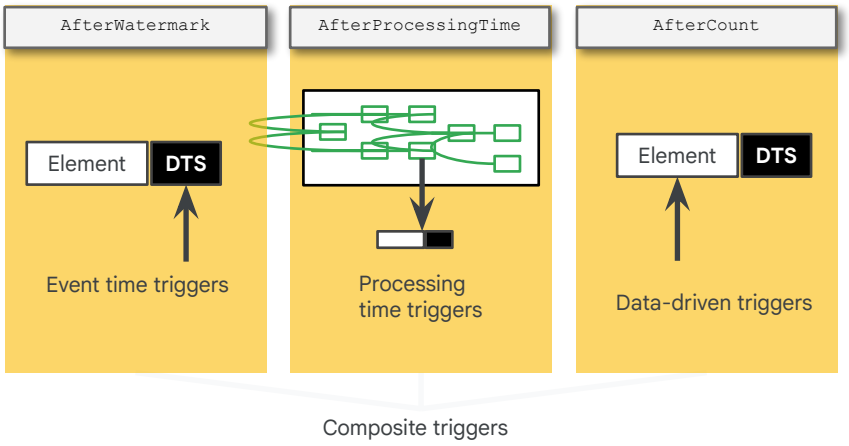


So, what Dataflow is going to do is continuously compute the watermark, which is how far behind are we. Dataflow ordinarily is going to wait until the watermark it has computed has elapsed, so if it is running a system lag of three of four seconds, it is going to wait four seconds before it flushes the window because that is when it believes all of the data should have arrived for that time period.

What, then, happens to late data? Let's say it gets an event with a timestamp of 8:04, but now it is 8:06. It is two minutes late, one minute after the close of the window, what does it do with that data? The answer is, you get to choose that. The default is just to discard it, but you can also tell it to reprocess the window based on those late arrivals.

"Beam's default windowing configuration tries to determines when all data has arrived (based on the type of data source) and then advances the watermark past the end of the window. This default configuration does not allow late data."
https://beam.apache.org/documentation/programming-guide/#windowing-basics

The default is to trigger at the watermark, but we can also add custom trigger(s)



Event time triggers operate on the date-timestamp associated with each element. The default trigger is of this type. The **AfterWatermark** trigger is the only event-type trigger currently supported. Apache Beam determines when all the elements with a date-timestamp that falls within the window have been processed. This is the Watermark. The passing of the Watermark causes the aggregation step to be performed.And after the Watermark has passed, the default event time trigger is activated. Its behavior is to emit the results of the aggregation one time, and to discard any data that arrives late. In Java pipelines you can override this behavior and do something with late data. In a Python pipeline, currently, late data is discarded unconditionally.

Processing time triggers operate on the time at which an element is processed at some point in the pipeline as determined by a system clock. You could set an **AfterProcessingTime** trigger on unbounded data contained in a global window. For example, emitting data every 30 seconds. The data never ends. The Window never closes. But interim results are emitted every 30 seconds by the trigger.

And a **data-driven trigger** is associated with the condition of data contained in the element itself. Currently, this simply counts each element that has been processed in the window. You could set **AfterCount** to 15, and every 15 elements processed would cause and emit.

Composite triggers combine effects. For example, consider if you had a data-driven AfterCount trigger set to 15. Every 15 elements it would emit. However, if there were

14 elements in the PCollection, and no more data arrived, the 14 would sit in the window forever. In this case you could add in an Event time trigger to ensure that the last 14 were serviced by an emit.

Some example triggers

```
pcollection | WindowInto(
    SlidingWindows(60, 5),                              # Sliding window of 60 seconds, every 5 seconds
    trigger=AfterWatermark(                             # Relative to the watermark, trigger:
      early=AfterProcessingTime(delay=30),              # -- fires 30 seconds after pipeline commences
      late=AfterCount(1))                               # -- and for every late record (< allowedLateness)
    accumulation_mode=AccumulationMode.ACCUMULATING)    # the pane should have all the records
```

```
pcollection | WindowInto(
    FixedWindows(60),                                   # Fixed window of 60 seconds
    trigger=Repeatedly(                                 # Set up a composite trigger that triggers ...
        AfterAny(                                       # whenever either of these happens:
            AfterCount(100),                            # -- 100 elements accumulate
            AfterProcessingTime(1 * 60))),              # -- every 60 seconds (ignore watermark)
    accumulation_mode=AccumulationMode.DISCARDING)      # the trigger should be with only new records
```


https://beam.apache.org/documentation/programming-guide/#composite-triggers

The opposite of Repeatedly is or Finally which tells the trigger to stop

We'll talk about accumulation mode shortly.

You can allow late data past the watermark

Allowing Late Data

Java

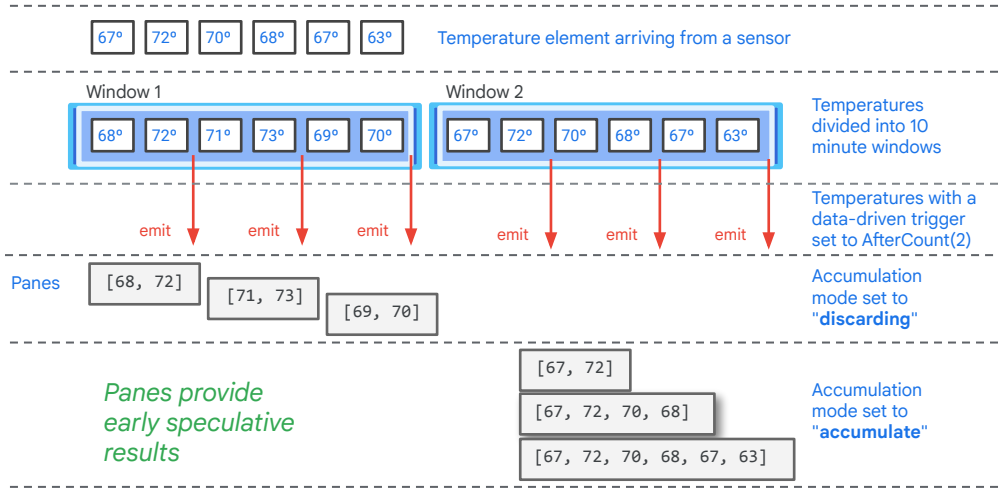
```
PCollection<String> items = ...;

    PCollection<String> fixedWindowedItems = items.apply(

Window.<String>into(FixedWindows.of(Duration.standardMinutes(1)))
              .withAllowedLateness(Duration.standardDays(2)));
```

Python

```
pc = [Initial PCollection]
   pc | beam.WindowInto(
              FixedWindows(60),
              trigger=trigger_fn,
              accumulation_mode=accumulation_mode,
              timestamp_combiner=timestamp_combiner,
              allowed_lateness=Duration(seconds=2*24*60*60)) # 2 days
```


This is how the window re-processes. This late processing works in both Java and Python. Implementation of Apache Beam watermark support is part of the Open Source Software and not directly implemented by Google.

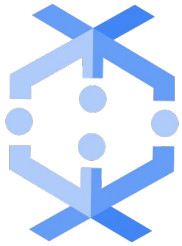
Accumulation modes: what to do with additional events



When you set a trigger, you need to choose either accumulate mode or discard mode. This example shows the different behaviors caused by the intersection of windowing, triggers, and accumulation mode.

Dataflow Templates

Dataflow Templates



Dataflow Templates

- Premade templates for creating Dataflow jobs with no coding
- Support both streaming and batch operations
- Support many different sources and sinks
  - Pub/Sub
  - Storage
  - BigQuery
  - Firestore
  - Spanner JDBC
  - Etc.

Although Dataflow is a solution that requires a user to provide their own Java or Python code, there are also premade templates that can be used which provide very little coding.

The templates support both batch and streaming examples.

The premade templates also support a variety of different sources and sinks including Pub/Sub, Cloud Storage, BigQuery, Firestore, and Spanner JDBC.

Dataflow Templates

Allow you to stage your pipelines on Google Cloud and run them using the Google Cloud Console, the Google Cloud CLI, or REST API calls

- **Classic Templates**: staged as execution graphs on Cloud Storage
- **Flex Templates**: package the pipeline as a Docker image and stage these images on your project's Container Registry or Artifact Registry

Dataflow Templates
https://cloud.google.com/dataflow/docs/concepts/dataflow-templates

Dataflow templates allow you to stage your pipelines on Google Cloud and run them using the Google Cloud Console, the Google Cloud CLI, or REST API calls.

- Classic templates are staged as execution graphs on Cloud Storage while
- Flex Templates package the pipeline as a Docker image and stage these images on your project's Container Registry or Artifact Registry.

You can use one of the Google-provided templates or create your own.

Templated Dataflow Jobs
https://cloud.google.com/dataflow/docs/concepts/dataflow-templates#templated-dataflow-jobs

If you use classic templates or Flex Templates, staging and execution are separate steps. This separation gives you additional flexibility to decide who can run jobs and where the jobs are run from.

Dataflow Templates

Feature	Classic templates	Flex Templates
Separate staging and execution steps	Yes	Yes
Run the template using the Google Cloud console, `gcloud` CLI, or REST API calls	Yes	Yes
Run pipeline without recompiling code	Yes	Yes
Run pipeline without development environment and associated dependencies	Yes	Yes
Customize pipeline execution with runtime parameters	Yes[1]	Yes
Can update streaming jobs[2]	Yes	Yes
Supports FlexRS[2]	Yes	Yes
Run validations upon job graph construction to reduce runtime errors	No	Yes
Can change job execution graph after the template is created	No	Yes
Supports SQL parameters	No	Yes
Supports I/O interfaces beyond `ValueProvider`	No	Yes

https://cloud.google.com/dataflow/docs/concepts/dataflow-templates

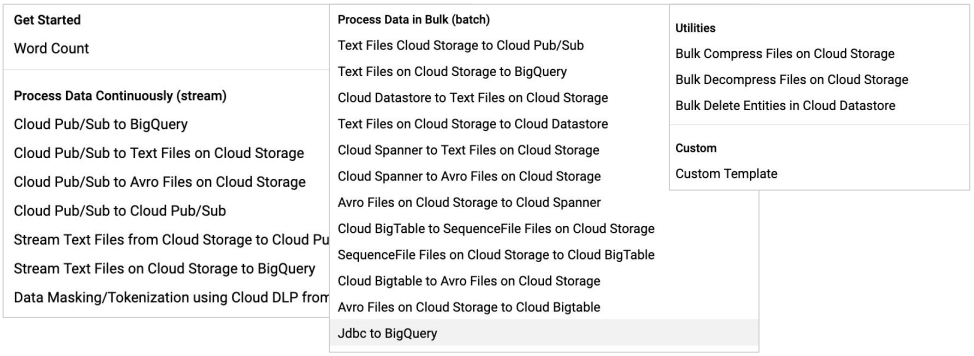
Google Provided Templates

Allow you to stage your pipelines on Google Cloud and run them using the Google Cloud Console, the Google Cloud CLI, or REST API calls

- **Streaming templates -** Templates for processing data continuously
- **Batch templates -** Templates for processing data in bulk
- **Utility templates**

Get started with Google-provided templates
https://cloud.google.com/dataflow/docs/guides/templates/provided-templates

Dataflow Templates Illustrated



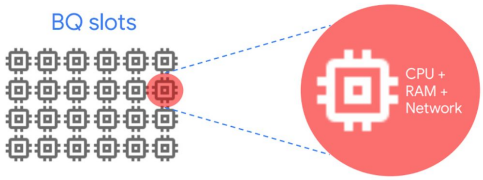
The templates can be run from the Google Cloud console. Here is an example of the templates available.

Slots

What are BigQuery Slots

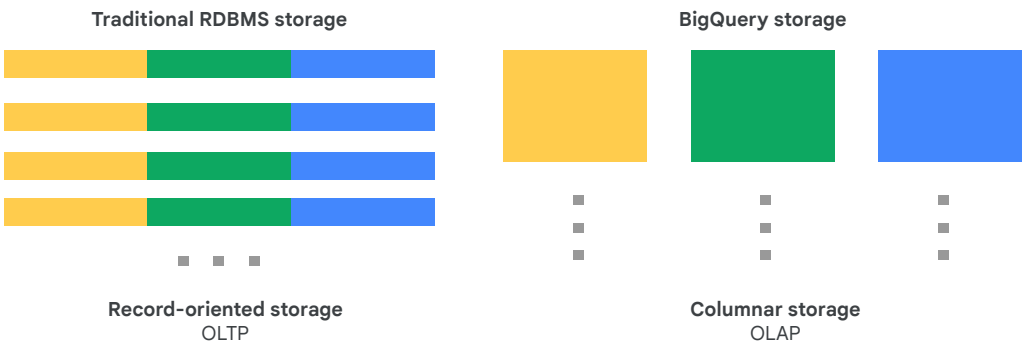
1. A Slot is a virtual CPU

2. When you provision slots you purchase processing capacity. More slots gives you faster queries and/or more concurrency.



So in BigQuery essentially you analytics runs on slots. And a BigQuery slot is a piece of compute, memory and network that is optimized for processing data. So optimizing your spend essentially means purchasing slots. Lets look at the pricing model on the next slide.

BigQuery Managed Columnar storage



This makes it good for transactional updates. On GCP, use Cloud SQL for a large amount of transactional updates. BQ supports 1000 transactions per day through single statement transactions.

BigQuery, on the other hand, uses columnar storage, where each column is stored in a separate, file block. BQ is a great fit for immutable, massive datasets.
You can stream (append data) easily to BigQuery tables, but not change existing values.

BigQuery does not require indexes, keys or partitions.

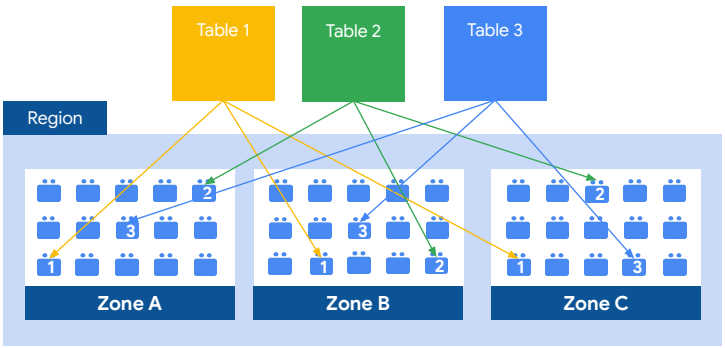
OTLP = Online Transaction Processing

OLAP = Online Analytical Processing

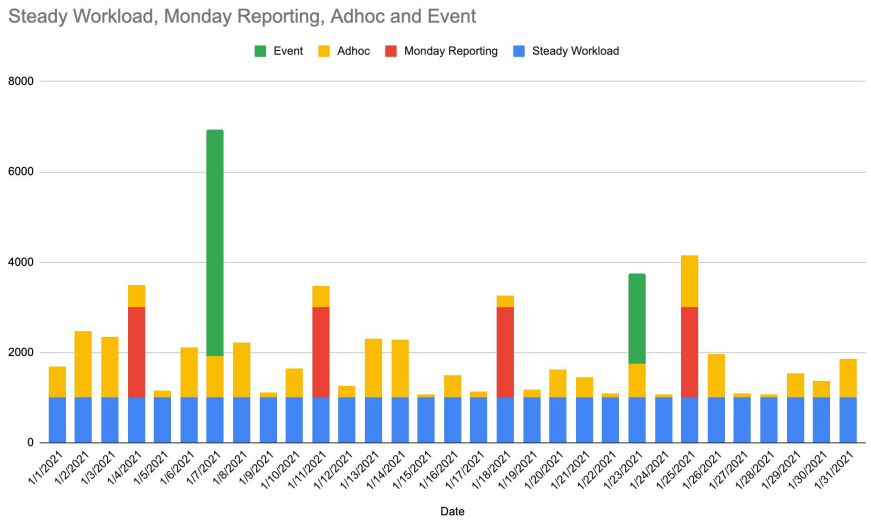
This is why SELECT * is expensive and (partly) why LIMIT does not change the amount of data analyzed.

BigQuery Managed Columnar storage

- Tables are stored in optimized columnar format
- Each table is compressed and encrypted on disk
- Storage is durable and each table is replicated across datacenters



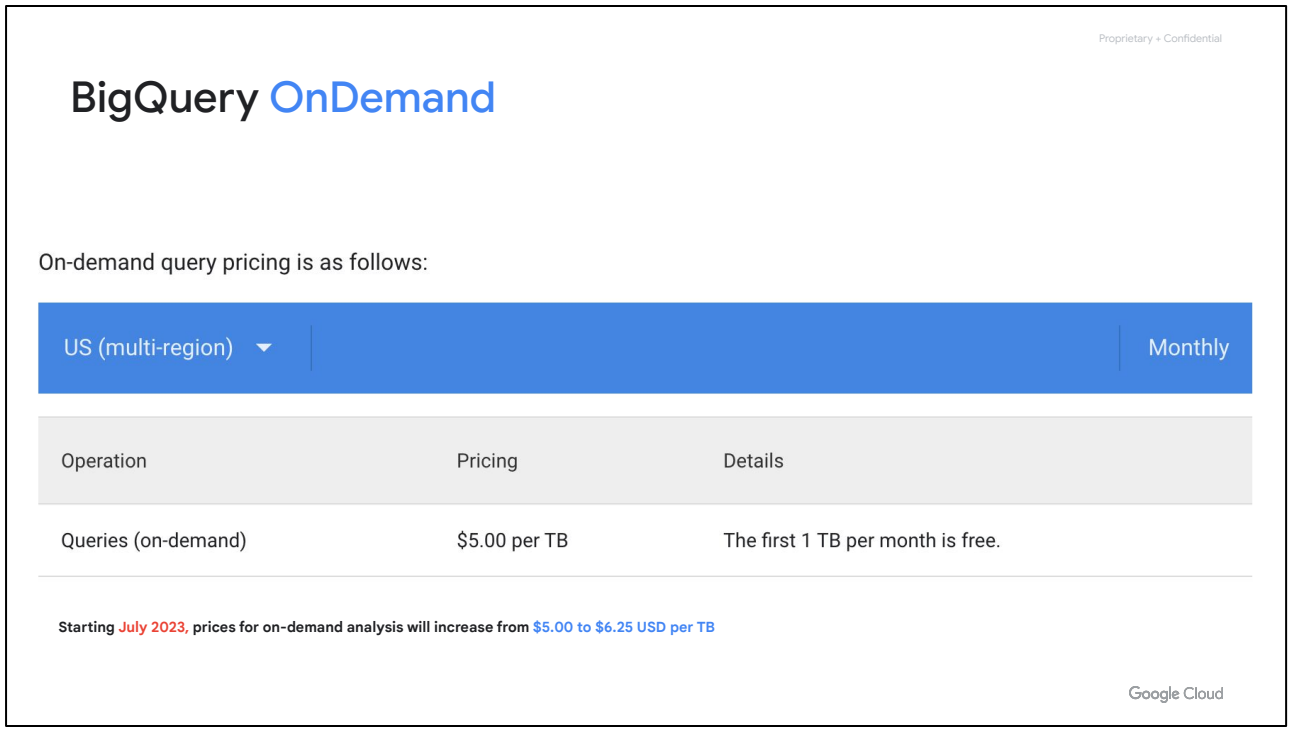
BigQuery Pricing model: Different Use-cases



By default BQ provides on demand capacity which can burst upto 2000 slots and beyond and your pricing is simply the amount of data processed. However as your organization process more and more data, it seems viable to purchase slots and process unlimited data with fixed predictable cost.

BigQuery OnDemand

ttps://cloud.google.com/bigquery/pricing



By default BQ provides on demand capacity which can burst upto 2000 slots and beyond and your pricing is simply the amount of data processed. However as your organization process more and more data, it seems viable to purchase slots and process unlimited data with fixed predictable cost.

https://cloud.google.com/bigquery/pricing

BigQuery Reservations and Flex Slots

Deprecated: Ending July 2023

ttps://cloud.google.com/bigquery/pricing

Query Pricing Options

Deprecated: Ending July 2023

Ad Hoc

- Perpetual Free Tier
- Pay-as-you-go (On Demand)

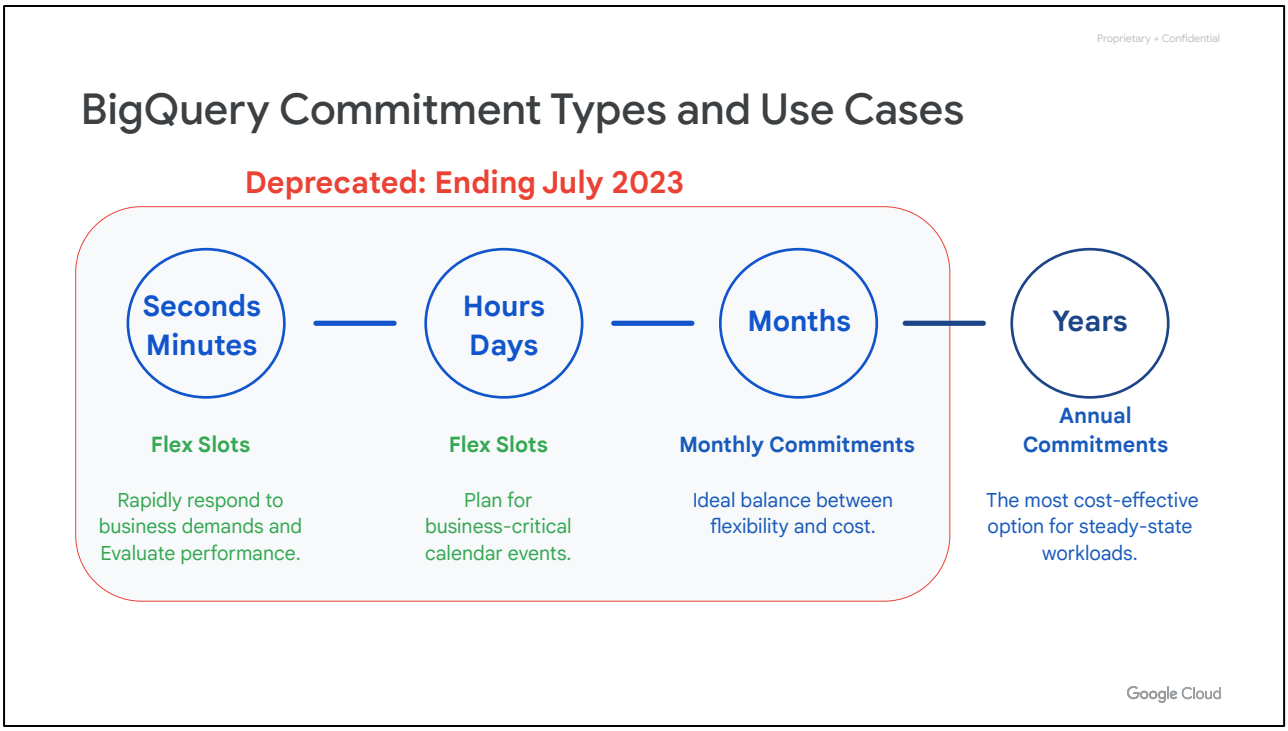
Scheduled

- Flex Slots
  - 60 Second Minimum
- Flat Rate
  - Monthly
  - Annual (15% Discount)

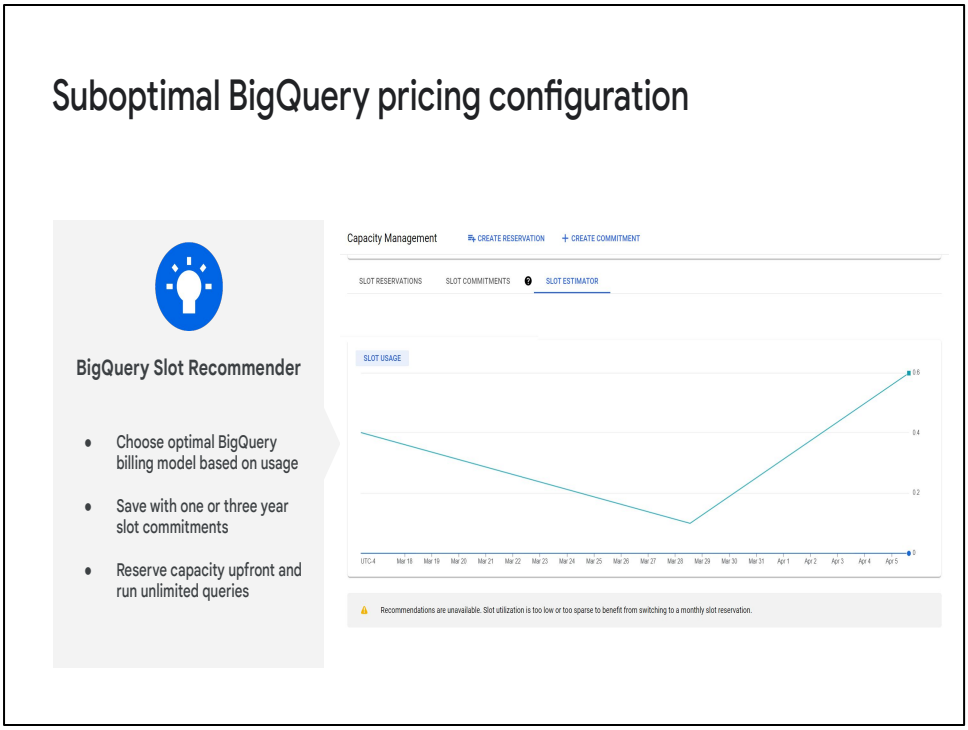
Efficient

Manage Slots with Reservations API!

Predictable



As you can see you can choose what's best based on your use case. Election season is around the corner and lets see if a media organization is going to process huge data for a month or 2, monthly commitments makes the most sense. If it's Blackfriday/CM type of event, you might want your analytics to be supercharged with flex slots. And in case where your analytics is steady state regardless of the season that's where annual commitment will pass along maximum savings.



Last but not the least, for heavy users of BigQuery - Slot Estimator will recommend the best billing model based for your usage.

BigQuery Editions

ttps://cloud.google.com/bigquery/pricing

BigQuery editions

Flexibility

- Match workloads to tiers, so you **only pay for the features you need**
- **Mix and match** editions for the best price performance per workload

Price Predictability

- 1-year and 3-year commitments with **higher discounts**
- Leverage autoscale to **pay only for what you use**. No need to overpay for underused capacity
- **Low cost storage**

Control

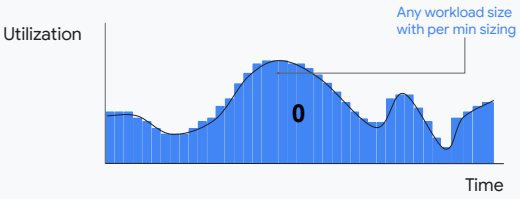
- Autoscaling with **baseline and max thresholds** to intelligently manage costs
- **BigQuery BI Engine dynamically** manages slots while queries are running!

BigQuery Autoscaler

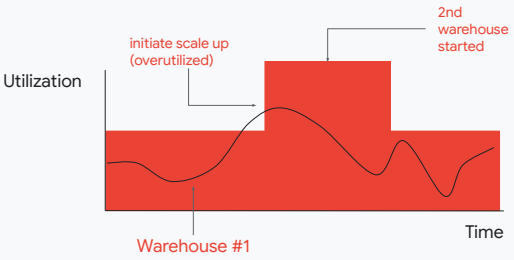
Dynamically adjusts the capacity in response to planned or unplanned changes in demand to help ensure you **pay only for what you use**

Autoscaling is the core functionality available in a new commercial model

Intelligently plan capacity in advance and only pay for what you use



vs. market alternatives with upfront provisioning and unpredictable costs



BigQuery editions, all with autoscaling

For detailed pricing information visit cloud.google.com/bigquery/pricing

Standard

$0.04/slot hour

Low-cost option for standard SQL analysis

- Autoscaling slots
- Capped at 1,600 slots per reservation
- Max 5 reservations per admin project
- 99.95% SLA
- Zonal High Availability
- Google Cloud platform wide certifications1 (Foundational compliance)
- Covered by Google Cloud BAA (for HIPAA compliance)
- Google managed keys

[1] All Google Cloud certifications including ISO 9001, ISO 27001, SOC 1-3, PCI

Enterprise

$0.06/slot hour

Advanced enterprise workloads

Standard features +

- Unlimited reservation size
- 99.99% SLA
- BI query acceleration
- Integrated ML models
- Full-text search
- Object tables
- VPC Service Controls to prevent data exfiltration
- Data masking, column security and row security

Optional 1 yr commitment ($0.048/slot hour)
Optional 3 yr commitment ($0.036/slot hour)

Enterprise Plus

$0.10/slot hour

Business critical enterprise workloads

Enterprise features +

- Regional-level disaster recovery2
- Customer-managed encryption keys
- Support for PCI, FedRAMP, ITAR and other compliance regimes available through Assured Workloads

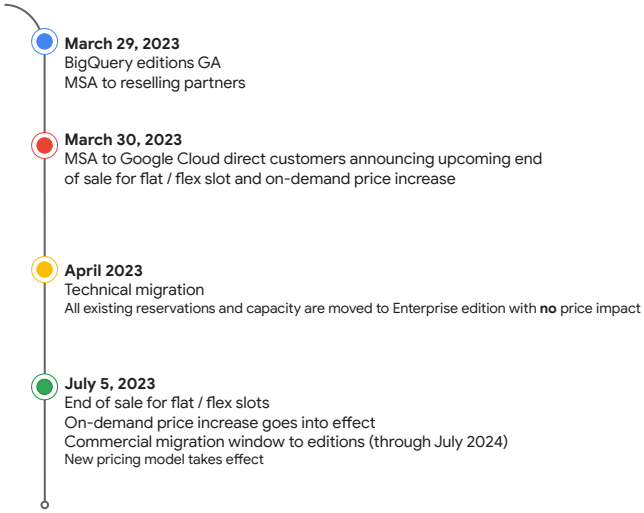
Optional 1 yr commitment ($0.08/slot hour)
Optional 3 yr commitment ($0.06/slot hour)

*Roadmap functionality

On-demand (pay-as-you-go for data processed)

$6.25/TB scanned. First 1 TB/month is free
Includes all capabilities of Enterprise Plus

Timeline



March 29, 2023
BigQuery editions GA
MSA to reselling partners

March 30, 2023
MSA to Google Cloud direct customers announcing upcoming end of sale for flat / flex slot and on-demand price increase

April 2023
Technical migration
All existing reservations and capacity are moved to Enterprise edition with **no** price impact

July 5, 2023
End of sale for flat / flex slots
On-demand price increase goes into effect
Commercial migration window to editions (through July 2024)
New pricing model takes effect

Information Schemas

Overview of BigQuery Schemas

- Schemas are defined at the table level and provide structure to the data
- Schema describes column definitions with their name, data type, description and mode
- **Data types** can be simple data types, such as integers, or more complex, such as ARRAY and STRUCT for nested and repeated values.
- **Column modes** can be NULLABLE, REQUIRED, or REPEATED
- Specified:
  - When loading data into the table or when creating an empty table
  - Via schema auto-detection when loading data (Avro, Parquet, ORC or (Cloud Firestore or Cloud Datastore exports)
  - Manually or in a JSON file

BigQuery explained: How to query your data (Sep 23, 2020)
https://cloud.google.com/blog/topics/developers-practitioners/bigquery-explained-querying-your-data

- Table schema is specified when loading data into the table or when creating an empty table.
- Alternatively, when loading data, you can use schema auto-detection for self-describing source data formats such as Avro, Parquet, ORC, Cloud Firestore or Cloud Datastore export files.
- Schema can be defined manually or in a JSON file

Schema auto-detection
https://cloud.google.com/bigquery/docs/schema-detect#auto-detect

Self-describing
https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-parquet#parquet_schemas

Manually specifying schemas
https://cloud.google.com/bigquery/docs/schemas#manually_specifying_schemas

Via JSON file
https://cloud.google.com/bigquery/docs/schemas#specifying_a_json_schema_file

INFORMATION_SCHEMA views

The following INFORMATION_SCHEMA views are now available

- TABLES
- TABLE_OPTIONS
- COLUMNS
- COLUMN_FIELD_PATHS
- VIEWS
- ROUTINES
- ROUTINE_OPTIONS
- PARAMETERS
- SCHEMATA
- SCHEMATA_OPTIONS

https://cloud.google.com/bigquery/docs/release-notes#October_14_2020

The following INFORMATION_SCHEMA views are now generally available (GA).

- TABLES
- TABLE_OPTIONS
- COLUMNS
- COLUMN_FIELD_PATHS
- VIEWS
- ROUTINES
- ROUTINE_OPTIONS
- PARAMETERS
- SCHEMATA
- SCHEMATA_OPTIONS

INFORMATION_SCHEMA limitations

- BigQuery INFORMATION_SCHEMA is subject to the following limitations:
  - BigQuery INFORMATION_SCHEMA queries must be in standard SQL syntax
- INFORMATION_SCHEMA does not support legacy SQL
- INFORMATION_SCHEMA query results are not cached
- Currently, INFORMATION_SCHEMA cannot be used to retrieve metadata on partitions in partitioned tables
- Currently, INFORMATION_SCHEMA views cannot be used in DDL statements

https://cloud.google.com/bigquery/docs/information-schema-intro#limitations

Recall the minimum charge of 10MB per query in BQ → so this is something to be aware of using this, results are not cached.

INFORMATION_SCHEMA pricing

On-demand Projects:
Queries against INFORMATION_SCHEMA views incur a minimum of 10 MB of data processing charges, even if the bytes processed by the query are less than 10 MB. 10 MB is the minimum billing amount for on-demand queries

Flat Rate Projects:
Queries against INFORMATION_SCHEMA views and tables consume your purchased BigQuery slots

INFORMATION_SCHEMA Queries:
Are not cached, you are charged each time you run an INFORMATION_SCHEMA query

Introduction to BigQuery INFORMATION_SCHEMA | Google Cloud

Qualifying INFORMATION_SCHEMA

Current Dataset or default Project:

SELECT * FROM INFORMATION_SCHEMA.SCHEMATA;

Returns all Datasets in the current Project

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.2 sec elapsed, 10 MB processed)

Job information | Results | JSON | Execution details

Row	catalog_name	schema_name	schema_owner	creation_time	last_modified_time	location
1	mpa-dev1-233020	fn	*null*	2021-04-15 15:55:44.341 UTC	2021-04-15 15:55:44.341 UTC	US
2	mpa-dev1-233020	GeoViz	*null*	2021-05-19 18:47:13.443 UTC	2021-05-19 18:47:13.443 UTC	US
3	mpa-dev1-233020	billingdemo	*null*	2020-10-08 20:43:15.041 UTC	2020-10-08 20:43:15.041 UTC	US
4	mpa-dev1-233020	demos	*null*	2020-06-09 03:13:14.142 UTC	2020-06-09 03:13:14.142 UTC	US
5	mpa-dev1-233020	babynames	*null*	2021-05-26 16:26:56.038 UTC	2021-05-26 16:26:56.038 UTC	US
6	mpa-dev1-233020	ecommerce	*null*	2020-04-26 23:09:21.137 UTC	2020-04-26 23:09:21.137 UTC	US
7	mpa-dev1-233020	billing	*null*	2020-10-27 20:21:51.737 UTC	2021-06-18 21:09:53.919 UTC	US

[Introduction to BigQuery INFORMATION_SCHEMA | Google Cloud]

Dataset metadata

Each dataset contains metadata that can be queries using INFORMATION_SCHEMA.SCHEMATA_OPTIONS

```
SELECT * FROM INFORMATION_SCHEMA.SCHEMATA_OPTIONS WHERE option_name="labels"
```

Row	catalog_name	schema_name	option_name	option_type	option_value
1	mpa-dev1-233020	GeoViz	labels	ARRAY<STRUCT<STRING, STRING>>	[STRUCT("environment", "mpa-dev1"), STRUCT("purpose", "geoviz_data")]
2	mpa-dev1-233020	babynames	labels	ARRAY<STRUCT<STRING, STRING>>	[STRUCT("environemnt", "map-dev1"), STRUCT("purpose", "data2insights")]
3	mpa-dev1-233020	billing	labels	ARRAY<STRUCT<STRING, STRING>>	[STRUCT("environment", "mpa-dev1"), STRUCT("purpose", "bq-billing")]

Getting dataset metadata using INFORMATION_SCHEMA | BigQuery (google.com)
https://cloud.google.com/bigquery/docs/information-schema-datasets

Reservation metadata

Query the INFORMATION_SCHEMA reservation views to retrieve real-time metadata about BigQuery reservations. These views contain a list of changes to reservations, assignments, and capacity commitments, along with a timeline of reservations.

Excellent technique for retrieving real-time Slot information about:

- Assignments by Project
- Capacity Commitments by Project
- Reservation History
- Reservations by Project
- Slot Usage Over Time

Getting reservations metadata using INFORMATION_SCHEMA | BigQuery (google.com)

Monitoring BigQuery reservations and slot utilization with INFORMATION_SCHEMA (June 11, 2021)
https://cloud.google.com/blog/topics/developers-practitioners/monitoring-bigquery-reservations-and-slot-utilization-information_schema

BigQuery Reservations help manage your BigQuery workloads. With flat-rate pricing, you can purchase BigQuery slot commitments in 100-slot increments in either flex, monthly, or yearly plans instead of paying for queries on demand. You can then create/manage buckets of slots called reservations and assign projects, folders, or organizations to use the slots in these reservations. By default, queries running in a reservation automatically use idle slots from other reservations. In this way, organizations have greater control over workload management in a way that ensures high-priority jobs always have access to the resources they need without contention. Currently, two ways to monitor these reservations and slots are via the BigQuery Reservations UI or Cloud Monitoring.

But how does an organization know how many slots to delegate to a reservation? Or if a reservation is being over or underutilized? Or what the overall slot utilization is across all reservations? In this blog post, we will discuss how we used BigQuery's INFORMATION_SCHEMA system tables to create the System Tables Reports

Dashboard and answer these questions.

Example: Results - Slot usage over time

The query joins RESERVATIONS_BY_PROJECT with JOBS_TIMELINE_BY_PROJECT to associate the job time slices with the reservation information.

reservation_name	period_start	slot_capacity	period_slot_ms	job_id	job_type
my_reservation	2021-04-30 17:30:54	100	11131	bquxjob_66707...	QUERY
my_reservation	2021-04-30 17:30:55	100	49978	bquxjob_66707...	QUERY
my_reservation	2021-04-30 17:30:56	100	9038	bquxjob_66707...	QUERY
my_reservation	2021-04-30 17:30:57	100	17237	bquxjob_66707...	QUERY

Note:

This query uses the RESERVATIONS_BY_PROJECT view to get reservation information. If the reservations have changed in the past hour, the reservation_slot_capacity column might not be accurate.

Getting reservations metadata using INFORMATION_SCHEMA | BigQuery (google.com)
https://cloud.google.com/bigquery/docs/information-schema-reservations

Monitoring BigQuery reservations and slot utilization with INFORMATION_SCHEMA

INFORMATION_SCHEMA tables that are specifically relevant to monitoring slot utilization across jobs and reservations

- JOBS_BY_ORGANIZATION
- CAPACITY_COMMITMENT_CHANGES_BY_PROJECT
- RESERVATION_CHANGES_BY_PROJECT
- ASSIGNMENT_CHANGES_BY_PROJECT

Monitoring BigQuery reservations and slot utilization with INFORMATION_SCHEMA (Jun 11, 2021)
https://cloud.google.com/blog/topics/developers-practitioners/monitoring-bigquery-reservations-and-slot-utilization-information_schema

BigQuery Reservations help manage your BigQuery workloads. With flat-rate pricing, you can purchase BigQuery slot commitments in 100-slot increments in either flex, monthly, or yearly plans instead of paying for queries on demand. You can then create/manage buckets of slots called reservations and assign projects, folders, or organizations to use the slots in these reservations.

By default, queries running in a reservation automatically use idle slots from other reservations. In this way, organizations have greater control over workload management in a way that ensures high-priority jobs always have access to the resources they need without contention.

Currently, two ways to monitor these reservations and slots are via the BigQuery Reservations UI or Cloud Monitoring.

But how does an organization know how many slots to delegate to a reservation? Or if a reservation is being over or underutilized? Or what the overall slot utilization is across all reservations?

In this blog post, we will discuss how we used BigQuery's INFORMATION_SCHEMA

system tables to create the **System Tables Reports Dashboard** and answer these questions.

Using INFORMATION_SCHEMA tables
The INFORMATION_SCHEMA metadata tables contain relevant, granular information about jobs, reservations, capacity commitments, and assignments. Using the data from these tables, users can create custom dashboards to report on the metrics they are interested in in ways that inform their decision making.

While there are several tables that make up INFORMATION_SCHEMA, there are a few that are specifically relevant to monitoring slot utilization across jobs and reservations.

- The JOBS_BY_ORGANIZATION table is the primary table to extract job-level data across all projects in the organization.
- This information can be supplemented with data from the CAPACITY_COMMITMENT_CHANGES_BY_PROJECT, RESERVATION_CHANGES_BY_PROJECT, and ASSIGNMENT_CHANGES_BY_PROJECT tables to include details about specific capacity commitments, reservations, and assignments.
- It's worth noting that the data retention period for INFORMATION_SCHEMA is 180 days and all timestamps are in UTC. For information about the permissions required to query these tables, follow the links above.

Example: Joining Reservation Metadata

```
WITH
 job_data AS (
 SELECT
  job.period_start,job.reservation_id,job.period_slot_ms,job.job_id,job.job_type
 FROM
  `mpa-dev1-233020.region-us`.INFORMATION_SCHEMA.JOBS_TIMELINE_BY_PROJECT AS job
 WHERE
  job.period_start > TIMESTAMP_SUB(CURRENT_TIMESTAMP(), INTERVAL 1 HOUR))
SELECT
 reservation.reservation_name AS reservation_name,job.period_start,reservation.slot_capacity,
 job.period_slot_ms,job.job_id,job.job_type
FROM
 job_data AS job
INNER JOIN
 `mpa-dev1-233020.region-us`.INFORMATION_SCHEMA.RESERVATIONS_BY_PROJECT AS reservation
ON
 (job.reservation_id = CONCAT(reservation.project_id, ":", "US", ".", reservation.reservation_name));
```


Getting reservations metadata using INFORMATION_SCHEMA | BigQuery (google.com)
https://cloud.google.com/bigquery/docs/information-schema-reservations
https://cloud.google.com/blog/topics/developers-practitioners/monitoring-bigquery-reservations-and-slot-utilization-information_schema

BigQuery OMNI

What is BigQuery OMNI?

- A flexible, fully managed, multi-cloud analytics solution that allows you to analyze data across clouds such as AWS and Azure
- Use standard SQL and BigQuery's familiar interface to quickly answer questions and share results from a single pane of glass across your datasets
- Take compute to where the data resides
- Powered by Anthos

Bringing multi-cloud analytics to your data with BigQuery Omni (Jul 14, 2020)
https://cloud.google.com/blog/products/data-analytics/introducing-bigquery-omni

Today, we are introducing BigQuery Omni, a flexible, multi-cloud analytics solution that lets you cost-effectively access and securely analyze data across Google Cloud, Amazon Web Services (AWS), and Azure (coming soon), without leaving the familiar BigQuery user interface (UI). Using standard SQL and the same BigQuery APIs our customers love, you will be able to break down data silos and gain critical business insights from a single pane of glass. And because BigQuery Omni is powered by Anthos, you will be able to query data without having to manage the underlying infrastructure.

Press Release - Google Cloud Announces BigQuery Omni for Multi-Cloud Analytics (Jul 14, 2020)
https://cloud.google.com/press-releases/2020/0714/bigqueryomni

BigQuery Omni, powered by Anthos, makes it easy for customers to securely and cost-effectively access, analyze and find new insights to power innovation in their business

BigQuery OMNI value proposition

- Single pane of glass to view all your data
- Cross cloud analytics at scale
- Secure analytics on a fully-managed platform

Single pane of glass to view all your data

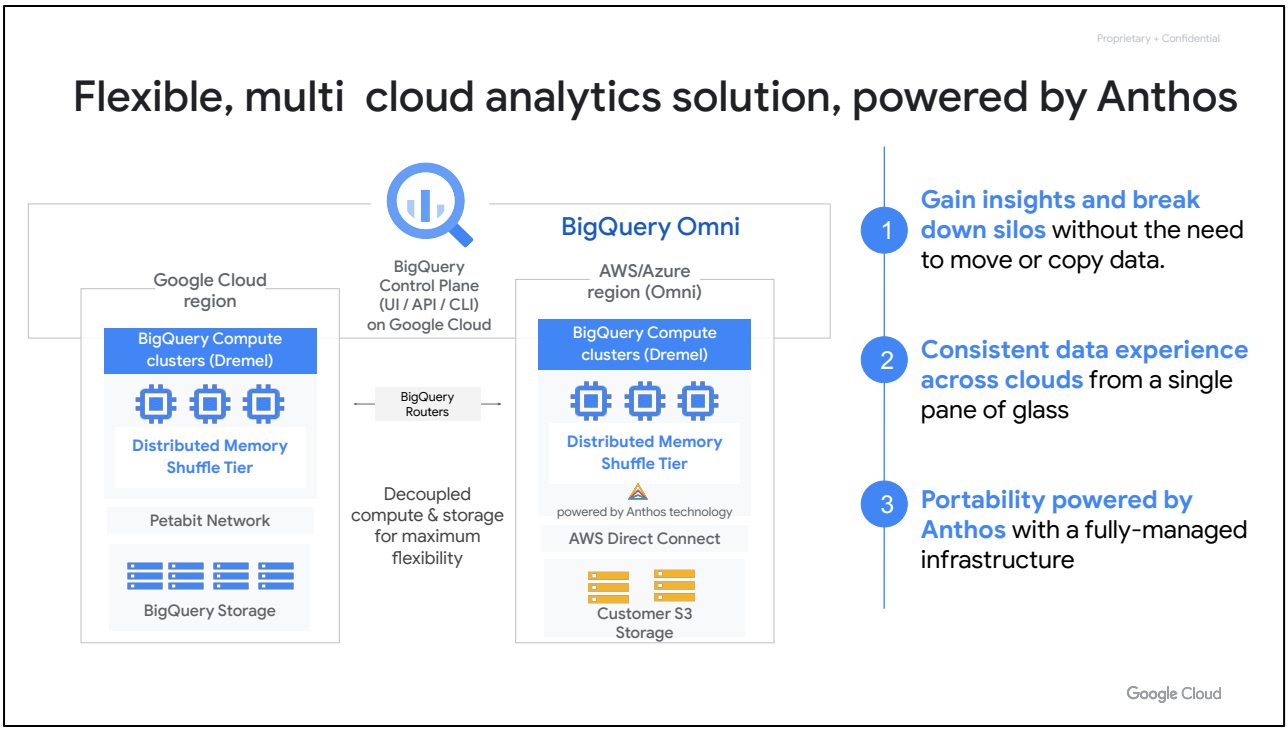
- Break down the data silos by accessing all your data using standard SQL queries and build dashboards across clouds

Cross cloud analytics at scale

- Analyze and integrate data across multiple clouds. Move resultsets to Google Cloud using table copy and leverage embedded ML for advanced analytics

Secure analytics on a fully-managed platform

- Securely analyze data across Google Cloud, AWS and Azure with fully managed infrastructure, without underlying hardware or infrastructure complexity

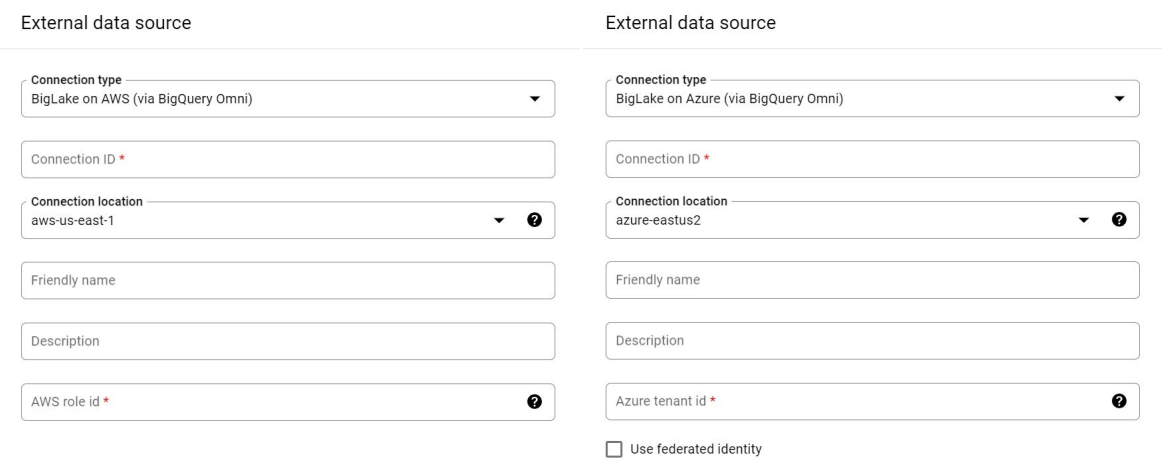


https://cloud.google.com/blog/products/data-analytics/introducing-bigquery-omni

It's also very exciting that we are focusing on multi-cloud analytics scenarios too.

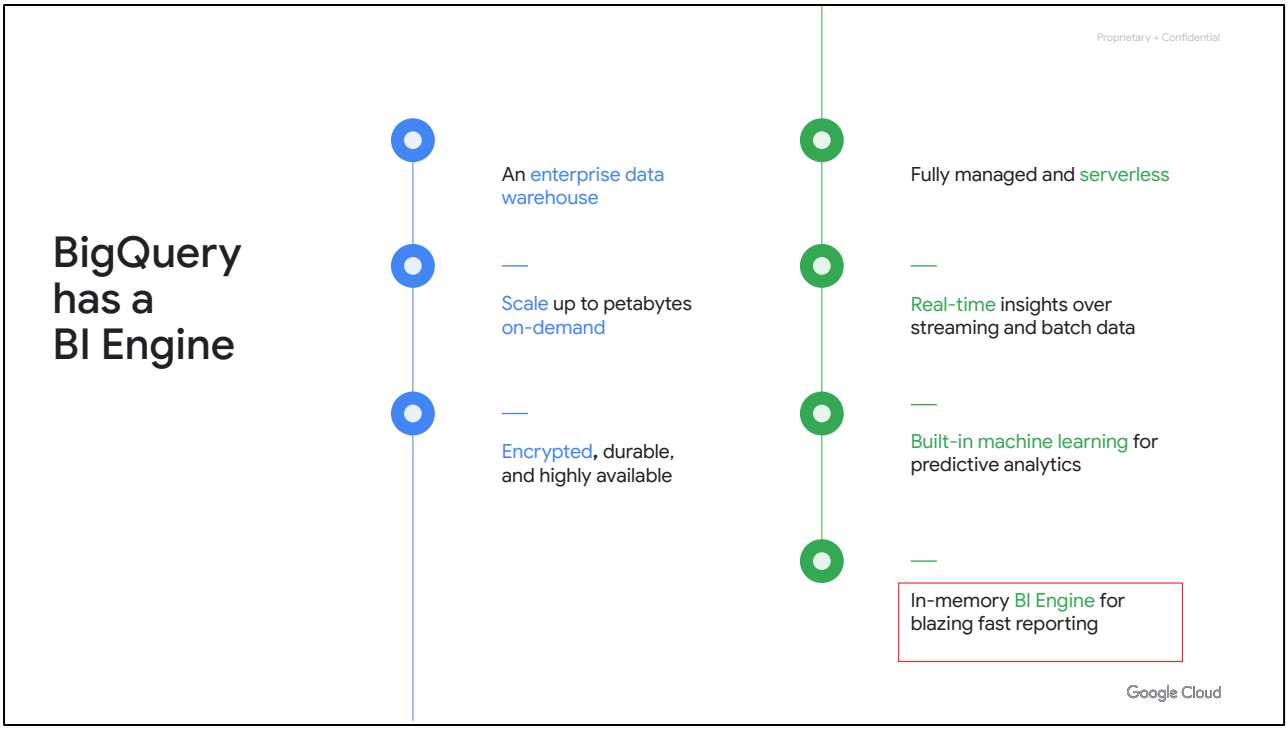
BigQuery Omni, a flexible, multi-cloud analytics solution that lets you cost-effectively access and securely analyze data across Google Cloud, Amazon Web Services (AWS), and Azure (coming soon), without leaving the familiar BigQuery user interface (UI). Using standard SQL and the same BigQuery APIs our customers love, you will be able to break down data silos and gain critical business insights from a single pane of glass. And because BigQuery Omni is powered by Anthos, you will be able to query data without having to manage the underlying infrastructure.

BigQuery OMNI value proposition



BigQuery BI Engine

BigQuery BI Engine



[What is BigQuery] -> BigQuery is Google Cloud's highly scalable, enterprise data warehouse solution. It is fully managed and serverless, offers real-time insights from streaming data, has built-in ML out of the box, and has a high-speed in memory BI engine for faster reporting and analysis. BigQuery is designed to make data analysts and data scientists more productive and it is a secure offering that has data encryption by default to ensure data security.

BI Engine
BigQuery BI Engine is a fast, in-memory analysis service that allows you to analyze data stored in BigQuery

BigQuery BI Engine

A fast, in-memory analysis service that allows you to analyze data stored in BigQuery

- Sub-second query response time and with high concurrency
- Integrates with familiar Google tools like Google Looker Studio
- Build rich, interactive dashboards and reports without compromising performance, scale, security, or data freshness
  - Securely share insights at scale
- SQL Interface
  - Expands BI Engine to integrate with other BI tools such as Looker, Tableau, Power BI and custom applications to accelerate data exploration and analysis
  - Enrollment

https://cloud.google.com/bi-engine/docs
BigQuery BI Engine is a fast, in-memory analysis service that allows you to analyze data stored in BigQuery

GA Feb 20, 2020
https://cloud.google.com/bi-engine/docs/release-notes#February_20_2020

BigQuery BI Engine now interacts with popular BI tools such as Looker, Tableau, and more, by means of an SQL interface. You must enroll to participate in the preview.
https://cloud.google.com/bi-engine/docs/release-notes#February_25_2021

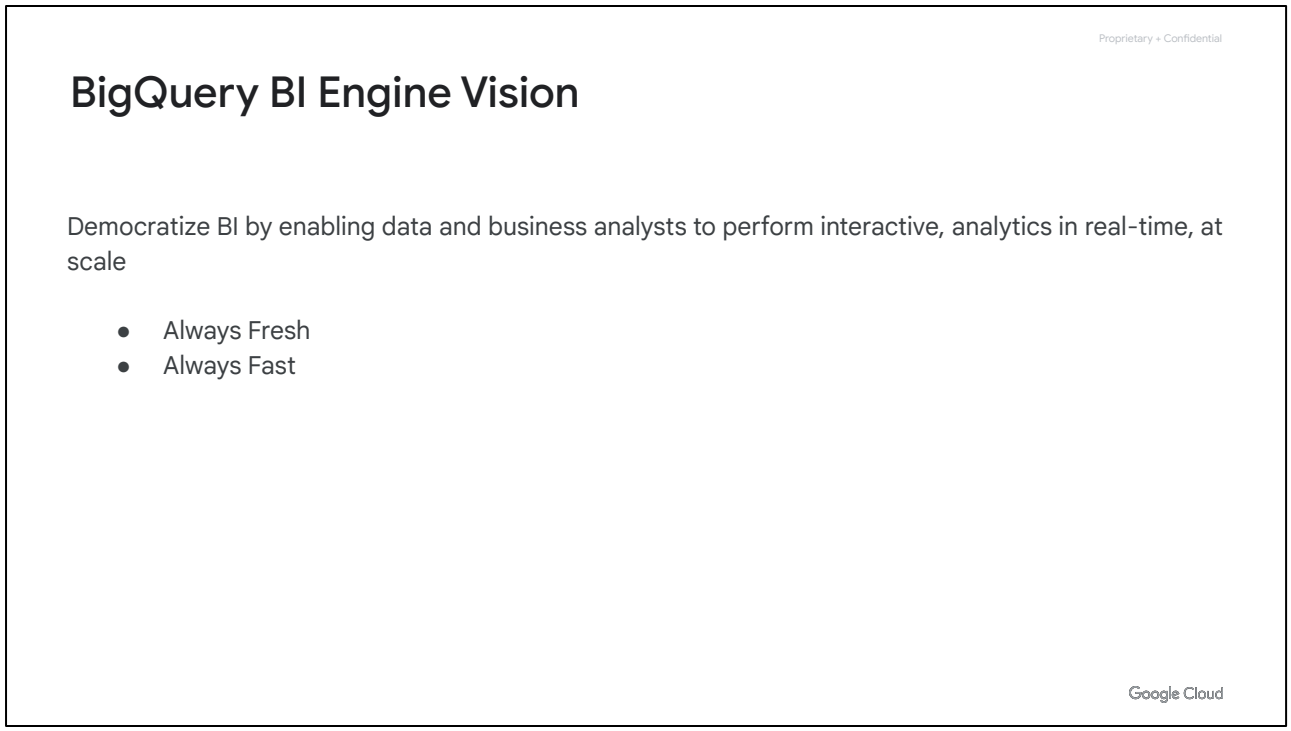
SQL Interface Preview
https://cloud.google.com/bi-engine/docs/sql-interface-overview#requesting_access_to_the_preview
Access to the BI Engine SQL interface in this preview phase is provided through an enrollment process. For access, submit the BI Engine preview enrollment form with your project details. You will be notified by email once your project is enrolled.

BI Engine Introduction
https://cloud.google.com/bi-engine/docs/introduction
BigQuery BI Engine is a fast, in-memory analysis service. By using BI Engine you can analyze data stored in BigQuery with sub-second query response time and with high concurrency

BI Engine integrates with familiar Google tools like Google Data Studio. The BI Engine SQL interface feature also integrates with other popular business intelligence (BI) tools, such as Looker, Tableau, Power BI, and custom applications to accelerate data exploration and analysis. The BI Engine SQL interface feature is in preview and requires enrollment.

With BI Engine, you can build rich, interactive dashboards and reports without compromising performance, scale, security, or data freshness.

Analyze large and complex datasets interactively with sub-second query response time by combining BigQuery BI Engine, an in-memory, column-oriented analysis service, with your favorite BI tools. Easily create stunning reports and dashboards using popular BI tools like Looker, Tableau, Google Data Studio, Google Sheets and more. out of the box and securely share insights at scale.



Always fast and fresh dashboards: Inside BigQuery BI Engine (Sep 15, 2020)
https://www.youtube.com/watch?v=kifUXfwqzVo

Where BI Engine is very powerful is that it as an acceleration layer between BigQuery and your BI tool.

BigQuery BI Vision
Democratize BI by enabling data and business analysts to perform interactive, analytics in real-time, at scale

- Always Fresh
- Always Fast

Today, BI tools compromise between data freshness and speed

- If you want data to be really fast and interactive, you compromise on freshness
- If you want data to be fresh, the latest without staleness, you compromise on speed
-

With BI Engine, you don't have to compromise between freshness and speed

BigQuery BI Engine Advantages

- **Fast**:
  - Match performance to the speed of business by reducing time to insights
  - Sub-second queries
- **Simplified architecture**:
  - Get started quickly without managing complex data pipeline or servers
- **Smart tuning**:
  - Very few configuration settings

https://cloud.google.com/bi-engine/docs/introduction
BigQuery BI Engine is a fast, in-memory analysis service. By using BI Engine you can analyze data stored in BigQuery with sub-second query response time and with high concurrency

BI Engine integrates with familiar Google tools like Google Data Studio. The BI Engine SQL interface feature also integrates with other popular business intelligence (BI) tools, such as Looker, Tableau, Power BI, and custom applications to accelerate data exploration and analysis. The BI Engine SQL interface feature is in preview and requires enrollment.

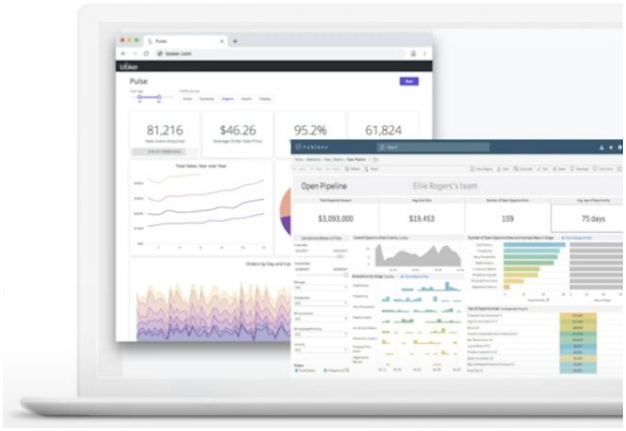
With BI Engine, you can build rich, interactive dashboards and reports without compromising performance, scale, security, or data freshness.

Always fast and fresh dashboards: Inside BigQuery BI Engine (Sep 15, 2020)
https://www.youtube.com/watch?v=kifUXfwqzVo

Where BI Engine is very powerful is that it as an acceleration layer between BigQuery and your BI tool.

BI Engine Sub-second queries

1. Column oriented, Dynamic in-memory execution engine
2. Horizontal scaling to support higher concurrency
3. Native integration with BigQuery Streaming for real-time data refreshes



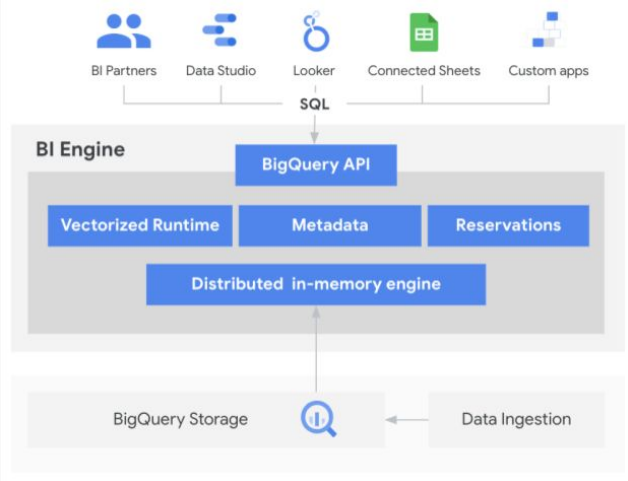
Always fast and fresh dashboards: Inside BigQuery BI Engine (Sep 15, 2020)
https://www.youtube.com/watch?v=kifUXfwqzVo

Where BI Engine is very powerful is that it as an acceleration layer between BigQuery and your BI tool.

- BI Engine is a column-oriented in-memory execution engine, which enables it to achieve sub-second queries
- It can scale horizontally so the more request s you have coming in, BI Engine can load balance those requests by making copies of data
- BI Engine works with BQ Streaming - as data is streaming in, the data is automatically refreshed into the in-memory queue so you get the latest data

BI Engine Simplified Architecture

1. Built on existing BigQuery Storage
2. Eliminates the need to manage BI Servers, ETL pipelines or complex extracts
3. No need to build and manage traditional OLAP cubes



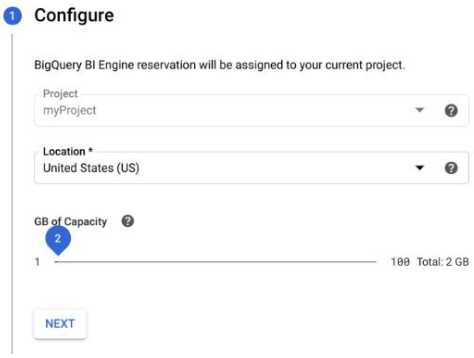
Always fast and fresh dashboards: Inside BigQuery BI Engine (Sep 15, 2020)
https://www.youtube.com/watch?v=kifUXfwqzVo

Graphic from:
https://cloud.google.com/bi-engine/docs/sql-interface-overview

- No need to extract data out of BigQuery. BI Engine automatically moves data between the storage tier and the memory tier
- BI Engine doesn't require you to build OLAP cubes on your own, it handles that for you
- Currently integrated with Data Studio
- BI Engine SQL API (preview) is integrated with BigQuery API directly. This will enable other BI tools such as Looker, Tableau, PowerBI, etc … to work with BI Engine

BI Engine Smart Tuning

1. In-memory cache optimization avoids querying raw data
2. Ability to scale capacity up and down inside BigQuery UI
   a. Purchase capacity on-demand
   b. Purchase BI Engine capacity by creating a Reservation
3. Full visibility into metrics including aggregate refresh time, cache hit ratios, query latency inside Google Operations



Always fast and fresh dashboards: Inside BigQuery BI Engine (Sep 15, 2020)
https://www.youtube.com/watch?v=kifUXfwqzVo

Where BI Engine is very powerful is that it as an acceleration layer between BigQuery and your BI tool.

Smart Tuning
BI Engine's self-tuning design automatically tunes queries by moving data between BI Engine's in-memory storage, the BigQuery query cache, and BigQuery storage to ensure optimal performance and load times for dashboards. Your BigQuery administrator can easily add and remove BI Engine memory capacity by using the Cloud Console.

BigQuery query cache
https://cloud.google.com/bigquery/docs/cached-results

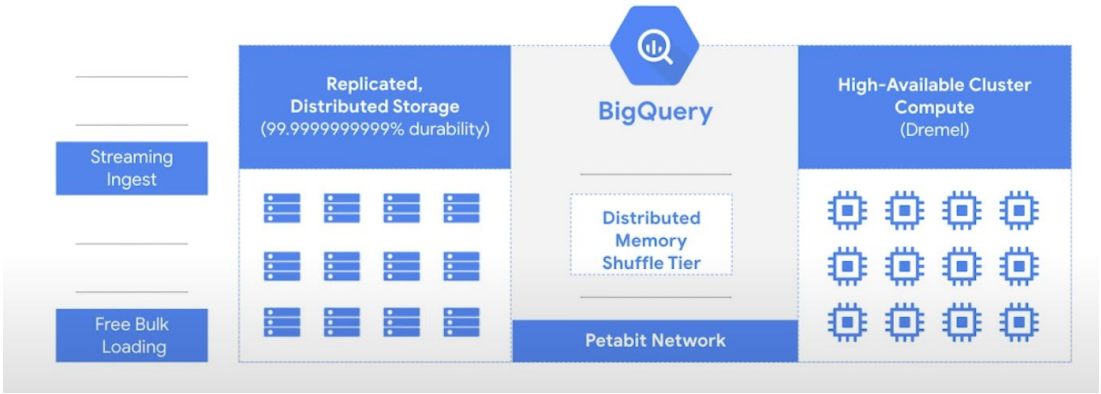
BI Engine Pricing
https://cloud.google.com/bi-engine/pricing

- On-demand
  - https://cloud.google.com/bi-engine/pricing#on_demand_pricing
- Flat-rate via Reservation
  - https://cloud.google.com/bi-engine/pricing#flat_rate_pricing

Creating a Reservation

https://cloud.google.com/bi-engine/docs/reserving-capacity#creating_a_reservation
You specify the location you want to have that reservation and the size in GB, BI Engine automatically takes care of tuning the data based on your query patterns

BigQuery Architecture review



Always fast and fresh dashboards: Inside BigQuery BI Engine (Sep 15, 2020)
https://www.youtube.com/watch?v=kifUXfwqzVo

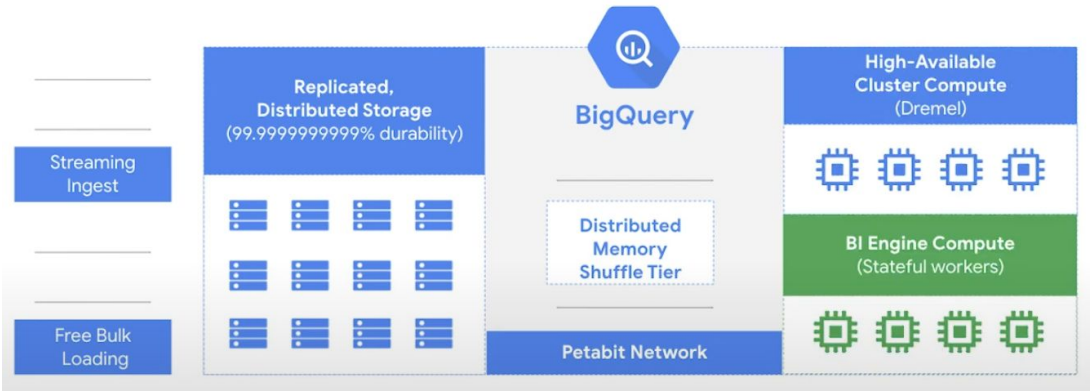
Where BI Engine is very powerful is that it as an acceleration layer between BigQuery and your BI tool.

You can think of BI Engine as a feature of BigQuery, it's not a separate product

BQ Architectural Advantage

- Decoupled storage and compute for maximum flexibility

BigQuery BI Engine



Always fast and fresh dashboards: Inside BigQuery BI Engine (Sep 15, 2020)
https://www.youtube.com/watch?v=kifUXfwqzVo

Where BI Engine is very powerful is that it as an acceleration layer between BigQuery and your BI tool.

You can think of BI Engine as a feature of BigQuery, it's not a separate product

BQ BI Engine
Stateful execution engine on top of existing BigQuery Storage

- It works on top of the same storage, same network, same shuffle layer and working alongside BigQuery workers (which are Slots)
- So if this is the case, what makes it possible for BI Engine to get such dramatic performance improvements

Statefule workers, persist unlike standard BQ slots that get unallocated after they are used

Accelerate data exploration and unlock insights

Super fast reporting

Enable real-time dashboarding over data inside BigQuery with sub-second query response and with high concurrency.

Simplified architecture

Simplify BI architecture and accelerate time to insight by reducing additional ETL, extracts or BI server management.

Smart tuning

Optimize visual analytics over big data sets with automatic performance tuning and capacity flexibility.

BI Engine for faster reporting and analysis

BigQuery BI Engine is a fast, in-memory analysis service. By using BI Engine you can analyze data stored in BigQuery with sub-second query response time and with high concurrency.

BI Engine integrates with familiar Google tools like Google Data Studio to accelerate data exploration and analysis. With BI Engine, you can build rich, interactive dashboards and reports in Data Studio without compromising performance, scale, security, or data freshness.

BigQuery Cost Optimization

BigQuery Pricing

Query processing

The cost to process queries in BigQuery

- **Ondemand pricing:** You are charged for the number of bytes processed by each query
- **Flat-rate pricing:** you purchase slots, which are virtual CPUs - a dedicated processing capacity that you can use to run queries.

Storage

The cost to store data that you load into BigQuery.

Cost optimization best practices for BigQuery

The cost to process queries, including SQL queries, user-defined functions, scripts, and certain data manipulation language (DML) and data definition language (DDL) statements that scan tables

Cost optimization techniques

Query processing

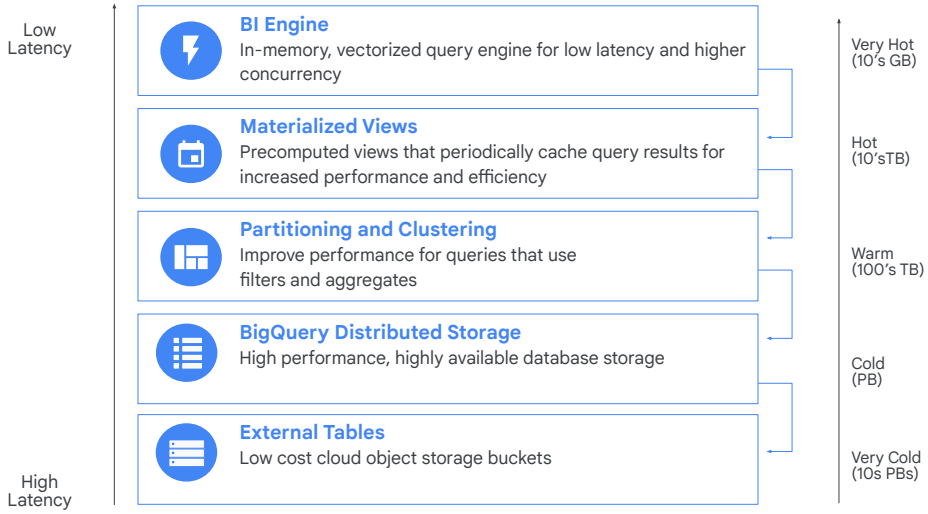
- Ondemand pricing
  - Query the data you need
  - Query cost controls
  - Partition and Cluster your tables
    (includes zero maintenance auto-reclustering)
- Flat-rate pricing

Storage

- Data Retention
- Long term storage
- Avoid duplicate storage - use federated data access model
- Streaming Insert
- Backup and recovery

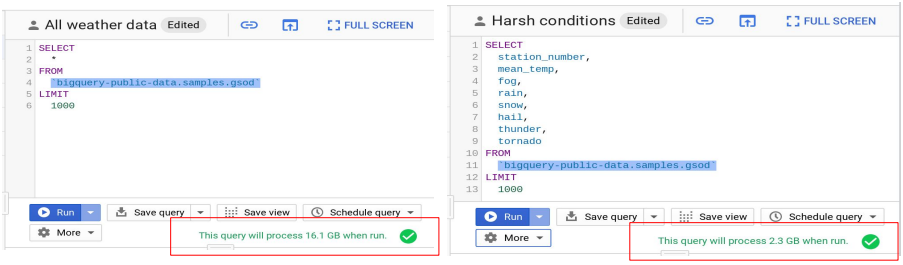
BigQuery adaptive caching

Multi-level, smart caching architecture supports real-time analytics.



Query the data you need

- Avoid SELECT *
- Use preview option to explore your data - its free!
- Filter your query as early and as often as possible to improve performance and reduce cost.
- Check how much your query is going to be charged
- Avoid SQL anti-patterns



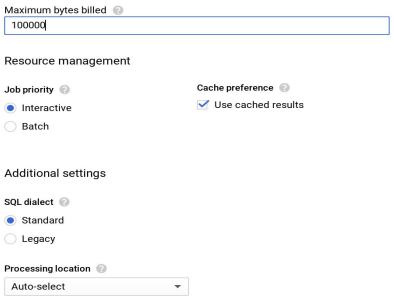
Optimize querying

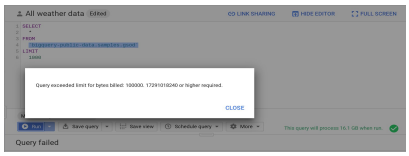
1. Query required data
2. Enforce cost control
3. Partition and cluster
4. Flat-rate pricing

Using `SELECT *` is the most expensive way to query data. When you use `SELECT *`, BigQuery does a full scan of every column in the table.

Avoid human errors

- Enforce MAX limits on bytes processed at query, user and project level.
- Cancelling a query may cost $
- Use caching intelligently

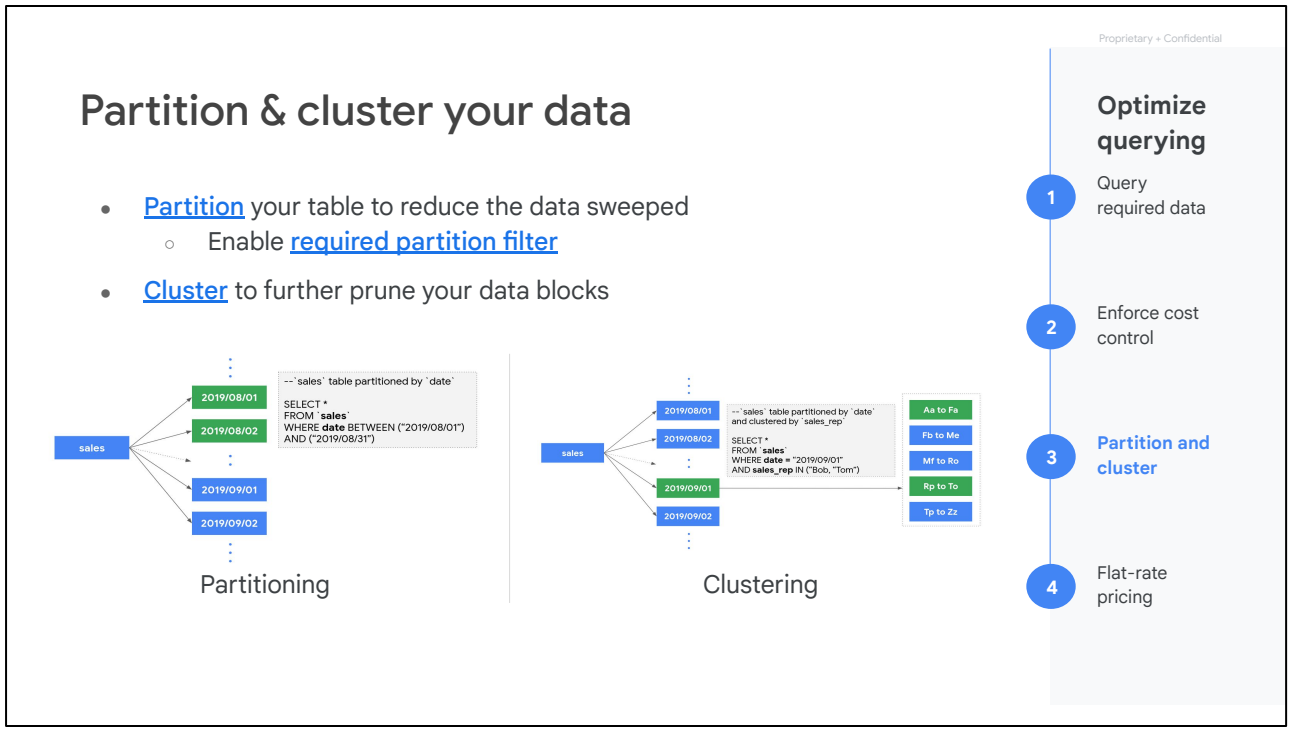




Optimize querying

1. Query required data
2. **Enforce cost control**
3. Partition and cluster
4. Flat-rate pricing

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. There are no storage costs for temporary tables



A partitioned table is a special table that is divided into segments, called partitions, that make it easier to manage and query your data. By dividing a large table into smaller partitions, you can improve query performance, and you can control costs by reducing the number of bytes read by a query.

You can partition BigQuery tables by:

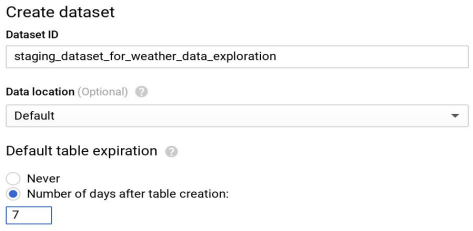
- **Ingestion time**: Tables are partitioned based on the data's ingestion (load) time or arrival time.
- **Date/timestamp/datetime**: Tables are partitioned based on a `TIMESTAMP`, `DATE`, or `DATETIME` column.
- **Integer range**: Tables are partitioned based on an integer column.

When you create a clustered table in BigQuery, the table data is automatically organized based on the contents of one or more columns in the table's schema. The columns you specify are used to colocate related data.

Flat-rate or On-Demand

- Think about flat-rate once your BigQuery processing cost > $XXK
  - Familiarize with BigQuery cost using our pricing calculator
- How many slots you should buy? - Visualize slot utilization slot utilization in BigQuery Monitoring, BigQuery Information Schemas and DevOps Monitoring.

How long are you keeping your data?



Dataset level

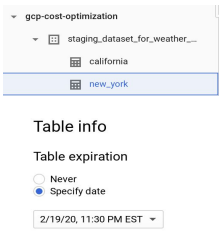


Table level

Similar to dataset-level and table-level, you can also set up expiration at partition-level.

Optimizing Storage

1. Data Retention
2. Long term storage
3. Avoid duplicate storage
4. Streaming inserts
5. Backup and Recovery

Be wary how you edit your data?

- If your table or partition has not been edited for 90 days, the the storage price drops by 50%.
- Watchout for any **actions** that edits your table: Loading into BQ, DML operations, streaming inserts, ..
- For long term archives with access frequency at most once a year - leverage **Coldline or archival class** in Cloud Storage.

The first 10 GB of storage per month is free.

US (multi-region)		Monthly
Operation	Pricing	Details
Active storage	$0.020 per GB	The first 10 GB is free each month.
Long-term storage	$0.010 per GB	The first 10 GB is free each month.

https://cloud.google.com/bigquery/pricing#storage

Avoid duplicate copies of data

Leverage BigQuery's federated data access model for your data stored on:

- Cloud Drive
- Cloud BigTable
- Cloud Storage
- Cloud SQL
- Cloud Spanner
- AWS with OMNI
- Azure with OMNI

Use cases:

- Infrequently changing data set
- Querying is less performant - **gotcha!**

Loading the data

- **Batch upload** is free. Use **streaming inserts** only if it consumed by downstream processes in real time.

Understanding DR and backup processes

- By default your 7-day history is tracked by BigQuery at the service level.
  - You can find **examples** in our public documentation for point in time restore.
- If you delete your table, you cannot restore it after 7 days.

Avro compressed is faster
Avro
Parquet and ORC
CSV and JSON is slowest

External table limitations

- BigQuery does not guaranteed external table data changes
- Performance with external tables may be slower than native BigQuery queries.
- **Failure to colocate the BigQuery dataset with the External datasource can result in cross zone egress fees on top of the query cost**

https://cloud.google.com/bigquery/external-data-sources

External table definition files

- A table definition file contains an external table's schema definition and metadata, such as the table's data format and related properties.
- **Table definition files are used with the bq command-line tool**
- When you create a table definition file you can:
  - Use schema auto-detection to define the schema for an external data source
  - Provide the schema inline (on the command line)
  - Provide a JSON file containing the schema definition

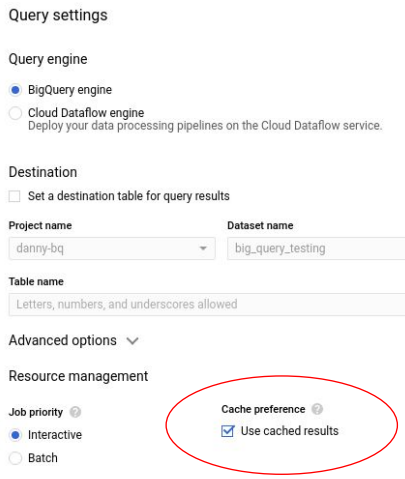
https://cloud.google.com/bigquery/external-table-definition

Query cache

Hash of
- Data modification times
- Tables used

Cache skipped if
- Referenced tables or views have changed
- Non-deterministic function used (e.g. NOW())
- Permanent result table requested
- Source tables have streaming buffers

Hash becomes output table name
- Cache is per-user



Exceptions to query caching

Query results are not cached:

- When a destination table is specified in the job configuration, the GCP Console, the classic web UI, the command line, or the API
- If any of the referenced tables or logical views have changed since the results were previously cached
- When any of the tables referenced by the query have recently received streaming inserts (a streaming buffer is attached to the table) even if no new rows have arrived
- If the query uses non-deterministic functions; for example, date and time functions such as `CURRENT_TIMESTAMP()` and `NOW()`, and other functions such as `CURRENT_USER()` return different values depending on when a query is executed
- If you are querying multiple tables using a wildcard
- If the cached results have expired; typical cache lifetime is 24 hours, but the cached results are best-effort and may be invalidated sooner
- If the query runs against an external data source
-

Interpreting the query plan

- **Significant difference between avg and max time?**
  - Probably data skew—use APPROX_TOP_COUNT to check
  - Filter early to workaround
- **Most time spent reading from intermediate stages**
  - Consider filtering earlier in the query
- **Most time spent on CPU tasks**
  - Consider approximate functions, inspect UDF usage, filter earlier

Considerations when interpreting the query plan:
If there is a significant difference between avg and max time for workers consider using Approx Top Count
https://cloud.google.com/bigquery/docs/reference/standard-sql/approximate_aggregate_functions#approx_top_count

Also if there are a lot of reads in intermediate steps consider filtering early

For long periods of time spent on CPU tasks consider approx functions https://cloud.google.com/bigquery/docs/reference/standard-sql/approximate_aggregate_functions, filtering early and optimize your UDF usage with a persistent UDF or avoid using all together (see Best Practices for functions section at end)

Visualize cost

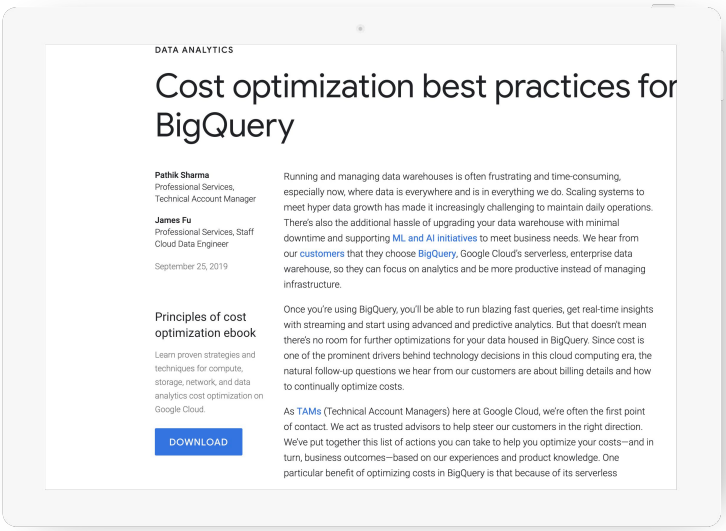
- Create your own dashboard
- Analyze spending trend & query trend over time
- Breakdown cost per project and per user
- Be proactive about tracking your expensive queries and optimize them
- Checkout another **dashboard** for slot utilization metrics and job report summary

Plug n play dashboard

https://lookerstudio.google.com/u/0/reporting/1y1AsO-hOSS6U6RkXkQKxEr3P7KXA8RL9/page/nSaN

Blogpost

For more details



DATA ANALYTICS

Cost optimization best practices for BigQuery

Pathik Sharma
Professional Services, Technical Account Manager

James Fu
Professional Services, Staff Cloud Data Engineer

September 25, 2019

Principles of cost optimization ebook

Learn proven strategies and techniques for compute, storage, network, and data analytics cost optimization on Google Cloud.

DOWNLOAD

Running and managing data warehouses is often frustrating and time-consuming, especially now, where data is everywhere and is in everything we do. Scaling systems to meet hyper data growth has made it increasingly challenging to maintain daily operations. There's also the additional hassle of upgrading your data warehouse with minimal downtime and supporting ML and AI initiatives to meet business needs. We hear from our customers that they choose BigQuery, Google Cloud's serverless, enterprise data warehouse, so they can focus on analytics and be more productive instead of managing infrastructure.

Once you're using BigQuery, you'll be able to run blazing fast queries, get real-time insights with streaming and start using advanced and predictive analytics. But that doesn't mean there's no room for further optimizations for your data housed in BigQuery. Since cost is one of the prominent drivers behind technology decisions in this cloud computing era, the natural follow-up questions we hear from our customers are about billing details and how to continually optimize costs.

As TAMs (Technical Account Managers) here at Google Cloud, we're often the first point of contact. We act as trusted advisors to help steer our customers in the right direction. We've put together this list of actions you can take to help you optimize your costs—and in turn, business outcomes—based on our experiences and product knowledge. One particular benefit of optimizing costs in BigQuery is that because of its serverless

Blog:
https://cloud.google.com/blog/products/data-analytics/cost-optimization-best-practices-for-bigquery

BigQuery Partitioning and Clustering

BigQuery Partitioning

Benefits of partitioning:

- Improve query performance
- Control costs by reducing the number of bytes read by a query

Introduction - Partitioning

BigQuery Partitioning

You can partition BigQuery tables by:

- Time-unit column: Tables are partitioned based on a TIMESTAMP, DATE, or DATETIME column in the table
- Ingestion time: Tables are partitioned based on the timestamp when BigQuery ingests the data
- Integer range: Tables are partitioned based on an integer column.

Note:

Each partitioned table can have up to 4,000 partitions. If you exceed this limit, consider using clustering in addition to, or instead of, partitioning

Introduction - Partitioning

BigQuery Partitioning

How to choose partitioning granularity?

When you partition a table by time-unit column or ingestion time, you choose whether the partitions have daily, hourly, monthly, or yearly granularity.

- Daily Partitioning: Default
- Hourly Partitioning: High Volume of data that spans a short date range
- Monthly or Yearly Partitioning: Tables span a wide date range

Introduction - Partitioning

Daily partitioning is the default partitioning type. Daily partitioning is a good choice when your data is spread out over a wide range of dates, or if data is continuously added over time.

Choose hourly partitioning if your tables have a high volume of data that spans a short date range — typically less than six months of timestamp values. If you choose hourly partitioning, make sure the partition count stays within the partition limits.

Choose monthly or yearly partitioning if your tables have a relatively small amount of data for each day, but span a wide date range. This option is also recommended if your workflow requires frequently updating or adding rows that span a wide date range (for example, more than 500 dates). In these scenarios, use monthly or yearly partitioning along with clustering on the partitioning column to achieve the best performance.

When to use Partitioning

Use partitioning under the following circumstances:

- You want to know query costs before a query runs. Partition pruning is done before the query runs, so you can get the query cost after partitioning pruning through a dry run. Cluster pruning is done when the query runs, so the cost is known only after the query finishes.

- You need partition-level management. For example, you want to set a partition expiration time, load data to a specific partition, or delete partitions.

- You want to specify how the data is partitioned and what data is in each partition. For example, you want to define time granularity or define the ranges used to partition the table for integer range partitioning.

BigQuery Clustering

Benefits of clustering:

- Improve performance of queries that use filter clauses.
- Improve performance of queries that aggregate data.
- You can combine partitioning with clustering. The data is first partitioned and then data in each partition is clustered by the clustering columns.
- You might not see a significant difference in query performance between a clustered and unclustered table if the table or partition is under 1 GB.

Introduction - Clustering
https://cloud.google.com/bigquery/docs/clustered-tables

BigQuery Clustering

Clustering columns must be top-level, non-repeated columns of one of the following types:

- DATE
- BOOL
- GEOGRAPHY
- INT64
- NUMERIC
- BIGNUMERIC
- STRING
- TIMESTAMP
- DATETIME

You can specify up to four clustering columns.

When using STRING type columns for clustering, BigQuery uses only the first 1,024 characters to cluster the data. The values in the columns can themselves be longer than 1,024

Daily partitioning is the default partitioning type. Daily partitioning is a good choice when your data is spread out over a wide range of dates, or if data is continuously added over time.

Choose hourly partitioning if your tables have a high volume of data that spans a short date range — typically less than six months of timestamp values. If you choose hourly partitioning, make sure the partition count stays within the partition limits.

Choose monthly or yearly partitioning if your tables have a relatively small amount of data for each day, but span a wide date range. This option is also recommended if your workflow requires frequently updating or adding rows that span a wide date range (for example, more than 500 dates). In these scenarios, use monthly or yearly partitioning along with clustering on the partitioning column to achieve the best performance.

When to use Clustering

Use clustering under the following circumstances:

- You don't need strict cost guarantees before running the query.
- You need more granularity than partitioning alone allows. To get clustering benefits in addition to partitioning benefits, you can use the same column for both partitioning and clustering.
- Your queries commonly use filters or aggregation against multiple particular columns.
- The cardinality of the number of values in a column or group of columns is large.

Introduction - Clustering

Prefer Clustering over Partitioning

Prefer clustering over partitioning under the following circumstances:

- Partitioning results in a small amount of data per partition (approximately less than 1 GB).
- Partitioning results in a large number of partitions beyond the limits on partitioned tables.
- Partitioning results in your mutation operations modifying most partitions in the table frequently (for example, every few minutes).

Introduction - Clustering

BigQuery Clustering - Best Practices

A sample table with partitioning and clustering

```
CREATE TABLE `mydataset.ClusteredSalesData`
PARTITION BY
                DATE(timestamp)
CLUSTER BY
                customer_id,
                product_id,
                order_id
AS
SELECT * FROM   `mydataset.SalesData`
```


[Clustering - Best Practices]

BigQuery Clustering - Best Practices

Filter clustered columns by sort order

```
SELECT
  SUM(totalSale)
FROM
  `mydataset.ClusteredSalesData`
WHERE
  customer_id = 10000
  AND product_id LIKE
  'gcp_analytics%'
```

```
SELECT
  SUM(totalSale)
FROM
  `mydataset.ClusteredSalesData`
WHERE
  product_id LIKE 'gcp_analytics%'
  AND order_id = 20000
```

The query on the left includes a filter expression that filters on customer_id and then on product_id. This query optimizes performance by filtering the clustered columns in sort order—the column order given in the CLUSTER BY clause.

The query on the right does not filter the clustered columns in sort order. As a result, the performance of the query is not optimal. This query filters on product_id then on order_id (skipping customer_id).

BigQuery Clustering - Best Practices

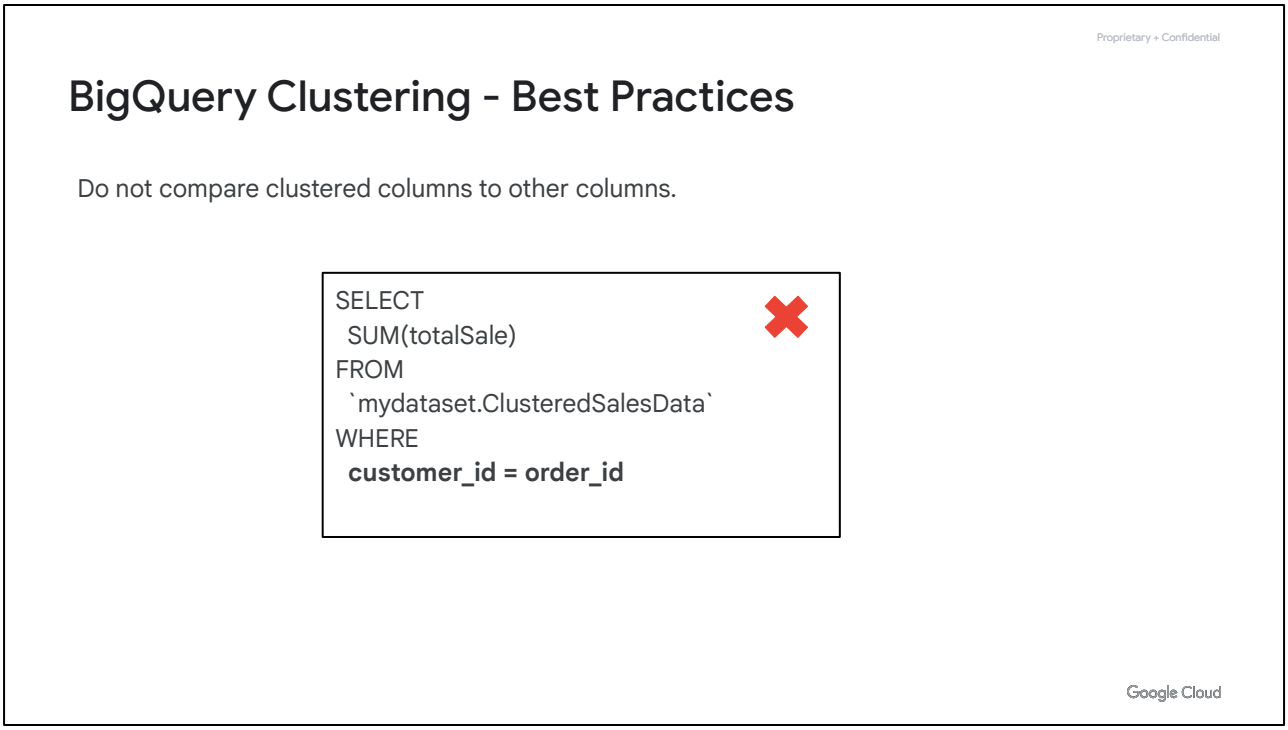
Do not use clustered columns in complex filter expressions.

```
SELECT
 SUM(totalSale)
FROM
 `mydataset.ClusteredSalesData`
WHERE
 CAST(customer_id AS STRING) = "10000"
```

[Clustering - Best Practices]

If you use a clustered column in a complex filter expression, the performance of the query is not optimized because block pruning cannot be applied.

For example, the above query will not prune blocks because a clustered column—customer_id—is used in a function in the filter expression.



[Clustering - Best Practices]

If a filter expression compares a clustered column to another column (either a clustered column or a non-clustered column), the performance of the query is not optimized because block pruning cannot be applied.

The following query does not prune blocks because the filter expression compares a clustered column—customer_id to another column—order_id.

By our next meeting

1. Continue to review the exam guide
2. Labs and Quests for this week:
   a. **Course:** Serverless Data Processing with Dataflow: Foundations
   b. **Course:** Serverless Data Processing with Dataflow: Develop Pipelines
3. Review the recommended materials shared during this workshop
4. Labs and Quests for next week: (or review the content):
   a. **Course:** Serverless Data Processing with Dataflow: Operations
   b. **Quest:** Perform Foundational Data, ML and AI Tasks - Skill Badge

Week 1
Course: Google Cloud Big Data and Machine Learning Fundamentals
https://partner.cloudskillsboost.google/course_templates/3

Quest: Create and Manage Cloud Resources (this is an introductory quest) - Skill Badge
https://partner.cloudskillsboost.google/quests/120

Course: Modernizing Data Lakes & Data Warehouses with Google Cloud
https://partner.cloudskillsboost.google/course_templates/54

Week 2
Course: Building Batch Data Pipelines on Google Cloud
https://partner.cloudskillsboost.google/course_templates/53

Week 3
Course: Building Resilient Streaming Analytics Systems on GCP
https://partner.cloudskillsboost.google/course_templates/52

Course: Smart Analytics, Machine Learning, and AI on GCP
https://partner.cloudskillsboost.google/course_templates/55

Week 4
Course: Serverless Data Processing with Dataflow: Foundations

https://partner.cloudskillsboost.google/course_templates/218

Course: Serverless Data Processing with Dataflow: Develop Pipelines
https://partner.cloudskillsboost.google/course_templates/229

Week 5
Course: Serverless Data Processing with Dataflow: Operations
https://partner.cloudskillsboost.google/course_templates/264

Quest: Perform Foundational Data, ML and AI Tasks - Skill Badge
https://partner.cloudskillsboost.google/quests/117

Week 6
Lab: Optimizing BigQuery for Cost and Performance v1.5
https://partner.cloudskillsboost.google/focuses/18091?parent=catalog

Quest: Build and Optimize Data Warehouses with BigQuery - Skill Badge
https://partner.cloudskillsboost.google/quests/147

Lab: ETL Processing on Google Cloud Using Dataflow and BigQuery
https://partner.cloudskillsboost.google/focuses/11581?parent=catalog

Week 7
Quest: Engineer data in Google Cloud - Skill Badge
https://partner.cloudskillsboost.google/quests/132

Course: Preparing for the Google Cloud Professional Data Engineer Exam
https://partner.cloudskillsboost.google/course_templates/72

Practice: Professional Data Engineer Sample Questions
https://cloud.google.com/certification/practice-exam/data-engineer

Thank you

Google Cloud