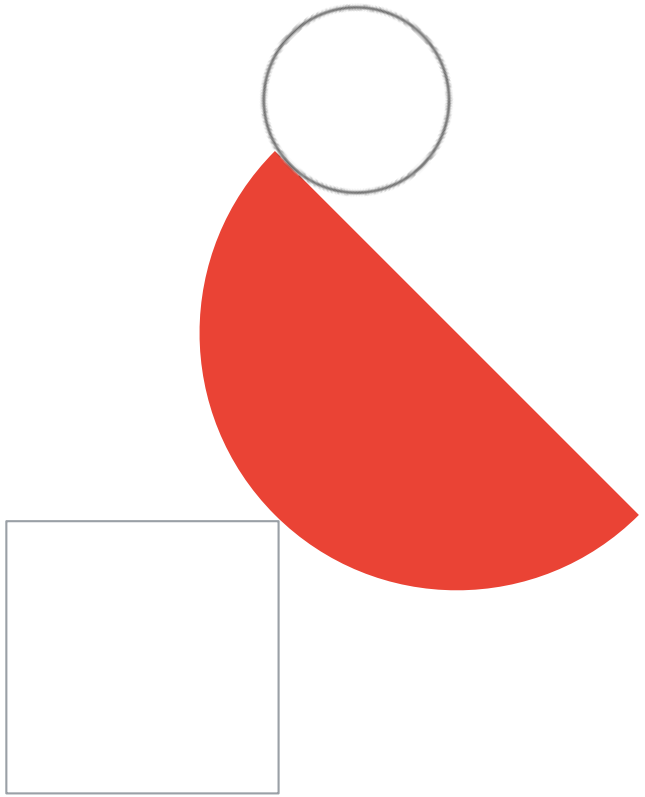
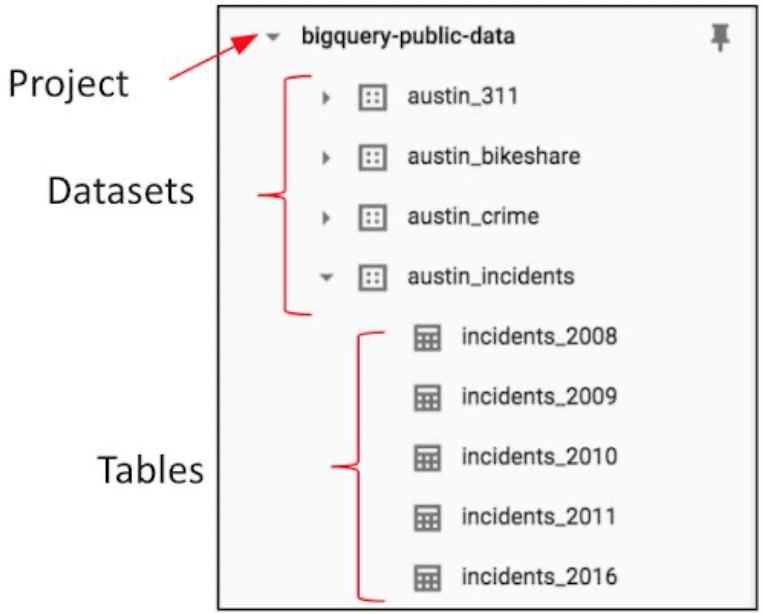
Bigquery



BigQuery hierarchy

Project -> Dataset -> Tables (-> Partitions)

- For each query, BigQuery executes a full-column scan.
- BigQuery performance and query costs are based on the amount of data scanned.
- You can set the geographic location of a Dataset at creation time only.
- All tables that are referenced in a query must be stored in datasets in the same location.
- When you copy a table (bq cp), the datasets that contain the source table and destination table must reside in the same location.
  - You can copy a dataset (NOT with bq cp, but with BigQuery Data Transfer Service) within a region or from one region to another
- Dataset names are case-sensitive



BigQuery: Controlling access to datasets

Common BigQuery predefined roles

Exam Tips: *It's a common practice to have a Dataset in one project and perform queries from another one (split billing!).*

Admin	Full Access to all datasets
Data Editor	Access to edit all contents of the datasets
Data Owner	Full access to datasets and all of their contents
Data Viewer	Access to view datasets and all of their contents
Job User	Access to run jobs
Metadata Viewer	Access to view table and dataset metadata
User	Access to run queries and create datasets
Read Sessions User	Access to create and use read sessions

Capability	dataViewer	dataEditor	dataOwner	user	jobUser	admin
List/get projects	✓	✓	✓	✓	✓	✓
List tables	✓	✓	✓	✓	⊘	✓
Get table data/metadata	✓	✓	✓	⊘	⊘	✓
Create tables	⊘	✓	✓	⊘	⊘	✓
Modify/delete tables	⊘	✓	✓	⊘	⊘	✓
List/get datasets	✓	✓	✓	✓	⊘	✓
Create new datasets	⊘	✓	✓	✓	⊘	✓
Modify/delete datasets	⊘	⊘	✓	Self-created datasets	⊘	✓
Create jobs/queries	⊘	⊘	⊘	✓	✓	✓
Cancel jobs	⊘	⊘	⊘	Self-created jobs	Self-created jobs	Any jobs
Get/list saved queries	⊘	⊘	⊘	✓	⊘	✓
Create/update/delete saved queries	⊘	⊘	⊘	⊘	⊘	✓
Get transfers	⊘	⊘	⊘	✓	⊘	✓
Create/update/delete transfers	⊘	⊘	⊘	⊘	⊘	✓

BigQuery: Controlling access to datasets

You can grant access at the following BigQuery resource levels:

- organization or Google Cloud project level
- dataset level
- table or view level
  a. Authorized Views
- You can also restrict access to data on more granular level by using the following methods:
  a. column-level access control
  b. dynamic data masking (aka “some columns may be hidden, depending on privileges”)
     i. Works together with column-level security.
     ii. no need to modify existing queries by excluding the columns that the user cannot access
  c. row-level security (aka “some rows may be hidden, depending on privileges”)
     i. One table can have multiple row-level access policies. Row-level access policies can coexist on a table with column-level security as well as dataset-level, table-level, and project-level access controls.

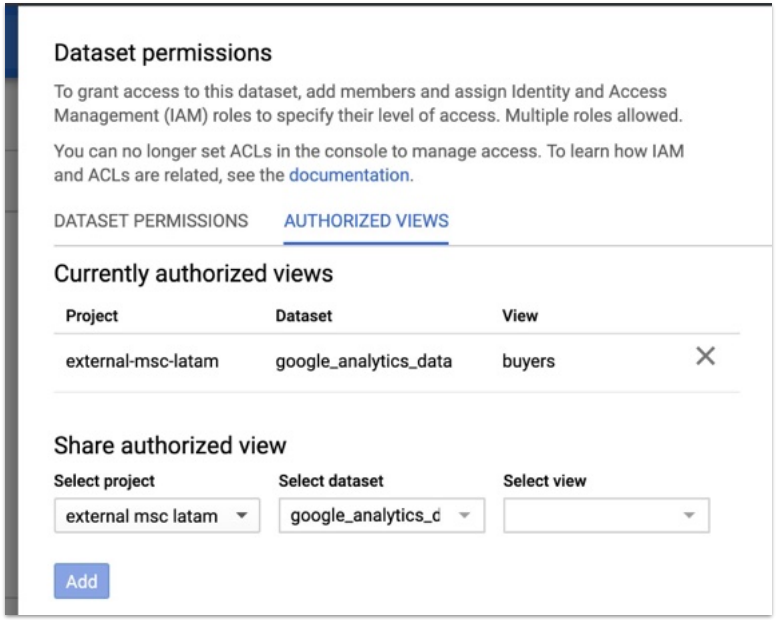
BigQuery: Controlling access to datasets

Authorized Views

1. **View:** View is a virtual table defined by a SQL query. When you create a view, you query it in the same way you query a table

2. **Query:** When a user queries the view, the query results contain data only from the tables and fields specified in the query that defines the view.

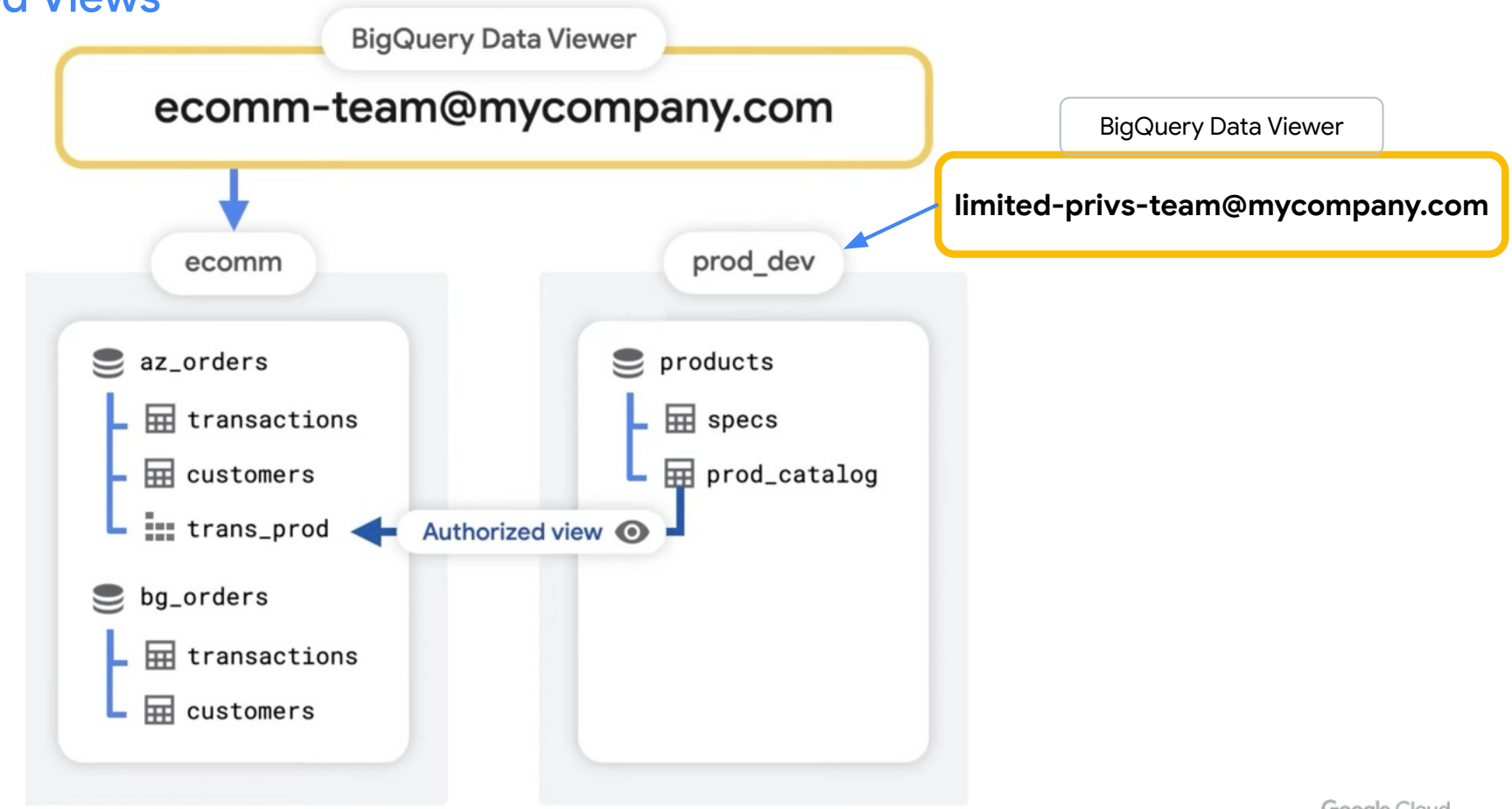
3. **Authorized Views:** An authorized view allows you to share query results with particular users and groups without giving them access to the underlying tables.



Exam Tip: *Authorized Views were especially useful when there were no table/column-level permissions. However, they're still often-used way to selectively share access to datasets (and they pop up on the exam!).*
MAKE SURE TO UNDERSTAND HOW TO CREATE AND SHARE SUCH A VIEW.

BigQuery: Controlling access to datasets

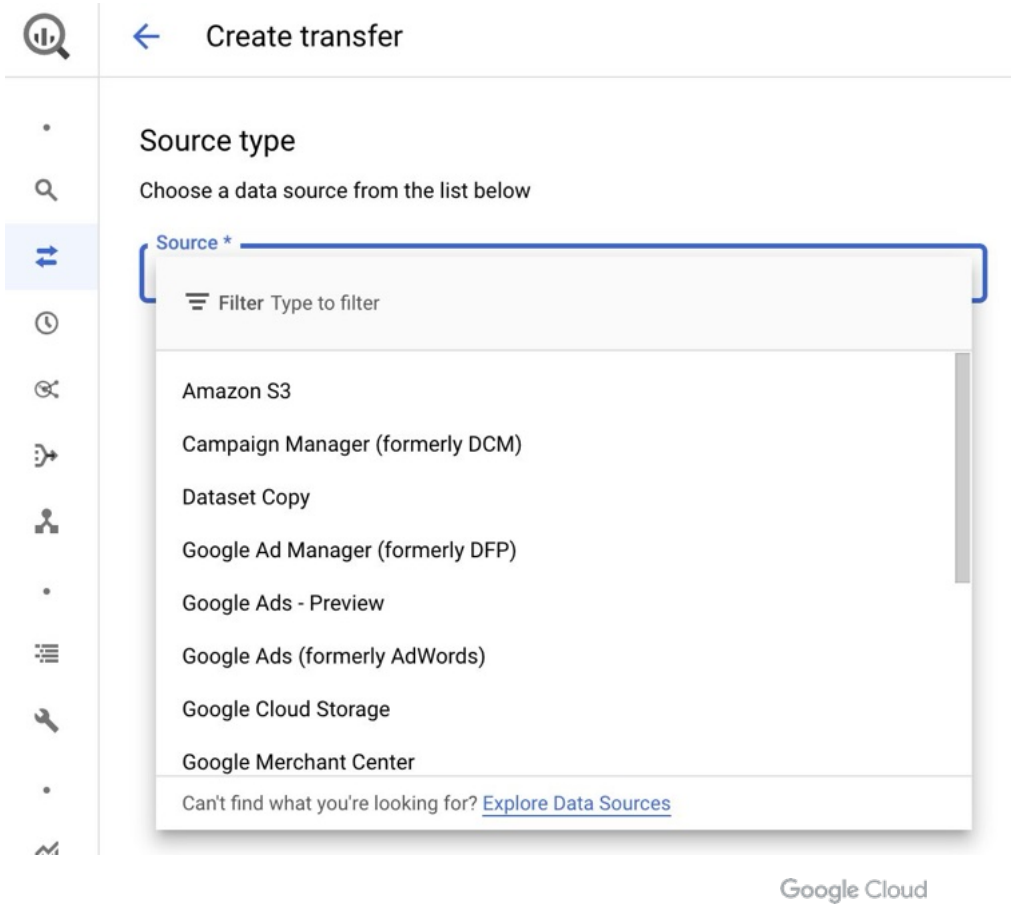
Authorized Views



BigQuery - Data Transfer Service

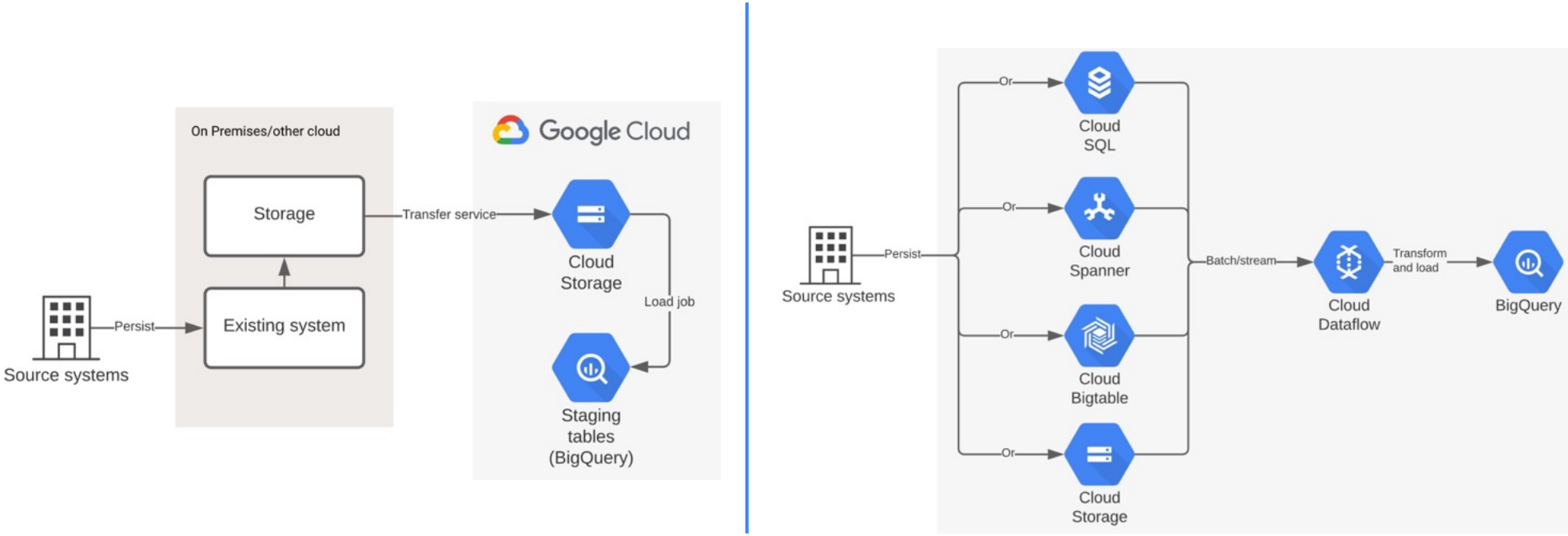
Mostly useful for regular data transfers to BigQuery

- BigQuery Data Transfer Service automates data movement from various sources into BigQuery on a scheduled, managed basis.
- You can initiate data backfills to recover from any outages or gaps.



BigQuery - Batch vs Streaming inserts

Most common architectures



Exam Tip: *There is additional cost for streaming (both inserts and reads) in BigQuery.*

BigQuery: Controlling access to datasets

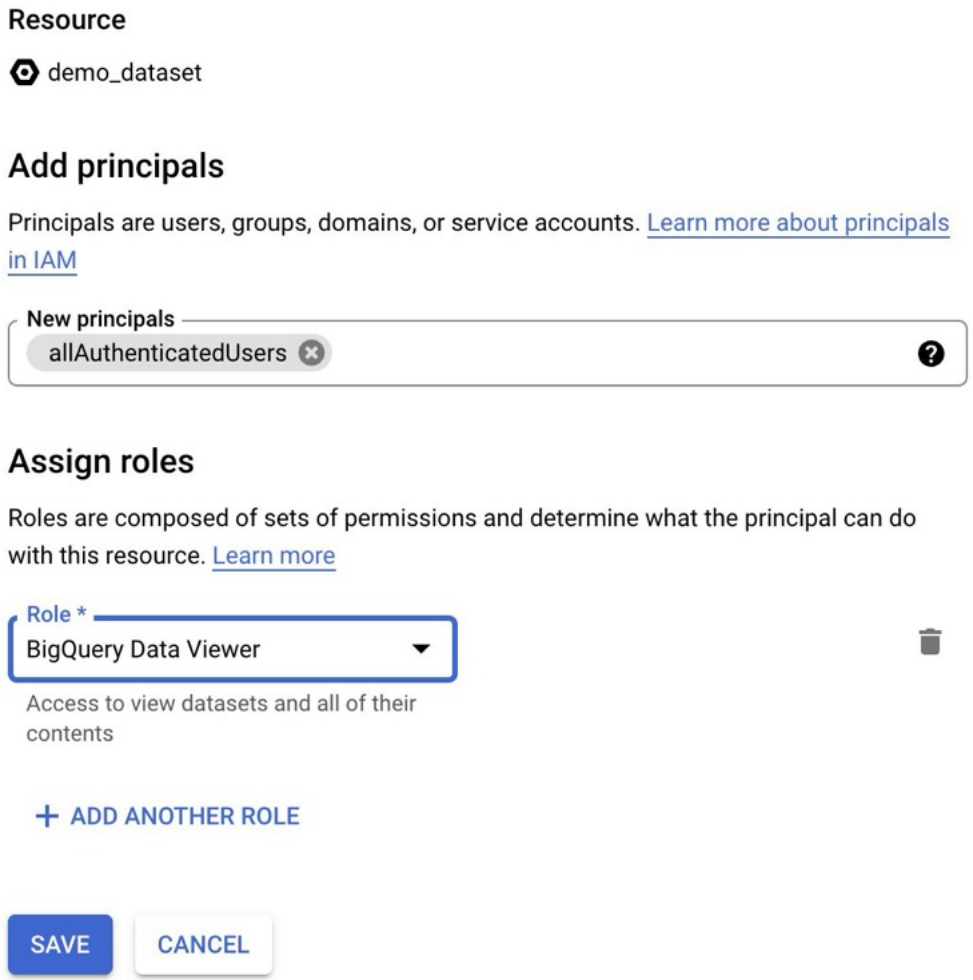
You can grant access at the following BigQuery resource levels:

- organization or Google Cloud project level
- dataset level
- table or view level
  a. Authorized Views
- You can also restrict access to data on more granular level by using the following methods:
  a. column-level access control
  b. dynamic data masking (aka “some columns may be hidden, depending on privileges”)
     i. Works together with column-level security.
     ii. no need to modify existing queries by excluding the columns that the user cannot access
  c. row-level security (aka “some rows may be hidden, depending on privileges”)
     i. One table can have multiple row-level access policies. Row-level access policies can coexist on a table with column-level security as well as dataset-level, table-level, and project-level access controls.

BigQuery: Sharing Datasets with others

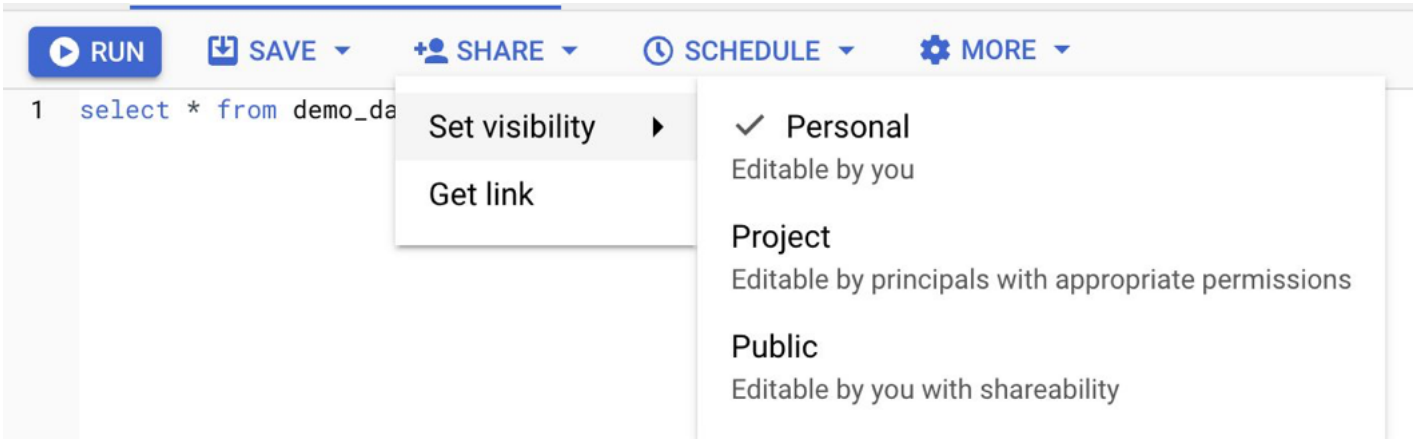
AllAuthenticatedUsers

The special setting allAuthenticatedUsers makes a dataset public. Authenticated users must use BigQuery within their own project and have access to run BigQuery jobs so that they can query the Public Dataset. The billing for the query goes to their project, even though the query is using public or shared data. In summary, the cost of a query is always assigned to the active project from where the query is executed.



BigQuery: Sharing Queries with others

Mostly for collaboration

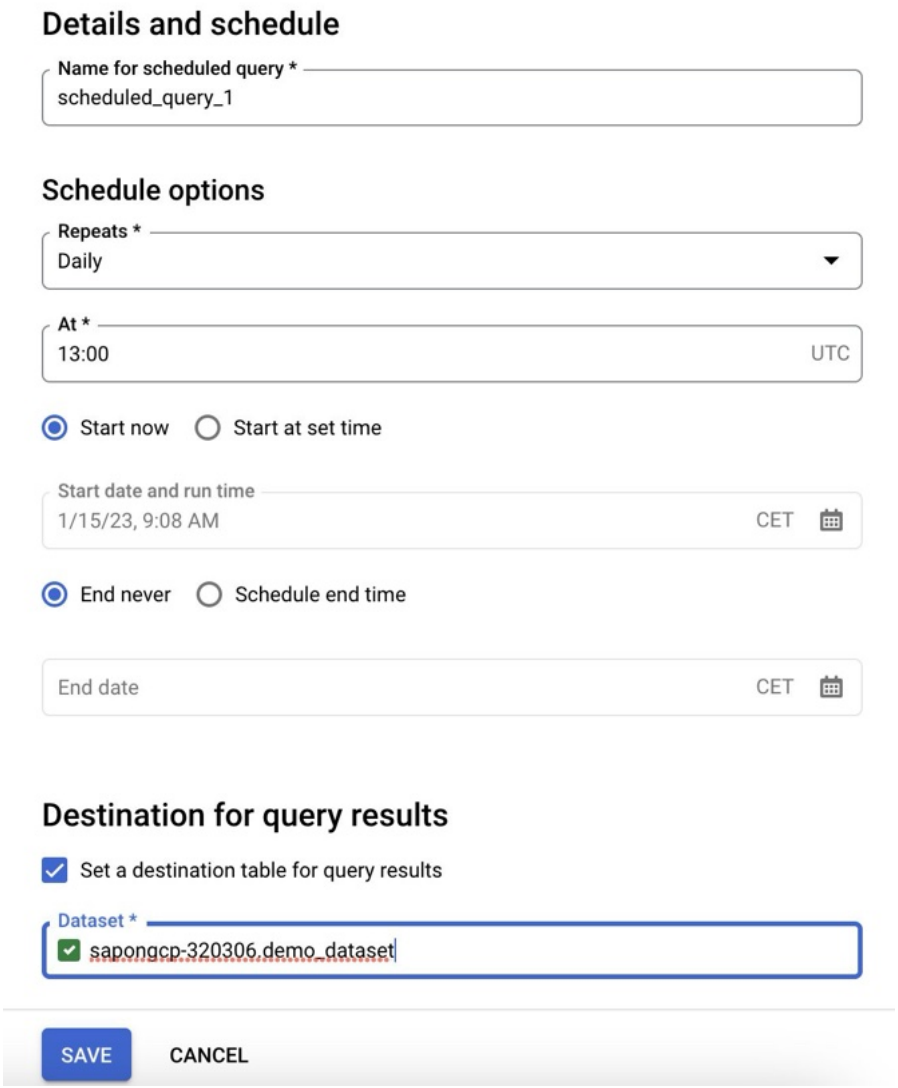


- Query needs to be saved first, before it's shared;
- Can share incomplete / invalid queries -> collaboration;
- Project-level saved queries are visible to principals with the required permissions;
- Public saved queries are visible to anyone with a link to the query;

BigQuery: Scheduling queries

Mostly useful for regular execution

- Scheduled queries use features of BigQuery Data Transfer Service.
- If the destination table for your results doesn't exist when you set up the scheduled query, BigQuery attempts to create the table for you.
- You can set up a scheduled query to authenticate as a service account.



BigQuery: Query results caching

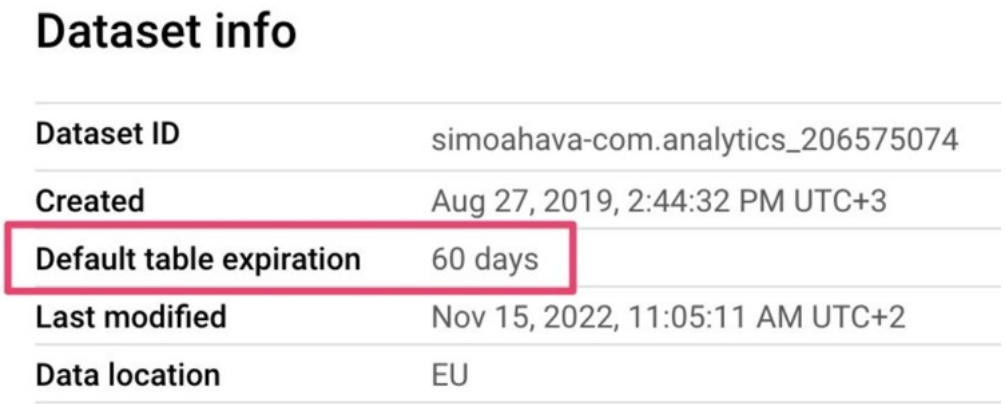
Limit Access by Data Lifecycle Stages

- Query results are cached to improve performance and reduce costs for repeated queries
- Cache is per user
- Still subject to quota policies
- Cache results have a size limit of 128 MB compressed
- No charge for queries that use cached results
- Results are cached for approximately 24 hours
- Lifetime extended when a query returns a cached result
- Use of cached results can be turned off (useful for benchmarking)

BigQuery: table/partition (automatic) data expiration

Can be set for dataset / table / partition

Best practice for data lifecycle management. Expiration in BigQuery automatically implements retention policy.



- Dataset expiration
  - = “default table expiration time” for a dataset
- Table expiration
  - If Dataset expiration is set, each table inherits this setting by default
- Partition expiration:
  - The setting applies to all partitions in the table, but is calculated independently for each partition based on the partition time.
  - At any point after a table is created, you can update the table's partition expiration

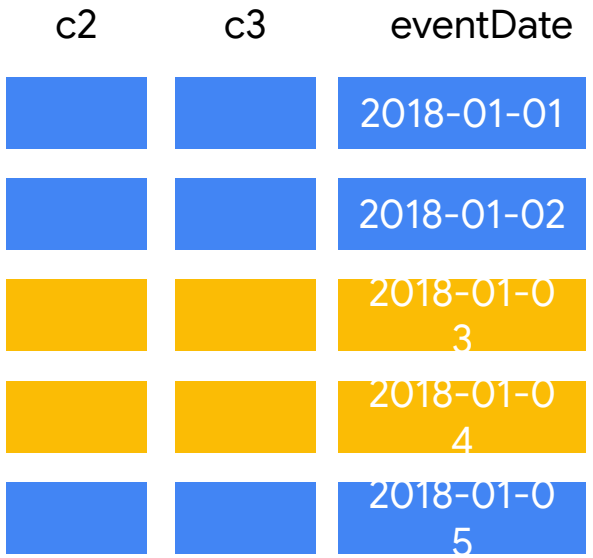
BigQuery: Table Partitioning

Partitioning versus sharding:

- Table sharding is the practice of storing data in multiple tables, using a naming prefix such as [PREFIX]_YYYYMMDD. Partitioning is recommended over table sharding, because partitioned tables perform better.

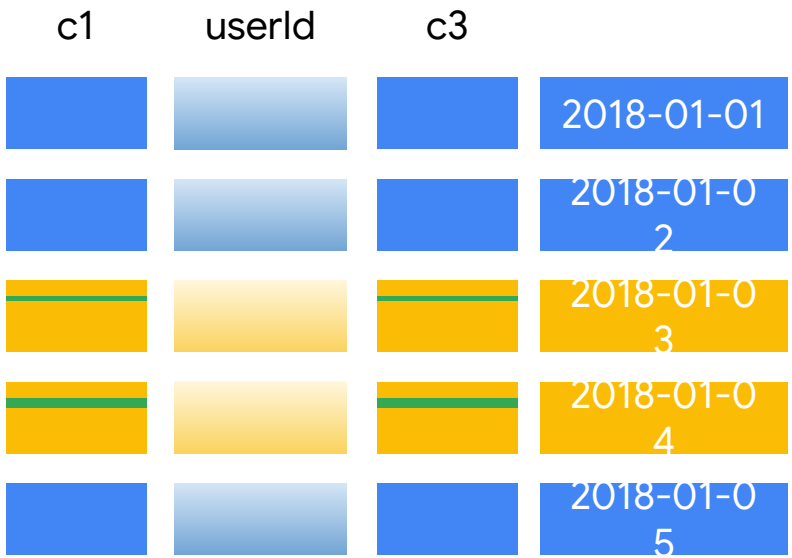
You can partition BigQuery tables by:

- 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.




```
SELECT * FROM ...
WHERE eventDate BETWEEN
“2018-01-03” AND
“2018-01-04”
```

BigQuery: Table Clustering



SELECT c1, c3 FROM ... WHERE userId BETWEEN 52 and 63
AND eventDate BETWEEN “2018-01-03” AND “2018-01-04”

BigQuery - table partitioning vs clustering

Decision making

- Clustering gives you more granularity than partitioning alone allows
- Use clustering if your queries commonly use filters or aggregation against multiple particular columns.

Use case	Recommendation
You're using on-demand pricing and require strict cost guarantees before running queries.	Partitioned tables
Your segment size is less than 1 GB after partitioning the table.	Clustered tables
You require a large number of partitions beyond the BigQuery limits	Clustered tables
Frequent mutations in your data modify a large number of partitions.	Clustered tables
You frequently run queries to filter data on certain fixed columns.	Partitions plus clustering

BigQuery: table partitioning AND clustering

Both partitioning and clustering can improve performance and reduce query cost

Orders table
Not Clustered; Not partitioned

Order_Date	Country	Status
2022-08-02	US	Shipped
2022-08-04	JP	Shipped
2022-08-05	UK	Canceled
2022-08-06	KE	Shipped
2022-08-02	KE	Canceled
2022-08-05	US	Processing
2022-08-04	JP	Processing
2022-08-04	KE	Shipped
2022-08-06	UK	Canceled
2022-08-02	UK	Processing
2022-08-05	JP	Canceled
2022-08-06	UK	Processing
2022-08-05	US	Shipped
2022-08-06	JP	Processing
2022-08-02	KE	Shipped
2022-08-04	US	Shipped

Orders table
Clustered by Country; Not partitioned

Order_Date	Country	Status
2022-08-04	JP	Shipped
2022-08-04	JP	Processing
2022-08-05	JP	Canceled
2022-08-06	JP	Processing
2022-08-06	KE	Shipped
2022-08-02	KE	Canceled
2022-08-04	KE	Shipped
2022-08-04	KE	Shipped
2022-08-02	KE	Shipped
2022-08-05	UK	Processing
2022-08-06	UK	Canceled
2022-08-02	UK	Canceled
2022-08-06	UK	Processing
2022-08-02	US	Shipped
2022-08-05	US	Processing
2022-08-05	US	Shipped
2022-08-04	US	Shipped

Exam Tip: *You can combine partitioning with clustering. Data is first partitioned and then data in each partition is clustered by the clustering columns.*

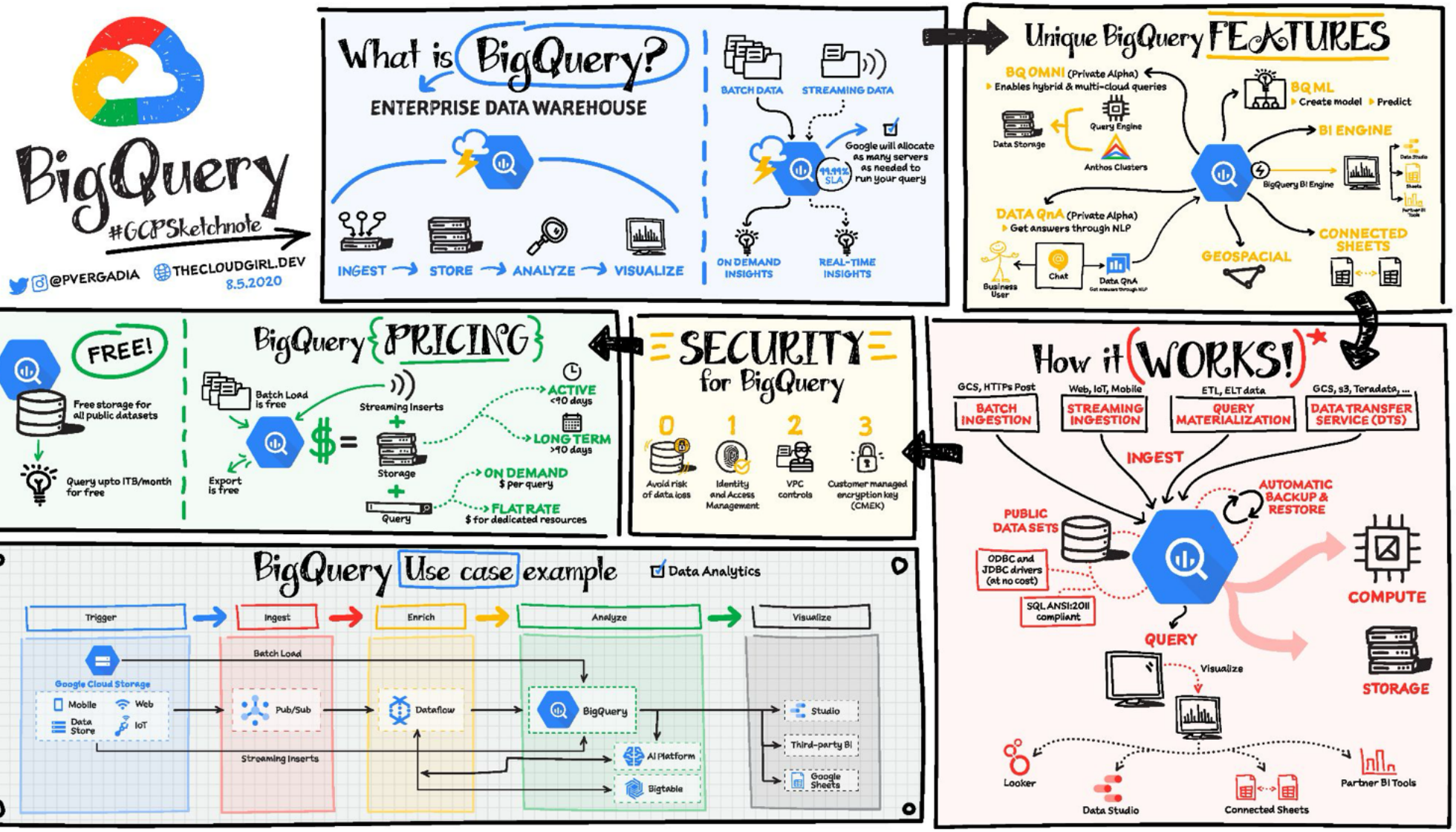
Orders table
Clustered by Country; Partitioned by Order_Date (Daily)

	Order_Date	Country	Status
Partition: 2022-08-02	2022-08-02	KE	Shipped
	2022-08-02	KE	Canceled
Clusters: Country	2022-08-02	UK	Processing
	2022-08-02	US	Shipped

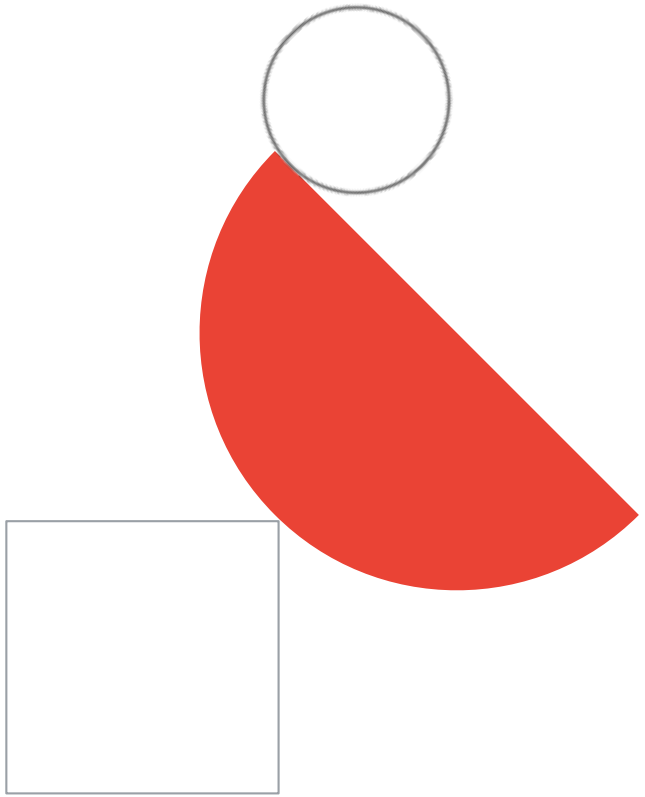
	Order_Date	Country	Status
Partition: 2022-08-04	2022-08-04	JP	Shipped
	2022-08-04	JP	Processing
Cluster: Country	2022-08-04	KE	Shipped
	2022-08-04	US	Shipped

	Order_Date	Country	Status
Partition: 2022-08-05	2022-08-05	JP	Canceled
	2022-08-05	UK	Canceled
Cluster: Country	2022-08-05	US	Shipped
	2022-08-05	US	Processing

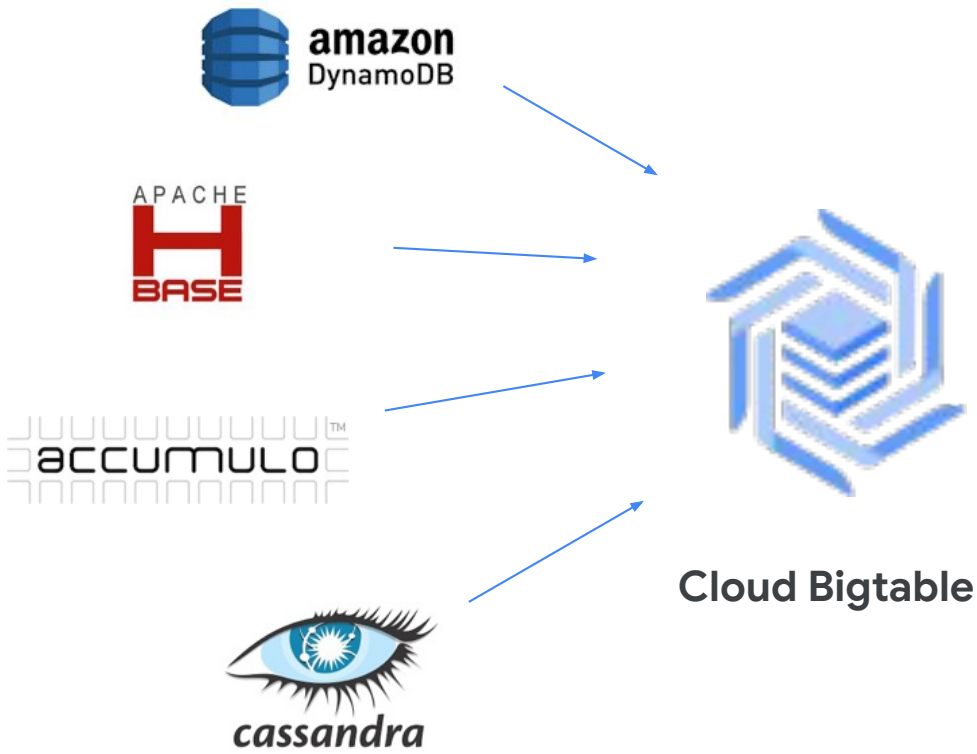
	Order_Date	Country	Status
Partition: 2022-08-06	2022-08-06	JP	Processing
	2022-08-06	KE	Shipped
Cluster: Country	2022-08-06	UK	Canceled
	2022-08-06	UK	Processing


BigQuery
#GCPSketchnote
@PVERGADIA
THECLOUDGIRL.DEV
8.5.2020
What is BigQuery?
ENTERPRISE DATA WAREHOUSE
INGEST
STORE
ANALYZE
VISUALIZE
BATCH DATA
STREAMING DATA
Google will allocate as many servers as needed to run your query
99.9% SLA
ON DEMAND INSIGHTS
REAL-TIME INSIGHTS
Unique BigQuery FEATURES
BQ OMNI (Private Alpha)
Enables hybrid & multi-cloud queries
Data Storage
Query Engine
Anthos Clusters
BQ ML
Create model
Predict
BI ENGINE
BigQuery BI Engine
DATA QnA (Private Alpha)
Get answers through NLP
Business User
Chat
Data QnA
GEOSPACIAL
CONNECTED SHEETS
How it WORKS!
GCS, HTTPs Post
BATCH INGESTION
Web, IoT, Mobile
STREAMING INGESTION
ETL, ELT data
QUERY MATERIALIZATION
GCS, s3, Teradata, ...
DATA TRANSFER SERVICE (DTS)
INGEST
AUTOMATIC BACKUP & RESTORE
PUBLIC DATA SETS
ODBC and JDBC drivers (at no cost)
SQL ANSI:2011 compliant
COMPUTE
STORAGE
QUERY
Visualize
Looker
Data Studio
Connected Sheets
Partner BI Tools
SECURITY for BigQuery
0
Avoid risk of data loss
1
Identity and Access Management
2
VPC controls
3
Customer managed encryption key (CMEK)
BigQuery PRICING
FREE!
Free storage for all public datasets
Query upto 1TB/month for free
Batch Load is free
Streaming Inserts
Export is free
Storage
ACTIVE <90 days
LONG TERM >90 days
Query
ON DEMAND $ per query
FLAT RATE $ for dedicated resources
BigQuery Use case example
Data Analytics
Trigger
Ingest
Enrich
Analyze
Visualize
Google Cloud Storage
Mobile
Web
Data Store
IoT
Batch Load
Pub/Sub
Dataflow
BigQuery
AI Platform
Bigtable
Streaming Inserts
Studio
Third-party BI
Google Sheets

Bigtable



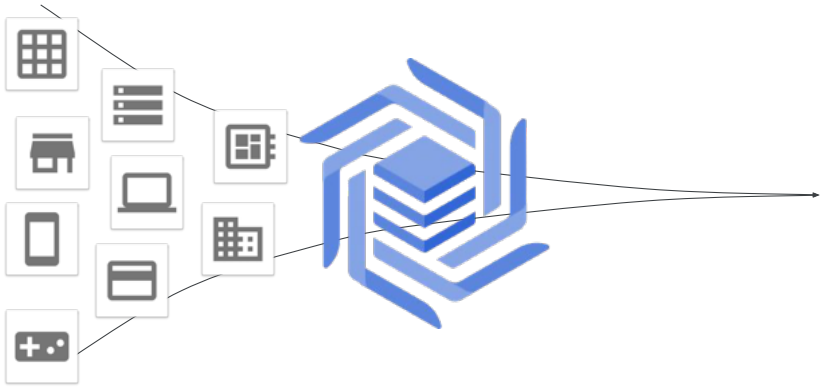
Bigtable is a common migration target for key-value, wide-column and time-series databases



- **Petabyte-scale**
- **fully managed** NoSQL database service for use cases where **low latency** random data access, **scalability** and **reliability** are critical.
- **scales seamlessly**
- **integrates** with the Apache® ecosystem and supports the HBase™ API.

Bigtable is all about speeeeed

High throughput



10MB/s write throughput per node

Up to **220MB/s** scan throughput

10.000 queries (r/w) **per second per node**

Low latency

Consistent 99th percentile **low single digit** read and write latency

What is Bigtable good for?

Use Case Examples

- **Time-series** data, such as CPU and memory usage over time for multiple servers.
- **Marketing data**, such as purchase histories and customer preferences.
- **Financial data**, such as transaction histories, stock prices, and currency exchange rates.
- **Internet of Things** data, such as usage reports from energy meters and home appliances.
- **Graph data**, such as information about how users are connected to one another.

Applications that need...

- Very high throughput
- Scalability
- Non-Structured key/value data where each value is no larger than 10MB

Storage Engine

- Batch MapReduce
- Stream Processing/Analytics
- ML applications

Exam Tip: *types of apps where you'd consider using Bigtable: recommendation engines, personalizing user experience, Internet of Things, real-time analytics, fraud detection, migrating from HBase or Cassandra, Fintech, gaming, high-throughput data streaming for creating / improving ML models.*

What is Bigtable not good for?

Not good for...

- Not a relational database
- No SQL Queries or Joins
- No Multi-Row Transactions

Considerations

- You need full SQL support for OLTP
  → consider Spanner or CloudSQL
- Interactive querying for OLAP
  → consider BigQuery
- Need to store immutable blobs larger than 10MB (e.g. movies, images)
  → consider Cloud Storage

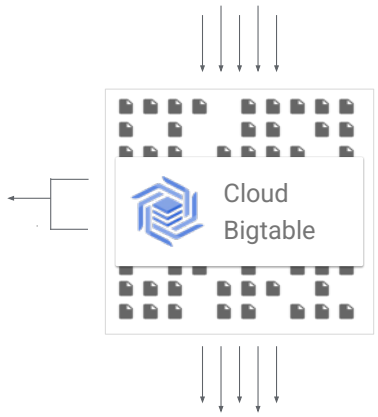
Bigtable for analytics... ?

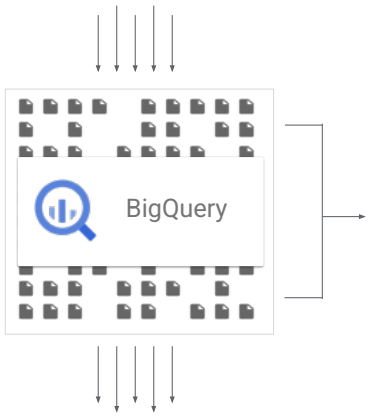
Bigtable vs BigQuery

NoSQL wide column database

Low latency per-entry access

Heavy read/write events





Enterprise data warehouse for relational structured data

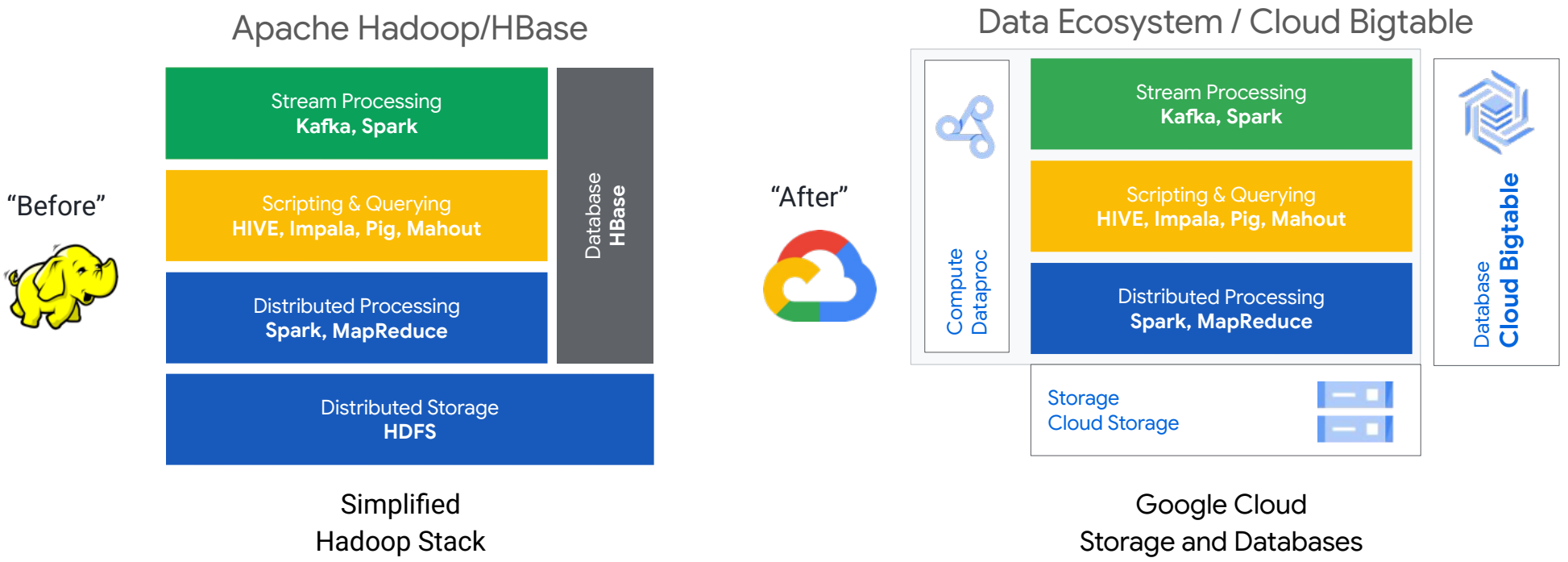
Large scale, ad-hoc SQL-based OLAP analysis

Organizational insights

Analyze data from Cloud Bigtable database

Optimized for	**Point read/write**	**Ad-hoc analysis and reporting**
Typical target	**User/entity level**	**Cohort /population level**

Bigtable: Hadoop migration and modernization



Exam Tip: *Main goal: decoupling of storage & compute. As a consequence, you can treat Dataproc clusters as job-specific / ephemeral*

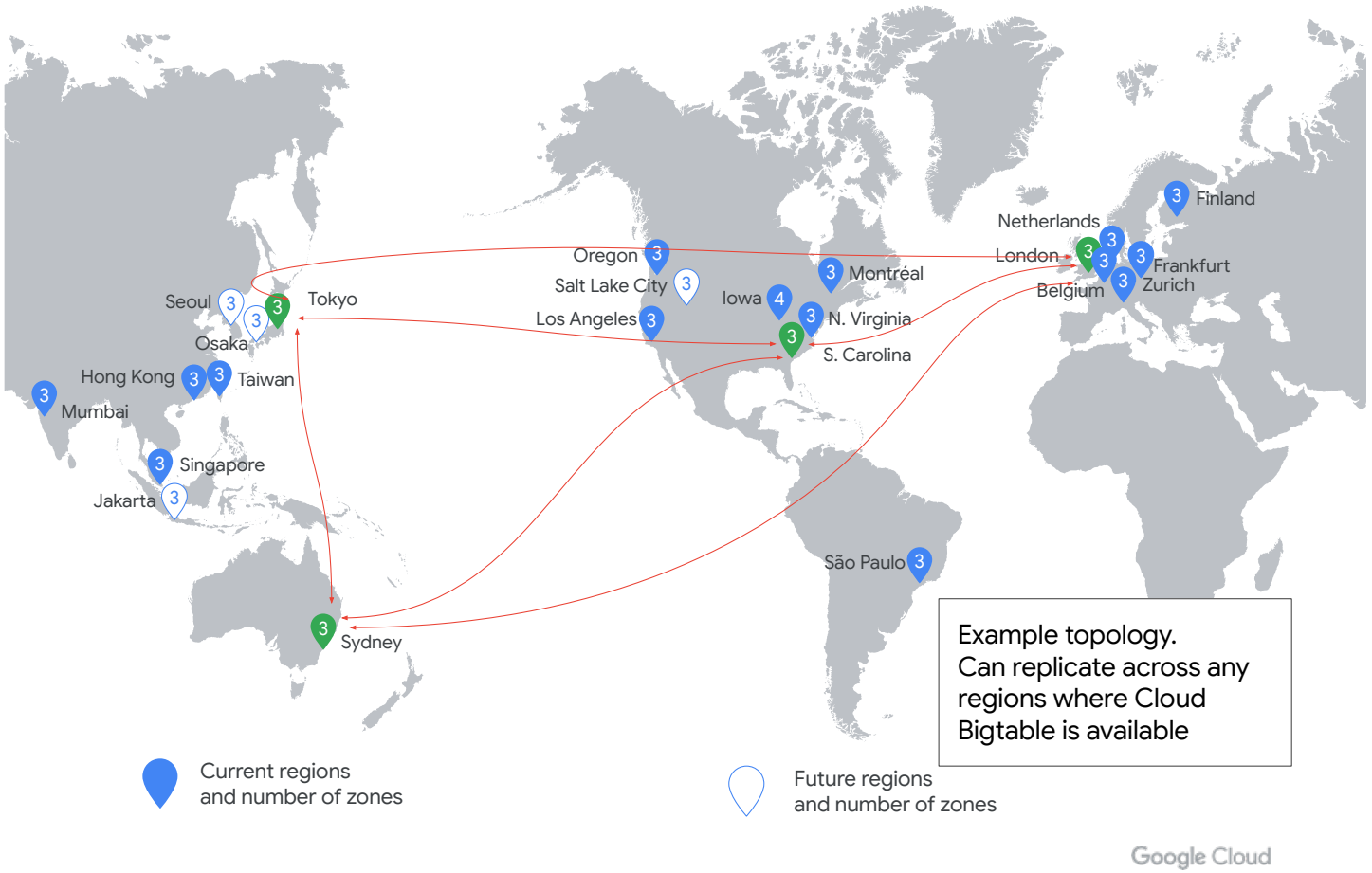
Bigtable: Replication

Regional replication

- SLA up to 99.999%
- Isolate serving and analytics
- Independently scale clusters
- Automatic failover in case of a zonal failure

Global replication

- **Increases durability/availability beyond one region**
- **Fastest region-specific access**
- **Option for DR replica for regulated customers**



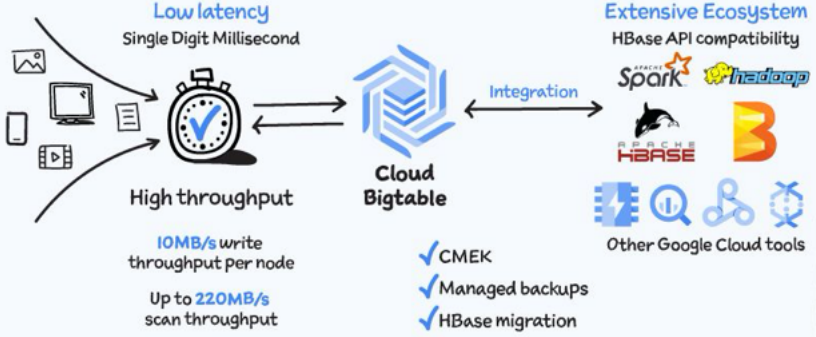
Cloud Bigtable

#GCPSketchnotes

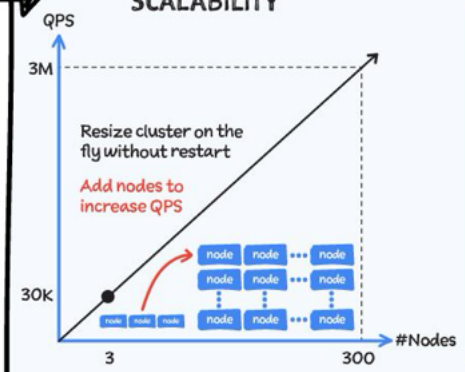
@PVERGADIA
THECLOUDGIRL.DEV 4.14.2021

What is Cloud Bigtable?

FULLY MANAGED PETABYTE-SCALE NOSQL DATABASE
For heavy reads/writes



SCALABILITY



Each node provides upto 10,000 QPS (read/write)

FAILOVER FOR HIGH AVAILABILITY (HA)

Single/multi-region/global Replication

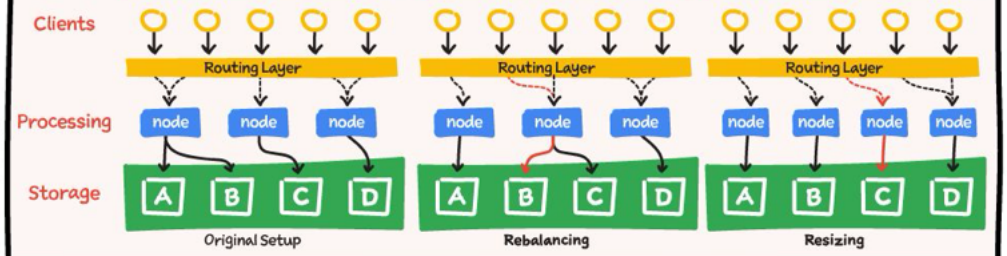
No manual steps to ensure consistency, repair data, or synchronize writes/deletes

How does Cloud Bigtable Optimize throughput?

Cloud Bigtable separates processing from storage, each node has access to a group of database rows

Rebalance load automatically to improve performance

Resize nodes (with no downtime) for best overall throughput



Cloud Bigtable «Use case examples»

REAL-TIME ANALYTICS FOR HUGE WORKLOADS

REAL-TIME AD BIDDING PLATFORM