BigQuery Scheduling and Workload Management

Scheduling Transfers

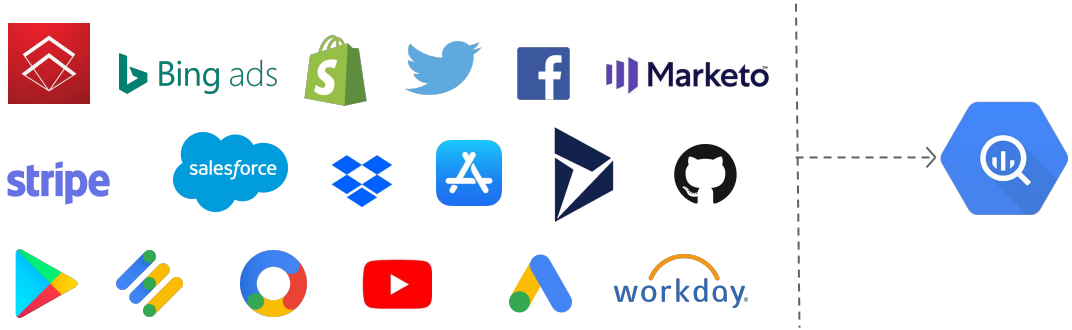
BigQuery | Data Transfer Service

The BigQuery Data Transfer Service (DTS) simplifies and automates data movement to BigQuery on a scheduled, managed basis.

With its flexible infrastructure, DTS is also used to power:

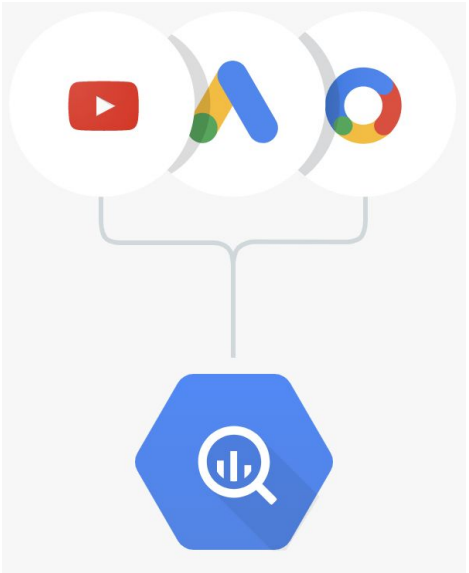
- BigQuery scheduled query
- GCS/S3 mirroring
- Recurring copy jobs
- Data warehouse migration
- 3rd-party data source transfers

100+ SaaS apps...



Data Transfer Service | Options

- Automatically transfers data from multiple sources (YouTube, Ads, Google Play, Teradata, AWS S3 & Redshift, GCS, BigQuery, etc.)
- Transfer jobs can be **scheduled** as Daily, Weekly, Monthly, Custom, or On Demand
- Provides **wildcard support** for Google Cloud Storage URIs
- Provides **run time parameterization** of source and destination paths
- Enables capturing transfer job run **notifications** via Pub/Sub Topic and failure notifications via Email
- Tracks multiple runs of the transfer jobs in the UI (Transfer Details and Job History) and provides detailed logs via Cloud Operations



Data Transfer Service | Terminologies

Data Source

Data source refers to
1) where DTS reads data from;
2) a DTS data source service listening to a Pub/Sub topic to receive and process Transfer Runs.

Most DTS data source services download data from the source to staging bucket on GCS.

Transfer Config

Transfer Config is created by the users with credentials, schedule, BigQuery destination and metadata required to schedule Transfer Runs.

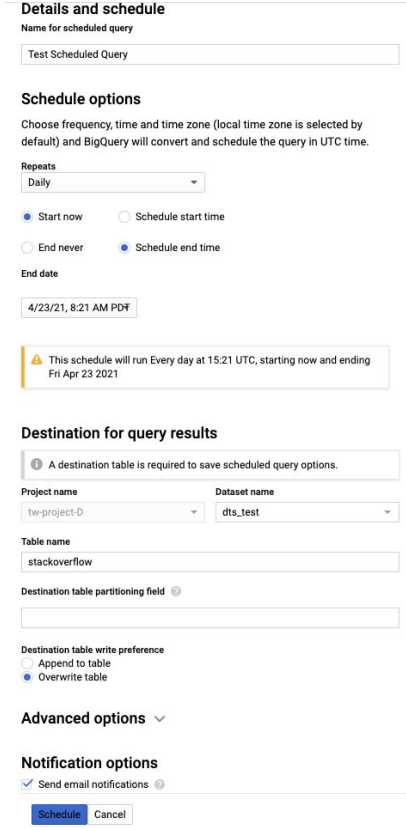
Transfer Run

Transfer Run is the scheduled job created by DTS from the corresponding Transfer Config. Transfer Run is sent to corresponding DTS data source service for processing.

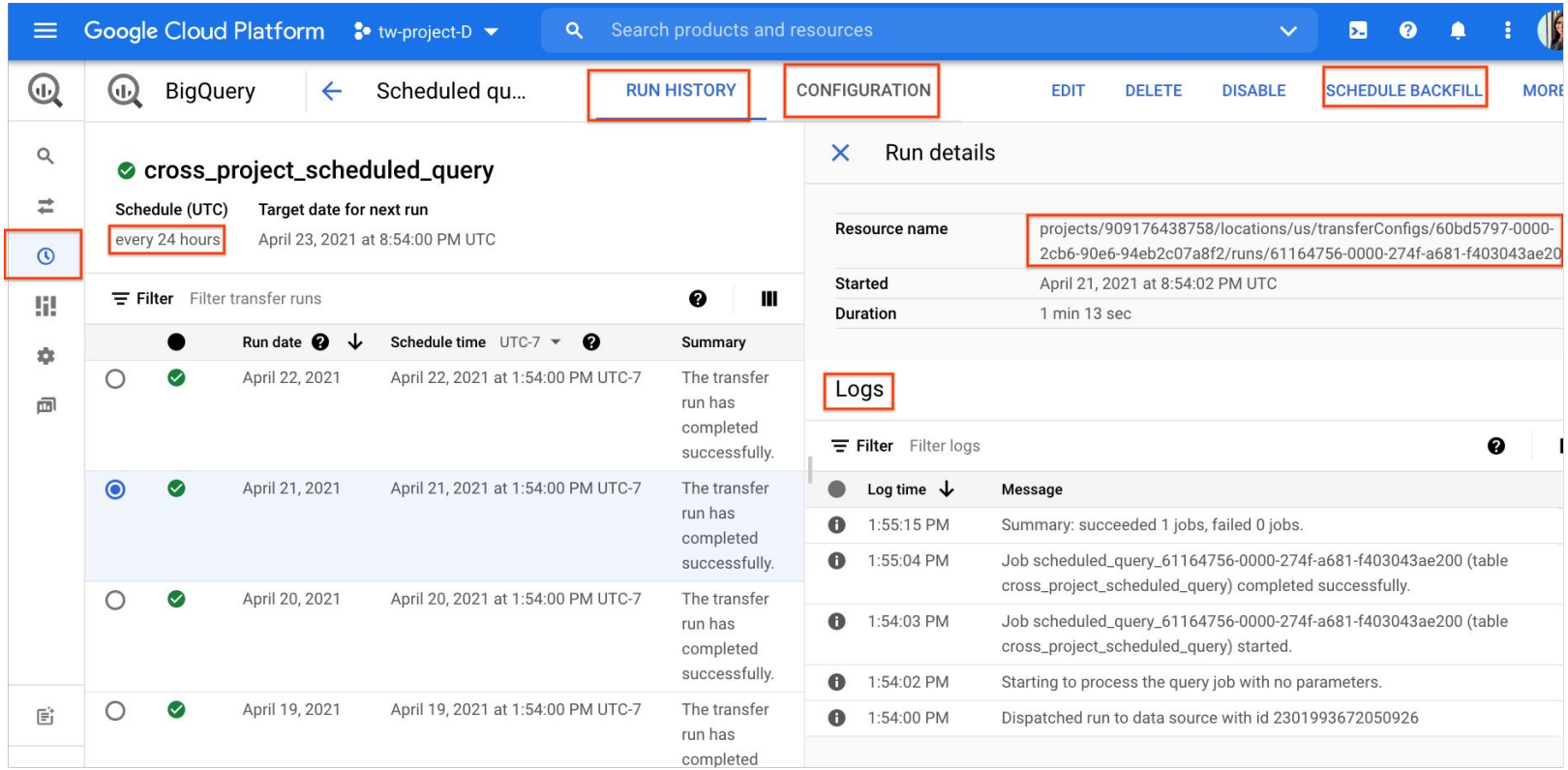
BigQuery | Scheduled Query

```
bq query \
--use_legacy_sql=false \
--project_id=<compute_project> \
--destination_table=mydataset.mytable \
--display_name='Test Scheduled Query' \
--schedule='every 24 hours' \
--replace=true \
'SELECT COUNT(*) FROM `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE start_station_name LIKE "%Broadway%"'
```

```
bq mk \
--transfer_config \
--project_id=<compute_project> \
--location=US \
--target_dataset=mydataset \
--display_name='Test Scheduled Query' \
--params='{"query":"SELECT COUNT(*) FROM
`bigquery-public-data.new_york_citibike.citibike_trips`
WHERE start_station_name LIKE '%Broadway%'"
,"destination_table_name_template":"mytable"
,"write_disposition":"WRITE_TRUNCATE"}' \
--data_source=scheduled_query
```



Scheduled Query | UI Options



Scheduled Query | Parameterization

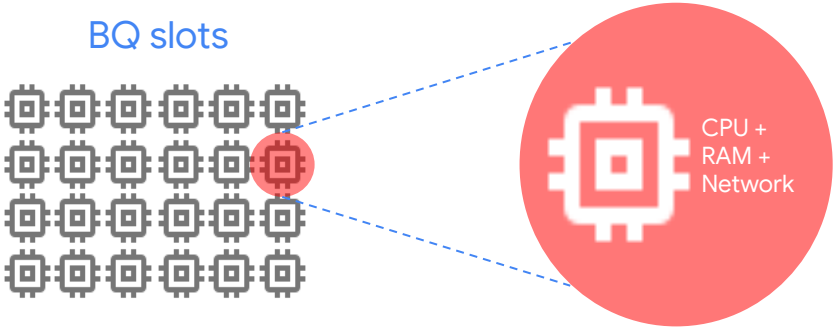
run_time (UTC)	Templated Parameter	Output destination table name
2018-02-15 00:00:00	mytable	mytable
2018-02-15 00:00:00	mytable_{**run_time**|"%Y%m%d"}	mytable_20180215
2018-02-15 00:00:00	mytable_{**run_time+25h**|"%Y%m%d"}	mytable_20180216
2018-02-15 00:00:00	mytable_{**run_time-1h**|"%Y%m%d"}	mytable_20180214
2018-02-15 00:00:00	mytable_{**run_time+1.5h**|"%Y%m%d%H"} or mytable_{**run_time+90m**|"%Y%m%d%H"}	mytable_2018021501
2018-02-15 00:00:00	{**run_time+97s**|"%Y%m%d"}_mytable_{**run_time+97s**|"%H%M%S"}	20180215_mytable_000137

BigQuery Slots

BigQuery | Slots

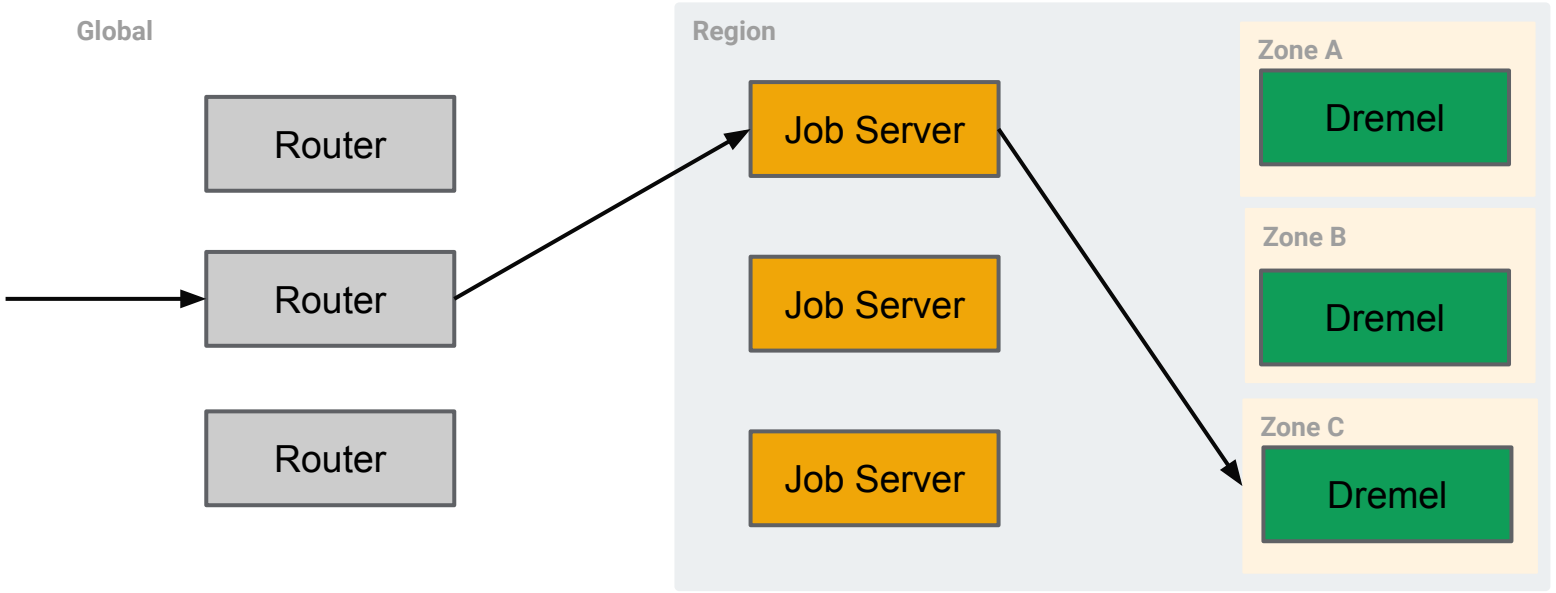
Under the hood

- A BigQuery slot is a unit of computational capacity (in terms of CPU, Memory & Networking Resources) required to execute workloads (SQL Queries)
- A slot commitment is equivalent to buying a dedicated portion of a BigQuery cluster.
- BQ Compute Cluster → Regional Resource (Zonal Redundancy)
- BigQuery automatically calculates how many slots are required by each query, depending on query size and complexity



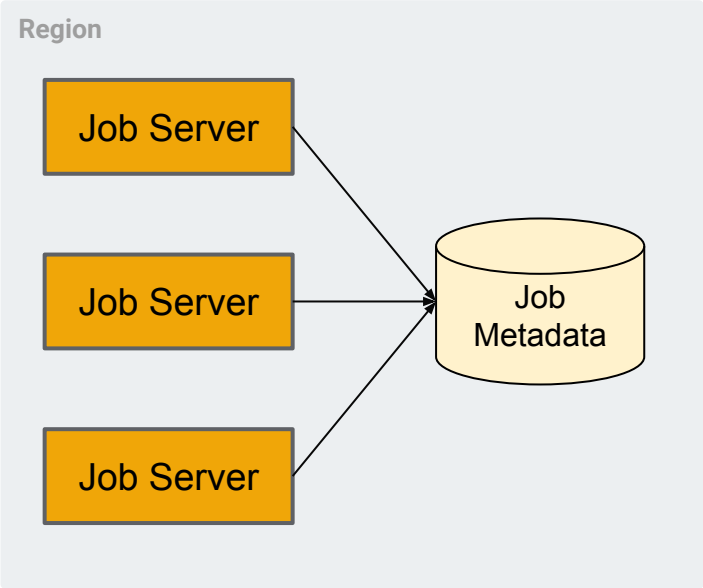
Query —*consists of*→ **Stages** —*consists of*→ **Tasks** —*executes on*→ **Slots**

BigQuery | Query Overview

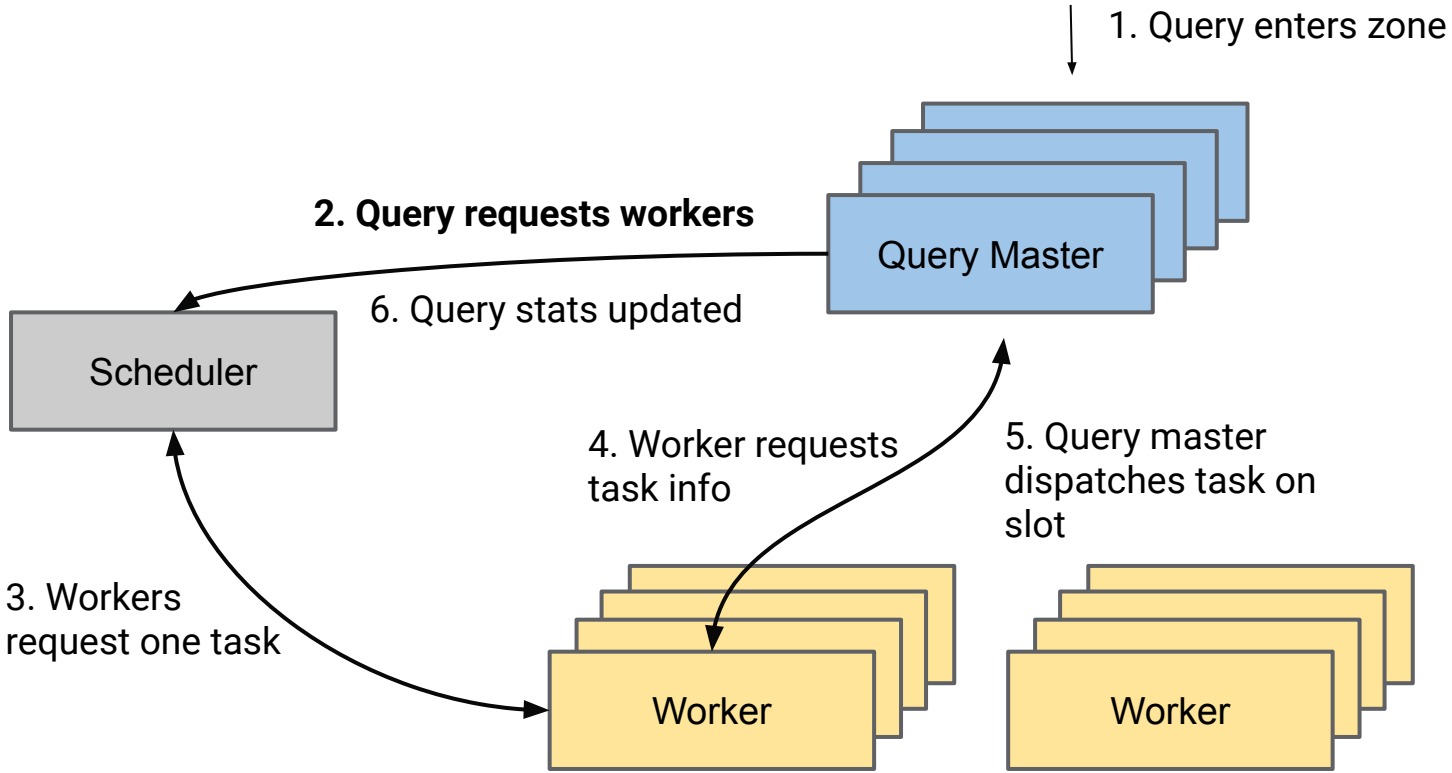


Query Overview | Job Queuing

- Jobs start in the `PENDING` state.
  - Can transition to either `RUNNING` or `DONE` (due to timeout).
  - Most jobs immediately enter the `RUNNING` state.
- Jobs defer their `RUNNING` transition when:
  - `BATCH` priority: always defer at least 1 minute, longer if awaiting quota or the individual server is nearing capacity.
  - `INTERACTIVE` priority: never.
- The Job server will periodically re-evaluate the deferment, with exponential backoff.

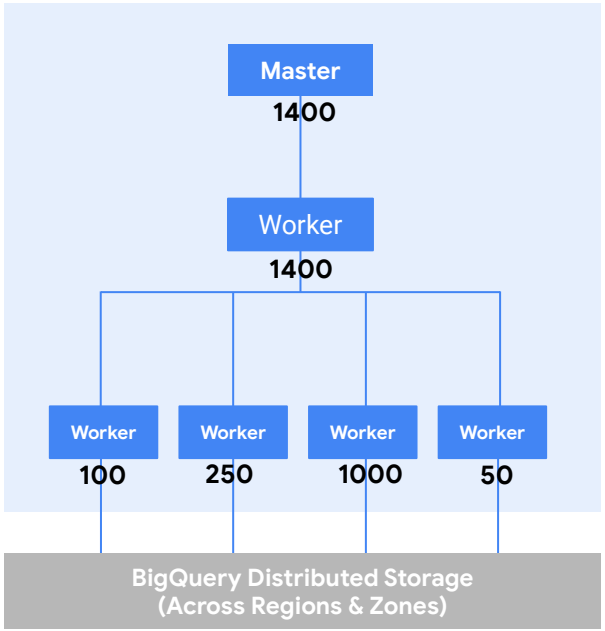


Query Overview | Scheduling Lifecycle



Query Processing | Example

Execution Plan



Availability, Disaster Recovery, Durability & performance

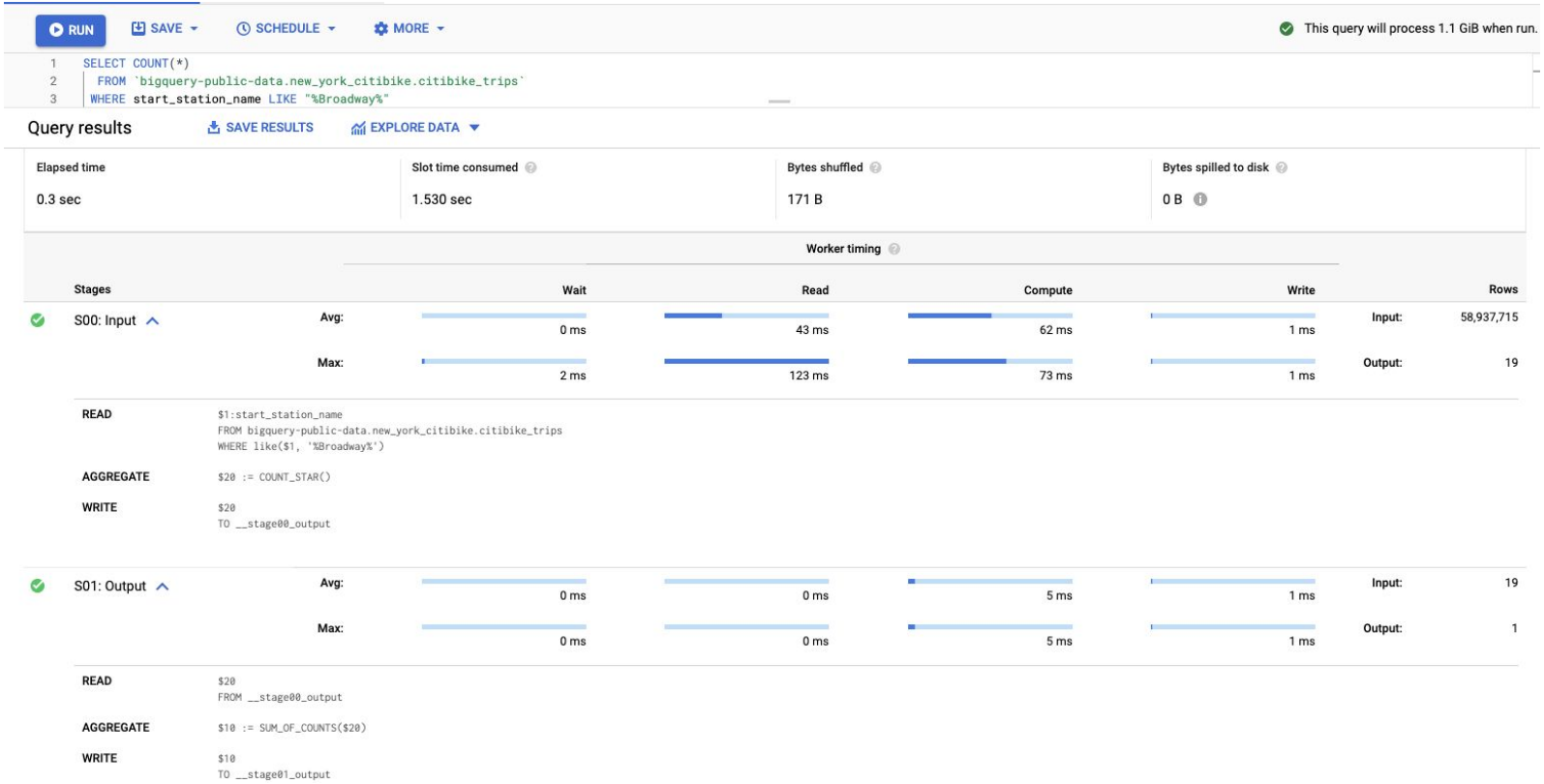
```
SELECT COUNT(*)
 FROM
`bigquery-public-data.new_york_citibike.citibike_trips`
WHERE start_station_name LIKE "%Broadway%"
```

Stage 2: Sum (1 worker)

Stage 1: Filter, count (4 workers)

- Read data from source (bq storage or federated) (Network Capacity = $)
- Operations \ Calculations (CPU Capacity = $)
- Shuffle & Store Intermediate results (Memory Capacity = $)

Query Processing | Execution Details



Query Processing | Slot Consumption Details

```
1  SELECT query, reservation_id, user_email, total_bytes_processed, total_slot_ms
2  FROM `region-us`.INFORMATION_SCHEMA.JOBS_BY_PROJECT
3  WHERE job_id = 'bquxjob_5d395f5a_178eb8cc810'
```

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.6 sec elapsed, 742.8 MB processed)

Job information | Results | JSON | Execution details

Row	query	reservation_id	user_email	total_bytes_processed	total_slot_ms
1	SELECT COUNT(*) FROM `bigquery-public-data.new_york_citibike.citibike_trips` WHERE start_station_name LIKE "%Broadway%"	zr-prod-data-warehouse:US.vrishali-test	vrishalishah@google.com	1135353688	1530

job_stages.records_read	job_stages.records_written	job_stages.parallel_inputs	job_stages.completed_parallel_inputs	job_stages.status	job_stages.steps.kind	job_stages.steps.substeps	job_stages.slot_ms
58937715	19	19	19	COMPLETE	READ	$1:start_station_name	1429
						FROM bigquery-public-data.new_york_citibike.citibike_trips	
						WHERE like($1, '%Broadway%')	
					AGGREGATE	$20 := COUNT_STAR()	
					WRITE	$20	
						TO __stage00_output	
19	1	1	1	COMPLETE	READ	$20	100
						FROM __stage00_output	
					AGGREGATE	$10 := SUM_OF_COUNTS($20)	
					WRITE	$10	
						TO __stage01_output	

Workload Management

BigQuery Slots | Predictable and Flexible Pricing

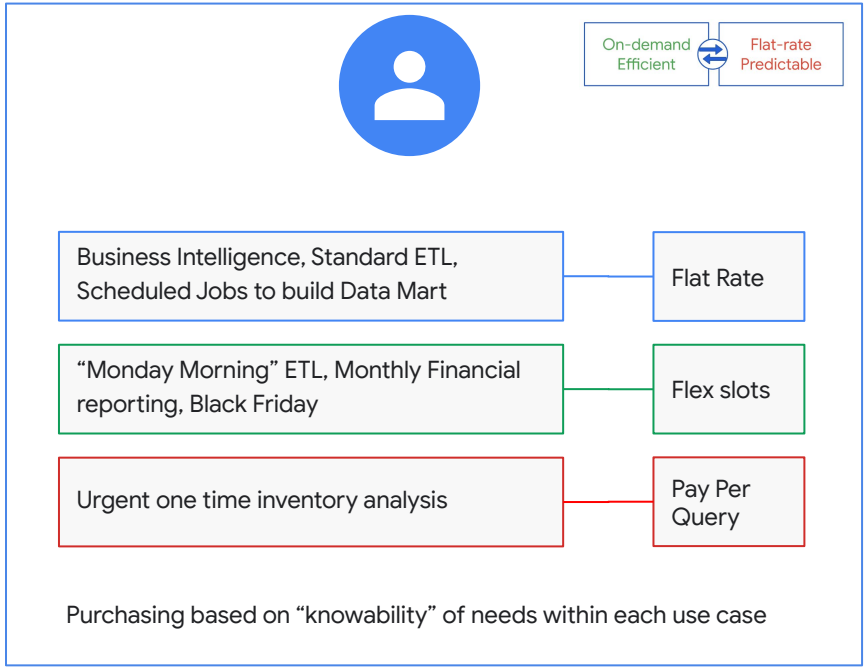
Mix and match pricing models to get the best value for money

1. **Pay per Query**
   Bytes Processed, pay as you go

2. **Flex Slots**
   Burst Capacity, pay as you go | Minutes/Hours/Days

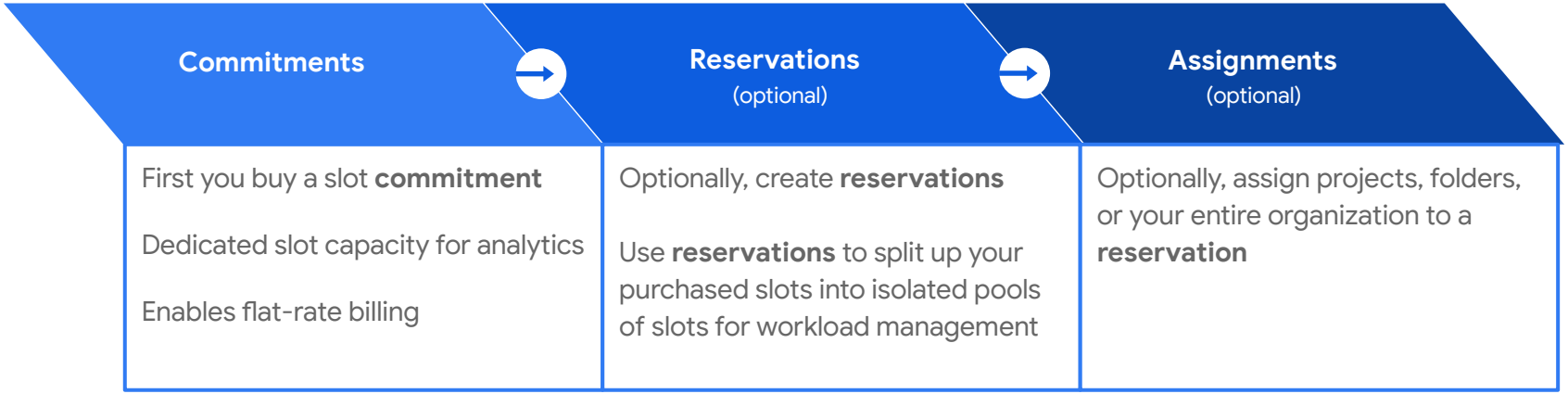
3. **Flat Rate**
   Predictable lowest price | Monthly/Yearly

* Storage, Streaming and BI Engine is charged separately



** Ref: BigQuery Analysis Pricing Model*

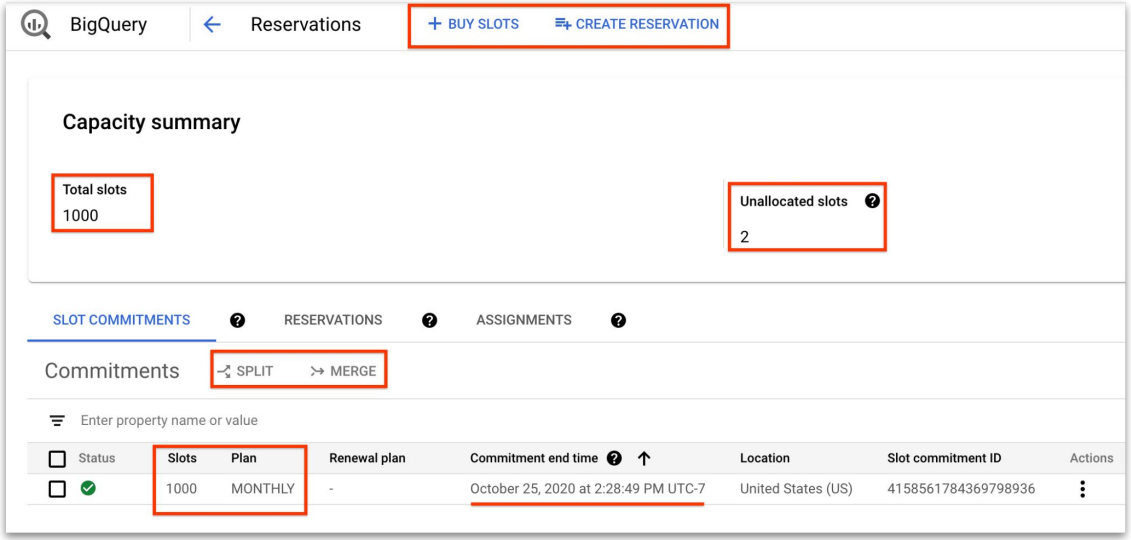
BigQuery Slots | Workload Management



- The first time you buy a slot **commitment**, BigQuery creates a default **reservation** and **assigns** the org to it
- Admin project centralizes the billing and management of your commitments

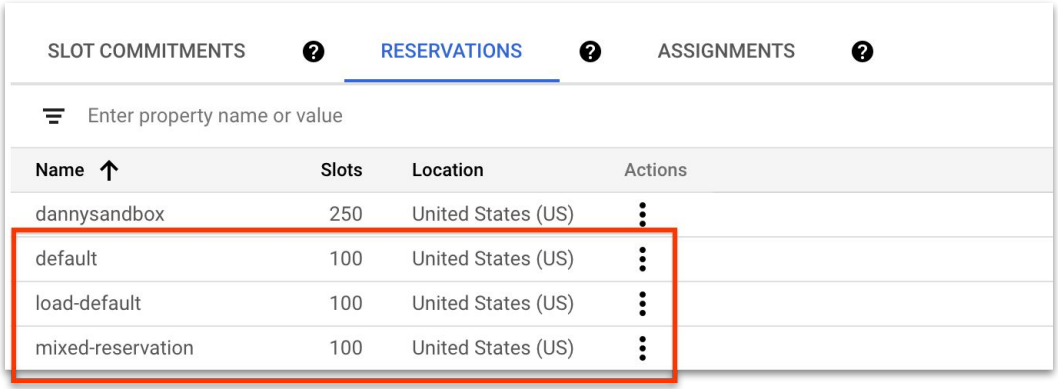
Chart Ref: Introduction to BigQuery Reservations

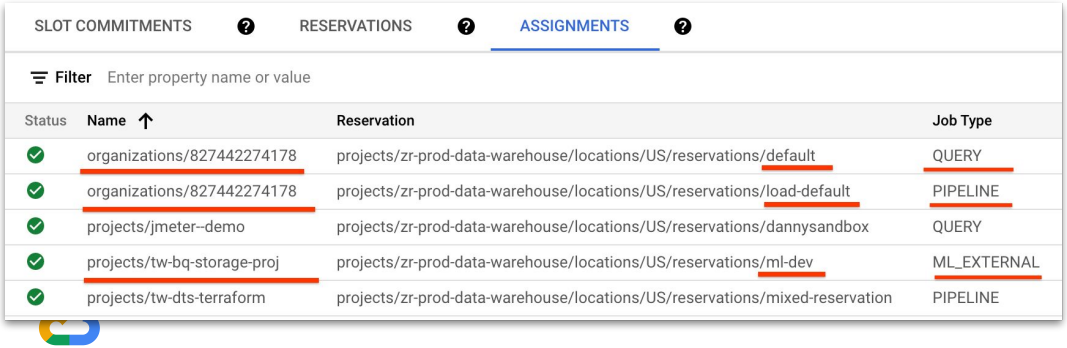
BigQuery Slots | Commitments



- Three different commitment plans: **annual, monthly, and flex**
- Must be purchased in **100-slot increments**
- Regional resources

BigQuery Slots | Reservations and Assignments





- Reservation is an **isolated pool of slots** from a commitment
- Assign **projects, folders, or organizations** to reservations to manage slot resources
- Assignments for **QUERY, PIPELINE, or ML_EXTERNAL** types

BigQuery Slots | Slots Flow

- Purchase slots at organization level (commitment)
- Create reservation and allocate slots to reservations per team or use case (reservation)
- Assign projects to respective reservations (assignment)
- Any unused capacity is seamlessly shared at org level (idle slot sharing)

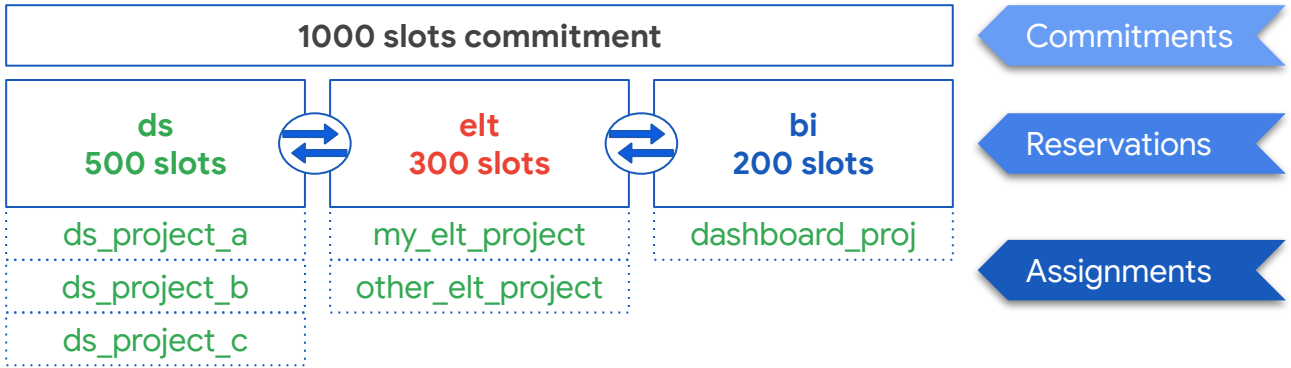


Chart Ref: *Workload management using Reservations*

Reservations | Resource Queueing

Within a reservation

- If resource demand exceeds available capacity, BigQuery queues up additional slots.
- You are not charged for additional slots, and you are not charged for additional on-demand rates.
- As query execution progresses, BigQuery automatically works through the queued up work until none is left

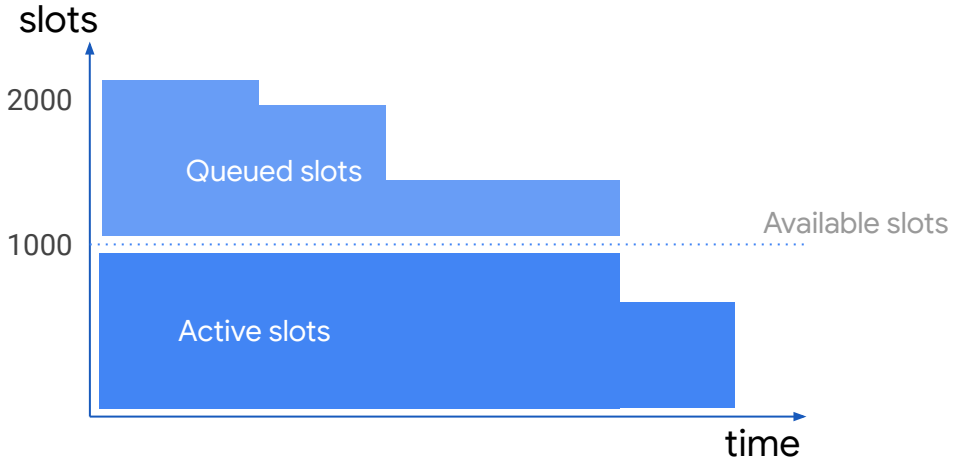
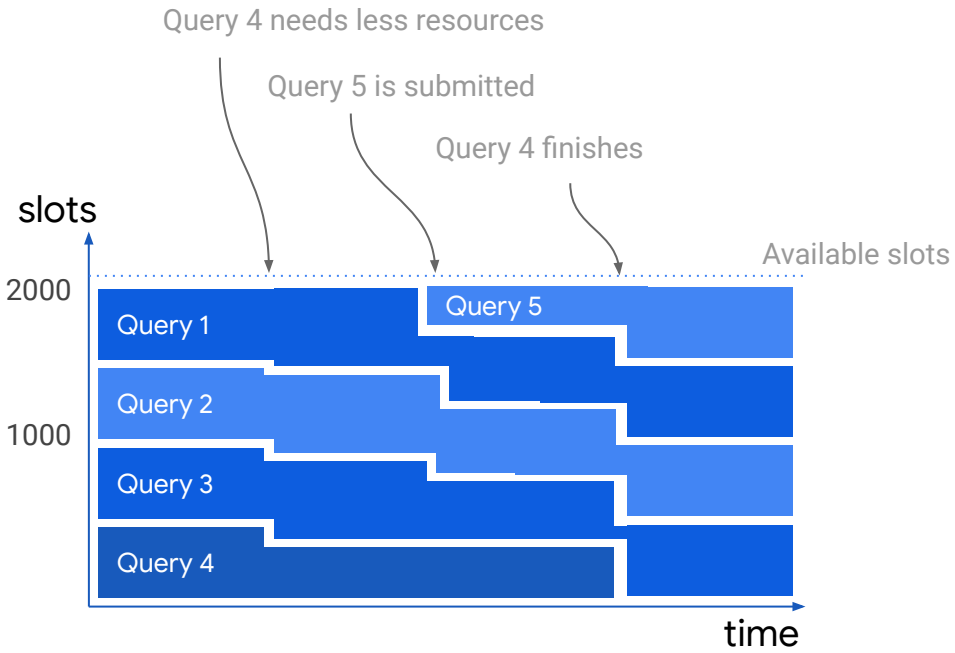


Chart Ref: Introduction to BigQuery Reservations

Reservations | Fair Sharing

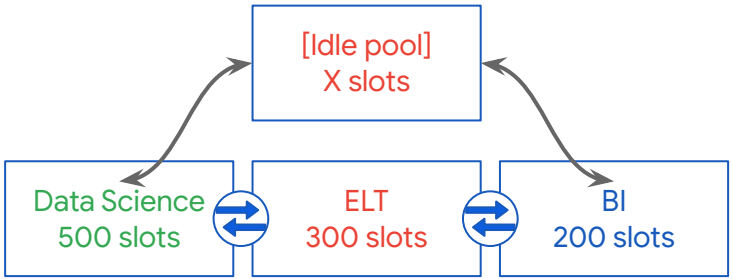
Seamless resource reallocation for queries

- BigQuery dynamically allocates capacity
- Each query gets “fair share” of resources
  - Slots are distributed fairly across all projects
  - Slots are distributed fairly across all active queries within the projects
- Subdividing of compute resources will continue to happen as more queries are executed
- **Avoids workload starvation**



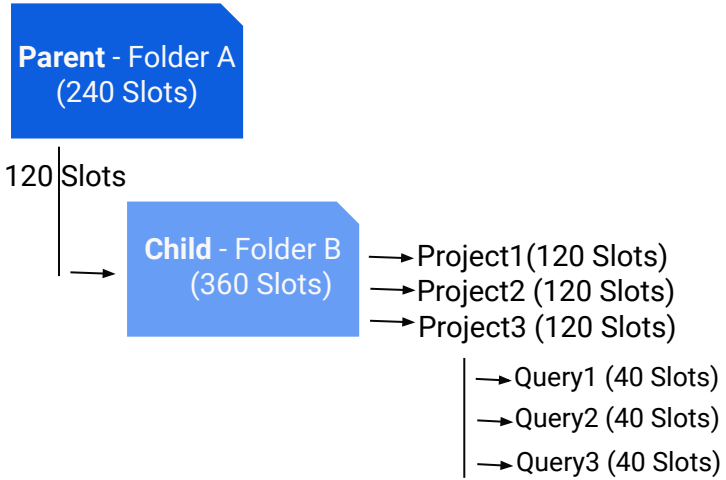
Reservations | Idle Slot Sharing

- Any idle capacity in a reservation is seamlessly available for other reservations to use
- As soon as the reservation needs that capacity back, it gets it
- Queries whose work was preempted will not fail - they simply go back to using their resources as before
- This happens in real-time for every slot
- Net effect is all capacity in an organization is available to be used at any time



Reservations | Slot Hierarchy

- **Parent-Child Relationship**
  Parent can always utilize any idle slots from the child's reservation
  - **Burst (Into parent reservation)**: Queries using the child reservation, get equal access to the parent's slots
  - **No-Burst**: The child can only use its own reservations



Reservations | Unusual Applications

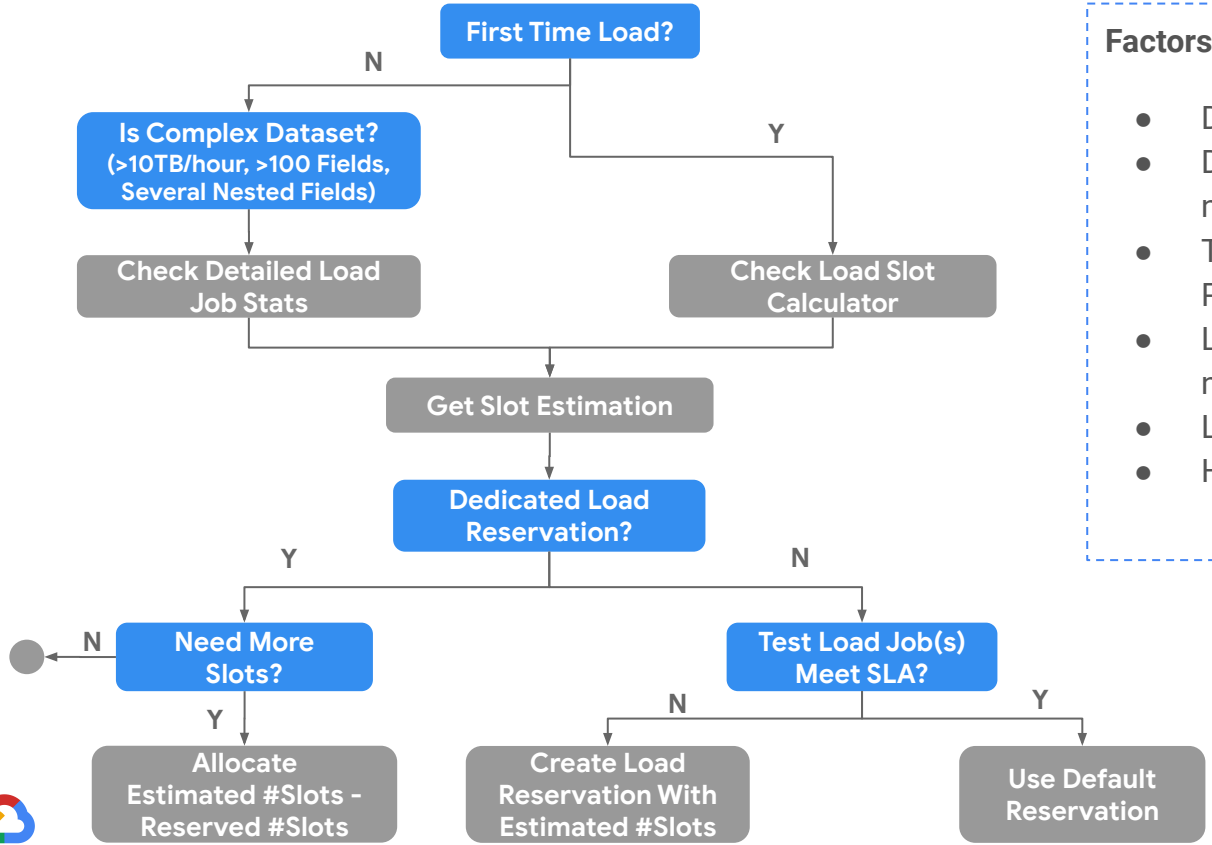
AKA things we've observed our customers doing that we've never planned for.

- Zero slot reservations for running jobs with background priority. The system will only ever run these jobs with idle slots when available.
- Creating penalty box reservations with a small number of slots with idle slots usage disabled. Used for penalizing overly expensive queries.
- Reservations called high-pri/low-pri with frequent movement of jobs in and out of these reservations. Note that if high-pri reservation is unused, its idle slots can still be used by the low-pri reservation.

(*) any unused capacity seamlessly shared

Capacity Management

Slot Projection | Loading Data

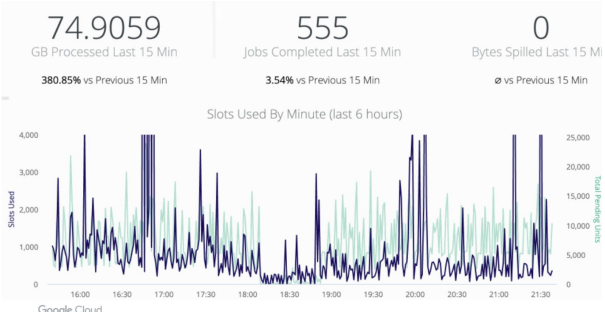


Factors for load slot estimations:

- Dataset size
- Dataset Complexity: #fields, # nested/repeated fields
- Table Schema: Is the table Partitioned or Clustered?
- Load frequency: Hourly, Daily, Every n-hours
- Load SLA
- Historical Load Throughput

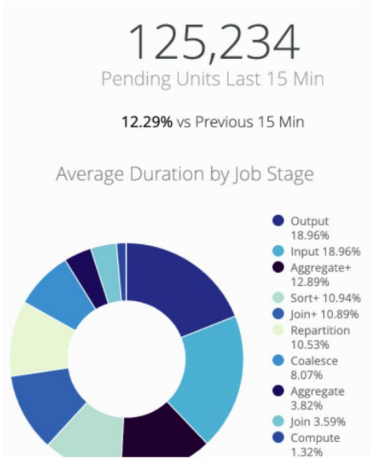
Slot Projection | Look for the Signals

How do we know if we need more slots..??



- Monitor trend (week over week, month over month)
  - Are more users being onboarded within same slot allocation?
  - Monitor user growth/trend

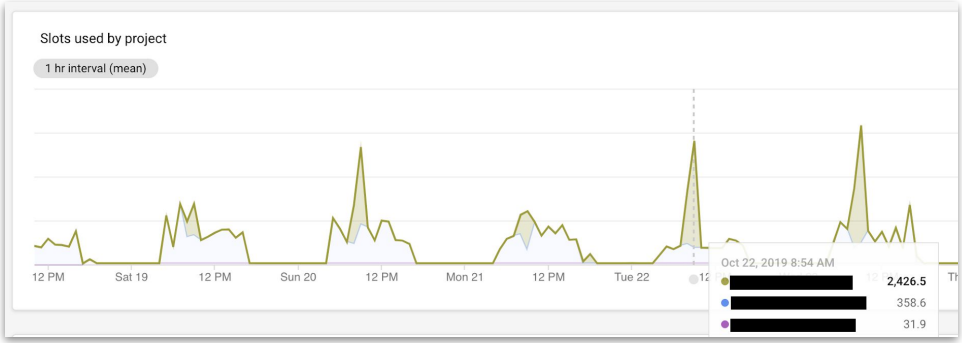
- More slots needed for user queries, if for a given slot allocation observes:
  - Concurrency - consistently increasing
  - Throughput - consistently decreasing
  - Slot Utilization - consistently increasing or keeping beyond, let's say 90%
    - If slot utilization has spikes, are they on regular frequency?
    - Some of non-critical workload can be time shifted
  - For given set of priority/label/project - group of queries/users
    - Avg. Wait Time - consistently increasing
    - Avg. query run-time - consistently increasing



BigQuery | Monitoring Options

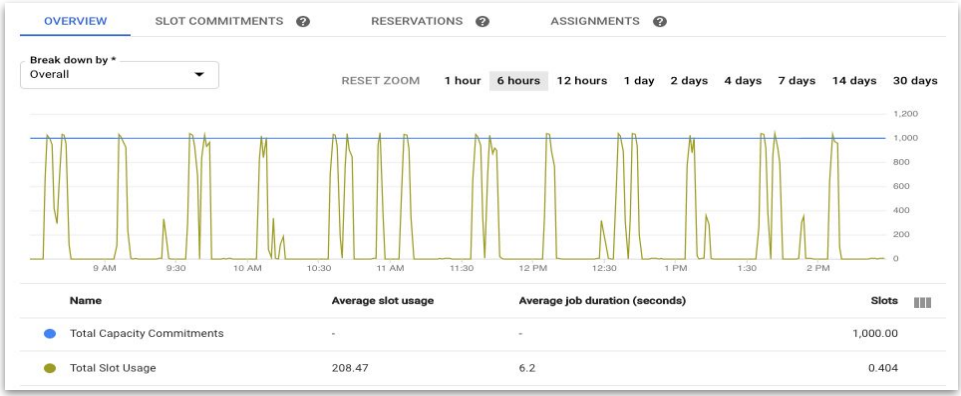
Cloud Monitoring

- Subset of job metrics available
- 350-project limit per workspace
- Limited reservations data



BQ Admin Panel (Preview)

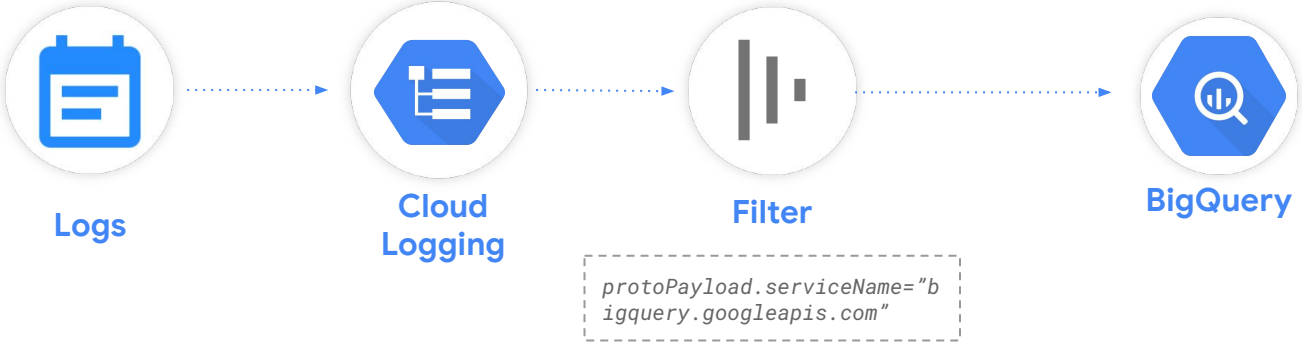
- No view for detailed job metrics
- No alerting functionality
- Permissions required to view at org level



BigQuery Monitoring | Audit Logs

Logs provide various details about the executed BigQuery job:

- **query** - The BigQuery SQL executed
- **startTime** - Time when the job started
- **endTime** - Time when the job ended.
- **totalProcessedBytes** - Total bytes processed for a job
- **totalBilledBytes** - Processed bytes, adjusted by the job's CPU usage
- **totalSlotMs** - The total number of slot-ms consumed by the query job
- **referencedFields** - The columns of the underlying table that were accessed



BigQuery Monitoring | Information_Schema

- Leverage INFORMATION_SCHEMA views (real-time updates)
- Run ad hoc SQL queries against all BigQuery jobs across your project

```
SELECT job_id, user_email, total_bytes_processed
FROM `region-us`.INFORMATION_SCHEMA.JOBS_BY_PROJECT
WHERE EXTRACT(DATE FROM creation_time) = CURRENT_DATE()
ORDER BY total_bytes_processed DESC
LIMIT 3
```

Query complete (1.2 sec elapsed, 53 MB processed)

Job information | Results | JSON | Execution details

Row	job_id	user_email	total_bytes_processed
1	script_job_3da815dcf7e6495a2b000dbebe6ee6c9_0	bigquery-adminbot@system.gserviceaccount.com	21906600899080
2	materialized_view_refresh_GIlh81T2G0mKELkVJAyUHtSDy4JrCK	bigquery-adminbot@system.gserviceaccount.com	21906600899080
3	bqjob_r69c7484a9bf14acd_000001774768dced_1	vincegonzalez@google.com	15244347073472

Source: Getting jobs metadata using INFORMATION_SCHEMA