


BEAM  
SUMMIT



# Design considerations to operate a stateful streaming pipeline as a service

Stateful processing with Beam



**Israel Herraiz**

Strategic Cloud Engineer, Google Cloud



**Bhupinder Sindhwani**

Customer Engineer, Google Cloud

# TOC

Context

Example Problem

Design Principles

Example Solution

Operating Principles

# Context

Data pipelines are services

Streaming is unforgiving

Observability is critical

User Expectations

Data Correctness

Data Quality

Performance

Latency

Alert fatigue

SRE principles



Photo by [Sigmund](#) on [Unsplash](#)

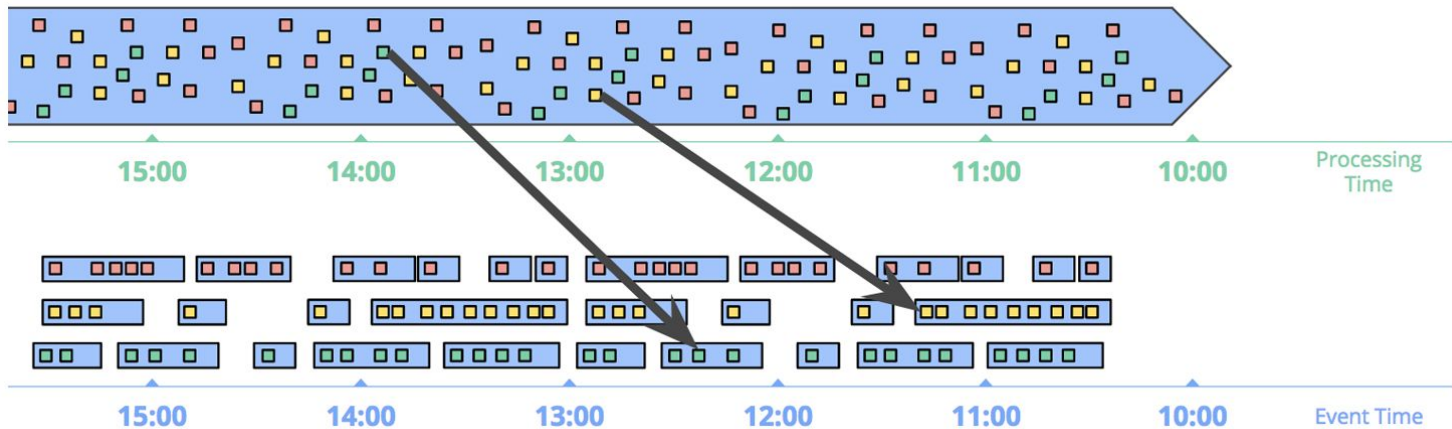


Photo by [Quaritsch Photography](#) on [Unsplash](#)



Photo by [Chris Liverani](#) on [Unsplash](#)

# Example Problem



How taxi ride events arrive to PubSub

How we will group them to be able to calculate business properties e.g. calculating sessions for income taxi ride events based on event attribute

# Design Principles





Extract

---

Reads (input only)

Transform

---

Enriching / hydrating →   
External state → 

Load

---

Output only

# Design Principles: but sinks have side effects?

Yes, and that's fine, but the details matter

Speaker(s):



John Casey

## How to write an IO for Beam

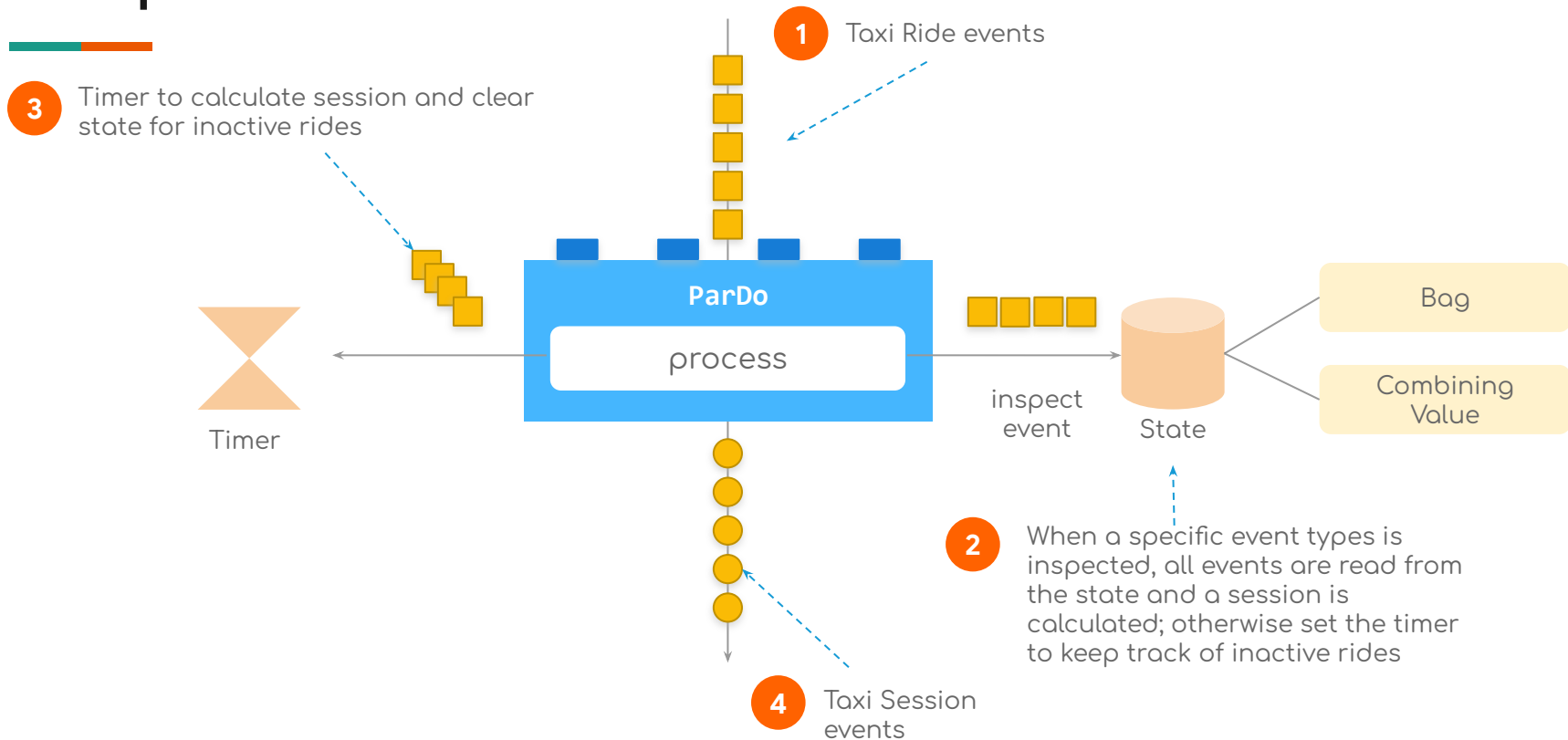
Jun-13 11:00-11:50 EDT

Room: Palisades

Writing an IO in Beam is hard. Distributed data reading and writing are inherently challenging, and its easy to make mistakes. This session is a walk through on the key design hurdles, and how to use Beam features to write a high quality IO.

<https://beamsummit.org/sessions/2023/how-to-write-an-io-for-beam/>

# Example Solution





## Example Solution



And cannot we just use windows?

- Windows are also possible
- If logic is not based on temporal properties, state & timers might be a better fit

In any case, for SRE principles, produce custom metrics

- Distribution metrics are easier to use for SLOs
  - Stable metric that should not deviate too much from a "good" value

# Example Solution



Step name	Status	Wall time	Stages	Input steps	Output steps
▶ Read TaxiRide events	Running	0 seconds	F16	—	Read JSON
Read JSON	Running	5 days 13 hours 53 minutes	F16	Read TaxiRide events/Read	TaxiRideEvent with EventTimestamp
TaxiRideEvent with EventTimestamp	Running	5 hours 32 minutes	F16	Read JSON	Parse Raw to Row
Parse Raw to Row	Running	18 minutes	F16	TaxiRideEvent with EventTimestamp	Write RawEvents to BQ/.../AppendDestination
▶ Write RawEvents to BQ	Running	15 days 20 hours 22 minutes	F15	Parse Raw to Row	—
▶ Create Keys with ride_id	Running	1 hour 41 minutes	F16	TaxiRideEvent with EventTimestamp	Generate Sessions
Generate Sessions	Running	84 days 14 hours 27 minutes	F17	Create Keys with ride_id/Map(<lambda at util.py:910>)	Parse Stats to Row
Parse Stats to Row	Running	12 minutes	F17	Generate Sessions	Write StatEvents to BQ/.../AppendDestination
▶ Write StatEvents to BQ	Running	9 hours 35 minutes	F14	Parse Stats to Row	—

Counter name	Value	Step
ride_events_processed	173,155,334	Generate Sessions
ride_events_received	173,713,880	Generate Sessions
sessions_processed	449,895	Generate Sessions

# Example Solution

ride_id	point_idx	latitude	longitude	timestamp	meter_reading	meter_increment	ride_status	passenger_count
615d0eec-f7ae-48f4-8e63-00fd...	302	40.7663	-73.979...	2023-06-09 00:55:21.854890 U...	13.585863	0.0449863	enroute	1
77594d39-cdd0-4db4-a0c3-51a...	235	40.760...	-73.956...	2023-06-09 00:55:46.710280 U...	6.413646	0.027292112	enroute	1
69c90d6c-35a2-472d-b289-9c3...	47	40.762...	-73.999...	2023-06-09 00:55:00.608380 U...	1.8488153	0.039336495	enroute	5
1182495c-8100-4826-9086-7c7...	365	40.753...	-73.996...	2023-06-09 00:55:11.731600 U...	12.111111	0.033181123	enroute	2
5efb3bbf-c6d7-4308-912e-ead...	296	40.757...	-73.996...	2023-06-09 00:55:45.747990 U...	14.89705	0.05032787	enroute	1
c3c23afa-3c4d-46a9-b6bd-58d...	47	40.768	-73.952...	2023-06-09 00:55:24.280720 U...	1.7284635	0.03677582	enroute	5
954878ef-6a6b-40e2-9035-8c0...	153	40.739...	-74.001...	2023-06-09 00:56:03.446680 U...	6.6670394	0.04357542	enroute	2
1ad40796-97cf-4898-bd74-755...	41	40.759...	-73.965...	2023-06-09 00:54:57.456690 U...	1.3948454	0.034020618	enroute	6
294336f3-5d10-4c6d-8a48-ae2...	387	40.756...	-73.974...	2023-06-09 00:55:35.273970 U...	12.544704	0.032415256	enroute	2
c4e8aa27-855e-4acc-bd71-417...	28	40.744...	-73.995...	2023-06-09 00:55:49.093920 U...	1.0347826	0.036956523	enroute	2
91b2bd0d-dc2b-4ff1-b160-096...	208	40.765...	-73.964...	2023-06-09 00:55:13.035110 U...	10.207284	0.04907348	enroute	1
562f42c7-1960-44ac-86d8-c97...	928	40.724...	-73.993...	2023-06-09 00:54:56.496310 U...	28.40294	0.030606616	enroute	2
1e53e585-e6b8-4235-8647-66d...	2304	40.741...	-73.947...	2023-06-09 00:55:16.300200 U...	53.22046	0.023099158	enroute	1
71f29f45-4c82-403b-bf4a-202f...	444	40.7959	-73.976...	2023-06-09 00:55:45.510820 U...	13.015213	0.029313544	enroute	1
0e711967-29a1-41e9-829e-a08...	18	40.766...	-73.956...	2023-06-09 00:55:10.906360 U...	0.838835	0.046601944	enroute	1
d237a808-3e00-4ae8-aaed-f62...	153	40.755...	-73.979...	2023-06-09 00:55:52.182470 U...	5.881214	0.038439307	enroute	1
b9bcd1a0-337e-4035-95b4-5ee...	179	40.764...	-73.973...	2023-06-09 00:55:39.544730 U...	6.237879	0.034848485	enroute	2
cc3ca1f0-54da-454a-b023-cc2...	101	40.781...	-73.971...	2023-06-09 00:54:54.731600 U...	17.0	0.16831683	dropoff	1
69657b37-a728-469e-a616-c72...	139	40.7655	-73.997...	2023-06-09 00:54:57.385810 U...	3.637383	0.026168223	enroute	1
1ac202d3-39a5-42f0-b4ed-019...	29	40.730...	-73.9865	2023-06-09 00:56:07.073180 U...	1.7945545	0.06188119	enroute	1
197b90c8-6983-43e8-9cd6-3ce...	65	40.761...	-73.986...	2023-06-09 00:56:04.838550 U...	3.6671126	0.056417115	enroute	1
f3600c53-2bfa-4502-a1e6-0b0...	2	40.757...	-73.963...	2023-06-09 00:55:34.929400 U...	0.1021978	0.0510989	enroute	2

# Example Solution

session_id	total_meter_reading	passenger_count	journey_length_sec	number_of_ride_event	session_reason	session_start_time	session_end_time
8408e3fd-9933-41e4-a2fa-0f83...	63.629997	1	1964.0	3205	DROPOFF	2023-06-09 23:21:38.731600 U...	2023-06-09 23:54:22.731600 U...
1692e757-4833-4962-a6e8-02...	60.8	3	2221.0	2730	DROPOFF	2023-06-09 23:17:42.731600 U...	2023-06-09 23:54:43.731600 U...
d8d5a1ce-1b9d-4cdb-9a56-510...	69.6	1	1935.0	2964	DROPOFF	2023-06-09 23:22:35.731600 U...	2023-06-09 23:54:50.731600 U...
ceabf1c0-2afe-4e6b-807d-f6db...	64.3	1	1792.0	3177	DROPOFF	2023-06-09 23:24:57.731600 U...	2023-06-09 23:54:49.731600 U...
a7c25ff0-5ca0-46e6-87b3-75b...	37.63	1	1770.0	1360	DROPOFF	2023-06-09 23:25:26.731600 U...	2023-06-09 23:54:56.731600 U...
be841d7a-c923-4150-9540-4f7...	42.36	1	1742.0	1802	DROPOFF	2023-06-09 23:25:56.731600 U...	2023-06-09 23:54:58.731600 U...
6afae0b3-b317-4b88-b9a6-113...	26.3	3	1846.0	1032	DROPOFF	2023-06-09 23:25:12.731600 U...	2023-06-09 23:55:58.731600 U...
437b85fe-abd8-49fc-8ad1-717...	51.6	6	1774.0	1510	DROPOFF	2023-06-09 23:26:40.731600 U...	2023-06-09 23:56:14.731600 U...
f42d3e27-41ce-45f4-84f4-8425...	47.4	2	1838.0	1674	DROPOFF	2023-06-09 23:25:47.731600 U...	2023-06-09 23:56:25.731600 U...
a8b729bc-e813-4db2-9ddd-88...	14.0	1	1764.0	245	DROPOFF	2023-06-09 23:27:10.731600 U...	2023-06-09 23:56:34.731600 U...

session_id	total_meter_reading	passenger_count	journey_length_sec	number_of_ride_event	session_reason	session_start_time	session_end_time
1343fdc3-2142-4d2e-be84-90d...	5.255085	1	159.63559	113	GARBAGE_COLLECTION	2023-06-09 07:27:25.731600 U...	2023-06-09 07:30:05.367190 U...
c7c4e27a-65cb-4d98-8627-c9e...	5.5551996	1	71.424	114	GARBAGE_COLLECTION	2023-06-09 08:23:36.731600 U...	2023-06-09 08:24:48.155600 U...
5ab567d2-0f95-4d4f-b98c-f4f2...	5.558823	5	96.28676	115	GARBAGE_COLLECTION	2023-06-09 10:07:20.731600 U...	2023-06-09 10:08:57.018360 U...
a560ea1d-3673-4fa5-8e88-1e4...	5.7543306	1	132.94488	115	GARBAGE_COLLECTION	2023-06-09 10:58:30.731600 U...	2023-06-09 11:00:43.676480 U...
4b0c1b03-dc4c-466b-8c22-801...	5.7588654	2	151.91489	116	GARBAGE_COLLECTION	2023-06-09 11:03:37.731600 U...	2023-06-09 11:06:09.646490 U...
05fc5e42-93e3-48a2-97ca-f21...	6.905197	3	168.66142	116	GARBAGE_COLLECTION	2023-06-09 11:03:26.731600 U...	2023-06-09 11:06:15.393020 U...
607f1fbd-46ba-4085-8905-c80...	5.2576003	3	178.56	117	GARBAGE_COLLECTION	2023-06-09 07:01:23.731600 U...	2023-06-09 07:04:22.291600 U...
0946f5ca-2a05-4bdc-b9b4-f35...	5.757037	1	178.66667	117	GARBAGE_COLLECTION	2023-06-09 07:11:44.731600 U...	2023-06-09 07:14:43.398270 U...
95a3eb7c-587c-4143-b01b-ce6...	5.7579713	1	160.82609	117	GARBAGE_COLLECTION	2023-06-09 08:03:33.731600 U...	2023-06-09 08:06:14.557690 U...
b01c6356-cd4c-43ac-9e22-0c5...	6.2533336	5	160.58518	118	GARBAGE_COLLECTION	2023-06-09 07:33:54.939010 U...	2023-06-09 07:36:35.524190 U...

# Example Solution

```
def process(self,
            element: Tuple[str, TaxiRideEvent],
            element_timestamp: Timestamp = beam.DoFn.TimestampParam,
            taxi_ride_events_bag=beam.DoFn.StateParam(TAXI_RIDE_EVENTS_BAG),
            max_timestamp_seen=beam.DoFn.StateParam(MAX_TIMESTAMP),
            gc_timer=beam.DoFn.TimerParam(GC_TIMER)) -> Iterable[TaxiStatEvent]:

    taxi_ride_events_bag.add(element)
    max_timestamp_seen.add(element_timestamp.seconds())
    GenerateSessionsDoFn.ride_events_received.inc()

    if element[1].ride_status == "dropoff":
        taxi_stat_event = GenerateSessionsDoFn.calculate_session(taxi_ride_events_bag,
                                                                TaxiStatEvent.Reason.DROPOFF)

        # Generate session and output TaxiStatEvent
        GenerateSessionsDoFn.sessions_processed.inc()
        yield beam.window.TimestampedValue(taxi_stat_event, Timestamp(max_timestamp_seen.read
        ()))

        # Clear state for the key
        taxi_ride_events_bag.clear()
        max_timestamp_seen.clear()
        gc_timer.clear()
    else:
        # Set the timer to be 5 minutes to keep track of inactive keys
        expiration_time = Timestamp(max_timestamp_seen.read()) + Duration(seconds=5 * 60)
        gc_timer.set(expiration_time)
```

```
@on_timer(GC_TIMER)
def expiry_callback(self,
                   taxi_ride_events_bag=beam.DoFn.StateParam
                   (TAXI_RIDE_EVENTS_BAG),
                   max_timestamp_seen=beam.DoFn.StateParam(MAX_TIMESTAMP)) ->
    Iterable[TaxiStatEvent]:

    # We have not seen the drop-off message 5 minutes after the max timestamp, so
    # let's emit this session now
    taxi_stat_event = GenerateSessionsDoFn.calculate_session(taxi_ride_events_bag,
                                                            TaxiStatEvent.Reason.
                                                            GARBAGE_COLLECTION)

    # Generate session and output TaxiStatEvent
    GenerateSessionsDoFn.sessions_processed.inc()
    yield beam.window.TimestampedValue(taxi_stat_event, Timestamp
    (max_timestamp_seen.read()))

    # Clear state for the key
    taxi_ride_events_bag.clear()
    max_timestamp_seen.clear()
```

## Example Solution: adding business-related SLIs



```
class GenerateSessionsDoFn(beam.DoFn):  
    def __init__(self):  
        self._distribution = Metrics.distribution('My sessions DoFn', 'vehicle_speed')
```

```
    journey_length_seconds = (end_time - start_time).total_seconds()  
    distance = last_event.meter_reading # Just an example, this could be actual distance  
    session_speed = distance / journey_length_seconds  
    self._distribution.update(session_speed)
```

# Operating Principles



**SLIs**

---

Indicator of the level of  
service

**SLOs**

---

Target levels for the reliability  
of service

**Error Budget**

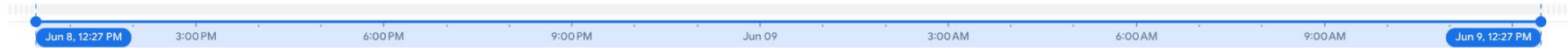
---

Allowed bad events; user  
tolerance

# Operating Principles



Alerts timeline No service alerts Time selection is Jun 08, 12:27 PM to Jun 09, 12:27 PM GMT-4 RESET 1 day HIDE TIMELINE



Current status of 3 SLOs Status calculated at 12:27 PM GMT-4 + CREATE SLO

Status	Objective	Type	Alerts firing	Error budget
✓	Processing latency (90s)	Windowed SLI	0/0	✓ 20.98%
✓	BigQuery session write latency (10s)	Windowed SLI	0/0	✓ 5.17%
✓	Data age (90s)	Windowed SLI	0/0	✓ 13.98%



# Operating Principles: SLI

Alerts timeline No service alerts Time selection is Jun 08, 12:27 PM to Jun 09, 12:27 PM GMT-4

RESET 1 day HIDE TIMELINE



Current status of 3 SLOs Status calculated at 12:27 PM GMT-4

+ CREATE SLO

Status	Objective	Type	Alerts firing	Error budget
✓	Processing latency (90s)	Windowed SLI	0/0	20.98%
<p>Service level indicator 100%</p> <p>Service level indicator represents the current fraction of successful requests to your service</p> <p>Alerts firing 0/0</p>				
✓	BigQuery session write latency (10s)	Windowed SLI	0/0	5.17%
✓	Data age (90s)	Windowed SLI	0/0	13.98%

# Operating Principles: SLI

Alerts timeline No service alerts Time selection is Jun 08, 12:27 PM to Jun 09, 12:27 PM GMT-4 RESET 1 day HIDE TIMELINE



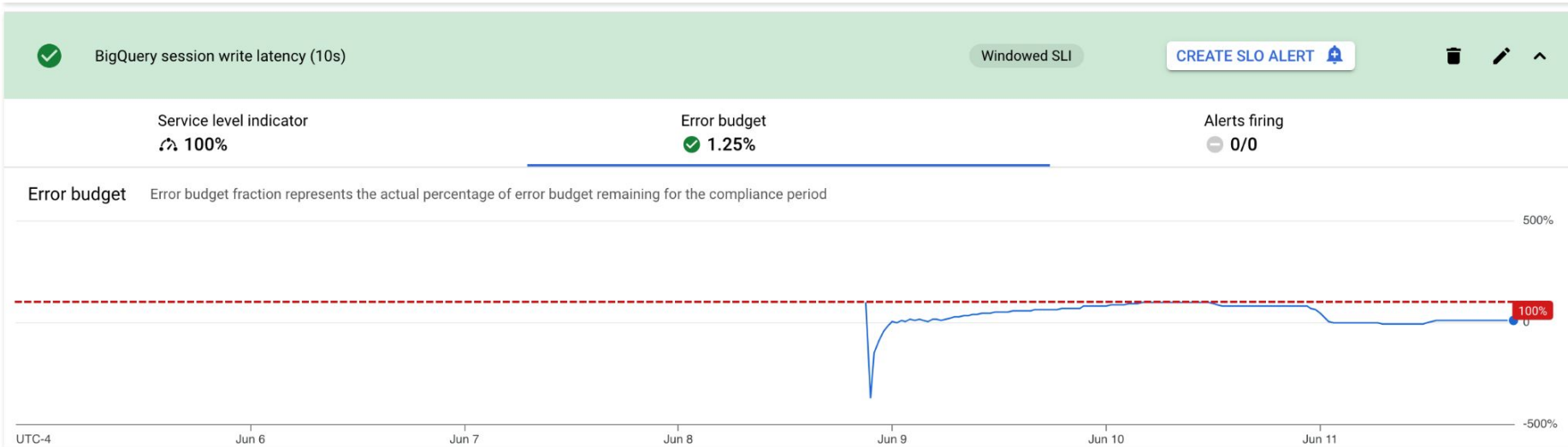
Current status of 3 SLOs Status calculated at 12:27 PM GMT-4 + CREATE SLO

Status	Objective	Type	Alerts firing	Error budget
✓	Processing latency (90s)	Windowed SLI	0/0	20.98%
✓	BigQuery session write latency (10s)	Windowed SLI	CREATE SLO ALERT	
<b>Service level indicator</b> 🔄 100%		<b>Error budget</b> ✓ 5.17%	<b>Alerts firing</b> 0/0	
<b>Service level indicator</b> Service level indicator represents the current fraction of successful requests to your service				
✓	Data age (90s)	Windowed SLI	0/0	13.98%

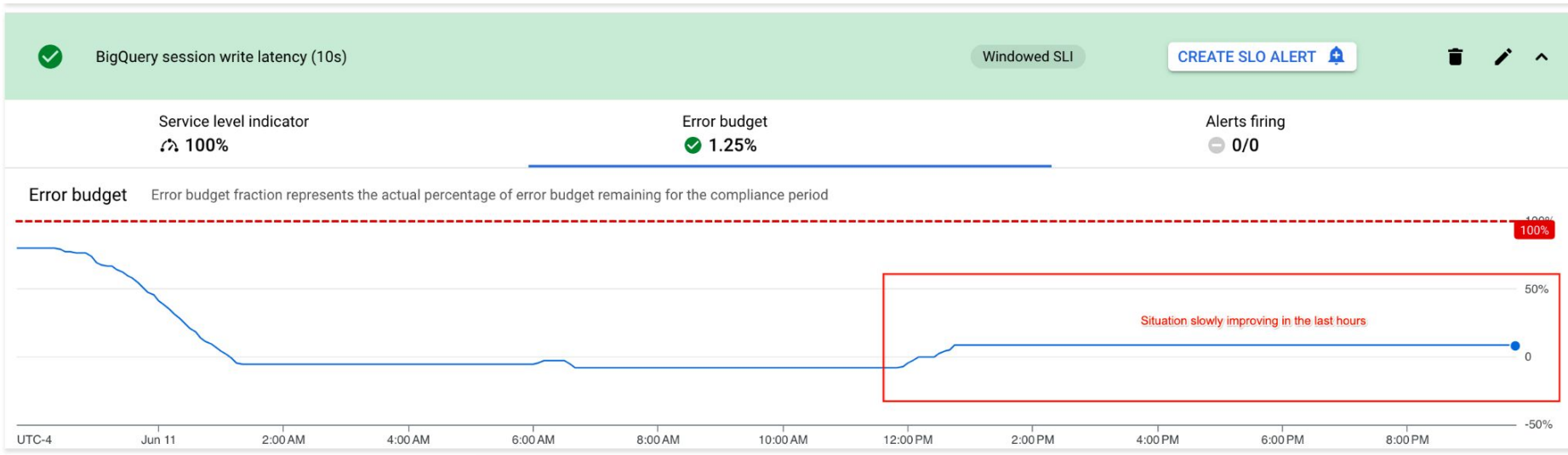
# Operating Principles: error budget



# Operating Principles: error budget



# Operating Principles: error budget



# Reminder

Data pipelines are services

User Expectations



Photo by [Sigmund](#) on [Unsplash](#)

Streaming is unforgiving

Data Correctness  
Data Quality  
Performance  
Latency



Photo by [Quaritsch Photography](#) on [Unsplash](#)

Observability is critical

Alert fatigue  
SRE principles



Photo by [Chris Liverani](#) on [Unsplash](#)

# QUESTIONS?

Israel Herraiz

[linkedin.com/in/herraiz](https://www.linkedin.com/in/herraiz)

[github.com/iht](https://github.com/iht)

Bhupinder Sindhvani

[linkedin.com/in/bhupindersindhvani](https://www.linkedin.com/in/bhupindersindhvani)

[github.com/BhupiSindhvani](https://github.com/BhupiSindhvani)

**Example implementation:**

[github.com/BhupiSindhvani/beam-stateful-processing](https://github.com/BhupiSindhvani/beam-stateful-processing)