TRUELAYER

Detecting the Dip:
Turning noisy metrics into reliable production signals

BUILDING WITH TRUELAYER

Hello

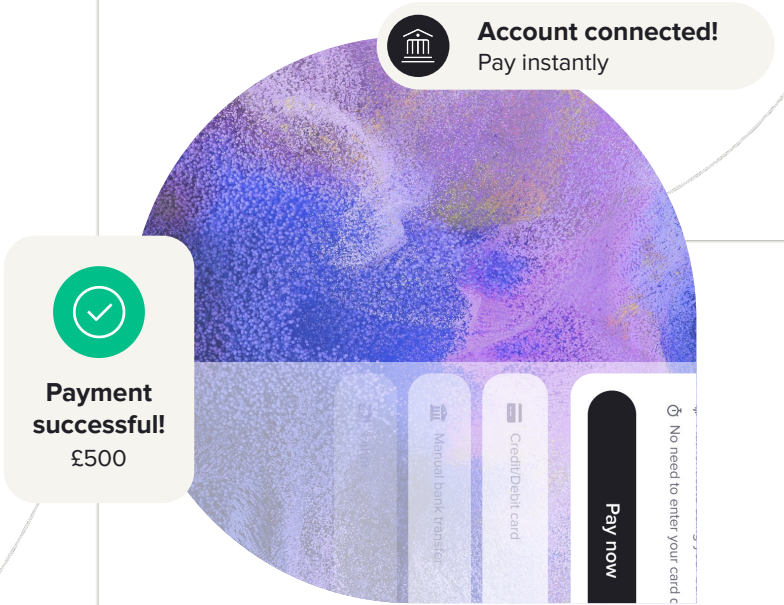
Karina Celis Jones

Senior Engineer Manager @ TrueLayer

Managing Payin Team and Credit Team

Supporting Identity Team and User Network Team.

What is TrueLayer?



TRUELAYER

How to detect when something goes wrong in places we don’t control

The most painful truth...

If you created an API/SDK/Library for others to integrate with...

...each client will do it their own way

Let’s set the scene...

Traffic changed for one client — very specific case.
Our systems were green.
Our alerts were silent.

We want to know when a client's traffic changes before they even know...

Client Operations / Client Relations / Integration

We asked ourselves some questions

What information do we have?

What do we need to know?

What should we monitor?

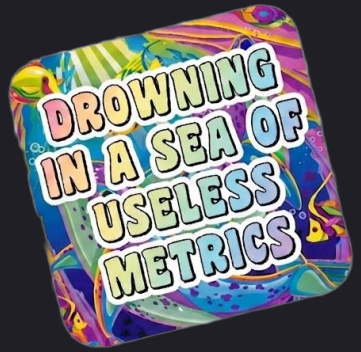
Define our data

Define our aggregates

Define our alerts

What information do we have?

We have so much data...



*Thank you Honeycomb

What do we need to know?

Use Case 1:
We want to know when a client is having zero volume at any given state for a long period of time.

Use Case 2:
We want to know when a client's conversion or volumes change but are not completely zero.

One day of traffic in TrueLayer

Any night Owler making their last minute shops?

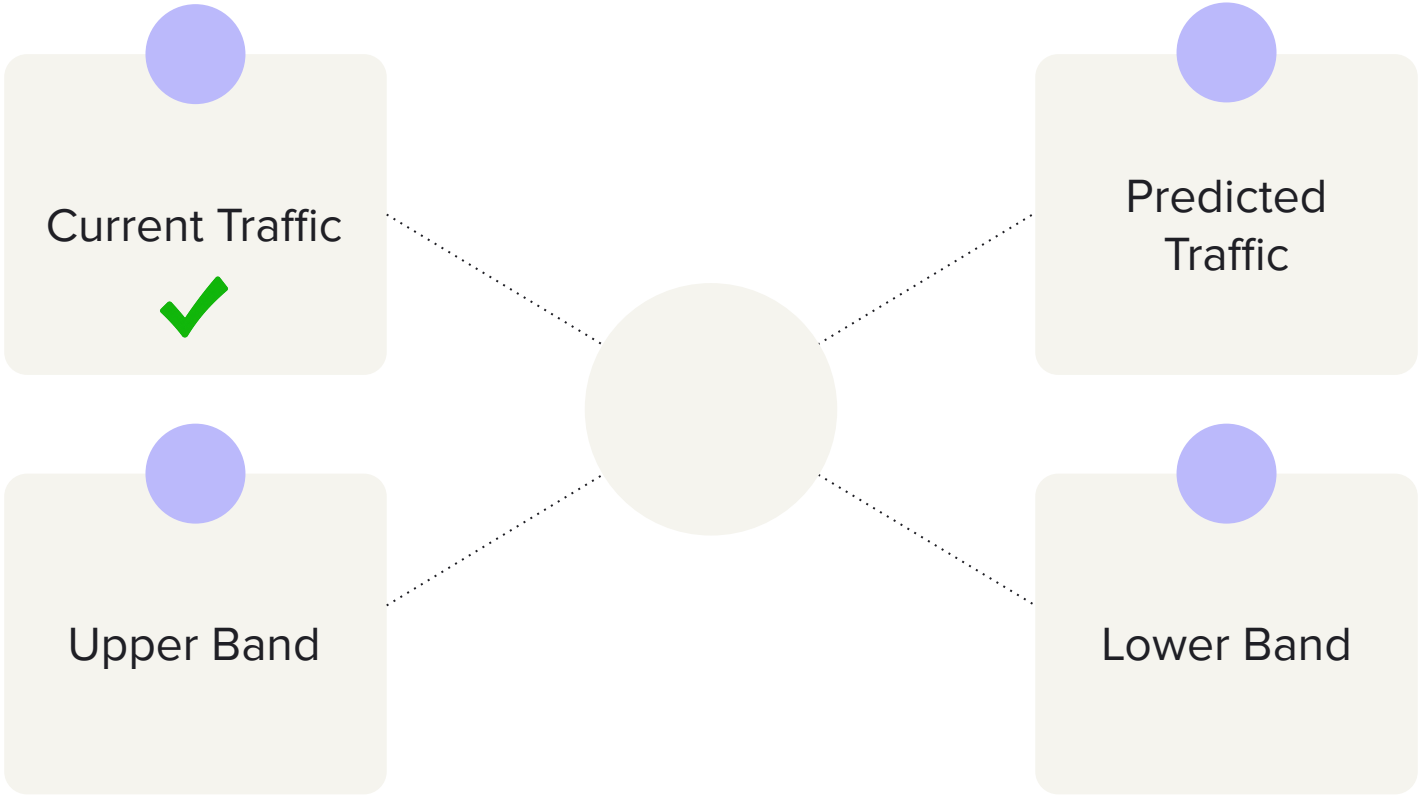
What sport was it playing that night?

Batches of payments are fun until they are not...

Key word:

Anomaly Detection Systems

What do you need for an Anomaly Detection?



Predicting Traffic: can it be this obvious?

Average over a day

```
avg(signal over 1d)
where signal = rate(payment_status[15m] grouped by client_id, method
```

Predicting Traffic: the reality sinks in

Previous day's traffic plus growth over a day

```
avg(signal over 1h, centred on 1 day ago)
    + avg(signal over 1d, now)          ← today's trend
    - avg(signal over 1d, 1 day ago)    ← that day's trend
```

Predicting Traffic: the reality sinks in

Previous day's traffic plus growth over a day

```
avg(signal over 1h, centred on 1 day ago)
    + avg(signal over 1d, now)          ← today's trend
    - avg(signal over 1d, 1 day ago)    ← that day's trend
```

Predicting traffic: history is valuable

Predicting traffic: what a beautiful line!

```
estimate_Nd = avg(signal over 1h, centred on N days ago)
            + avg(signal over 1d, now)
            - avg(signal over 1d, N days ago)

prediction = median(estimate_1d, estimate_2d, estimate_3d)
```

The Lower Band: how low can you go?

Accepted margin of error

```
prediction × (1 - 0.40)          ← margin floor: never tighter than -40%
```

Standard Deviation: wait what?

Standard Deviation calculation

```
stddev_raw      = stddev(signal over 1h)

stddev_filtered = stddev_raw   IF stddev_raw > mean(signal over 1h) × 0.3
  ELSE discard

stddev_smoothed = avg(stddev_filtered over 24h)
```

The Lower Band: naturally deviating from standard

Standard Deviation using prediction

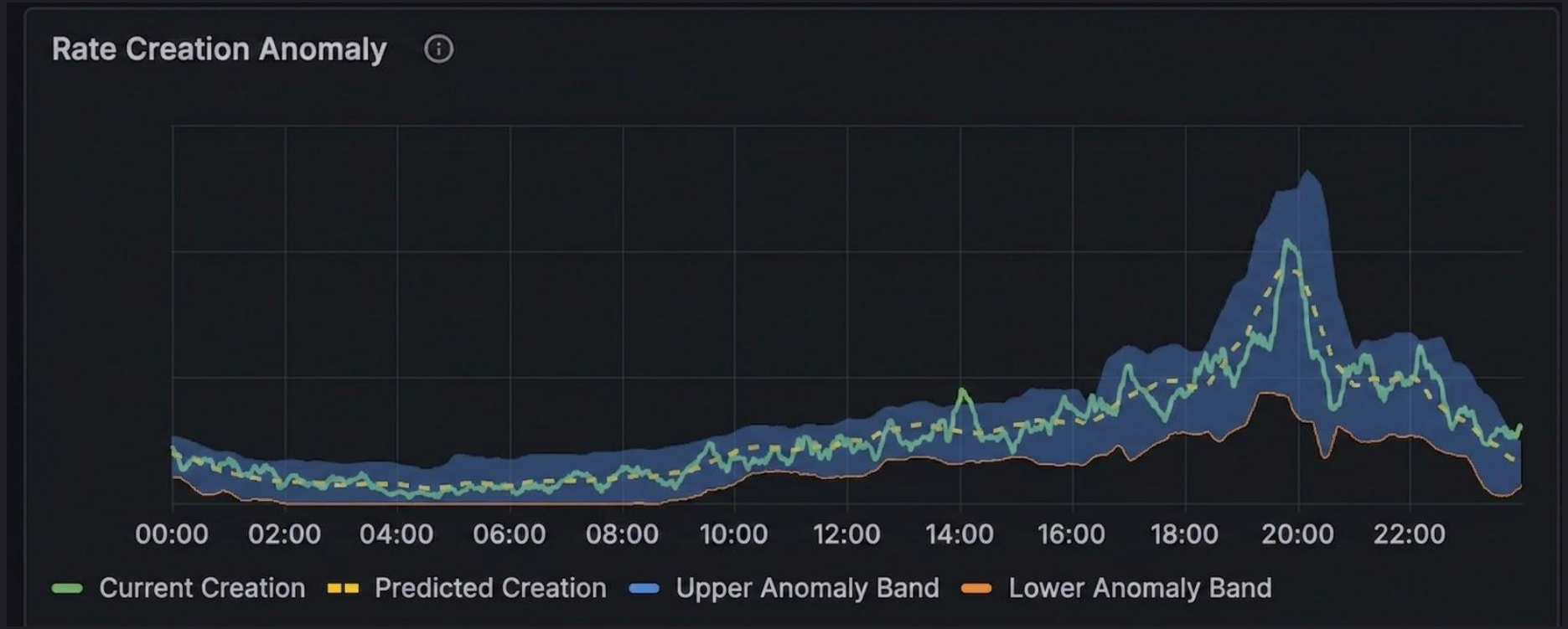
```
prediction - stddev_smoothed,      ← adaptive: data-driven floor
```

The Lower Band: history is repeating itself

Long term bands

```
lt_lower = min(
    avg(signal[1h] centred at 1d ago) - 2 × stddev(signal[1h] centred at 1d ago),
    avg(signal[1h] centred at 3d ago) - 2 × stddev(signal[1h] centred at 3d ago)
)
```

Voila!


Rate Creation Anomaly
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
Current Creation
Predicted Creation
Upper Anomaly Band
Lower Anomaly Band

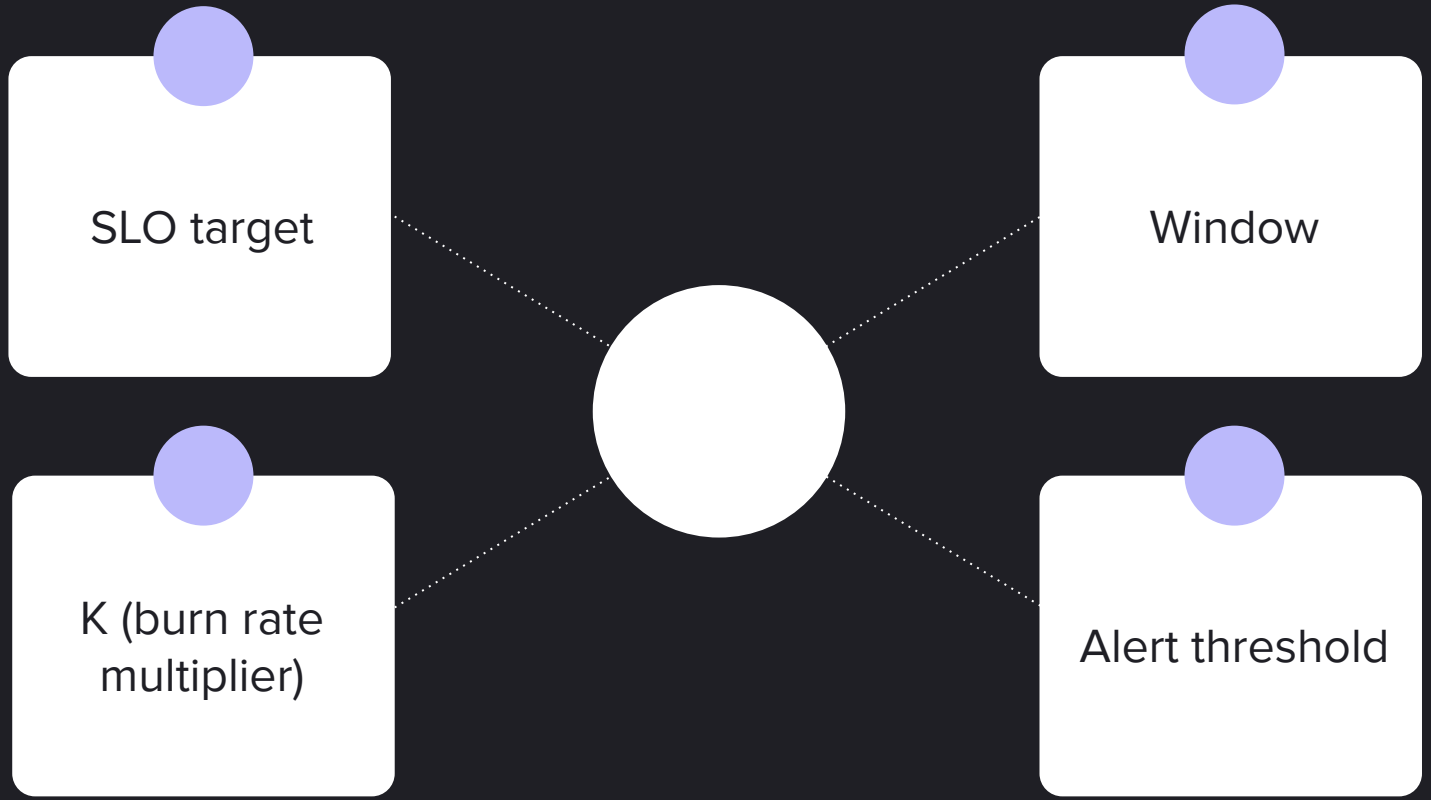
What should we monitor?

Building alerts that are not too noisy

First Draft: Static Alerts

Current traffic below Lower Band	→	Configured per metric to identify different breaking points
Over 45 mins	→	Long enough to be a significant downtime for any client
Static List of clients	→	Only Tier 1 and Tier 2 clients manually added to the list after traffic is ramped up

Burn Rates: burn baby burn!



V2: Burn Rate and Tiers

	Drop Threshold	Burn rate multiplier	Breach to alert
Very High Traffic	15% drop in threshold	Burn rate 2	> 30% of samples
High Traffic	25% drop in threshold	Burn rate 2.5	> 62.5% of samples
Medium Traffic	20% drop in threshold	Burn rate 3	> 60% of samples * OVER LONGER WINDOW

Burn Rate Alerts

Dynamic clients depending on traffic	→	This value gets also update as traffic fluctuates so alerts become more sensitive with movement
Different windows per Tier	→	Each tier evaluates to answer differently to how long to measure over
Dual band gate	→	Current vs Predicted AND Lower Band = confidence + speed

A merchant broke their own integration. We noticed before they did.

- Anomaly alert fired 90 mins after their deployment (medium traffic tier)
- Change was reverted before the zero-rate alert even triggered

30 minutes. That's how fast we knew.

- Banking partner with low traffic started degrading — alerts fired within 30 mins
- Almost an hour before any other system flagged it

For you to take back to the office...

What information do you have?

What do you need to know?

What should you monitor?

Define your data

- Client ID
- Payment Method
- Status
- Provider ID

Define your aggregates

- Volume per status
- Conversion between status
- Prediction
- Anomaly Bands

Define your alerts

- Tiers
- Ranges and limits
- Process when triggered

Additional References

Anomaly detection framework: https://github.com/grafana/promql-anomaly-detection/tree/main

Prediction detection improvements: https://about.gitlab.com/blog/anomaly-detection-using-prometheus/

Burn Rates Multi-window explanation: https://sre.google/workbook/alerting-on-slos/

Thank you