



Server Side Tracking with Snowplow

Paul Ferrett
Head of Development, Catch Group
Snowplow Analytics Sydney Meetup - June 25, 2019

A brief history of data at Catch

- Not so long ago we used to do things like most others...
 - **Google Analytics** for website tracking
 - **Custom reports** in our back office systems
 - **TM1** for sales reporting, planning, and forecasting
 - **Bespoke** prediction models for stock movement and warehousing
 - Daily **email reports** from the “Reporting Server(s)”
 - Rely on dashboards within **individual business systems** (Campaigns, marketing, personalisation)
- Everything grew ad-hoc, and there was no centralised “BI” team or platform
- *... and rarely did they communicate the same numbers!*

We do things better now

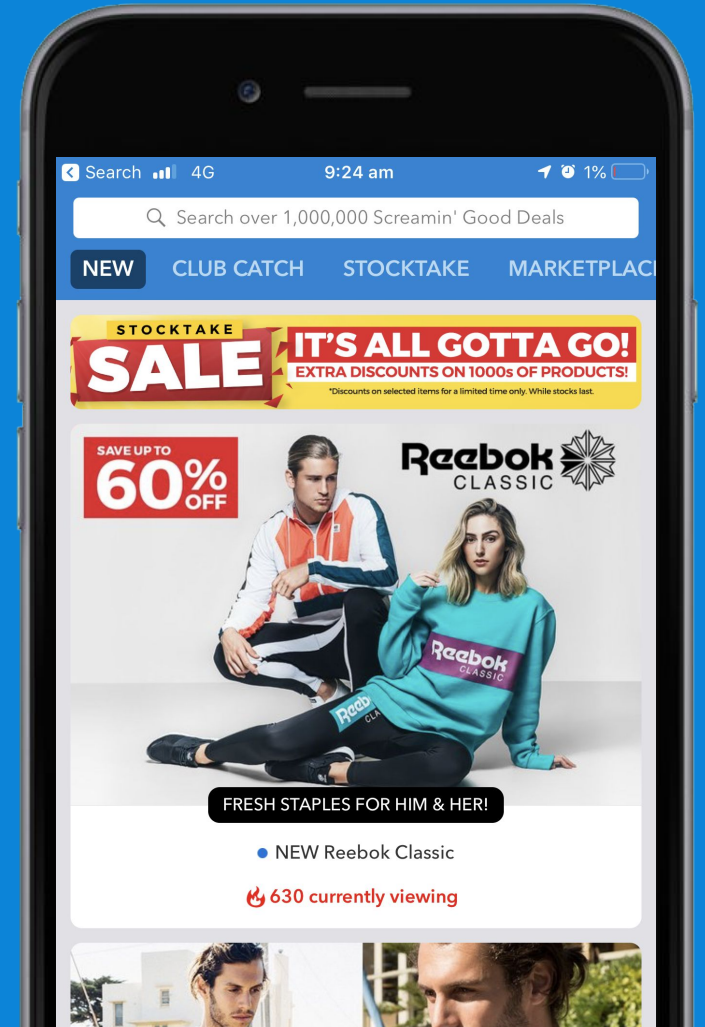
- Centralised BI team
- Data lake in S3/Redshift
 - Snowplow Batch Pipeline
 - External data loaded in
 - Internal databases feed in
- Dashboards in Periscope
- *...and people almost never look at GA!*

Quick Snowplow stats

- Last 12 months
 - 3TB data collected
 - 25B rows
- Daily growth
 - 500k sessions
 - 95M records
- Monthly growth
 - 2.9B rows
 - 350 GB

Why Server Side Tracking?

- Avoid sending context data via API
- Decouple tracking changes to app releases
 - Started tracking the app with no release
 - force-update = uninstalls
 - Able to make schema changes more often
 - Fix data issues on the API side
- Multiple platforms with less work ...one day



Our Server Side Tracker

- Built from scratch in PHP
- Emits data to the collector after client request complete
- Uses the same context building code as the front end
- `/tracking` endpoint for interactions
- Pluggable Emitter
 - Curl; Send to the Snowplow Collector
 - Logfile; Simple debugging
 - Event bus; Websocket connector, retryable queue for collector errors, Microservice consumers

Our biggest problem; Verifying server side data sent to the collector!

The screenshot shows the Poplin Data interface with several elements highlighted by green circles:

- Stream Websocket Data**: A button in the top navigation bar.
- catch_api**: The APP name in the event details section.
- websocket**: The COLLECTOR name in the event details section.

The interface includes a top navigation bar with buttons: Clear Events, Clear Schema Cache, Import Bad Rows, Stream Live Data, Stream Websocket Data, and Import HAR Session. A left sidebar contains a Filter input and a list of event types: Websocket Data, Pageview, SD Event: catalogue_view, SD Event: event_click, and SD Event: add_to_cart. The main content area displays event details for 'catch_api' (APP) and 'Pageview' (EVENT), with a time of 'Tue, 25 Jun 2019 02:40:51 GMT'. Below this, it shows 'websocket' (COLLECTOR) and 'GET' (METHOD). At the bottom, a 'Beacon' section is visible with 'Event Type' set to 'Pageview' and a 'string' button.

Benefits of **Client Side**

- 👍 Simple build & verification workflow
 - Open source tracker + Poplin extension
- 👍 Discover environment + browser capabilities
- 👍 No application load for interaction tracking

Benefits of **Server Side**

- 👍 Not impacted by ad-blockers
- 👍 Unaffected by Browser / client network errors
- 👍 No sending/leaking of (secret) context data
- 👍 No front end release required

Thank you!