Wolfram Wingerath

Big Data Analytics With AWS Athena

Big Data

code.talks

Big Data Analytics With AWS Athena

Wolfram Wingerath, Code.Talks 2019

Baqend

I Am Wolle

Research:
- Stream Processing
- Real-Time Databases
- NoSQL & Cloud Systems
- …

Practice:
- Web Caching
- Big Data Analytics
- Anger Management
- …

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

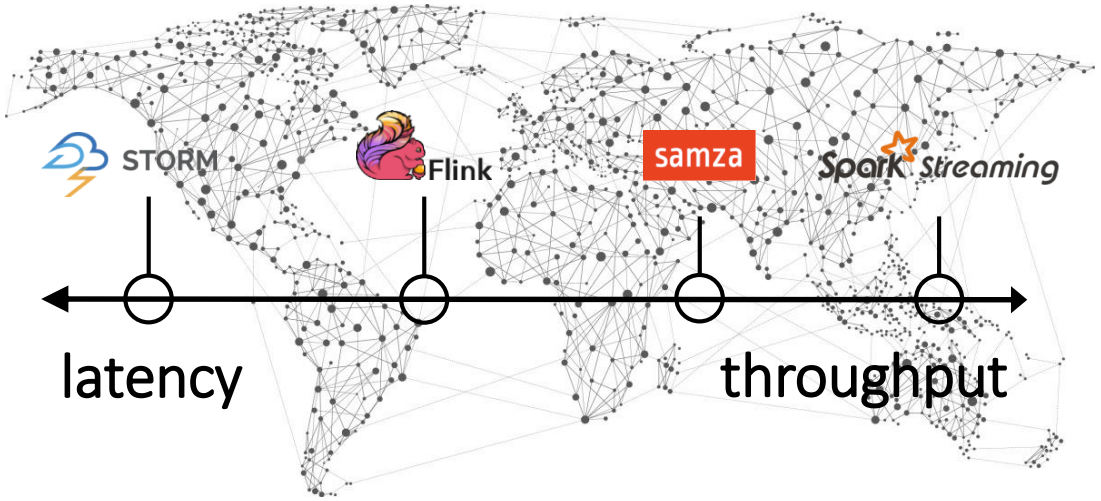
Baqend

I Like Real-Time Stuff

2018

Real-Time Processing Explained

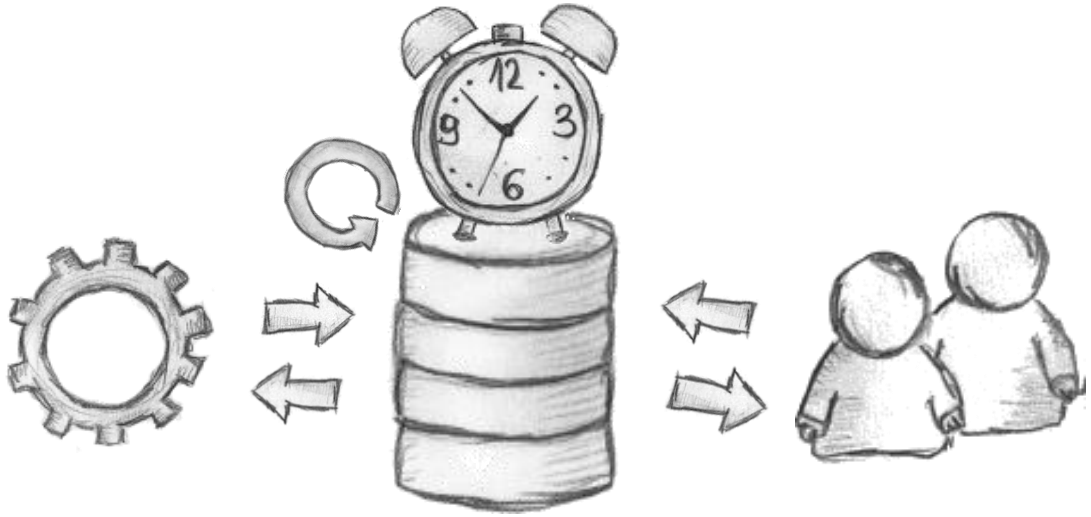
A Survey of Storm, Samza, Spark & Flink



2017

Real-Time Databases Explained

Why Meteor, RethinkDB, Parse & Firebase Don't Scale



METEOR RethinkDB Firebase BaQend

Let's Talk About Batch Analytics

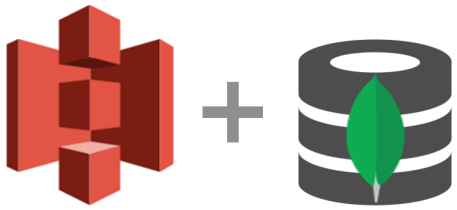
Collection

Ingestion

Analytics

Reporting

Tracking
(RUM)



SQL Interface

Performance
Dashboard

QA
Dashboard

Real-Time
Alerting

Ad-hoc
SQL Interface

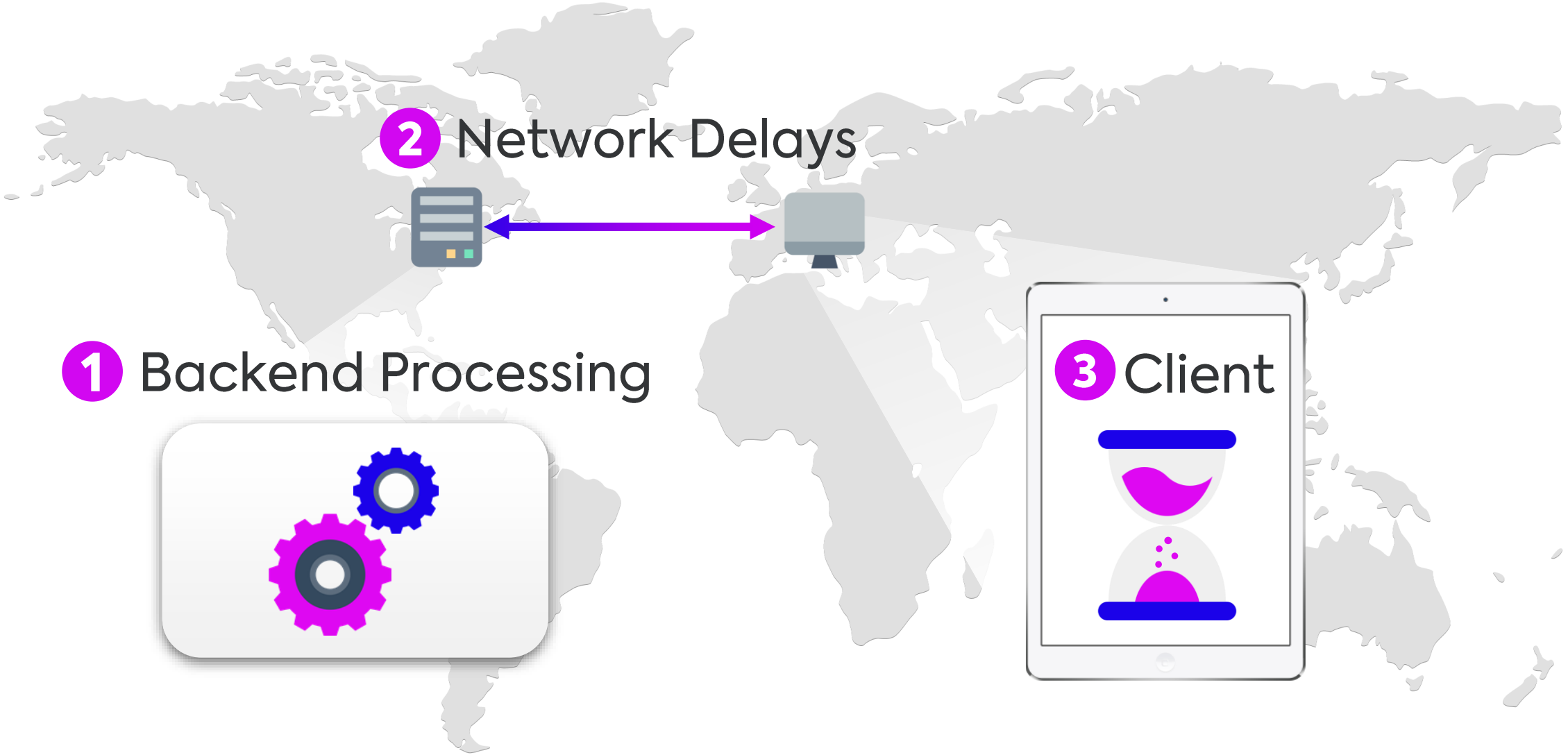
Custom
Reporting

- Raw PI tracking & meta data
- Custom tracking

- Materialized views & aggregations
- Historical data


What‘s in the Data?

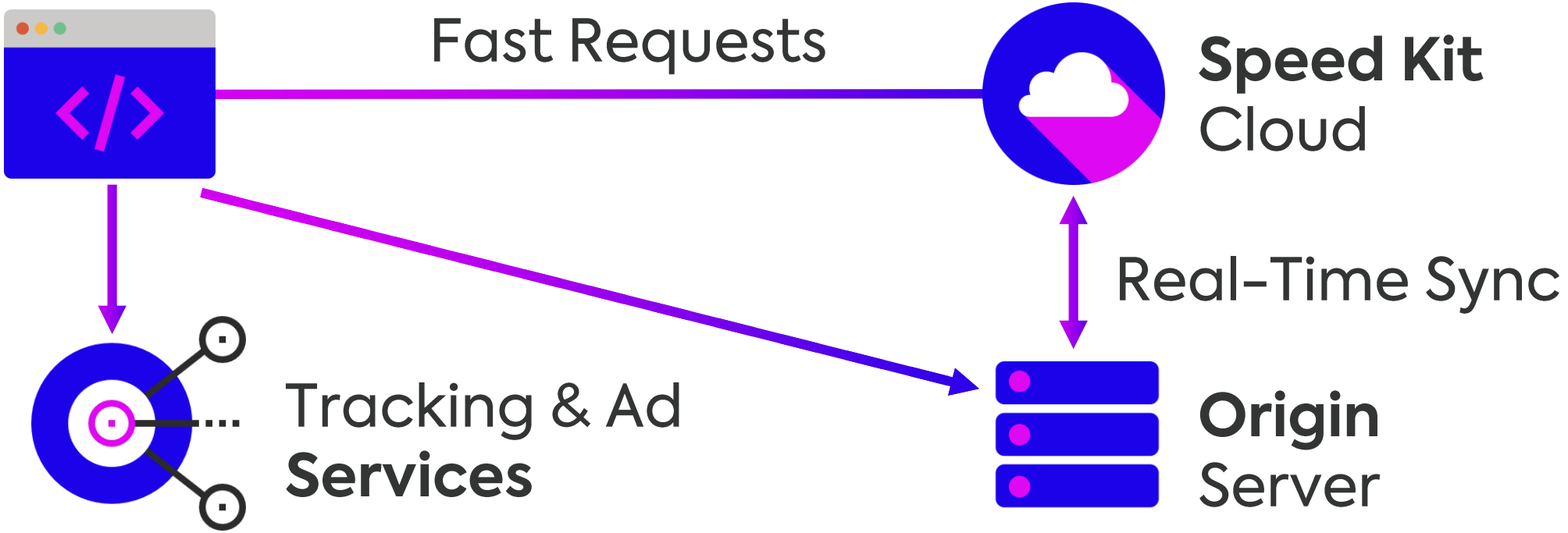
3 Things Make Your Website Slow



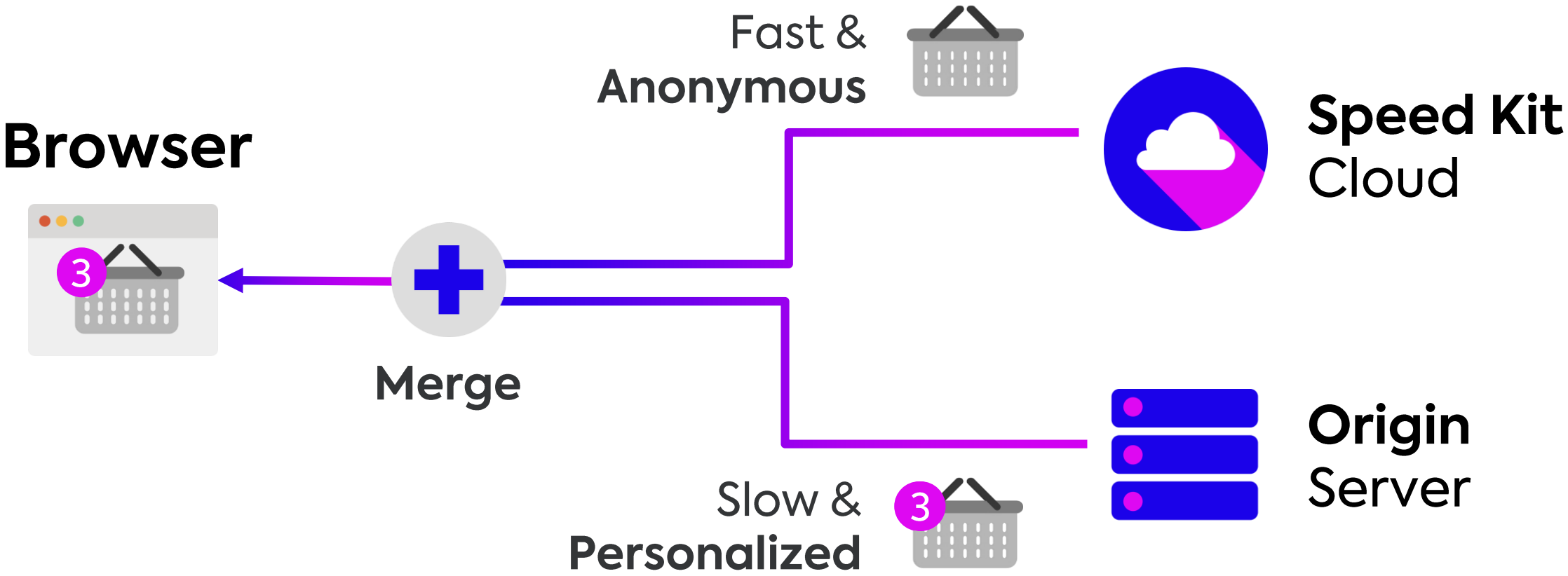
We Make Websites Fast

Website + **Speed Kit**
(Service Worker)

Kino 7
Friday, 16:00



Accelerating Personalized HTML



Split Testing for Web Performance

Speed Kit Users

- Speed Kit enabled

Tracking

vs.

Measurable uplift:

+ Performance
+ User engagement
+ Business success

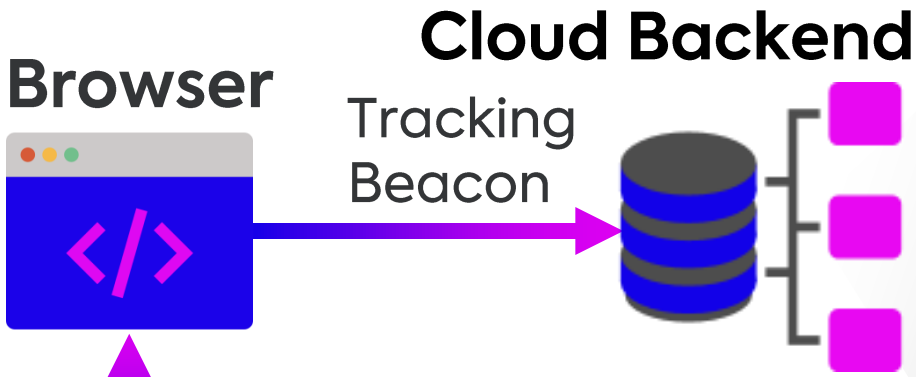
Tracking

Normal Users

Kino 7
Friday, 11:00

- Speed Kit disabled (no acceleration)

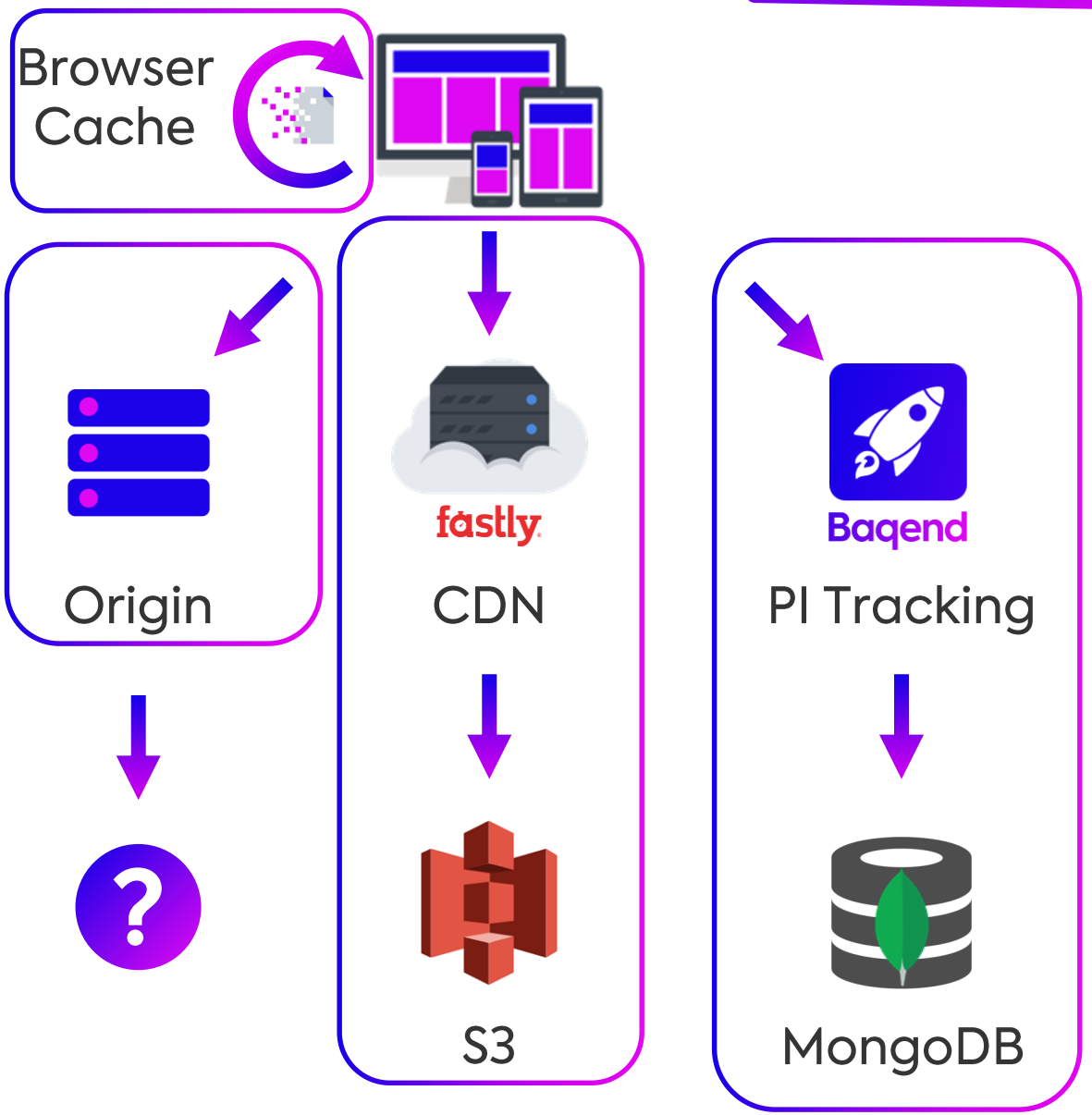
Goal: Performance & Business Insights



- Timing API
- Service Worker
- Unhandled Errors

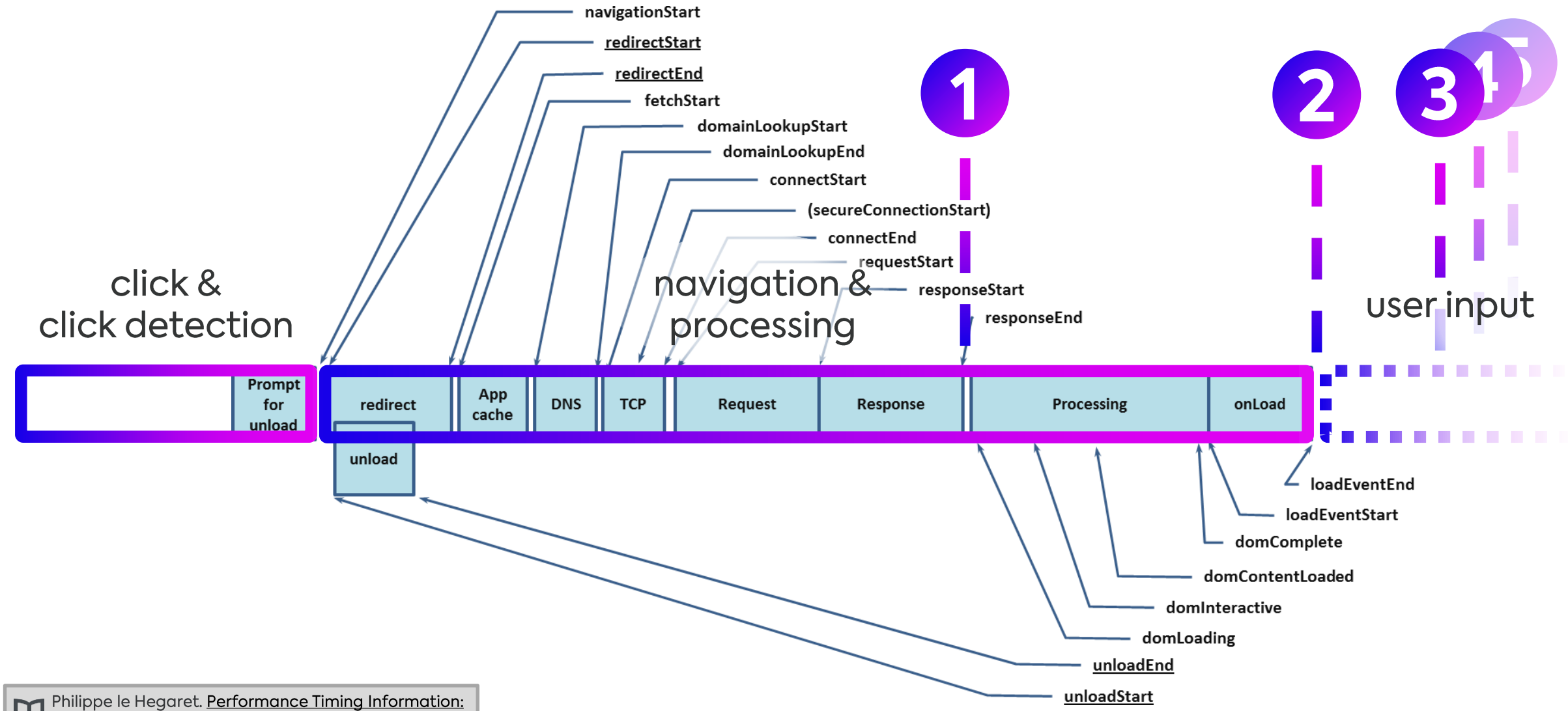
▪ Time-to-First-Byte ▪ First (Contentful) Paint ▪ DOM Timer ▪ First Input Delay	Performance
▪ Session Length ▪ Time on Site ▪ First User Interaction ▪ Bounce Rate	User Engagement
▪ Cart Size ▪ Transactions ▪ Conversion Rate ▪ Revenue	Business KPIs
▪ Page Views & Sessions ▪ Browser Distribution ▪ JavaScript Errors ▪ Caching Insights	QA Metadata

How to Monitor Performance?



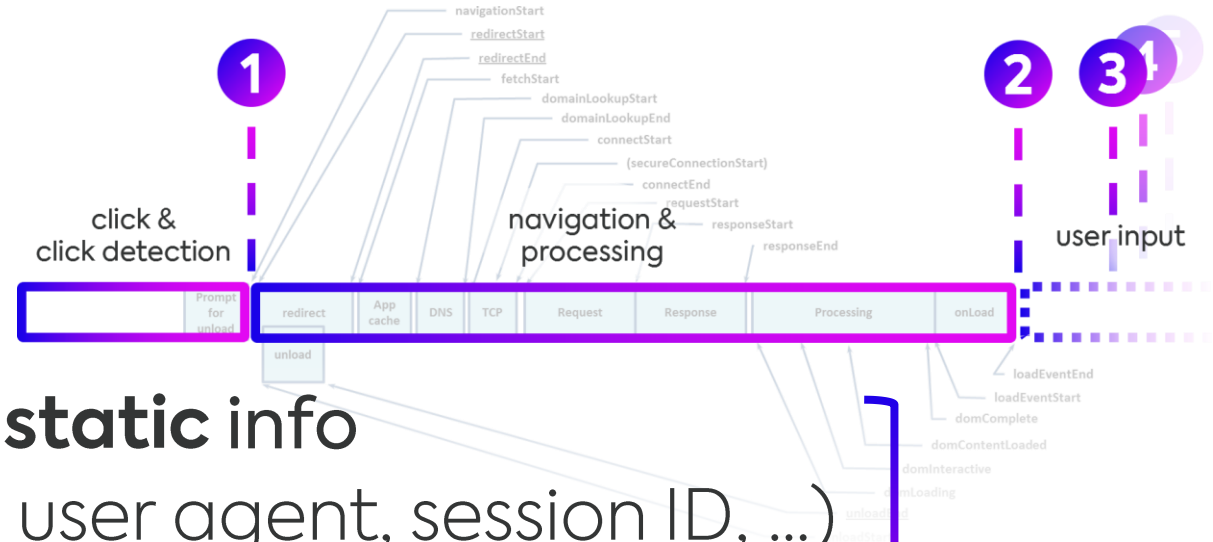
- Logging requests is not enough:
  - ✗ User? Rendering? ...
  - ✗ Browser cache (invisible)
  - ✗ Origin requests (no logs)
  - ✓ CDN requests
- Solution: **Tracking every PI** (page impression)

When to Send Data Beacons?



Philippe le Hegaret. Performance Timing Information: Part 1 – Navigation Timing, W3C Blog, 2012

Types of Data Beacons



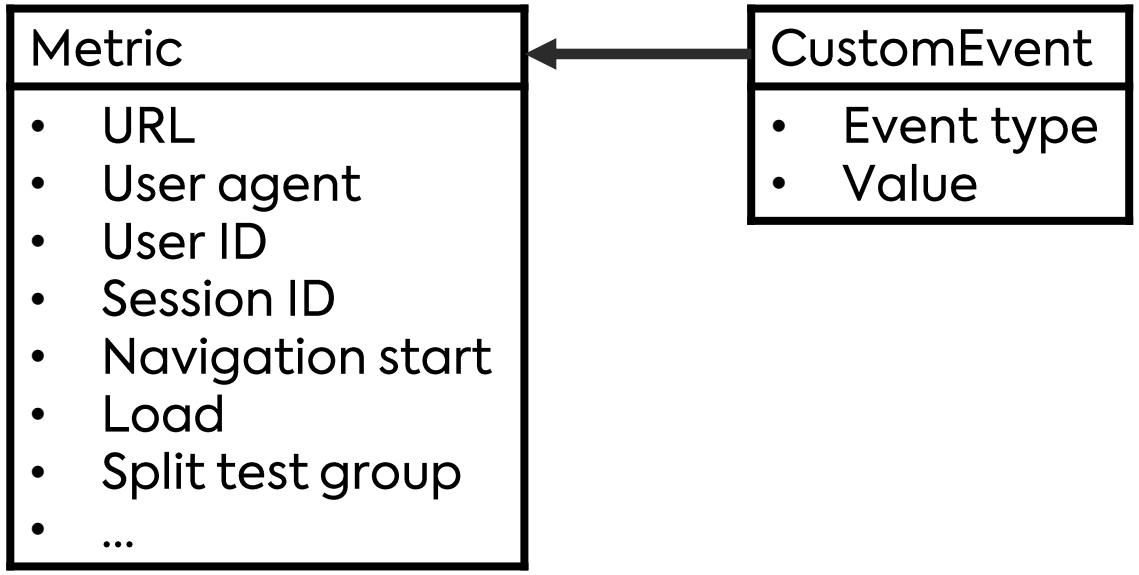
1. 1 for **static** info
   (URL, user agent, session ID, ...)
2. 1 for **timings**
   (TTFB, load time, FCP, ...)
3. 0-n for **custom** events
   (first input, add-to-cart, ...)

Metric Beacon

CustomEvent Beacon

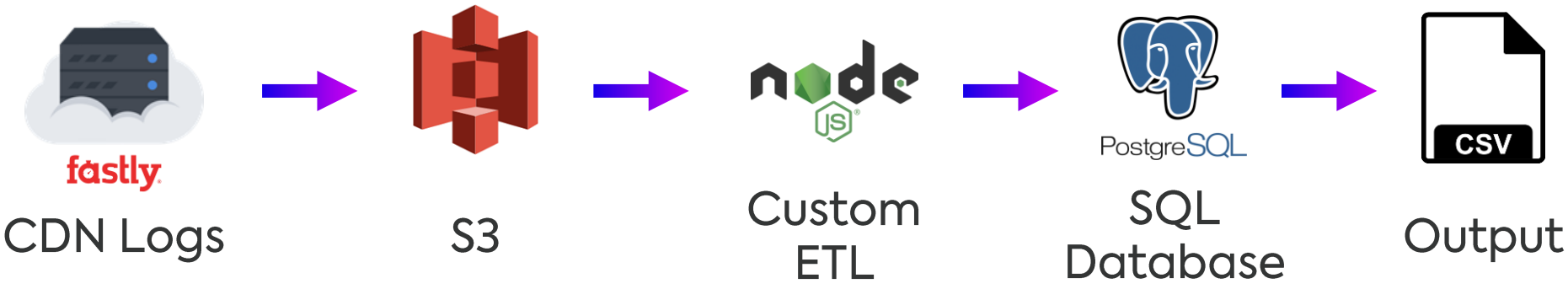

Let‘s Analyze
the Data

Tracking Data in MongoDB



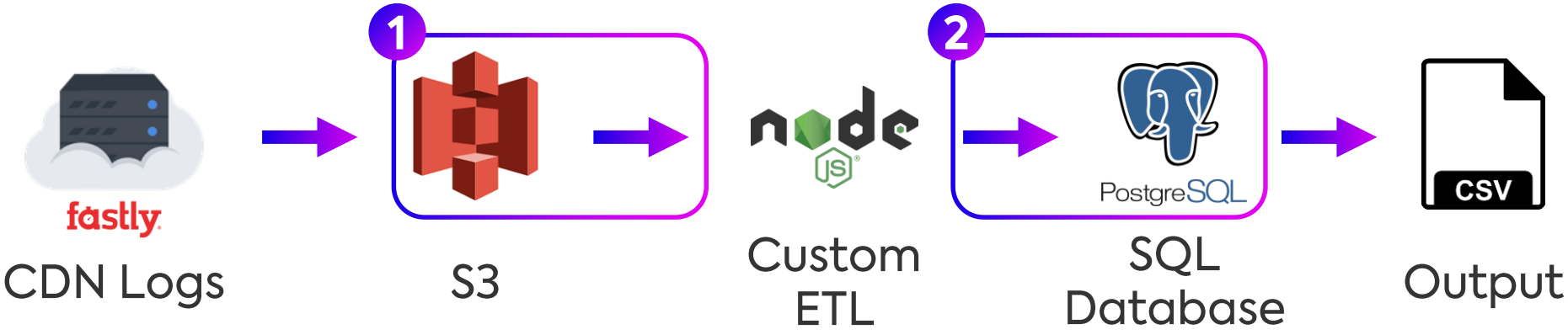
- Debugging with OLTP **queries**
- Analyses with **aggregation pipeline**, e.g.:
  - Average session length
  - Uplift vs. Acceleration
    (example: Conversion rate vs. first paint uplift)
  - Different browser timings by device/browser/...

CDN Logs in S3



- Automation via Jenkins jobs
- **Caching-related** statistics, e.g.:
  - Request latency distribution (histogram)
  - Image optimization efficiency

Problems I: CDN Data Import



1. **Partitioning** by hour, but <u>not</u> by customer
   → Not scalable!

2. **Indexing** & database import:
   a) Import into indexed table
   b) Drop index → import → create index
   c) Query table without index

   } each takes forever

Problems II: Aggregation Pipeline

Indexing

Queries over non-indexed attributes were infeasible

Runtime

Even with indexes in place, queries could take 30+ min.

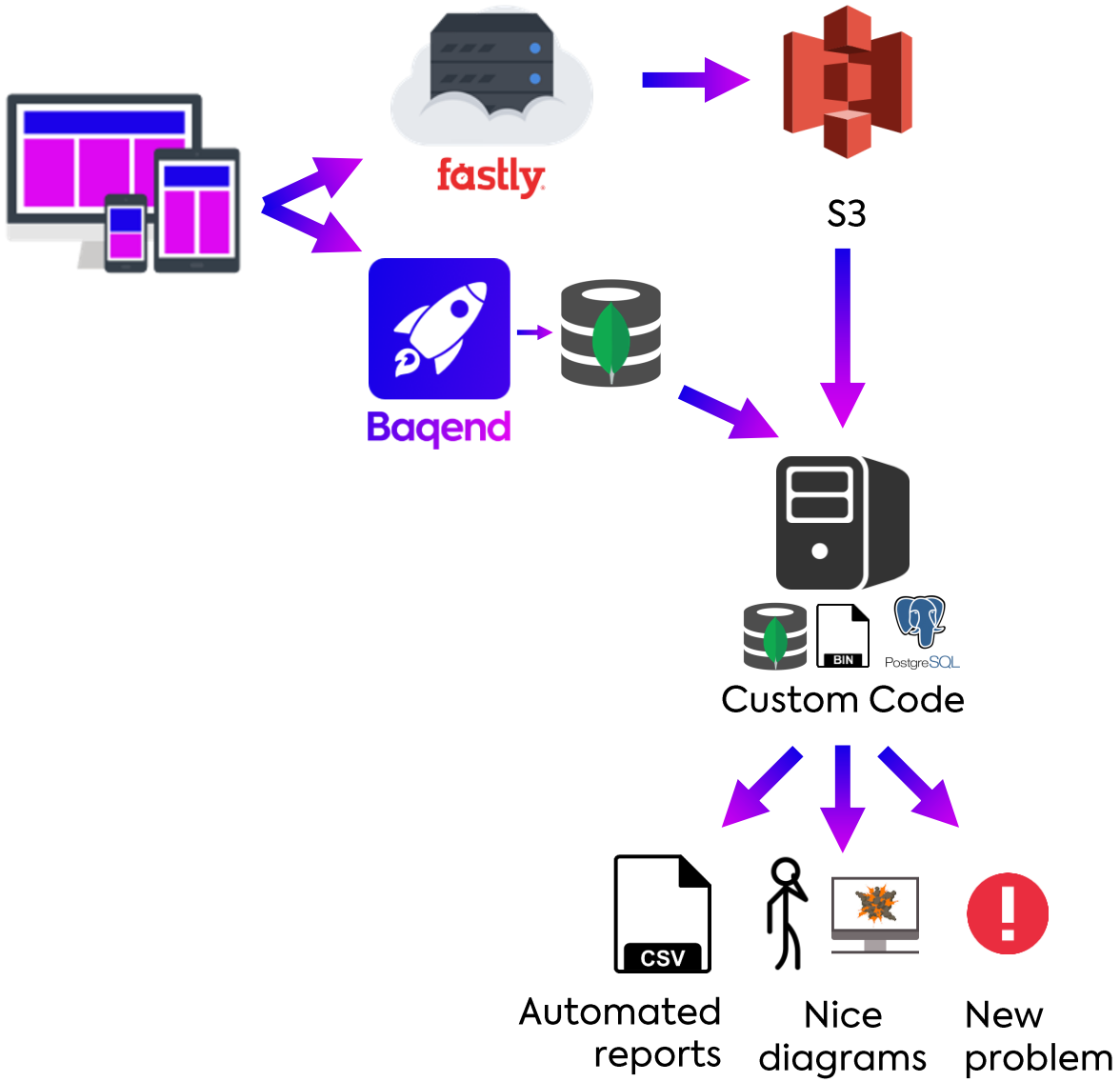
Scalability

Queries got slower with increasing amounts of data

Complexity

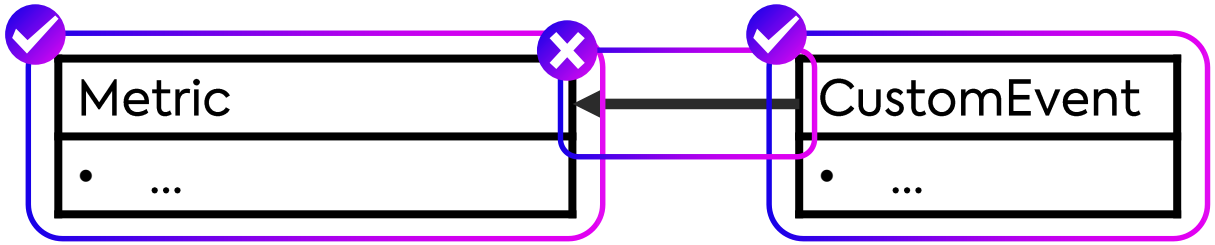
MongoDB aggregation pipelines become sophisticated quickly

Problems III: Reporting

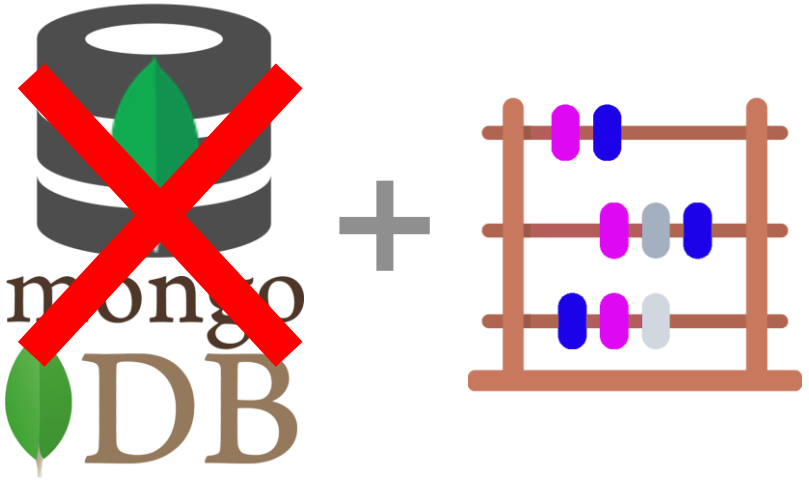


- Scheduling:
  1. On-demand
  2. Periodic job
- Problems:
  - Cumbersome to build & maintain
  - Awkward to extend
  - Unreliable

Problems IV: Joins in MongoDB



- ✕ **Joins infeasible** for data-intensive queries!
  - → No conversion analysis
  - → No business uplift validation
  - → **Not acceptable!**




Fixing My Life With
~~Flex Tape~~ Athena

The „A“ Stands for „AWSome“

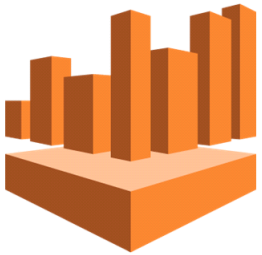
- Desperate **attempt**:
  1. Dump MongoDB collection
  2. Upload to S3
  3. Query with Athena

- Typical analysis:
  - 1 equi-join
  - 3 mio. PIs
  - ~15+ min.

The „A“ Stands for „AWSome“

✓ ~~Desperate **attempt**:~~New best practice:

1. Dump MongoDB collection
2. Upload to S3
3. Query with Athena



AWS Athena

- Typical analysis:
  - 1 equi-join
  - 3 mio. PIs
  - ~10 **seconds**

What's an Athena?

Athena

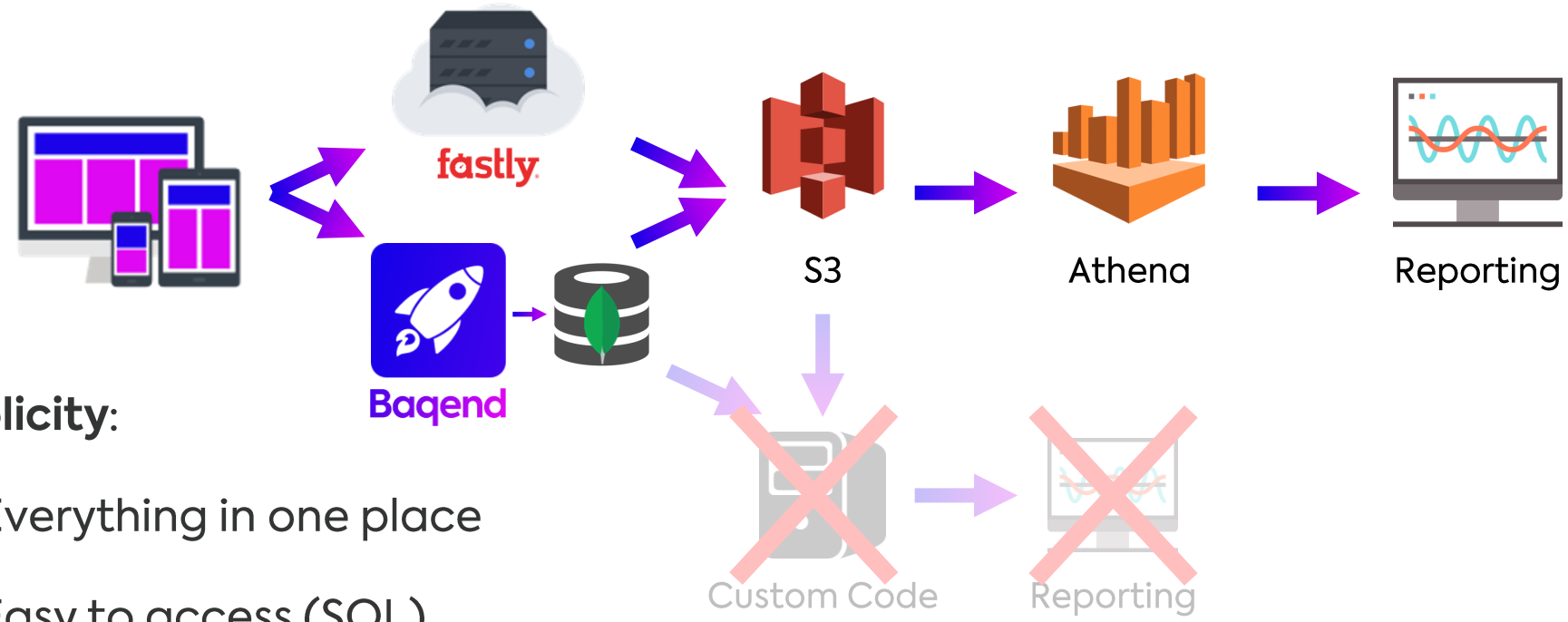
=

presto

- **Managed Presto**:
  - Interactive analytics with SQL
  - Heterogeneous datastores
  - Petabyte-scale (Facebook)
- **Pricing** by scanned data volume:
  - → Efficient storage formats!
  - → Partitioning or clustering!
  - → Careful query design!

Raghav Sethi, Martin Traverso, Dain Sundstrom, David Phillips et al.. Presto: SQL on Everything, ICDE 2019

Upgrading Our ETL Pipeline

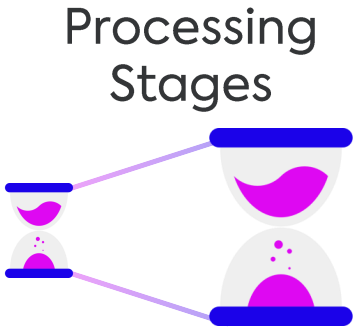


- **Simplicity**:
  - Everything in one place
  - Easy to access (SQL)
- **Scalability & efficiency**:
  - Hundreds of gigabytes scanned in a query
  - Response time on the order of seconds

Processing Stages & Latency

Alerting

- Simple metrics
  - Counters
  - Extreme values
  - Specific errors



Trend Analysis

- Complex aggregations
  - Conversion rate
  - Performance by day
  - Seasonal effects

Stage 0: Data Preparation

```
{
  "_id": "ABC",
  "loadEvent": {
    "$numberLong": "1571101211368"
  },
  "createdAt": {
    "$date":"2019-10-15T01:00:11.462Z"
  },
  ...
}
```

_id	loadEvent	createdAt	…
ABC	Oct. 15, 2019 1h 0m 11s 368ms	Oct. 15, 2019 1h 0m 11s 462ms	…
…	…	…	…

- Schema Definition
  1. **Tables** for raw data
  2. **Views** on top to hide artifacts

Example: Timestamps

```
FROM_UNIXTIME(
  CAST(
    CAST(
      JSON_EXTRACT(loadEventEndRaw, '$["$numberlong"]')
    AS VARCHAR)
  AS DECIMAL (38,3)) / 1000
) AS loadEventEnd
```

1. Extract UNIX timestamp from JSON
2. Cast to varchar
3. Cast to decimal
4. Divide by 1000
5. Convert to timestamp

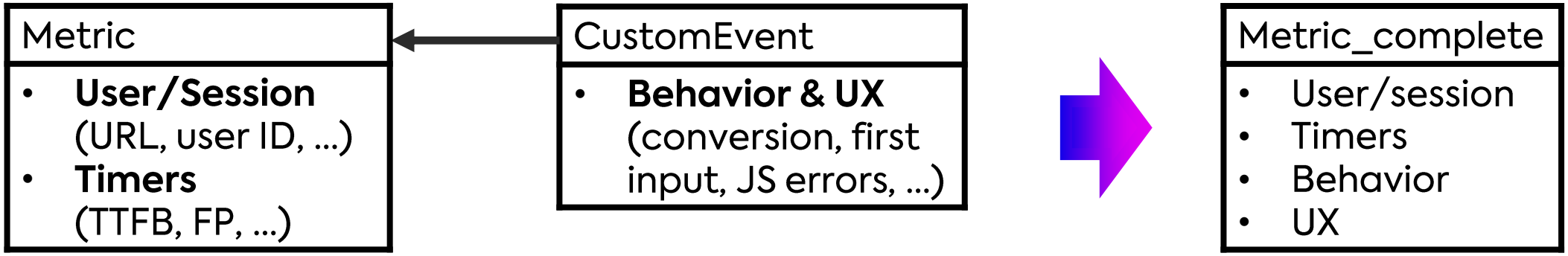
Example: Timestamps

```
CASE WHEN
  CAST(
    CAST(
      JSON_EXTRACT(loadEventEndRaw, '$["$numberlong"]')
    AS VARCHAR)
  AS DECIMAL (38,3)) > 0
THEN
  FROM_UNIXTIME(
    CAST(
      CAST(
        JSON_EXTRACT(loa                                  "]')
      AS VARCHAR)
    AS DECIMAL (38,3)) /
  )
END AS loadEventEnd
```

Note: still a timestamp (no timing in ms)

1. Extract UNIX timestamp from JSON
2. Cast to varchar
3. Cast to decimal
4. Divide by 1000
5. Filter out rubbish
6. Convert to timestamp

Stage 1: Join Beacons

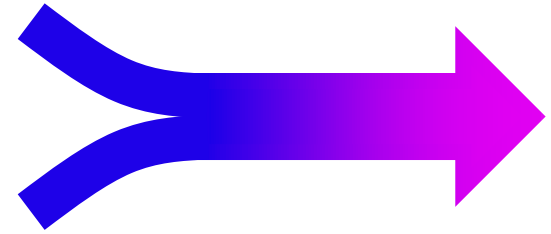


- Consolidate PI data into **single rows**
- Data **cleaning**
  (e.g. nullify when `loadEnd < navigationStart`)

Stage 2: Resolve User Agents

Metric_complete
• User agent • …

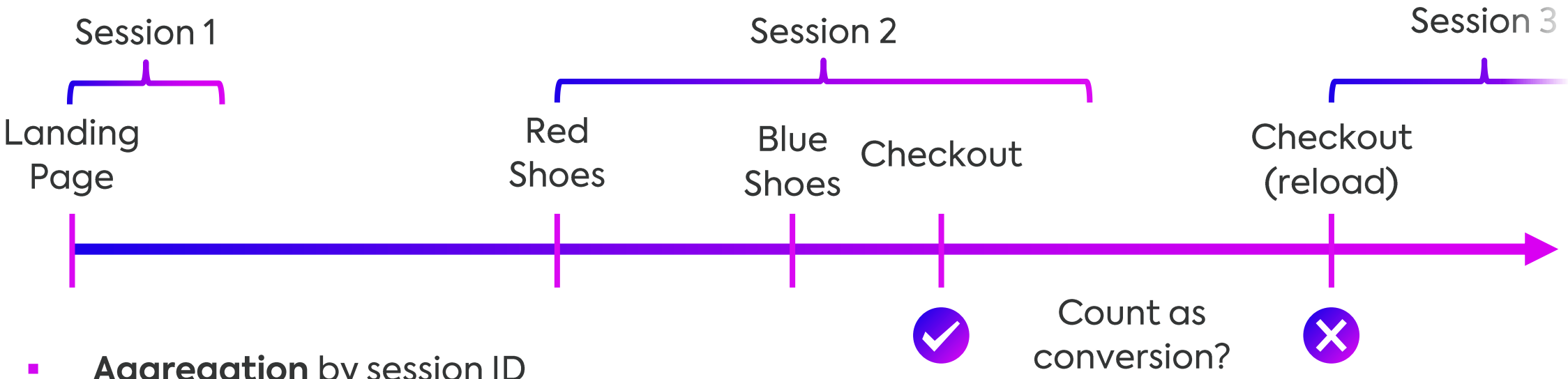
WhatIsMyBrowser.com



Metric_complete
• browser (+ version) • Device type • Device • OS (+ version) • …

- **Paid service**: interpreting user agents is complex!
- **Fallback**: simple `case-when` logic for browsers and device type
- **Simplification** required, e.g.:
  - Device: mobile/desktop/tablet~~/server/game console/wearable/vehicle/…~~
  - Browser: Chrome/Firefox/Safari/Opera/IE~~/Netscape/Tesla Browser/…~~

Stage 3: Session Metrics



- **Aggregation** by session ID
  - Session length, bounces
  - Time on site
  - Performance (e.g. median FCP)
  - Conversions

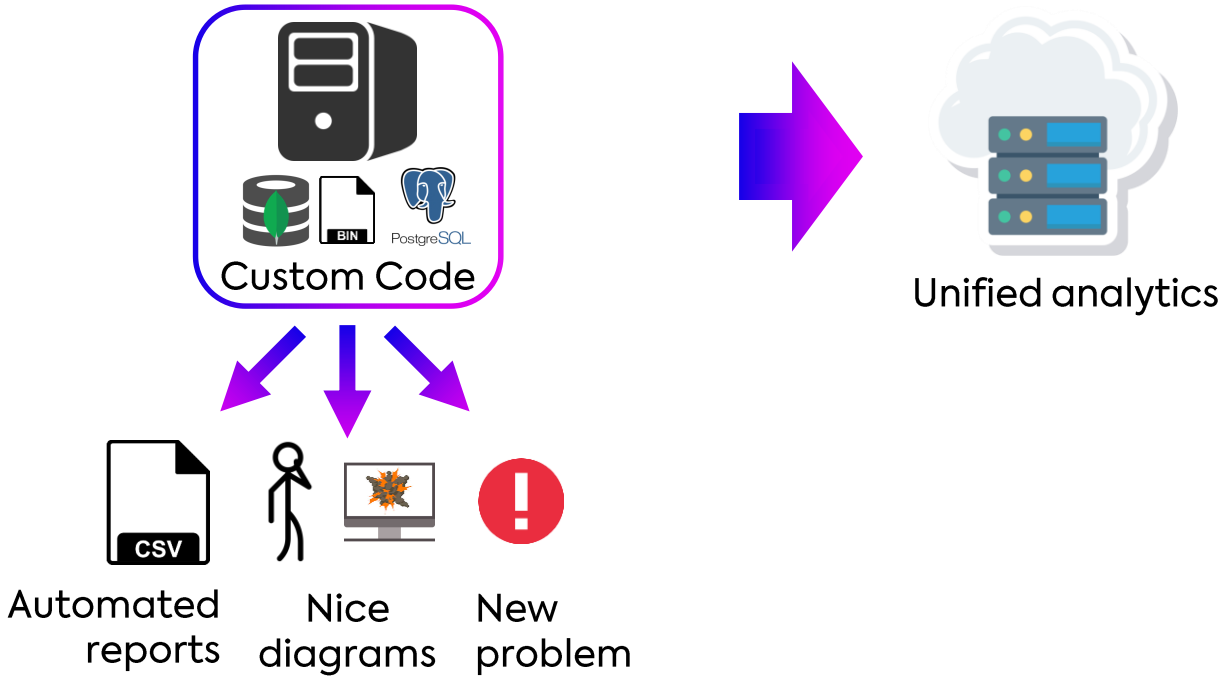
Stage 4: Materialized Views

Business uplift during Proof of Concept (PoC)

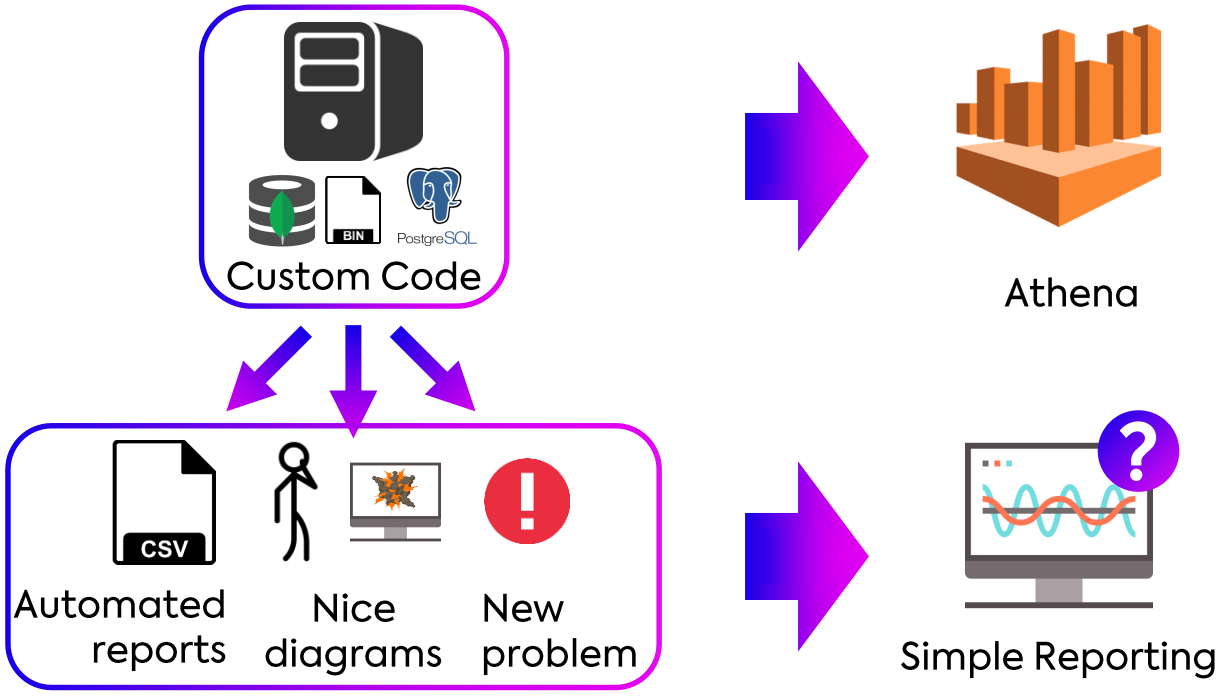
Aggregation over days or weeks

Performance and business trend analysis

Reporting: The Right Tool for the Job

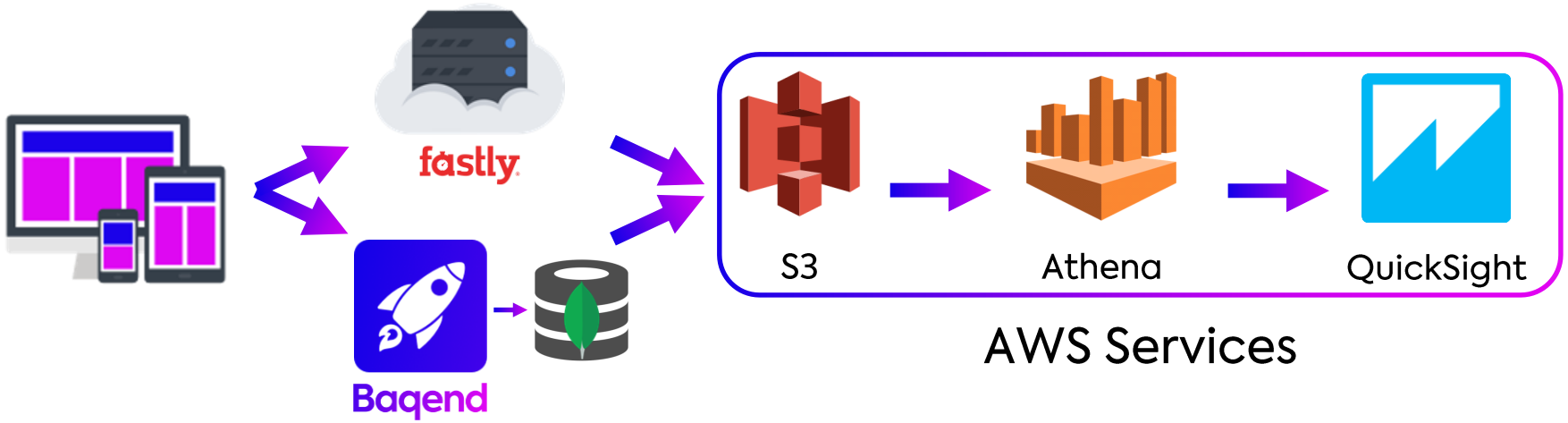


Reporting: The Right Tool for the Job



- **Requirements**:
  - Automation
  - Easy data exploration
  - Robustness

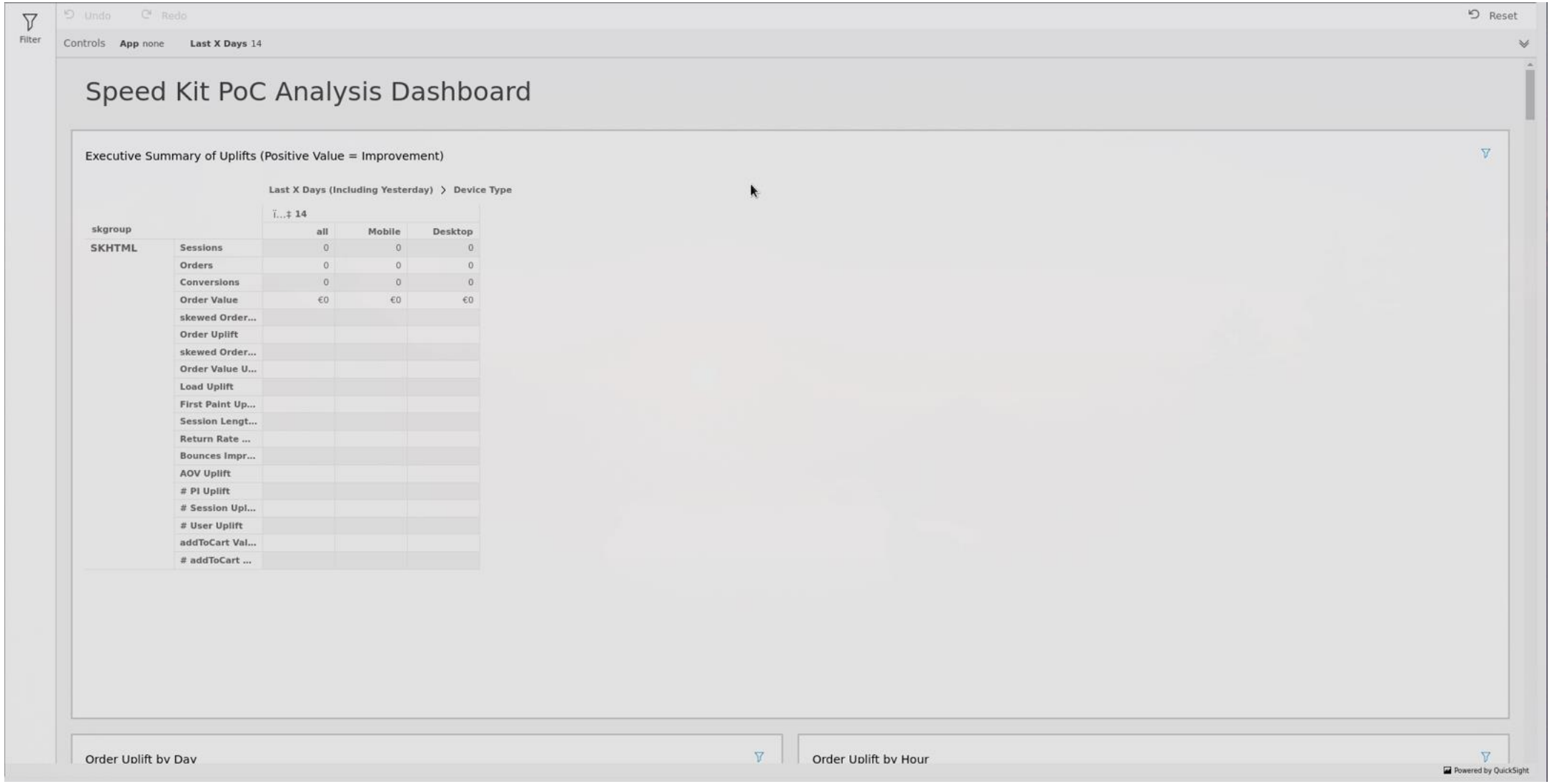
Why QuickSight?



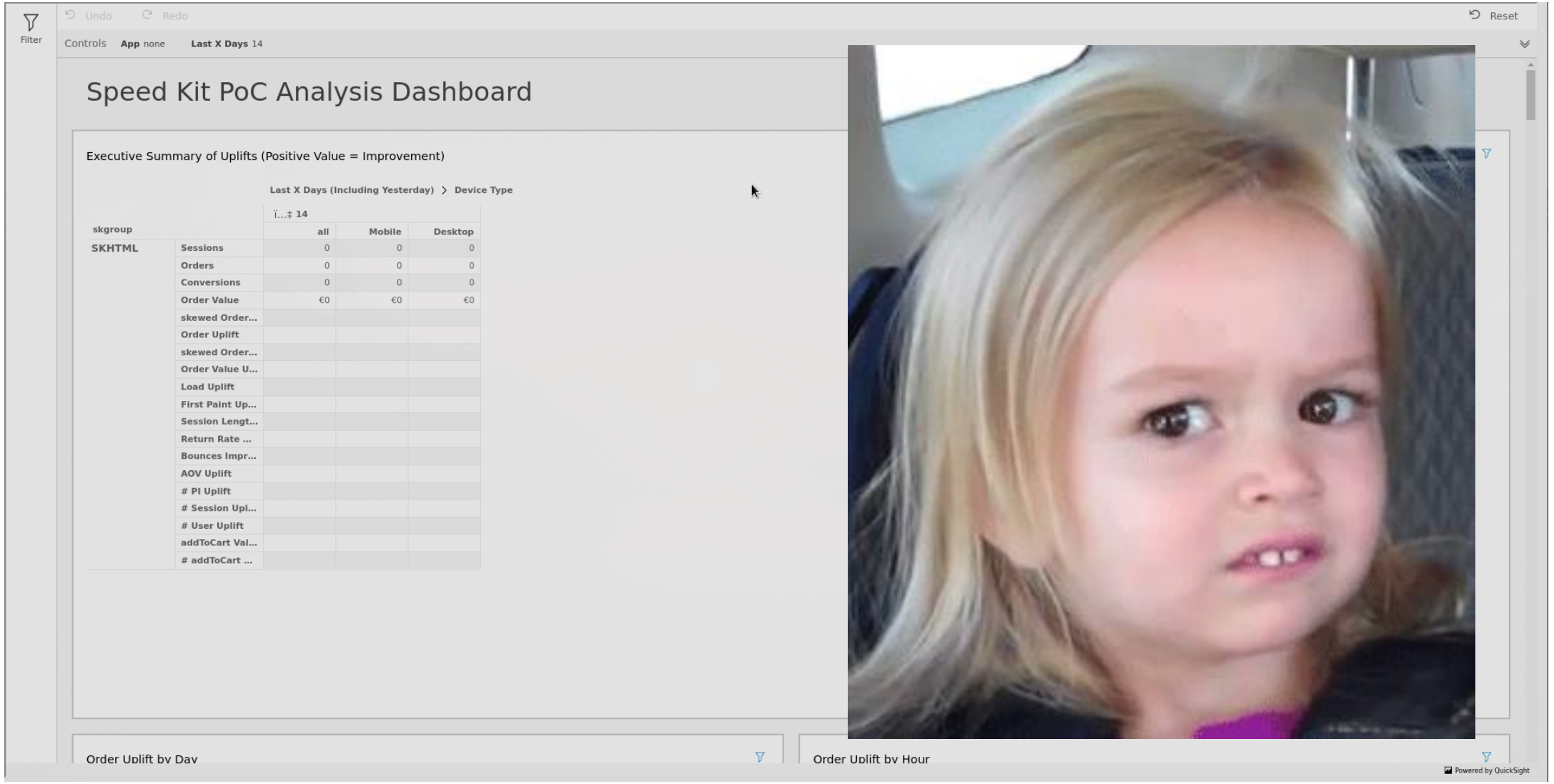
- Easy integration with Athena
- Ease-of-use
- Quick results


Worst Practices

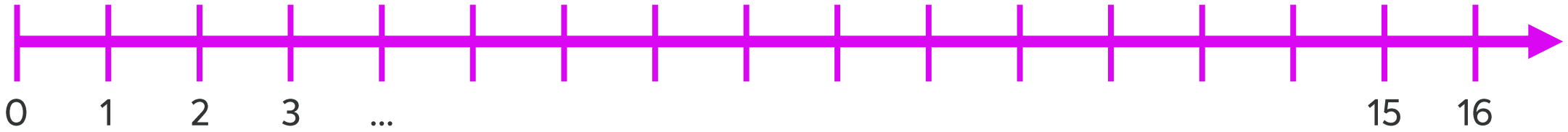
Why, Oh, Why, QuickSight?

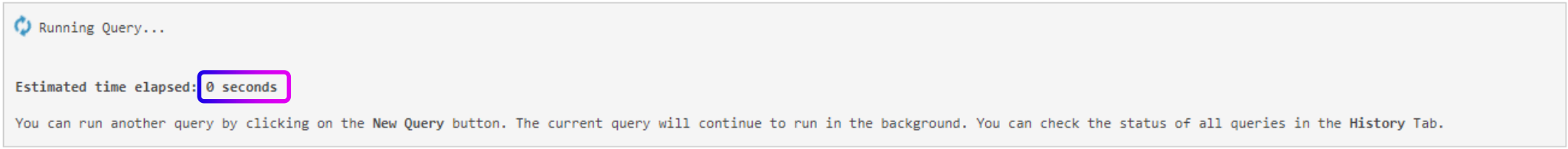


Why, Oh, Why, QuickSight?

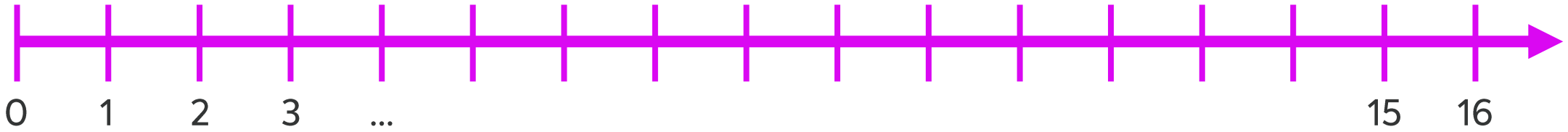


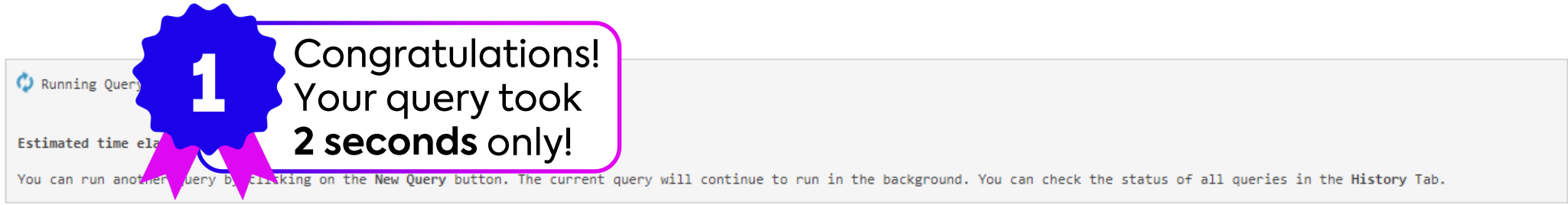
Time Works Differently in AWS Dashboards



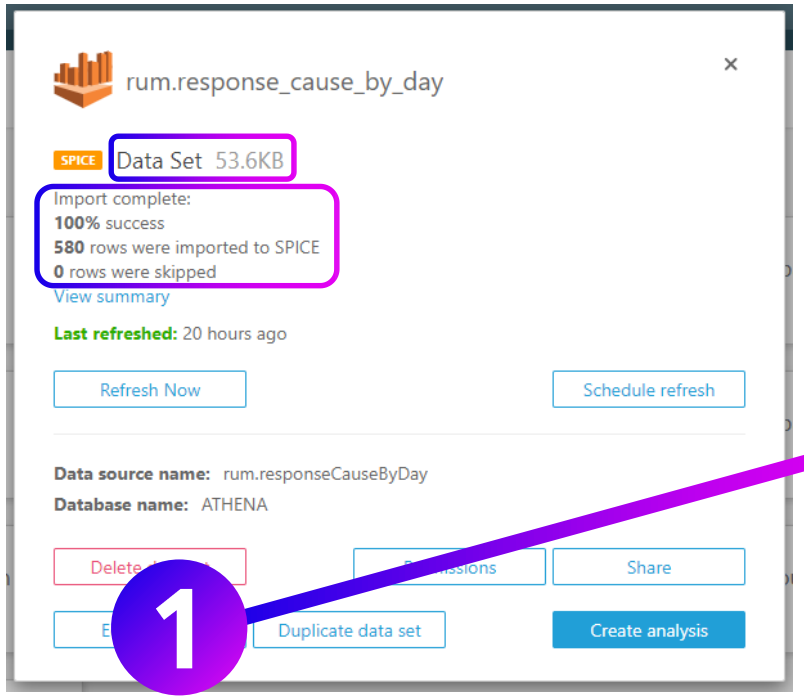


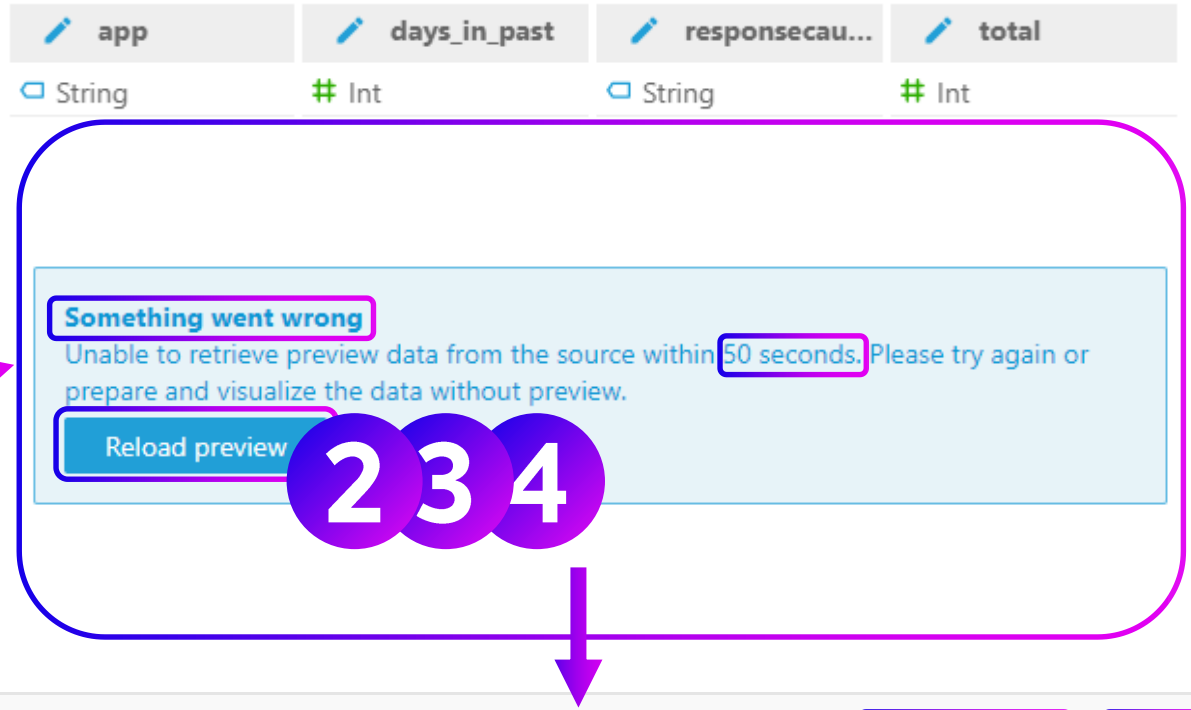
Time Works Differently in AWS Dashboards





Click Once, Pay Twice





Note: You haven't updated the data set, yet.

```
/* QuickSight c26063a5-b35a-4682-a6d8-e5e7f0a1ea1f */ SELECT "app", "days_in_past", "responsecause",...
```

N/A | Succeeded | 55.97 | 42.82 GB

But There is More!

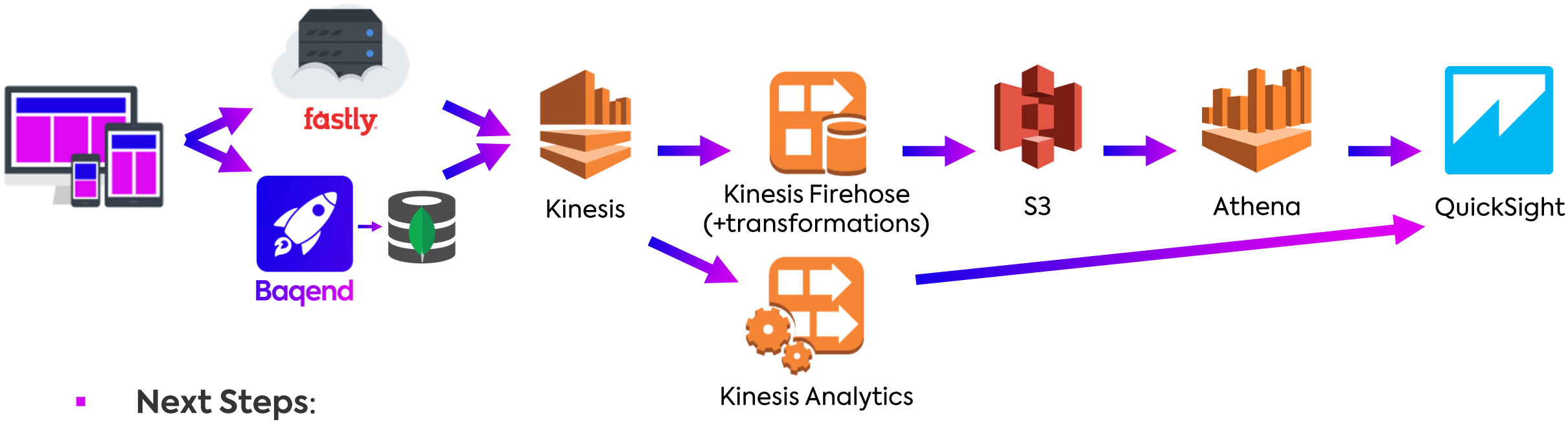
Questionable Limits
Diagrams in the dashboard, query timeouts, etc.

Exhausted Resources in Athena
Queries fail depending on time, day, and weather

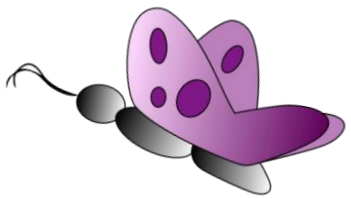
Implicit Query Rewriting in QuickSight
Hard to predict, hard to circumvent, hardly helpful


Outlook

Real-Time Analytics & Reporting

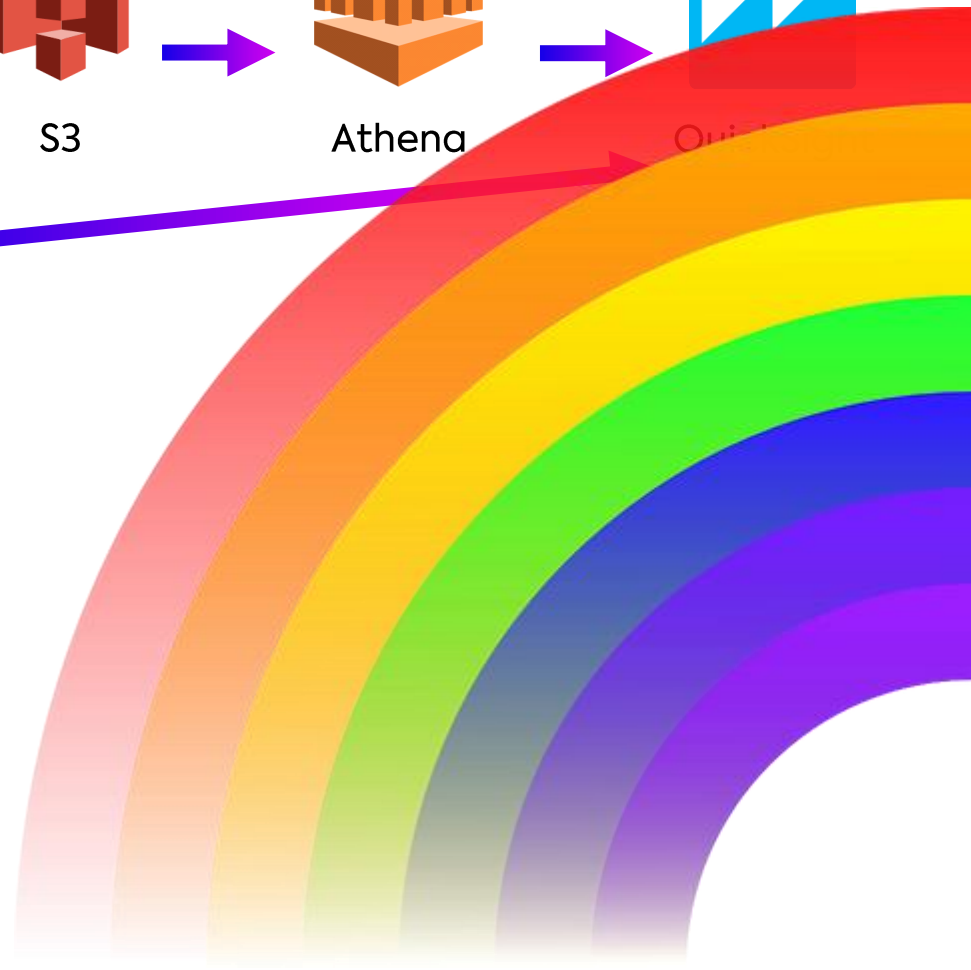


- **Next Steps**:
  - Continuous ingestion
  - Streaming analytics & real-time dashboards
  - Drop QuickSight



Real-Time Analytics & Reporting

- **Next Steps**:
  - Continuous ingestion
  - Streaming analytics & real-time rds
  - Drop QuickSight



Thanks! Any Questions?

Kino 3
Thursday, 16:00
GDPR Panel Discussion

Kino 7
Friday, 11:00
Web Performance Talk

Kino 7
Friday, 16:00
Service Worker Hackathon

Wolfram Wingerath, Data Engineering ww@baqend.com

Baqend

Thanks! Any Questions?

Right Here Right Now	GDPR Panel Discussion
Kino 7 Friday, 11:00	Web Performance Talk
Kino 7 Friday, 16:00	Service Worker Hackathon

Wolfram Wingerath, Data Engineering ww@baqend.com

Baqend