Introducing the Bidder-as-a-Service

Applying Design To Solve Scaling Problems and Evolve an Architecture

DataEngConf, NYC Oct. 30, 2017

Mark Weiss
Senior Software Engineer
mark@beeswax.com
@marksweiss

What is Beeswax?

We Built a Better Bidder

About Beeswax

- Beeswax is a 3-year-old ad tech startup based in NYC
- Founded by three ex-Googlers, CEO has deep roots in ad tech
- 40 employees in NYC and London

Why we are Different

- Customers get the benefits of a custom bidder stack, without the development and operating cost and risk
- Give customers access to all of their data
- Provide APIs for customers to customize bidding strategy, API-driven
- SaaS model and pricing, customers pay to use the platform

BEESWAX

RTB: Real Time Bidding (AKA "Please Let Us Do This")

Publisher

Step 1:
Send ad request & userid

Ad Exchange

Step 2:
Broadcast bid request

Beeswax Bidder

Scale: 1M QPS
Latency_99 : 20 ms
- Target campaigns
- Target user profiles
- Optimize for ROI
- Customize

< 200 ms

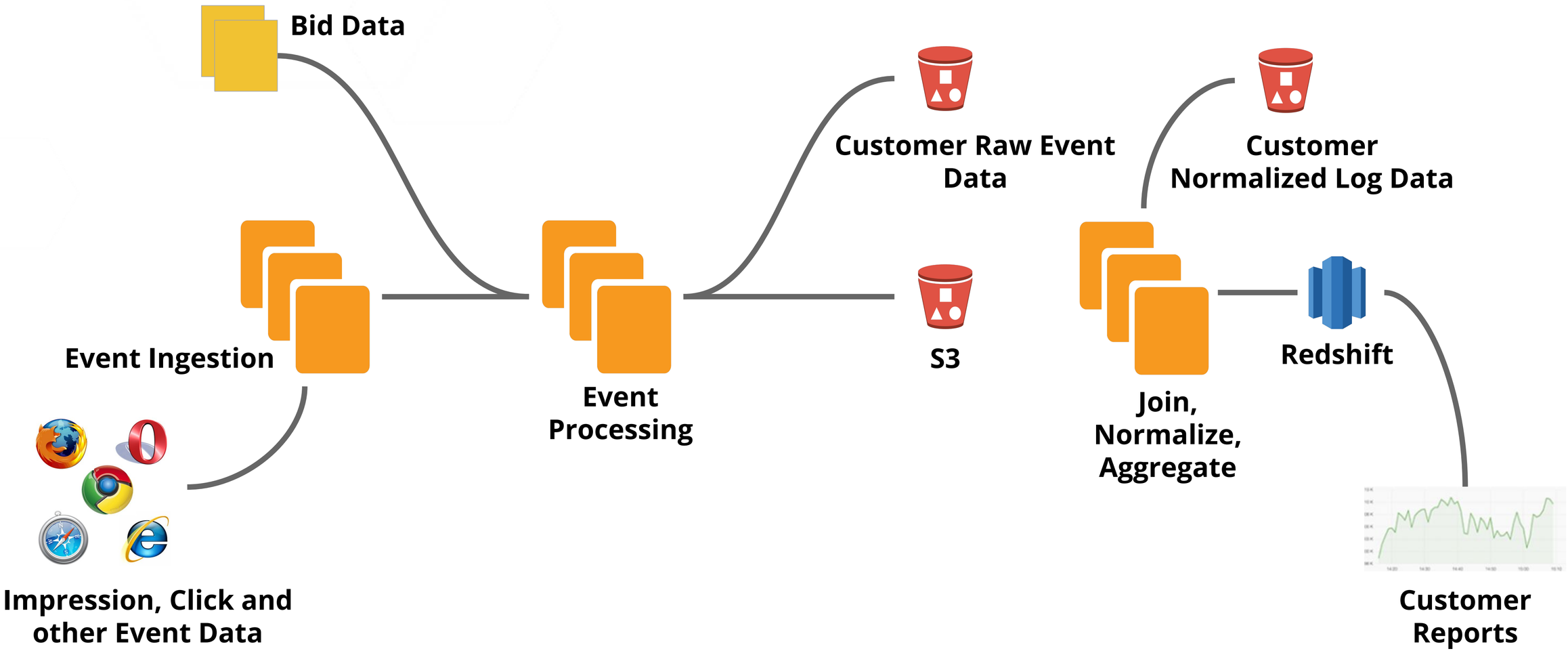
Step 3:
Submit bid & ad markup

Auction

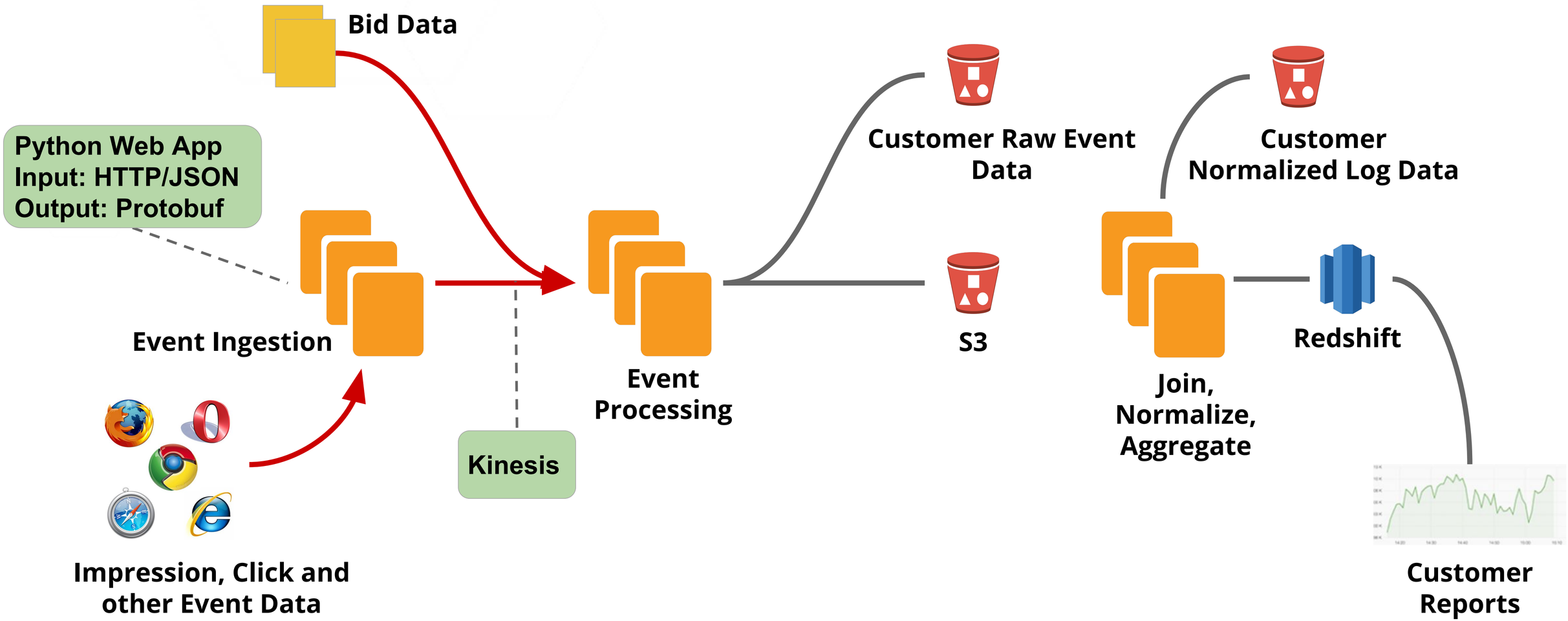
Step 4:
Show ad to user

What is the Beeswax Data Platform?

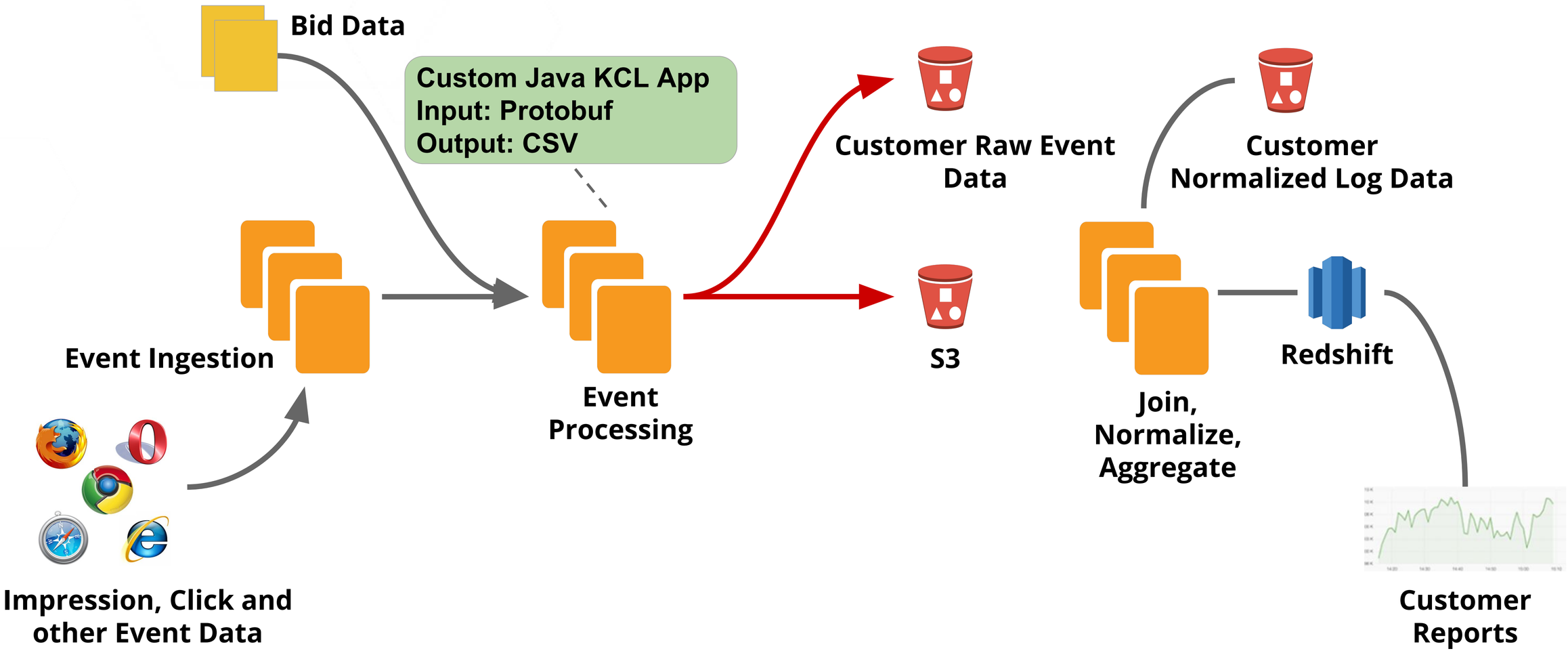
Beeswax Data Platform



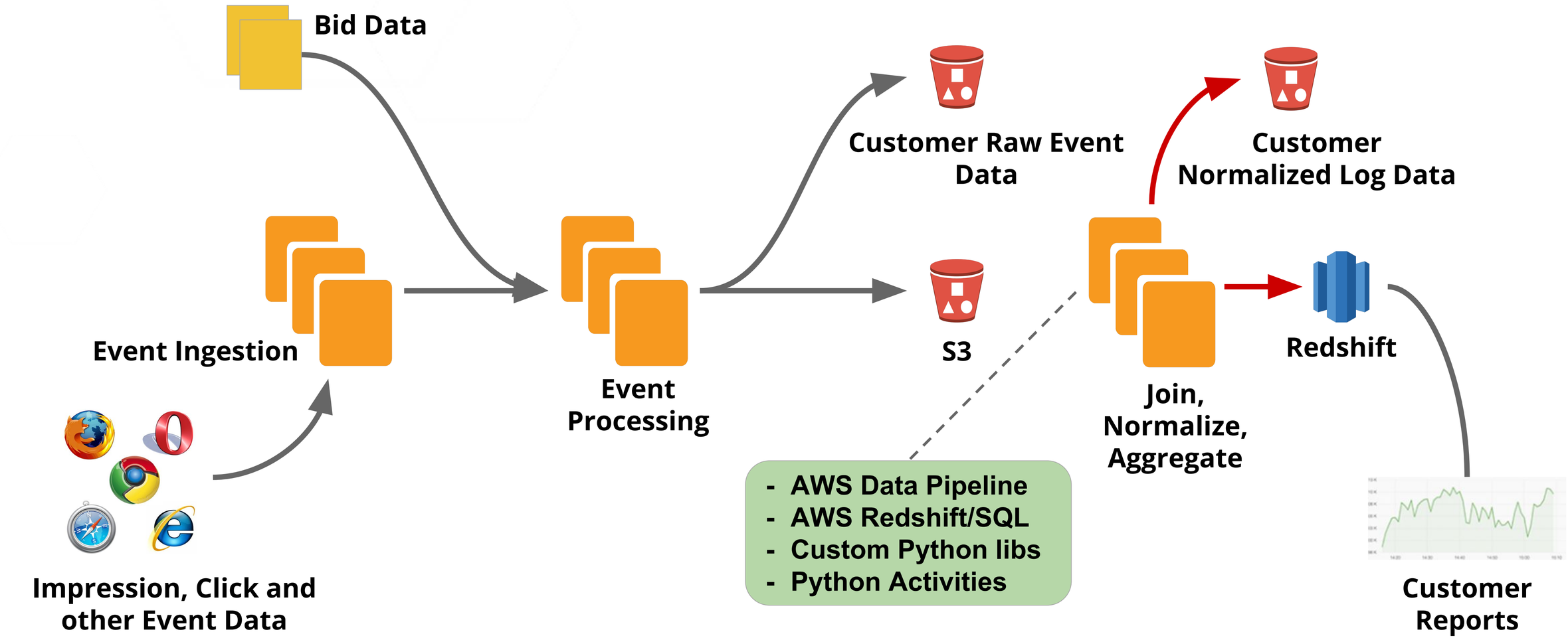
Beeswax Data Platform: Event Stream



Beeswax Data Platform: Event Processing



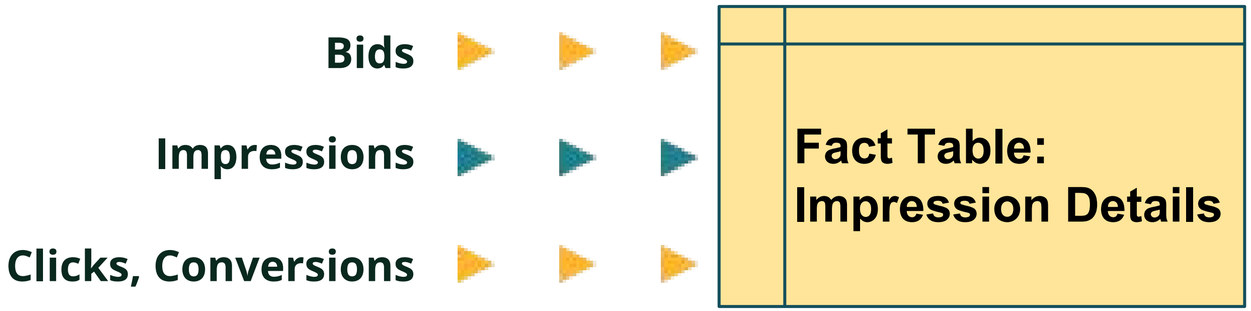
Beeswax Data Platform: Event Processing



What Was the State of the System?

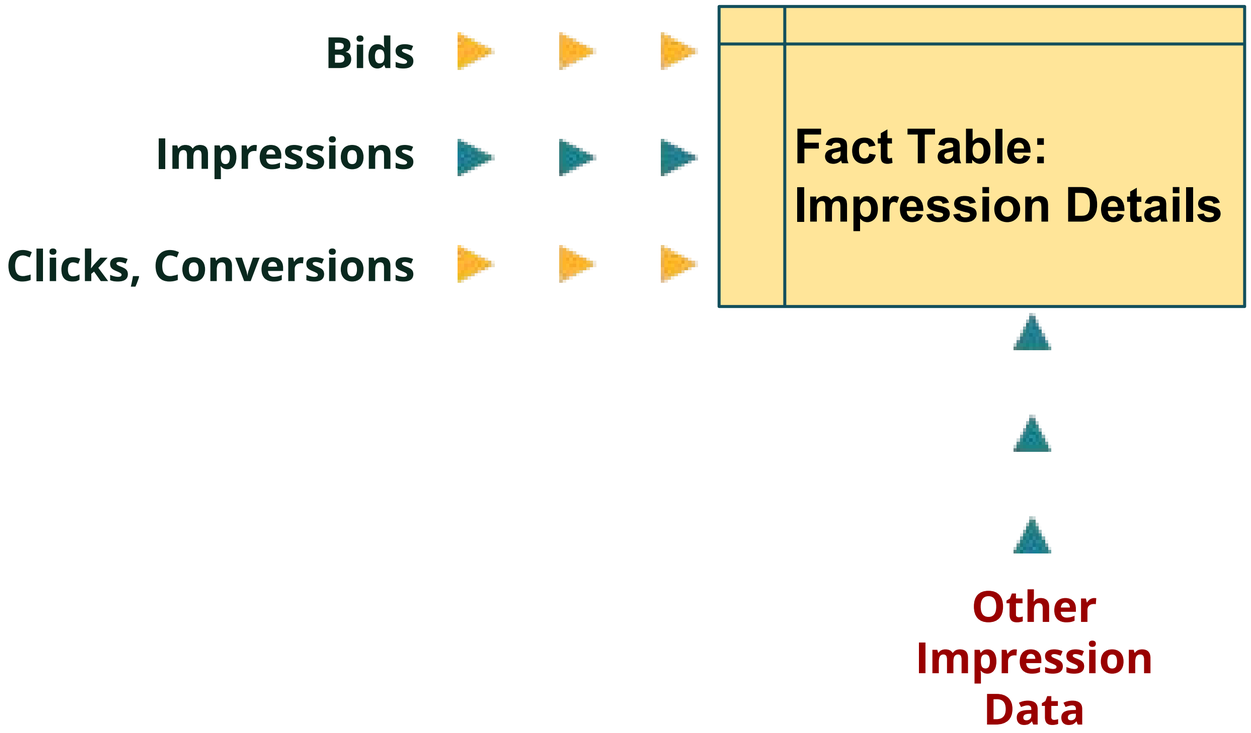
BEESWAX

Event Join and Aggregation ("Everything Looks Good ...")



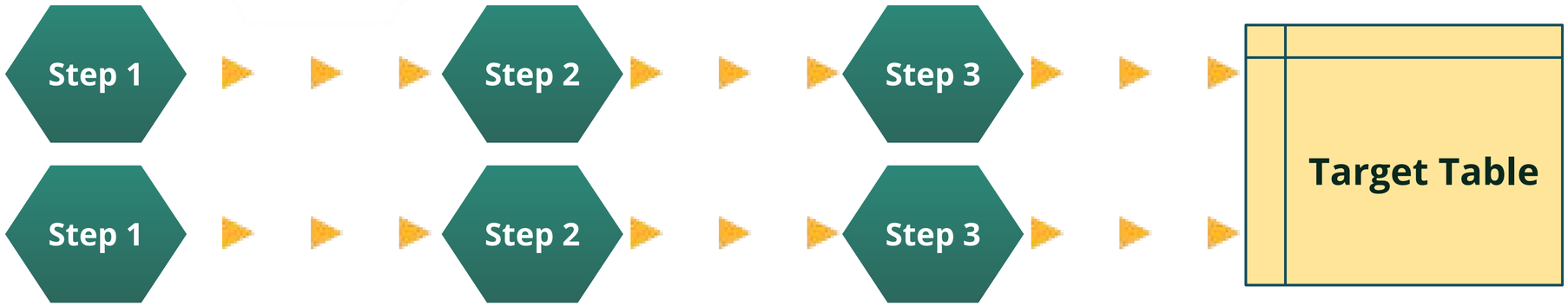
BEESWAX

Event Join and Aggregation ("Everything Looks Good ...")



BEESWAX

Pipeline Problems: Monolithic and Inflexible



BEESWAX

We were a lucky startup with a bunch of "good problems to have"

BEESWAX

System Goals for Architectural Evolution

- Support separate pipelines writing to the same target tables
- Support any pipeline depending on the data from any other
- Centralize job-level state management and job control

BEESWAX

System Goals for Architectural Evolution

- Support separate pipelines writing to the same target tables
- Support any pipeline depending on the data from any other
- Centralize job-level state management and job control
- Continue to use the existing platform technologies … for now

BEESWAX

From Goals to Principles to Patterns to Design

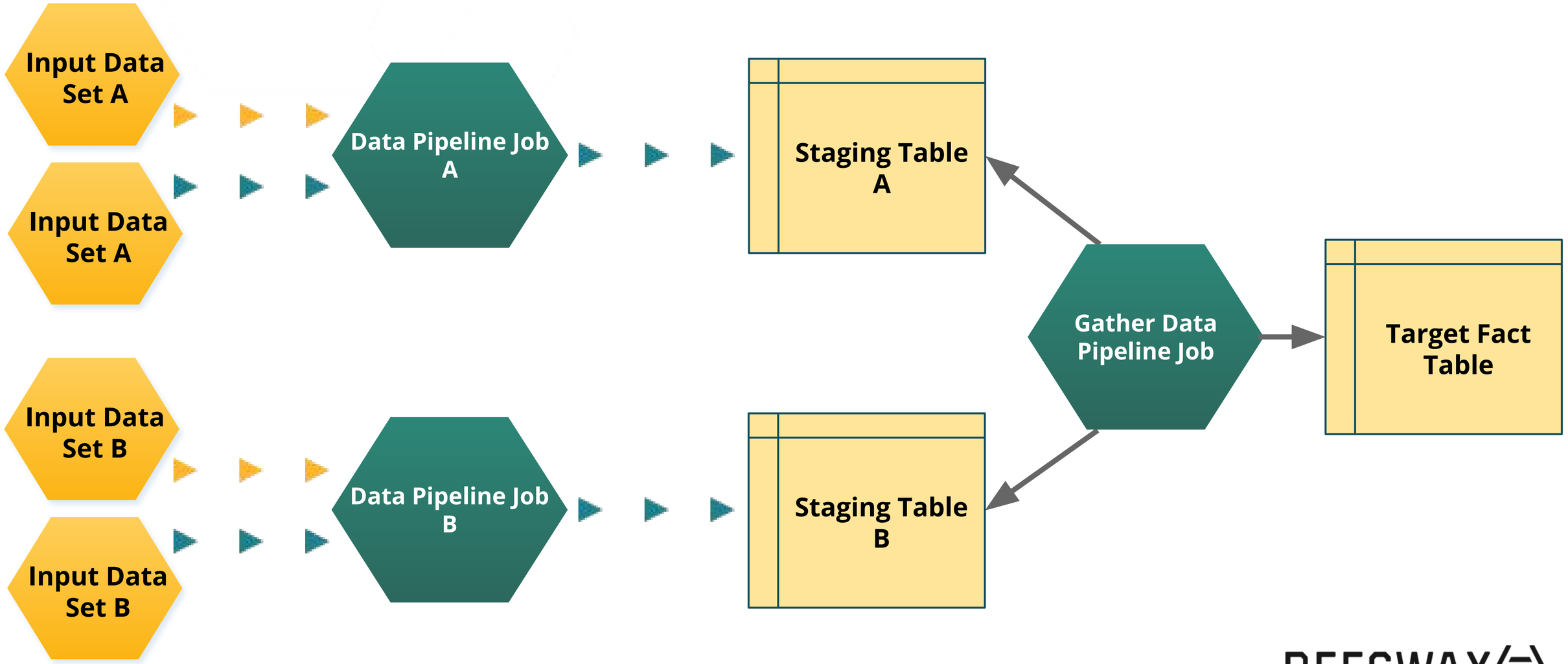
BEESWAX

Goals to Principles: Remove Contention

Goal	Principle
Multiple asynchronous pipelines with no write contention	Jobs always write to new versioned instances of target tables
Multiple pipelines land data in same master fact table	One job per master target table reads from multiple sources and writes into the target table sequentially

BEESWAX

Principles to Patterns: Remove Contention



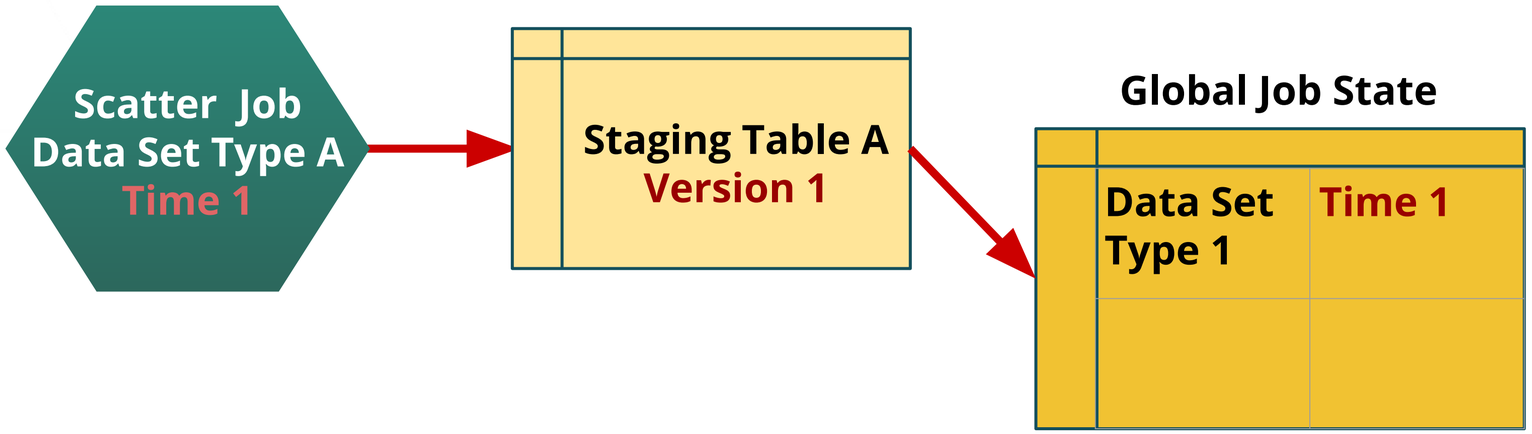
BEESWAX

Goals to Principles: Job Composition and Job State

Goal	Principle
Any job can depend on any other job	Jobs record completion of uniquely identifiable, timestamped data sets into one source of truth for all jobs
Jobs always consume the most recent source data available	Jobs can query one source of truth to discover the the most recent data sets available upon which they depend

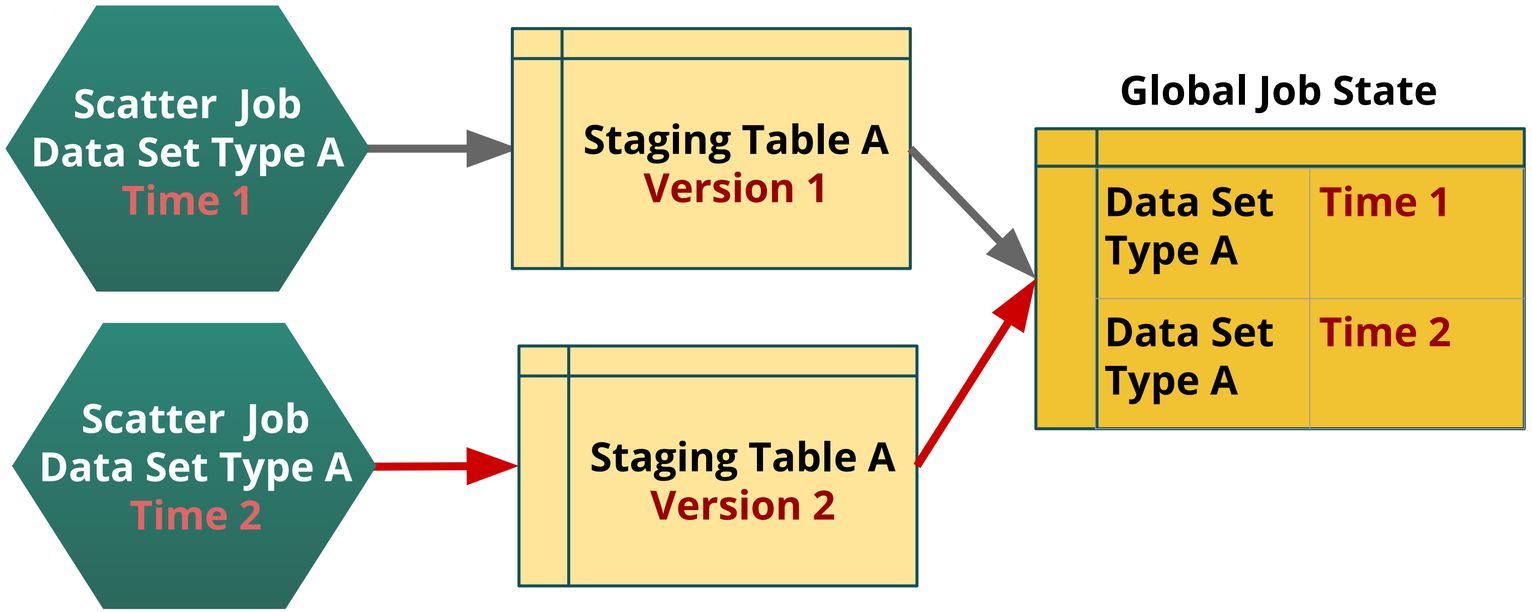
BEESWAX

Principles to Patterns: Job Composition and Job State



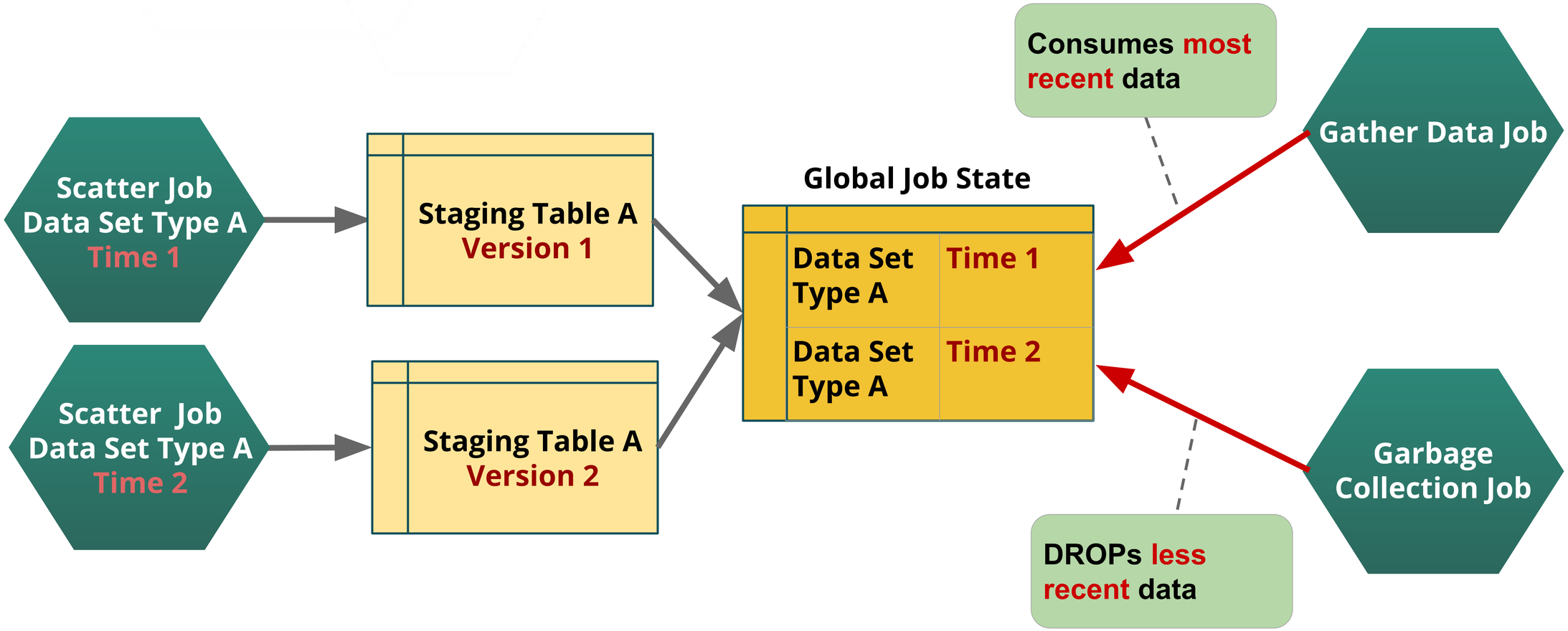
BEESWAX

Principles to Patterns: Job Composition and Job State



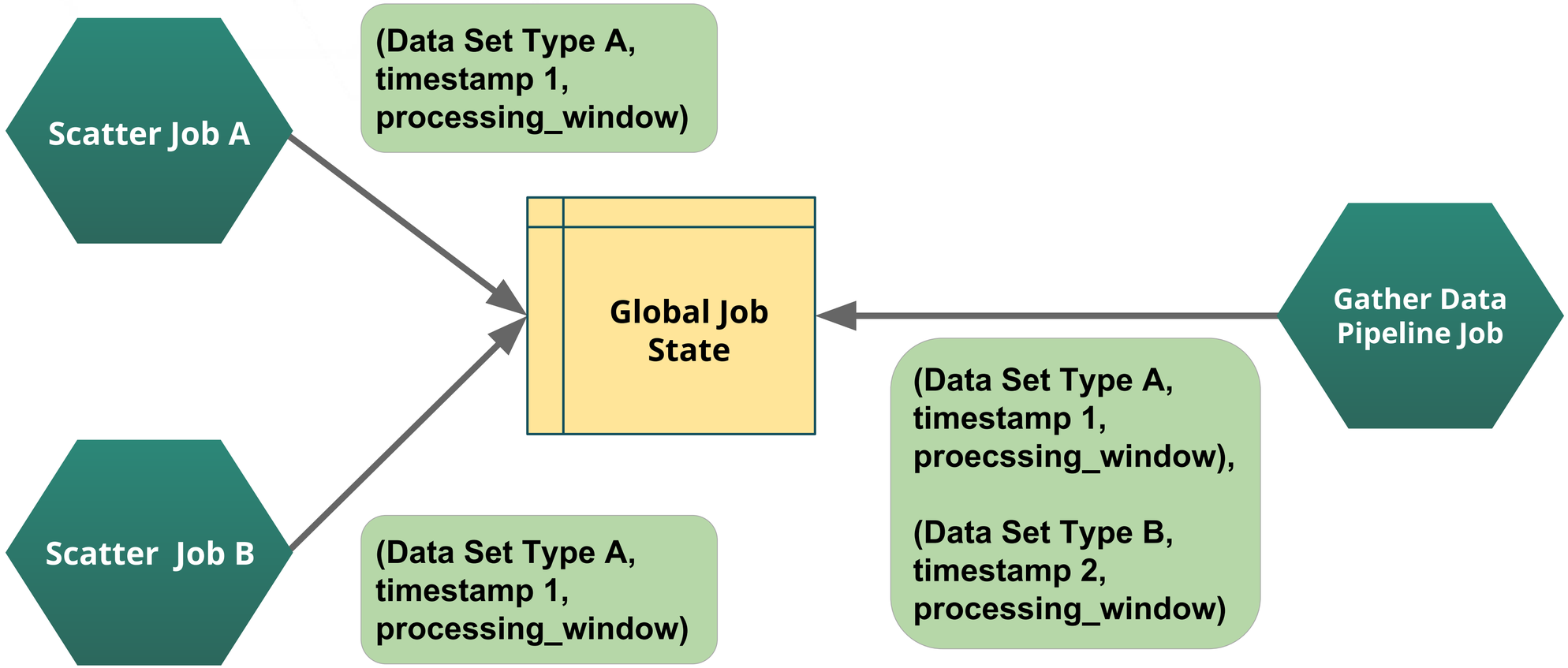
BEESWAX

Principles to Patterns: Job Composition and Job State



BEESWAX

Patterns to Design: Job Composition and Job State

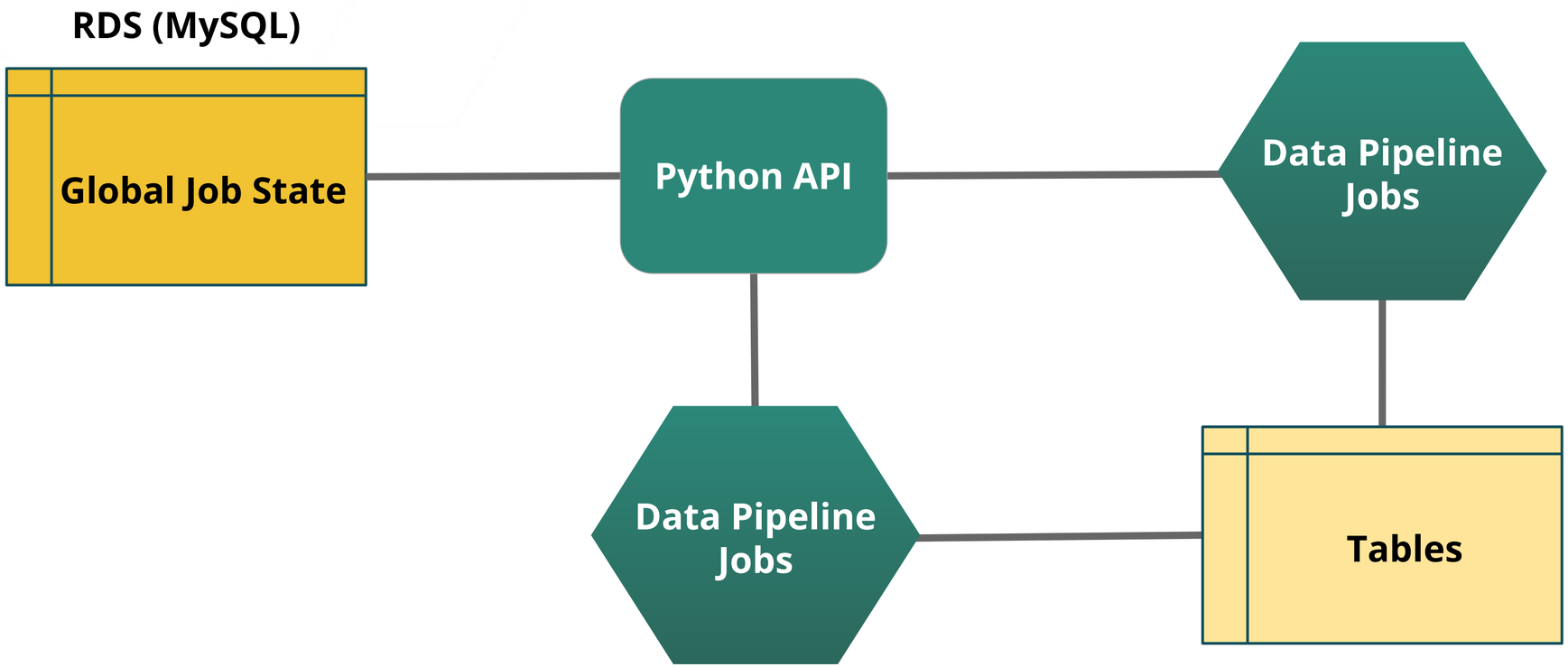


BEESWAX

Implementing the Design with What we Have on Hand

BEESWAX

Implementing the Design



BEESWAX

Conclusions

- You can evolve data architecture without adopting new technology
- Carefully chosen invariants define a design that can solve present problems and supports future flexibility
- Invariants are system Goals
- Identifying goals suggest Principles
- Patterns embody Principles
- Design applies patterns

BEESWAX

BEESWAX

Introducing the Bidder-as-a-Service

Questions?

Mark Weiss
Senior Software Engineer
mark@beeswax.com
@marksweiss

We have a great team!
We have lots of fun problems to solve!
We have LaCroix and Kind Bars!
We're hiring!
https://www.beeswax.com/careers/