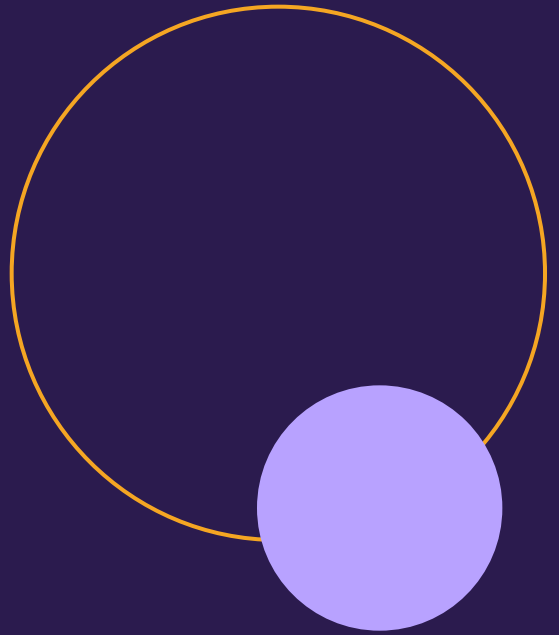
OSA CON 2026 · NOVEMBER 2 · ROOM 1

# Serverless Eventing

## Simplifying Message Delivery



**Jason (Jay) Smith**
“Cloud Alchemist” · Staff Customer Engineer, Google Cloud

ABOUT THE SPEAKER

# Jason “Jay” Smith, the Cloud Alchemist

I help people “transmute” legacy apps into something cloud native.

**DAY JOB**

Staff Customer Engineer - Application Modernization @ Google Cloud

**COMMUNITY**

CNCF Ambassador, co-lead of Cloud Native San Francisco, organizer of KCD San Francisco Bay Area

FIRST, THE DEFINITION

# What makes something serverless

01

**Zero infra**

No servers for the developer to manage.

02

**Auto-scaling**

The platform scales with demand.

03

**Pay per use**

You pay only for what you use.

04

**Event-driven**

Code runs when something happens.

The last one is the one this talk is about.

# Something has to wake up the serverless app.

Once it scales down, it isn’t just listening.

WHY I STARTED WRITING

# It started with a happy accident

**Kafka**

I was working with it a lot.

**Serverless**

I was learning it at the same time.



THE HAPPY ACCIDENT

**Knative Eventing**

I found it by chance while searching.

MY CONCLUSION

# To be event-driven, serverless needs serverless delivery and consumption.

**Delivery**

No brokers to provision, size or babysit.

+

**Consumption**

Consumers scale with events, down to zero.

THE IDEA

# An eventing plane abstracts the data from the application.

HOW IT WORKS

# How events flow



THE ECOSYSTEM

# The question changed

2020

**“We want to be data-driven.”**

People read dashboards and reports, then decide.

TODAY

**“We want to be agent-driven”**

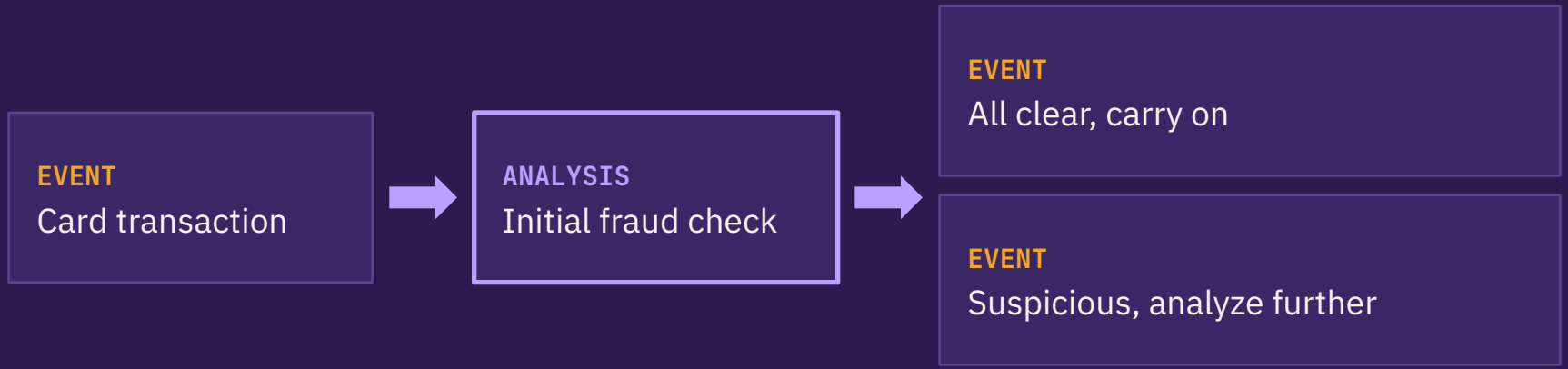
Software reads the data and acts on it.

What data needs now:

**Fresher** **More consumers** **Two-way**

MY 2020 POST

# An agent workflow, before agents




Today the analysis step can be an agent. The pattern is the same.

NOW, AGENTS

# Agents are moving onto serverless

## Cloud platforms

**Vercel**
AI Cloud, Fluid compute

**Cloudflare**
Agents SDK

**AWS**
Bedrock AgentCore Runtime

**Google Cloud**
Cloud Run with ADK

## Serverless compute

**Modal**
Compute and sandboxes

**Cerebrium**
Serverless GPU inference

**RunPod**
Serverless GPU workers

**Beam**
GPUs and sandboxes

## Agent sandboxes

**E2B**
Secure sandboxes for agents

**Daytona**
Sandboxes for agent code

**Blaxel**
Agent-native sandboxes

**Together AI**
Code sandboxes

Different layers, same contract: pay for use, scale with demand.

A sample, not a complete list.

SINCE I STOPPED WRITING

# The stack changed

| Topic | Then | Now |
| --- | --- | --- |
| **The bus** | NATS Streaming | JetStream replaced it; support ended June 2023 |
| **Knative** | Incubating CNCF project | Graduated CNCF project (October 2025) |
| **Consumers** | Tweet analysis, data apps | AI agents |
| **Constant** | Events must be routed and scaled | Still true; the bus is the part that moves |

WHAT ELSE CHANGED

# Data went serverless too

## Serverless Postgres

**Neon**
Scale-to-zero, branching

**Aurora Serverless**
Auto-scaling, can pause

**Prisma Postgres**
Usage-based, pooled

**Databricks Lakebase**
Autoscaling, scale-to-zero

**Nile**
Built for multi-tenant SaaS

**Xata**
Branching, serverless access

## Event streams

**Redpanda Serverless**
Kafka API, scale-to-zero

**Cloudflare K2**
Serverless event streams, public beta

Databases and event streams now scale with demand, many to zero.
The wiring between them is still manual.

THE STARTING POINT

# Agents are going serverless. Their data pipelines are not.

THE PROBLEM

# Where the manual work lives

## Provision

Stand up and size a message bus for every source.

## Scale

Tune consumers and partitions as traffic shifts.

## Configure

Wire each data source to each application by hand.

The agent scales to zero. The plumbing around it does not.

THE ENVELOPE

# A common event format

The application reads one format, whatever produced the event.

The plane routes on the attributes. The payload stays in data.

```
{
  "specversion": "1.0",
  "type": "com.example.order.created",
  "source": "/shop/orders",
  "id": "[unique-event-id]",
  "data": { "orderId": "[id]" }
}
```

AGENTS

# Agents on the plane

## Context in

An agent subscribes to the event types it cares about. Fresh data reaches it as events.

## Actions out

An agent publishes what it decides as events. Other services react to them.

WHAT HAS TO CHANGE

# Events agents can read

## Agents work in text, not just fields.

CloudEvents already lets an event carry any content type. Marking the body as Markdown is a convention, not a change to the spec.

The attributes stay structured, so the plane can still route. The body is text an agent can read as it is.

ATTRIBUTES · FOR ROUTING

```
type: com.example.order.delayed
source: /shop/orders
datacontenttype: text/markdown
```

BODY · FOR THE AGENT

```
## Order 4417 is late
Carrier reports a two-day delay.
Customer is on the premium plan.
```

DESIGNED FOR AGENTS

# A plane for agents and subagents




### Fan out

One task event reaches many subagents. Each scales up from zero.

### Correlate

Every result carries the parent task ID, so the parent can gather them.

### Set limits

Budgets and rate limits stop a runaway agent from flooding the bus.

### Trace

One trace follows a task from the parent through every subagent.

These are design goals for the plane, not a list of what exists today.

OPEN SOURCE

# Building blocks you can use today

| Project | Role in the plane |
| --- | --- |
| **CloudEvents** | Common format for describing events |
| **Knative Eventing** | Brokers, triggers and routing on Kubernetes |
| **KEDA** | Event-driven autoscaling for workloads |
| **Kafka, NATS** | Example message systems underneath the plane |

WRAP-UP

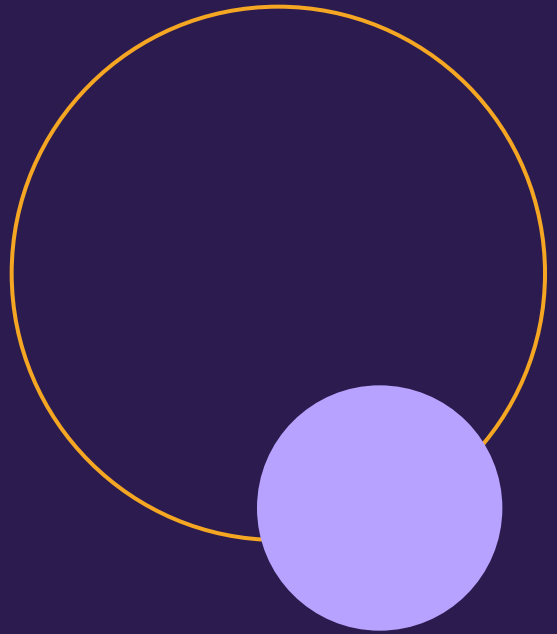
# Three takeaways

1. Treat events as a platform service, not application plumbing.
2. Standardize on one event format across every source.
3. Let event volume drive scaling, all the way down to zero.

QUESTIONS

# Thank you

Let’s talk eventing.



**Jason (Jay) Smith · Cloud Alchemist**
linkedin.com/in/jasonsmithio
https://serverlesseventing.com
https://thecloudisserverless.com