Per diem

The hard parts of white-label ordering on AWS

Technical lessons from Per Diem

Doron Segal
CTO & Co-founder, Per Diem

AWS Conference

About Doron

Doron Segal

CTO & Co-founder, Per Diem

I’m Doron Segal, CTO and co-founder of Per Diem, where I lead the engineering behind our white-label ordering platform.

I’m also a father of three, so concurrency, retries, and unpredictable inputs are part of life both at work and at home.

The restaurant technology spectrum

Restaurant technology spans proprietary enterprise apps, branded platforms for growing operators, and consumer marketplaces.

Enterprise-owned apps

McDonald's
Starbucks
Chick-fil-A

Fund proprietary apps with internal product and engineering teams.

More restaurant ownership

PER DIEM

Independent and regional brands

Subscribe to branded mobile and web ordering. The restaurant keeps its direct customer relationship.

Consumer marketplaces

DoorDash
Uber Eats
Wolt

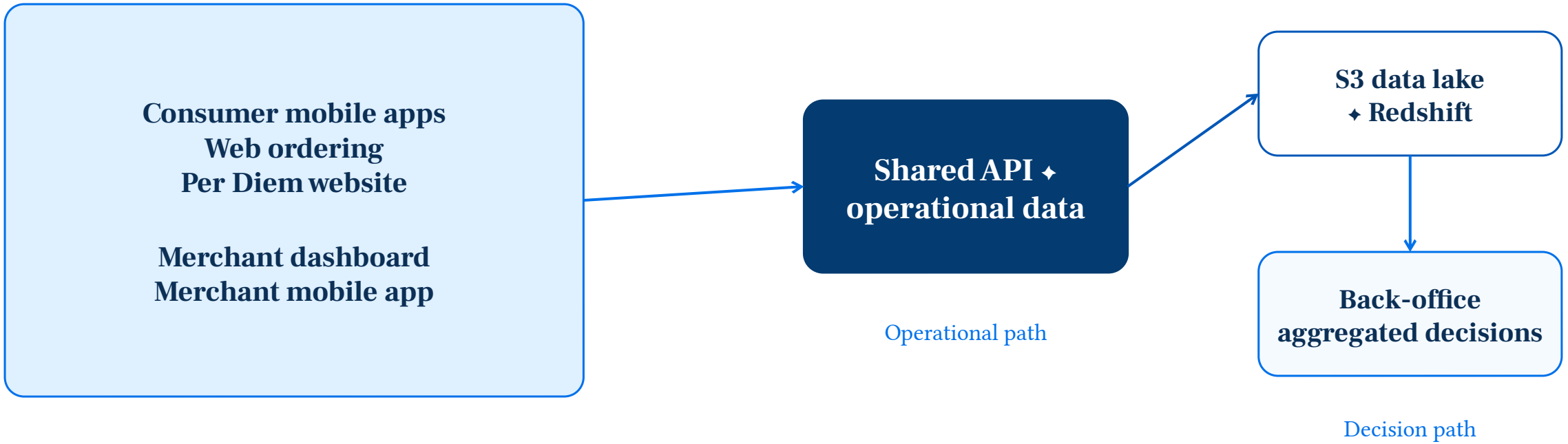
Aggregate consumer demand and delivery inside a shared marketplace.

More marketplace aggregation

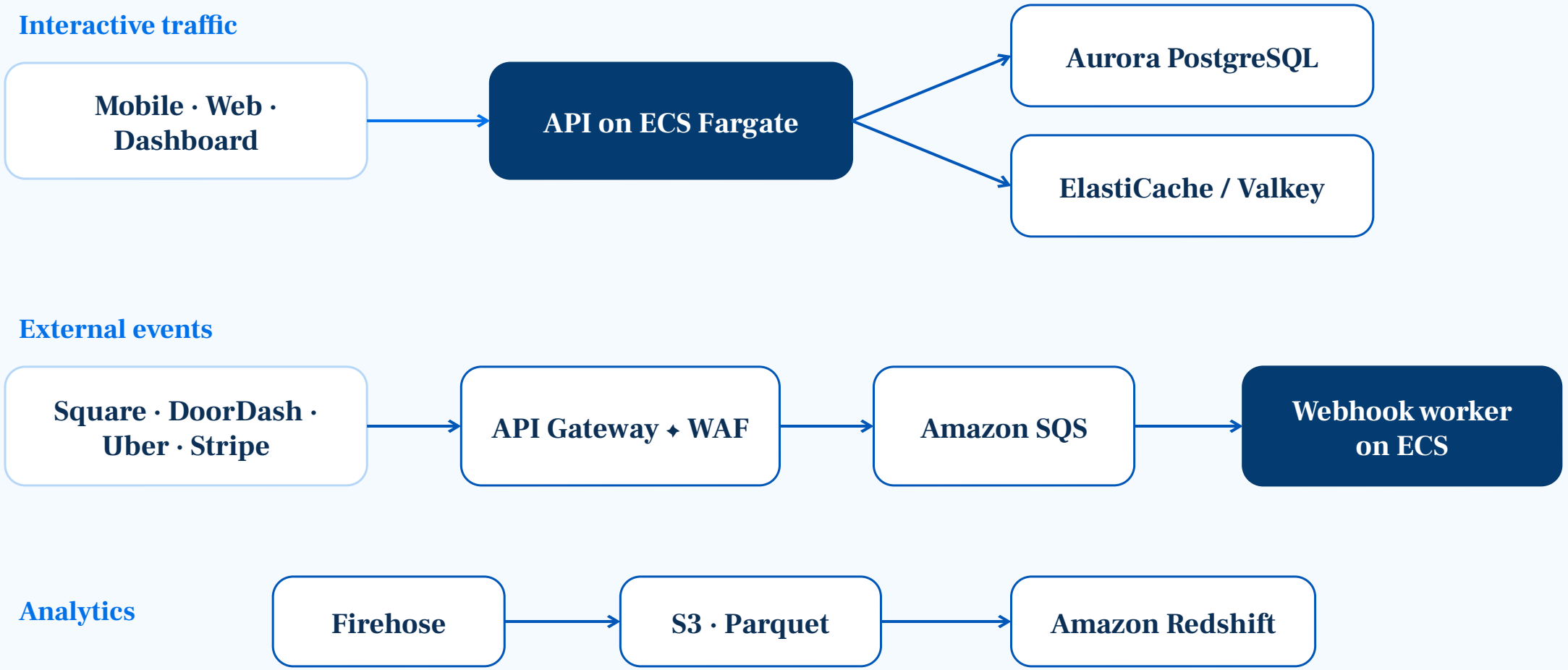
Uber Direct supports delivery for orders placed through a restaurant's own app or website.

One platform behaves like many products

Customer and merchant experiences share the platform. Analytics follows a separate path for back-office decisions.



Per Diem technical stack



Tenant isolation is also the scaling strategy

> **The tenant key carries authorization, index locality, cache ownership, and a future shard route.**

Unsafe shape

```
SELECT … FROM orders
ORDER BY created_at DESC
LIMIT 50
```

Cross-tenant exposure
Full-table pressure

Per Diem shape

```
WHERE store_id = ?
AND created_at < ?
ORDER BY created_at DESC
LIMIT 50
```

Composite index
(store_id, created_at DESC)

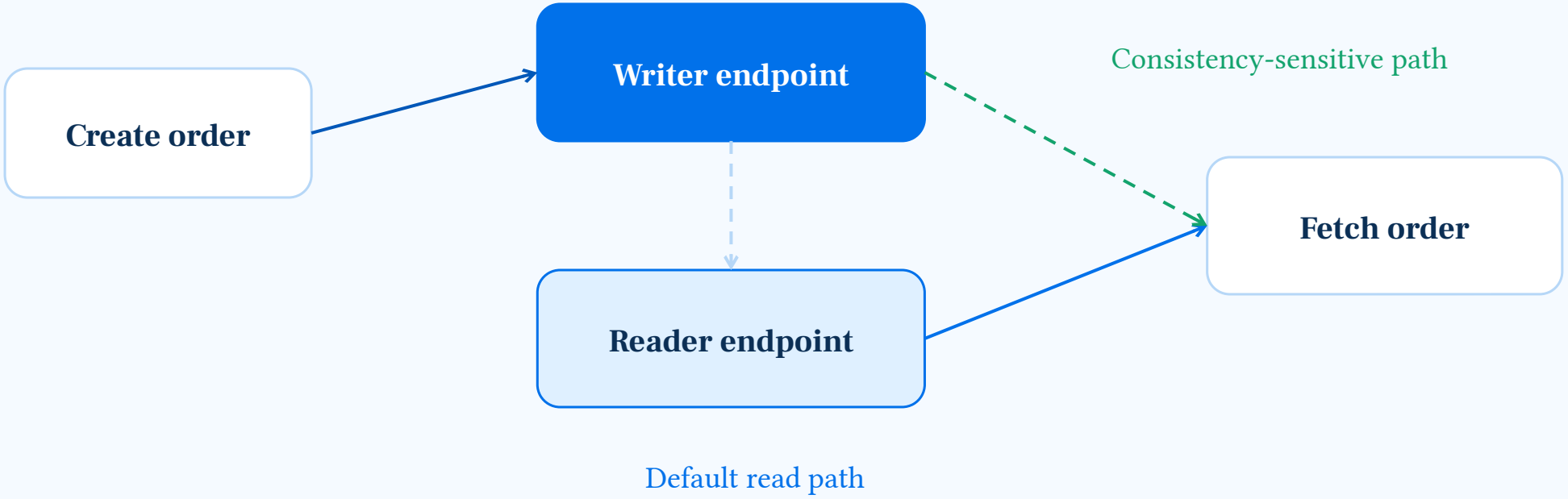
At-least-once delivery moves correctness into the handler

A queue absorbs bursts. It also makes duplicate delivery an application concern.

Design rule: acknowledge only after the durable state transition succeeds.

Read scaling creates a consistency decision

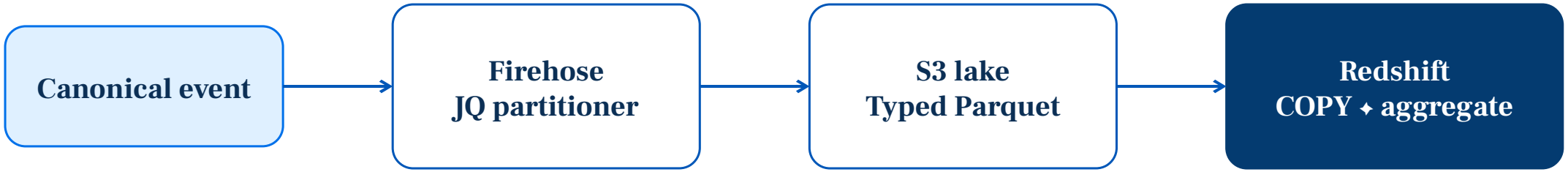
Aurora reader endpoints increase read capacity, but an immediate read can observe the previous state.



> **Per Diem routes read-after-write flows through the writer path. Most reads still use replicas.**

Transactional data and analytical data need different paths

The operational database answers one merchant's current request. Redshift answers historical questions across time.



Partition keys reduce the scan surface

Bulk COPY respects the columnar engine

> **Rule: dashboards never turn Aurora into a warehouse.**

Large tables need a lifecycle before they become large

Per Diem partitions append-only tables by created_at and treats archival as part of the design.

Why it matters

Partition pruning keeps recent queries bounded. New partitions avoid the expensive conversion of a live table. A dry-run archive path protects stateful data.

The API serves clients that cannot deploy together

A mobile release can stay active for months after the backend changes.

Preference order Additive contract · server-side version gate · new endpoint version · forced upgrade only as a last resort

Infrastructure as code still needs a blast-radius gate

CDK makes infrastructure repeatable. Stateful replacement remains stateful replacement.

Cross-stack values travel through explicit CloudFormation exports. Secrets stay in Secrets Manager. Production images pin a commit-specific ECR tag.

The remaining coupling is visible and testable

The most useful architecture review asks where two definitions can drift.

01 **Event schema**

The Glue payload struct and the analytics service's canonical event type must change together.

02 **Stack contracts**

Service stacks depend on named CloudFormation exports. Renaming an export is an API change.

03 **Runtime policy**

Local Node versions and CodeBuild runtime versions can drift unless CI validates them as one policy.

Per diem

What held up under real product constraints

1. Put the tenant key in every boundary
2. Make retries safe before adding throughput
3. Choose consistency per user flow
4. Treat schemas, clients, and stacks as versioned contracts

Per diem

Thank you

Questions or suggestions? Let me know.

doron@tryperdiem.com

tryperdiem.com