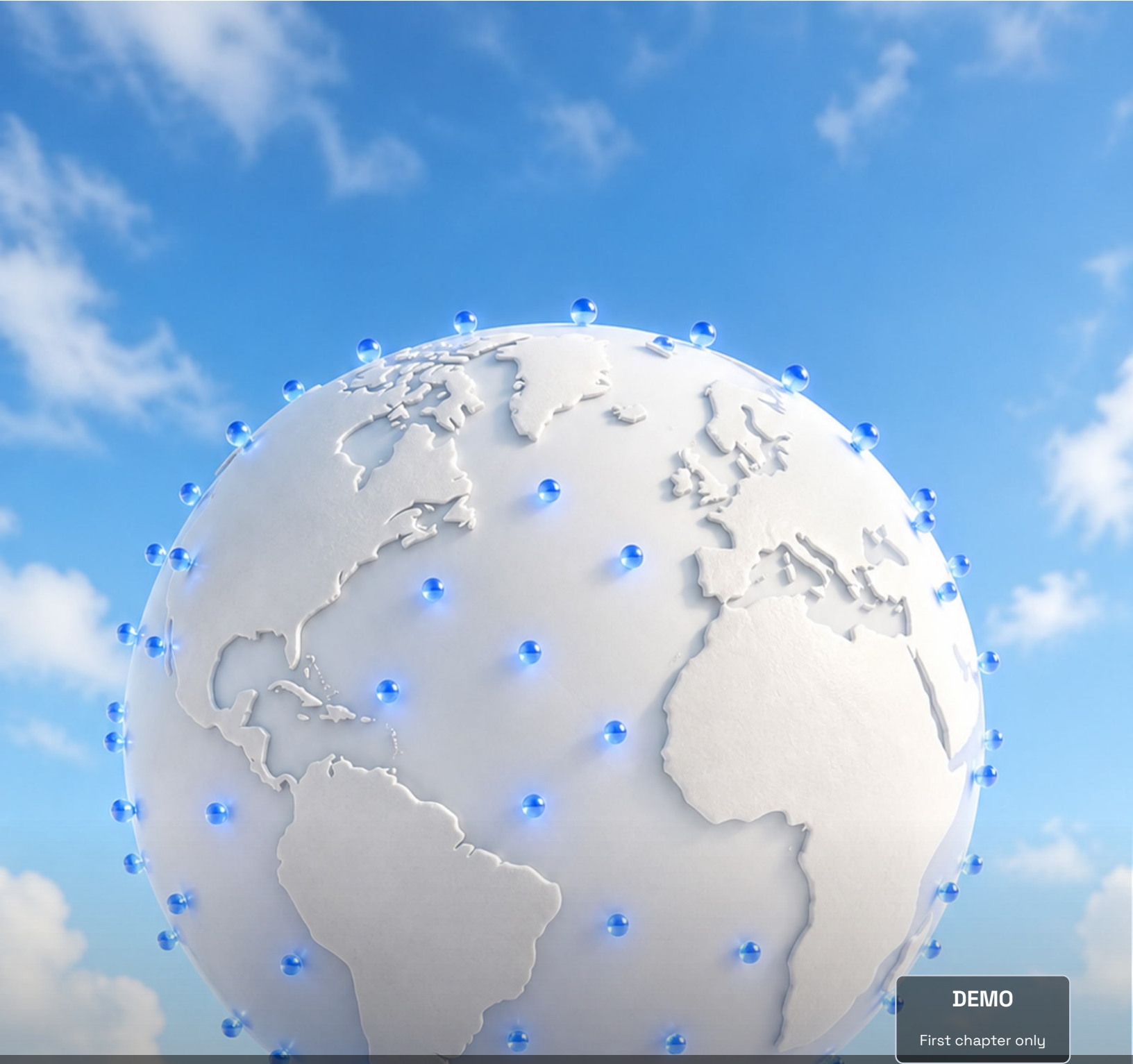

DEMO

First chapter only

Edge-First Architecture

Building Commerce on Cloudflare Workers

pragma.vision

SOFT.HOUSE

Edge-First Architecture

Trademark Notice

Google, Google Pay, Google Cloud, and Android are trademarks of Google LLC. Stripe is a trademark of Stripe, Inc. Cloudflare and Cloudflare Workers are trademarks of Cloudflare, Inc. Supabase is a trademark of Supabase, Inc. OpenAI and ChatGPT are trademarks of OpenAI, Inc. Claude is a trademark of Anthropic, PBC. W3C is a trademark of the World Wide Web Consortium. Visa is a trademark of Visa International Service Association. OWASP is a trademark of the OWASP Foundation. Midjourney is a trademark of Midjourney, Inc. Canva is a trademark of Canva Pty Ltd. Etsy is a trademark of Etsy, Inc. Amazon is a trademark of Amazon.com, Inc. All other trademarks are the property of their respective owners.

No Affiliation

This book is an independent publication. It is not authorized, sponsored, or endorsed by any of the companies or organizations whose products or services are mentioned herein.

No Professional Advice

The information in this book is provided for educational purposes only. It does not constitute legal, financial, investment, tax, or other professional advice. Readers should consult qualified professionals for guidance specific to their situation.

Code Examples

Code examples in this book are provided for illustration only. They may not be suitable for production use without additional validation, error handling, and security review.

Published by Pragma Vision LLC

First edition, 2026.

Contents

1

Why Edge-First for Commerce

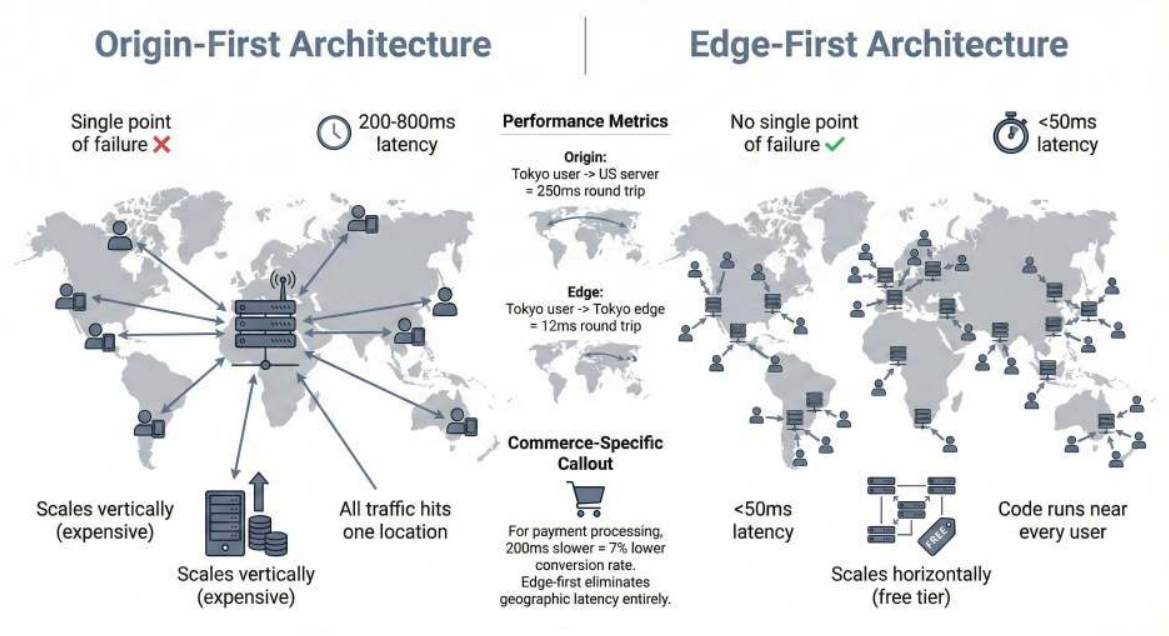

Figure 1. Origin-first versus edge-first: routing all traffic to one region gives a single point of failure and 200–800ms latency (a Tokyo-to-US round trip of 250ms), while edge-first runs code near every user for under 50ms (12ms within Tokyo)—and for payments, every 200ms of delay costs 7% conversion

Every millisecond matters in commerce. A customer in Sydney adding an item to their cart should not wait 280 ms for a round trip to a server in Virginia. A payment confirmation in São Paulo should not depend on the health of a single data center in Frankfurt. And an idempotency check that prevents a duplicate charge should execute at the same location where the request arrived – not after a cross-continental hop.

Edge-first architecture inverts the traditional model. Instead of deploying your application to one or two cloud regions and relying on a CDN for static assets, you deploy your *entire application* to hundreds of locations worldwide. Every user, on every continent, hits compute that is physically close to them.

1.1 The Origin-First Problem

Traditional commerce backends follow an origin-first pattern. Your API server runs in `us-east-1` or `eu-west-1`. A CDN sits in front, caching static responses, but every cart mutation, every payment intent, every mandate verification must travel back to the origin. The architecture looks like this:

```
// Origin-first: all requests funnel to one region
// Sydney user -> CDN edge (cache miss) -> us-east-1
// Sao Paulo user -> CDN edge (cache miss) -> us-east-1
// London user -> CDN edge (cache miss) -> us-east-1

// Result: 50-300ms added latency for every write operation
// Result: single point of failure in one region
// Result: scaling means bigger machines, not more locations
```

For static content this is tolerable. For commerce – where every interaction mutates state – it is a fundamental bottleneck. Payment processing, cart management, session validation, rate limiting, and idempotency checks all require compute. In an origin-first world, that compute lives in one place.

1.2 The Edge-First Alternative

Edge-first deploys your application logic alongside the user. Cloudflare Workers run on V8 isolates across more than 300 cities in over 100 countries. When a request arrives, it executes in the same data center where it was received. There is no origin to route to – the edge *is* the application.

```
// Edge-first: compute at the point of presence
// Sydney user -> Sydney PoP -> execute Worker -> respond
// Sao Paulo user -> Sao Paulo PoP -> execute Worker -> respond
// London user -> London PoP -> execute Worker -> respond

// Result: sub-10ms compute latency (no network hop)
// Result: automatic failover (300+ locations)
// Result: scaling means more requests, not more machines
```

<5ms

Worker cold start time with V8 isolates – compared to 100ms-1s for container-based functions

1.3 Why Commerce Specifically Benefits

Not every application benefits equally from edge deployment. A batch data processing pipeline cares about throughput, not latency. A machine learning training job needs GPU clusters, not geographic distribution. But commerce has specific characteristics that make edge-first transformative:

- **Global customers**: Commerce is inherently international. Your buyers are everywhere.

- **Latency-sensitive writes**: Cart adds, payment captures, and mandate verifications are write operations that block the user experience.
- **Financial idempotency**: Duplicate charge prevention must execute atomically at the request boundary – not after a round trip to a distant origin.
- **Session management**: Authentication tokens, rate limits, and nonce validation are checked on every request.
- **Regulatory proximity**: Data residency requirements may demand that certain operations execute within specific geographic boundaries.

Key Insight

Edge-first is not about making static pages faster. It is about moving *compute* – the cart logic, the payment verification, the idempotency check – to where the customer is. In commerce, every interaction is a transaction, and every transaction benefits from proximity.

1.4 About Pragma.Vision

This book draws from implementation experience. The Pragma.Vision ecosystem – an AI-native commerce platform model spanning nine interconnected services – is designed to run on Cloudflare Workers. The architecture keeps payment mandates, agent verification, and marketplace transactions close to the edge.

We did not choose edge-first because it was trendy. We chose it because a commerce platform processing financial transactions across protocols (Google AP2, Stripe ACP) cannot afford the latency, reliability, or cost profile of traditional origin-first architecture. With a total infrastructure budget of $1,800 and a requirement for global availability from day one, the edge was the only viable path.

The patterns in this book are not theoretical. They are extracted from an implementation codebase designed to achieve:

- Sub-10ms response times for payment operations worldwide
- Idempotency-protected checkout that prevents duplicate charges (via KV-backed idempotency keys)
- high availability across 300+ global edge locations
- $0 infrastructure cost through free tier optimization

1.5 What You Will Learn

Each chapter covers a specific component of edge-first commerce architecture:

1. **Workers Fundamentals** – V8 isolates, the execution model, CPU limits, and bindings
2. **KV for Session and Idempotency** – Eventual consistency patterns for rate limiting and duplicate prevention
3. **D1 for Relational Data** – SQLite at the edge for mandate storage and transactional queries
4. **R2 for Asset Storage** – Object storage without egress fees for product images and documents
5. **Durable Objects for Real-Time** – Strongly consistent state for WebSocket coordination and live carts
6. **Wrangler Configuration** – The deployment toolchain that ties everything together
7. **Performance Optimization** – Cold start mitigation, caching strategies, and global tuning

By the end, you will have a complete mental model for building commerce applications that run at the edge – and the concrete code patterns to implement it.

DEMO

This is a free preview of the full edition.

Get the complete book at:

https://shop.pragma.vision