ML-Driven IP Detection at Scale: CI/CD Meets Cloud-Native Enforcement

Kanika Gupta, Amazon

The Challenge: IP Violations at Unprecedented Scale

Digital marketplaces face explosive growth in intellectual property violations. With millions of listings submitted daily, enforcement systems must make near real-time decisions that are directly seller, brand and customer impacting. The stakes couldn't be higher.

Volume Challenge

Millions of listings per day requiring immediate review and action

High-Stakes Decisions

Enforcement actions directly impact brand, sellers and customers

Precision Imperative

False positives damage sellers; false negatives enable violators

Trust & Compliance

Regulatory, legal, and brand reputation demand accuracy

Why Traditional Approaches Can't Keep Up

Legacy detection systems fundamentally break at modern scale. The limitations aren't just technical–they're architectural. Understanding these failure modes is critical to designing systems that actually work in production.

01

Manual Review Doesn't Scale

Human reviewers can't process millions of listings with consistent quality and speed

02

Rule-Based Systems Are Brittle

Static rules are easily evaded by adversaries who adapt faster than rules update

03

ML-Only Focus Ignores Operations

Focusing solely on model accuracy neglects deployment, monitoring, and incident response

04

Static Models Degrade Rapidly

Models decay under evolving adversarial behavior without continuous retraining

05

Siloed Teams Increase Risk

Separation between ML and infrastructure teams creates blind spots and incident escalation

Core Insight: Production ML Is a Systems Problem

> **Detection accuracy alone is insufficient in production**

The hard truth: even the most accurate ML model fails without operational rigor. Real-world enforcement systems must handle rollbacks, maintain visibility, and earn business trust through reliability. This isn't just an ML problem–it's a full-stack engineering challenge.

Reliability Over Raw Accuracy

Consistent performance builds stakeholder confidence

Observability Enables Trust

Visibility into decisions and drift detection

System-Level Thinking

Enforcement requires orchestration, not just models

CI/CD as First-Class Requirement

Rapid, safe iteration is non-negotiable

High-Level System Architecture

Our architecture separates concerns while maintaining end-to-end traceability. Real-time ingestion feeds multimodal detection, event-driven orchestration routes decisions, and continuous feedback loops enable rapid iteration. The key innovation: treating the entire pipeline as a unified, observable system.

01

Real-Time Data Ingestion

Streaming pipeline for millions of listings per day

02

Multimodal Detection

Image, text, and behavioral analysis working in concert

03

Event-Driven Orchestration

Intelligent routing based on confidence and risk

04

Human + LLM Verification

Hybrid approach for edge cases and explainability

05

Observability & Feedback

Continuous monitoring drives model improvement

06

Online/Offline Separation

Inference and retraining operate independently

Multimodal Detection Engine

Different signals capture different evasion tactics. Images can be manipulated, text can be obfuscated, but combining modalities creates resilience. Our weighted ensemble approach enables independent upgrades without full system retraining–critical for rapid response to emerging threats.

Image Embeddings

Fine-tuned YOLOv7 for image detection

Text Analysis

Domain-adapted BERT models for semantic understanding

Behavioral Signals

Seller history and historical abuse patterns

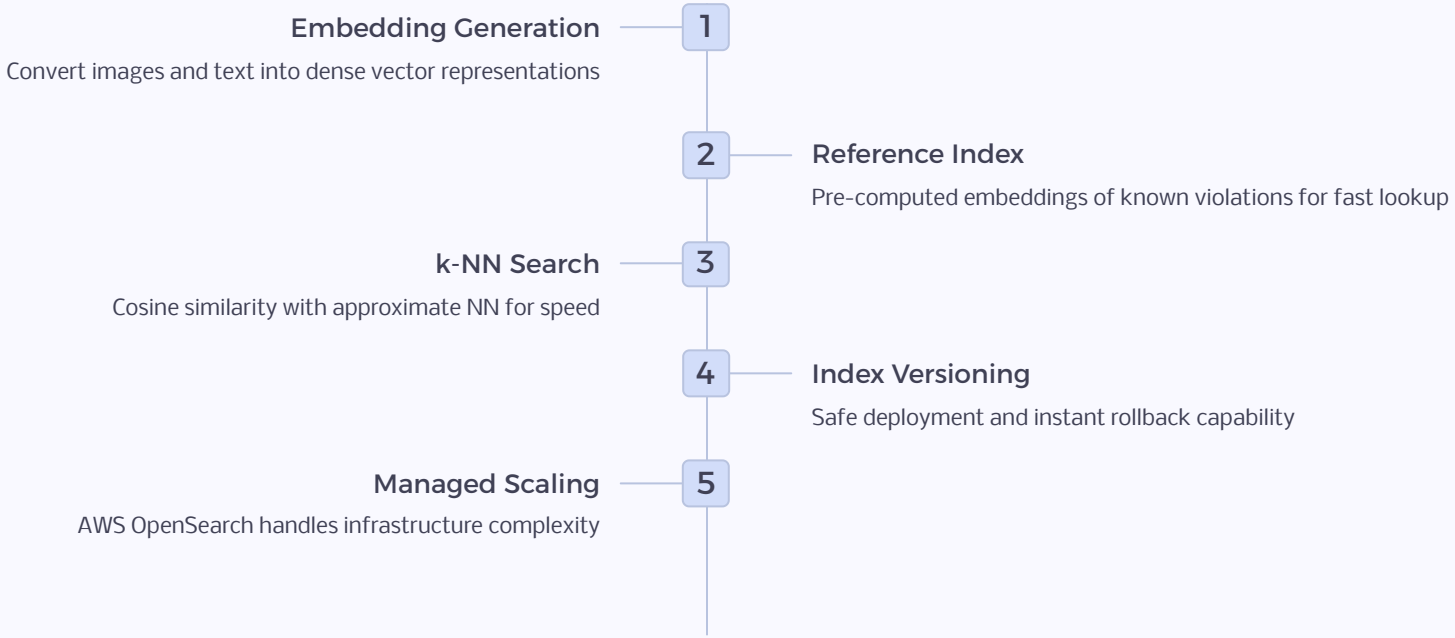
Weighted Ensemble

Flexible scoring system optimized for precision-recall tradeoffs

Key Advantage: Modality independence means we can upgrade image detection without touching text models, reducing regression risk and deployment complexity.

Vector Similarity Search at Scale

Embedding-based similarity is the engine behind our detection system. By pre-computing reference embeddings and using approximate nearest neighbor search, we balance high recall with sub-second latency. Index versioning provides the safety net: instant rollback if a new index degrades quality.



Technical Implementation

- Cosine similarity for embedding comparison
- Approximate NN trades minimal recall for major latency gains
- Immutable index versions enable A/B testing
- Horizontal scaling through managed services

Operational Benefits

- Sub-second p99 latency at millions of QPS
- Zero-downtime index updates
- Automatic failover and redundancy
- Clear rollback path reduces deployment risk

Event-Driven Orchestration

Our event-driven architecture provides the backbone for reliable, scalable processing. Immutable event streams enable replay for debugging, while confidence-based routing ensures appropriate handling for each decision type. Built-in backpressure prevents cascading failures during traffic spikes.

1

Kinesis Streams

Real-time ingestion of listing events at massive scale

2

SNS/SQS Routing

Confidence-based decision routing to appropriate handlers

3

Step Functions

Orchestrate complex workflows with built-in retry logic

4

Error Handling

Dead letter queues and exponential backoff

5

Backpressure

Rate limiting prevents downstream service overload

6

Replay Capability

Immutable streams enable incident investigation

Architecture Philosophy: Every component is designed for failure. Retries, circuit breakers, and graceful degradation are first-class design requirements.

Confidence-Based Enforcement Strategy

Not all enforcement actions carry the same risk. Account suspension requires near-certainty, while flagging for manual review tolerates ambiguity. Our threshold-based routing converts ML uncertainty into explicit policy, minimizing customer harm while maximizing coverage.

Manual Review Queue

Low to medium confidence cases reviewed by specialists

Seller Warnings

Medium confidence triggers educational interventions

Listing Removal

High confidence justifies immediate content takedown

Account Suspension

Highest confidence threshold for severe enforcement

Risk Calibration

Each enforcement level has calibrated precision requirements. Account suspension demands 99%+ precision, while review queues optimize for recall. This tiered approach balances business risk with violation coverage.

Policy Translation

ML confidence scores map directly to enforcement policies, making the system auditable and adjustable. Legal and trust teams can modify thresholds without model retraining.

LLM-Assisted Verification

Large language models provide context-aware reasoning for medium-confidence cases. Rather than replacing human judgment, LLMs augment it—reducing review burden while improving appeal transparency through structured explanations. The system considers similarity scores, seller history, and product metadata to generate defensible decisions.

Context Gathering

Aggregate similarity scores, seller history, and metadata

Feedback Loop

Corrections improve future LLM performance

LLM Reasoning

Generate structured decision with supporting rationale

Structured Output

Decision + explanation + confidence score

Human Oversight

Spot-check LLM decisions for quality assurance

78%

Auto-Verified

Medium-confidence cases handled without human review

68%

Review Reduction

Decrease in manual review workload

92%

Appeal Win Rate

Improved decision quality reduces overturned actions

CI/CD & Model Governance

Treating models as first-class software artifacts transforms our deployment velocity. Strict versioning, automated evaluation pipelines, and shadow mode testing enable weekly releases with confidence. GitOps for thresholds means policy changes deploy through the same battle-tested pipeline as code.

1

Model Registry

Centralized versioning with metadata, lineage, and approval workflows

2

Automated Evaluation

Every model candidate runs against holdout sets and production replays

3

Shadow Mode Testing

New models run alongside production without impacting decisions

4

Gradual Rollout

Traffic percentage increases with automated quality gates

5

Auto-Rollback

Degradation triggers automatic reversion to last known good version

6

GitOps for Config

Thresholds and enforcement rules version-controlled and peer-reviewed

Cultural Impact: ML teams ship confidently knowing rollback is automatic. This psychological safety accelerates experimentation and innovation.

Observability & Monitoring

We monitor what matters: business outcomes, system health, and model behavior. Unified dashboards serve ML engineers, SREs, and operations teams with role-appropriate views. The critical insight: monitor decision distributions, not just aggregate accuracy. Shifts in confidence distributions signal degradation before customer impact.

Business Metrics

False positive rates, enforcement latency, violation coverage, appeal rates

System Metrics

Infrastructure health, API latency percentiles, error rates, throughput

Model Metrics

Feature drift, confidence distributions, precision/recall per segment

Early Warning System

- Confidence distribution shifts detected within hours
- Feature drift alerts before accuracy degrades
- Anomaly detection on decision patterns
- Automated alerts route to appropriate teams

Unified View

- ML, SRE, and Ops share common metrics
- End-to-end traceability from listing to decision
- Correlation between model and business metrics
- Real-time dashboards for incident response

Incident Response & Debugging

Production incidents are inevitable. Our architecture makes them manageable. End-to-end trace IDs connect listings to decisions through every system hop. Immutable event streams enable deterministic replay. Structured logging and correlation turn debugging from art into science.

1

Latency Spikes

Cause: Downstream service degradation or index hotspots

Response: Circuit breakers activate, traffic shifts to backup regions

2

False Positive Waves

Cause: Feature drift or threshold misconfiguration

Response: Auto-rollback triggers, affected decisions reversed

3

Data Corruption

Cause: Upstream schema changes or parsing errors

Response: Validation catches at ingestion, dead letter queue captures bad events

Debugging Arsenal

- **Trace IDs:** Follow decisions end-to-end across services
- **Event Replay:** Reproduce incidents deterministically
- **Structured Logs:** Query by listing, seller, or model version
- **Runbooks:** Step-by-step response for common scenarios
- **Error Budgets:** Quantify reliability and prioritize improvements

Mean time to resolution (MTTR) dropped from hours to minutes through systematic incident engineering.

Results & Operational Impact

1	2
Throughput Throughput increase - 2x	**Latency** p99 Latency decreased by half
3	**4**
Manual Review 68% decrease in Manual review percentage	**Release Velocity** Release velocity increased by 2x

The system processes a high volume of listings daily, demonstrating strong precision and recall. More importantly, we ship model improvements frequently, leading to a substantial increase in iteration velocity. The business impact extends beyond metrics: trust in automated enforcement grew significantly as reliability improved.

Key Takeaways: Building Trustworthy ML Systems

Production ML success requires convergence of machine learning expertise and DevOps discipline. The systems that win aren't just accurate–they're observable, reliable, and rapidly improving. Here's what we learned building enforcement at scale.

ML + DevOps Convergence

Model accuracy without operational rigor fails in production. CI/CD, observability, and incident response are first-class ML requirements.

Observability Enables Trust

Visibility into decisions, drift, and system health builds stakeholder confidence and enables fast recovery from incidents.

Velocity Through Safety

Automated testing, shadow mode, and rollback mechanisms increase deployment velocity by reducing risk.

Engineer for Failure

Production systems must handle degradation gracefully. Circuit breakers, retries, and backpressure prevent cascading failures.

Deployment Over Training

The best model is worthless if it can't be deployed safely and monitored effectively. Trustworthy ML systems are deployed, not just trained.

> "Building ML systems for production isn't about choosing between accuracy and reliability–it's about achieving both through disciplined engineering."