BEYOND MIGRATION

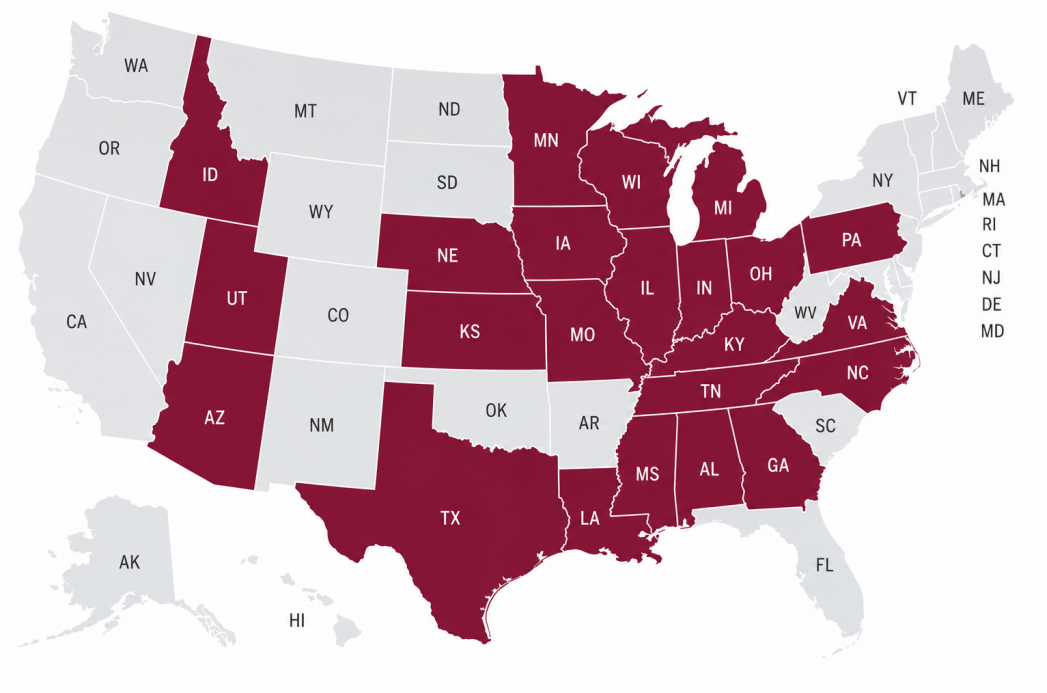
Building a Platform That Lasts

Pekin Insurance's Journey on WSO2 — Two Years After Go-Live

WSO2Con 2026 | Pekin Insurance + WSO2 Managed Services

About Pekin Insurance

- U.S.-based insurance provider with 100+ years of experience
- Products: Property & Casualty, Life, and Health insurance



- Two core business units: Life and P&C — each with distinct integration needs and delivery cadences
- Platform backbone: Guidewire P&C, LIDP and Resonant (Life), and 10+ other systems
- Infrastructure: 5 dedicated Kubernetes clusters · 13 named environments across Dev, QA, UAT, Prod

The Migration in 90 Seconds

2023 WSO2 engagement begins — MuleSoft-to-WSO2 migration starts

2024 Migration declared complete — platform running on WSO2 APIM + Micro Integrator

What We Built:

- Shared APIM gateway (Life + P&C) with separate internal and external gateways
- Separate MI deployments: batch (P&C) | realtime (Life) | dashboard
- 13 environments on AWS EKS — Dev → QA → UAT → Prod
- Immutable-image model: two distinct images (Life and P&C); env params via persistent volume

💡 **"Done" was a starting line, not a finish line.**

Post-Migration: Identifying the Gaps

How we assessed the platform — and what we found

How We Assessed:

Platform Review

Structured analysis of the integration platform — delivery practices, pipeline coverage, standards maturity

Internal Team Assessment

Life and P&C teams reviewed integration patterns, build pipelines, and delivery practices

Operational Observations

Post-go-live monitoring: incident patterns, manual effort levels, deployment error rates

Gaps Found:

Governance

No standards, no linting, no enforcement across teams

APIM CI/CD

Manual deployments 1–2 hours each 48-hour lead time, error prone manual steps

Security Posture

Lack of modern authentication methods, limited threat protection in APIs

Observability

Inconsistent correlation IDs, no alerting, all diagnosis reactive

Developer Experience

Inconsistent API standards and best practices across Life and P&C

Gap analysis converted known problems into a committed five-pillar improvement plan — with clear owners, timelines, and measurable outcomes.

What "Beyond Migration" Actually Means

1

Governance

Standards & enforcement at scale

- Enforcing PR checklists for code promotion
- Spectral: REST, OWASP & APIM rules
- Global gateway policies improvemens

2

CI/CD Maturity

Automation across the full platform

- APIM CI/CD improvements · Spectral liniting for API definitions
- 104 Karate tests · 100% scripted
- 1-2 hr deploys → 10–15 min target
- Synapse unit test gates for CI phase of MI

3

Security Hardening

Closing gaps left from migration

- Authentication modernization across all layers
- OWASP Spectral gates at dev + CI
- API threat protection policies
- Fluent Bit → Rapid7 SIEM for threats

4

Observability

Knowing what's happening, when it matters

- Correlation IDs: APIM → MI → backend
- OpenSearch · CloudWatch · Rapid7 SIEM
- PII masking before log ingestion
- Alerting on failures from observability tools

5

Developer Enablement

AI-powered tooling to accelerate delivery

- WSO2 MI Copilot + GitHub Copilot
- WSO2 Integration Studio to VS Code migration
- Unified VS Code workflow: Life & P&C
- WSO2 MI unit testing and API definition linting with Spectral rules.

The Governance Problem

- Post-migration: two teams, two very different histories
- Life team: MuleSoft 3-tier heritage (System → Process → Experience layer model)
- P&C team: pragmatic, faster-moving, less modular
- Result: inconsistent API naming, missing security policies, no linting, no shared standards

Our Governance Model

Principles

Why: intent and goals

Standards

What: naming, security, schema rules

Enforcement

How: linting, CI gates, deployment gates

Three Spectral Rulesets:

- wso2-rest.spectral.yaml — WSO2-specific REST API conventions
- owasp.spectral.yaml — OWASP API Security Top 10 checks
- apim.spectral.yaml — APIM deployment and gateway policy rules

Governance at four layers:

→ IDE / Developer — Spectral linting + MI unit testing

→ CI/CD Pipeline — Spectral lint gate for API Manager CI + Sypanse unit testing for CI phase of MI

→ Deployment at APIM Level — Spectral lint gate for deploy time for APIs

→ Runtime — Global gateway policies for gateway runtime

★ **Same rules enforced at both developer and CI/CD level — CI/CD is the safety**

Centralized delivery:

Rules packaged and delivered as centralized set of rules · Teams install via Git · One update propagates to all API and Integration repos instantly

CI/CD: The Gap and How We Fixed It

New CI/CD Layer — Strong Foundation ✅

→ Detect changes → build immutable image → Trivy scan

→ Push to Docker registry → update *-deploy branch → Helm deploy

→ TeamCity smoke test gate for the integrations at the CI stage

→ Spectral lint gate — API definitions validated against REST, OWASP & APIM rulesets

Current CI/CD Layer — Not Yet Propagated ❌

⚠ API deployments: largely manual including the pre-prod environments

⚠ Manual deployments 1–2 hours each, error prone manual steps

⚠ Target: 10–15 minutes · Lower error rate

Toolchain: GitHub Actions · Spectral · Nexus · TeamCity · Trivy · Helm · Secret Store CSI | 13 environments: Dev → QA → UAT → Prod

The integration layer got automation first — the APIM layer played catch-up.

End-to-End SDLC Pipeline

Developer Phase	CI/CD Build	QA Gate	Production Deploy	Monitor & Review
API Spectral linting	Spectral lint gate	Deploy to QA env	Deploy to server	Health monitoring
Integration unit tests	Maven build	Karate integration tests	Apply gateway policy	Alerting
PR submission	Unit tests (safety net)		QA → Prod promotion	✓ Pass → done
	Security scan			✗ Fail → rollback

Similar pipeline for API Manager and MI artifacts · Developer runs gates locally; CI/CD is the safety net

The Security Debt We Inherited

- ⚠ Limited OAuth / modern authentication mechanisms
- ⚠ Limited threat protection on the gateway
- ⚠ Limited usage of API throttling
- ⚠ API definitions not adhering to OWASP API Security Top 10

Critical risk: System was functional but lacking API security best practices.

Security as a Platform Property

- Rollout of Modern Authentication (Oauth2)
- OWASP Spectral rules: catch API security issues at dev time and CI gate
- Fluent Bit → Rapid7 SIEM for log-based threat detection
- Trivy image scan on every build — blocking on critical findings
- Threat protection policies in API gateway
- PII masking before log ingestion

Principle Shift:

🔒 Security embedded in the pipeline runs every time

🔒 Modern authentication mechanisms across all layers — rolling forward

🔒 Security review before prod changes

🔒 Make the secure path the easy path

Observability: Before and After

What We Couldn't See ❌

- ✗ Inconsistent correlation IDs across APIM → MI service boundaries
- ✗ No proactive alerting — all diagnosis was reactive log-searching
- ✗ Lagging performance in the OpenSearch

What We are Building ✅

WSO2 APIM + MI → **Fluent Bit** → **OpenSearch** → **Rapid7 SIEM**

- ✓ Correlation IDs propagated end-to-end (APIM → MI → backend)
- ✓ PII masking before log ingestion
- ✓ CloudWatch for node-level metrics and log retention
- ✓ Heap/thread monitoring for runtime health
- ✓ Implemented OpenSearch alerting on failures
- ✓ Improved performance with OpenSearch file system

Lesson: observability is a first-class engineering concern, not an ops afterthought.

The Developer Experience Gap

- Two teams, different codebases, different tooling familiarity — no shared baseline
- VS Code adoption required active verification across both Life and P&C teams
- Integration Framework will include local Docker testing — developers independent of shared environments
- Integration Studio → MI VS Code extension: every project needed restructuring
- Build stability gaps surfaced during local developer runs — addressed through improved pipeline coverage

The insight

Developer experience problems are invisible until you measure onboarding time and watch someone struggle through it.

Fix developer experience before you ask developers to adopt new tooling.

AI-Powered Development

WSO2 MI Copilot — integrated into VS Code

- Natural language → Synapse sequences, APIs, endpoints
- API definition ingestion → auto-generated integration code
- AI-powered data mapping for JSON / XML / CSV transformations
- Prompt-driven refinement of routing and error-handling logic
- Follow-up prompts refine without manual XML edits

GitHub Copilot — across the broader stack

- Incident triage: correlate logs → source → root cause
- Documentation: runbooks, handover docs, onboarding guides
- Cross-service debugging with consistent code context
- Test generation, security review, IaC assistance

Adoption model: pilot with a high-value scenario → demonstrate savings → phased rollout

The Upgrade Is Not Just a Version Bump

APIM 4.2.0 → 4.6.0 | MI 4.2.0 → 4.5.0

Business drivers

- Reduce platform and support risk on older versions
- Improve security, governance, and operational visibility
- Create a cleaner path for future API and integration needs
- Modernize deployment and support workflows for delivery teams
- Make the change with limited consumer disruption

Expected outcome

- Safer platform operations
- Better change control
- Stronger long-term scalability

What this is not

- Not a redesign of customer-facing APIs
- Not a consumer URL migration
- Not a big-bang platform replacement

The upgrade became a forcing function: fix accumulated security gaps, build fragility, APIM CI/CD lag, and observability debt — all in one structured arc.

Honest Lessons: What Worked

✓ **Phased governance rollout**	Warnings before enforcement prevented backlash and built trust with both teams
✓ **Immutable image model**	Knowing exactly what's deployed eliminates "works on my machine" across 13 environments
✓ **Security embedded in pipelines**	Security that requires human memory will be skipped — automation runs every time
✓ **Centralised governance rules**	Centralized rules repository — consistency at zero marginal cost
✓ **Dual-level testing**	Same tests run by developer and CI/CD — CI/CD is the safety net that catches what was missed

The Platform Roadmap — 2026 and Beyond

1	**Domain-based MI slicing**	From coarse batch/realtime to domain-aligned deployments — better isolation and ownership
2	**Modern auth adoption**	Auth modernization across all layers: XAPI, PAPI, SAPI
3	**APIM CI/CD parity**	Bring API automation to the same level as the integration layer
4	**Observability maturity**	End-to-end distributed tracing, proactive alerting, complete SIEM integration
5	**Broader AI enablement**	AI assisted code development with Integration Framwork, MI Copilot adoption, standardised VS Code-first workflow, measured onboarding improvement

Thank You

Questions & Discussion

- → Governance adoption in multi-team environments
- → CI/CD pipeline design for WSO2 on Kubernetes
- → AI copilot workflows for integration teams
- → Handling two-team platform dynamics