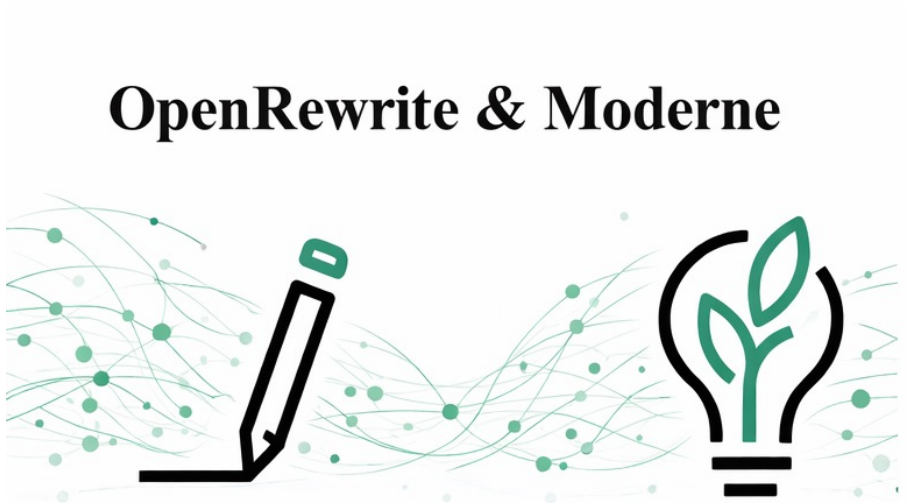
2026 OPEN SOURCE IN FINANCE FORUM Toronto

Deterministic Modernization at Enterprise Scale



Track / Category

Developer Tools, Best Practices, and Talent Development

Session Type / Duration

Lightning Talk

Morgan Stanley

Liborio Ciccarello

Technology Director

Full-Stack Engineer | DevOps Lead

- Drive CI/CD and infrastructure best practices
- Leverage Advanced Tooling to accelerate delivery
- Optimize DevOps through automation

https://www.linkedin.com/in/liborio-thomas-ciccarello-a09a69191

Operational Hygiene: The Hidden Scaling Constraint

Challenges

Constant Change

Tooling swap, vendor contracts, platform migrations

Security is Urgent

CVEs / Zero-day change requirements, broad remediation

Unscalable Manual Hygiene

Humans can't keep up with software estate

Legacy Drag

Old dependencies lead to slower mandatory change

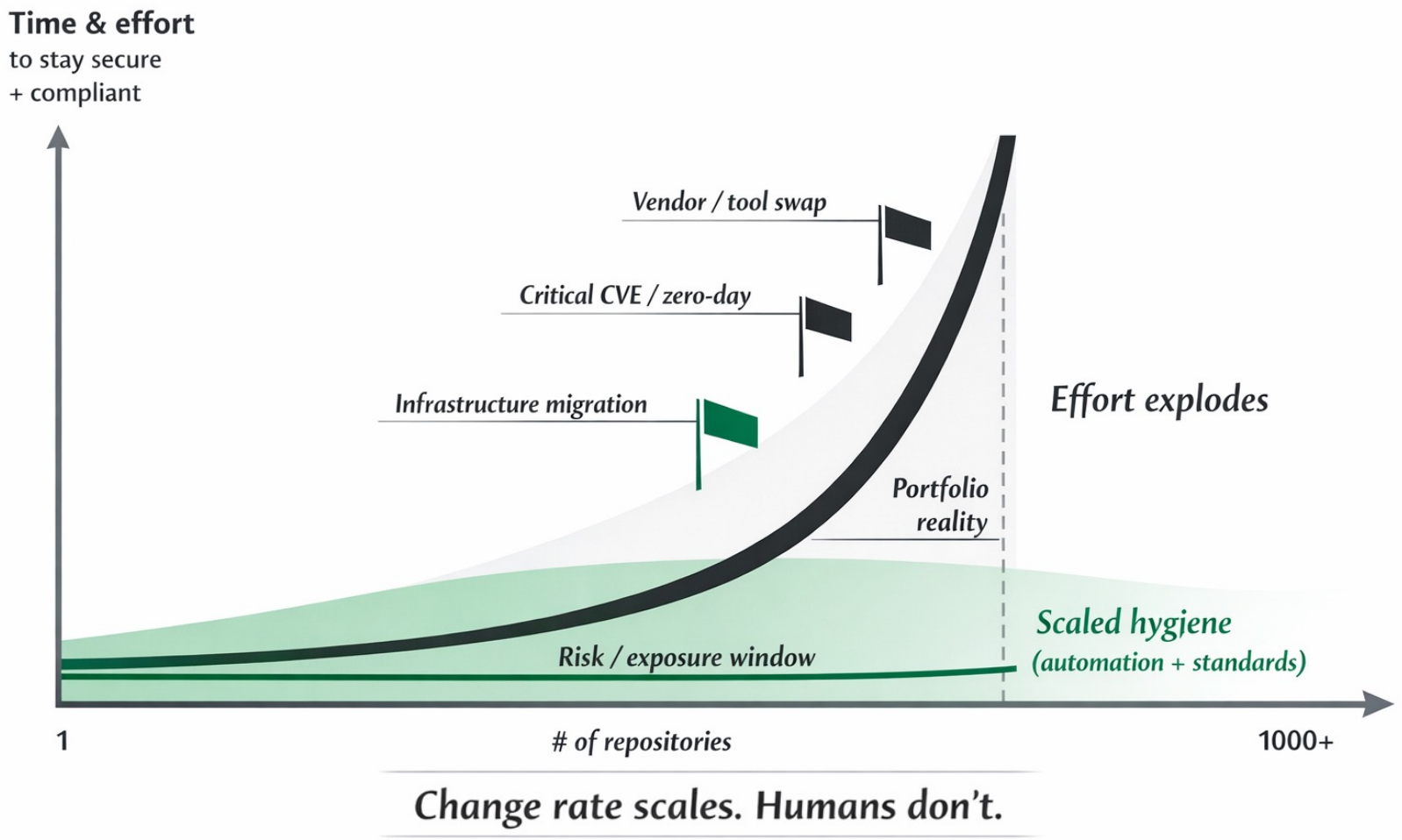
Business Impact

Longer risk exposure, slower delivery

Goal

Automation / Scalable Tooling

Reducing effort and flattening the curve



The Manual Scaling Problem

- **Visionary** — Defined DevOps Software Hygiene Plan
- **“Simple Fix”** — Starts with one upgrade
- **N - times** — Applied on 100s/1000s of repositories
- **Resource Constraints** — Every repository needs time, risky, expensive
- **Priority Collision** — Deadlines, delivery, incidents affect rollout
- **Hygiene Accumulates** — Partial Adoption, Inconsistency
- **Cycle Repeats** — Next Upgrade, CVE arrives

Even with an LLM/Agent it’s still “manual scaling” when you have 50,000 repos!

OpenRewrite & Moderne

How OpenRewrite works

1. **Abstract Syntax Tree**
   OpenRewrite parses source code into an enriched Lossless AST (“LST”) representing the code's structure in a tree format.
2. **Run a Recipe**
   It applies a series of refactoring ops or "visitors" that traverse the LST and apply transformations.
3. **Produce Diff Files and Data Tables**
   Results of recipe can produce diff/code changes as well as data tables / analysis.
4. **Apply Changes**
   Apply patch file / diff to a branch or fork of your repo, run tests, and commit your code

OpenRewrite vs. Moderne

OpenRewrite by Moderne

- Open-Source Solution
- Multi-Recipe/Single repo at a time
- Runs via Gradle/Maven plugin
- All OSS Recipes available

Moderne

- Integrated Vendor Solution built on OpenRewrite
- LST persistence and reuse
- No context window limitations
- SaaS platform / CLI Tool
  - Recipe Marketplace
  - Multi-Repo/Multi-Recipe Execution (@ Scale)
- Model Context Protocol (MCP) Server

My Journey - Adoption

Pre-Built Recipes: Java & Spring Boot

Manual Upgrade

Time Consuming

Manual Effort scales poorly

Research Heavy

Hidden Investigations per change

Reduced Dependency Freshness

Upgrades land too slowly

Multi – Repository Upgrades

Fast & Deterministic

Automated execution scales efficiently

Low Research Overhead

Standardized, repeatable changes

High Freshness Gains

Upgrades land quickly and stay current

Reduced Refactoring Time

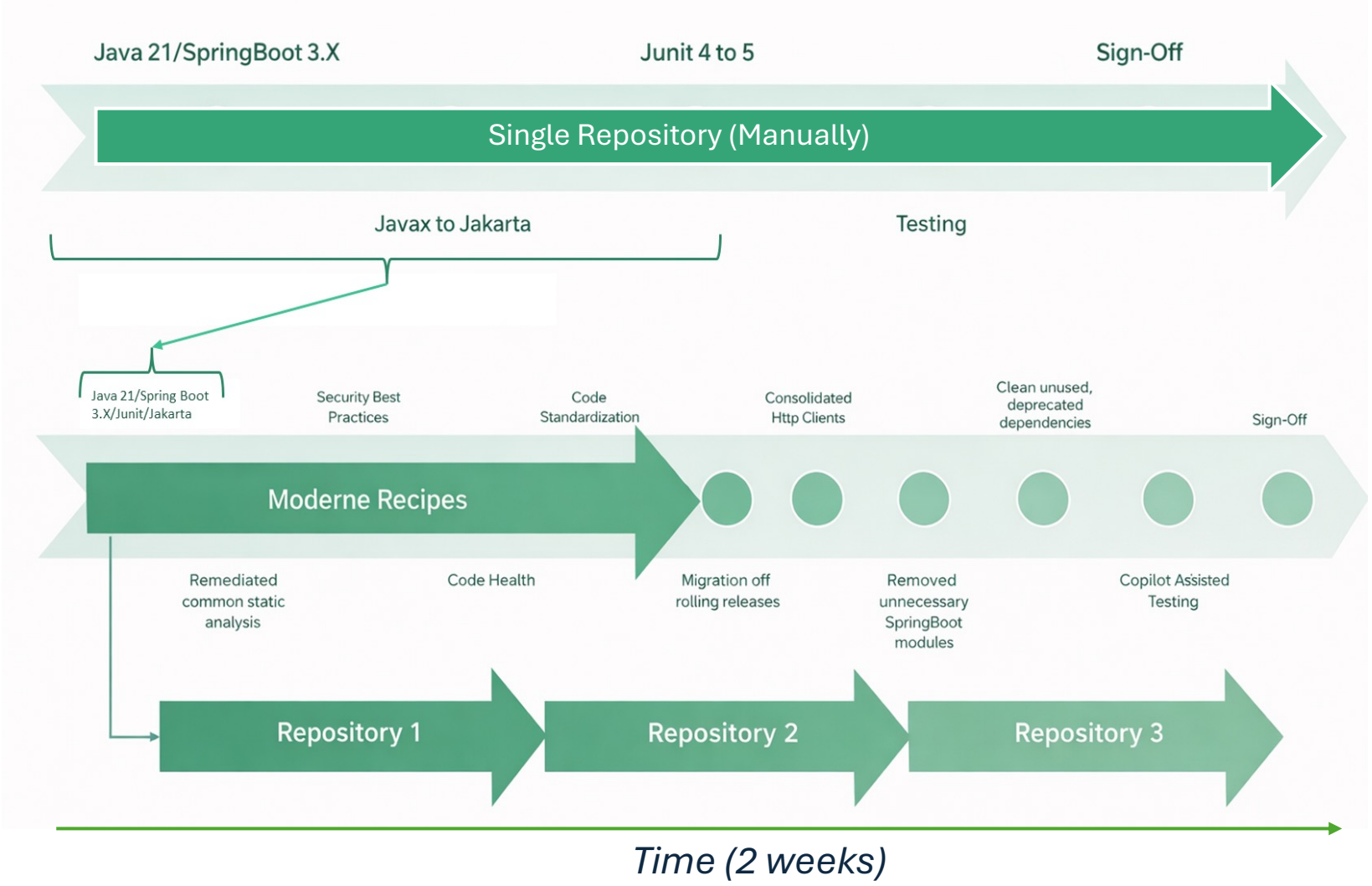
Time to improve structure

Automated Testing

Confidence without rushing

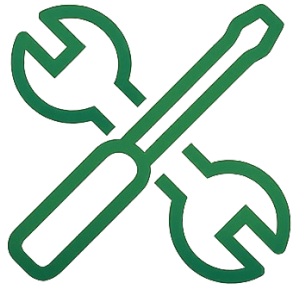
Continuous Cleanup

Debt reduced incrementally



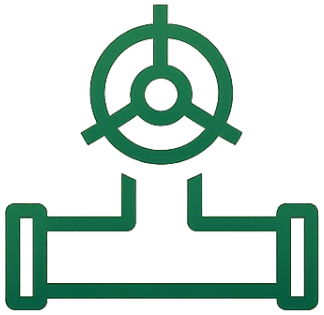
Recipe Authorship

Deterministic Transformation at Scale



Tooling Onboarding Across Languages

- Recipe-driven **repeatable, testable, cost-effective and predictable** onboarding model
- Standard **common setup steps** across **heterogeneous** codebases
- **Reduced variability** while respecting **language specific differences**
- Tooling enables **sustainable coding practices, quality and security levels**

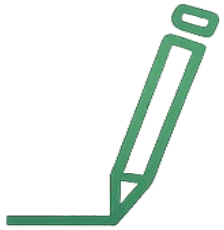


CI/CD Pipeline Transformation Flow

- Recipe-driven **CI/CD Modernization**
- **Controlled** rollout strategy and applied changes (minimally-invasive)
- **Weekly pipeline builds** while preserving existing project configurations
- Leveraged patented **Almanac** data warehouse:
  - Identify target repositories
  - Sequence adoption safely
  - Continuously validate and report on build status

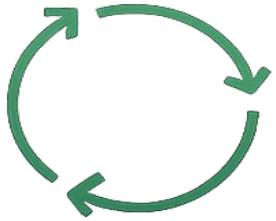
Key Takeaways

AI-Driven Deterministic Scalability



Deterministic Refactoring

- **Predictable** changes
- **Composable** flows for modernization programs
- **Reduced Risk**

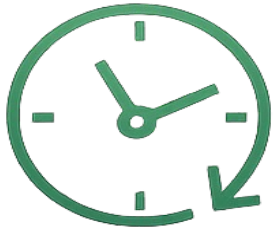


Repeatable Process

- Build recipe **apply at scale**
- **Continuously enhance** recipes

Enterprise-Safe Automation

- **Reviewable** and traceable
- **Controlled rollout**
- **SDLC-compliant** execution
- **Minimal** surprises



Increased Developer Productivity

- **Reduce** upgrade **toil**
- **Ship** features **faster**
- **Improve developer experience**
- **Less disruption**

Leverage AI

- AI **accelerates** recipe **creation** and **discovery**
- **Codify last-mile** fixes back into recipes

"Let's build the recipes that financial services need together, in FINOS!"