



Connect

Beyond the Zero CVE Myth: Building the Agentic Software Factory of the Future

James Labocki

Senior Director of Product Management, Red Hat



Enterprises are managing A LOT of vulnerable software

90%

of enterprise code is open source
or third-party dependencies

9.8 T

package downloads in 2025
across npm, PyPI, Maven, NuGet

581

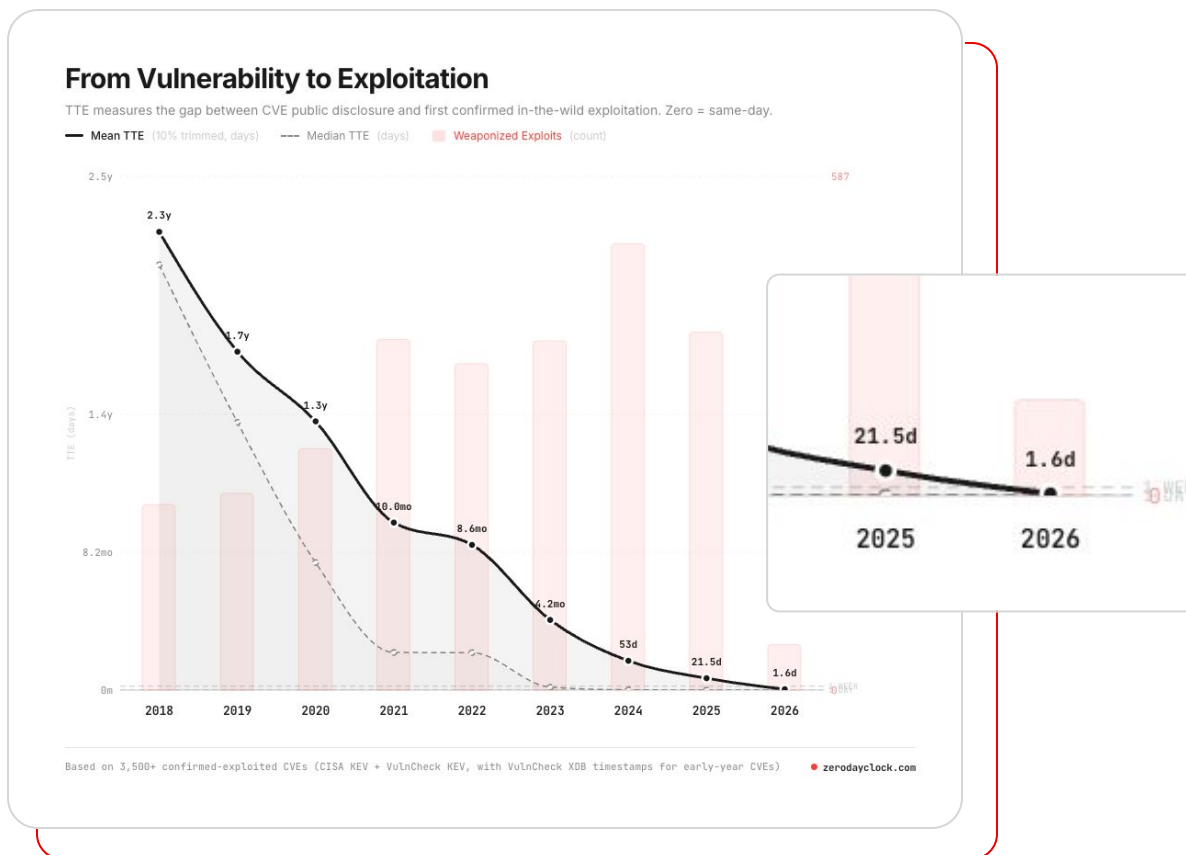
average known
vulnerabilities per
codebase

Source:

Anthropic "Project Glasswing" Apr 2026 · NVD Apr 2026 · Sonatype 2026 SSSSC

Cybersecurity in the age of AI is changing

Time-to-exploit is approaching zero

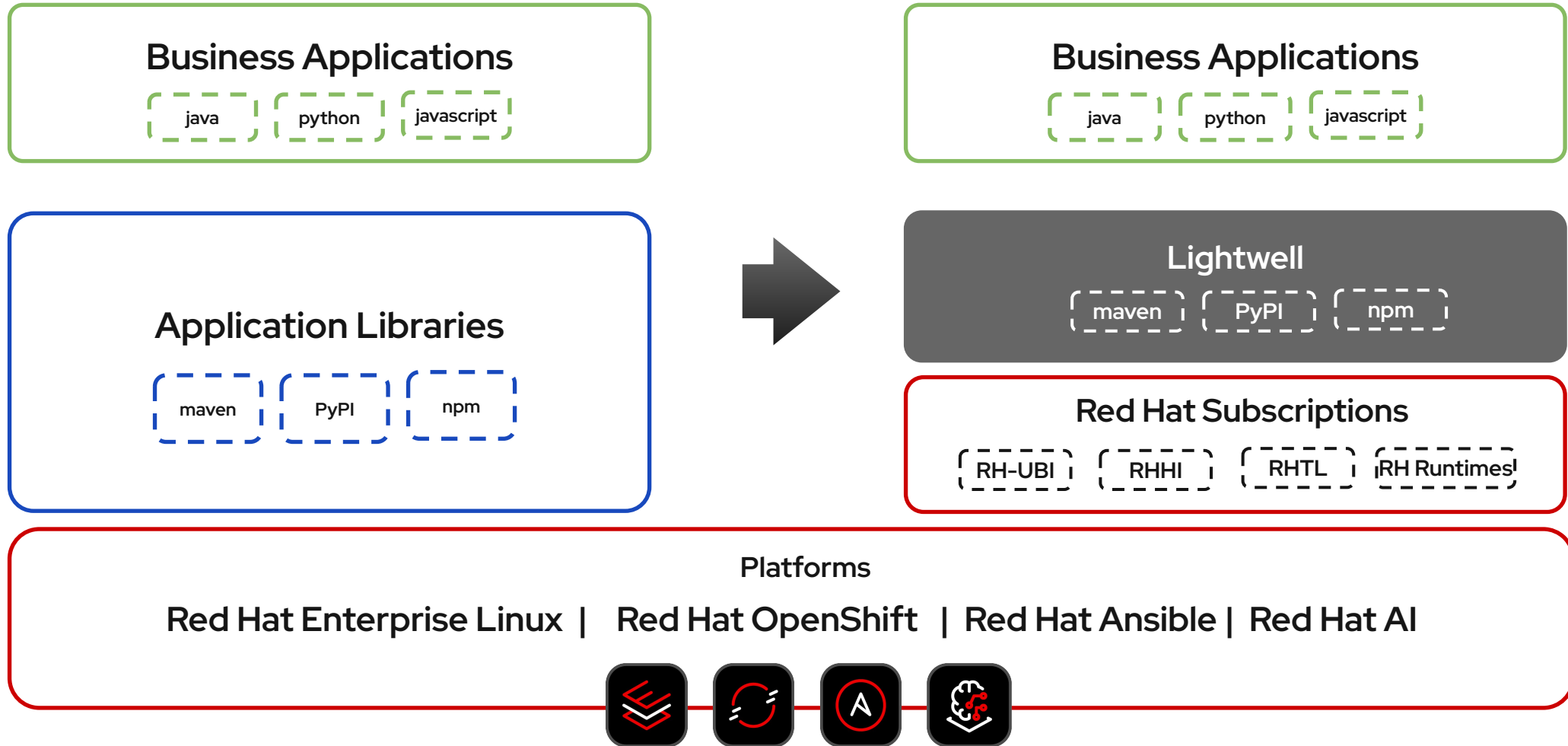


What will change...

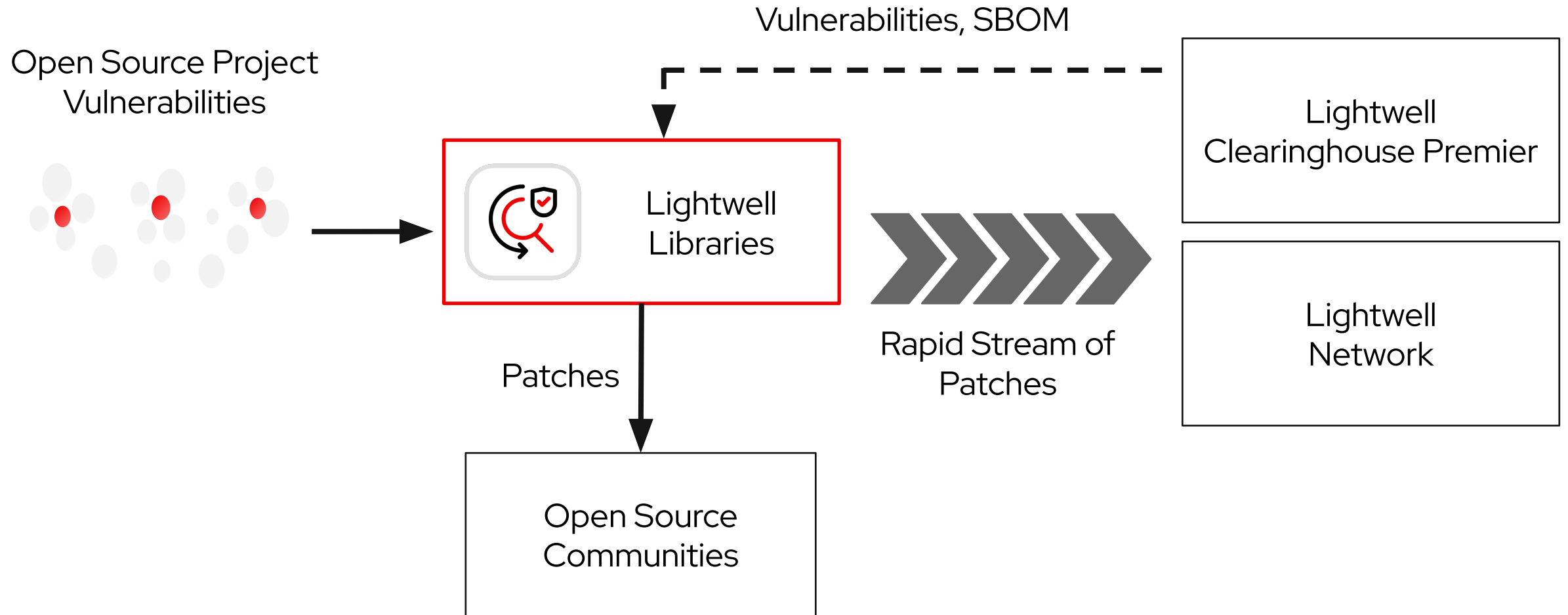
- ▶ Vulnerability disclosure processes
- ▶ Software update processes
- ▶ Open source and supply chain security
- ▶ Software development lifecycle and secure-by-design practices
- ▶ Standards for regulated industries
- ▶ Triage scaling and automation
- ▶ Patching automation



Lightwell remediates the Vulnerability Gap

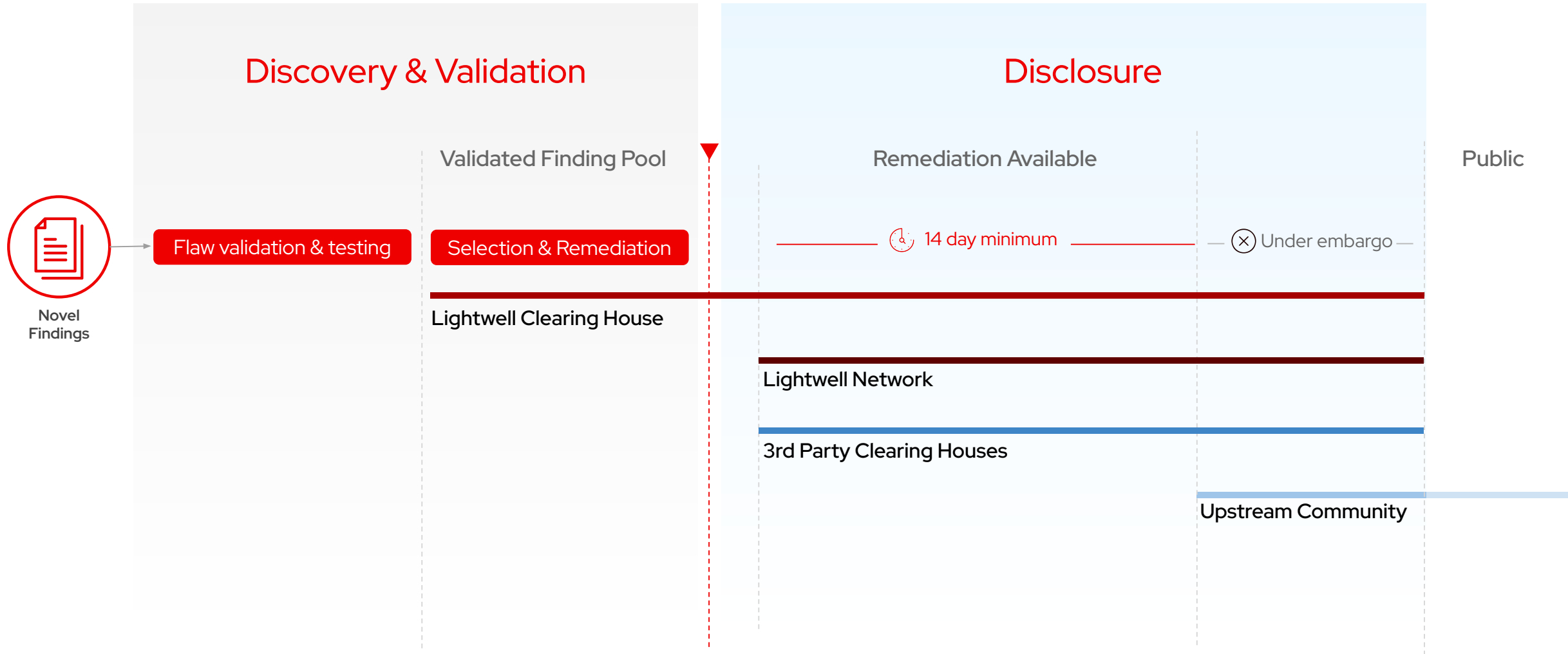


Lightwell: How it works



Lightwell Discovery & Disclosure

Confidential



Project Lightwell

Securing the OSS ecosystems you depend on

Curated Package Registries	Patched Packages	Trusted Verified Sources	Open Source Commitment	AI-Powered
Scanned, patched, signed, certified	At the exact versions you already run. No upgrades needed.	SLSA Level 3 builds, HSM-backed signing, SBOMs for every package	Remediations will be submitted to upstreams to fix bugs at the source	Vulnerability triage with automated severity assessment and remediation prioritization

AI-powered Remediation Engine + Human Expertise



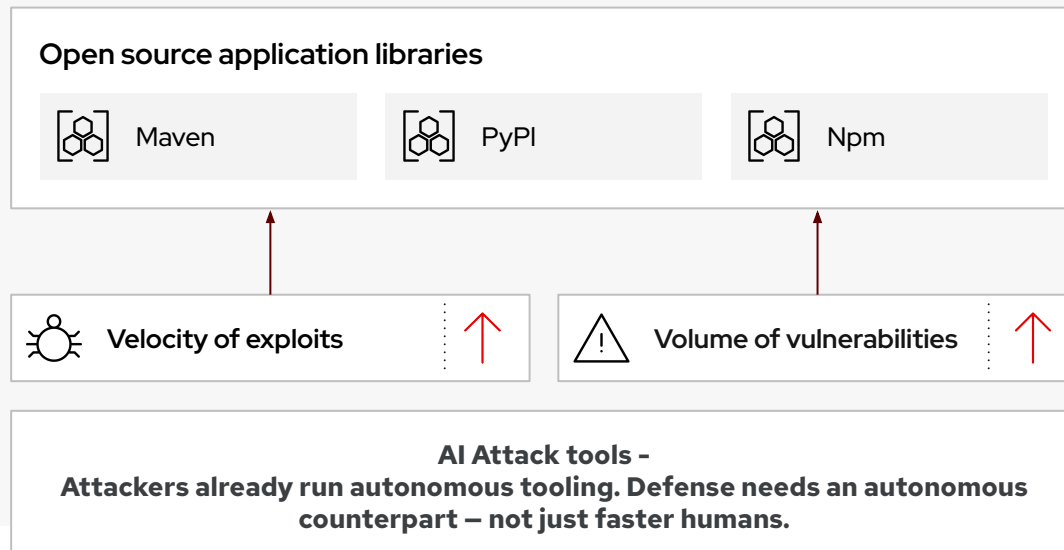


The Mythos moment

Accelerating two parallel challenges: “The vulnerability gap”

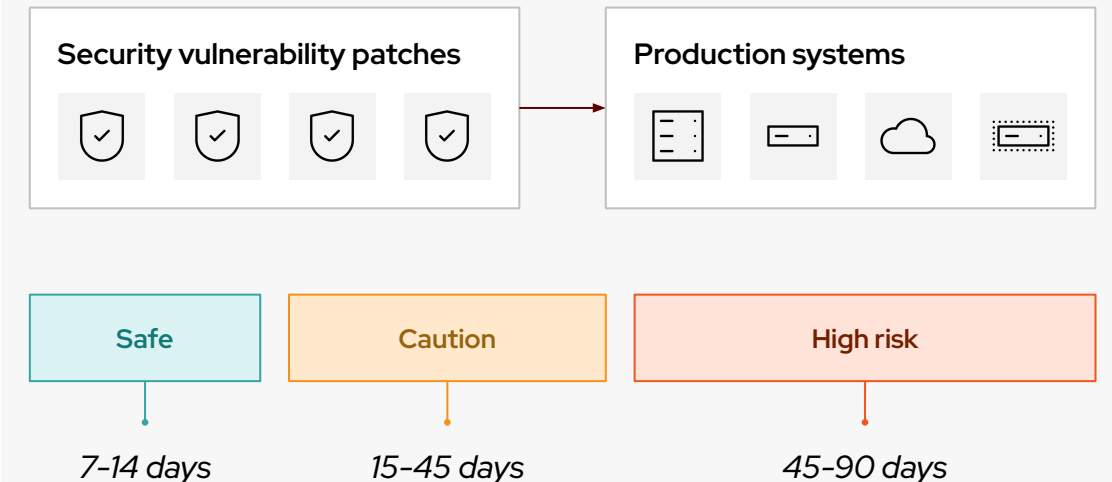
Massive CVE growth

Problem: Time-to-exploitation dropping

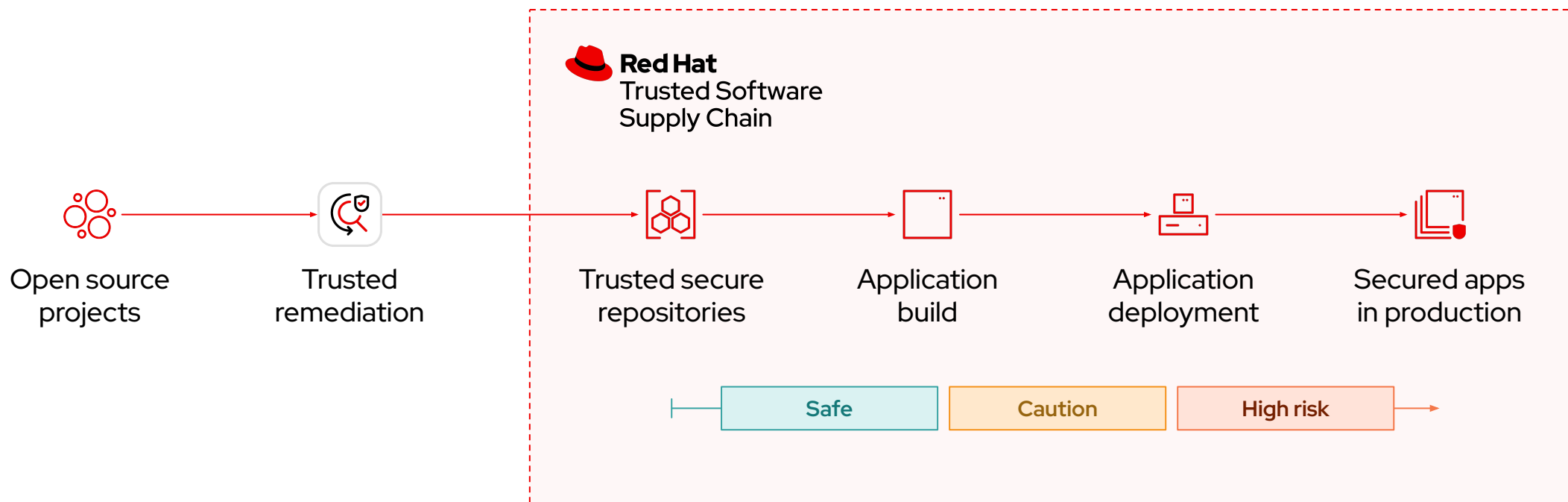


Average days patch to production: **40+**

Problem: Rebuild and deployment velocity



Secure remediation and rapid deployment



Red Hat has been building and releasing software for a long time ...

RHEL 2.1
Released



JBoss
Acquired



OpenShift
Launched



Ansible
Acquired



Red Hat AI
Launch



2002

2006

2011

2015

2023



Red Hat

intel

Red Hat has been building and releasing software for a long time ...

RHEL 2.1
Released



JBoss
Acquired



OpenShift
Launched



Ansible
Acquired



Red Hat AI
Launch



2002

2006

2011

2015

2023



"Software Supply Chain attacks" start in 2011

2002 OpenSSH
2002 Sendmail

2007 SquirrelMail

2011 RSA Security
2015 XcodeGhost

2017 CCleaner

2020 SolarWinds
2024 XZ Utils

2022 CNCF Secure Software
Factory Guidance Published

2000 - 2010

2011 - 2015

2017 - 2019

2020-2024

Distribution Compromises or
Mirror Hijacks

Targeted
Espionage

Update Hijack

Mega-Breach



Supply-chain Levels for Software Artifacts, or SLSA ("salsa")

Level	Requirements
0	No guarantees. SLSA 0 represents the lack of any SLSA level.
1	The build process must be fully scripted/automated and generate provenance.
2	Requires using version control and a hosted build service that generates authenticated provenance.
3	The source and build platforms meet specific standards to guarantee the auditability of the source and the integrity of the provenance respectively.



Components of a Trusted Software Factory

Distribution

Artifacts made available to downstream consumers via secured storage

Management

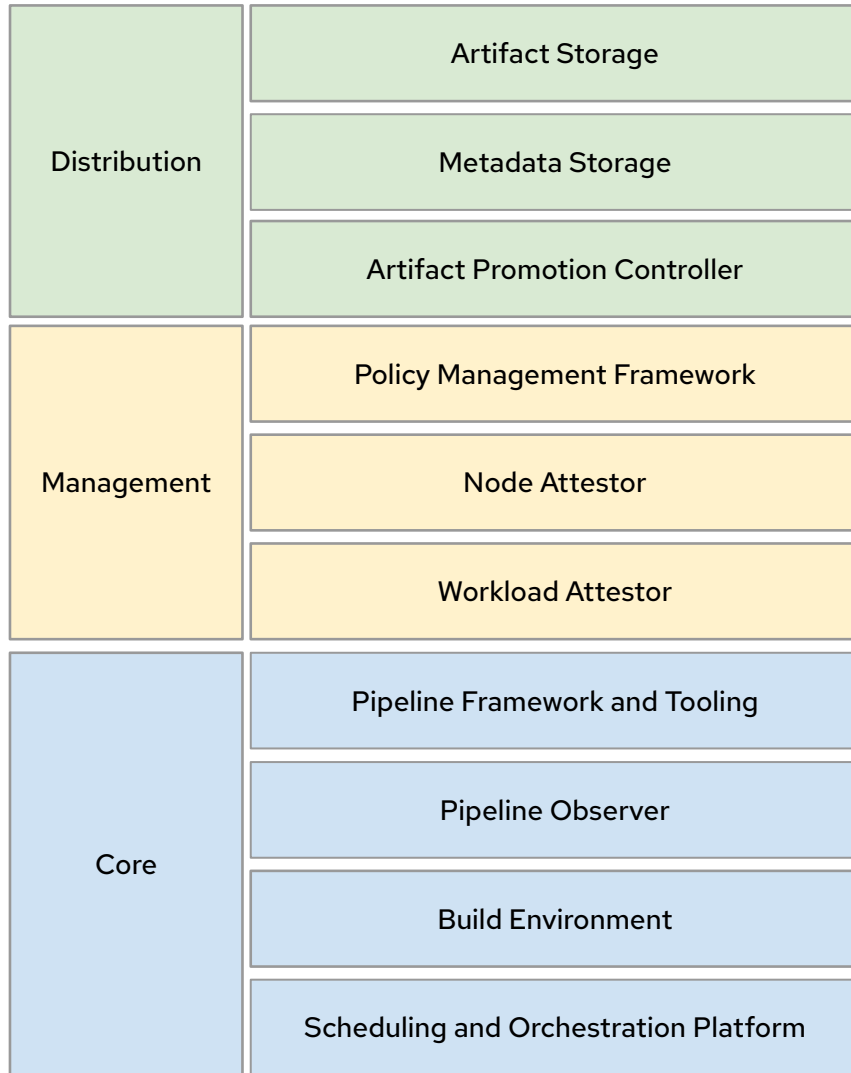
A Policy Management Framework to enforce controls and gates

Core

Takes inputs of the factory and processes them into artifacts



Components of a Trusted Software Factory



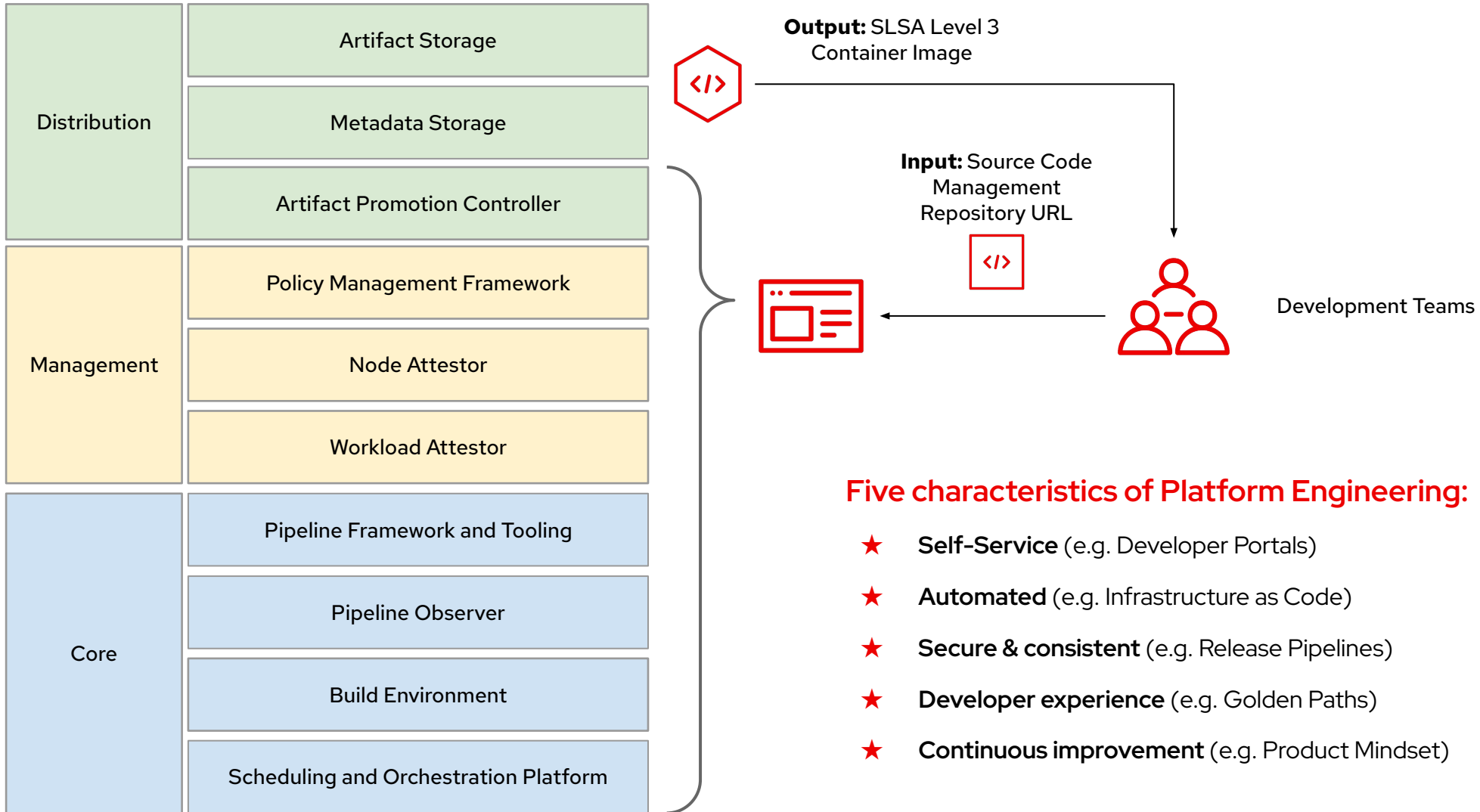
Components of a Trusted Software Factory

Upstream Projects

Distribution	Artifact Storage	Project Quay
	Metadata Storage	
	Artifact Promotion Controller	Konflux Release Service
Management	Policy Management Framework	Conforma
	Node Attestor	CoreOS
	Workload Attestor	SPIFFE/SPIRE
Core	Pipeline Framework and Tooling	Trusted Tasks
	Pipeline Observer	Tekton Chains
	Build Environment	Tekton
	Scheduling and Orchestration Platform	Kubernetes



Developed and Delivered via Platform Engineering



The Results



Months → Days

Red Hat can implement security checks across product portfolio in days instead of months



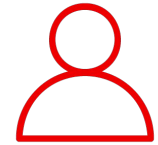
Manual → Automated

Automated updates of external documents like errata pages and vex files



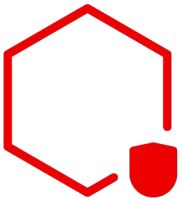
Reduced Attack Surface

Significant reduction in disparate systems involved in the build and release process (200 → ~10)

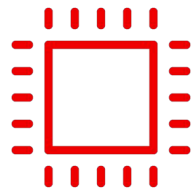


Improved User Experience

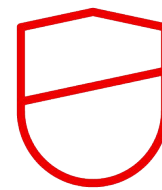
Onboarding source code in a few simple steps compared with dozens of steps previously



Over 2 Million Images and Artifacts built and released in 2025 alone



Achieved Multi-arch Support (z, x86_64, ppc-64, arm)



Achieved SLSA Level 3 Builds

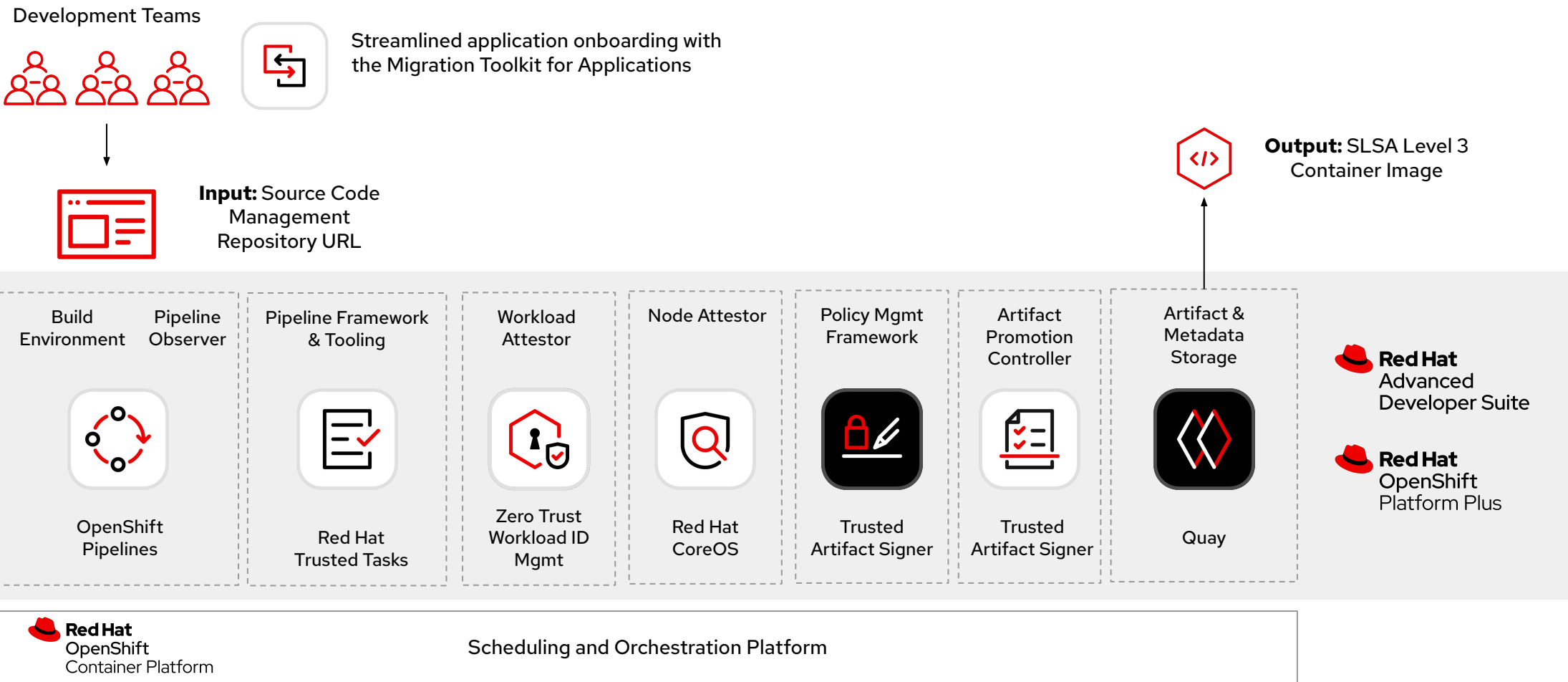


24 hour critical CVE response achieved



Implementing a Trusted Software Factory

with OpenShift Platform Plus and Red Hat Advanced Developer Suite





LOOKING FORWARD



The agentic SDLC spectrum

From Simple Prompts to Autonomous Agents



Traditional SDLC

Pipeline execution is automated, but every decision is human.

Manual and bottlenecked

- ▶ Humans write all code, review all PRs, run and review all tests.
- ▶ Humans triage failed pipelines and gate deployments.
- ▶ Humans coordinate stages and phases between teams.



Solo activities

Developers go deeper into backlogs, faster

Copy and paste

- ▶ Prompting LLMs and manual pasting into editors.

IDE assistant

- ▶ Cursor, Claude Code, Copilot integrated into your workflow.



Team capabilities

Fast feedback eliminates requirements toil

Review agents

- ▶ Qodo, Coderabbit: AI reviewing as a dedicated assistant.

Autonomous with review

- ▶ Agents work in the background, presenting work for human sign-off.



Higher autonomy

Developers focus on unique value-add. Agent does the rest.

Autonomous + escalation

- ▶ Agents work in the background; ask for help only when stuck.

Discovery

- ▶ Agents actively find bugs, tech debt, and new opportunities.

»»»» Increasing agentic capability and autonomy »»»»



← → ↻ 🔍 /jlabocki/tailwind/3100/ 📁 🔍 📄 🔄 ⭐ 🧑

← → 🔍 (Workspace) 🗨️

📄 📁 🔍 🧑 6 🚀 📦 🌟 📝

EXPLORER

▼ (WORKSPACE)

▼ tailwind

- > installer
- > mcp
- > policies
- > prompts
- > sandbox
- > schema
- > scripts
- ▼ skills
 - > account-investigate
 - > context-setup
 - > cross-bu-synthesis
 - > customer-signal-intake
 - > dataverse-query
 - > domain0-sync-prep
 - > domain0-sync-summarize
 - > eol-comms
 - > feature-triage-report
 - > gws-morning-briefing
 - > jira-brief
 - > jira-feature-create
 - > jira-inbox
 - > jira-outcome-create
 - > jira-outcome-describe
 - > jira-rice-scoring
 - > jira-status
 - > knowledge-registry
 - > mr-check
 - > official-brand-messaging
 - > official-brand-personality
 - > official-brand-voice
 - > official-find-cust-ref
 - > partner-research
 - > pmm-ebook
 - > pmm-field-feedback
 - > pmm-geo-optimization
 - > pmm-jira-buyer-messaging
 - > pmm-messaging-ansible
 - > pmm-messaging-lightwell
 - > pmm-messaging-openshift

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

🔍 claude ⚠️ + ▾ 🗑️ ⋮ | 📄 ×

tailwind (main) \$ claude

 **Claude Code** v2.1.267
Sonnet 4.6 · Google Vertex AI
/projects/tailwind

▲ 2 MCP servers need authentication · run /mcp

Can you open RFEs for me using the rfe-create skill? If so, how does it work?

Thought for 2s (ctrl+o to expand)

● Skill(rfe-create)
└ Successfully loaded skill · 10 tools allowed

Thought for 3s (ctrl+o to expand)

● Yes, `rfe-create` is available and ready to use. Here's how it works:

How rfe-create Works

What you give it:

- Plain-language description of a customer need
- Pasted doc content, meeting notes, or Slack thread
- A Google Doc URL (if GWS MCP is connected)
- An existing Jira issue key as context

What it does, step by step:

1. Loads your config from `~/.config/tailwind/tailwind.json` (Jira project key, components, auth, etc.)
2. Dedup check – searches your Jira RFE backlog for existing issues that match, so you don't create duplicates
3. JTBD enrichment (Red Hat AI only) – if your BU is `red-hat-ai`, it matches your input against the Jobs-to-be-Done registry and embeds evidence into the draft
4. Asks 2-5 clarifying questions – who's affected, business justification, what the user can't do today, what success looks like
5. Drafts a structured RFE – problem statement, customer context, business justification, acceptance criteria, saved locally to `artifacts/rfe-tasks/`
6. Waits for your review – you approve, request edits, or abandon
7. Creates the Jira issue – only after explicit approval

Key safeguard: It will not create a Jira issue without your explicit go-ahead.

To get started, just say something like:

| "Help me write an RFE – a customer needs the ability to..."

Or paste your notes/doc content directly and I'll kick off the flow. Do you want to try one now?

* Baked for 14s · done 6:32 PM

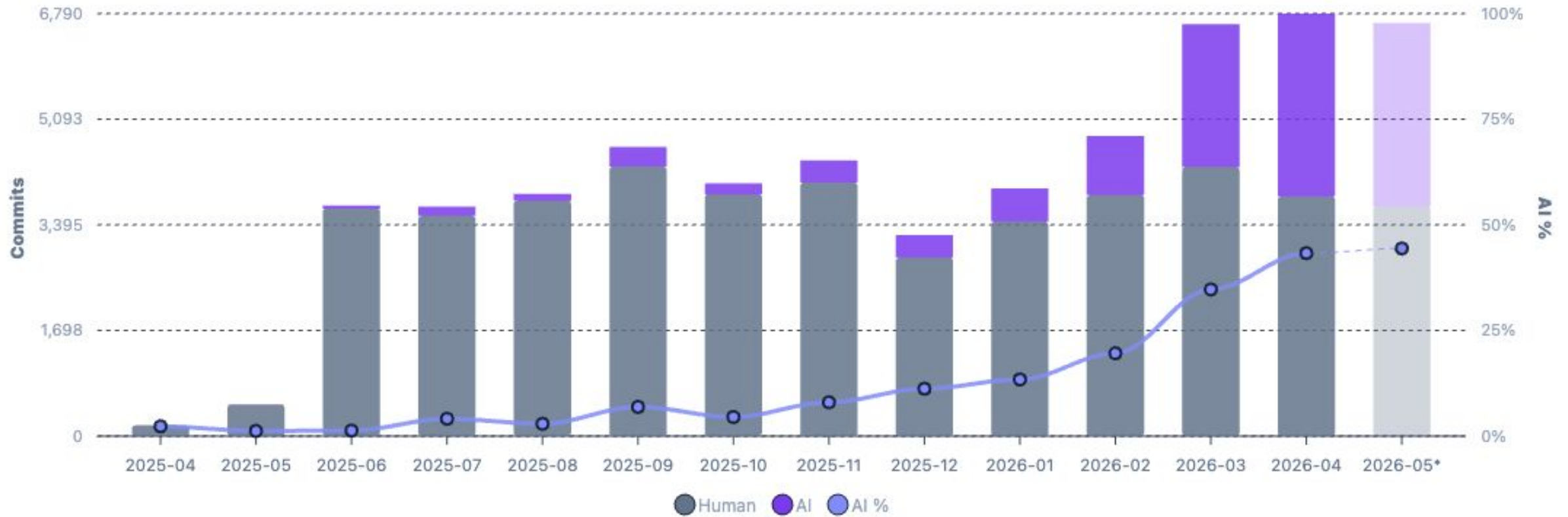
> |

⌨ manual mode on · ? for shortcuts · ← for agents



Agentic Code Development at Red Hat

1900% increase in AI commits in 12 months



The agentic SDLC spectrum

From Simple Prompts to Autonomous Agents



Traditional SDLC

Pipeline execution is automated, but every decision is human.

Manual and bottlenecked

- ▶ Humans write all code, review all PRs, run and review all tests.
- ▶ Humans triage failed pipelines and gate deployments.
- ▶ Humans coordinate stages and phases between teams.



Solo activities

Developers go deeper into backlogs, faster

Copy and paste

- ▶ Prompting LLMs and manual pasting into editors.

IDE assistant

- ▶ Cursor, Claude Code, Copilot integrated into your workflow.



Team capabilities

Fast feedback eliminates requirements toil

Review agents

- ▶ Qodo, Coderabbit: AI reviewing as a dedicated assistant.

Autonomous with review

- ▶ Agents work in the background, presenting work for human sign-off.



Higher autonomy

Developers focus on unique value-add. Agent does the rest.

Autonomous + escalation

- ▶ Agents work in the background; ask for help only when stuck.

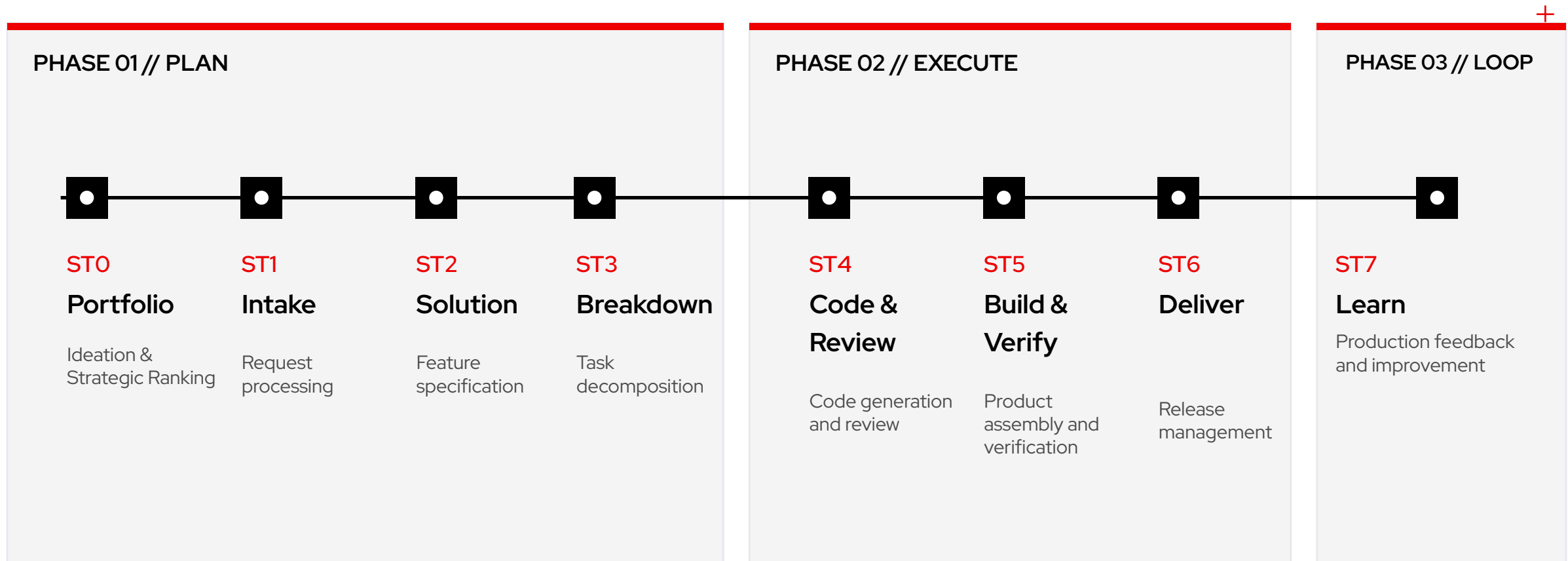
Discovery

- ▶ Agents actively find bugs, tech debt, and new opportunities.

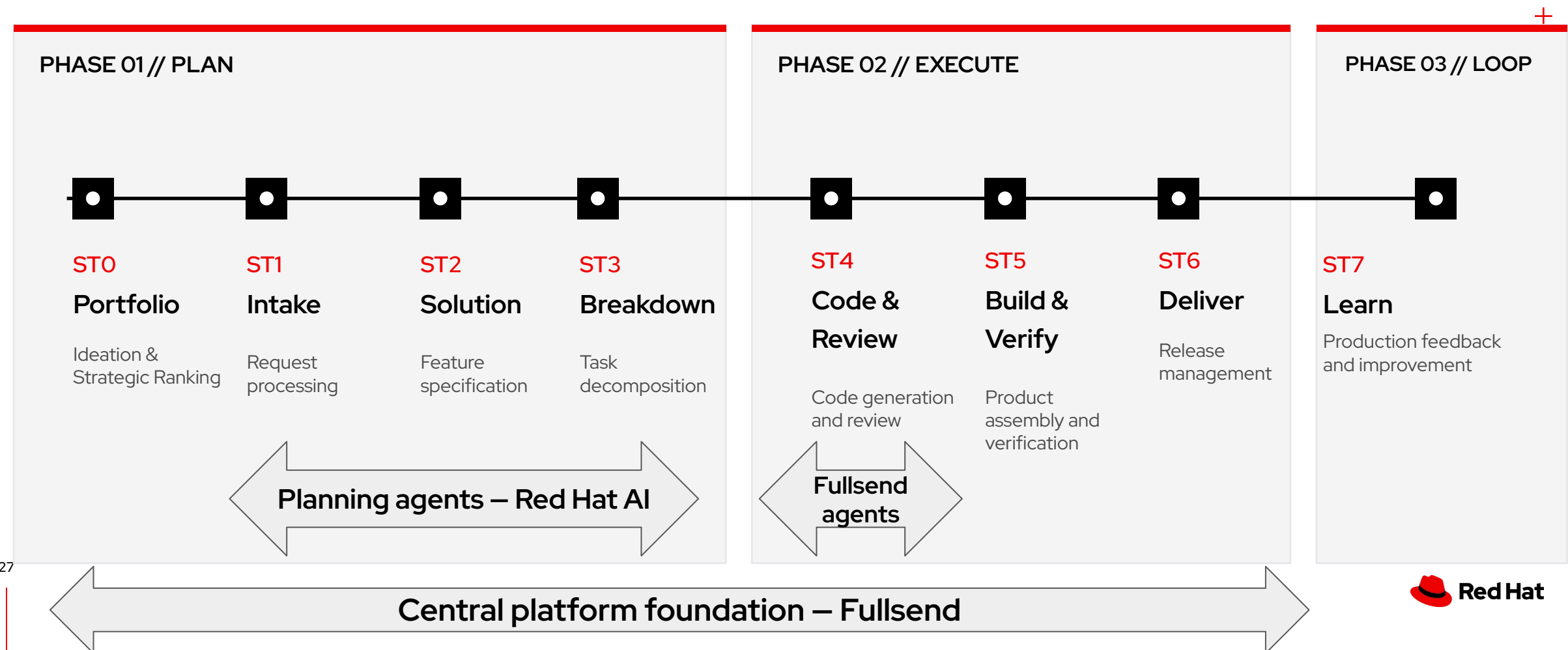
»»»» Increasing agentic capability and autonomy »»»»



Central Agentic SDLC: Eight stations across the software lifecycle



Combining the best from our pilots



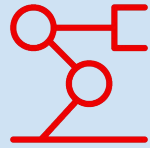


Fullsend Agents

Pre configured agents for common development tasks

Triage

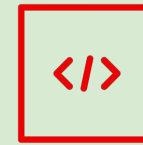
Issues → Issues



Inspect issues, asks questions, applies labels

Code

Issues → PRs



Implements bug fixes and features, opens PRs

Review

PRs → PRs



Code review for correctness, security, and style

Prioritize

Schedule → Issues



RICE scores for consistent backlog ranking

Fix

PRs → PRs



Implements review feedback, pushes fixes to existing PRs

Retro

PRs → Issues



Analyzes completed workflows, proposes improvements

Execution phase: ST4 – Code & Review



Code production on git repositories

Agents take the issue and repository context, produce a PR, and iterate until it is mergeable.



fullsend-ai-coder[bot] BOT opened a pull request 1min ago

#848 feat: add sliding-window rate limiter
wrote `middleware/ratelimit.go` + `ratelimit_test.go`



fullsend-ai-review[bot] BOT reviewed PR #848 45 sec ago

Reviewed across correctness, security, performance, and intent. 1 finding: potential race condition in `CreateBranch` under concurrent requests.



fullsend-ai-coder[bot] BOT pushed a fix 30 sec ago

Fix agent — iteration 1. Fixed race-condition finding from review: added retry with backoff after `CreateBranch`. Tests passed.

Adoption

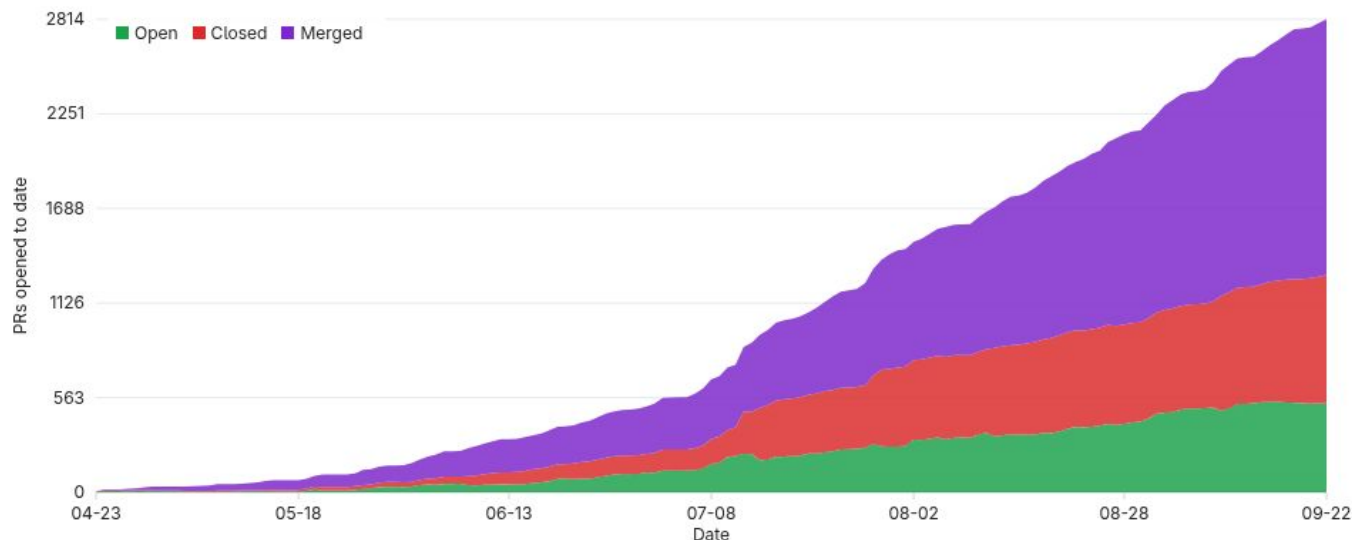
Deployed to 126 repositories

Across 30 organizations.

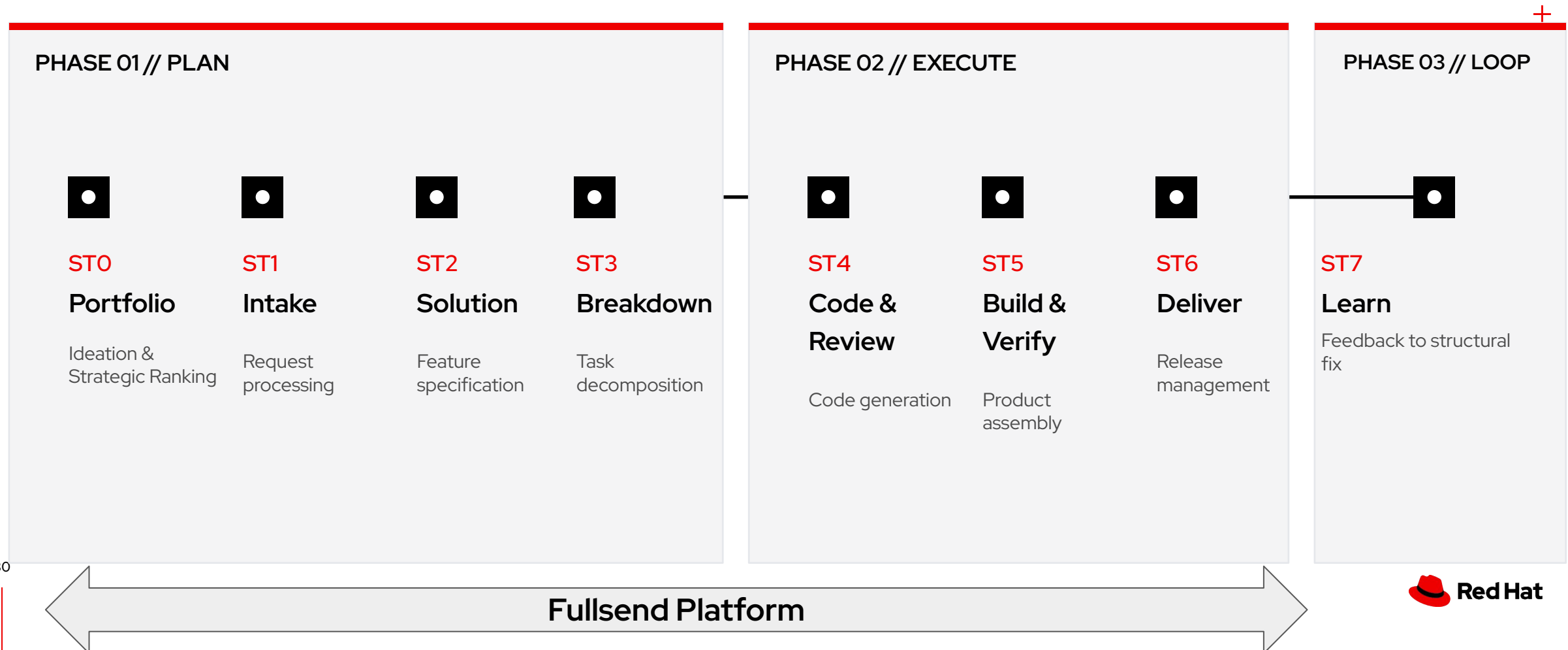
Throughput

2810 pull requests opened

1520 PRs merged, 529 currently open



Central Agentic SDLC: Fullsend Platform



Fullsend Run – A layered model



Running an agent

Outside -> inside. Control flows down; the model is least privileged.

Dispatcher Filters forge events, selects agent role · fullsend admin install

Compute Allocates the environment for the agent run

fullsend run Host runner · load harness, create sandbox, pre & post scripts

Agent harness e.g. code-agent.md, skills, tools, context – configures what goes in

Sandbox Ephemeral isolation – network and filesystem – constrains the runtime

Agent runtime LLM tool-use loop, instrumented · OTLP · runs inside the sandbox

LLM Reasoning and code generation – center, least privilege

Lightwell Roadmap – October Update at a Glance

	October	November	December	Future
NETWORK	Publish CVE remediation data to osv.dev Lens – CVE severity details & PDF export Network membership digests Network API (content queries & disclosures) Renovate integration	VEX support for scanner feeds & SBOM User access control via console.redhat.com Network API – security & CVE resource Network Skill Automated patch verification Active SLO – validated packages within 24h	Network CLI Controls for gating & batching updates Audit Evidence Record Configurable human-in-the-loop gate SLO for Moderate+ CVEs	Forward scanning for novel vulnerabilities (scope & frequency under review)
CLEARINGHOUSE	Direct API vulnerability submission Improved CVSS scoring Beacon status improvements	Beacon package-level remediation status Model & harness scanning receipts Improved submission scoring		Priority queue recommendations Custom usage & remediation reports Dev Spaces for Security TAMs
LANGUAGE ECOSYSTEMS	Python pre-disclosure	JavaScript introduction	Expanding JavaScript frameworks	.NET (January) Go, Rust, & more Expanding beyond x86 & Linux
ECOSYSTEM	PANW virtual patching JFrog catalog integration Trusted Profile Analyzer OSV integration	Sonatype integration Snyk/Semgrep consuming security feed SonarQube		



Let's get you started

Trusted software supply chain assessment

Full implementation in 6-12 weeks

[Get started](#)



Developer guide

Supply chain security ▶



Engagements

Red Hat Services for
Application Security ▶



Overview

Red Hat Advanced
Developer Suite ▶

