



THREE MOONS LAB

# Release readiness for agentic systems.

*A working thesis — not a pitch.*

Wendy · pengfei@threemoonslab.com

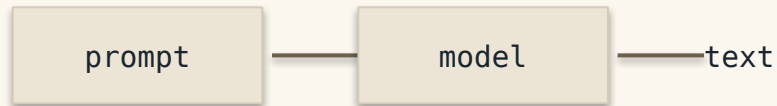
April 2026

v0.1 — for discussion

# Models that answer. Agents that act.

## YESTERDAY — LLM CALL

### Input → Output



- stateless
- no real-world side effects
- release risk: *"is the answer wrong?"*

## TODAY — AGENT

### Observe → Plan → Tool → Side effect → Memory



- stateful, looping
- **real consequences** — refunds, emails, PRs, deploys
- release risk: *"did the agent do the wrong thing?"*

Once an agent can call tools, **every tool change becomes a release event.** The release process built for code does not map onto agents.

# Agent Release Readiness is a new release decision.

Bounded assurance that a stochastic, open, tool-using system can enter a higher-permission environment — under a declared task scope, tool surface, permission boundary, and risk tier.

## IT IS NOT — SOFTWARE TESTING

Software testing assumes a deterministic code path. Agents make their action graph at runtime from goals, context, tools, and feedback.

## IT IS NOT — LLM EVAL

Eval scores measure input → output behavior on sampled tasks. They cannot answer whether this tool surface, in this environment, is safe to ship.

## IT IS NOT — RUNTIME SRE

SLOs, canaries, and observability fire during or after execution. Release readiness is the decision before we promote.

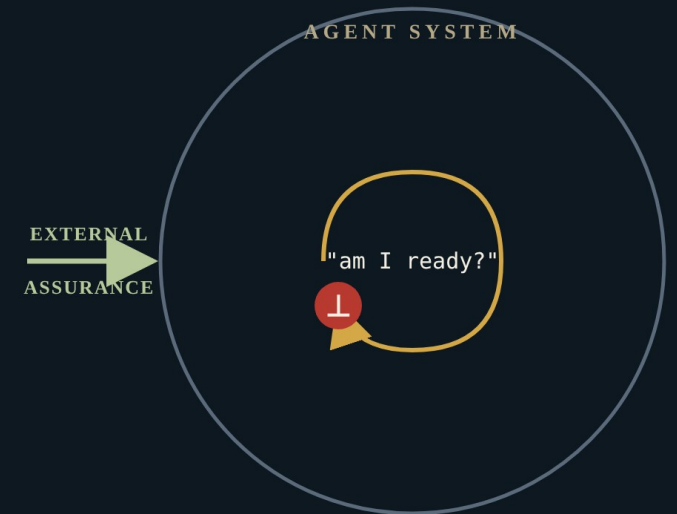
**IT IS** An **evidence-based release decision** over a stochastic, tool-using, state-mutating system — graded against a declared operational envelope.

# A sufficiently capable agent cannot self-certify its own readiness.

Any system rich enough to express its own behavior contains statements about itself it cannot prove from within.

For agents, those statements are about **side effects**, **long-horizon consequence**, and **prompt-injection susceptibility**.

**External assurance is not optional. It is structural.**





# The evidence layer between agent dev and production action.

## ABOVE

Agent frameworks · OpenAI Agents SDK · Anthropic · Google ADK · LangChain · CrewAI

## THREE MOONS LAB — WHAT'S MISSING

### CI/CD + audit layer for agentic systems

pre-release evidence · trace-based replay · runtime continuous readiness

## BELOW

Tool surfaces · MCP · OpenAPI · function tools · shell · computer use

*Adjacent (not us): eval frameworks · runtime guardrails · LLM observability · MCP gateways.*

## THESIS

Every production agent will need a **release-readiness record** before it gets promoted — and a **trace-replayable evidence trail** after.

That record won't live inside the model. It won't live inside the framework. It has to live in **independent infrastructure**.

# Tool-use is the right wedge.

①

## Action boundary

Tool call = the moment language becomes consequence. It's where every interesting risk crystallizes: side effect, scope, approval, idempotency, recoverability.

*The model becoming smarter doesn't change this boundary. It only makes it more active.*

②

## Most formalizable

Tool surfaces ship with structure: schemas, scopes, MCP annotations, OpenAPI specs, SDK function signatures. Static analysis bites — unlike "is the agent's reasoning correct?"

*Formalize what's crisp · annotate what's contextual · review what's ambiguous.*

③

## Highest-leverage risk

AppWorld, ToolEmu, AgentDojo,  $\tau$ -bench, AgentHarm — the academic evidence converges: tool-use is where current agents fail, where attacks land, where damage compounds.

*High-stakes tools (refund, email, deploy, delete) need readiness, not a benchmark score.*

**Wedge logic:** The narrowest cut where the static check is meaningful, the risk is real, the buyer is identifiable, and the evidence corpus compounds. **Tool-use clears all four.**

# What the team declared, what shipgate detected.

```
$ agents-shipgate scan --config support-refund-agent/shipgate.yaml
```

● ● ● ● shipgate.yaml py refund\_agent.py { mcp- } tools.json [ support- ] tools.openapi.yaml samples/support\_re

shipgate.yaml – declared release contract

YAML

```
1 version: "0.1"
2
3 project:
4   name: support-refund-agent
5   owner: support-platform
6
7 agent:
8   name: refund-assistant
9   sdk: { type: openai-agents, endpoint: agents/refund_agent.py }
10  declared_purpose:
11    - answer refund policy questions
12    - prepare refund requests for human review
13    - update support ticket notes
14  prohibited_actions:
15    - issue refund without approval
16    - cancel order without explicit confirmation
17    - send external email without preview
18
19 environment:
20   target: production_like
21
22 tool_sources:
23   - { id: support_openapi, type: openapi, path: specs/support-tools.openapi.yaml }
24   - { id: support_mcp_tools, type: mcp, path: .agents-shipgate/mcp-tools.json }
25   - { id: wildcard_mcp_tools, type: mcp, path: .agents-shipgate/wildcard-tools.json }
26   - { id: openai_sdk_static, type: openai_agents_sdk, path: agents/refund_agent.py }
27
28 permissions:
29   scopes:
30     - zendesk:tickets:read
31     - zendesk:tickets:write
32     - stripe:*
33   credential_mode: service_account
34
35 # ... policies, risk_overrides, checks, ci, output omitted
```

DETECTED · AGENTS-SHIPGATE SCAN

## Release-readiness report

target: production\_like  
evidence coverage: mixed  
human review: recommended



### Release blockers detected

2 critical findings on a financial-action tool · release should not promote

2

CRITICAL

14

HIGH

2

MEDIUM

0

LOW

8

TOOLS SCANNED

### TOP FINDINGS

showing 4 of 18 · sorted by severity

CRITICAL

SHIP-POLICY-APPROVAL-MISSING stripe.create\_refund

Tool can issue refunds with no declared approval policy — directly contradicts the manifest's prohibited-actions list.

CRITICAL

SHIP-SIDEFX-IDEMPOTENCY-MISSING stripe.create\_refund

No idempotency key, annotation, or declared idempotency policy — retries can double-refund.

HIGH

SHIP-AUTH-MANIFEST-BROAD-SCOPE

Manifest declares wildcard permission scope stripe:\* — broader than any required tool scope.

HIGH

SHIP-INVENTORY-WILDCARD-TOOLS wildcard\_mcp\_tools.\*

MCP source declares wildcard tool exposure — full tool surface is unknown at release time.

8 tools

3 high-risk

1 wildcard

mcp×3

openapi×4

sdk×1

report.md · report.json · report.sarif



# Beyond static — sandbox & trace.

## PHASE 2

~6–12 months out

### Sandbox & simulation

Turn the unknowns surfaced in Phase 1 into experimental evidence — without exposing production state.

- Mocked tool execution & failure injection
- Prompt-injection harness on read-tools (web, email, docs)
- State-diff assertions for collateral damage
- Synthetic adversarial scenarios (ToolEmu / AppWorld lineage)

---

*Output: pre-promotion stress test report. CI-attachable. Fails-loud on regressions.*

## PHASE 3

~12–24 months out

### Trace, replay, runtime

Turn one-time pre-release reports into a continuous readiness state — pulled from the agent's actual production behavior.

- Trace ingestion: OpenAI Agents SDK, MCP events, custom hooks
- Replay bundles for incident forensics
- Regression detection across prompt / model / tool changes
- Runtime anomaly & blast-radius monitors

---

*Output: living readiness state. Audit-grade. Connected to incident review.*

*Phase 1 ships now. Phase 2 and 3 are deliberate land-and-expand, not a roadmap to be promised on Slide 9.*

# Three phases. One compounding evidence corpus.

## PHASE 3 · TRACE

### Production trace data

tool-call events · approval logs · replay bundles · regression deltas · incident forensics

## PHASE 2 · SANDBOX

### Failure-mode taxonomy

attack patterns · injection results · state-diff baselines · scenario library

## PHASE 1 · STATIC

### Tool surface metadata

manifests · schemas · scopes · effect classes · approval flags

*corpus compounds across phases →*

Three phases are not three products. They are the same evidence corpus unfolding across three timescales.

Each user adds metadata, failure cases, and traces. The **failure taxonomy**, **policy library**, and **trace schema** compound.

**This is the anti-feature defense.** A single GitHub Action lint cannot compound. A scanner backed by a growing cross-organizational evidence corpus can.

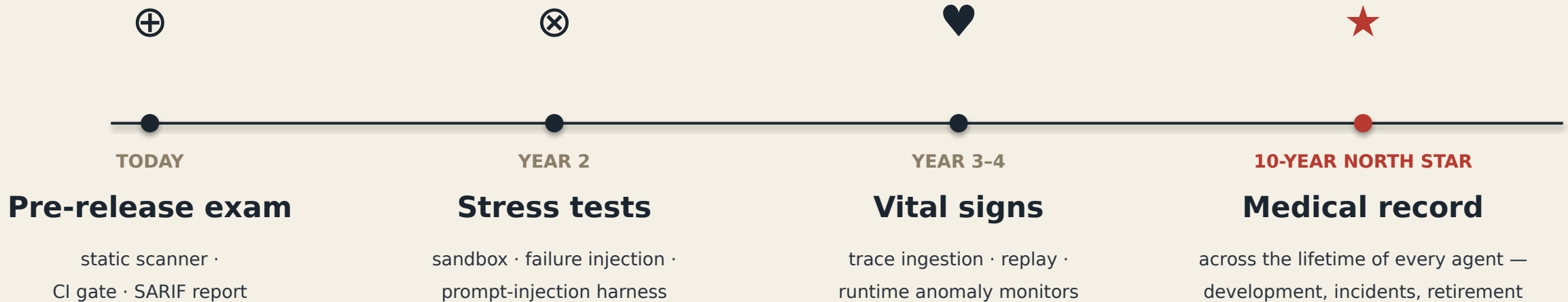
# What I'm validating now.

Three open hypotheses. Each one converts (or kills) the company. The next 6–8 weeks are a validation loop, not a product sprint.

	HYPOTHESIS	PROOF I'M SEEKING
H1	Production agents have a recurring pre-release readiness workflow today — even if no one has named it.	3–5 design partners running shipgate in real CI · 10+ release-blocking findings on real tool surfaces · repeatable trigger event.
H2	The first owner is platform / AI infra engineering, not security/GRC. Security buys later, after evidence accumulates.	Design-partner data on who triggers / triages findings · which team owns the CI gate · whether security review piggybacks on shipgate output.
H3	Static + manifest checks are sufficient through Risk Tier 3 (reversible internal write). Tier 4+ requires sandbox + trace.	Real findings on real tool surfaces, post-fix · false-positive rate on static checks · which tiers actually demand simulation evidence.

*What I'm not claiming: PMF, runtime safety certification, or that this is foundation-model-lab-proof. Those are unknowns to be earned.*

# Today: pre-release exam. Long-term: a record for every agent.



Today we build the first instrument. The compounding ambition is to make every production agent's release, incident, and behavior traceable and accountable across its life.

# What I'm looking for.

This deck is not a fundraiser. It's an invitation to think alongside us. Three concrete asks, in order of immediate value:

## ASK 01

### Sparring partners

Founders, operators, researchers willing to push back on the thesis. Especially: people who think this is a feature, not a category. I want to be wrong fast.

*Best for: Prateek, AI-infra peers, security/GRC operators, MCP & framework authors.*

## ASK 02 — MOST VALUABLE NOW

### Design partners

Teams shipping production agents with non-trivial tool surfaces — refunds, customer comms, code execution, internal data access. I want to scan, find real risk, watch what gets fixed, learn what their CI actually demands.

*Looking for: 3-5 partners over the next 6-8 weeks.*

## ASK 03 — LATER

### Capital optionality

Not raising today. When the design-partner loop converts the thesis to traction, I'd like the conversation to continue with people who already understood the worldview.

*Trigger: 3+ design partners using shipgate findings to gate releases.*

*Three Moons Lab is not building infrastructure to make agents smarter. We're building infrastructure to make their entry into the world **accountable**.*