Guide for Critical Production-Ready AI

Practical steps and assessments for controlled AI at enterprise scale.

A Practitioner's Guide for Controlled Autonomy

Table of Contents

What You Can Find in this Guide

1. Tangible Examples and Failure Patterns
2. Readiness Assessments and Implementation Checklists
3. The **Controlled Autonomy** Framework
4. Clear Path to Production AI in Critical Operations

Who This Guide is For

This guide is made for enterprise AI leaders and practitioners, looking to implement production-ready AI in the core business processes that demand full oversight and reliability assurances.

How to Use This Guide

Use this as a best practice and self assessment tool for innovation leaders to implement AI systems that can safely act at scale within set boundaries the organization controls and trusts.

The New Reality of Production-Grade AI

Enterprise AI has crossed a threshold. The conversation is no longer about whether AI can generate value. The question now is whether AI can execute safely and reliably at the core operations that drive the business forward.

This is a fundamentally different problem when thinking of AI in terms of blast radius. When an AI agent writes to a financial system or updates an compliance record, the consequences of failure are measured in financial loss and regulatory exposure.

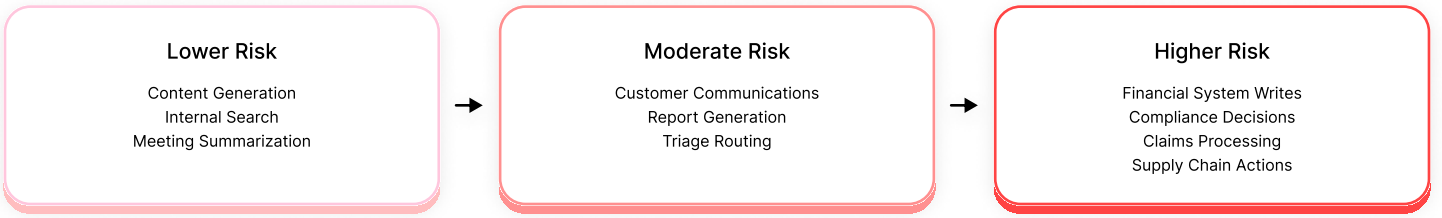
The architecture that enabled AI to draft a polite complaint email lacks the robust state management and guardrails required for an enterprise scale AI workflow that orchestrates payments, customer billings, or inputs to inventory systems. Yet the majority of **AI solutions are built on infrastructure designed for email-draft level consequences.**

Only 14%
of organizations classify their architecture as fully AI-ready, while 78% view AI as a cornerstone of future success.

State of the Enterprise AI Stack 1H 2026. HyperFRAME Research, March 2026. Survey of 544 global decision-makers.

The Critical Scope That Can Now Be Automated with Agents

The processes that comprise critical enterprise operations, are now technically addressable with AI. Processes that require synthesizing data across dozens of sources, applying complex regulatory mandates, and making decisions with real financial or legal consequences should only be safely executed with the right guardrails in place.



Criticality and risk escalates as processes move right on this spectrum - requiring fundamentally different architectural assurances.

Example Critical Use Cases:

Know Your Customer (KYC) in Banking
Requiring agents to access sensitive data across multiple systems while maintaining complete regulatory audit trails.

Invoice Processing to Financial Systems
AI agents write directly to ledgers, post entries, and update financial records. In these environments, 95% accuracy is not a success metric; it's a liability/

Compliance Permitting for Infrastructure Construction Projects
Demands ingestion of geotechnical surveys, and jurisdiction-specific legal frameworks. Lacking oversight can delay projects by months and trigger significant regulatory fines.

RFP Processing for Enterprise Sales
Requires access to proprietary information, understanding of complex business logic, and generation of responses that represent binding commitments.

These are the processes where controlled autonomy becomes essential →

The Controlled Autonomy Framework

Defining Controlled Autonomy

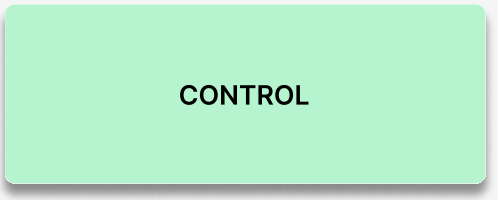
Controlled Autonomy is the architecture that ensures autonomous agents can execute consistently, only within explicitly defined operational boundaries the organization controls and trusts.

The path from pilot to production requires more than better models or bigger datasets. It requires a governing principle for how AI operates within your organization. That principle is Controlled Autonomy.

Controlled Autonomy is possible via an architectural standard, built on three interdependent pillars.

Pillar One: Control

Defines the execution boundaries.
Without explicit boundaries, agents accumulate permissions, access systems beyond their scope, and create attack surfaces that grow with every new workflow.

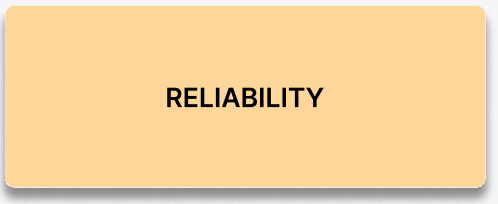


Pillar Two: Governance

Ensures those boundaries hold.
Without full observability and traceability, AI decisions become indefensible under audit, invisible to compliance teams, and impossible to debug when something goes wrong.

Pillar Three: Reliability

Guarantees consistent process execution.
Without fault tolerance, state management, and intelligent recovery, a workflow that succeeds once will eventually fail at scale, in ways that are difficult to predict and expensive to fix.



These three pillars are interdependent requirements deeply rooted within the underlying infrastructure. Control without governance creates boundaries no one can verify. Governance without reliability produces audit trails for systems that don't work consistently. Reliability without control builds a fast, scalable agent that operates outside acceptable limits.

Production-grade AI requires all three to be operating together, and made possible via an architetural foundational

The Foundation: Composable Architecture

While the pillars of controlled autonomy define the operational requirements for production-grade AI.

The **underlying architecture** determines the foundation for these to be met in real enterprise environments.

Any AI infrastructure that assumes a stable, uniform environment will break on contact with production. Technical reality is complex. Organizations operate across heterogeneous tech stacks where legacy systems, modern apps, databases and internal tools coexist. Teams adopt and retire tools on their own timelines. API schemas change without warning and compute demand fluctuates unpredictably.

The composable architecture advantage

	Abstraction Frameworks	Composable Architecture
Integration Approach	Wrapper libraries around APIs. When APIs change, the wrapper breaks and the workflow fails	Direct protocol communication (REST, gRPC, GraphQL). When APIs change, the protocol persists and only configuration updates.
Vendor Dependency	Locked to framework's supported integrations and model gardens	Connects rapidly to any system that speaks standard protocols as well as customized builds
Maintenance Burden	Framework updates, dependency conflicts, glue code, as well as engineering talent	Managed at the infrastructure layer with auto-scaling demand elasticity and processing engines embedded as part of the underlying architecture

The four properties this architecture unlocks:

1 Protocol-Level Integration

Production AI must integrate to enterprise systems at the protocol layer (REST, GraphQL, gRPC), not through abstraction frameworks that wrap APIs in layers of intermediary code. Protocol-level integration lets agents speak your systems native language, reducing failure surface area.

2 Vendor Agnosticity

Business logic should connect to any model, API, or data source without creating hard service dependencies. For each new model, vendor price change, or a compliance requirement the tool switch should happen at the configuration level. Not as a re-architecture project.

3 Elastic Scalability

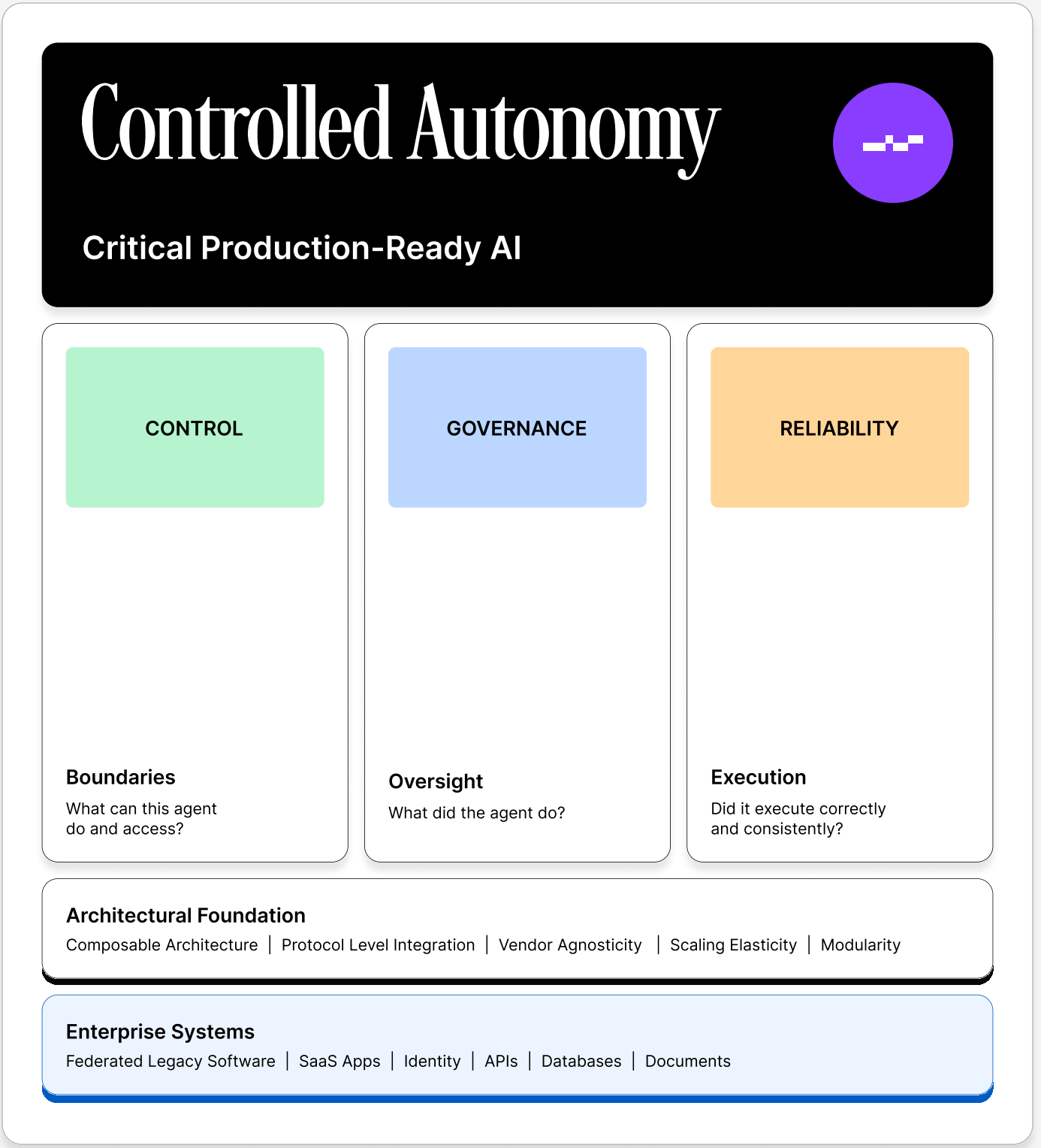
Production AI must scale horizontally with auto-scaling compute and data layers that adjust to fluctuating demand without manual intervention or workflow redesign. Infrastructure that works at pilot scale and collapses under production load is not production infrastructure.

4 Component Modularity

Every element of a workflow, from a data connector to a validation step to a human review gate, should function as an independent, reusable module. This allows swapping components without rebuilding entire workflows.

These four properties are the practical requirements we believe an AI orchestration infrastructure requires to operate within real enterprise environments. We recommend treating them as foundational criteria when evaluating platforms or designing internal solutions.

With this foundation in mind, let's examine the controlled autonomy framework →



The Pillars and Foundation that Enable Controlled Autonomy for Critical Enterprise Operations
Allowing AI agents to consistently run but only within defined and trusted boundaries.

The Pillars

Pillar One: Control

What can this agent access and do?

Control defines the boundaries within which AI agents operate.

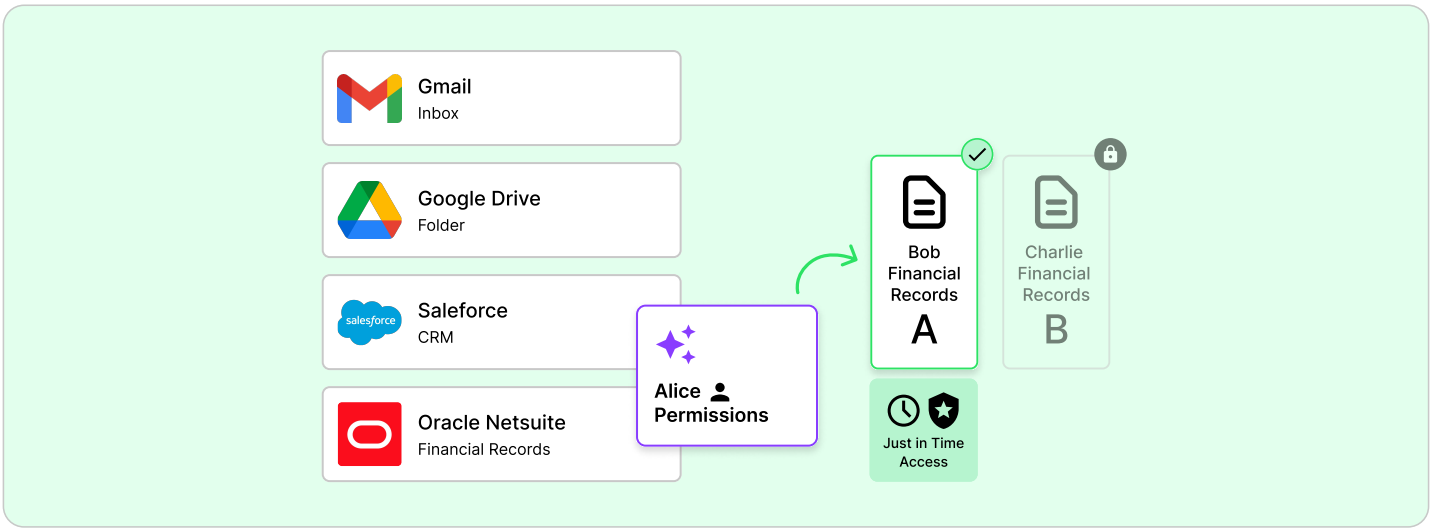
It ensures that agents only possess the permissions necessary to complete their tasks. No more, no less.

The Failure Pattern

Consider a health insurance claims workflow. An agent is tasked with processing a reimbursement for a member on a family plan. It queries the plan record to verify the member's eligibility and match the procedure code to the policy. **Without explicit access boundaries**, it pulls medical records for other family members to check for related claims, accessing protected health information for individuals who are not party to this claim.

The agent did nothing malicious. It followed a logical path to be thorough. This is what makes unbounded agents dangerous. Without defined boundaries, agents follow the path of least resistance to complete their task, accessing data and systems that were never intended to be in scope.

The Principle: Agents should operate on a least-privilege basis, with access that is scoped to the specific task and expires when the task is complete.



The Control Pillar sets defined operational boundaries for agent resource access and actions.

Architectural Requirements

Scoped, Time-Bound Access

Rather than providing agents with persistent credentials or broad permissions, access should be **granted dynamically per task, on a time-bound basis.**

Identity Management for Human and Non-Human Actors

Workflows often mix identities. An agent might access data on behalf of a user (inheriting that person's permissions) while executing other steps as a system identity.

Explicit Tool and Data Boundaries

Explicitly **defining which tools and data sources an agent can invoke** is essential to maintain Control.

Agents should not be allowed to discover and access any available resource.

Secure Connections at the Infrastructure Layer

Secure connections to applications, MCP servers, and enterprise systems should be established and managed through the infrastructure layer, not configured at the workflow level where they become fragmented.

Readiness Assessment

What Can the Agent Access?

Use these questions to evaluate your current state. If you answer "no" to three or more, Control should be a priority area of focus.

Can you define, at the platform level, exactly which systems and data sources each agent can access?	Y	N
Are agent credentials time-bound and scoped to specific tasks, or do agents hold persistent access?	Y	N
Do you have a single identity layer that governs both human users and AI agents?	Y	N
When a workflow step executes on behalf of a user, does it inherit exactly that user's permissions (no more, no less)?	Y	N
Are secure connections (OAuth, API keys, encryption) managed at the infrastructure layer, or configured within individual workflows?	Y	N
Can you produce a complete map of every system and data source each active agent has accessed in the past 30 days?	Y	N
Do you have a process for revoking agent access when a workflow is decommissioned or modified?	Y	N

Agents must operate within explicitly defined, scoped boundaries.

"NIST AI RMF recommends organizations to implement access controls, least-privilege principles, and continuous monitoring for AI systems."

NIST Cybersecurity Framework Profile for AI, December 2025.

"80% of organizations reported risky agent behaviors including unauthorized system access.
*While **only 21% claimed visibility** into agent permissions, tool usage, or data access patterns."*

The AIUC-1 Consortium with Stanford's Trustworthy AI Lab and 40+ security executives.

Implementation Checklist

→ Identity and Access Configuration

- [ ] Connect your enterprise identity provider (e.g., Okta, Microsoft Entra ID) to your orchestration platform.
- [ ] Enable bi-directional SCIM syncing to mirror existing user groups and permission structures.
- [ ] Map Role-Based Access Control (RBAC) to establish baseline system access and functional roles.
- [ ] Deploy Fine-Grained Authorization (FGA)—leveraging relationship-based (ReBAC) or attribute-based (ABAC) policies—to strictly control access to individual resources based on user context and object ownership.

→ Dynamic Identity and Scoping

- [ ] Configure execution contexts: determine which workflow steps must execute as a system identity or service accounts vs. on behalf of a specific user.
- [ ] Verify that on-behalf-of steps inherit the exact permission scopes of the initiating user to prevent privilege escalation.
- [ ] Implement time-bound access grants that expire automatically when each task completes.
- [ ] Establish a recurring review cadence for agent access logs to verify that active workflows are operating within their defined permission boundaries.

→ Secrets and Connection Management

- [ ] Remove all hardcoded API keys from workflow logic.
- [ ] Centralize authentication (OAuth, AWS Signatures, API Keys) at the platform infrastructure level using secrets management.
- [ ] Ensure per-customer encryption keys are active for data at rest.
- [ ] Establish and manage secure connections to all external systems through the infrastructure layer, not within individual workflow definitions.

Pillar Two: Governance

What did this agent do, and why?

Governance provides the visibility and oversight into AI actions and reasoning.

If Control defines where agents can operate, Governance ensures those **boundaries are maintained** and provides the transparency to verify it.

The Failure Pattern

A bank deploys an AI workflow to accelerate lending decisions. The system performs well for months: faster processing times, consistent outputs, positive feedback from the operations team. Then a regulatory examiner asks a straightforward question: "Walk me through exactly how this loan decision was made."

The team can't answer. They can show the input data and the final output, but the reasoning chain in between, which data sources were consulted, which rules were applied, where the model's judgment influenced the outcome, is not recorded in a defensible or reproducible way. The system is a black box. Not because the technology is opaque, but because the **infrastructure was never designed to capture the agent's decision process.**

The Principle: This is not an edge case. In regulated industries every AI-driven decision that touches a customer, a financial record, or a compliance obligation must be explainable and auditable. Neither customers nor regulators care if a human or agent issued the loan, **the organization is accountable for it.**

Architectural Requirements

Complete Observability

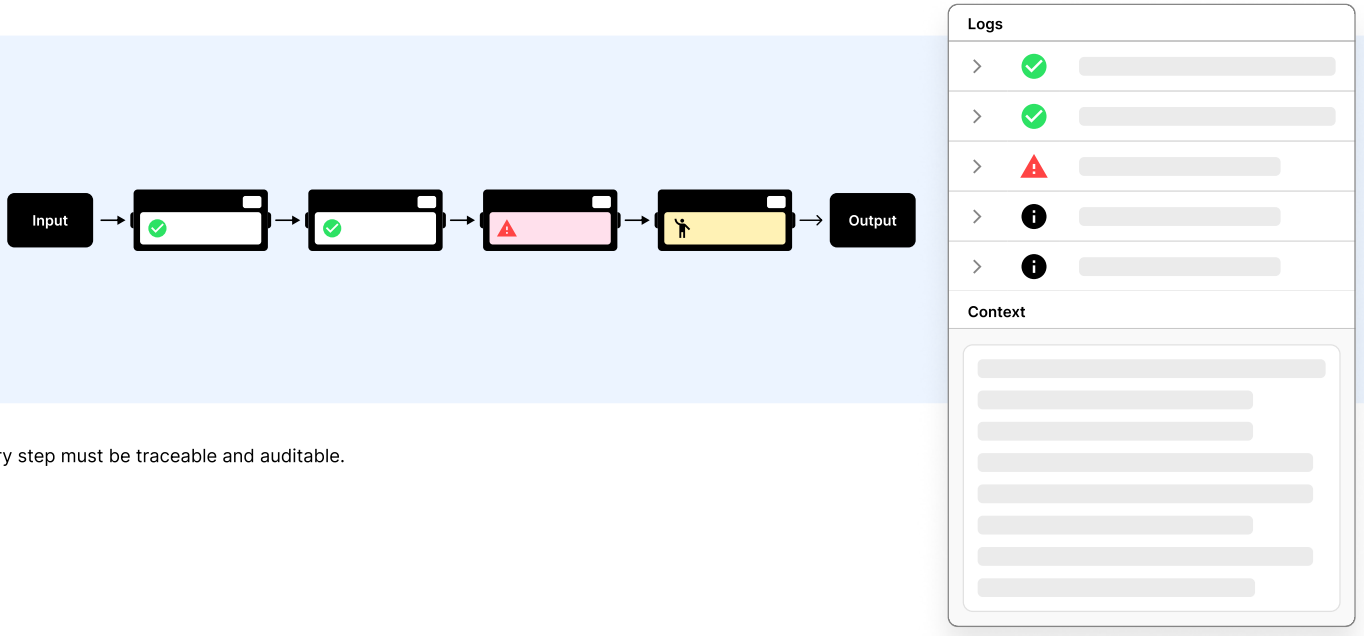
Every AI-to-system interaction should be logged with full context: what data was accessed, which systems were involved, what prompts were provided, and which decisions were made. This means capturing the intermediate reasoning and logic path at each step, not just the final output.

Process Traceability for Audit and Compliance

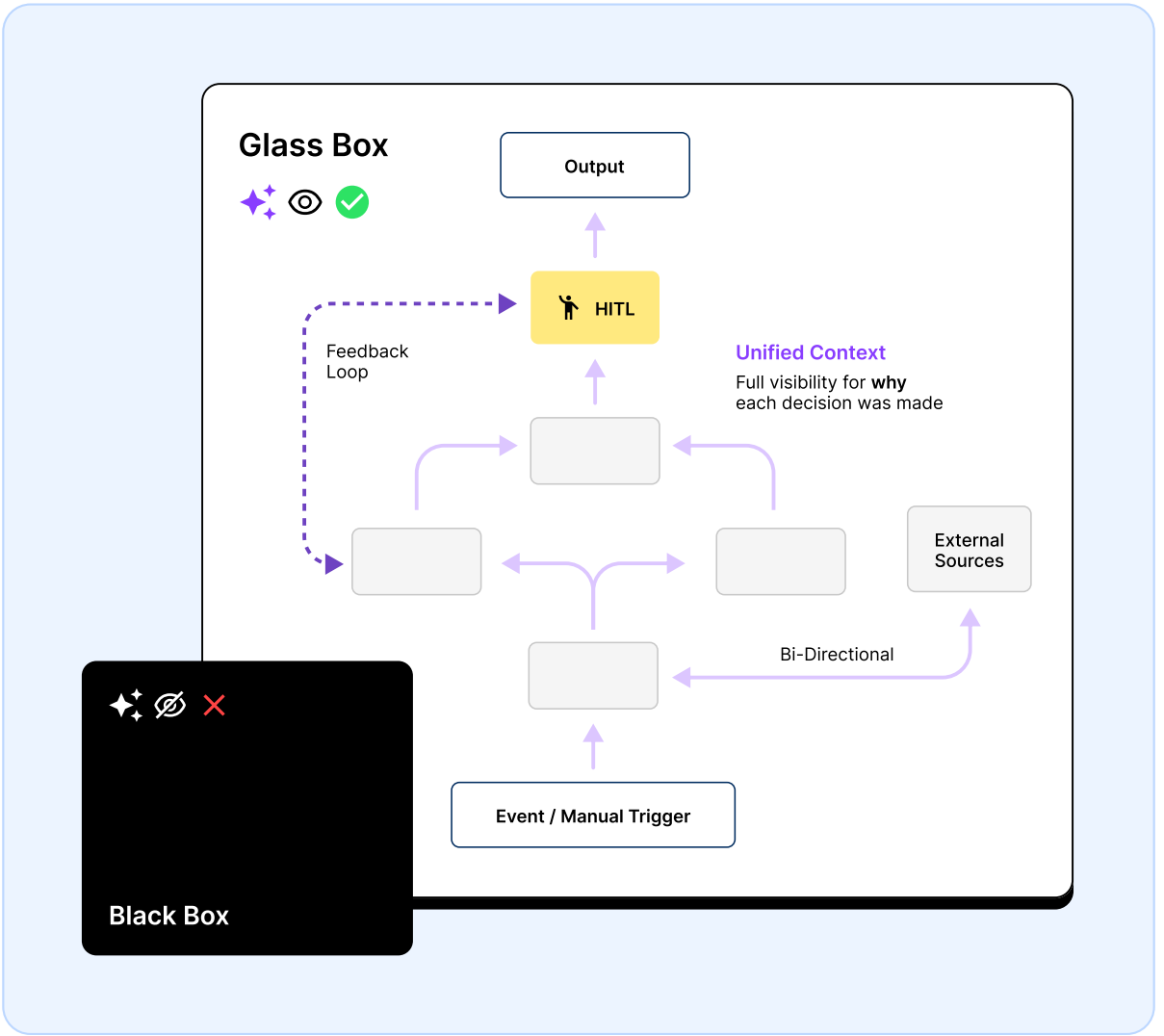
When regulators ask how a decision was made, the answer must be complete and defensible: the reasoning process, the data inputs, and the human touchpoints along the way. Designing traceability into the workflow architecture is critical for regulated industries.

Data Retention and Redaction Policies

Governance includes knowing where data lies, what to keep, and what PII must be redacted. We recommend defining data retention policies at the platform level so they apply uniformly across all workflows rather than being configured per workflow.



Every step must be traceable and auditable.



The Governance Pillar enables complete visibility into agent actions and decision-making.
Turning agentic reasoning into an observable glass box.

Human-in-the-Loop as a Core Architectural Component.

Human oversight transforms dynamic agentic behavior into auditable, deterministic processes. Human-in-the-Loop (HITL) gates pause workflows at critical decision points, surface relevant context to domain experts, and incorporate feedback or approval before proceeding.

Effective HITL design requires distinguishing between three archetypes:

1

Choose

The human selects a path. Approve or reject, route A or route B. This is appropriate for binary decisions with clear outcomes, such as payment approvals above a defined threshold.

2

Modify

The human corrects or refines logic before execution continues. This is appropriate when AI-generated outputs need domain expertise before being committed.

3

Context

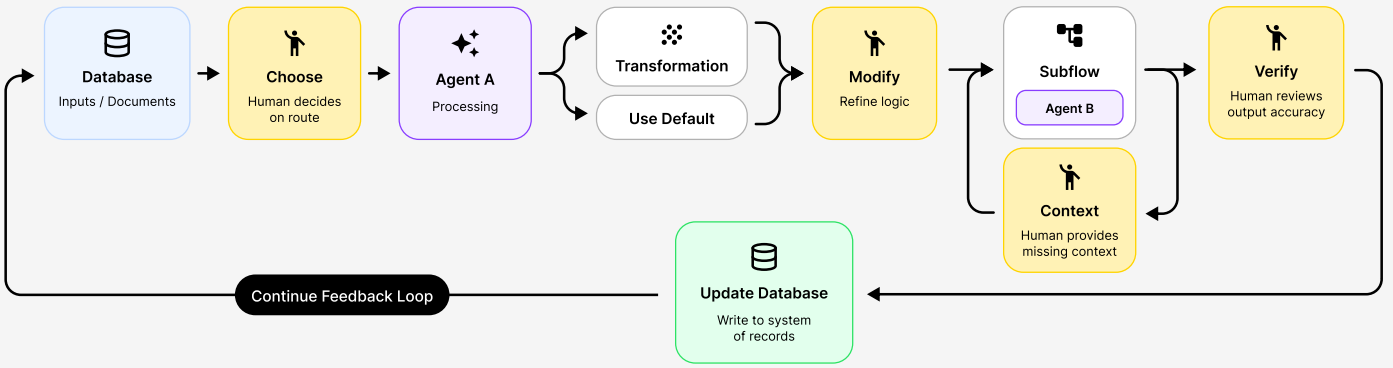
The human provides missing domain expertise that the AI cannot access or infer. This is appropriate when the workflow encounters ambiguity that requires institutional knowledge or professional judgment.

4

Verify

The human reviews and confirms the accuracy and completeness of AI-generated output before it is committed to a system of record.

We recommend implementing HITL as a native infrastructure component that leverages human input not just as a decision gate, but as a data point for continuous workflow improvement.



Readiness Assessment

What Did the Agent Do?

Use these questions to evaluate your current governance posture.
If you answer "no" to three or more, Governance should be a priority area of focus.

Can you reproduce the exact reasoning chain for any AI-driven decision made in the past 30 days, including what data was accessed, which models were invoked, and what logic was applied?	Y	N
Are your HITL gates triggered by business rules and confidence thresholds, or only by error states?	Y	N
Do your audit trails differentiate between what the AI planned (the decision) and what was executed (the action)?	Y	N
Is PII automatically redacted from logs and audit trails before storage?	Y	N
Are data retention periods defined and enforced at the platform level, and aligned to your compliance requirements?	Y	N
When a human intervenes in a workflow, is their input, rationale, and identity captured as part of the audit trail?	Y	N
Can your compliance team access workflow audit trails without requiring engineering support?	Y	N

"The EU AI Act requirements for high-risk AI systems become enforceable on August 2, 2026. Mandating risk management systems, human oversight, technical documentation, and audit trails."

EU AI Act: first regulation on artificial intelligence. European Parliament, updated February 2025.

AI governance is the strongest predictor of AI readiness.

The State of AI Security and Governance. Cloud Security Alliance, 2026.

Implementation Checklist

→ Observability Configuration

- [ ] Configure telemetry pipelines to capture full inputs, outputs, and reasoning logic (chain-of-thought) for every AI-to-system interaction.
- [ ] Implement distributed tracing to clearly differentiate between the "decision" (what the AI planned) and the "action" (the API call or system write that was executed).
- [ ] Establish observability dashboards and query interfaces that allow compliance and operations teams to inspect workflow execution without engineering involvement.

→ Audit Trail and Compliance

- [ ] Define data retention periods (e.g., 30 days, 6 months, 3 years) based on applicable compliance frameworks.
- [ ] Configure PII masking rules to redact sensitive information from logs before storage.
- [ ] Verify that the complete decision chain, from trigger through every intermediate step to final output, is captured and retrievable for any workflow execution.

→ Human-in-the-Loop Design

- [ ] Identify critical decision nodes in each workflow that require human oversight (e.g., financial writes above a defined threshold, compliance-sensitive classifications, edge cases with low AI confidence).
- [ ] Select the appropriate HITL archetype for each node: Choosing, Modifying, Verification, or Context.
- [ ] Define confidence thresholds that determine when HITL is triggered vs. when the workflow auto-proceeds. Not every execution needs human review; the gate should activate based on risk level, output confidence, or business rules.
- [ ] Configure notification channels (email, Slack, or other platforms) to alert the designated human when their input is required.
- [ ] Ensure that all human inputs, approvals, modifications, and contextual additions are logged as part of the workflow's audit trail.
- [ ] Define response SLAs for each HITL gate and configure escalation rules when thresholds are exceeded (e.g., reassign to a designated reviewer, or trigger an incident management workflow).

Pillar Three: Reliability

Did it execute correctly, and will it do so consistently?

Reliability ensures that workflows execute correctly, even when individual components fail.

For critical processes where failures has real consequences, every workflow must **execute correctly at scale, every time.**

The Failure Pattern

A payments team automates vendor invoice processing. The workflow extracts invoice data, validates it against purchase orders, posts the entry to the ledger, and then issues payment. During a routine batch run, the ledger entry posts successfully, but the payment step times out due to a transient network issue.

The system retries. Because there is no state management at the individual step level, the retry re-executes the entire workflow. The ledger entry is posted a second time, and a **duplicate payment request is issued.** Neither duplicate is detected until month-end reconciliation, where it surfaces as a discrepancy requiring manual investigation across multiple systems.

The Principle: The issue is not that the workflow failed. Failures are inevitable in distributed systems. The issue is that the recovery mechanism created a worse outcome than the original failure. In financial, compliance, and operational systems, this kind of silent error is more dangerous than a visible crash.

Without State Management

Step 1 ✓ → Step 2 ✓ → Step 3 ✓ → Step 4 × Failure

↓

Re-executes Workflow from Step 1:

Step 1 → Step 2 → Step 3 → Step 4 ✓ Complete

With State Management

Step 1 ✓ → Step 2 ✓ → Step 3 ✓ → Step 4 × Failure

↓

State Preserved: Steps 1 – 3 ✓ Marked Complete

Resume Workflow at Step 4 ✓ Complete

Intelligent state management ensures a workflow is executed only once, correctly.

Architectural Requirements

Fault Tolerance Through Intelligent State Management

When a workflow fails mid-execution, the system should know exactly which step completed, which step failed, and resume from the point of failure.

Each action must execute exactly once, preventing duplicate writes to financial, compliance, or operational systems. This precision, often called **exactly-once execution**, protects data integrity and prevents cascading errors when autonomous systems interact with sensitive systems.

Intelligent Retry Policies

Transient faults like network latency, API rate limits, or brief service degradation must not trigger cascading workflow failures. Implement automated retries utilizing exponential backoff and jitter to absorb these errors without overwhelming degraded downstream services.

Crucially, pair these policies with robust telemetry and execution checkpointing, enabling operations teams to inspect distributed traces and deterministically resume workflows from the exact point of failure when automated retries are exhausted.

Consistent Execution Across Runs

A workflow that produces correct results once must produce correct results consistently. This requires output validation guardrails and state isolation between executions, ensuring no data leaks between runs.

These guardrails are elementary when updating systems of record that have strict output schemas. Without them results become neither predictable nor auditable, making workflows unusable for production operations.

Readiness Assessment

Does it Consistently Execute Correctly?

Use these questions to evaluate your reliability posture.
If you answer "no" to three or more, Reliability should be a priority area of focus.

If a workflow fails at step 7 of 12, does your infrastructure resume from step 7 or restart from step 1?	Y	N
Are retry policies configured with backoff strategies, or do failed steps retry immediately and indefinitely?	Y	N
Is state fully isolated between concurrent workflow executions, ensuring no data leakage between runs?	Y	N
Can your infrastructure handle a significant increase in volume (e.g., end-of-month processing spikes) without manual intervention or workflow redesign?	Y	N
Do you have clear error reporting that identifies exactly which step failed, why, and what data was in scope at the time of failure?	Y	N
Can you replay a failed workflow from its point of failure rather than re-running the entire process?	Y	N
Do you have real-time monitoring and alerting configured for workflow failures, with clear escalation paths to the responsible team?	Y	N

"Production hardening is not an add-on but must be designed as foundational interlocking architectural layers."

"Agentic systems possess a dynamic, generative core that expands the potential blast radius of failures, where a single hallucination can cascade into an incorrect database write."

From Prompt-Response to Goal-Directed Systems: The Evolution of Agentic AI Software Architecture. Alenezi, February 2026.

Implementation Checklist

→ State Management

- [ ] Enable exactly-once execution logic to prevent duplicate transactions during retries. This is critical for any workflow that writes to financial, compliance, or operational systems.
- [ ] Verify state isolation between concurrent workflow runs to ensure no memory or context leakage across executions.
- [ ] Confirm that each completed step is recorded durably, so recovery always resumes from the correct position.

→ Resilience and Recovery

- [ ] Define intelligent retry policies with backoff strategies for transient failures (e.g., 429 rate limits, 503 service unavailable responses).
- [ ] Configure timeout thresholds for long-running or expensive actions to prevent runaway costs and hung workflows.
- [ ] Ensure failed workflows can be replayed from their point of failure with full visibility into the execution state at the time of the error.

→ Validation

- [ ] Verify that integration points are resilient to upstream API changes, whether through protocol-level connections or versioned interfaces, and do not introduce single points of failure.
- [ ] For legacy systems without modern APIs, confirm that fallback integration methods (SFTP, browser automation, or similar) are configured and tested.
- [ ] Test horizontal auto-scaling under realistic burst conditions to confirm the infrastructure handles production-level demand without degradation or manual intervention.

The Strategic Outcome: Compounding Value

Why the Right Architecture Pays Off Exponentially →

The three pillars and the composable foundation ensures these requirements hold in real enterprise environments. Together, they solve the immediate deployment challenge. But there is a longer-term outcome that separates platform-level investment from project-level builds: **the compounding value effect.**

From Standalone Projects to Organizational Assets

In most organizations, each AI initiative starts from zero.

Two teams can build similar component workflows without being aware of each other's work, making the organization's AI investments grow linearly: the tenth workflow costs roughly the same as the first.

Treating components like reusable modules changes this.

It starts with organizational discoverability and connectivity, creating a shared centralized workflow and agent registry.

Truly operationalizing it, requires infrastructure-level capabilities, like dynamic authentication, **making each workflow executable under different user or system identities.**

Creating two compounding effects:

First, each new workflow draws from a growing library of **production-proven components**, delivering value faster and at lower cost.

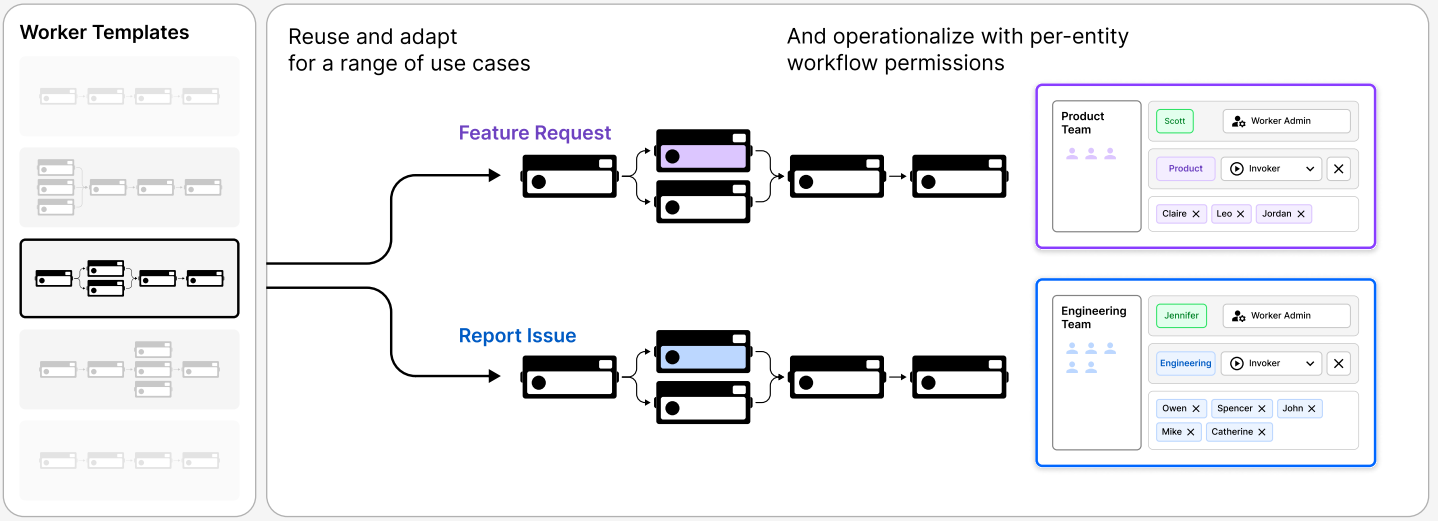
Second, **components can be refined to embed unique business logic**, whether it's through human feedback or via proprietary logic through context.

These effects not only accelerate value, but improve outcomes over time.

Reframing the Build vs. Buy Calculation

Custom solutions can address the initial use case effectively, but when each additional solution requires its own integration, reliability engineering, and governance configuration. **Engineering teams will increasingly spend time maintaining infrastructure rather than delivering outcomes.**

A platform approach inverts this. Infrastructure maintenance is centralized. Governance, traceability and reliability apply across all workflows. **Every component built for one use case contributes to the library available for the next.** Organizations should factor not just the cost of the first workflow, but the projected cost of the fifth and tenth and their maintenance.



Getting Started

Implementing the full controlled autonomy framework is a significant undertaking, but it does not need to happen all at once. Based on patterns we have observed working with enterprise teams, we recommend a phased approach.

1 Start with One High-Impact, Bounded Workflow

Select a process that is operationally significant but well-defined: a workflow with clear inputs, outputs, and decision points. Invoice processing, document extraction, or compliance reporting are common starting points. The goal is to prove the architecture on a real problem, not to solve the hardest problem first.

2 Apply the Three-Pillar Assessment Before Building

Use the readiness assessments in this guide to evaluate your posture across Control, Governance, and Reliability for your chosen workflow. Knowing where you stand before you build prevents the most common cause of stalled initiatives: discovering architectural gaps mid-implementation.

3 Think Compoundability From Day One

Even for a single workflow, build components as reusable modules. This is where compounding value begins: not at workflow ten, but at workflow one, through deliberate design choices.

4 Expand Based on Proven Patterns

Once the first workflow reaches production with full control, governance, and reliability guarantees, use it as the reference architecture for the next.

A Tangible Path to Production

Start with Discovery

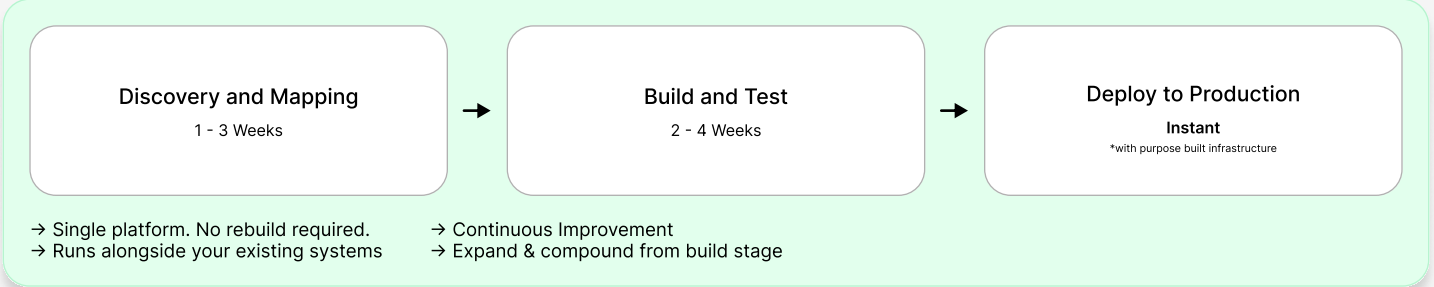
The most effective starting point is discovery-led. Pairing engineers with domain experts to map a target workflow, focusing on where manual effort, errors, and delays occur, consistently surfaces higher-impact opportunities than top-down use case selection.

Parallel Test Before any Replacement

We recommend running new AI workflows in parallel with existing systems rather than replacing them, to reduce risk and give stakeholders a direct comparison before any transition happens.

Prevent Rebuild

Lastly, prioritize a platform where experimentation and production share the same infrastructure. When a workflow proves its value, the path to production should be a configuration change, not a rebuild.



→ Single platform. No rebuild required.
→ Runs alongside your existing systems
→ Continuous Improvement
→ Expand & compound from build stage

For a clear path to implementation →

About

This guide was written by **Thread AI**, an AI infrastructure company founded by the former heads of AI product and engineering at Palantir Technologies. They identified that an infrastructure gap is preventing enterprises from deploying AI in critical core operations. That gap is what they set out to address with Lemma.

Lemma is an AI orchestration platform designed for critical enterprise operations. It enables teams to build, deploy, and manage AI workflows and agents within strict operational boundaries the organization controls and trusts.

THREAD AI

The controlled autonomy framework described in this guide reflects the architectural principles Lemma was built on.

Organizations looking to deploy AI into their business-critical processes, we welcome the opportunity to discuss how these principles apply to your specific requirements.

Learn more at www.threadai.com | Contact us at info@threadai.com | Reach out directly here

Thread AI, Inc. Built in New York.