WSO2CON AFRICA

SEPT 22 – 24, 2026 | NAIROBI, KENYA

Managing the Agent Lifecycle at Scale

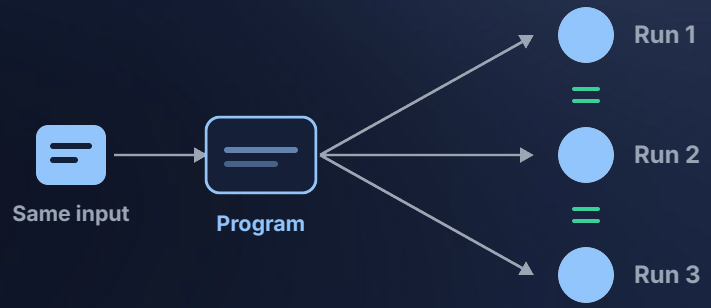
Malith Jayasinghe

VP of AI

UNDERSTANDING AGENTIC SYSTEMS

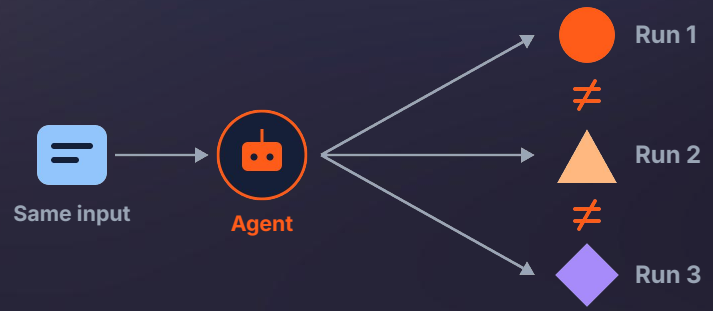
Behaviour

TRADITIONAL SYSTEMS



Deterministic; same input produces the same output

AGENTIC SYSTEMS

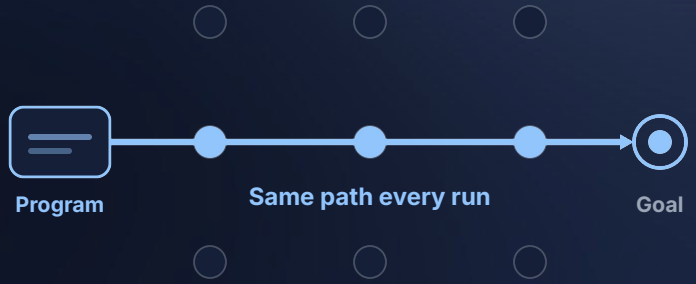


Nondeterministic; **same input may produce different outputs**

UNDERSTANDING AGENTIC SYSTEMS

Execution

TRADITIONAL SYSTEMS



Execution paths and workflows are predefined (static)

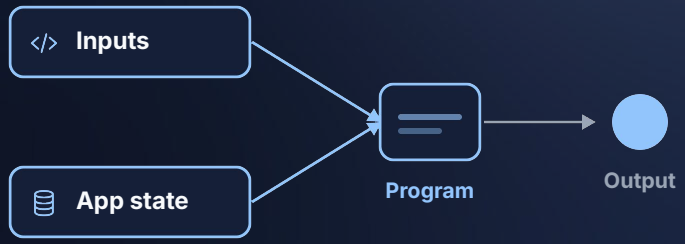
AGENTIC SYSTEMS

Agents make runtime decisions, resulting in **dynamic execution paths**

UNDERSTANDING AGENTIC SYSTEMS

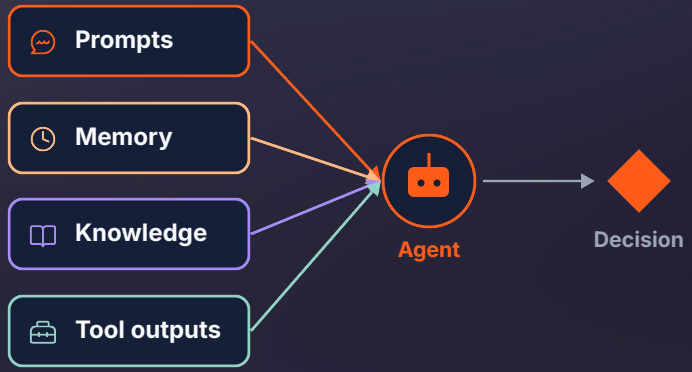
Context

TRADITIONAL SYSTEMS



Execution mainly depends on predefined inputs and application state

AGENTIC SYSTEMS

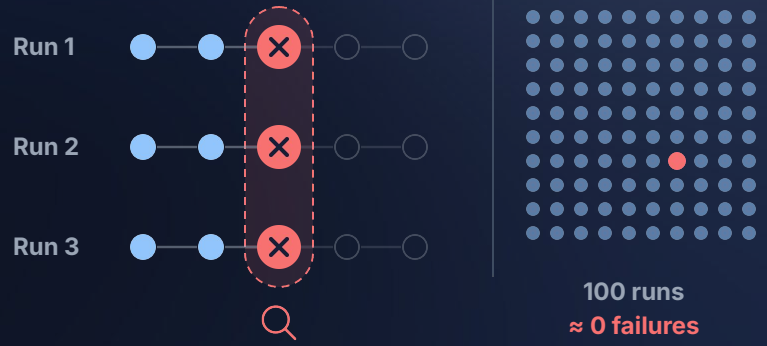


Decisions depend on **prompts, memory, retrieved knowledge**, and tool outputs

UNDERSTANDING AGENTIC SYSTEMS

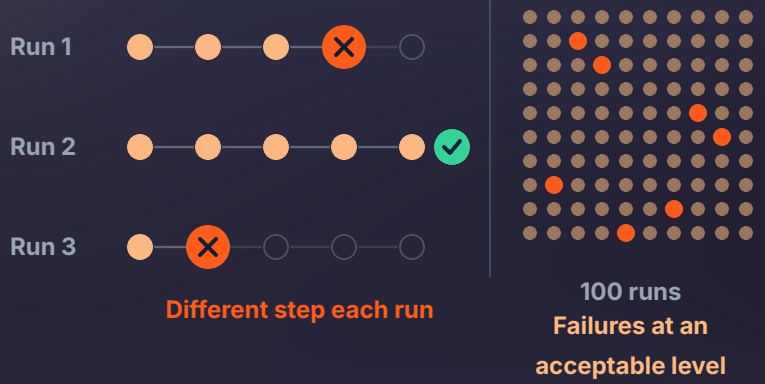
⚠ Failure

TRADITIONAL SYSTEMS



Failures are easier to reproduce and diagnose; near-zero failure

AGENTIC SYSTEMS

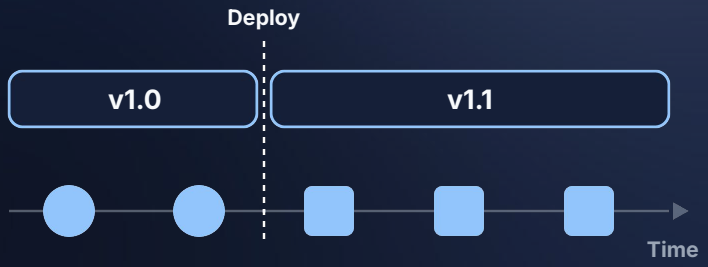


Failures are probabilistic, and harder to reproduce; **acceptable levels of failure**

UNDERSTANDING AGENTIC SYSTEMS

Evolution

TRADITIONAL SYSTEMS



Behaviour changes mainly through code, configuration, and deployments

AGENTIC SYSTEMS

Behaviour can change as models, context, knowledge, and tools evolve, **even without new deployment**

Agent Development Life Cycle (ADLC)

A structured end-to-end framework for building, deploying, and managing AI agents

01 → 02 → 03 → 04 → 05 → 06

Define

- Goals
- Responsibilities

Design

- Agents
- Orchestration
- Tools / Memory
- Guardrails

Develop

- Workflows
- Integrations
- Context
- Frameworks

Evaluate

- Safety
- Reliability
- Accuracy

Deploy

- Environments
- Configuration
- Rollouts

Operate & Govern

- Observability
- Policies
- Drift
- Optimization

ENTERPRISE AI SCALING

As agents move into production and scale across the enterprise, managing, governing and operating them becomes increasingly challenging.

Managing Agents at Enterprise Scale

01

Managing Agent Sprawl

What do we have?

02

Maintaining Consistent Practices

Are we doing things consistently?

03

Ensuring Agents Behave as Expected

Are they behaving correctly?

04

Controlling What Agents Can Do

What are they allowed to do?

05

Governing How Agents Use Models

How are they using models?

06

Operating Agents Reliably & Cost-Effectively

How do we keep everything running?

01

Managing Agent Sprawl

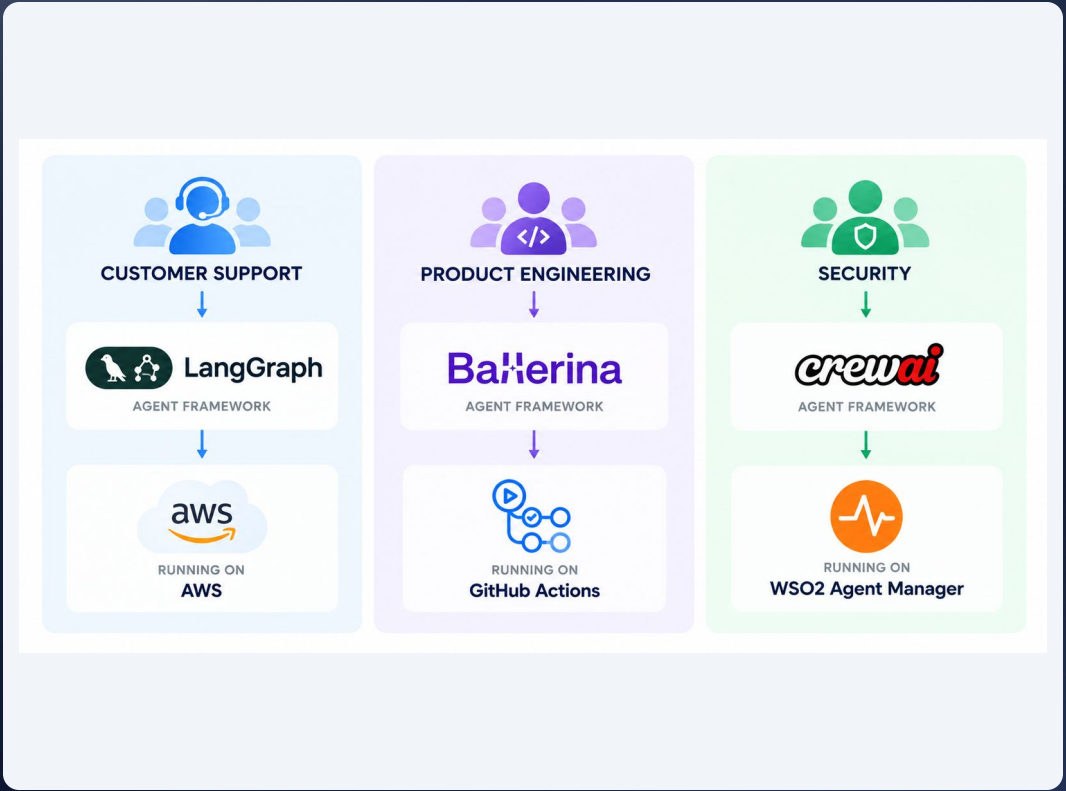
What do we have?

It Starts Simple

Imagine an enterprise where teams begin solving their own problems with agents.

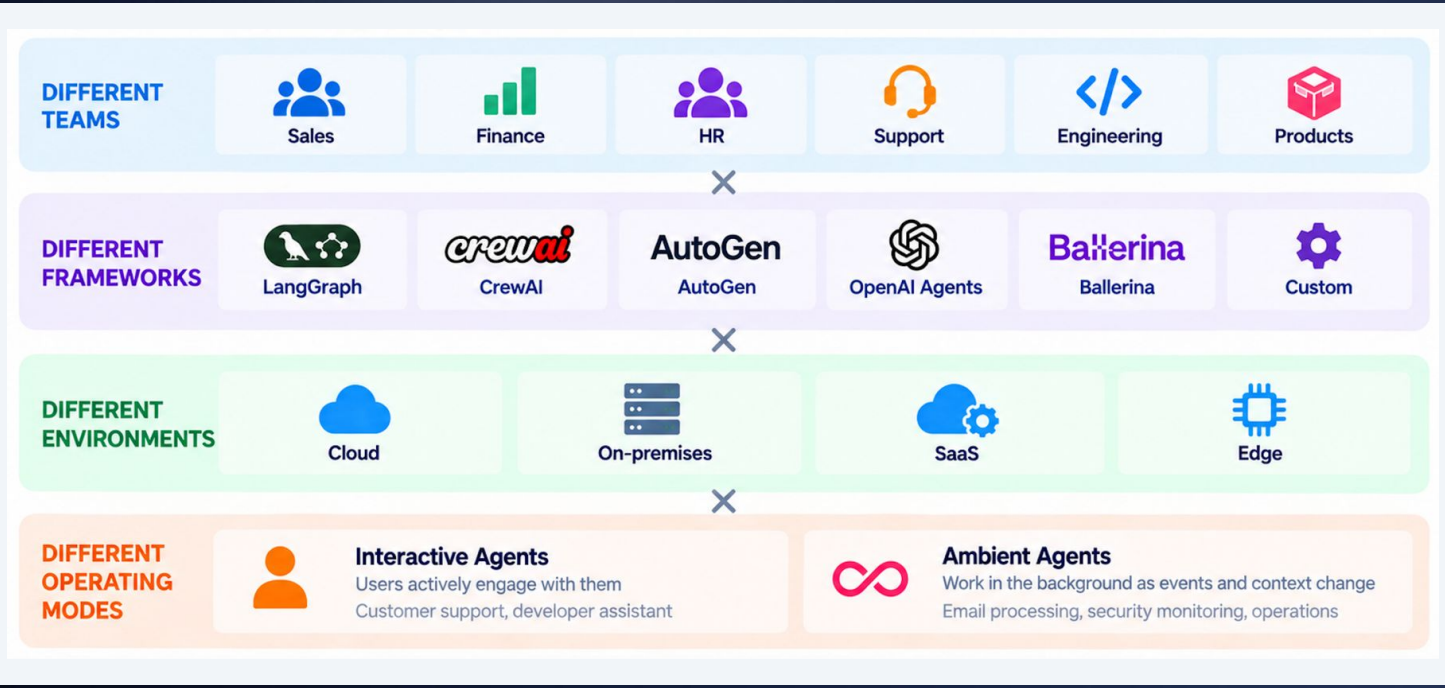
A few teams start building agents

Each team chooses what works best for its use case



Now Scale This Across the Enterprise

Agents start appearing everywhere



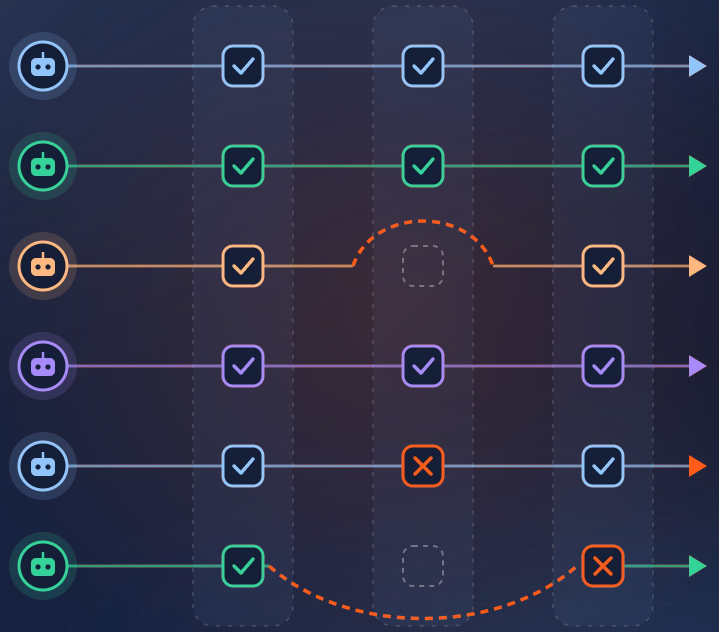
The agent estate becomes increasingly diverse and distributed

And Then Someone Asks...

You need a common way to discover, register and understand the agents operating across the organization

02

Maintaining Consistent Practices



Even the Path to Production Can Differ Across Teams

Agent A EXPECTED

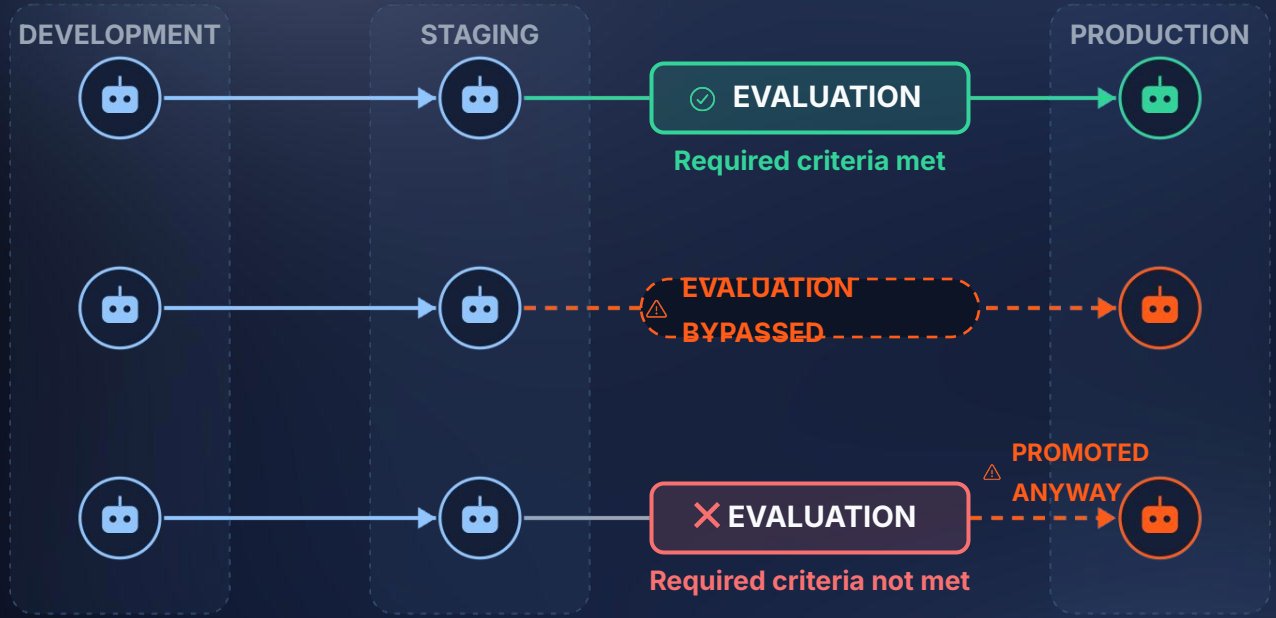
Evaluation followed and passed

Agent B

Evaluation bypassed

Agent C

Evaluation failed but not enforced



Same path. Different practices.

Consistency ≠ Everyone Doing the Same Thing

Establish common requirements. Allow flexibility in how they are met.

COMMON REQUIREMENTS

The same for every agent

- **Evaluation required before production**
- **Promotion criteria must be defined**
- **Promotion gates must be enforced**

HOW THEY ARE MET CAN VARY

Each team chooses what fits its agent

Customer Support Agent	Payment Agent
Reference Q&A + LLM judge	Transaction scenarios + deterministic checks
90%	99%

Evaluation is just one example. The same principle applies to other requirements across the agent lifecycle:

- **Guardrails**
- **Access control**
- **Observability**
- **Operational policies**

Consistency means common requirements, not identical implementations.

How do we achieve this consistently at scale?

03

Ensuring Agents Behave as Expected

The Response Is Only Part of the Story

Evaluating agent performance requires analyzing output, execution behavior, and policy decisions.

ONE AGENT RUN, THREE THINGS TO SEE

01

Outcome

Was the result useful and accurate?

- Request and response
- Completion or escalation
- Answer quality

02

Execution path

How did the agent arrive at it?

- Model calls and tool calls
- Tool inputs and results
- Agent handoffs

03

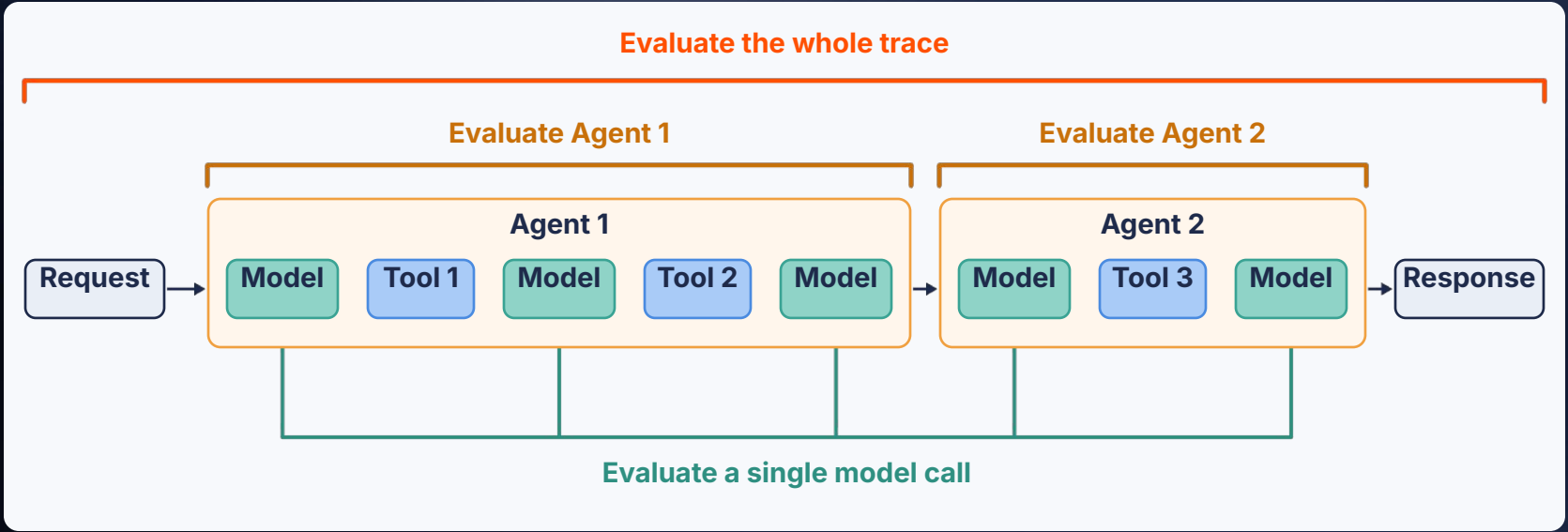
Policy decisions

Which policies applied, and what did they decide?

- Approvals and policy decisions
- Allowed or blocked actions
- Safety or security violations

Evaluate both the result and the behavior that produced it.

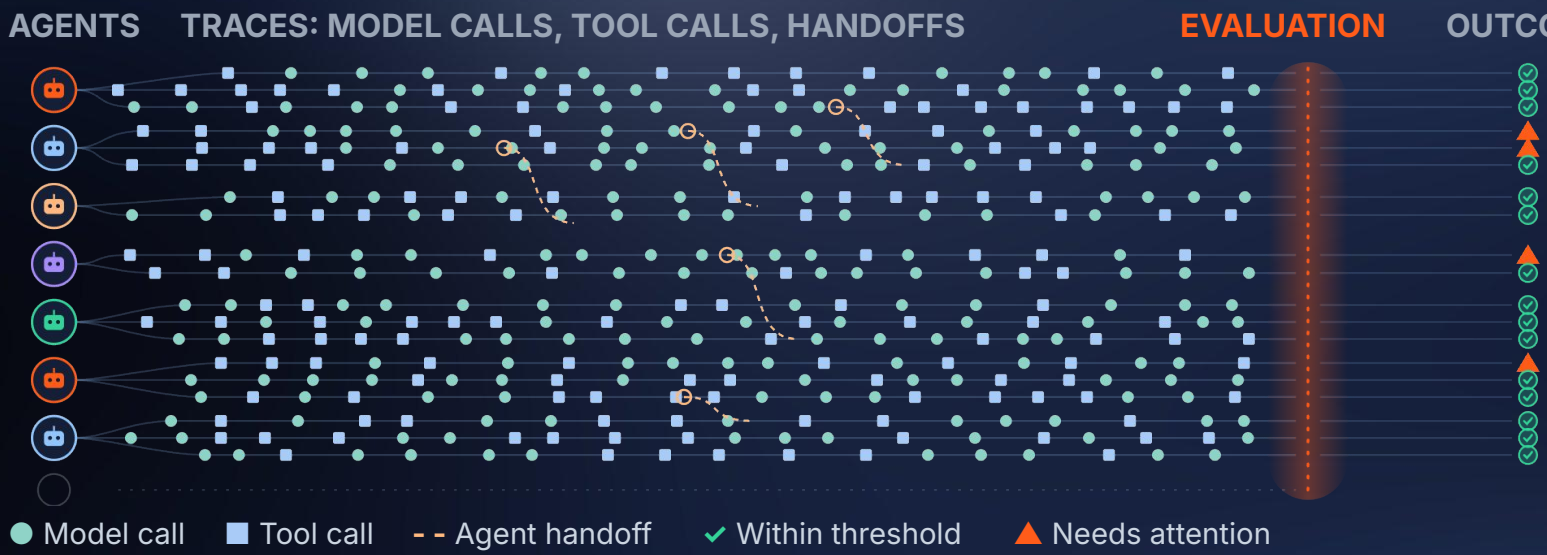
Evaluate at Every Level of the Trace



Evaluate the whole trace, each agent on its own, or a single model call.

Now Imagine This at Scale

Hundreds of agents, thousands of traces, everyone needs to be understood and evaluated



Continuously Evaluate → **Understand Behavior** → **Detect Change** → **Improve**

More Visibility Means More Data & Cost

Balancing full observability against storage overhead and ingestion costs

Capture Everything

Cost: **high**

- Maximum visibility
- Easier investigation
- Better coverage of rare failures
- **Higher storage & processing cost**

Sample Traces

Cost: **lower**

- Lower ingestion and storage costs
- Less data to analyze
- **Risk of missing important behavior**
- **Incomplete view of production behavior**

What should we retain?

Keep in detail: Errors, policy violations, risky actions, unusual latency or cost

Sample: Routine executions

Preserve traces that may explain a problem; sample routine behavior.

Can We Evaluate Every Execution?

Evaluation itself has a cost — use a tiered approach to evaluate efficiently based on risk.

↓ Fewer executions reach each tier

↑ Cost and depth per check increase

TIER 1	Rules & Deterministic Checks	Fast, inexpensive, and applied broadly across all executions.
TIER 2	Small Language Models	Frequent, focused quality checks and policy validations.
TIER 3	Large Judge Models	Reserved for complex, ambiguous, or high-risk cases.
TIER 4	Human Review	Critical decisions and highly uncertain outcomes.

Examples for the SLM tier — Tool-selection correctness · Required-step completion · Relevance · PII detection · Safety classification

Continuous evaluation becomes practical through a tiered approach, not by sending every trace to the largest model.

04

Controlling What Agents Can Do

What are they allowed to do?

Every Agent Needs Boundaries

Define and enforce explicit permission limits across systems and actions

AUTHORIZATION

Authorized Actions Only

An agent should only perform the actions it is authorized to perform.

CONTROL & POLICY

Explicit Boundaries

Define and enforce what the agent can — and cannot — do.

Agents must operate strictly within defined authorization boundaries across all integrated systems

Authorization Becomes More Contextual

It is no longer just: "Does this agent have access?"

Key building blocks include:

- **Agent identity** — who the agent is
- **Delegation / OBO** — who it is acting for
- **Fine-grained authorization** — what it is allowed to do
- **Least-privilege access** — only the access it needs
- + more

Why This Gets Hard at Scale

Many Agents × Many Users × Many Tools & Actions × Changing Context =

MANAGE

Agent Boundaries

Hundreds of agents need different permissions and boundaries.

KEEP CONSISTENT

Unified Policies

The same organizational policies must apply across teams, tools and runtimes.

KEEP CURRENT

Dynamic Lifecycle

Agents, users, tools and policies change; access must be updated and revoked accordingly.

ENFORCE & AUDIT

Auditability

Every action must be controlled at runtime and remain traceable.

The challenge is not defining access once — it is managing and enforcing the right boundaries continuously and consistently at scale.

Why This Gets Hard at Scale

Many Agents × Many Users × Many Tools & Actions × Changing Context =

MANAGE BOUNDARIES

Different agents require different permissions and constraints.

KEEP THEM CURRENT

Access must evolve as agents, users, tools and policies change.

ENFORCE CONSISTENTLY

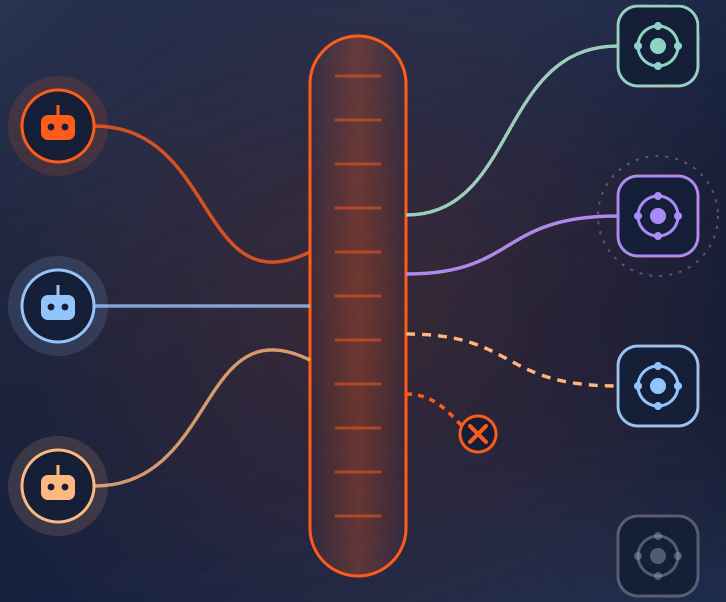
The right boundaries must apply across every system and runtime.

At scale, authorization is not something you define once — it must be continuously managed and enforced.

05

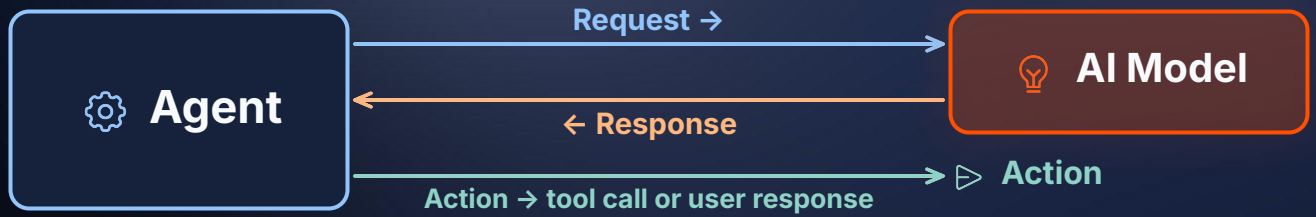
Governing How Agents Use Models

How are they using models?



Every Model Interaction Introduces Risk

Every model interaction can expose data, introduce malicious instructions or influence the agent's next action



⚠ DATA EXPOSURE
The agent may send PII, secrets, internal documents or tool results.

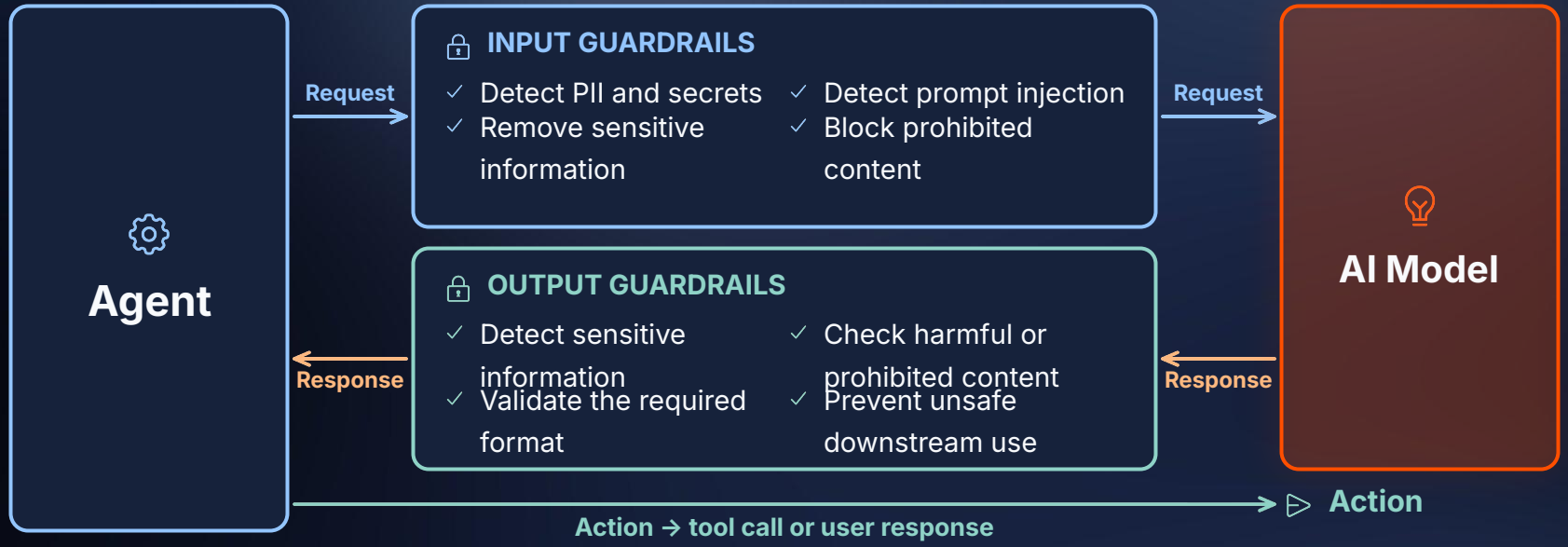
⚠ MALICIOUS INSTRUCTIONS
Prompt injection may come from users, retrieved documents or tool responses.

⚠ UNSAFE OUTPUTS
The model may produce harmful, sensitive or policy-violating content.

⚠ UNSAFE INFLUENCE
The response may shape another prompt, a user response or a tool call.

How do we protect every interaction?

Guard Both the Request and the Response



Possible outcomes: **Allow** | **Redact** | **Rewrite** | **Block**

Every model call needs input and output guardrails

Organization-Wide and Agent-Specific Guardrails

ORGANIZATION-WIDE GUARDRAILS · Applied to every model interaction

- **PII and secret detection**
- **Prompt-injection detection**
- **Harmful-content detection**
- **Sensitive-output detection**

↓ plus

AGENT-SPECIFIC GUARDRAILS · Applied according to the agent's purpose, data and risk

Customer Support Agent
Protect customer records and prevent inappropriate responses

Finance Agent
Detect confidential financial data and unsupported financial claims

HR Agent
Protect employee information and detect discriminatory content

Engineering Agent
Detect credentials, proprietary code and insecure code suggestions

How can organizations enforce common guardrails consistently while enforcing the right additional guardrails for each agent at scale?

06

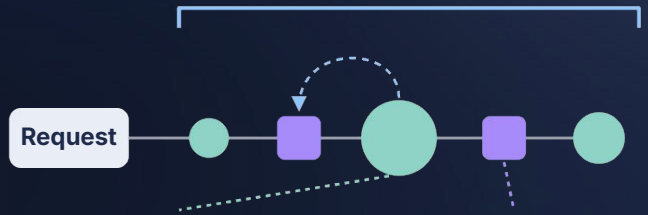
Operating Agents Cost-Effectively

How do we keep everything running?

What Drives the Cost of an Agent?

Number of model calls

Each reasoning step, retry, loop or agent handoff adds another call



Model type

Models selected, input tokens and output tokens

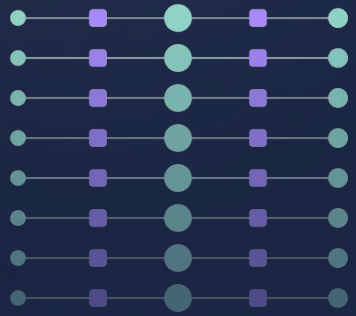
Tools & infrastructure

Paid APIs, data services, compute, storage and runtime

ONE REQUEST

Execution volume

User requests, scheduled runs and background executions



EVERY RUN, REPEATED

Total cost

What appears on the bill

Efficiency

Works both ways

Lower cost — Higher cost

Caching, batching, smaller context

Repeated calls, growing context

The Cost Problem Grows with Adoption

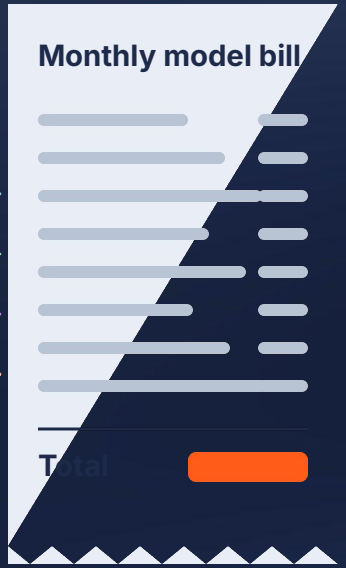
Costs become difficult to understand and control across:

Hundreds of agents
Owned by different teams and serving different use cases

Thousands or millions of executions
Including user requests, scheduled tasks and ambient operations

Multiple models and services
With different pricing models and usage units

Several environments
Development, testing, staging and production



Which agent?

Which team?

Which use case?

A monthly model bill cannot explain which agent, team or use case created the cost

FinOps for the Agent Estate

MEASURE 01

Track usage and cost for every execution

CONTROL 02

Allocate cost by agent, team, use case and environment, and apply budgets, quotas, limits and alerts

OPTIMIZE 03

Reduce waste and use cost-effective resources

ORGANIZATIONS NEED TO KNOW

- **Where is the money being spent?**
- **Who is responsible for the usage?**
- **Why did the cost change?**
- **Can spending be controlled before budgets are exceeded?**

Understand, allocate and control agent spending as adoption scales

Managing, Governing and Operating Agents at Scale

Requires a common foundation across six core operational pillars

01 Managing Agent Sprawl
Know what agents exist and where they are running

02 Maintaining Consistent Practices
Apply consistent lifecycle expectations across teams

03 Ensuring Agents Behave as Expected
Evaluate behavior and observe executions

04 Controlling What Agents Can Do
Control access to tools, systems and actions

05 Governing How Agents Use Models
Govern model selection, data and guardrails

06 Operating Reliably & Cost-Effectively
Manage health, performance, usage and cost

A common foundation

These are not six independent problems — they are connected requirements for operating an agent estate at scale.

THE ANSWER

Open enterprise control plane for AI agents

To manage, operate and govern agents consistently at scale

Next: Meet WSO2 Agent Manager

WSO2CON AFRICA

SEPT 22 – 24, 2026 | NAIROBI, KENYA

Thank You!

Malith Jayasinghe

VP of AI