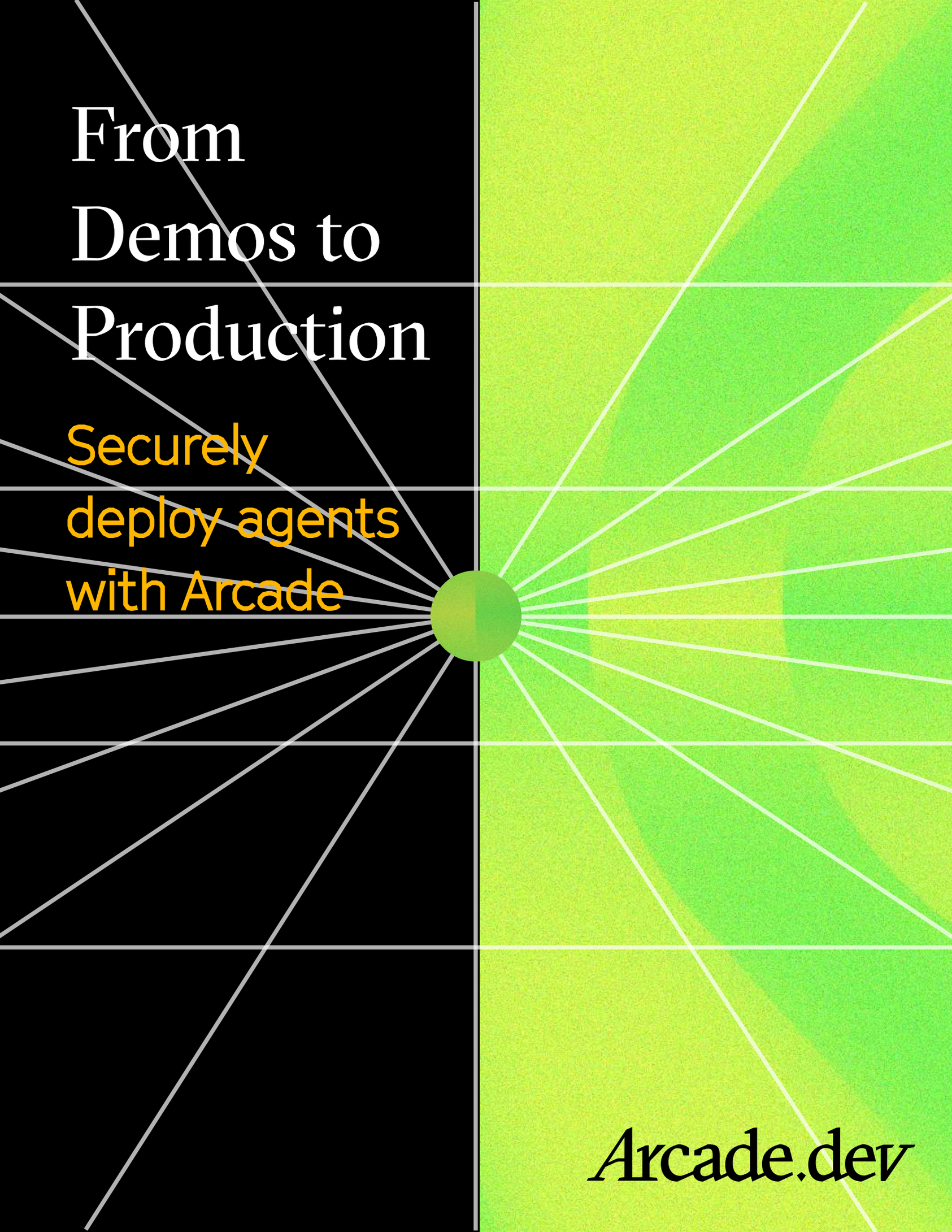
From Demos to Production

Securely deploy agents with Arcade



Arcade.dev

Executive Summary

Your AI agent works perfectly in demos. It understands context, makes decisions, and impresses stakeholders. Then deployment stalls. Security blocks hardcoded credentials. Single-user architecture won't scale. Tool execution breaks under load.

This is an industry-wide problem. 70% of AI agent projects fail to reach production. Gartner predicts 30% of generative AI projects will be abandoned after proof of concept by the end of 2025.

The Real Problem Isn't AI

Models can reason and execute complex tasks. The Model Context Protocol (MCP) standardized how agents communicate with tools. But MCP is a protocol, not a platform—it defines the language, not the production infrastructure.

What MCP leaves unsolved: Multi-user authentication is the critical gap. Teams resort to service accounts or hardcoded credentials because building OAuth for agentic workflows requires expertise few organizations have. Beyond auth, teams must build hundreds of tool integrations, prevent hallucinations in tool selection, and establish governance with audit trails.

For one agent and one tool, this seems tractable. At scale—dozens of tools, hundreds of users—custom code becomes unmaintainable. The gap isn't intelligence; it's execution infrastructure.

The Arcade Solution provides the end-to-end infrastructure MCP alone cannot deliver:

1. **1,000+ Production MCP Tools:** Pre-built integrations eliminating months of connectivity work
2. **Tool Development Kit:** Simple framework for building custom tools beyond the standard catalog
3. **Multi-User Authentication:** Secure, user-specific permissions across every tool—what MCP's protocol cannot provide
4. **Execution Engine:** Intelligent execution handling tool discovery, hallucination prevention, credential management, and governance

Production-ready architecture means engineering teams focus on differentiated features instead of maintaining custom code. Security teams approve deployments with enterprise standards and audit trails. Operations teams gain centralized visibility and control. Business stakeholders see ROI from agents that execute workflows, not just recommend actions.

This whitepaper examines the technical and organizational challenges preventing AI agents from reaching production and demonstrates how Arcade addresses the execution gaps MCP leaves unsolved.

If your team is building agentic applications and facing the prototype-to-production transition, request a demo https://www.arcade.dev/contact to see how Arcade accelerates your path to shipping.

The MCP Reality Check

The Model Context Protocol represents a genuine breakthrough. Before MCP, every agent platform spoke a different language. Building a tool for one system meant rebuilding it from scratch for another. MCP standardized the communication layer—agents and tools now share a common protocol for discovering capabilities, calling functions, and exchanging data.

But standardizing how agents talk to tools isn't the same as making agents work in production. Notice the pattern in agent demonstrations: weather APIs, web search, Wikipedia lookups—public data that requires no authentication. These examples work beautifully until you need the agent to actually do something useful: read YOUR emails, update YOUR CRM, access YOUR calendar. The moment tools require user-specific access, demos end and production problems begin.

MCP is a protocol, not a platform. It defines the format of the conversation, not the infrastructure needed to have that conversation at scale, securely, and reliably.

What Teams Discover After the Demo

Development teams building on MCP hit the same walls in the same order:

Tool Integration: MCP defines how to describe a tool and how an agent should call it. But someone still needs to build the actual integration to Slack, Gmail, Salesforce, or your internal systems. MCP gives you the schema format—you write the code that connects to APIs, handles authentication, manages rate limits, and processes responses.

Multi-User Authentication and Authorization: MCP has no concept of user context. Consider this standard MCP tool call:

```
{
  "method": "tools/call",
  "params": {
    "name": "send_email",
    "arguments": {
      "to": "customer@company.com",
      "subject": "Order Update"
    }
  }
}
```

Whose email account sends this message? MCP doesn't specify. The protocol can't pass user credentials or authorization tokens to tools. Teams building multi-user agents must architect their own authentication layer outside MCP—handling OAuth flows, token storage, refresh logic, and user-specific permissions.

Tool Selection Logic: When your agent has 50 tools available, MCP helps it understand what each tool does. But MCP doesn't help the agent choose correctly. Teams must build the logic that filters tools by context, ranks relevance, handles selection errors, and retries failures. This selection layer sits on top of MCP, not within it.

Hallucination Prevention: Agents sometimes call tools with invalid parameters, request non-existent functions, or misinterpret user intent. MCP defines the tool schema but doesn't validate agent behavior. Teams need validation layers that catch errors before they reach production systems—checking parameters, confirming intent, and providing guardrails.

Governance and Compliance: Production deployments need audit trails showing which user triggered which agent to call which tool with what permissions. MCP doesn't provide this visibility. Teams must build logging infrastructure, permission management systems, and compliance reporting around their MCP implementation.

The Production Gap

Production Requirement	What MCP Provides	What You Must Build
Tool connectivity	Protocol specification	Actual integrations to every system
User authentication	Nothing—no user context support	Complete OAuth infrastructure, token management, refresh logic
Tool selection	Discovery and schema format	Selection logic, filtering, error handling, retries
Error prevention	Tool schema definitions	Parameter validation, intent confirmation, guardrails
Audit and compliance	Communication standard	Logging, permission tracking, audit trails, governance

Why Custom Code Breaks at Scale

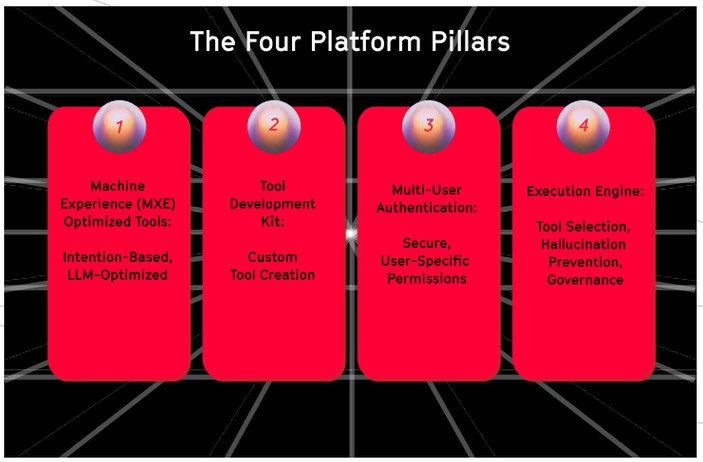
Your first team builds a Slack integration. They write 5,000 lines connecting MCP tool calls to Slack's API. It works. Six months later, a second team needs Slack access. They don't know about the first implementation, so they build their own. Now you have two Slack integrations with different error handling and auth patterns.

Slack updates their API. Both integrations break. The second team doesn't know the first exists, so they fix their version. The first team's agent keeps failing. Nobody notices for two weeks.

Meanwhile, your third team needs Gmail integration. They copy the Slack pattern and adapt it. Now any improvements must be replicated across three codebases. A security audit reveals all three store tokens differently—environment variables, database, custom encryption.

The Arcade MCP Execution Platform

MCP made agent-tool connectivity possible by standardizing the protocol. Arcade made it practical by building the production infrastructure that sits on top of that protocol. The platform combines four integrated capabilities that work together as a unified system—not isolated features, but architectural layers designed to solve the production gap.



The Four Platform Pillars

1. Machine Experience (MXE) Optimized Tools: Intention-Based, LLM-Optimized

Arcade provides 1,000+ tools engineered specifically for agent cognition through its Machine Experience (MXE) framework. Unlike service-based approaches that mirror API structures, Arcade organizes tools around agent intentions—how LLMs actually reason about tasks. Each tool undergoes continuous evaluation-driven optimization to maximize model selection accuracy, with schemas and descriptions refined based on actual LLM behavior. This fundamental engineering reduces hallucinations, improves first-time selection accuracy, and minimizes token consumption. The catalog covers major SaaS platforms (Slack, Salesforce, Gmail, Google Workspace, Microsoft 365, Jira, GitHub), databases (PostgreSQL, MySQL, MongoDB), and enterprise systems—all with built-in error handling, rate limiting, and retry logic.

2. Tool Development Kit: Custom Tool Creation

When your team needs connectivity beyond the standard catalog—internal systems, proprietary databases, specialized APIs—Arcade's Tool Development Kit (TDK) provides a framework for building custom MCP tools. The TDK handles schema generation, parameter validation, error handling, and MCP protocol compliance automatically. Developers write business logic; the platform generates the infrastructure. Custom tools integrate seamlessly with Arcade's auth, orchestration, and governance layers—no separate authentication or logging code required.

3. Multi-User Authentication: Secure, User-Specific Permissions

Arcade's authentication layer solves what MCP's protocol cannot—propagating user context through every tool call. The platform manages OAuth flows for 50+ services, handling authorization requests, token storage, automatic refresh, and credential isolation. When an agent needs to access a tool, Arcade ensures it uses that specific user's permissions through their authorized OAuth token. Security teams get user-level audit trails. Compliance teams can answer "who accessed what, when?" The agent never sees credentials—authorization happens in Arcade's infrastructure layer, invisible to agent code.

4. Execution Engine: Tool Selection, Hallucination Prevention, Governance

The execution engine coordinates tool selection, validates parameters, prevents errors, and maintains governance. Powered by MXE's intention-based design, the engine optimizes for how agents actually reason—reducing cognitive load and improving selection accuracy through continuous evaluation. Built-in parameter validation catches errors before execution—malformed inputs, invalid formats, out-of-range values—preventing failures from reaching production systems. When agents make errors, the engine's retry logic with context-aware fallbacks recovers automatically. Every action generates compliance-grade audit logs with full context: user identity, agent identity, tool called, parameters, timestamp, and authorization scope. Centralized management provides operations teams real-time visibility and control.

How the Arcade Platform Works: End-to-End Workflow

Consider the Slack-CRM agent deployment—the scenario that revealed the production barriers in earlier sections. With Arcade, here's how the platform solves what custom code couldn't.

Integration: The Slack and Salesforce tools already exist in Arcade's catalog. No custom API code. No authentication logic. No rate limit handling. Import the tools, configure which Slack channels and Salesforce objects your agent should access, done. Deployment time: minutes, not weeks.

Authentication: Your sales team installs the agent. When a rep first uses it, Arcade prompts them to authorize Slack and Salesforce through standard OAuth flows—the same "Connect to Slack" experience they've seen in other apps. They authorize with their own credentials. Arcade stores the OAuth tokens securely, isolated per user. The agent code never sees these credentials.

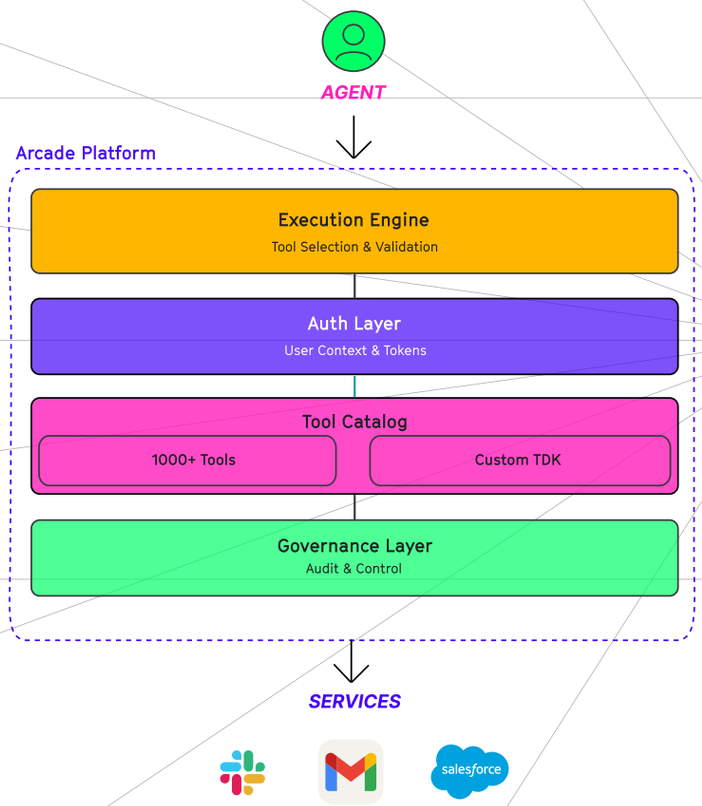
Execution: A sales rep asks the agent to monitor a customer conversation in Slack and update the deal in Salesforce. The agent formulates the request. Arcade's execution engine:

1. **Tool Selection:** Identifies the Slack "read_channel" tool and Salesforce "update_opportunity" tool from the semantic catalog
2. **Authentication:** Retrieves that specific user's OAuth tokens for Slack and Salesforce
3. **Authorization Check:** Verifies the user has permission to read that Slack channel and update that Salesforce opportunity
4. **Parameter Validation:** Confirms the agent provided valid channel IDs, opportunity IDs, and field values
5. **Execution:** Calls the tools with user-specific credentials
6. **Error Handling:** If Slack rate limits trigger, automatic retry with exponential backoff
7. **Audit Logging:** Records user identity, agent action, tools called, data accessed, timestamp

Governance: The operations team views a dashboard showing all agent activity across the sales organization. They see which reps are using which tools, error rates by tool, authorization patterns. When a rep leaves the company, admin deactivates their account—all their agent authorizations terminate immediately, no orphaned tokens. Security audits show complete trails: "Rep A accessed Customer X's Slack channel and updated Deal Y in Salesforce at timestamp Z with authorization scope [read:channels, write:opportunities]."

The Integration Advantage: These four pillars don't operate independently—they form a unified architecture where each layer enhances the others. Tools built with the TDK automatically inherit the auth layer's user context propagation and the execution engine's error handling. Custom tools get the same governance, monitoring, and audit trails as built-in tools. Authentication tokens are managed consistently across all tools. The execution engine learns from usage patterns across all agents and all tools, improving execution for everyone.

Platform Architecture



This integrated platform architecture addresses the production gaps that MCP's protocol alone cannot fill—transforming agent-tool connectivity from a communication standard into production-ready infrastructure. The next section examines the business impact these technical capabilities deliver.

Solving the Last-Mile Problem

Teams building agents—or any multi-tool, multi-user system—discover that solving AI capabilities doesn't solve production readiness. There are four gaps that represent the last mile between demo and deployment.

Tool Availability: Building and Maintaining Integrations

The Problem: Your agent needs to connect to Slack, Salesforce, Gmail, Google Calendar, Jira, and a dozen other systems. Each integration requires building API connectors, managing authentication, handling rate limits, processing responses, and maintaining code as APIs evolve. Your first integration takes two weeks. Your tenth takes three weeks because you're also maintaining the previous nine. The MCP ecosystem now offers community servers for popular services, but connectivity alone doesn't solve the problem—tool quality matters. Agents must be able to select the right tool and use it correctly. Most tools are API wrappers that agents struggle to use effectively.

The Solution: Arcade provides 1,000+ production-grade tools across major platforms (Slack, Salesforce, Gmail, Google Workspace, Microsoft 365, Jira, GitHub), databases (PostgreSQL, MySQL, MongoDB), and enterprise systems. Each tool is maintained, tested, and optimized for agent selection accuracy—not just wrapped from API documentation. When you need connectivity beyond the catalog, the Tool Development Kit (TDK) provides a framework for building custom tools with schema generation, parameter validation, error handling, and MCP compliance built in.

Multi-User Authentication and Authorization: The Critical Differentiator

The Problem: A Slack-CRM agent works perfectly when you hardcode your credentials. But deploying to your sales team exposes two failure patterns teams encounter:

Service Account Bypass: Using a service account (the common RAG approach) creates a privilege escalation vulnerability. When the intern runs the agent, they inherit admin-level access through the agent's credentials. Security teams recognize the risk and force developers to scope service accounts to near-public data—resulting in chatbots that can recite the FAQ but can't answer "where's my order?" because proving Bob can't access Alice's data is impossible.

User Credential Danger: The alternative—embedding user credentials directly into agent configuration—doesn't scale (every user manually enters credentials) and creates safety risks. Agents inherit full user permissions, including delete access. We've seen coding agents attempt to delete root directories, email agents with carte blanche to wipe inboxes. Without fine-grained authorization scoping what agents can do versus what users can do, deployment becomes reckless.

Why OAuth Fails in Agents: Standard OAuth workflows assume a trusted application that can securely store secrets and complete redirect flows. Agents are non-deterministic, adversarial, and can't hold secrets. Building OAuth that works inside agentic workflows—where authorization happens after the agent starts executing, not before—requires fundamental architectural innovation. Most teams lack the specialized expertise at the intersection of ML systems, identity protocols, and distributed computing required to solve this. MCP provides no user context, leaving teams to architect authentication infrastructure from scratch. The result: projects stall for months or ship with dangerous workarounds.

The Solution: This is why Arcade built multi-user authentication into the platform architecture—because MCP alone cannot deliver it. Arcade's auth layer sits between your agent and every tool, managing user-specific OAuth flows automatically. When the Slack-CRM agent needs to read Slack, Arcade ensures it accesses that specific user's authorized channels with their permissions—not a bot account, not shared credentials. The platform handles token lifecycle management, secure storage, automatic refresh, and credential isolation. Each user authorizes tools through standard OAuth flows; Arcade propagates that context to every tool call. The agent never sees credentials. Security teams get user-level audit trails. Compliance teams get answers to "who accessed what, when?"

This isn't a feature—it's foundational architecture recognizing agents are applications serving multiple users, not users themselves. As Sam Partee, CTO of Arcade, frames it: "Agents shouldn't just be able to work for you. They need to be able to work as you."

The distinction: "work for you" means reactive. You tell the agent what to do, and it does it. "Work as you" means proactive. You authorize what the agent can do, and it operates autonomously within those boundaries. The agent acts with your permissions, within your access levels, on your behalf; but you're not manually triggering every action. Without this layer, multi-user deployment remains unsolvable.

Tool Selection & Execution: Right Tool, Right Time, Minimal Hallucinations

The Problem: When your agent has access to 50 tools, it needs to select correctly. Should the Slack-CRM agent update Salesforce after every message or batch updates hourly? Should it read all Slack channels or filter by keywords? Agents sometimes call tools with invalid parameters, select irrelevant tools, or misinterpret ambiguous requests. Custom tool selection logic becomes complex quickly—context filtering, relevance ranking, parameter validation, error recovery, and retry logic for every tool combination.

The Solution: Arcade solves this through Machine Experience (MXE)—tools designed from the ground up for agent cognition. Consider Slack: the API exposes 200+ methods organized by technical function (chat.postMessage, conversations.list, users.info, files.upload...). Most MCP servers expose all 200, forcing agents to navigate service-based structures designed for developers. Arcade provides intention-based tools instead: "notify team," "get channel updates," "share with channels"—abstractions that match how agents reason about tasks. The platform runs continuous evaluations on tool selection accuracy, optimizing schemas and descriptions based on actual model behavior. This isn't just better documentation—it's fundamental engineering that structures tools around agent intentions rather than technical endpoints. Built-in parameter validation catches errors before execution—malformed inputs, invalid formats, out-of-range values. When agents make errors, the execution engine's retry logic with context-aware fallbacks recovers automatically. The platform continuously improves selection accuracy through evaluation-driven optimization, learning from patterns across all agents using Arcade.

Governance: Who Can Do What, Audit Trails, Compliance

The Problem: Production deployments demand answers your demo avoided. How do you audit which sales rep accessed which customer data? How do you prevent the agent from exposing EMEA deals to North America teams? How do you revoke access when someone leaves? How do you prove SOC2, GDPR, or HIPAA compliance when audit trails show "bot@company" performed all actions? Custom governance infrastructure means building permission management systems, detailed logging, compliance reporting, and administrative dashboards—infrastructure complexity that delays deployment by months.

The Solution: Arcade solves this with centralized management providing granular governance at user and agent levels. Every tool call traces back to a specific user with specific permissions granted through OAuth flows. Operations teams get real-time visibility into which agents access which tools on behalf of which users. Permission revocation is immediate—deactivate a user, and all agent authorizations terminate instantly. Compliance-grade audit logs capture every action with full context: user identity, agent identity, tool called, parameters, timestamp, and authorization scope. The governance layer integrates with existing enterprise identity providers, so agent permissions flow through your Okta, Azure AD, or SSO infrastructure.

Business Impact & Getting Started

The technical capabilities described in previous sections translate directly to business outcomes. Organizations building agents face a stark choice: invest months building custom infrastructure or deploy on a platform designed for production. The difference shows up in velocity, cost, and risk.

Engineering Velocity: Unblocking Stalled Projects

The primary value of production-ready infrastructure is enabling projects that would otherwise fail. Analysis shows:

- 91% faster service integration: Teams building custom integrations average 2-3 weeks per connector. With Arcade's pre-built tools, integration drops to minutes for common services, hours for custom tools using the TDK. A team integrating 10 services saves 15-25 weeks.
- 7 minutes to secure access: Getting a developer to securely access an important system with proper user-level authentication takes 7 minutes with Arcade versus months building OAuth flows manually.
- Rescued blocked projects: The most significant impact is enabling agents that couldn't reach production otherwise. Teams discover the authorization problem after building sophisticated reasoning. Without a platform solution, projects stall or get canceled. Even the smallest agent team—6 people at $1.2M per year—makes failure expensive.

Infrastructure Efficiency: Reducing Operational Costs

Beyond enabling deployment, platform infrastructure reduces ongoing operational burden:

- 73% reduction in manual workflow processing: Production agents automate workflows that previously required human intervention. The Slack-CRM agent example eliminates manual data entry from customer conversations to CRM systems—work that consumed hours weekly across sales teams.
- LLM cost optimization: Arcade's tools reduce token consumption three ways: (1) intention-based tools require less context than raw API wrappers, (2) semantic tool descriptions improve first-time selection accuracy, reducing retries, (3) parameter validation catches errors before expensive LLM calls. Teams report 40-60% reduction in token costs for tool-heavy workflows.
- Eliminated maintenance burden: Teams maintaining custom integrations spend 20-30% of engineering time on updates—API version changes, new authentication requirements, bug fixes. Platform-managed tools eliminate this overhead.

Risk Mitigation: Security, Compliance, and Scale

Production deployment requires passing security reviews and maintaining compliance—areas where custom infrastructure often fails:

- 84% improvement in security audit outcomes: User-level authorization with complete audit trails enables agents to pass security reviews that reject service account approaches. Security teams can answer: Who accessed what? When? With what permissions?
- 100% user attribution for compliance: Every action traces to a specific user through OAuth flows. SOC2, GDPR, HIPAA, and industry regulations require this attribution. Service accounts create compliance gaps that block deployment.
- User trust through familiar flows: When users authorize agents through standard OAuth—the same "Connect to Google" experience they've used with other apps—they recognize a trusted pattern. This familiarity builds confidence compared to custom authorization flows. Users see what permissions they're granting and can revoke access through their service provider settings.

Arcade.dev

Get Started Today

Two paths to deploying production-ready agents with Arcade:

Request a Demo See the platform in action. Our team will walk through your specific use case, show how Arcade's architecture solves your production barriers, and design a deployment plan for your agents.
[Schedule Demo: https://www.arcade.dev/contact

Sign Up and Start Building Developers can start immediately with Arcade's free tier. Build your first agent with production tools, test multi-user auth flows, and validate the platform with your use cases before committing.
[Get started: https://docs.arcade.dev/en/home/custom-mcp-server-quickstart]

The difference between agents that demo well and agents that actually work comes down to infrastructure. MCP made the connection possible. Arcade makes it practical.