STACKLOK

MCP Maturity Model

Identify where your enterprise is today, and how to move to the next stage(s)

Contents

Introduction

The Model Context Protocol (MCP) represents a fundamental shift in how AI systems integrate with enterprise tools and data sources. Introduced by Anthropic in November 2024, MCP standardizes how AI systems connect to external services and data, extending the capabilities of AI models, agents and assistants.

MCP has come a long way in its first year. More than 30,000 MCP servers have been published, from both individual developers and some of the world's largest enterprises. And MCP has evolved from proposal to fully governed protocol with a formal structure, release cadence and roadmap. While it's still early days, leaders are diving into MCP now because they recognize the opportunity for competitive advantage. They're multiplying the value of their AI agents and assistants and moving pilots into production to drive business outcomes.

The MCP Imperative

There's another reason for leaders to embrace MCP ... your employees are already using MCP servers, and that's creating risk for your enterprise. MCP adoption inside enterprises is not currently a controlled, top-down initiative, but it should be. With more than 30,000 MCP servers now publicly available, it is easier than ever for individuals to wire external AI agents directly into corporate data, tools, and APIs. This creates a growing operational and security gap.

This is the shadow AI problem; unsanctioned MCP servers running on local machines and unknown tools accessing sensitive systems. Shadow AI emerges as users reach for whatever helps them move faster. And unless an enterprise provides a safe, governed alternative, usage continues invisibly and without oversight.

You can change the equation by standardizing the way your teams (and their AI agents and assistants) access context behind your corporate firewall. Rather than blocking MCP adoption (an approach that inevitably fails), organizations must offer a governed MCP platform that delivers the same speed and flexibility users seek, but with enterprise-grade security, auditability, and control. The imperative is clear: adopt MCP intentionally, or inherit an ungoverned landscape of ad-hoc integrations that put data, compliance, and operational resilience at risk.

MCP Before and After

A common question we receive from enterprises at the earliest stages of the maturity model is, "What will we be able to do with MCP that we cannot do without MCP?"

Well, today, most AI agents and assistants show up like interns. They're smart and capable, but they know nothing about your organization, and consequently, they require a high degree of micro-management to be kept on task and to produce useful work. That's why approximately 46% of AI pilots are currently failing, and why many of those pilots leave end users (and internal champions) frustrated.

Conversely, let's consider what it'd be like to work at an MCP-native organization. Here are some concrete examples of what's possible:

- Developers will have AI agents and assistants that asynchronously raise issues, search for and summarize context, propose solutions for developer approval, and then update documentation and artifacts. Developers will be freed to focus on innovations and your organization will ship more, higher-quality code faster than ever before.
- Customer support agents will know everything about a customer's entire history with your organization; every interaction, transaction, etc. Those agents will be able to offer customers fully personalized services and solutions, massively increasing resolution rates.
- Knowledge workers will be able to query a central service that will put the exact right data or document at their fingertips in seconds (in accordance with their permissions). They will never have to navigate internal hierarchies or wait in queues to get the context they need.
- All of the above will happen under your watch and within the full control of your private cloud. Every interaction will align with your governance and security policies.

In short, the promise of AI made on conference stages and paid commercials has a massive dependency on context acquisition and optimization, and therefore (as it stands), on MCP.

MCP FAQ

Before we get to the MCP Maturity Model, let's first address a couple of other FAQs, because while we assume that you're reading this paper due to interest in MCP, we won't assume you've pushed in all your chips.

Why should I bet on MCP as opposed to other protocols like A2A (introduced by Google), ACP (introduced by IBM), and others?

At present, MCP is the only protocol with real momentum; As of December 2025, more than 30,000 MCP servers had been published. Major solution providers including GitHub, Atlassian, Figma, Notion and more have introduced MCP servers, but have not invested in other protocols. In summary, MCP is the only real game in town.

I've read / heard that developers can eliminate the need for MCP by simply revising the .md files of their favorite AI agent and/or enabling Anthropic's "skills".

Tuning agent performance via .md files and skills are low friction steps towards better performance, but neither provides agents (or the developers and knowledge workers using those agents) with real, valuable context. For example, .md files instruct an agent on your coding style, but they don't connect it to data and tools behind your corporate firewall. And skills provide specific executables, but they're not portable across agents or discoverable across your organization, and so they're for individual users, not enterprises.

What are the security risks of adopting MCP? I've read/heard that malicious actors have targeted the MCP ecosystem with tool poisoning and other attacks.

Security is a real issue when you're talking about connecting external AI systems to your internal data and systems. And when the MCP spec was first introduced, the approach to security was overly simplistic. However, the spec has since been revised to introduce OAuth, and MCP platforms have come to market that offer advanced security capabilities. Security is something we'll feature prominently in the MCP Maturity Model, which is a nifty segue to the next portion of this document.

If you have other questions, don't hesitate to ask. We have an active Discord community and welcome you to post there: https://discord.gg/stacklok or just send us an email at enterprise@stacklok.com and we'll get right back to you.

MCP Maturity Model

The following Maturity Model is intended to provide organizations with a framework to assess their current MCP adoption level and then chart a path toward becoming an MCP-native organization.

Let's start with a summary of the stages, along with both technical and process markers of each stage:

MCP STAGE	Experiment & Prepare	Build Pilots & Capabilities	Scale MCP Infrastructure	MCP-Native Organization
DEPLOYMENT	Local UI / CLI with containers	Local UI / CLI with containers	Local and K8s Operator	Service Mesh
FOCUS	Discovery & learning	Pilots & deployment	Platform & governance	Innovation & revenue
TECHNICAL MARKERS	MCP servers run on personal laptops (no local containerization) No standardized connection practices No containerization, RBAC or audit trail Secrets stored locally or in plain text files	MCP servers deployed in container (Docker, Podman, etc) Standardized authentication Basic monitoring and logging in place Secrets managed via Vault, 1Password, etc	MCP servers deployed via K8s Consistent application of RBAC across namespaces Network policies enforced Observability stack deployed / integrated	Autonomous orchestration across MCP servers Hybrid registries combine public, private and SaaS Dynamic permissions and token exchange Policies engine to evaluate actions
PROCESS MARKERS	Individual-led experiments Ungoverned use Out-of-the-box integrations with current agents	Team-led experiments Definition of AI pilot success metrics Agent-specific pilots (coding assistants, customer support, etc.)	Enterprise-led deployments Governance frameworks Context acquisition strategies	Enterprise-wide context infrastructure AI-optimized SDLC AI embedded in customer-facing tools Asynchronous, agentic workflows

Now, let's dig into each of the stages in a little more detail.

Stage 1: Experiment & Prepare

Organizations at Stage 1 are just beginning to explore what MCP can do. Early adopters—usually developers or analysts—experiment with public MCP servers or simple local integrations to accelerate their work. Adoption is organic, uncoordinated, and largely invisible to IT, creating the first signs of shadow AI risk. This stage is defined by the realization that AI agents become dramatically more useful once connected to real tools and data.

Process Markers

- Individual developers or analysts are running MCP servers locally without visibility
- Multiple AI tools are being used inconsistently across teams
- No standard for authentication, secrets, or connectivity
- Security learns about AI usage ad hoc, often after a policy concern is raised
- Early wins exist, but they're isolated and cannot be repeated across teams

Technical Markers

- MCP servers run on laptops or personal sandboxes
- Basic filesystem or public MCP servers used for experimentation
- No containerization, network isolation, RBAC, or audit trail
- Secrets stored locally instead of using an enterprise vault
- No standardized connection patterns for AI tools

Risks if You Stay in This Stage

- Shadow AI usage expands faster than IT can understand or govern
- Sensitive data is accessed or moved without visibility
- Teams duplicate work because no shared registry or patterns exist
- Pilot AI successes stall because they cannot scale into production

Success Metrics (to complete Stage 1)

- Inventory of all known MCP usage across teams
- At least 2–3 high-value MCP-assisted workflows identified
- Baseline security requirements agreed upon with InfoSec
- Decision made on deployment path (local + containerization vs. centralized)

Required Roles

- Developer / engineer experimenting with MCP
- Security partner providing initial review guidance
- Program sponsor or innovation lead

Stage 2: Build Pilots & Capabilities

In Stage 2, organizations move beyond experimentation and begin deploying MCP servers for real workflows and early production use cases. Teams start building custom MCP servers, standardizing connection patterns, and establishing security controls. Pilots demonstrate clear productivity gains, but each team is still operating semi-independently. This stage marks the transition from isolated experiments to repeatable, policy-aligned capabilities.

Process Markers

- Teams are running MCP-based pilots for real workflows
- Early internal champions are emerging and sharing best practices
- Security has begun formalizing policies but lacks automation
- Leadership recognizes MCP's potential but expects validated outcomes

Technical Markers

- MCP servers deployed in containers, not on laptops
- Standardized authentication established (OAuth/OIDC)
- Reusable MCP configurations for dev/test environments
- Basic monitoring and logging instrumented (stdout, container logs)
- Secrets managed through Vault, 1Password, or Kubernetes Secrets

Risks if You Stay in This Stage

- Pilots remain siloed and fail to converge into a platform strategy
- Security fatigue grows due to inconsistent policy implementation
- Operational burden increases as more pilot servers are added
- Fragmentation makes it harder to migrate later to centralized governance

Success Metrics (to complete Stage 2)

- ≥ 5 pilot-grade MCP servers deployed with consistent container standards
- Reusable connection templates and permission profiles created
- MCP-assisted workflows producing measurable productivity impact
- Ability to deploy new MCP servers < 1 day (including configuration + testing)

Required Roles

- Platform engineer / DevOps practitioner
- Security architect (MCP permission profiles, secrets policies)
- Pilot workflow owners (developers, analysts, support teams)
- AI program manager tracking reuse and outcomes

Stage 3: Scale MCP Infrastructure

Stage 3 represents the shift from team-level pilots to an enterprise-wide platform. MCP servers are centrally orchestrated, governed, and deployed at scale through Kubernetes, CRDs, and automated policies. Multiple business units now rely on MCP-powered workflows, and the platform team is focused on reliability, observability, and security enforcement. MCP becomes a shared organizational capability rather than a niche tool.

Process Markers

- Multiple business units are requesting access to MCP-powered workflows
- Pressure increases to provide a governed, reliable, enterprise-wide platform
- AI assistants need MCP toolchains to support cross-system workflows
- Teams begin expecting production-grade SLAs for MCP availability

Technical Markers

- MCP servers orchestrated via Kubernetes Operator or similar automation
- RBAC applied consistently across namespaces and teams
- Network policies enforced; egress is controlled
- Centralized registry or catalog of MCP servers introduced
- Observability stack deployed (OpenTelemetry, Prometheus, log aggregation)
- GitOps used for MCPServer CRDs and configuration updates

Risks if You Stay in This Stage

- Platform team becomes a bottleneck for MCP server deployment
- Security posture weakens as the number of servers grows
- Business units build their own shadow platforms if the central one stalls
- Operational cost and toil scale non-linearly due to manual processes

Success Metrics (to complete Stage 3)

- ≥ 70% of MCP servers deployed via centralized platform (Operator or equivalent)
- Mean time to approve + deploy new MCP server: 3–5 days
- Full audit log retention meeting compliance requirements
- ≥ 80% of internal API domains exposed via MCP (where appropriate)
- Cross-team adoption: ≥ 5 business units using MCP-powered workflows
- MCP platform operating at ≥ 99% availability

Required Roles

- Platform engineering team (2–5 FTEs depending on scale)
- Security engineering (policy automation, anomaly detection)
- SRE / observability engineer
- AI enablement lead (user training & workflow design)
- Governance committee (security, platform, business stakeholders)

Stage 4: MCP-Native Organization

At Stage 4, MCP becomes part of the organization's core fabric. AI assistants orchestrate complex workflows across systems, using governed context to automate tasks that once required human coordination. Business teams build and adapt their own assistants, while security and platform controls ensure safe, compliant execution. MCP is no longer perceived as AI tooling; it is embedded infrastructure enabling continuous innovation, faster decision-making, and competitive advantage.

Organizational Symptoms

- AI-driven workflows are embedded in daily operations across the business
- Multi-MCP assistants automate recurring tasks with minimal human oversight
- MCP usage is no longer considered "AI"—it's standard organizational plumbing
- AI governance and platform operations run as a mature, continuous program

Technical Markers

- Autonomous orchestration across MCP servers (agent-led or platform-led)
- Hybrid registries combining public, private, and SaaS MCP servers
- Dynamic permissioning and token exchange with least privilege
- Cost, performance, and context optimization tooling integrated
- Behavioral guards and policy engines evaluate AI actions before execution
- Multi-model routing for agents across LLM providers

Risks if You Stay in This Stage (plateau risk)

- Pilots remain siloed and fail to converge into a platform strategy
- Security fatigue grows due to inconsistent policy implementation
- Operational burden increases as more pilot servers are added
- Fragmentation makes it harder to migrate later to centralized governance

Success Metrics (steady-state indicators)

- Majority of mission-critical workflows rely on MCP orchestration
- Business units deploy or modify their own assistants with guardrails
- ≥ 95% audit log coverage across all MCP interactions
- Time to onboard new team with MCP-enabled workflows: < 1 week
- Enterprise demonstrates measurable revenue, margin, or velocity gains attributable to MCP

Required Roles

- Platform & SRE team operating MCP as core infrastructure
- AI governance board ensuring responsible use
- Business workflow architects building and sharing assistant templates
- Security analysts monitoring AI-enabled interactions and policies

Enterprise MCP Platforms

By now it's apparent that to advance through the stages of the MCP Maturity Model, your enterprise will need an MCP platform. At this point in time, an MCP platform must offer at least four core components:

1 **Registry**

The foundation of a platform is a catalog of trusted MCP servers. Those servers can include official upstream servers, public vetted servers and internally developed servers. It's important that all servers (eventually) be stored and managed centrally for full control and visibility. The better registries allow administrators to permission and pre-configure every MCP server.

2 **Runtime**

Enterprises need to insist on running MCP servers in their private cloud; your data must stay in your environment. And so a hardened runtime allows you to containerize and deploy MCP servers to a Kubernetes cluster, using Kubernetes to orchestrate access policies, network endpoints and more.

3 **Gateway**

The gateway ensures the integrity of your MCP estate. As of this writing, everyone and their brother offers an MCP gateway, and there is a wide spectrum of "enterprise-grade" capabilities. Most gateways are content to manage security with an implementation of MCP spec'd OAuth. Enterprise-worthy solutions push further to offer federated token exchange, full IDP integration, OTEL-driven analytics and more. A properly designed portal provides every user (and AI agent) in the organization with a single endpoint to securely and efficient access the context they need.

4 **Portal**

Finally, for enterprise-wide adoption, it has to be dead simple for end users (developers and knowledge workers) to discover the context they need and install MCP servers with a single click. An intuitive portal is a key element of any MCP platform.

Now that we've defined a platform, let's consider the decision to build vs. buy, and in the case that you're buying, the criteria you need to consider.

Build or Buy an MCP Platform

If you've decided you need an MCP platform, the next decision is whether to build your own (proprietary solutions or assembled point solutions) or buy a complete platform. Here are specific criteria you need to consider in making that decision:

Time-to-value

If you face limited internal pressure to stop shadow AI and centralize control (6+ month timeframe), you can consider building a platform. If there's more immediate pressure to deploy a governed solution in weeks, then buying a proven solution is the obvious choice.

Competency

If your internal engineering organization has a short queue and platform-building sensibilities, then building a platform is an option. If there's competition for internal resources or your engineering team's capabilities skew more towards apps and services than platforms, then you should likely buy an MCP platform.

Ongoing capacity

As noted above, it's early days for MCP; both the protocol and the ecosystem are fast evolving. So, even if you have access to internal resources at the outset, you'll need ongoing access to resources and an aptitude to work through change to build a platform. If that's unlikely, a homegrown platform will struggle with breaking integrations, outdated practices, etc. and you'll need to buy an MCP platform.

Operational complexity

If you expect to deploy a smaller number of MCP servers for use by a defined set of departments, then building a platform is feasible. If you anticipate your organization pursuing MCP-native status, deploying hundreds (or more) MCP servers and automating MCP lifecycle management (health checks, failover, etc.) then it's time to find a leading solution to buy.

Criteria for Buying an MCP Platform

Most enterprises will make the decision to buy an MCP platform given the pressure to minimize time-to-value and maximize control. If you're heading down that path, you'll find the MCP market is noisy and a lot of platforms sound the same (on websites and in sales pitches). How do you really tell them apart? Here are a few criteria that we've seen matter in practice:

Track record

There are a lot of upstarts in the MCP ecosystem. Many of those upstarts are vibe-coded gateways from first-time founders. A few other providers have solutions deployed in production with large enterprise customers; they're likely led by individuals who have long-served enterprises. Consequently, they understand how to meet enterprise requirements and work through enterprise-specific complexities. Look for pedigree and ask for references.

Completeness

As noted above, an MCP platform includes four core components. A provider who offers just one or two of those components will require integrations with other pieces or your own proprietary efforts. That creates work for your team and extends your time-to-value. At present, only a small number of providers offer all four components at an enterprise standard.

Security

The MCP ecosystem faces both internal (shadow AI) and external (malicious actors) security threats; of all capabilities it's most critical that your MCP platform has a strong security posture. The minimum viable offerings will address authentication, often applying OAuth to the MCP spec. Enterprise-worthy solutions will offer advanced security features. Specific to MCP platforms, insist on federated token exchange, SSO and IDP integration, OTEL analytics, secrets management and (cryptographic) server provenance checks.

Openness

Finally, enterprises will need to ensure that any solution they buy has a sustainable future. Given that many MCP solution providers are startups, you can consider investor strength and runway, but more confidence can come from an established open source solution. An underlying open source project offers access to innovation and a path forward. More specifically, look for open source solutions that have external maintainers as an indicator of their momentum, relevance and longevity.

Stacklok Enterprise MCP Platform

Stacklok offers a complete Enterprise MCP Platform that simplifies the deployment and management of MCP servers. Fortune 500 enterprises are using this platform in production to move through the Maturity Model and deliver more return on their AI investments.
The Stacklok Enterprise MCP Platform includes all four core components referenced above. These components are all tightly integrated, and integrations can extend to your existing authentication, observability, secrets management and other solutions.

Deployment Modes

Stacklok's platform is available in three modes to suit different maturity stages:

UI

Desktop application for individual developers to discover, deploy, and manage MCP servers locally with a user-friendly interface.

CLI

Command line interface for quick deployment with advanced features like custom permissions and telemetry.

Kubernetes Operator

Enterprise-grade operator for teams to run MCP servers in multi-user environments with centralized management and security controls.

Key Enterprise Features

Secure by Default

Every server runs in isolated containers with only necessary permissions. Secrets managed securely, never in plaintext.

Declarative Management

MCP Server Custom Resource Definition enables GitOps workflows and infrastructure-as-code practices.

Observability

OpenTelemetry and Prometheus metrics to bridge the monitoring gap in MCP deployments.

Enterprise Authentication

OAuth/OIDC SSO integration, secure token exchange, and audit logging.

Multi-Namespace Support

Cluster-wide or namespace-scoped deployment modes following the principle of least privilege.

Secrets Integration

Native support for 1Password, HashiCorp Vault, and Kubernetes Secrets.

Architecture Overview

The Stacklok Enterprise MCP Platform is based on Stacklok's popular open source project, ToolHive. ToolHive is the only MCP platform that has multiple external maintainers (in this case, from Red Hat), and has a growing and vibrant community.

Stacklok exposes an HTTP proxy to forward requests to MCP servers running in containers. The proxy communicates via standard input/output (stdio), server-sent events (SSE), or Streamable HTTP. The Kubernetes Operator watches MCP server resources and creates proxy pods that manage actual MCP server containers, handling lifecycle management automatically.

For more information about the Stacklok Enterprise MCP Platform, visit stacklok.com, email **enterprise@stacklok.com** or engage us via Discord at **https://discord.gg/stacklok.**

To try the ToolHive open source project, check out our GitHub repo: **https://github.com/stacklok/toolhive.**