



Modellieren Sie Context-Protokoll-Strategien auf AWS

AWS Prescriptive Guidance



AWS Prescriptive Guidance: Modellieren Sie Context-Protokoll-Strategien auf AWS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Handelsmarken und die Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Einführung	1
Zielgruppe	2
Ziele	3
Was ist MCP?	5
Tools verstehen	5
Wann sollte MCP verwendet werden	8
Strategie für die Entwicklung von MCP-Tools	12
Umfang der Tools	13
Granular	13
Coarse-grained	14
Bewährte Methoden für die Festlegung des Umfangs von MCP-Tools	15
Werkzeugdefinitionen	16
Ansatz zur Werkzeugspezifikation	17
Docstring-Ansatz	18
Bewährte Methoden für Definitionen von MCP-Tools	19
Erkennung von Tools	19
Statische Definition	19
Dynamische Erkennung	20
Suchfunktion	21
Bewährte Methoden für die Entdeckung von MCP-Tools	21
Organisation der Tools	21
Bewährte Methoden für die Organisation von MPC-Tools	22
MCP-Hosting-Strategie	24
Hosting-Ansätze	24
Lokales Hosting	24
Remote-Hosting	26
MCP-Gateway	26
Bewährte Methoden für das Hosten von MCP-Servern	27
MCP-Verwaltungsstrategie	28
Authentifizierung und Autorisierung	28
Bewährte Methoden für die MCP-Authentifizierung und -Autorisierung	30
Steuerung der Last	30
Bewährte Methoden zur Steuerung der Auslastung	31
Operationelle Metriken	31

Mitwirkende	33
Verfassen	33
Überprüfend	33
Technisches Schreiben	33
Dokumentverlauf	34
.....	xxxv

Modellieren Sie Context-Protokoll-Strategien auf AWS

Amazon Web Services ([Mitwirkende](#))

März 2026 ([Verlauf der Dokumente](#))

Dieser Leitfaden kann Ihnen bei der Entwicklung und Implementierung von MCP-Strategien (Model Context Protocol) in Ihrem gesamten Unternehmen helfen, um Ihre KI-Aktivitäten zu unterstützen. Da Agenten und Sprachmodelle für den Geschäftsbetrieb immer zentraler werden, ist die Etablierung einer MCP-Strategie für erfolgreiche Agentenlösungen von entscheidender Bedeutung.

In diesem Leitfaden werden drei grundlegende Säulen für den Aufbau einer MCP-Strategie untersucht: MCP-Tool design, MCP-Serverhosting und MCP-Governance. Durch die Berücksichtigung dieser miteinander verbundenen Komponenten können Unternehmen skalierbare, sichere und effektive Systeme für die Verwaltung des Modellkontextes in ihren KI-Implementierungen einrichten. Diese Leitlinien bieten umsetzbare Erkenntnisse und strategische Leitlinien für Unternehmen in jeder Phase ihrer KI-Entwicklung, von ersten Experimenten bis hin zu umfassenden Produktionsbereitstellungen. Dies hilft ihnen, maßgeschneiderte MCP-Lösungen zu entwickeln, die auf ihre spezifischen Bedürfnisse und Ziele zugeschnitten sind.

Diese Best Practices basieren auf realen Implementierungen von Unternehmen, die MCP auf Unternehmensebene einsetzen, einer Analyse der aktuellen MCP-Spezifikationsstandards und Erfahrungen aus kundenspezifischen Large Language Model (LLM) -Anwendungen in der Produktion.

KI-Systeme werden LLMs in einer Vielzahl von Anwendungsfällen immer ausgefeilter und robuster. LLMs zeichnen sich dadurch aus, dass sie natürliche Sprache verstehen, menschenähnliche Reaktionen erzeugen und komplexe Informationen nachvollziehen können. Um jedoch LLMs von Konversationsschnittstellen zu Systemen überzugehen, die komplexe Aufgaben autonom ausführen können, setzen Unternehmen auf agentische KI-Architekturen, KI-Systeme, die ihre Umgebung wahrnehmen, über Ziele nachdenken, autonome Entscheidungen treffen, mehrere Schritte koordinieren und Maßnahmen ergreifen können, um Ziele im Namen der Benutzer zu erreichen. Dieser agentische Ansatz hilft Unternehmen dabei, KI-Systeme zu entwickeln, die Benutzerabsichten mithilfe natürlicher Sprache verstehen, autonom über mehrere Datenquellen und Tools hinweg koordinieren und personalisierte Erlebnisse in einem Umfang bieten können, der mit herkömmlichen Anfrage-Antwort-Mustern nicht möglich war. Um diese Agenten leistungsfähiger zu machen, müssen Unternehmen Zugriff auf ihre vorhandenen Tools und Daten gewähren, um das

kontextuelle Verständnis der Agenten zu erweitern und es ihnen zu ermöglichen, im Namen eines Benutzers zu handeln.

[MCP](#) bietet ein standardisiertes Protokoll für die Integration von KI-Tools, das eine konsistente Kommunikation zwischen Agenten und externen Ressourcen ermöglicht. MCP selbst definiert den Kommunikationsstandard, aber seine effektive Implementierung erfordert eine sorgfältige Berücksichtigung von Architekturmustern, Sicherheitsmodellen, betrieblichen Praktiken und Strategien zur Leistungsoptimierung, um skalierbare, sichere und wartungsfreundliche Lösungen zu erreichen.

[Dieser Leitfaden fasst die Erfahrungen aus MCP-Implementierungen in Unternehmen zusammen und bietet umsetzbare Empfehlungen, die auf das Well-Architected Framework abgestimmt sind.](#)

[AWS](#) Es behandelt Strategien für das Design von MCP-Tools, das MCP-Serverhosting und die MCP-Governance, die für die Entwicklung Ihrer eigenen MCP-Lösungen unerlässlich sind. Die Empfehlungen in diesem Leitfaden beziehen sich auf die folgenden fünf Säulen des AWS Well-Architected Framework:

- Sicherheit — Token-Isolierung, eingeschränkte Zugangsdaten, separate Autorisierung read/write
- Operational Excellence — Genauigkeitsmetriken für die Werkzeugauswahl, wertvolle Datensätze für Regressionstests
- Zuverlässigkeit — Geschwindigkeitsbegrenzung pro Benutzer und pro Tool, Lastabbau
- Leistungseffizienz — Workflow-spezifische Tools, Toolfilterung, semantische Suche zur Reduzierung der Nutzung von Kontextfenstern
- Kostenoptimierung — Teamübergreifend wiederverwendbare MCP-Server, geringere Token-Kosten pro Anfrage durch Toolfilterung

Zielgruppe

Dieser Leitfaden richtet sich an Architekten, Entwickler und Technologieführer, die agentische KI-Lösungen in ihren Organisationen implementieren. Um die Konzepte in diesem Leitfaden zu verstehen, sollten Sie die LLMs Funktionsweise verstehen und über grundlegende Kenntnisse in Bezug auf MCP, Tools und Prompt Engineering verfügen.

Ziele

Um agentische KI-Systeme zu entwickeln, die einsatzbereit sind, müssen Sie gemeinsam Lösungen für Governance, Optimierung und Sicherheit finden, um die Richtlinien Ihres Unternehmens zu unterstützen. Im Folgenden wird erklärt, wie dieser Leitfaden diese Ziele erreicht:

- **Steuerung** — Ohne zentralisierte Steuerung können Sie keine Prüfungsfragen zu Ihren KI-Workloads beantworten. Dazu gehört auch, welche Agenten wann mit welchen Berechtigungen auf welche Daten zugegriffen haben. Sie können auch keine Versionierung erzwingen. Im Abschnitt [zur MCP-Hosting-Strategie](#) dieses Handbuchs wird erklärt, wie Benutzer aufgrund fehlender systematischer Maßnahmen veraltete lokale MCP-Server mit bekannten Sicherheitslücken betreiben könnten.

Für regulierte Branchen ist Unternehmensführung von entscheidender Bedeutung. Prüfer möchten die Durchsetzung von Richtlinien und die Nachverfolgung der Toolnutzung für alle Agenten von einem einzigen Fenster aus verfolgen können. MCP Governance bietet das.

Wenn Sie die Empfehlungen in diesem Leitfaden befolgen, können Sie die Aufgabengenaugkeit bei von Experten begutachteten Benchmarks um 28-32% verbessern. Weitere Informationen finden Sie unter [MARCO: Multi-Agent-Echtzeit-Chat-Orchestrierung](#) (ACL Anthology-Website). Bei Governance geht es nicht nur um Compliance, sondern auch um die Verbesserung der Leistung Ihres KI-Systems für Agenturen.

- **Optimierung** — Ihre Teams können dieselben Integrationen mehr als einmal erstellen. Wenn beispielsweise fünf verschiedene Teams ihr eigenes Datenbankabfrageskript für ihre KI-Anwendung schreiben, um mit ihren Datenbanken zu kommunizieren, sind das Fünffache der Entwicklungskosten und fünf Sätze von Buglisten, die verwaltet werden müssen. Mit MCP können Sie es einmal erstellen und dann mit der gesamten Engineering-Community teilen. Die Einsparungen summieren sich, wenn die Anzahl Ihrer Agenten wächst.

Es gibt auch ein Problem mit den Kosten pro Anfrage, das die meisten Teams zunächst nicht bemerken. Jede Tooldefinition verwendet Kontextfenster-Token. Bei 20 Tools geben Sie zusätzlich zu den Benutzeranfragen 5.000 bis 10.000 Token pro Aufruf allein für Beschreibungen aus. Dies erhöht die Latenz und die Kosten für LLM-Inferenzen und verschlechtert die Genauigkeit, da das Modell Schwierigkeiten hat, das richtige Tool aus der Liste der verfügbaren Tools auszuwählen.

Agenten, die strukturierte Tool-Wrapper verwenden, sind bei Datenbankaufgaben etwa dreimal genauer als Agenten, die APIs direkt darauf zugreifen (weitere Informationen finden Sie unter [Middleware für LLMs: Tools sind entscheidend für Sprachagenten](#) in komplexen Umgebungen).

Es ist wichtig, wie Sie Tools für ein KI-Modell entwerfen und präsentieren. In diesem Leitfaden wird empfohlen, den Tools klare Schemata zu geben, sie auf tatsächliche Workflows statt auf unformatierte Endpunkte zu beschränken und die Informationen im Kontextfenster einzuschränken. Der Abschnitt zur [Strategie zur Entwicklung von MCP-Tools](#) in diesem Leitfaden befasst sich eingehend mit diesen Aspekten.

- Sicherheit und Compliance — Stellen Sie sich ein agentisches KI-System vor, das einen Säuberungsschritt halluziniert und versucht, eine Produktionsdatenbank zu löschen. Wenn der Agent die vollständigen Administratoranmeldeinformationen des Benutzers geerbt hat, wird die Löschung möglicherweise durchgeführt. Mit Token-Isolierung und abgespeckten Anmeldeinformationen, die nur Lese- und Erstellungszugriff gewähren, schlägt der Vorgang problemlos fehl.

Geregelte Arbeitsabläufe verschärfen dies noch weiter. Der Leitfaden enthält Beispiele (Pipelines im Gesundheitswesen, für die eine HIPAA-Validierung und Anonymisierung personenbezogener Daten erforderlich sind, bevor Patientendaten verarbeitet werden). Die Einbettung einer solchen Logik in MCP-Tools bedeutet, dass die Einhaltung der Vorschriften jedes Mal deterministisch erfolgt.

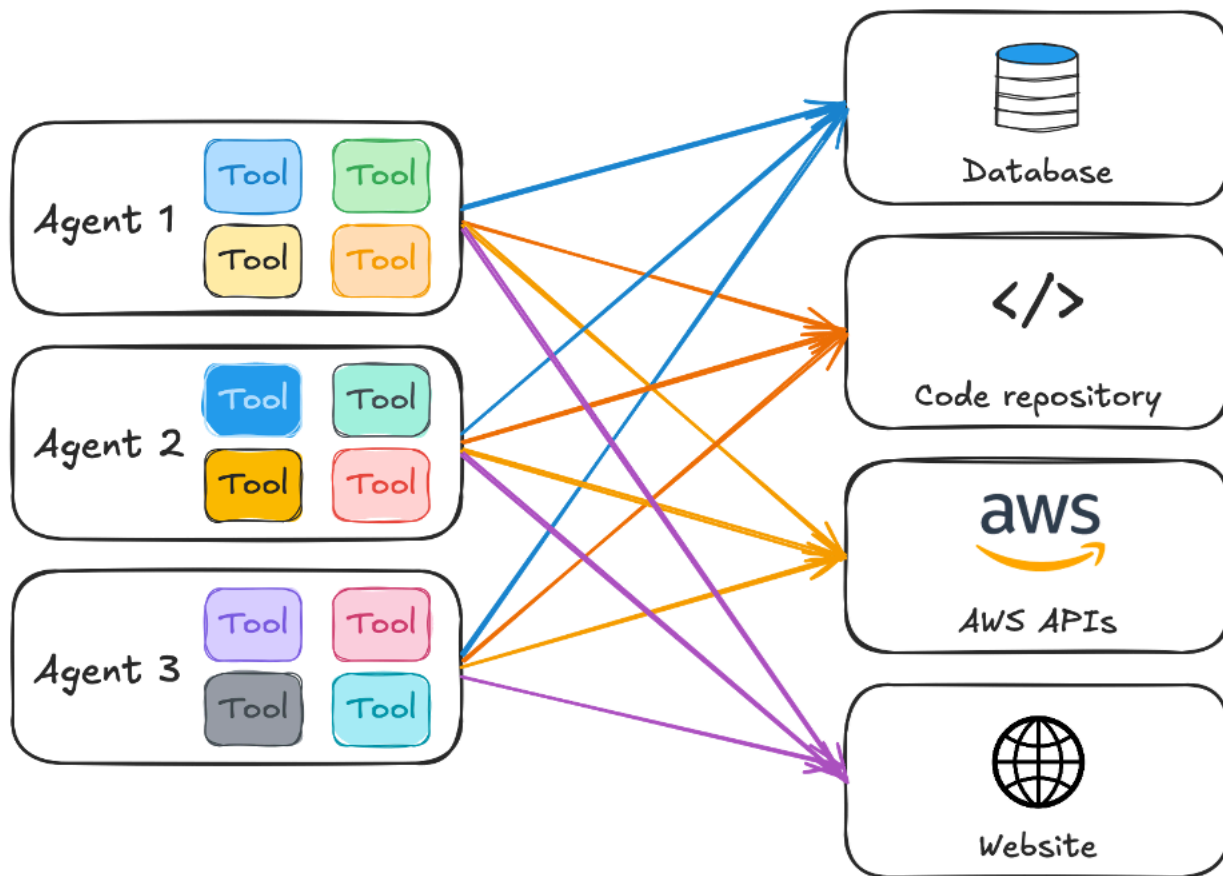
Was ist MCP?

LLMs funktionieren, indem sie anhand ihrer Trainingsdaten eine Antwort auf eine Aufforderung vorhersagen. Das bedeutet, dass das LLM nur Antworten auf Daten und Ereignisse geben kann, die es bereits gesehen hat. Methoden wie Retrieval Augmented Generation (RAG) und Wissensdatenbanken ermöglichen es Ihnen, Kontextdaten einzubeziehen. Wenn Sie jedoch einen LLM fragen würden, wie die Wettervorhersage für morgen aussehen wird oder wie viele Kunden sich in Ihrer Datenbank befinden, würde er wahrscheinlich halluzinieren oder nicht in der Lage sein, eine Antwort zu geben, da diese nicht dem vorab geschulten Wissen des LLM entsprechen. Um diese Art von Fragen beantworten zu können, benötigt ein Agent Zugriff auf externe Funktionen, Daten und APIs außerhalb des systemeigenen Kontextes des LLM.

Tools verstehen

Wir können dem LLM mithilfe von Tools Zugriff auf zusätzliche Systeme und Kontexte gewähren. Tools sind Funktionen, die dem LLM zur Erreichung eines klaren Ziels übertragen werden. Ein Tool könnte eine API aufrufen, eine Datenbank abfragen, Rechenoperationen durchführen, eine Code-Sandbox bedienen, eine Websuche durchführen und sogar ein anderes KI-System oder ein anderes Agent-as-a-Tool aufrufen. Jedes Tool sollte eine Beschreibung enthalten, die dem LLM mitteilt, was das Tool tut, wann es verwendet werden soll und welche Parameter es akzeptiert. Auf diese Weise kann das LLM auf der Grundlage der Benutzereingaben nuancierte Entscheidungen darüber treffen, welches Tool oder welche Kombination von Tools aufgerufen werden soll. Der LLM wird darüber informiert, welche Tools dem Agenten zur Verfügung stehen, sodass er Antworten generieren kann, die den Agenten anweisen, das Tool aufzurufen. Wenn Sie den LLM beispielsweise fragen, wie viele Kunden sich in Ihrer Datenbank befinden, sendet der LLM eine Antwort an den Agenten zurück und fordert ihn auf, das `query_database` Tool mit bestimmten Eingabeparametern auszuführen. Das LLM bestimmt, welches Tool aufgerufen werden soll und welche Eingaben für den Werkzeugabruf erforderlich sind. Der Agent führt dann das Tool aus, das die Eingabe in natürlicher Sprache in einen syntaktisch korrekten Funktionsaufruf umwandelt und die Abfrage ausführt. Der Agent ruft das Tool oder die Tools auf der Grundlage der Anweisung des LLM auf, und diese Ergebnisse werden an das LLM zurückgegeben. Dabei wird die Fähigkeit des LLM zur Argumentation gegenüber textbasierten Eingaben ausgenutzt und die geeigneten Tools für den Job ausgewählt.

Die folgende Abbildung zeigt, wie jeder Agent sein eigenes Toolset für jedes Ziel verwaltet.



Die Skalierung des Zugriffs auf Tools kann KI-Lösungen für Agenturen vor Herausforderungen stellen:

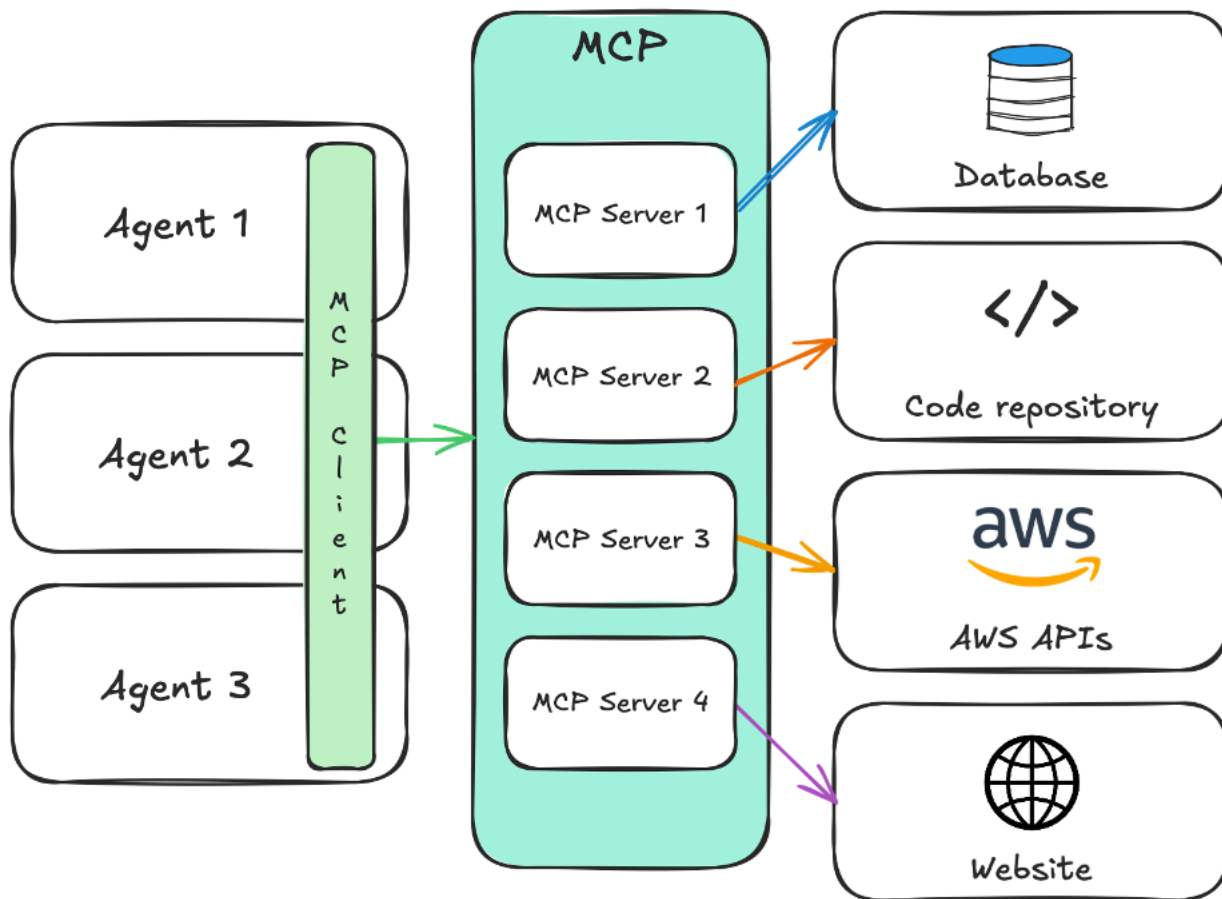
- Wenn jeder Entwickler sein eigenes Tool für dieselben externen Funktionen entwickelt, bedeutet das eine Menge doppelten Aufwand und unstandardisierte Interaktionsmöglichkeiten mit diesen externen Funktionen. Dies führt zu inkonsistenten Implementierungen bei Ihren Agenten. Sie könnten dieses Problem zwar lösen, indem Sie Standardtools in Bibliotheken entwickeln und verteilen, aber es fehlt eine zentrale Steuerung. Dies macht es schwierig, Sicherheitsrichtlinien durchzusetzen, die Nutzung von Tools nachzuverfolgen, die Versionierung teamübergreifend zu verwalten oder die Einhaltung von Unternehmensstandards sicherzustellen. Wenn Sie Tools direkt in den Agenten einbetten, müssen Sie Ihren Agenten außerdem jedes Mal neu bereitstellen, wenn ein neues Tool erstellt oder ein vorhandenes aktualisiert wird.
- Wenn Sie einem LLM Tools zur Verfügung stellen, wird dessen Kontextfenster beansprucht. Das Kontextfenster ist die Anzahl der Tokens (Texteinheiten, die LLMs verarbeiten — die in der Regel Wörter, Wortteile oder Satzzeichen darstellen), die ein Modell gleichzeitig berücksichtigen kann. LLMs haben begrenzte Kontextfenstergrenzen. Tools und ihre Dokumentation verwenden dieses endliche Kontextfenster zusammen mit System- und Benutzeraufforderungen. Wenn sich das

Kontextfenster füllt, kann es bei LLMs zu Leistungseinbußen kommen, was auf mehrere Faktoren zurückzuführen ist: Schwierigkeiten bei der Identifizierung relevanter Informationen, erhöhte Verarbeitungskomplexität und verringerte Fähigkeit, Argumentation zu finden. Die Herausforderung wird noch verschärft, wenn Tooldefinitionen, Systemaufforderungen und der Konversationsverlauf um den begrenzten Platz im Kontextfenster konkurrieren, da sie bei jedem LLM-Aufruf bereitgestellt werden.

Daher wirken sich die Anzahl der Tools und deren Dokumentation direkt auf die Leistung des LLM aus, z. B. auf Reaktionszeit und Genauigkeit.

MCP etabliert einen universellen Standard für die Verbindung von Agenten mit externen Funktionen. Er wird allgemein als „USB-C für KI-Anwendungen“ bezeichnet. [Anstatt Tools direkt bei Agenten zu registrieren, fungieren MCP-Server als Vermittler für das Hosten von Tools, die über 2.0 erkannt und aufgerufen werden. JSON-RPC](#) Anstatt Ihrem Agenten Dutzende oder Hunderte verschiedener Tools hinzuzufügen und diese im Laufe der Zeit zu verwalten, können Sie mit MCP MCP-Server registrieren, die die Tools kapseln, auf die Ihr Agent zugreifen kann. Dieser Ansatz standardisiert die Art und Weise, wie Tools verpackt, präsentiert und aufgerufen werden. Dies kann dazu beitragen, die Herausforderungen in Bezug auf Umfang und Steuerung zu bewältigen, die mit der Nutzung von Tools in Ihren Agenten verbunden sind. Außerdem werden die Entwicklung und der Betrieb der Agenten von den Tools entkoppelt, die für externe Funktionen verwendet werden.

Die folgende Abbildung zeigt Agenten, die MCP für den Zugriff auf externe Ressourcen verwenden.



Der MCP-Standard löst jedoch nicht alle Skalierungs- und Governance-Herausforderungen. Die Implementierung von MCP-Servern muss mit effektiven Strategien für Tooldesign, Hosting und Unternehmensführung kombiniert werden. Dieser Leitfaden enthält bewährte Methoden für jede Strategie, um Sie bei der Entwicklung und Verwendung von MCP als Teil Ihrer agentischen KI-Lösungen zu unterstützen.

Wann sollte MCP verwendet werden

MCP bietet eine strategische Infrastruktur für die Skalierung Ihrer agentischen KI-Initiativen. Durch die Zentralisierung von Toolmanagement und -steuerung reduzieren MCP-Server die Gesamtkosten für die Erstellung und Wartung benutzerdefinierter Integrationen für mehrere Agenten. Dies führt zu steigenden Renditen, wenn Ihr Agenten-Ökosystem wächst.

MCP wird wahrscheinlich Teil Ihrer Strategie, wenn:

- Sie benötigen eine zentrale Steuerung dafür, wie Agenten auf Unternehmenssysteme und -dienste wie Datenbanken, APIs, interne Tools und Integrationen von Drittanbietern zugreifen.

- Entwickler verbringen zu viel Zeit damit, benutzerdefinierte Integrationen zu schreiben, die nicht in allen Implementierungen konsistent sind.
- Sie verfügen über doppelte Tools, die allgemeine Funktionen erfüllen könnten.
- Sie möchten Ihre firmeneigenen Tools oder Daten externen Verbrauchern oder Agentensystemen von Drittanbietern über standardisierte, geregelte MCP-Schnittstellen anbieten und so neue Einnahmequellen erschließen und gleichzeitig Sicherheit und Kontrolle gewährleisten.

Nachdem Sie entschieden haben, dass MCP-Server Teil Ihrer Strategie sein werden, sollten Sie prüfen, ob die vorhandenen Open-Source-MCP-Serverimplementierungen Ihren Anforderungen entsprechen, ob sie erweitert werden müssen oder ob Sie benutzerdefinierte Server erstellen müssen. Viele vorgefertigte MCP-Serverimplementierungen sind in öffentlichen Repositories verfügbar und decken allgemeine Funktionen wie Dateisystemzugriff, Web-Browsing, Code-Sandboxes, Datenbankzugriff und API-Integrationen ab.

In vielen Fällen sind bereits vorhandene MCP-Server ausreichend. AWS Stellt beispielsweise einen verwalteten Remote-MCP-Server bereit [AWS MCP -Server](#), der KI-Assistenten und -Agenten sicheren, authentifizierten Zugriff auf Interaktionen in natürlicher Sprache ermöglicht. AWS-Services Sie können den verwenden, AWS MCP -Server um komplexe, mehrstufige AWS Aufgaben auszuführen, indem Sie Echtzeitzugriff auf AWS Dokumentation, syntaktisch korrekte API-Aufrufe und vorgefertigte Workflows, sogenannte [Agenten-SOPs](#), kombinieren, die bewährten Methoden folgen. AWS testet sie kontinuierlich AWS MCP -Server , um sicherzustellen, dass Kundenagenten sie erfolgreich einsetzen können.

Sie sollten diese vorhandenen MCP-Server zusammen mit Ihren Agenten testen, um festzustellen, ob sie Ihren Anwendungsfällen entsprechen. Wenn ein Agent Workflows nicht abschließt, falsche oder suboptimale Antworten generiert, komplexe mehrstufige Prozesse nicht bewältigt oder wichtige domänenspezifische Best Practices oder Sicherheitsüberlegungen übersieht, sollten Sie Verbesserungen in verschiedenen Dimensionen in Betracht ziehen.

Wenn vorhandene MCP-Server Ihre Anforderungen nicht vollständig erfüllen und sie Schwierigkeiten haben, die vorhandenen Tools korrekt zu verwenden oder präzise Antworten zu liefern, sollten Sie diese Verbesserungsansätze in Betracht ziehen, bevor Sie benutzerdefinierte Server erstellen:

- Den Agentenkontext erweitern — Wenn Ihr Agent Schwierigkeiten hat, die Tools auf einem vorhandenen MCP-Server korrekt oder effizient zu verwenden, sollten Sie erwägen, diese Tooldefinitionen durch zusätzliche Dokumentation oder Beispiele zu ergänzen. Dies trägt dazu bei, dem LLM zusätzlichen Kontext zu bieten.

- **Zusätzliche Tools hinzufügen** — Erweitern Sie bestehende MCP-Server um Tools, die auf zusätzliche Unternehmensdaten oder den Kontext zugreifen, den Agenten benötigen, um Workflows erfolgreich abzuschließen.
- **Verbessern Sie die zugrunde liegenden APIs** — Vereinfachen Sie Ihre Service-APIs, um mehr aus ihnen zu machen, LLM-freundlich indem Sie die Komplexität der Parameter reduzieren, klarere Fehlermeldungen bereitstellen und sinnvolle Standardwerte anbieten, die Agenten verwenden können.

Die Verwendung vorhandener MCP-Serverimplementierungen beschleunigt zwar die Entwicklung allgemeiner Funktionen, aber der Aufbau kundenspezifischer MCP-Server ist unverzichtbar, wenn Ihr Anwendungsfall spezielle Funktionen erfordert. Benutzerdefinierte MCP-Server helfen Ihnen dabei, Fachwissen zu bündeln, Unternehmensstandards durchzusetzen, die Zuverlässigkeit der Agenten für komplexe Workflows zu verbessern und die Einhaltung von Sicherheitsanforderungen zu unterstützen. Erwägen Sie die Erstellung eines benutzerdefinierten MCP-Servers in den folgenden Situationen:

- **Domain-specific Workflows** — Multi-step Workflows, die Fachwissen erfordern, sollten in benutzerdefinierten MCP-Tools zusammengefasst werden, wenn das erforderliche Wissen nicht in der API-Dokumentation erfasst ist. Anstatt beispielsweise Agenten die Orchestrierung komplexer Daten-Pipelines im Gesundheitswesen zu überlassen, die die Einhaltung des Health Insurance Portability and Accountability Act (HIPAA) validieren, personenbezogene Daten anonymisieren und in das [HL7-FHIR-Format](#) umwandeln müssen, sollten Sie ein Tool bereitstellen, das das Fachwissen direkt einbettet. `process_patient_data` Dadurch entfällt die Abhängigkeit vom LLM für die korrekte Orchestrierung und Ausführung der Workflow-Schritte, wodurch Konsistenz und Compliance verbessert werden.
- **Abstraktionen auf dem Goldenen Pfad** — Agenten haben möglicherweise Schwierigkeiten, optimale Ansätze zu implementieren, weil ihnen der organisatorische Kontext fehlt und sie sich eher an grundlegende Muster als an den bewährten Methoden der Organisation orientieren. In diesen Szenarien können Sie verbindliche Standards in Bezug auf Kosten, Leistung oder Sicherheit durchsetzen, indem Sie diese goldenen Pfade in benutzerdefinierten MCP-Tools zusammenfassen. Anstatt beispielsweise Agenten die Bereitstellung einer Infrastruktur mit Standardeinstellungen zu überlassen, die möglicherweise unsicher oder ineffizient sind, sollten Sie ein `deploy_secure_infrastructure` Tool bereitstellen, das die Standards Ihres Unternehmens direkt integriert.
- **Komplexe Orchestrierung mehrerer Dienste** — Anstatt den Agenten dazu zu bringen, komplexe Workflows zu orchestrieren, indem er versucht, die richtige Reihenfolge und den richtigen Satz

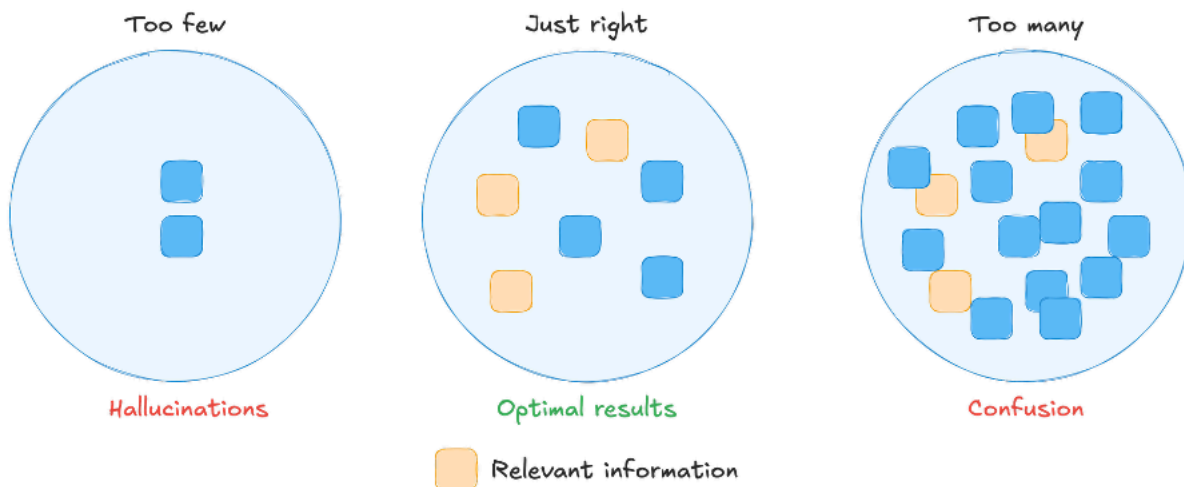
von Diensten abzuleiten, die in jedem Schritt verwendet werden sollen, können Sie diese Logik deterministisch in einem MCP-Tool aufbauen. Möglicherweise möchten Sie auch Fachwissen über optimale Serviceintegrationsmuster bereitstellen, von denen der Agent möglicherweise nichts weiß. Dies kann auch die Genauigkeit und Effizienz Ihrer Agenten verbessern.

- **Service-specific bewährte Methoden** — Dies ist bei sicherheitsorientierten Tools üblich, die Agenten bei der Implementierung von Verschlüsselungsrichtlinien, Zugriffskontrollen und Compliance-Mustern unterstützen, die für den Service spezifisch sind, auf den über das Agententool zugegriffen wird. Wenn es dienstspezifische bewährte Methoden für den Betrieb gibt, die nicht offensichtlich sind, kann die Verwendung eines MCP-Servers Ihnen helfen, sicherzustellen, dass diese implementiert werden und nicht einem Agenten überlassen werden, sich darüber Gedanken zu machen.

Strategie für die Entwicklung von MCP-Tools

Die Hauptaufgabe des MCP-Clients und -Servers besteht darin, Tools zu finden und dem LLM vorzustellen, damit es sie zur Verbesserung seiner Antworten verwenden kann. Dies macht das MCP-Tool-Design zu einer der wichtigsten Strategien für die Entwicklung effektiver MCP-Lösungen. Aus Sicht des Modells handelt es sich bei Tools um Funktionen, die sie bei Bedarf aufrufen können, um genauere und vollständigere Antworten zu liefern. Die Funktionsschnittstelle abstrahiert die zugrunde liegende Implementierung eines Tools, die von einem Wrapper für einen einzelnen API-Aufruf bis hin zu komplexer Workflow-Logik reichen kann.

Sie müssen jedoch ein Gleichgewicht mit der Menge der Tools finden, die dem LLM zur Verfügung gestellt werden. Wenn zu wenige Tools zur Verfügung stehen, ist das LLM möglicherweise nicht in der Lage, den richtigen Kontext und die richtigen Informationen zu sammeln, sodass es anhand der im Modell verfügbaren Informationen die beste Schätzung anstellt. Wenn zu viele Tools zur Verfügung stehen, kann das LLM verwirrt sein, was die richtige Auswahl und Reihenfolge der Werkzeuge angeht, was zu Halluzinationen führen kann. Ihr Ziel ist es, die Anzahl der Tools genau richtig einzustellen. Die folgende Abbildung zeigt die Herausforderungen, die mit zu wenigen und zu vielen Tools verbunden sind.



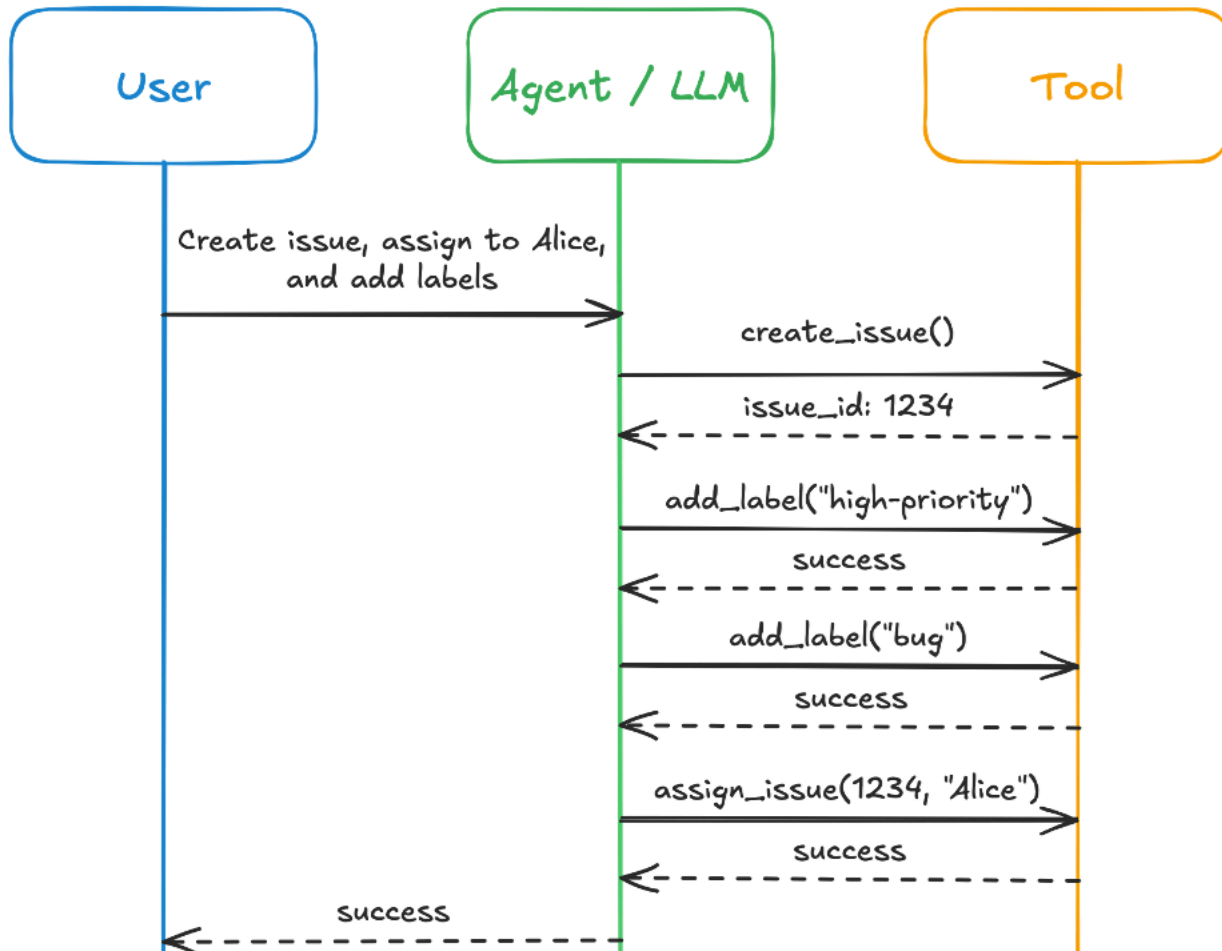
Die Lösung setzt voraus, dass Sie wissen, wie viele Tools bereitgestellt werden müssen und wie die einzelnen Tools auszulegen sind. Die Granularität Ihrer Tools, unabhängig davon, ob sie einzelnen API-Aufrufen oder kompletten Workflows zugeordnet sind, wirkt sich direkt auf die Gesamtzahl der Tools aus, die Agenten benötigen, und darauf, wie effektiv sie diese verwenden können. In diesem Abschnitt finden Sie bewährte Methoden für die Festlegung des Anwendungsbereichs von MCP-Tools, die Erstellung von Tooldefinitionen, die Suche nach Tools und deren Organisation.

Umfang der Tools

Es gibt zwei Ansätze für die Entwicklung von Tools: detaillierte und grobkörnige.

Granular

Bei einem detaillierten Ansatz würden Sie ein Tool pro API, Aktion oder Abfrage erstellen. Sie könnten beispielsweise `create_issue`, `get_issue`, `add_label`, `assign_issue`, und `close_issue` Tools für Ihr Git-Repository erstellen. Dies würde es dem LLM ermöglichen, detaillierte Aufrufe an jede API zu tätigen und jede API nach Bedarf zu orchestrieren. Stellen Sie sich die folgende Aufforderung vor: „Erstellen Sie ein Problem für den Produktservice mit dem Namen 'Abfrage gibt nur Teilergebnisse', kennzeichnen Sie es als Fehler mit hoher Priorität und weisen Sie es Alice zu.“ Die folgende Abbildung zeigt, wie ein Tool-per-API-Ansatz auf diese Aufforderung reagieren würde.

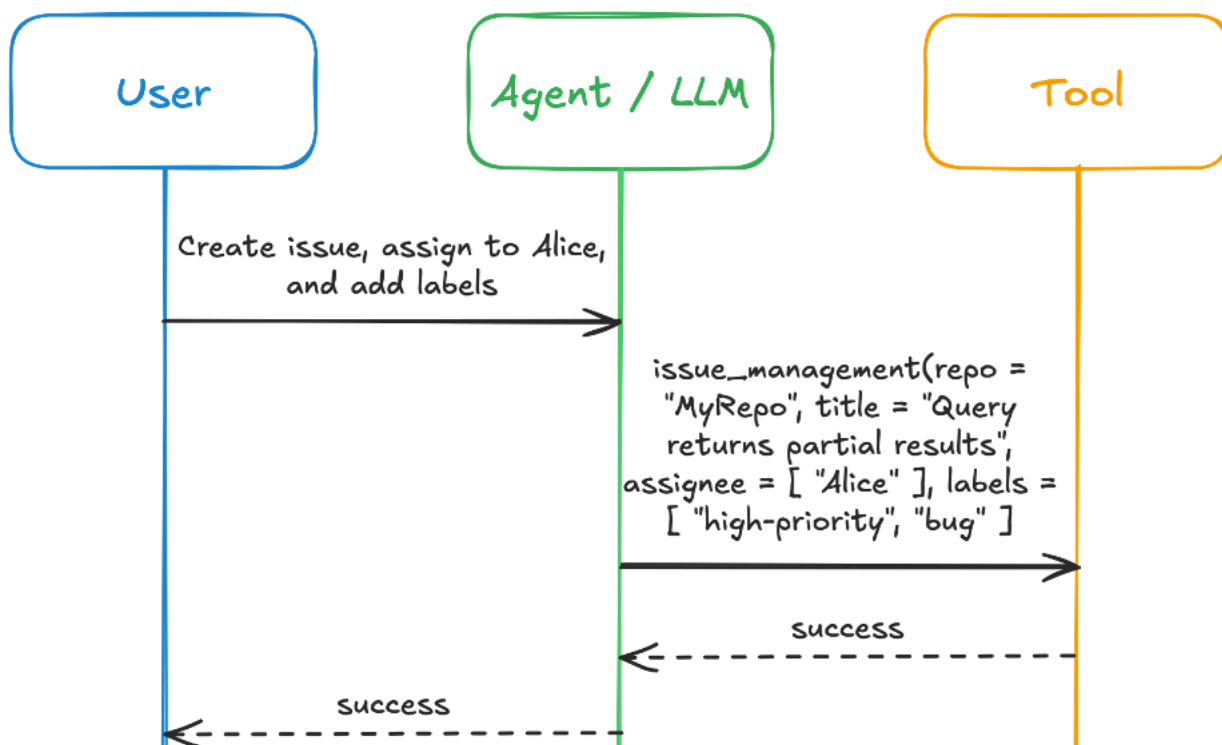


Bei diesem Ansatz werden dem LLM bei jedem Aufruf die Systemaufforderung und jede registrierte Tooldefinition zur Verfügung gestellt. Dadurch wird zusätzlicher Kontext verbraucht und es kommt zu

einer Latenzeinbuße, da jeder Tool-Aufruf einen individuellen Aufruf an das LLM darstellt. Dies erhöht auch die Komplexität der Fehlerbehandlung innerhalb des Workflows.

Coarse-grained

Ein grobkörniger oder workflow-orientierter Ansatz wären Tools, die auf Workflows ausgerichtet sind. Das Tool konzentriert sich auf die durchgängige Benutzerabsicht und nicht auf die API-Struktur. Anstatt eines Tools pro API haben Sie ein Tool, das deterministisch viele APIs aufruft. Mit dem vorherigen Git-Repository-Beispiel könnten Sie ein `create_and_setup_issue` Tool erstellen, das einmal vom Agenten aufgerufen wird. Die Tool-Implementierung erstellt das Problem, fügt Labels hinzu und weist es einem Benutzer zu, basierend auf den Parametern, die dem Tool zur Verfügung gestellt wurden. Die folgende Abbildung zeigt, wie ein grober Ansatz dieselbe Aufforderung verarbeiten würde.



Dieser Ansatz zeigt, dass die gesamte Komplexität vor der LLM-Ebene verborgen bleibt. Wenn die Orchestrierungslogik in die Toolimplementierung eingebettet ist, werden alle sequentiellen Schritte, die Protokollierung, die Wiederholungslogik, die Schutzschalter und die Ratenbegrenzung deterministisch im Tool ausgeführt. Der workflow-gesteuerte Ansatz erleichtert es dem LLM, das richtige Tool mit den richtigen Parametern aufzurufen. Es ist wichtig zu beachten, dass einige APIs möglicherweise bereits Workflow-Intent bereitstellen, z. B. die Amazon EC2 RunInstances EC2-

API. In diesen Fällen kann ein Tool pro API das Workflow-orientierte Design bieten, das Sie sich wünschen.

Tools können jedoch auch zu grobkörnig werden. Wenn Ihr einzelnes Workflow-Tool versucht, zu viele Dinge zu tun und viele mögliche Parameter hat, kann es für den LLM schwierig sein, zu überlegen, wie das Tool richtig verwendet werden soll. Es kann auch zu Herausforderungen bei der Parameterauswahl und Fehlerbehandlung führen. Daher muss bei der Entwicklung von Tools ein Gleichgewicht gefunden werden, das den Benutzerabsichten entspricht und zu wenig oder zu viel Funktionalität in einem einzigen Tool vermeidet. Wir empfehlen Ihnen, Tools so zu entwickeln, dass sie komplette Benutzerworkflows berücksichtigen und Vorgänge, die häufig zusammen auftreten (z. B. drei oder mehr API-Aufrufe), bündeln. Wir empfehlen außerdem, Tools zu zerlegen, die acht oder mehr Parameter überschreiten oder mehrere unterschiedliche Benutzerabsichten berücksichtigen. Testen Sie mit echten Eingabeaufforderungen, um sicherzustellen, dass die Agenten jedes Tool korrekt verwenden können.

Wenn Sie über komplexe und dynamische Workflows verfügen, die sich nicht einfach als deterministisches Tool zusammenfassen lassen, sollten Sie das Agent-als-Tool -Muster in Betracht ziehen. Anstatt dass Ihr primärer Agent versucht, komplexe Aufgaben in einem Workflow zu orchestrieren, kann ein spezialisierter Agent als Tool fungieren. Mit diesen Tools können erweiterte Entscheidungsprozesse und Verzweigungen implementiert werden. Außerdem können sie Fehler und Wiederholungslogik behandeln, die in deterministischem Code nicht einfach zu handhaben ist. Dies ähnelt dem [Agent2Agent](#) (A2A) -Protokoll, unterscheidet sich jedoch von diesem. Das A2A-Protokoll ist komplementär und bietet Interoperabilität und Zusammenarbeit zwischen Agenten in jedem agentischen Framework.

Wir empfehlen Ihnen, mit Ihrer Workflow-Analyse zu beginnen, indem Sie Ihre gängigsten Benutzerworkflows abbilden, um die Kernfunktionen zu ermitteln, die jeder Agent benötigt. Auf diese Weise wird Ihr minimales praktikables Toolset festgelegt. Aufgrund unserer Erfahrung mit der Entwicklung von MCP-Servern in großem Maßstab empfehlen wir die folgenden Vorgehensweisen. Wenn diese Praktiken miteinander in Konflikt geraten, sollten Sie Benutzerabsicht und Arbeitsablauf priorisieren.

Bewährte Methoden für die Festlegung des Umfangs von MCP-Tools

- Denken Sie an Anwenderberichte und bündeln Sie gängige Abläufe — Tools sollten sich direkt den gesamten Benutzerinteraktionen zuordnen lassen, anstatt die Orchestrierung mehrerer Operationen zu erfordern. Wenn Workflows in der Regel drei oder mehr separate Aufrufe erfordern, kombinieren Sie sie zu einem einzigen Tool. Dies reduziert die kognitive Belastung des LLM,

minimiert die Anzahl der Tool-Aufrufe, reduziert den Kontextverbrauch und die Latenz, die für die Ausführung von Aufgaben erforderlich sind, und verbessert die Genauigkeit und Latenz.

- Beschränken Sie die Parameter auf acht oder weniger — Wenn ein Werkzeug acht Parameter überschreitet, teilen Sie es in mehrere Tools auf. LLMs haben mit zunehmender Komplexität Probleme mit der Parameterauswahl.

Note

Wenn für Bündelungsvorgänge mehr als acht Parameter erforderlich sind, sollten Sie der Bündelung Vorrang vor der Anzahl der Parameter einräumen, da die Vereinfachung des Workflows wertvoller ist als strenge Parametergrenzwerte.

- Getrennte Lese- und Schreibvorgänge — Stellen Sie verschiedene Tools zum Lesen und Ändern von Daten bereit. Diese Trennung macht deutlich, wann Agenten potenziell zerstörerische Operationen ausführen, ermöglicht unterschiedliche Autorisierungsrichtlinien und reduziert das Risiko unbeabsichtigter Änderungen bei der Datenerfassung.
- Stellen Sie sinnvolle Standardeinstellungen bereit — Entwerfen Sie die Tools so, dass das LLM nur die Parameter angeben muss, die für die jeweilige Anfrage spezifisch sind. Standardeinstellungen reduzieren die Komplexität der Parameter und verbessern die Genauigkeit der Werkzeugauswahl, indem sie die Informationen minimieren, über die das LLM nachdenken muss.
- Bevorzugen Sie die deterministische Ausführung — Machen Sie die Ausführung und Ausgabe des Werkzeugs nach Möglichkeit deterministisch. Deterministische Tools sind zuverlässiger und einfacher zu testen. Für komplexe Workflows, die intelligente Orchestrierung, Verzweigungslogik oder erweiterte Fehlerbehandlung erfordern und die sich in deterministischem Code nicht einfach verwalten lassen, sollten Sie die Verwendung spezialisierter Agenten als Tools in Betracht ziehen. Verwenden Sie dieses Muster jedoch selektiv, da es die Komplexität erhöht.

Werkzeugdefinitionen

Wenn ein LLM eine Anfrage erhält, die es nicht direkt bearbeiten kann, überprüft es die verfügbaren Tools, um die Anfrage zu bearbeiten. Das LLM wählt Tools auf der Grundlage seines semantischen Verständnisses der Namen und Beschreibungen der bereitgestellten Tools sowie aller in der Aufforderung enthaltenen Anweisungen aus. Es erstellt dann Eingaben auf der Grundlage des definierten Eingabeschemas und erwartet eine Ausgabe auf der Grundlage des Ausgabeschemas. Daher ist die Erstellung beschreibender Werkzeugdefinitionen und validierter Eingabe- und Ausgabeschemas von entscheidender Bedeutung, um dem LLM bei der effektiven Auswahl von Tools

zu helfen. Im Allgemeinen gibt es zwei Ansätze für die Erstellung dieser Dokumentation: den Ansatz der Toolspezifikation und den Docstring-Ansatz.

Ansatz zur Werkzeugspezifikation

Der empfohlene Ansatz besteht darin, sich bei der Definition des Tools direkt an die [MCP-Werkzeugspezifikation](#) zu halten. Das folgende Beispiel wird mit dem Tool-Decorator von [Strands Agent](#) gezeigt:

```
@tool(
    name = "search_website",
    description = "This tool searches the provided website for semantic matches to the
query provided",
    inputSchema = {
        "json": {
            "type": "object",
            "properties": {
                "url": {
                    "type": "string",
                    "description": "The url of the website to load and search."
                },
                "query": {
                    "type": "string",
                    "description": "The content you want to try and match in the website."
                }
            }
        },
        "required": ["url", "query"]
    },
    outputSchema = {
        "json": {
            "type": "object",
            "properties": {
                "results": {
                    "type": "array",
                    "items": {
                        "type": "string"
                    }
                }
            }
        }
    }
)
def search_website:
```

...

Durch die Verwendung von Standardfeldern wie, `namedescription`, und wird `outputSchema` sichergestellt `inputSchema`, dass jedes Tool über eine konsistente Dokumentation verfügt, die sowohl der LLM als auch die Menschen verstehen können. Jedes Tool sollte diese Felder mindestens definieren und optional einen Titel und Anmerkungen bereitstellen, bei denen es sich um optionale Hinweise zum Verhalten des Tools handelt. Verwenden Sie nach Möglichkeit Enums für Parameterwerte, um es dem LLM zu erleichtern, die richtigen Optionen auszuwählen. Aufzählungen eignen sich am besten für endliche Mengen wie Status- oder Prioritätswerte, eignen sich jedoch nicht für Freiformtext, dynamische Werte, willkürliche Zahlen oder Ressourcenkennungen. Geben Sie in diesen Fällen stattdessen klare Beschreibungen und Beispiele an. Geben Sie nach Möglichkeit auch einen Standardwert an, damit der LLM nicht erraten muss, was die richtige Option ist. Denken Sie daran, dass Tooldefinitionen bei jedem Aufruf in der LLM-Eingabeaufforderung enthalten sind und neben den Systemanweisungen und dem Konversationsverlauf auch Platz im Kontextfenster beanspruchen.

Docstring-Ansatz

Ein anderer Ansatz, wenn Sie Ihre Werkzeuge in Python schreiben, besteht darin, Docstrings zu verwenden, um die Beschreibung, Verwendung und Ausgabe des Werkzeugs bereitzustellen. Im Folgenden finden Sie ein Beispiel für diesen Ansatz:

```
def search_website(url: str, query: str) -> list:

    """
    This tool loads the specified website and then attempts to find content that
    matches the provided query through semantic search. It provides back a list of strings
    that are the sentences that match the query.
    Args:
        url: the website url to load
        query: the content you want to semantically match in the website
    """
```

Docstrings erzwingen weder ein Schema noch ein standardisiertes Format. Die Verwendung dieses Ansatzes kann zu inkonsistenten Ergebnissen führen, je nachdem, wie Toolentwickler die einzelnen Tools dokumentieren. Wenn Sie diesen Ansatz verfolgen, ist es wichtig, einen unternehmensweiten Standard zu definieren und durchzusetzen.

Bewährte Methoden für Definitionen von MCP-Tools

- Halten Sie sich an die MCP-Werkzeugspezifikation — geben Sie `outputSchema` Felder `namedescription`, `inputSchema`, und für jedes Werkzeug an. Verwenden Sie für Python-Implementierungen [Pydantic-Modelle](#), um Inline-Dokumentation durch Felddesreibungen, automatische Typvalidierung und eingeschränkte Werte durch Enums bereitzustellen. Dadurch dokumentieren sich Schemas selbst und das LLM-Verständnis gültiger Parameteroptionen verbessert sich.
- Schreiben Sie Beschreibungen als Eingabeaufforderungen — Toolbeschreibungen sind Anweisungen, die die LLM-Entscheidungsfindung erleichtern. Geben Sie die wesentlichen Bestandteile des Tools an (was das Tool tut), wann es verwendet werden soll (Muster oder Szenarien der Benutzerabsichten), den Kontext der Ausgabe (wofür die Ausgabe verwendet wird), Parameter und Fehlerbedingungen.
- Geben Sie konkrete Beispiele an — Die Einbeziehung von Workflow-Beispielen mit tatsächlichen Werten ist die effektivste Methode, um LLMs bei der korrekten Verwendung des Tools zu unterstützen.
- Dokumentieren Sie Abhängigkeiten explizit — Geben Sie Voraussetzungen, nummerierte Sequenzen, Statusänderungen und Folgemaßnahmen an.

Erkennung von Tools

Es gibt drei Ansätze für die Erkennung und Registrierung von Tools in Ihrem Agenten auf MCP-Servern: statische Definition, dynamische Erkennung und Suchfunktion.

Statische Definition

Zunächst können Sie die verfügbaren Tools direkt im Agentencode statisch definieren. Bei diesem Ansatz definieren Sie für jedes vom MCP-Server bereitgestellte Tool, auf das ein MCP-Client zugreift, ein Remote-Tool (ein clientseitiges Referenzobjekt in einem Framework wie dem Strands Agent SDK). Das folgende Beispiel verwendet streambaren HTTP-Transport:

```
from mcp.client.streamable_http import streamablehttp_client
from strands import Agent
from strands.tools.mcp import MCPClient

streamable_http_mcp_client = MCPClient(
    lambda: streamablehttp_client("https://mcp1:8000/mcp")
```

```
)

reverse_text = RemoteTool(
    name="reverseText",
    client=streamable_http_mcp_client
)

agent = Agent(tools=[reverse_text])
```

Durch die Registrierung einzelner Tools können Sie bei der Auswahl der Tools, die Sie dem LLM zur Verfügung stellen, sehr selektiv vorgehen, wodurch der Umfang des verwendeten Kontextfensters minimiert wird. Der Nachteil besteht darin, dass die Namen der verfügbaren Tools bekannt sein müssen. Außerdem kann es instabil sein, wenn sich die verfügbaren Tools auf dem MCP-Server ändern.

Dynamische Erkennung

Der nächste Ansatz besteht darin, die dynamische Erkennung zu verwenden und alle verfügbaren Tools beim Agenten zu registrieren. Bei diesem Ansatz wird der Kontext linear verbraucht, wenn dem MCP-Server mehr Tools hinzugefügt werden. Im Folgenden finden Sie ein Beispiel für diesen Ansatz:

```
from mcp.client.streamable_http import streamablehttp_client
from strands import Agent
from strands.tools.mcp import MCPClient

streamable_http_mcp_client = MCPClient(
    lambda: streamablehttp_client("https://mcp1:8000/mcp")
)

with streamable_http_mcp_client:
    tools = streamable_http_mcp_client.list_tools_sync()
    agent = Agent(tools=tools)
```

Stellen Sie sich ein Szenario vor, in dem eine typische Werkzeugdefinition etwa 250 bis 500 Token (einschließlich Name, Beschreibung und Schema) verbraucht. Die Registrierung von 20 Tools würde 5.000 bis 10.000 Token Ihres Kontextfensters verbrauchen. Wenn Sie über eine kleine Anzahl von MCP-Servern verfügen und die Kontrolle über die Anzahl der Tools haben, ist diese Option am einfachsten zu implementieren. Wenn jedoch erwartet wird, dass die Liste der Tools wächst, kann dies zu unbemerkten Problemen mit der Kontextverwaltung bei Ihren Agenten führen. Eine alternative Variante dieses Ansatzes besteht darin, beim Aufrufen einen Toolfilterparameter

zu verwenden `list_tools`, wie ihn das [Strands Agents SDK](#) bietet, um die Anzahl der Tools zu reduzieren, die beim Agenten registriert sind.

Suchfunktion

Die dritte Möglichkeit besteht darin, während der Laufzeit eine Suchfunktion zu verwenden, um relevante Tools zu finden. Sie listen alle verfügbaren Tools auf Ihrem MCP-Server auf und führen dann basierend auf der Benutzeraufforderung eine semantische Suche über diese Tools durch. Anschließend werden die resultierenden Tools bei Ihrem Agenten registriert. [Amazon Bedrock AgentCore Gateway](#) bietet eine [native semantische Suchfunktion](#), die die Implementierung dieser Art von Lösung erleichtern kann.

Bewährte Methoden für die Entdeckung von MCP-Tools

- Beibehaltung des Kontextfensters — Wählen Sie einen Ansatz zur Erkennung und Registrierung von Tools, bei dem Ihr Kontextfenster so weit wie möglich erhalten bleibt.
- Nutzen Sie Funktionen zum Filtern von Tools oder zur semantischen Suche — Stellen Sie dem LLM dynamisch eine Reihe von Tools zur Auswahl zur Verfügung, wodurch die Genauigkeit und Effektivität bei der Auswahl des richtigen Tools verbessert wird. Die Toolfilterung kann anhand von Werkzeugnamen (exakte Übereinstimmung oder Muster), Werkzeugbeschreibungen (semantischer Abgleich) oder Domänen- oder Kategorie-Tags erfolgen. Die semantische Suche ist besonders effektiv, wenn es darum geht, die Absicht der Benutzer mit den Werkzeugbeschreibungen abzugleichen. Beide Ansätze reduzieren die Nutzung von Kontextfenstern.

Organisation der Tools

Die richtigen Tools zu finden und sicherzustellen, dass das LLM sie effektiv einsetzen kann, ist einer der wichtigsten Aspekte einer effektiven Toolentwicklung. Wenn Sie mit der Entwicklung von MCP-Servern beginnen, benötigen Sie eine Strategie, die Folgendes bestimmt:

- Wie viele Tools gehören zu einem MCP-Server
- Welche Tools sollten nicht auf demselben MCP-Server installiert werden
- Wie benennt man Tools, um sie durchsuchbar zu machen und Namenskollisionen zu vermeiden (verschiedene Tools mit demselben Namen)
- Wie dokumentiert man die Tools und den MCP-Server, damit sie vom LLM einfach verwendet werden können

Die Namespace-Organisation ist ein Entwurfsmuster, das Kollisionen von Werkzeugnamen verhindert, Funktionen gruppiert und eine effiziente Werkzeugidentifikation durch LLMs ermöglicht. Das Muster legt eine strukturierte Kategorisierung fest, die eher mit organisierten Speichersystemen als mit einer unstrukturierten Akkumulation vergleichbar ist. Wir empfehlen das Muster Domain-Nomen-Verb für die Benennung von Tools. Zum Beispiel, `github_issue_create`, `github_issue_list`, `github_issue_update`, `github_pullrequest_create` oder `github_pullrequest_list`, `github_pullrequest_merge`. Der Vorteil dieses Musters wird deutlich, wenn man das alphabetische Sortierverhalten untersucht. Wenn Tools alphabetisch aufgelistet sind, gruppieren sich alle problembezogenen Operationen (`create`, `update`), `list`, gefolgt von Pull-Request-Vorgängen (`create`, `list`, `merge`). Das Substantiv (Ressourcentyp) dient als organisatorische Grenze. Diese Struktur erleichtert sowohl das Scannen von LLM-Tools als auch die Navigation in der menschlichen Dokumentation, da verwandte Funktionen auf natürliche Weise gruppiert werden.

Der MCP-Server sollte auf Domänenebene begrenzt sein, kann aber aufgrund der Aufgabentrennung für die von ihm bereitgestellten Funktionen unterteilt werden. Beispielsweise könnten Sie separate MCP-Server für Schreib- und Lesevorgänge in einer Datenbank haben. Um diese Trennung durchzusetzen, empfiehlt es sich, Leitplanken auf Agentenebene zu implementieren, die je nach Benutzerabsicht und -berechtigungen einschränken, auf welche MCP-Server zugegriffen werden kann. Dies kann durch eine Kombination der folgenden Maßnahmen erreicht werden:

- Bedingtes Laden des Servers — Der MCP-Server mit Schreibschutz wird nur geladen, wenn der Agent Lesevorgänge in der Benutzereingabe erkennt.
- Permission-based Filterung — Verwenden Sie die Benutzerautorisierung, um nur den entsprechenden MCP-Servern Zugriff zu gewähren.

Schließlich sollten Sie eine Obergrenze für die Anzahl der von einem MCP-Server bereitgestellten Tools festlegen. Machen Sie keine Annahmen darüber, wie Agenten Ihren MCP-Server verwenden werden. Sie listen möglicherweise naiv alle verfügbaren Tools auf und stellen sie alle dem LLM zur Verfügung. Wenn Sie mehr als 50 Tools auf einem einzelnen Server haben, sollten Sie erwägen, ihn auf mehrere Server aufzuteilen.

Bewährte Methoden für die Organisation von MPC-Tools

- Verwenden Sie den Standard zur Benennung von Domänen, Substantiven und Verben für Tools — Implementieren Sie Strategien, um Namenskollisionen sowohl auf MCP-Servern als auch auf Agenten zu verhindern.

- Legen Sie eine Obergrenze fest — Beschränken Sie die Anzahl der Tools auf einem einzelnen MCP-Server.
- MCP-Server aufteilen — Verwenden Sie die Aufgabentrennung, um MCP-Server in logische Gruppen zu unterteilen.

MCP-Hosting-Strategie

Durch die Zusammenfassung der verfügbaren Tools auf MCP-Servern wird Ihre Agentenentwicklung von den verfügbaren Tools entkoppelt. Dies führt zu den Herausforderungen, die sich daraus ergeben, wo Sie Ihren MCP-Server hosten und wie die Tools auf diesen Servern organisiert sind.

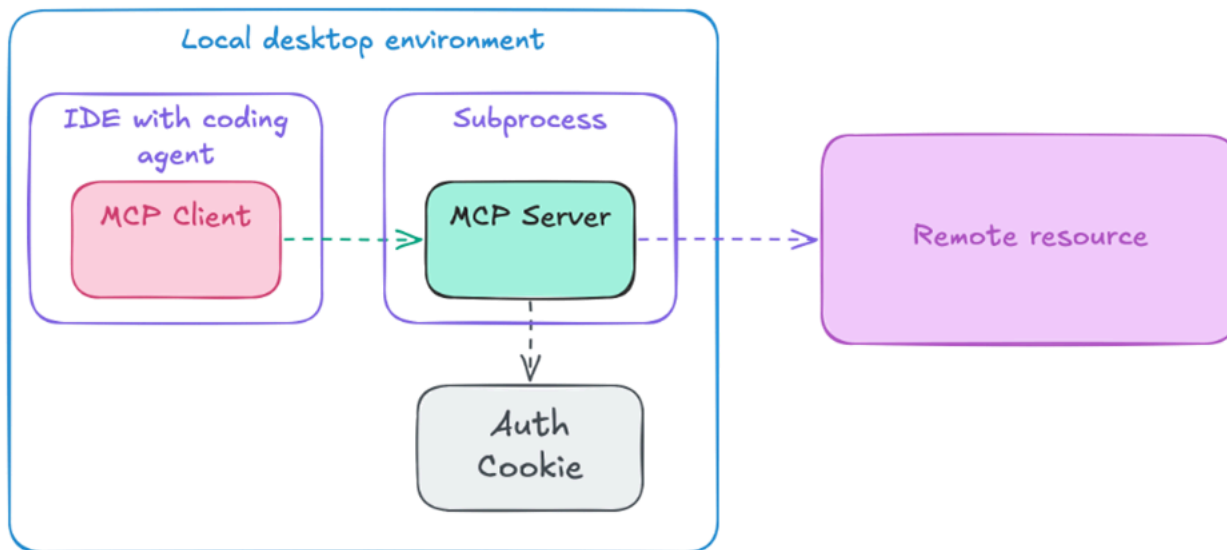
Hosting-Ansätze

Es gibt drei Möglichkeiten, Ihre MCP-Server zu hosten: Sie können sie lokal auf einem Endbenutzercomputer ausführen, sie remote hosten oder sie über ein MCP-Gateway hosten. Jede Option hat Vor- und Nachteile.

Lokales Hosting

Beim lokalen Hosting wird der MCP-Server als Unterprozess auf Ihrem lokalen Computer zusammen mit dem Agenten ausgeführt, der JSON-RPC über standardmäßige Eingabe- und Ausgabestreams mit dem Server kommuniziert. Dieser Ansatz erfordert keine Authentifizierung zwischen dem Client und dem Server. Tools können mit lokalen Anwendungen und Dateien interagieren, lokal gespeicherte Anmeldeinformationen verwenden und übernehmen den Netzwerkzugriff des lokalen Computers des Benutzers. Dies ist das einfachste Hosting-Muster und bietet mehrere Vorteile.

Viele Kunden beginnen mit MCP und verwenden lokale Server. Sie ermöglichen es Ingenieuren, eine Vielzahl von Problemen in ihrer lokalen Umgebung schnell zu wiederholen und zu lösen. Stellen Sie sich einen MCP-Server vor, der eine Verbindung zu einem Git-Repository herstellt, das der Programmierassistent eines Ingenieurs verwendet. Es ist sehr sinnvoll, den MCP-Server lokal zu halten, da er die eindeutigen Anmeldeinformationen des Technikers für den Zugriff auf das Repository verwenden kann und kein zusätzlicher Netzwerkaufruf zu einem entfernten MCP-Server hinzugefügt wird. Die folgende Abbildung zeigt einen lokal gehosteten MCP-Server, der mit einem Codierungsagenten in einer IDE verwendet wird.



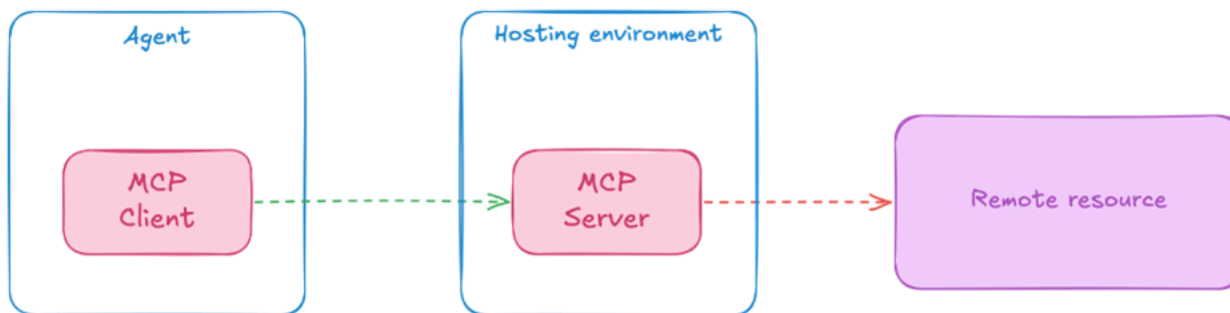
Bei diesen Bereitstellungstypen müssen Sie berücksichtigen, wie die MCP-Server entwickelt und verteilt werden. Die meisten Kunden entwickeln eine MCP-Registrierung, in der Server von Endbenutzern registriert und heruntergeladen werden können. Es ist einer Container-Registry sehr ähnlich, in der ein Benutzer nach bestimmten Funktionen suchen und die MCP-Server finden kann, die seinen Anforderungen entsprechen.

Es gibt öffentliche MCP-Register, wie z. B. [Official MCP Registry](#), und es gibt privat gehostete Registries. Organizations richten ihre MCP-Registrierungsstrategie in der Regel an bestehenden Richtlinien rund um die Verteilung von Open-Source-Software, Container-Registrierungen und internes Paketmanagement aus. Sie sollten Faktoren wie Sicherheitsscans, Genehmigungsworkflows und Compliance-Anforderungen berücksichtigen.

Lokales Hosting bringt jedoch betriebliche Herausforderungen mit sich, die Unternehmen berücksichtigen sollten. Zunächst müssen Endbenutzer MCP-Server unabhängig voneinander ermitteln, herunterladen und konfigurieren. Dies kann die Komplexität der ersten Schritte mit jedem einzelnen MCP-Server erhöhen, den sie lokal verwenden. Zweitens können Sie den Lebenszyklus des MCP-Servers nicht kontrollieren, was bedeutet, dass Benutzer möglicherweise weiterhin veraltete Versionen mit Sicherheitslücken oder fehlenden Funktionen lokal ausführen. Dies kann die Einhaltung von Compliance-Anforderungen erschweren. Einige IDEs und CLI-Tools, wie [Kiro](#), ermöglichen es Unternehmen, zu [verwalten und zu kontrollieren, welche MCP-Tools verfügbar sind](#), wodurch Konsistenz und Sicherheit zwischen den Teams gewährleistet werden.

Remote-Hosting

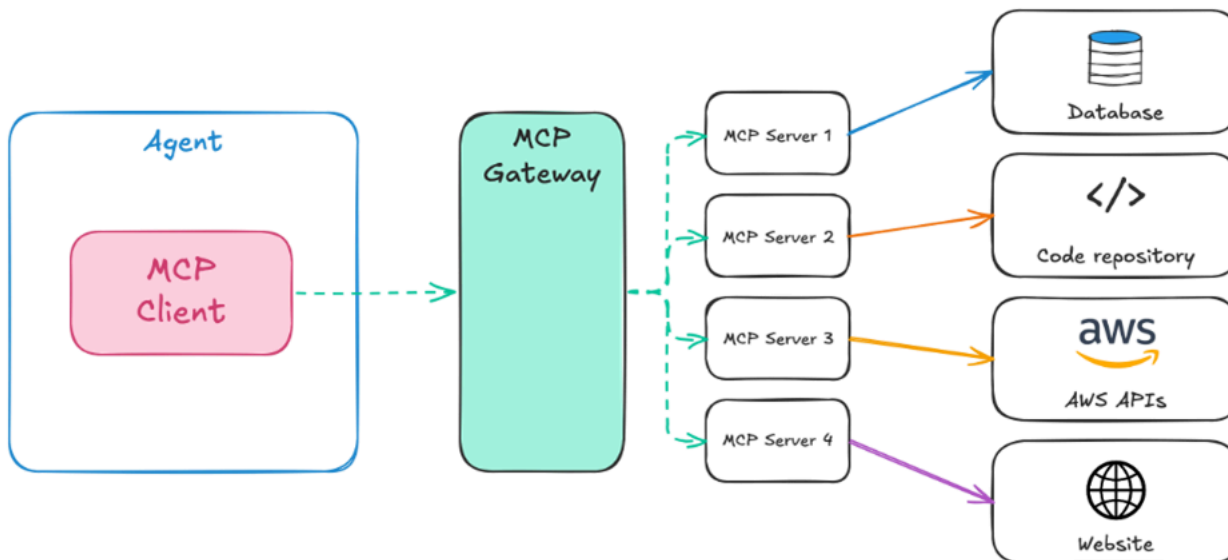
Die zweite Option besteht darin, Remote-MCP-Server zu hosten, auf die über HTTP oder HTTPS zugegriffen wird. Dies ermöglicht den Zugriff auf jeden mit dem Netzwerk verbundenen Client. Mithilfe von Remote-Hosting können Sie den Zugriff auf MCP-Ressourcen und -Funktionen zentral steuern, Authentifizierung und Autorisierung implementieren und die Versionierung und Aktualisierung der MCP-Serverlogik steuern. Für das Remote-Hosting ist weiterhin die Verwendung einer MCP-Registrierung erforderlich, damit Endbenutzer die MCP-Server ermitteln können, die sie mit ihrem Agenten verwenden möchten. Die folgende Abbildung zeigt den Remote-Hosting-Ansatz.



Aus Sicht der Agentenentwicklung ist die Erfahrung ähnlich, unabhängig davon, ob der MCP-Server lokal oder remote ist. Die wichtigste Änderung ist die Implementierung von Authentifizierung und Autorisierung, was sowohl den Zugriff des Agenten auf den MCP-Server als auch den Zugriff des Servers auf externe Ressourcen einschließt. Die Implementierung von Remote-MCP-Servern muss sorgfältig geplant werden, um den Mehrmandantenzugriff und die Rechteverwaltung zu berücksichtigen. Das Kapitel zur [MCP-Governance-Strategie](#) enthält weitere Informationen zu Überlegungen zur Authentifizierung und Autorisierung.

MCP-Gateway

Die letzte Option ist die Verwendung eines MCP-Gateways. MCP-Gateways fungieren als zentraler Proxy zwischen MCP-Clients und -Servern und orchestrieren den Zugriff auf die registrierten MCP-Server. Ohne Gateway muss jeder Agent jeden Remote-MCP-Server registrieren, den er möglicherweise verwenden möchte. Ein Gateway ermöglicht es dem Agenten, eine Verbindung zu einem einzelnen Endpunkt herzustellen, der die Authentifizierung, Autorisierung, das Routing und die Protokollübersetzung verwaltet. Neue MCP-Server und -Tools können dynamisch hinzugefügt und dem Agenten sofort zur Verfügung gestellt werden. Die folgende Abbildung zeigt den MCP-Gateway-Ansatz.



Einige Gateway-Lösungen, wie [Docker MCP Gateway](#), verwalten auch den Lebenszyklus der MCP-Server und starten Server bei Bedarf auf Abruf. MCP-Gateways wie [Amazon Bedrock AgentCore Gateway](#) können auch bei der Verwaltung der Tool-Erkennung helfen, indem sie [native semantische](#) Suchfunktionen bereitstellen. Dies bietet Agenten einen einzigen Endpunkt, über den sie sich mit einem MCP-Client verbinden können, und hilft, die Nutzung ihres Kontextfensters zu optimieren. Das Ergebnis sind einfache Agenten, die MCP-Tools auswählen und effektiv nutzen können. Es birgt jedoch ähnliche identitätsbezogene Herausforderungen wie der Ansatz mit Remote-MCP-Servern.

Bewährte Methoden für das Hosten von MCP-Servern

- Das Spektrum der Hosting-Optionen ist nicht einheitlich. Ein Großteil der Nutzung von MCP-Servern erfolgt heute lokal.
- Wenn Sie anfangen, MCP-Remote-Server zu verwenden, sollten Sie vor allem auf eine konsistente Authentifizierung und Autorisierung für den MCP-Server achten und darauf, wie der MCP-Server die Authentifizierung und Autorisierung für nachgelagerte Ressourcen durchführt.
- MCP-Gateways vereinfachen die Konnektivität sowie die Authentifizierung und Autorisierung für das Hosten mehrerer Remote-MCP-Server. Sie bieten auch Funktionen zur Verbesserung der Verwaltung von Kontextfenstern, indem nach geeigneten Tools gesucht wird.

MCP-Verwaltungsstrategie

Die andere wichtige Funktion, die MCP Unternehmen bietet, ist die Unterstützung zentralisierter Governance. Ihre MCP-Governance-Strategie sollte sich mit der Authentifizierung und Autorisierung sowohl der MCP-Server als auch der Ressourcen befassen, auf die sie zugreifen. Sie sollte sich auch mit der Ratenbegrenzung befassen, um nachgelagerte Ressourcen zu schützen, betriebliche Kennzahlen zur Überwachung der Nutzung und Leistung der Tools sowie die Verwaltung der Bereitstellung und Verteilung von MCP-Servern.

Authentifizierung und Autorisierung

Einer der wichtigsten Bestandteile Ihrer Authentifizierungs- und Autorisierungsstrategie ist die Verwaltung des Downstream-Ressourcenzugriffs von MCP-Servern aus. Wenn ein Benutzer einen Agenten aufruft, werden Authentifizierung und Autorisierung durchgeführt, um sicherzustellen, dass der Benutzer berechtigt ist, den Agenten anzurufen. Anschließend orchestriert der Agent den Aufruf bestimmter Tools auf MCP-Servern. Sie müssen für jedes Tool entscheiden, wie der Zugriff autorisiert werden soll.

Eine Option ist die machine-to-machine Autorisierung, bei der keine Zustimmung oder Interaktion des Benutzers erforderlich ist. Beispielsweise verwendet ein zeitbasierter Agentenaufruf einen MCP-Server, um Protokolle von einer Anwendung zu sammeln und zu analysieren. In diesem Szenario ist der Agent vorab autorisiert, auf die angegebenen Daten zuzugreifen. Die zweite Option ist der vom Benutzer delegierte Zugriff, bei dem ein Benutzer seine Zustimmung zum Zugriff auf benutzerspezifische Daten und Ressourcen erteilt.

Die folgende Tabelle zeigt Authentifizierungs- und Autorisierungsmuster.

Faktor	Vom Benutzer delegierter Zugriff	Machine-to-machine
Eigentum an Daten	Benutzerspezifische Autorisierung von Daten	System- oder organisationsweite Daten
Interaktion mit dem Benutzer	Der Benutzer ist anwesend und kann zustimmen	Keine Benutzerinteraktion

Zeitpunkt des Vorgangs	Interaktiv oder in Echtzeit	Hintergrund, geplant oder stapelweise
Umfang der Genehmigung	Die Berechtigungen variieren je nach Benutzer	Konsistente Berechtigungen auf Agentenebene

Der vom Benutzer delegierte Zugriff erfordert eine sorgfältige Implementierung und sollte zusammen mit Ihrem Sicherheitsteam entwickelt werden. Agenten müssen in der Lage sein, zu beurteilen, welche Tools ein LLM ausgewählt hat und ob für sie eine zusätzliche Autorisierung erforderlich ist. MCP-Tools müssen Beschreibungen enthalten, aus denen hervorgeht, welche Authentifizierungs- und Autorisierungsanforderungen sie haben und wo Zugriffstoken abgerufen werden können. Client-Anwendungen müssen Zwischenauthentifizierungsanforderungen unterstützen, und der MCP-Client muss die abgerufenen Anmeldeinformationen bei jedem Tool-Aufruf an den Agenten zurücksenden.

Sie sollten sicherstellen, dass MCP-Tools immer über eigene Token für den Zugriff auf externe Funktionen verfügen und dass der Zugriff protokolliert und geprüft wird. Benutzeranmeldedaten sollten nicht über Ihr Agentensystem weitergegeben werden. Beispielsweise sollten Ihre MCP-Server nicht dasselbe Token für den Zugriff auf Daten verwenden, mit dem der Agent aufgerufen wurde. Downstream-Aufrufe sollten speziell generierte Token mit explizitem Gültigkeitsbereich verwenden. Dies trägt dazu bei, zusätzliche Schutzmaßnahmen bereitzustellen, um einen unbeabsichtigten Datenzugriff im Rahmen von Aktionen zu verhindern. Auf diese Weise kann auch verhindert werden, dass Halluzinationen unbeabsichtigte Folgen haben. Stellen Sie sich vor, ein Benutzer mit vollen Administratorrechten bittet einen Agenten, eine Produktionsdatenbank zur Verwendung in der Vorproduktion zu klonen. Dazu benötigt der Benutzer lediglich CREATE Berechtigungen READ. Nehmen wir an, das LLM halluziniert und glaubt, dass es im Rahmen dieser Anfrage die alte Datenbank bereinigen muss. Wenn die Anmeldeinformationen des Benutzers wiederverwendet werden, wäre dies wahrscheinlich erfolgreich, da die ursprünglichen Anmeldeinformationen des Benutzers über Berechtigungen verfügen. DELETE Wenn der MCP-Server stattdessen ein Token verwendet, das bewusst auf den Umfang der Daten für die Anfrage beschränkt ist READ und nur die CREATE Rechte und Rechte hat, würde der Versuch, die Produktionsdatenbank zu löschen, fehlschlagen.

Sie können [Amazon Bedrock AgentCore Identity](#) verwenden, um diese Muster zu implementieren. Stellen Sie sicher, dass Sie bewusst entscheiden, ob die Berechtigungen zum Auflisten und Aufrufen von Tools, die von einem MCP-Server gehostet werden, auch Berechtigungen für die externen Funktionen beinhalten, die der MCP-Server bereitstellt. Dieser Identitätsfluss vom MCP-Server zur Ressource und zurück zum Benutzer hängt von der Art des verwendeten Authentifizierungs- und

Autorisierungsdienstes ab. Sie müssen entscheiden, wie dies in großem Umfang für Ihre MCP-Server gehandhabt wird.

Implementieren Sie bei der Gestaltung Ihrer Authentifizierungs- und Autorisierungsmuster Mechanismen zur Token-Isolierung, die für jedes Tool, auf das zugegriffen wird, unterschiedliche Zugriffstoken abrufen. Verwenden Sie Token nicht erneut zwischen Tools und Servern. AgentCore Identity bietet diese Fähigkeit zur Token-Isolierung. Es verwaltet automatisch sowohl Workload-Token (zur machine-to-machine Authentifizierung) als auch Benutzertoken (für vom Benutzer delegierten Zugriff), um eine korrekte Trennung sicherzustellen und eine Eskalation von Berechtigungen zu verhindern. Dies ist besonders wichtig, wenn Remote-MCP-Server oder MCP-Gateways integriert werden.

Bewährte Methoden für die MCP-Authentifizierung und -Autorisierung

- **Token-Trennung** — Geben Sie keine Inhaber-Token von Anrufern an nachgelagerte Dienste weiter. Stellen Sie sicher, dass das Feld `aud` (Audience) mit dem Server übereinstimmt, der das Token empfängt. Der Zielgruppenanspruch gibt an, für welchen Dienst das Token bestimmt ist, und verhindert so die unbefugte Wiederverwendung von Token auf verschiedenen MCP-Servern.
- **Wählen Sie einen Zugriffsansatz** — Wählen Sie für jedes Tool, das Ihre MCP-Server bereitstellen, zwischen machine-to-machine und vom Benutzer delegiertem Zugriff. Erwägen Sie, Tools auf demselben MCP-Server zu gruppieren, die dasselbe Authentifizierungsmuster verwenden.

Steuerung der Last

Wie bei jedem verteilten System müssen Sie überlegen, wie Sie die Last in Ihrer MCP-Serverflotte kontrollieren können. Zunächst überlegen Sie, ob Sie eine Ratenbegrenzung auf Ihren MCP-Servern implementieren sollten und wo die Grenzwerte implementiert werden sollen. Wenn Sie sich dafür entscheiden, keine Ratenbegrenzung zu implementieren, geben Sie jegliche Ratenbegrenzung weiter, die von nachgelagerten Ressourcen vorgenommen wird. Viele Systeme entscheiden sich für eine Ratenbegrenzung auf der Grundlage von Anforderungsattributen, wie z. B. einer Benutzer- oder Konto-ID. Stellen Sie sicher, dass die Anfragen, die an nachgelagerte Dienste gesendet werden, dieselben Attribute aufweisen, sodass mehrere Benutzer nicht durch die Belastung durch einen anderen Benutzer beeinträchtigt werden.

Wenn Sie sich für die Implementierung einer Ratenbegrenzung entscheiden, empfiehlt es sich, die primäre Ratenbegrenzung auf MCP-Serverebene zu implementieren, wobei Back-End-Dienste sekundären Schutz bieten und Agenten ihr Verhalten auf der Grundlage von Ratenbegrenzungen

anpassen. Überlegen Sie, ob die Ratenbegrenzungen pro MCP-Server oder pro Tool gelten. Serverratenbeschränkungen pro MCP tragen zum Schutz Ihrer MCP-Serverflotte und -Services in einer Umgebung mit mehreren Mandanten bei. Das kann jedoch sehr grobkörnig sein. Die Ratenbegrenzungen pro Werkzeug sollen verhindern, dass nachgelagerte Ressourcen überlastet werden, die sich selbst möglicherweise nicht ausreichend einschränken. Wenn ein Tool mehrere Funktionen aufruft APIs, sollten Sie das Ratenlimit so einstellen, dass es sich an der niedrigsten Rate orientiert, die für diese APIs Tools zulässig ist.

Die Weitergabe von Informationen zur Ratenbegrenzung in HTTP-Headern kann auch eine nützliche Metrik für Benutzer und automatisierte Systeme sein, um ihre eigene Anforderungsrate und Wiederholungsstrategie zu verwalten. Sie können diese Header beispielsweise von Ihrem MCP-Server an den Agenten zurücksenden, wie im folgenden Beispiel gezeigt:

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 45
X-RateLimit-Reset: 1640995200
```

Darüber hinaus sollten Sie einen Lastabbau in Betracht ziehen, um den gesamten Service zu schützen, wenn kein einziger Kunde ein Ratenlimit überschreitet, die Last jedoch die Systemleistung beeinträchtigt.

Bewährte Methoden zur Steuerung der Auslastung

- Wählen Sie einen Ansatz zur Ratenbegrenzung — Planen Sie eine Ratenbegrenzung für einzelne Benutzer ein, entweder auf der Grundlage ihrer Nutzung nachgelagerter Ressourcen oder aufgrund ihrer Nutzung Ihres MCP-Servers und Ihrer Tools.
- Ziehen Sie Lastabbau in Betracht — Schützen Sie Ihre MCP-Serverflotte vor allgemeiner Überlastung, die nicht von einem einzelnen oder einer Handvoll Kunden verursacht wird.

Operationelle Metriken

Die wichtigsten Kennzahlen, die für MCP-Implementierungen erfasst werden müssen, sollten sich auf das Kundenerlebnis konzentrieren, das sie bieten. Zu diesen Kennzahlen gehören in der Regel die Token-Nutzung, die Genauigkeit der Toolauswahl, die Anzahl der beim Agenten registrierten Tools und die Latenz der Tools. Durch die Überwachung der von den einzelnen Tools zurückgegebenen Ausgabetokens können Sie beispielsweise Alarme einrichten, wenn Tools einen Schwellenwert für die Nutzung von Kontextfenstern überschreiten. Wenn ein Tool diesen

Schwellenwert überschreitet, sollten Sie das Verhalten des Tools überprüfen. Dies steht auch im Zusammenhang mit der Designstrategie des MCP-Tools. Kennzahlen zur Genauigkeit der Toolauswahl geben Aufschluss darüber, wie gut die Agenten die geeigneten Tools für bestimmte Aufgaben auswählen, während Ausführungsgeschwindigkeit und Erfolgsquoten Leistungsengpässe und Zuverlässigkeitsprobleme aufzeigen.

Um beispielsweise die Genauigkeitsmetriken für die Toolauswahl und den Tool-Einsatz zu bewerten, erstellten AWS Teams Gold-Datasets für Regressionstests. Die Datensätze wurden synthetisch generiert, indem historische API-Aufrufprotokolle bei Benutzeranfragen verwendet LLMs wurden. Anhand der vordefinierten Metriken zur Werkzeugauswahl und zum Werkzeugeinsatz (wie Genauigkeit der Werkzeugauswahl, Genauigkeit der Werkzeugparameter und Genauigkeit von Funktionsaufrufen mit mehreren Turns) konnten die AWS Teams objektiv beurteilen, ob der KI-Agent in der Lage ist, die geeigneten Tools korrekt zu identifizieren, ihre Parameter mit genauen Werten zu füllen und über Gesprächsrunden hinweg kohärente Werkzeugaufrufsequenzen aufrechtzuerhalten.

Durch die Messung von Kennzahlen zur Anzahl der bei einem Agenten registrierten Tools können Sie potenzielle Herausforderungen bei der Verwaltung von Kontextfenstern sowie Änderungen an den verfügbaren Tools von MCP-Servern identifizieren. Sie sollten regelmäßig Betriebskennzahlen überprüfen, die Aufschluss über die Benutzererfahrung mit Ihrem MCP-Server und den Tools geben.

Mitwirkende

Verfassen

- Alex Torres, leitender Lösungsarchitekt, AWS
- Saikat Gomes, Senior Manager für Kundenlösungen, AWS
- Mike Haken, leitender Lösungsarchitekt, AWS
- Sreeja Das, Chefsingenieur, AWS

Überprüfend

- Ted Swinyar, Leiter des Lösungsarchitekten, AWS
- Raju Patil, leitender Datenwissenschaftler, AWS

Technisches Schreiben

- Lilly AbouHarb, leitende technische Redakteurin, AWS

Dokumentverlauf

In der folgenden Tabelle werden wichtige Änderungen in diesem Leitfaden beschrieben. Um Benachrichtigungen über zukünftige Aktualisierungen zu erhalten, können Sie einen [RSS-Feed](#) abonnieren.

Änderung	Beschreibung	Datum
Erste Veröffentlichung	—	16. März 2026

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.