Estratégias do Protocolo de Contexto de Modelo em AWS

AWS Recomendações

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

AWS Recomendações: Estratégias do Protocolo de Contexto de Modelo em AWS

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

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

Introdução 1
Público-alvo 2
Objetivos 2
O que é MCP? 5
Entendendo as ferramentas 5
Quando usar o MCP 8
Estratégia de design de ferramentas MCP 12
Escopo da ferramenta 13
Granular 13
Coarse-grained 14
Melhores práticas para o escopo da ferramenta MCP 15
Definições de ferramentas 16
Abordagem de especificação de 17
Abordagem Docstring 18
Melhores práticas para definições de ferramentas MCP 18
Descoberta de ferramentas 19
Definição estática 19
Descoberta dinâmica 20
Função de pesquisa 20
Práticas recomendadas para descoberta de ferramentas MCP 21
Organização de ferramentas 21
Melhores práticas para organização de ferramentas de MPC 22
Estratégia de hospedagem MCP 23
Abordagens de hospedagem 23
Hospedagem local 23
Hospedagem remota 25
Gateway MCP 25
Práticas recomendadas para hospedar servidores MCP 26
Estratégia de governança do MCP 27
Autenticação e autorização 27
Melhores práticas para autenticação e autorização de MCP 29
Controlando a carga 29
Melhores práticas para controlar a carga 30
Métricas operacionais 30

Colaboradores 32
Autoria 32
Análise 32
Redação técnica 32
Histórico do documento 33
..... xxxiv

Estratégias do Protocolo de Contexto de Modelo em AWS

Amazon Web Services (colaboradores)

Março de 2026 (histórico do documento)

Este guia pode ajudá-lo a desenvolver e implementar estratégias do Model Context Protocol (MCP) em toda a sua organização para apoiar sua jornada de IA agente. À medida que os agentes e os modelos de linguagem se tornam cada vez mais centrais para as operações comerciais, estabelecer uma estratégia de MCP é fundamental para soluções de agentes bem-sucedidas.

Este guia explora três pilares fundamentais para criar uma estratégia de MCP: design de ferramentas MCP, hospedagem de servidores MCP e governança de MCP. Ao abordar esses componentes interconectados, as organizações podem criar sistemas escaláveis, seguros e eficazes para gerenciar o contexto do modelo em suas implementações de IA. Essa orientação fornece insights acionáveis e orientação estratégica para organizações em qualquer estágio da jornada de IA de uma organização, desde a experimentação inicial até as implantações de produção em grande escala. Isso os ajuda a desenvolver soluções MCP personalizadas que se alinham às suas necessidades e objetivos específicos.

Essas melhores práticas são derivadas de implementações reais de organizações que implantam MCP em escala corporativa, de uma análise dos padrões atuais de especificação de MCP e das lições aprendidas com aplicativos personalizados de modelo de linguagem grande (LLM) em produção.

Os sistemas de IA são cada vez mais sofisticados e robustos LLMs em uma ampla variedade de casos de uso. LLMs se destacam na compreensão da linguagem natural, na geração de respostas semelhantes às humanas e no raciocínio sobre informações complexas. No entanto, para transformar interfaces LLMs de conversação em sistemas que podem realizar tarefas complexas de forma autônoma, as organizações estão adotando arquiteturas de IA agênticas, sistemas de IA que podem perceber seu ambiente, raciocinar sobre metas, tomar decisões autônomas, orquestrar várias etapas e realizar ações para alcançar objetivos em nome dos usuários. Essa abordagem agente ajuda as organizações a criar sistemas de IA que podem entender a intenção do usuário por meio da linguagem natural, coordenar de forma autônoma várias fontes de dados e ferramentas e oferecer experiências personalizadas em uma escala que não era possível com os padrões tradicionais de solicitação-resposta. Para tornar esses agentes mais capazes, as organizações precisam fornecer acesso às ferramentas e dados existentes para enriquecer a compreensão contextual do agente e permitir que ele atue em nome do usuário.

O MCP fornece um protocolo padronizado para integração de ferramentas de IA, permitindo uma comunicação consistente entre agentes e recursos externos. Embora o próprio MCP defina o padrão de comunicação, implementá-lo de forma eficaz requer uma análise cuidadosa dos padrões arquitetônicos, modelos de segurança, práticas operacionais e estratégias de otimização de desempenho para obter soluções escaláveis, seguras e sustentáveis.

Este guia sintetiza as lições aprendidas nas implantações corporativas de MCP, fornecendo recomendações práticas que se alinham ao Well-Architected Framework.AWS Ele abrange estratégias para design de ferramentas MCP, hospedagem de servidores MCP e governança de MCP, que são essenciais para criar suas próprias soluções MCP. As recomendações neste guia mapeiam os seguintes cinco pilares do AWS Well-Architected Framework:

- Segurança — isolamento de token, credenciais com escopo reduzido, autorização separada read/write
- Excelência operacional — métricas de precisão de seleção de ferramentas, conjuntos de dados dourados para testes de regressão
- Confiabilidade — Limitação da taxa por usuário e por ferramenta, redução de carga
- Eficiência de desempenho — ferramentas com escopo de fluxo de trabalho, filtragem de ferramentas, pesquisa semântica para reduzir o uso da janela de contexto
- Otimização de custos — servidores MCP reutilizáveis em todas as equipes, custos de token reduzidos por solicitação por meio da filtragem de ferramentas

Público-alvo

Este guia é destinado a arquitetos, desenvolvedores e líderes de tecnologia que estão implementando soluções de IA agênticas em suas organizações. Para entender os conceitos deste guia, você deve entender como LLMs funciona e ter conhecimento básico sobre MCP, ferramentas e engenharia rápida.

Objetivos

Criar sistemas de IA da Agentic que estejam prontos para produção significa resolver juntos a governança, a otimização e a segurança para apoiar as políticas da sua organização. A seguir, explica como esse guia aborda esses objetivos:

- Governança — Sem governança centralizada, você não pode responder a perguntas de auditoria sobre suas cargas de trabalho de IA, incluindo quais agentes acessaram quais dados, com quais

permissões e quando. Você também não pode impor o controle de versão. A seção de estratégia de hospedagem MCP deste guia explica como os usuários podem estar executando servidores MCP locais desatualizados com vulnerabilidades conhecidas devido à falta de fiscalizações sistemáticas.

Para setores regulamentados, a governança é fundamental. Os auditores querem ver a aplicação de políticas e o rastreamento do uso de ferramentas em todos os agentes em um único painel. A governança do MCP fornece isso.

Seguindo as recomendações deste guia, você pode melhorar a precisão das tarefas em 28 a 32% em benchmarks revisados por pares. Para obter mais informações, consulte MARCO: Multi-Agent Real-Time Chat Orchestration (site da ACL Anthology). A governança não se trata apenas de conformidade; ela também melhora o desempenho do seu sistema de IA agente.

- Otimização — Suas equipes podem criar as mesmas integrações mais de uma vez. Por exemplo, quando cinco equipes diferentes escrevem seu próprio script de consulta de banco de dados para que seu aplicativo de IA se comunique com seus bancos de dados, isso representa cinco vezes o custo de desenvolvimento e cinco conjuntos de bugs a serem mantidos. O MCP permite que você o crie uma vez e o compartilhe com toda a comunidade de engenharia. A economia aumenta à medida que seu número de agentes aumenta.

  Também há um problema de custo por solicitação que a maioria das equipes não percebe no início. Cada definição de ferramenta consome tokens da janela de contexto. Com 20 ferramentas, você está gastando de 5.000 a 10.000 tokens por invocação apenas nas descrições, junto com as consultas do usuário. Isso aumenta a latência e os custos de inferência do LLM e diminui a precisão à medida que o modelo se esforça para escolher a ferramenta certa na lista de ferramentas disponíveis.

  Os agentes que usam pacotes de ferramentas estruturados são aproximadamente três vezes mais precisos nas tarefas do banco de dados do que os agentes que acessam APIs diretamente (para obter mais informações, consulte Middleware for LLMs: Tools Are Instrumental for Language Agents in Complex Environments). A forma como você projeta e apresenta ferramentas para um modelo de IA é importante. Este guia recomenda fornecer às ferramentas esquemas claros, definindo-as para fluxos de trabalho reais em vez de endpoints brutos e limitando as informações na janela de contexto. A seção de estratégia de design de ferramentas MCP deste guia se aprofunda nesses aspectos.
- Segurança e conformidade — imagine um sistema de IA agente que alucina uma etapa de limpeza e tenta excluir um banco de dados de produção. Se o agente herdou as credenciais de

administrador completas do usuário, a exclusão pode ser concluída. Com isolamento de token e credenciais com escopo reduzido que concedem apenas acesso de leitura e criação, ele falha com segurança.

Fluxos de trabalho regulamentados aprimoram ainda mais isso. O guia fornece exemplos (canais de assistência médica que exigem a validação da HIPAA e a anonimização das informações de identificação pessoal antes do processamento dos dados do paciente). A incorporação dessa lógica nas ferramentas do MCP significa que a conformidade sempre acontece de forma determinística.

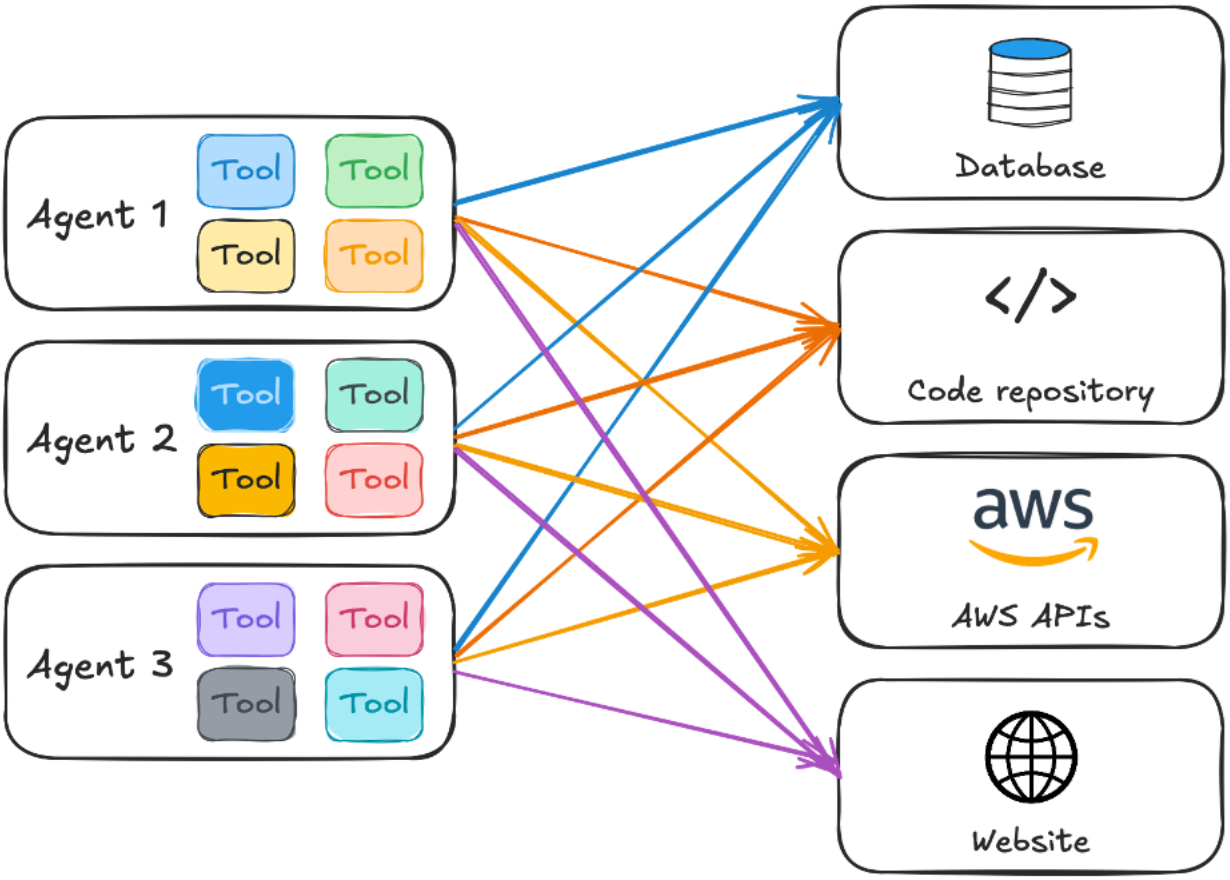
O que é MCP?

Os LLMs funcionam prevendo uma resposta a uma solicitação com base em seus dados de treinamento. Isso significa que o LLM só pode fornecer respostas sobre dados e eventos que já viu. Métodos como Retrieval Augmented Generation (RAG) e bases de conhecimento permitem que você inclua dados contextuais. No entanto, se você perguntasse a um LLM qual será a previsão do tempo para amanhã ou quantos clientes estão em seu banco de dados, ele provavelmente alucinaria ou não seria capaz de fornecer uma resposta porque isso está fora do conhecimento pré-treinado do LLM. Para poder responder a esses tipos de perguntas, um agente precisa acessar recursos externos, dados e APIs fora do contexto nativo do LLM.

Entendendo as ferramentas

Podemos dar ao LLM acesso a sistemas e contextos adicionais por meio de ferramentas.As ferramentas são funções atribuídas ao LLM para atingir um objetivo claro. Uma ferramenta pode chamar uma API, consultar um banco de dados, realizar operações de calculadora, operar uma sandbox de código, realizar uma pesquisa na web e até mesmo invocar outro sistema de IA ou agente como ferramenta. Cada ferramenta deve incluir uma descrição que diga ao LLM o que a ferramenta faz, quando usá-la e quais parâmetros ela aceita. Isso permite que o LLM tome decisões diferenciadas sobre qual ferramenta ou combinação de ferramentas invocar com base na entrada do usuário. O LLM é informado sobre quais ferramentas estão disponíveis para o agente, permitindo gerar respostas que instruem o agente a invocar a ferramenta. Por exemplo, quando você pergunta ao LLM quantos clientes estão em seu banco de dados, o LLM envia uma resposta de volta ao agente solicitando a execução da `query_database` ferramenta com parâmetros de entrada específicos. O LLM determina qual ferramenta invocar e as entradas para a chamada da ferramenta. O agente então executa a ferramenta, que converte a entrada da linguagem natural em uma chamada de função sintaticamente correta e executa a consulta. O agente invoca a ferramenta ou ferramentas com base nas instruções do LLM, e esses resultados são repassados ao LLM. Isso aproveita a capacidade do LLM de raciocinar sobre a entrada baseada em texto e selecionar as ferramentas apropriadas para o trabalho.

A imagem a seguir mostra como cada agente gerencia seu próprio conjunto de ferramentas para cada alvo.



O escalonamento do acesso às ferramentas pode apresentar desafios para soluções de IA agentes:

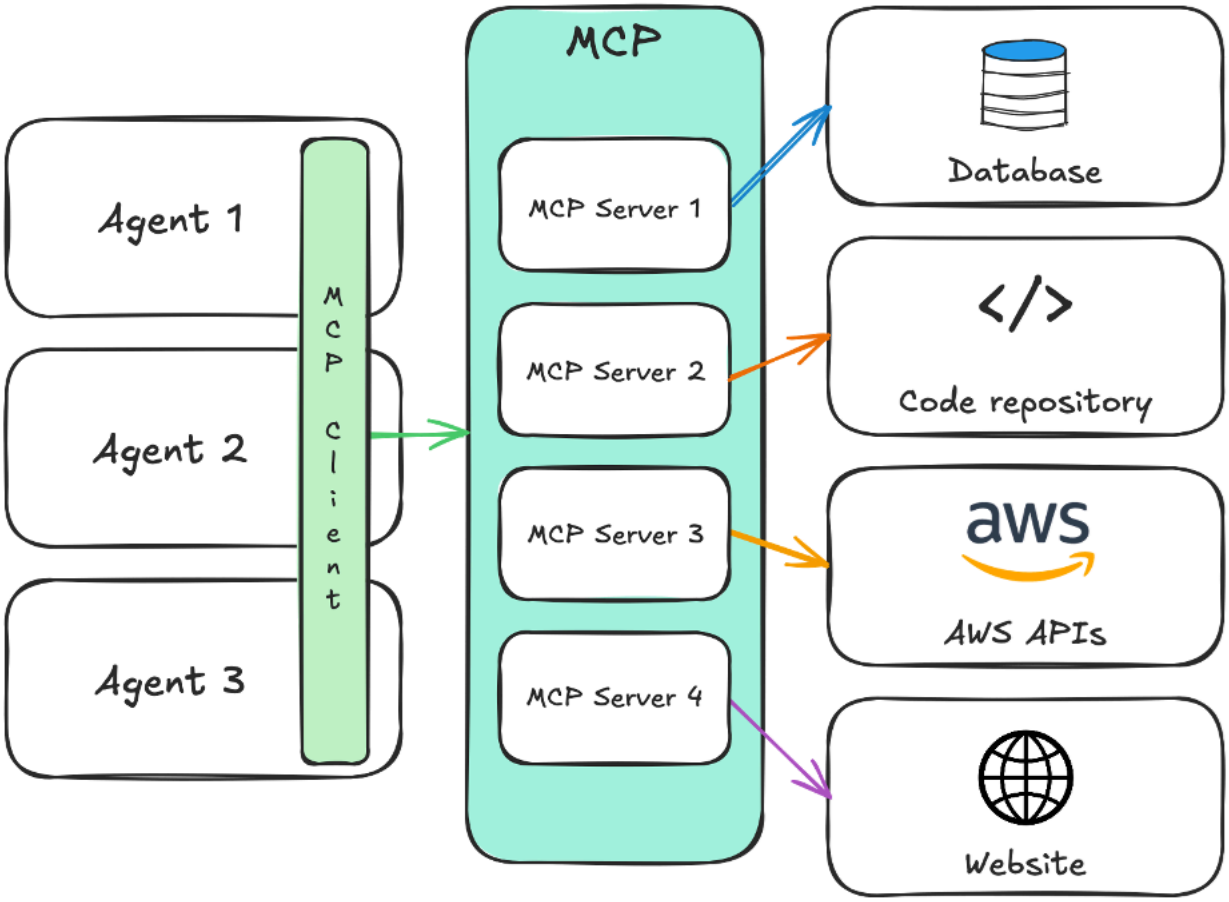
- Se cada desenvolvedor está criando sua própria ferramenta para os mesmos recursos externos, há muito esforço duplicado e formas não padronizadas de interagir com esses recursos externos. Isso produz implementações inconsistentes em seus agentes. Embora você possa resolver esse problema desenvolvendo ferramentas padrão em bibliotecas e distribuindo-as, isso carece de governança centralizada. Isso dificulta a aplicação de políticas de segurança, o rastreamento do uso de ferramentas, o gerenciamento de versões entre as equipes ou a garantia da conformidade com os padrões organizacionais. Além disso, ao incorporar ferramentas diretamente ao agente, você deve reimplantar seu agente sempre que uma nova ferramenta for criada ou uma existente for atualizada.
- Fornecer ferramentas para um LLM consome sua janela de contexto. A janela de contexto é o número de tokens (unidades de texto que os LLMs processam — normalmente representando palavras, partes de palavras ou pontuação) que um modelo pode considerar a qualquer momento. Os LLMs têm limites de janela de contexto. As ferramentas e sua documentação consomem essa janela de contexto finita junto com as solicitações do sistema e as solicitações do usuário. À medida que a janela de contexto é preenchida, os LLMs podem ter um desempenho degradado

devido a vários fatores: dificuldade em identificar informações relevantes, maior complexidade de processamento e capacidade reduzida de raciocínio. O desafio é agravado quando as definições de ferramentas, as solicitações do sistema e o histórico de conversas competem por um espaço limitado na janela de contexto porque são fornecidos em cada invocação do LLM.

Assim, o número de ferramentas e a forma como elas são documentadas têm impacto direto no desempenho do LLM, como tempo de resposta e precisão.

O MCP estabelece um padrão universal para conectar agentes a recursos externos. É comumente chamado de "USB-C para aplicativos de IA". Em vez de registrar ferramentas diretamente com os agentes, os servidores MCP atuam como intermediários para hospedar ferramentas que são descobertas e invocadas na versão 2.0. JSON-RPC Em vez de adicionar dezenas ou centenas de ferramentas diferentes ao seu agente e mantê-las ao longo do tempo, o MCP permite que você registre servidores MCP que encapsulam as ferramentas que seu agente pode acessar. Essa abordagem padroniza como as ferramentas são empacotadas, apresentadas e invocadas. Isso pode ajudar a enfrentar os desafios de escala e governança do uso de ferramentas em seus agentes. Ele também separa o desenvolvimento e as operações de agentes das ferramentas que usa para recursos externos.

A figura a seguir mostra os agentes usando o MCP para acessar recursos externos.



No entanto, o padrão MCP não resolve todos os desafios de escalabilidade e governança. A implementação de servidores MCP deve ser combinada com estratégias eficazes de design de ferramentas, hospedagem e governança corporativa. Este guia fornece as melhores práticas para cada estratégia para ajudá-lo a criar e usar o MCP como parte de suas soluções de IA para agentes.

Quando usar o MCP

O MCP fornece infraestrutura estratégica para escalar suas iniciativas de inteligência artificial. Ao centralizar o gerenciamento e a governança de ferramentas, os servidores MCP reduzem o custo cumulativo de criar e manter integrações personalizadas entre vários agentes. Isso proporciona retornos crescentes à medida que seu ecossistema de agentes se expande.

O MCP provavelmente se torna parte de sua estratégia quando:

- Você precisa de uma governança centralizada sobre como os agentes acessam sistemas e serviços corporativos, como bancos de dados, APIs, ferramentas internas e integrações de terceiros.

- Os desenvolvedores passam muito tempo escrevendo integrações personalizadas que não são consistentes em todas as implementações.
- Você tem ferramentas duplicadas que podem servir a recursos comuns.
- Você deseja oferecer suas ferramentas ou dados proprietários a consumidores externos ou sistemas de agências terceirizadas por meio de interfaces MCP padronizadas e governadas, liberando novos fluxos de receita e mantendo a segurança e o controle.

Depois de decidir que os servidores MCP farão parte de sua estratégia, avalie se as implementações de servidores MCP de código aberto existentes atendem às suas necessidades, se elas precisam ser aprimoradas ou se você precisa criar servidores personalizados. Muitas implementações de servidor MCP pré-criadas estão disponíveis em repositórios públicos e abrangem recursos comuns, como acesso ao sistema de arquivos, navegação na web, sandboxes de código, acesso ao banco de dados e integrações de API.

Em muitos casos, servidores MCP preexistentes são suficientes. Por exemplo, AWS fornece o Servidor AWS MCP, um servidor MCP remoto gerenciado que fornece aos assistentes e agentes de IA acesso seguro e autenticado Serviços da AWS por meio de interações em linguagem natural. Você pode usar o Servidor AWS MCP para realizar AWS tarefas complexas de várias etapas combinando acesso em tempo real à AWS documentação, chamadas de API sintaticamente corretas e fluxos de trabalho pré-criados chamados SOPs de agentes que seguem as melhores práticas. AWS AWS testa continuamente os Servidor AWS MCP para garantir que os agentes do cliente possam usá-los com sucesso.

Você deve testar esses servidores MCP existentes com seus agentes para determinar se eles atendem aos seus casos de uso. Se um agente não conseguir concluir fluxos de trabalho, gerar respostas incorretas ou inadequadas, não conseguir navegar por processos complexos de várias etapas ou perder importantes práticas recomendadas ou considerações de segurança específicas do domínio, você deve considerar aprimoramentos em várias dimensões.

Quando os servidores MCP existentes não atendem totalmente às suas necessidades e têm dificuldade em usar as ferramentas existentes corretamente ou produzir respostas precisas, considere estas abordagens de aprimoramento antes de criar servidores personalizados:

- Enriqueça o contexto do agente — Se seu agente tiver dificuldade em usar as ferramentas de forma correta ou eficiente em um servidor MCP existente, considere complementar essas definições de ferramentas com documentação ou exemplos adicionais. Isso ajuda a fornecer contexto adicional ao LLM.

- Adicione ferramentas complementares — amplie os servidores MCP existentes com ferramentas que acessam dados ou contextos organizacionais adicionais de que os agentes precisam para concluir fluxos de trabalho com êxito.
- Melhore as APIs subjacentes — Simplifique suas APIs de serviço para torná-las maiores, LLM-friendly reduzindo a complexidade dos parâmetros, fornecendo mensagens de erro mais claras e oferecendo padrões sensatos que os agentes possam usar.

Embora o uso de implementações de servidores MCP existentes acelere o desenvolvimento de recursos comuns, a criação de servidores MCP personalizados é uma necessidade quando seu caso de uso exige funcionalidade especializada. Os servidores MCP personalizados ajudam você a encapsular a experiência do domínio, aplicar padrões organizacionais, melhorar a confiabilidade dos agentes para fluxos de trabalho complexos e oferecer suporte à conformidade com os requisitos de segurança. Considere criar um servidor MCP personalizado nas seguintes situações:

- Domain-specific fluxos de trabalho — Multi-step fluxos de trabalho que exigem experiência no domínio devem ser encapsulados em ferramentas MCP personalizadas quando o conhecimento necessário não é capturado na documentação da API. Por exemplo, em vez de permitir que os agentes organizem canais complexos de dados de saúde que devem validar a conformidade com a Lei de Portabilidade e Responsabilidade de Seguros de Saúde (HIPAA), anonimizar as PII e transformar para o formato HL7 FHIR, forneça uma ferramenta que incorpore diretamente a experiência do domínio. `process_patient_data` Isso elimina a dependência do LLM para orquestrar e executar corretamente as etapas do fluxo de trabalho, o que melhora a consistência e a conformidade.
- Abstrações do caminho dourado — Os agentes podem ter dificuldade em implementar abordagens ideais porque não têm contexto organizacional e adotam padrões básicos em vez das melhores práticas organizacionais. Nesses cenários, você pode aplicar padrões prescritivos de custo, desempenho ou segurança encapsulando esses caminhos dourados em ferramentas MCP personalizadas. Por exemplo, em vez de permitir que os agentes implantem uma infraestrutura com configurações padrão que podem ser inseguras ou ineficientes, forneça uma `deploy_secure_infrastructure` ferramenta que incorpore diretamente os padrões da sua organização.
- Orquestração complexa de vários serviços — Em vez de fazer com que o agente orquestre fluxos de trabalho complexos tentando inferir a sequência correta e o conjunto de serviços a serem usados em cada etapa, você pode criar de forma determinística essa lógica dentro de uma ferramenta MCP. Talvez você também queira fornecer experiência sobre os padrões ideais de

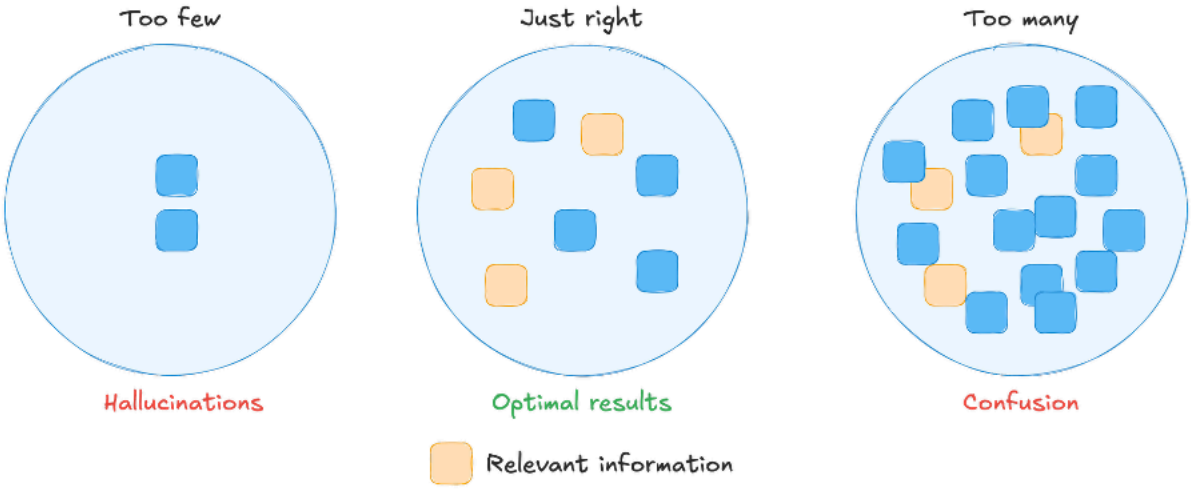
integração de serviços que o agente talvez não conheça. Isso também pode melhorar a precisão e a eficiência de seus agentes.

- Service-specific melhores práticas — Isso é comum para ferramentas focadas em segurança que ajudam os agentes a implementar políticas de criptografia, controles de acesso e padrões de conformidade específicos para o serviço que está sendo acessado por meio da ferramenta do agente. Além disso, se houver melhores práticas operacionais específicas do serviço que não sejam óbvias, o uso de um servidor MCP pode ajudá-lo a garantir que elas sejam implementadas e não deixadas para um agente raciocinar.

Estratégia de design de ferramentas MCP

A principal tarefa do cliente e servidor MCP é descobrir e apresentar ferramentas ao LLM para que ele possa usá-las para melhorar suas respostas. Isso faz do design da ferramenta MCP uma das estratégias mais importantes para criar soluções MCP eficazes. Do ponto de vista do modelo, as ferramentas são uma função que eles podem invocar conforme necessário para fornecer respostas mais precisas e completas. A interface da função abstrai a implementação subjacente de uma ferramenta, que pode variar de um invólucro em torno de uma única chamada de API até uma lógica complexa de fluxo de trabalho.

No entanto, você deve encontrar um equilíbrio com a quantidade de ferramentas fornecidas ao LLM. Se houver poucas ferramentas, o LLM pode não conseguir coletar o contexto e as informações corretos, portanto, fará a melhor suposição com as informações disponíveis no modelo. Se houver muitas ferramentas, o LLM pode ficar confuso sobre a seleção e a sequência corretas das ferramentas, levando a alucinações. Seu objetivo é obter o número certo de ferramentas. A imagem a seguir mostra os desafios de poucas e muitas ferramentas.



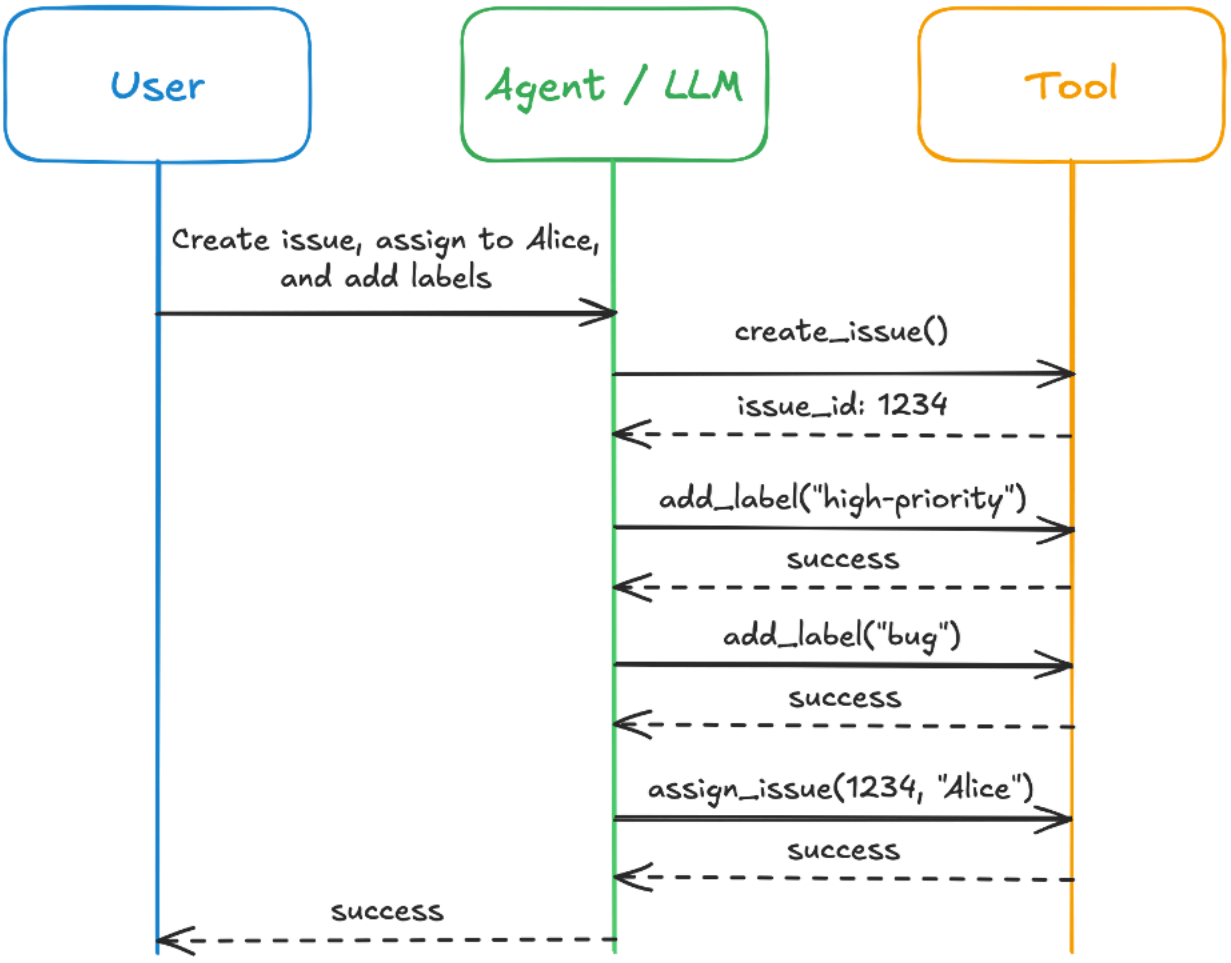
A solução requer a compreensão de quantas ferramentas fornecer e como definir o escopo de cada ferramenta. A granularidade de suas ferramentas, sejam elas mapeadas para chamadas de API individuais ou fluxos de trabalho completos, afeta diretamente o número total de ferramentas que os agentes precisam e a eficiência com que podem usá-las. Esta seção fornece as melhores práticas para definir o escopo das ferramentas MCP, criar definições de ferramentas, descobrir ferramentas e organizá-las.

Escopo da ferramenta

Há duas abordagens para o desenvolvimento de ferramentas: granular e granulada.

Granular

Em uma abordagem granular, você criaria uma ferramenta por API, ação ou consulta. Por exemplo, você pode criar`create_issue`,`get_issue`, `add_labelassign_issue`, e `close_issue` ferramentas para seu repositório Git. Isso permitiria que o LLM fizesse chamadas granulares para cada API e orquestrasse cada uma conforme necessário. Considere o seguinte aviso: “Crie um problema para o serviço do produto chamado 'A consulta retorna apenas resultados parciais', rotule-o como um bug e de alta prioridade e atribua-o a Alice”. A imagem a seguir mostra como uma abordagem de ferramenta por API responderia a essa solicitação.

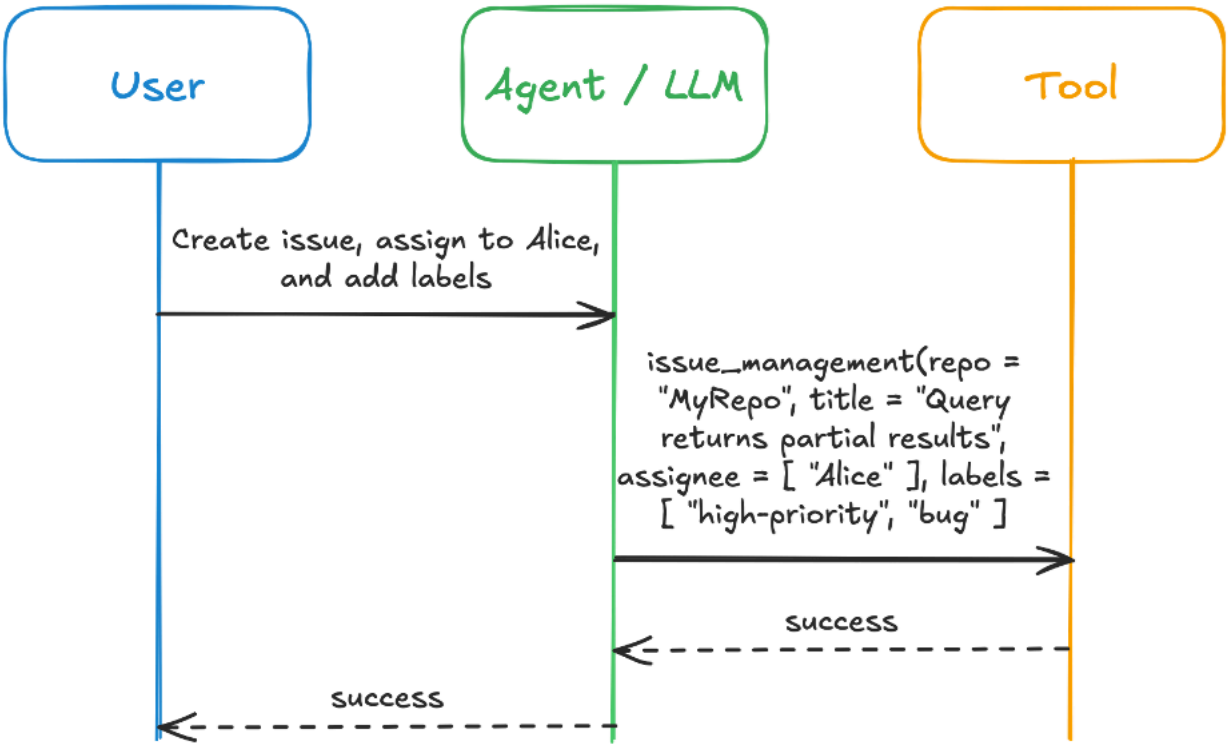


Nessa abordagem, o prompt do sistema e cada definição de ferramenta registrada são fornecidos ao LLM em cada chamada. Isso consome contexto adicional e incorre em uma penalidade de latência

porque cada chamada de ferramenta representa uma chamada individual para o LLM. Também aumenta a complexidade de lidar com erros no fluxo de trabalho.

Coarse-grained

Uma abordagem granulada ou orientada por fluxo de trabalho seriam ferramentas orientadas ao fluxo de trabalho. A ferramenta se concentra na intenção do usuário de ponta a ponta sobre a estrutura da API. Em vez de uma ferramenta por API, você tem uma ferramenta que chama deterministicamente muitas APIs. Usando o exemplo anterior do repositório Git, você pode criar uma `create_and_setup_issue` ferramenta que é chamada uma vez pelo agente. A implementação da ferramenta cria o problema, adiciona rótulos e o atribui a um usuário, com base nos parâmetros fornecidos à ferramenta. A imagem a seguir mostra como uma abordagem granulada processaria o mesmo prompt.



Essa abordagem mostra como toda a complexidade permanece oculta da camada LLM. Quando a lógica de orquestração é incorporada à implementação da ferramenta, todas as etapas sequenciais, o registro, a lógica de repetição, os disjuntores e a limitação de taxa são executadas de forma determinística na ferramenta. A abordagem orientada pelo fluxo de trabalho torna mais simples para o LLM invocar a ferramenta correta com os parâmetros certos. É importante observar que algumas APIs podem já fornecer a intenção do fluxo de trabalho, como a API do Amazon `RunInstances`

EC2. Nesses casos, uma ferramenta por API pode fornecer o design orientado ao fluxo de trabalho que você deseja.

No entanto, as ferramentas também podem ficar muito grossas. Se sua única ferramenta de fluxo de trabalho tentar fazer muitas coisas e tiver muitos parâmetros possíveis, o LLM poderá ter dificuldade em raciocinar sobre como usar a ferramenta corretamente. Também pode criar desafios com a seleção de parâmetros e o tratamento de erros. Portanto, o desenvolvimento de ferramentas deve encontrar um equilíbrio que se alinhe à intenção do usuário e evite pouca ou muita funcionalidade em uma única ferramenta. Recomendamos que você crie ferramentas com base em fluxos de trabalho completos do usuário, agrupando operações que normalmente ocorrem juntas (como três ou mais chamadas de API). Também recomendamos que você decomponha ferramentas que excedam oito ou mais parâmetros ou lidem com várias intenções de usuário distintas. Teste com instruções reais para verificar se os agentes podem usar cada ferramenta corretamente.

Se você tem fluxos de trabalho complexos e dinâmicos que não podem ser facilmente encapsulados como uma ferramenta determinística, considere usar o padrão de agente como ferramenta. Em vez de seu agente principal tentar orquestrar tarefas complexas em um fluxo de trabalho, um agente especializado pode atuar como uma ferramenta. Esses tipos de ferramentas podem implementar tomadas de decisão e ramificações avançadas, além de lidar com erros e repetir a lógica que não pode ser facilmente gerenciada em código determinístico. Isso é semelhante, mas distinto do protocolo Agent2Agent (A2A). O protocolo A2A é complementar, fornecendo interoperabilidade e colaboração entre agentes em qualquer estrutura de agente.

Recomendamos que você comece com a análise do fluxo de trabalho mapeando os fluxos de trabalho mais comuns dos usuários para identificar os principais recursos de que cada agente precisa. Isso estabelece seu conjunto mínimo de ferramentas viável. Com base em nossa experiência no desenvolvimento de servidores MCP em grande escala, recomendamos as seguintes práticas. Quando essas práticas entrarem em conflito, priorize a intenção do usuário e o fluxo de trabalho.

Melhores práticas para o escopo da ferramenta MCP

- Pense em histórias de usuários e agrupe operações comuns — as ferramentas devem ser mapeadas diretamente para concluir as interações do usuário, em vez de exigir a orquestração de várias operações. Se os fluxos de trabalho normalmente exigirem três ou mais chamadas separadas, combine-as em uma única ferramenta. Isso reduz a carga cognitiva no LLM, minimiza o número de chamadas de ferramentas, reduz o consumo de contexto e a latência necessários para concluir tarefas e melhora a precisão e a latência.

- Limite os parâmetros a oito ou menos — Se uma ferramenta exceder oito parâmetros, decomponha-a em várias ferramentas. Os LLMs têm dificuldades com a seleção de parâmetros à medida que a complexidade aumenta.

> **Note**
>
> Se as operações de agrupamento exigirem mais de oito parâmetros, priorize o agrupamento em vez da contagem de parâmetros, pois simplificar o fluxo de trabalho é mais valioso do que limites estritos de parâmetros.

- Operações separadas de leitura e gravação — forneça ferramentas diferentes para ler dados e modificá-los. Essa separação torna explícito quando os agentes estão realizando operações potencialmente destrutivas, permite políticas de autorização diferentes e reduz o risco de modificações não intencionais durante a coleta de informações.
- Forneça padrões razoáveis — Crie ferramentas para que o LLM precise especificar somente os parâmetros específicos da solicitação individual. Os padrões reduzem a complexidade dos parâmetros e melhoram a precisão da seleção de ferramentas, minimizando as informações sobre as quais o LLM deve raciocinar.
- Prefira a execução determinística — Torne a execução da ferramenta e a saída determinísticas quando possível. As ferramentas determinísticas são mais confiáveis e fáceis de testar. Para fluxos de trabalho complexos que exigem orquestração inteligente, lógica de ramificação ou tratamento avançado de erros que não podem ser facilmente gerenciados em código determinístico, considere usar agentes especializados como ferramentas. No entanto, use esse padrão seletivamente porque ele adiciona complexidade.

Definições de ferramentas

Quando um LLM recebe uma solicitação que não pode tratar diretamente, ele revisa as ferramentas disponíveis para ajudá-lo a concluir a solicitação. O LLM seleciona ferramentas com base em sua compreensão semântica dos nomes e descrições das ferramentas fornecidas e em todas as instruções fornecidas no prompt. Em seguida, ele criará uma entrada com base no esquema de entrada definido e esperará uma saída com base no esquema de saída. Portanto, criar definições descritivas de ferramentas e esquemas de entrada e saída validados é fundamental para ajudar o LLM a selecionar ferramentas de forma eficaz. Geralmente, há duas abordagens para criar essa documentação: a abordagem de especificação da ferramenta e a abordagem docstring.

Abordagem de especificação de

A abordagem recomendada é seguir diretamente a especificação da ferramenta MCP ao definir a ferramenta. O exemplo a seguir é mostrado usando o decorador de ferramentas Strands Agent:

```
@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:
  …
```

O uso de campos padrão, comoname,`description`,`inputSchema`, e `outputSchema` garante que cada ferramenta tenha uma documentação consistente que tanto o LLM quanto os humanos possam

entender. Cada ferramenta deve definir esses campos no mínimo e, opcionalmente, fornecer um título e anotações, que são dicas opcionais sobre o comportamento da ferramenta. Quando possível, use enums para valores de parâmetros para facilitar a seleção das opções corretas pelo LLM. As enumerações funcionam melhor para conjuntos finitos, como valores de status ou prioridade, mas não são adequadas para texto de formato livre, valores dinâmicos, números arbitrários ou identificadores de recursos. Nesses casos, forneça descrições e exemplos claros. Inclua também um valor padrão quando possível para que o LLM não precise adivinhar qual é a opção correta. Lembre-se de que as definições da ferramenta estão incluídas no prompt do LLM em cada invocação, consumindo espaço na janela de contexto junto com as instruções do sistema e o histórico de conversas.

Abordagem Docstring

Outra abordagem, se você estiver escrevendo suas ferramentas em Python, é usar docstrings para fornecer a descrição, o uso e a saída da ferramenta. Veja a seguir um exemplo dessa abordagem:

```
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 não impõem um esquema ou formato padronizado. O uso dessa abordagem pode gerar resultados inconsistentes com base em como os desenvolvedores de ferramentas escolhem documentar cada ferramenta. Definir e aplicar um padrão para toda a organização é essencial se você seguir essa abordagem.

Melhores práticas para definições de ferramentas MCP

- Siga as especificações da ferramenta MCP — Forneça `name`,`description`,`inputSchema`, e `outputSchema` campos para cada ferramenta. Para implementações de Python, use modelos Pydantic para fornecer documentação embutida por meio de descrições de campo, validação automática de tipos e valores restritos por meio de enumerações. Isso torna os esquemas autodocumentados e melhora a compreensão do LLM sobre as opções de parâmetros válidos.

- Escreva descrições como instruções — As descrições das ferramentas são instruções que orientam a tomada de decisão do LLM. Inclua os componentes essenciais da finalidade da ferramenta (o que a ferramenta faz), quando usá-la (cenários ou padrões de intenção do usuário), o contexto da saída (para que a saída é usada), os parâmetros e as condições de erro.
- Forneça exemplos concretos — Incluir exemplos de fluxo de trabalho com valores reais é a maneira mais eficaz de orientar os LLMs sobre o uso correto da ferramenta.
- Documente as dependências de forma explícita — inclua pré-requisitos, sequências numeradas, mudanças de estado e ações de acompanhamento.

Descoberta de ferramentas

Há três abordagens para descobrir e registrar ferramentas em seu agente com servidores MCP: definição estática, descoberta dinâmica e função de pesquisa.

Definição estática

Primeiro, você pode definir estaticamente as ferramentas disponíveis diretamente no código do agente. Nessa abordagem, você define uma ferramenta remota (um objeto de referência do lado do cliente em uma estrutura como o Strands Agent SDK) para cada ferramenta fornecida pelo servidor MCP que é acessada por um cliente MCP. O exemplo a seguir usa transporte HTTP simplificável:

```
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])
```

O registro individual de ferramentas ajuda você a ser muito seletivo quanto às ferramentas que você disponibiliza para o LLM, o que minimiza a quantidade de janela de contexto usada. A desvantagem

é que isso requer o conhecimento dos nomes das ferramentas disponíveis e pode ser frágil se as ferramentas disponíveis mudarem no servidor MCP.

Descoberta dinâmica

A próxima abordagem é usar a descoberta dinâmica e registrar todas as ferramentas disponíveis com o agente. Essa abordagem consome o contexto linearmente à medida que mais ferramentas são adicionadas ao servidor MCP. Veja a seguir um exemplo dessa abordagem:

```
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)
```

Considere um cenário em que uma definição de ferramenta típica consome aproximadamente 250 a 500 tokens (incluindo nome, descrição e esquema). O registro de 20 ferramentas consumiria de 5.000 a 10.000 tokens da sua janela de contexto. Quando você tem um pequeno número de servidores MCP e tem controle sobre o número de ferramentas, essa opção é a mais simples de implementar. No entanto, se se espera que a lista de ferramentas cresça, isso pode criar problemas silenciosos de gerenciamento de contexto em seus agentes. Uma variação alternativa dessa abordagem é usar um parâmetro de filtro de ferramentas ao chamar`list_tools`, como o que o SDK do Strands Agents fornece, para reduzir o número de ferramentas registradas com o agente.

Função de pesquisa

A terceira opção é usar uma função de pesquisa para encontrar ferramentas relevantes durante o tempo de execução. Você lista todas as ferramentas disponíveis do seu servidor MCP e, em seguida, executa uma pesquisa semântica sobre essas ferramentas com base no prompt do usuário. Em seguida, as ferramentas resultantes são registradas com seu agente. O Amazon Bedrock AgentCore Gateway fornece um recurso de pesquisa semântica nativa que pode facilitar a implementação desse tipo de solução.

Práticas recomendadas para descoberta de ferramentas MCP

- Preservação da janela de contexto — Escolha uma abordagem de descoberta e registro de ferramentas que conserve o máximo possível da janela de contexto.
- Use recursos de filtragem de ferramentas ou pesquisa semântica — Forneça dinamicamente ao LLM um conjunto de ferramentas com escopo reduzido para escolher, o que melhora sua precisão e eficácia na escolha da ferramenta certa. A filtragem de ferramentas pode operar em nomes de ferramentas (correspondência ou padrões exatos), descrições de ferramentas (correspondência semântica) ou tags de domínio ou categoria. A pesquisa semântica é particularmente eficaz para comparar a intenção do usuário com as descrições das ferramentas. Ambas as abordagens reduzem o uso da janela de contexto.

Organização de ferramentas

Descobrir as ferramentas certas e garantir que o LLM possa usá-las de forma eficaz é uma das partes mais críticas do desenvolvimento eficaz de ferramentas. Ao começar a desenvolver servidores MCP, você precisa de uma estratégia que determine:

- Quantas ferramentas entram em um servidor MCP
- Quais ferramentas não devem ser colocadas no mesmo servidor MCP
- Como nomear ferramentas para torná-las pesquisáveis e evitar colisões de nomes (ferramentas diferentes com o mesmo nome)
- Como documentar as ferramentas e o servidor MCP para torná-los fáceis de usar pelo LLM

A organização do namespace é um padrão de design que evita colisões de nomes de ferramentas, agrupa funcionalidades relacionadas e facilita a identificação eficiente de ferramentas pelos LLMs. O padrão estabelece uma categorização estruturada que é análoga aos sistemas de armazenamento organizados, em vez da acumulação não estruturada. Recomendamos o padrão domínio-substantivo-verbo para nomeação de ferramentas. Por exemplo, `github_issue_creategithub_issue_list`,`github_issue_update`,`github_pullrequest_creat` ou`github_pullrequest_merge`. A vantagem desse padrão é evidente ao examinar o comportamento de classificação alfabética. Quando as ferramentas são listadas em ordem alfabética, todas as operações relacionadas ao problema se agrupam (`create`,,`update`)`list`, seguidas pelas operações de pull request (,,). `create` `list` `merge` O substantivo (tipo de recurso) funciona como um limite organizacional. Essa estrutura facilita tanto a digitalização da ferramenta

LLM quanto a navegação na documentação humana porque as funcionalidades relacionadas se agrupam naturalmente.

O servidor MCP deve ser limitado no nível do domínio, mas pode ser subdividido com base na separação de tarefas dos recursos que ele fornece. Por exemplo, você pode ter servidores MCP separados para operações de gravação e leitura em um banco de dados. Para impor essa separação, é recomendável implementar grades de proteção no nível do agente que restrinjam quais servidores MCP podem ser acessados com base na intenção e nas permissões do usuário. Isso pode ser obtido por meio de uma combinação das seguintes opções:

- Carregamento condicional do servidor — Carregue o servidor MCP somente para leitura quando o agente detectar operações de leitura na entrada do usuário.
- Permission-based filtragem — Use a autorização do usuário para conceder acesso somente aos servidores MCP apropriados.

Finalmente, você desejará criar um limite superior no número de ferramentas fornecidas por um servidor MCP. Não faça suposições sobre como os agentes usarão seu servidor MCP. Eles podem listar ingenuamente todas as ferramentas disponíveis e fornecê-las todas ao LLM. Se você tiver mais de 50 ferramentas em um único servidor, considere dividi-las em vários servidores.

Melhores práticas para organização de ferramentas de MPC

- Use o padrão de nomenclatura domínio-substantivo-verbo para ferramentas — Implemente estratégias para evitar colisões de nomes em servidores MCP e em agentes.
- Defina um limite superior — Restrinja o número de ferramentas em um único servidor MCP.
- Divida os servidores MCP — Use a separação de tarefas para dividir os servidores MCP em grupos lógicos.

Estratégia de hospedagem MCP

A abstração das ferramentas disponíveis em servidores MCP separa o desenvolvimento de seu agente das ferramentas disponíveis. Isso apresenta os desafios de onde você hospeda seu servidor MCP e como as ferramentas são organizadas dentro desses servidores.

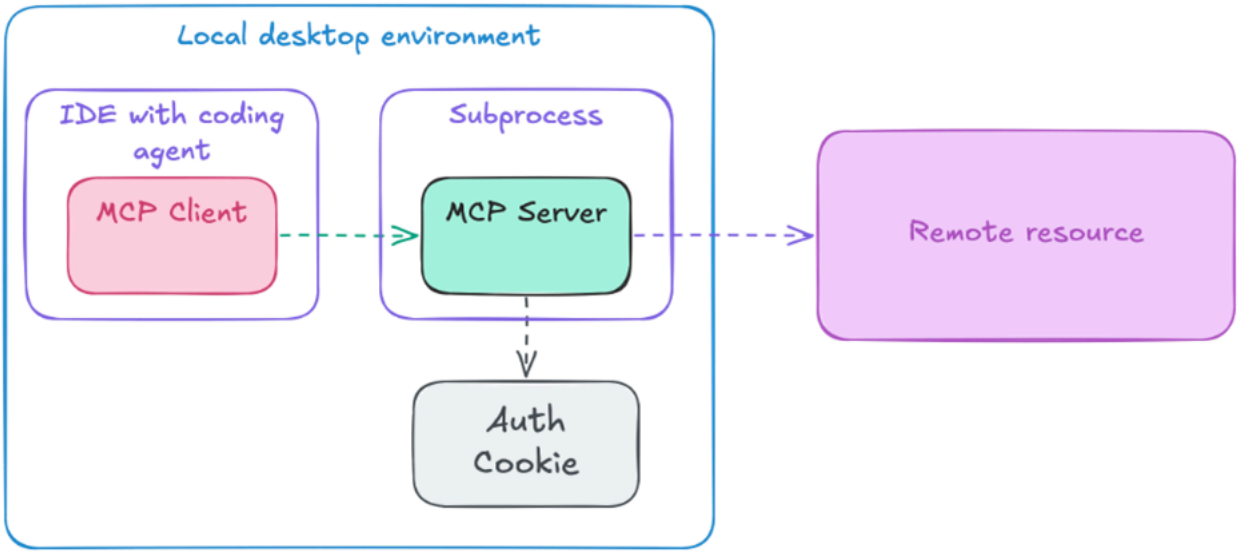
Abordagens de hospedagem

Há três opções para hospedar seus servidores MCP: executá-los localmente em uma máquina de usuário final, hospedá-los remotamente ou hospedá-los por meio de um gateway MCP. Cada opção tem vantagens e desvantagens.

Hospedagem local

A hospedagem local executa o servidor MCP como um subprocesso em sua máquina local junto com o agente que se comunica com o servidor usando fluxos JSON-RPC de entrada e saída padrão. Essa abordagem não exige autenticação entre o cliente e o servidor. As ferramentas podem interagir com aplicativos e arquivos locais, usar credenciais armazenadas localmente e herdar o acesso à rede da máquina local do usuário. Esse é o padrão de hospedagem mais simples e tem vários benefícios.

Muitos clientes começam a usar o MCP usando servidores locais. Eles permitem que os engenheiros iterem e resolvam rapidamente uma variedade de problemas em seu ambiente local. Considere um servidor MCP que se conecta a um repositório Git que o assistente de codificação de um engenheiro está usando. Manter o servidor MCP local faz muito sentido porque ele pode usar as credenciais exclusivas do engenheiro para acessar o repositório e não adiciona uma chamada de rede extra a um servidor MCP remoto. A imagem a seguir mostra um servidor MCP hospedado localmente sendo usado com um agente de codificação em um IDE.



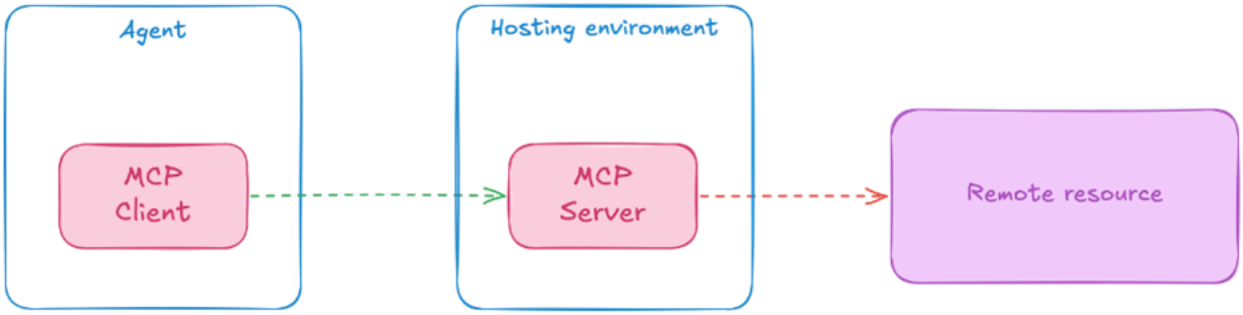
Para esses tipos de implantações, você deve considerar como os servidores MCP são desenvolvidos e distribuídos. A maioria dos clientes desenvolve um registro MCP em que os servidores podem ser registrados e baixados pelos usuários finais. É muito semelhante a um registro de contêiner em que um usuário pode pesquisar recursos específicos e encontrar os servidores MCP adequados às suas necessidades.

Existem registros públicos de MCP, como o Registro Oficial de MCP, e há registros hospedados de forma privada. Normalmente, as organizações alinham sua estratégia de registro de MCP às políticas existentes sobre distribuição de software de código aberto, registros de contêineres e gerenciamento interno de pacotes. Você deve considerar fatores como verificação de segurança, fluxos de trabalho de aprovação e requisitos de conformidade.

No entanto, a hospedagem local apresenta desafios operacionais que as organizações devem considerar. Primeiro, os usuários finais devem descobrir, baixar e configurar os servidores MCP de forma independente. Isso pode aumentar a complexidade para começar com cada servidor MCP individual que eles usam localmente. Segundo, você não pode controlar o ciclo de vida do servidor MCP, o que significa que os usuários podem continuar executando versões desatualizadas localmente com vulnerabilidades de segurança ou recursos ausentes. Isso pode complicar o cumprimento dos requisitos de conformidade. Alguns IDEs e ferramentas de CLI, como o Kiro, permitem que as organizações gerenciem e controlem quais ferramentas MCP estão disponíveis, garantindo consistência e segurança entre as equipes.

Hospedagem remota

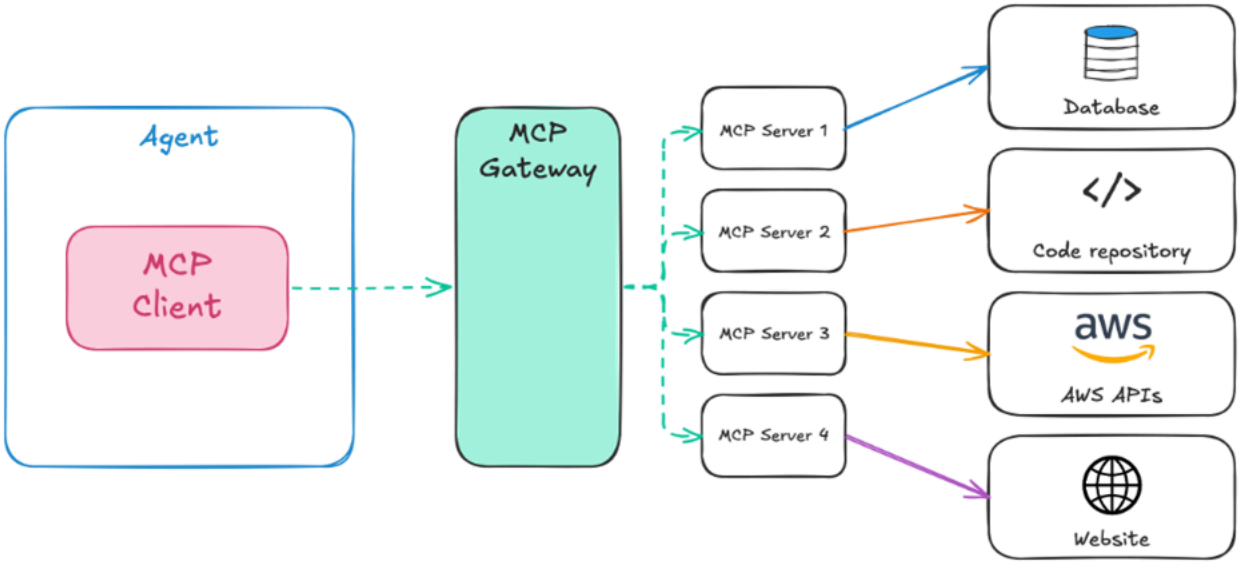
A segunda opção é hospedar servidores MCP remotos que são acessados por HTTP ou HTTPS. Isso fornece acesso a qualquer cliente conectado à rede. O uso da hospedagem remota permite controlar centralmente o acesso aos recursos e capacidades do MCP, implementar autenticação e autorização e controlar o controle de versão e as atualizações da lógica do servidor MCP. A hospedagem remota ainda exige o uso de um registro MCP para que os usuários finais possam descobrir os servidores MCP que desejam usar com seu agente. A imagem a seguir mostra a abordagem de hospedagem remota.



Do ponto de vista do desenvolvimento do agente, a experiência é semelhante, independentemente de o servidor MCP ser local ou remoto. A mudança mais significativa é implementar a autenticação e a autorização, incluindo o acesso do agente ao servidor MCP e o acesso do servidor aos recursos externos. As implementações remotas do servidor MCP devem ser cuidadosamente planejadas para considerar o acesso multilocatário e o gerenciamento de privilégios. O capítulo sobre estratégia de governança do MCP contém mais informações sobre considerações de autenticação e autorização.

Gateway MCP

A opção final é usar um gateway MCP. Os gateways MCP atuam como um proxy centralizado entre clientes e servidores MCP e orquestram o acesso aos servidores MCP registrados. Sem um gateway, cada agente precisa registrar cada servidor MCP remoto que queira usar. Um gateway permite que o agente se conecte a um único endpoint que gerencia a autenticação, a autorização, o roteamento e a tradução do protocolo. Novos servidores e ferramentas MCP podem ser adicionados dinamicamente e disponibilizados imediatamente para o agente. A imagem a seguir mostra a abordagem do gateway MCP.



Algumas soluções de gateway, como o Docker MCP Gateway, também gerenciam o ciclo de vida dos servidores MCP, lançando servidores sob demanda conforme necessário. Os gateways MCP, como o Amazon Bedrock AgentCore Gateway, também podem ajudar a gerenciar a descoberta de ferramentas fornecendo recursos de pesquisa semântica nativa. Isso fornece aos agentes um único endpoint para se conectar a um cliente MCP e ajuda a otimizar o uso da janela de contexto. O resultado são agentes simples que podem escolher e usar as ferramentas MCP de forma eficaz. No entanto, ele tem desafios relacionados à identidade semelhantes aos da abordagem de servidor MCP remoto.

Práticas recomendadas para hospedar servidores MCP

- O espectro de opções de hospedagem não é único para todos. Grande parte do uso de servidores MCP atualmente é local.
- Ao começar a usar servidores MCP remotos, sua principal consideração é a autenticação e a autorização consistentes para o servidor MCP e como o servidor MCP executa a autenticação e a autorização dos recursos downstream.
- Os gateways MCP simplificam a conectividade, a autenticação e a autorização para hospedar vários servidores MCP remotos. Eles também fornecem recursos para melhorar o gerenciamento de janelas de contexto pesquisando as ferramentas aplicáveis.

Estratégia de governança do MCP

A outra capacidade crítica que o MCP oferece às organizações é o suporte à governança centralizada. Sua estratégia de governança de MCP deve abordar a autenticação e a autorização tanto para os servidores MCP quanto para os recursos que eles acessam. Também deve abordar a limitação de taxas para proteger os recursos posteriores, as métricas operacionais para monitorar o uso e o desempenho das ferramentas e gerenciar implantações e distribuição de servidores MCP.

Autenticação e autorização

Uma das partes mais importantes de sua estratégia de autenticação e autorização é gerenciar o acesso a recursos downstream dos servidores MCP. Quando um usuário chama um agente, a autenticação e a autorização são realizadas para garantir que o usuário tenha permissões para chamar o agente. Em seguida, o agente orquestra a chamada de ferramentas específicas nos servidores MCP. Você precisa decidir como autorizar o acesso por ferramenta.

Uma opção é a machine-to-machine autorização, em que o consentimento ou a interação do usuário não são necessários. Por exemplo, uma invocação de agente baseada em tempo usa um servidor MCP para coletar registros de um aplicativo e analisá-los. Nesse cenário, o agente está pré-autorizado a acessar os dados especificados. A segunda opção é o acesso delegado pelo usuário, em que um usuário fornece seu consentimento para acessar dados e recursos específicos do usuário.

A tabela a seguir mostra os padrões de autenticação e autorização.

Fator	Acesso delegado pelo usuário	Machine-to-machine
Propriedade de dados	Autorização específica do usuário para dados	Dados de todo o sistema ou da organização
Interação com o usuário	O usuário está presente e pode consentir	Sem interação com o usuário
Tempo de operação	Interativo ou em tempo real	Em segundo plano, programado ou em lote
Escopo de permissões	As permissões variam de acordo com o usuário	Permissões consistentes no nível do agente

O acesso delegado pelo usuário requer uma implementação cuidadosa e deve ser desenvolvido com sua equipe de segurança. Os agentes devem ser capazes de avaliar quais ferramentas um LLM selecionou e se elas precisam de autorização adicional. As ferramentas MCP devem incluir descrições para indicar seus requisitos de autenticação e autorização e onde recuperar os tokens de acesso. Os aplicativos cliente devem oferecer suporte a solicitações de autenticação intermediárias, e o cliente MCP deve fornecer as credenciais recuperadas de volta ao agente para cada chamada de ferramenta.

Você deve garantir que as ferramentas MCP sempre tenham seus próprios tokens para acessar recursos externos e que o acesso seja registrado e auditado. As credenciais do usuário não devem ser propagadas por meio de seu sistema agente. Por exemplo, seus servidores MCP não devem usar o mesmo token para acessar dados que foram usados para invocar o agente. As chamadas downstream devem usar tokens com escopo explícito e gerados para fins específicos. Isso ajuda a fornecer proteções adicionais para impedir o acesso não intencional aos dados em nome das ações. Também pode ajudar a evitar que alucinações produzam resultados indesejados. Imagine que um usuário com permissões totais de administrador peça a um agente que clone um banco de dados de produção para uso na pré-produção. Para fazer isso, o usuário só precisa de `CREATE` permissões `READ` e permissões. Digamos que o LLM alucine e acredite que precisa limpar o banco de dados antigo como parte dessa solicitação. Se ele reutilizar as credenciais do usuário, provavelmente será bem-sucedido porque as credenciais originais do usuário têm permissões. `DELETE` Em vez disso, se o servidor MCP usar um token intencionalmente reduzido para a solicitação com apenas `READ` `CREATE` permissões, a tentativa de excluir o banco de dados de produção falhará.

Você pode usar o Amazon Bedrock AgentCore Identity para ajudar a implementar esses padrões. Certifique-se de fazer uma escolha intencional sobre se as permissões para listar e invocar ferramentas hospedadas por um servidor MCP implicam permissão para os recursos externos que o servidor MCP expõe. Esse fluxo de identidade do servidor MCP para o recurso e de volta para o usuário depende do tipo de serviço de autenticação e autorização que está sendo usado. Você deve decidir como isso é tratado em grande escala para seus servidores MCP.

Ao projetar seus padrões de autenticação e autorização, implemente mecanismos de isolamento de token que recuperem diferentes tokens de acesso para cada ferramenta acessada. Não reutilize tokens entre ferramentas e servidores. AgentCore A identidade fornece esse recurso de isolamento de token. Ele gerencia automaticamente os tokens da carga de trabalho (para machine-to-machine autenticação) e os tokens do usuário (para acesso delegado pelo usuário) para garantir a separação adequada e evitar o escalonamento de permissões. Isso é especialmente importante ao incorporar servidores MCP remotos ou gateways MCP.

Melhores práticas para autenticação e autorização de MCP

- Separação de tokens — Não passe tokens portadores dos chamadores para os serviços posteriores. Valide se o campo aud (audiência) corresponde ao servidor que está recebendo o token. A reivindicação do público especifica para qual serviço o token se destina, impedindo a reutilização não autorizada de tokens em diferentes servidores MCP.
- Selecione uma abordagem de acesso — Escolha entre machine-to-machine acesso delegado pelo usuário para cada ferramenta que seus servidores MCP fornecem. Considere agrupar ferramentas no mesmo servidor MCP que usam o mesmo padrão de autenticação.

Controlando a carga

Como em qualquer sistema distribuído, você deve considerar como controlar a carga em sua frota de servidores MCP. Primeiro, você considera se deve implementar a limitação de taxa em seus servidores MCP e onde implementar os limites. Se você optar por não implementar a limitação de taxa, você repassa qualquer limitação de taxa executada por recursos posteriores. Muitos sistemas optam pelo limite de tarifas com base nos atributos da solicitação, como ID do usuário ou da conta. Valide se as solicitações enviadas aos serviços downstream têm esses mesmos atributos para que vários usuários não sejam afetados pela carga gerada por outro usuário.

Se você optar por implementar a limitação de taxa, a abordagem recomendada é implementar a limitação de taxa primária no nível do servidor MCP, com serviços de back-end fornecendo proteção secundária e agentes adaptando seu comportamento com base no feedback do limite de taxa. Considere se os limites de taxa são por servidor MCP ou por ferramenta. Os limites de taxa por servidor MCP ajudam a proteger sua frota e serviços de servidores MCP em um ambiente multilocatário. No entanto, isso pode ser muito grosseiro. Os limites de taxa por ferramenta foram projetados para evitar a sobrecarga de recursos posteriores que podem não se limitar suficientemente. Se uma ferramenta chamar várias APIs, você deve definir o limite de taxa para se alinhar à taxa mais baixa permitida por elas APIs.

A transmissão de informações de limite de taxa em cabeçalhos HTTP também pode ser uma métrica útil para usuários e sistemas automatizados para ajudar a gerenciar sua própria taxa de solicitações e estratégias de repetição. Por exemplo, você pode enviar esses cabeçalhos de volta para o agente a partir do seu servidor MCP, conforme mostrado no exemplo a seguir:

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

```
X-RateLimit-Reset: 1640995200
```

Além disso, considere a redução de carga para proteger o serviço geral quando nenhum cliente estiver excedendo um limite de taxa, mas a carga estiver afetando o desempenho do sistema.

Melhores práticas para controlar a carga

- Escolha uma abordagem de limitação de taxa — Planeje limitar a taxa de usuários individuais com base no uso de recursos posteriores ou no uso de seu servidor e ferramentas MCP.
- Considere a redução de carga — proteja sua frota de servidores MCP da sobrecarga geral que não é causada por um único ou poucos clientes.

Métricas operacionais

As principais métricas a serem capturadas para implementações de MCP devem se concentrar na experiência do cliente que elas oferecem. Essas métricas geralmente incluem uso de tokens, precisão na seleção de ferramentas, número de ferramentas registradas com o agente e latência da ferramenta. Por exemplo, monitorar os tokens de saída retornados por cada ferramenta permite que você defina alarmes quando as ferramentas excedem um limite para o uso da janela de contexto. Quando uma ferramenta excede esse limite, talvez você queira revisar o comportamento da ferramenta. Isso também está relacionado à estratégia de design da ferramenta MCP. As métricas de precisão da seleção de ferramentas indicam o quão bem os agentes escolhem as ferramentas apropriadas para determinadas tarefas, enquanto a velocidade de execução e as taxas de sucesso destacam gargalos de desempenho e problemas de confiabilidade.

Por exemplo, para avaliar as métricas de seleção e precisão do uso de ferramentas, AWS as equipes criaram conjuntos de dados dourados para testes de regressão. Os conjuntos de dados foram gerados sinteticamente usando registros históricos LLMs de invocação da API em consultas de usuários. Usando as métricas predefinidas de seleção e uso de ferramentas (como precisão da seleção de ferramentas, precisão dos parâmetros da ferramenta e precisão das chamadas de funções em vários turnos), AWS as equipes poderiam avaliar objetivamente a capacidade do agente de IA de identificar corretamente as ferramentas apropriadas, preencher seus parâmetros com valores precisos e manter sequências coerentes de invocação de ferramentas em turnos de conversação.

Medir métricas sobre o número de ferramentas registradas com um agente pode ajudá-lo a identificar possíveis desafios de gerenciamento de janelas de contexto, bem como mudanças nas ferramentas

disponíveis apresentadas pelos servidores MCP. Você deve revisar regularmente as métricas operacionais que indicam a experiência do usuário com o servidor e as ferramentas MCP.

Colaboradores

Autoria

- Alex Torres, arquiteto de soluções sênior, AWS
- Saikat Gomes, gerente sênior de soluções para clientes, AWS
- Mike Haken, arquiteto sênior de soluções, AWS
- Sereja Das, engenheira principal, AWS

Análise

- Ted Swinyar, gerente de arquitetura de soluções, AWS
- Raju Patil, cientista de dados sênior, AWS

Redação técnica

- Lilly AbouHarb, redatora técnica sênior, AWS

Histórico do documento

A tabela a seguir descreve alterações significativas feitas neste guia. Se desejar receber notificações sobre futuras atualizações, inscreva-se em um feed RSS.

Alteração	Descrição	Data
Publicação inicial	—	16 de março de 2026

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.