



Padrões e fluxos de trabalho de IA da Agentic em AWS

AWS Recomendações



AWS Recomendações: Padrões e fluxos de trabalho de IA da Agentic 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 desprestige 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	1
Objetivos	1
Sobre esta série de conteúdo	2
Padrões de agentes	3
Agentes básicos de raciocínio	4
Arquitetura	4
Description	5
Capacidades	6
Limitações	6
Casos de uso comuns	6
Orientação para implementação	7
Resumo	7
Arquitetura	7
Description	8
Capacidades	9
Casos de uso comuns	9
Orientação para implementação	9
Resumo	10
Tool-based agentes para chamar funções	10
Arquitetura	10
Description	11
Capacidades	12
Casos de uso comuns	12
Orientação para implementação	12
Resumo	13
Tool-based agentes para servidores	13
Arquitetura	13
Description	14
Capacidades	15
Casos de uso comuns	15
Orientação para implementação	15
Resumo	16
Computer-use agentes	16

Arquitetura	16
Description	17
Capacidades	18
Casos de uso comuns	18
Orientação para implementação	19
Resumo	19
Agentes de codificação	19
Arquitetura	19
Description	20
Capacidades	21
Casos de uso comuns	21
Orientação para implementação	22
Resumo	22
Agentes de fala e voz	22
Arquitetura	22
Description	23
Capacidades	24
Casos de uso comuns	24
Orientação para implementação	25
Resumo	25
Agentes de orquestração de fluxo de trabalho	25
Arquitetura	26
Description	26
Capacidades	27
Casos de uso comuns	27
Orientação para implementação	27
Resumo	28
Memory-augmented agentes	28
Arquitetura	28
Description	29
Capacidades	30
Casos de uso comuns	30
Implementação de agentes com memória aumentada	30
Implementando a solicitação com injeção de memória	31
Resumo	32
Agentes de simulação e de teste	32

Arquitetura	32
Description	33
Capacidades	34
Casos de uso comuns	34
Orientação para implementação	35
Resumo	36
Agentes observadores e de monitoramento	36
Arquitetura	37
Description	37
Capacidades	38
Casos de uso comuns	38
Orientação para implementação	38
Resumo	39
Multi-agent colaboração	39
Description	41
Capacidades	42
Casos de uso comuns	42
Orientação para implementação	42
Resumo	43
Conclusão	43
Takeaways	44
Fluxos de trabalho do LLM	45
Visão geral da LLM-augmented cognição	45
Fluxo de trabalho para encadeamento imediato	46
Description	47
Capacidades	47
Casos de uso comuns	48
Fluxo de trabalho para roteamento	48
Capacidades	49
Casos de uso comuns	49
Fluxo de trabalho para paralelização	50
Capacidades	51
Casos de uso comuns	51
Fluxo de trabalho para orquestração	51
Capacidades	53
Casos de uso comuns	53

Fluxo de trabalho para avaliadores e ciclos de reflexão e refinamento	53
Casos de uso comuns	54
Capacidades	54
Conclusão	55
Padrões de fluxo de trabalho agentes	56
De sistemas orientados a eventos a sistemas aumentados por cognição	56
Event-driven arquitetura	57
Cognition-augmented fluxos de trabalho	58
Insights principais	59
Padrões de saga de encadeamento imediato	59
Coreografia saga	60
Padrão de encadeamento imediato	61
Coreografia do agente	61
Takeaways	63
Padrões de despacho dinâmico de roteamento	63
Envio dinâmico	64
LLM-based roteamento	65
Roteador de agente	66
Takeaways	67
Padrões de paralelização e coleta de dispersão	67
Scatter-gather	68
LLM-based paralelização (cognição de dispersão e coleta)	70
Paralelização de agentes	70
Takeaways	71
Padrões de orquestração da saga	71
Orquestração de eventos	72
Role-based sistema de agentes (orquestrador)	73
Supervisor	73
Takeaways	75
O avaliador reflete e refina os padrões de loop	75
Circuito de controle de feedback	76
Circuito de controle de feedback (avaliador)	78
Avaliador	78
Takeaways	79
Projetando fluxos de trabalho agentes em AWS	79
Conclusão	80

Histórico do documento	81
.....	lxxxii

Padrões e fluxos de trabalho de IA agentes em AWS

Aaron Sempf e Andrew Hooker, Amazon Web Services

Julho de 2025 ([histórico do documento](#))

As organizações estão adotando grandes modelos de linguagem (LLMs) e agentes de software para resolver problemas dinâmicos de vários domínios usando uma nova disciplina arquitetônica chamada padrões agentes. Os padrões de agência são projetos fundamentais e construções modulares que são usados para projetar e orquestrar agentes de IA orientados a objetivos em muitos contextos.

Público-alvo

Este guia é destinado a arquitetos, desenvolvedores e líderes de produtos que desejam criar aplicativos inteligentes que vão além da lógica estática, lógica simbólica e automação determinística.

Objetivos

Este guia fornece uma estrutura de design e uma abordagem de implementação para sistemas de agentes de IA que operam de forma autônoma, permanecendo controláveis e alinhados com suas metas. Ele conecta padrões arquitetônicos orientados por eventos a várias alternativas de agentes, demonstrando como criar sistemas de agentes de nível de produção usando arquiteturas nativas em nuvem. Os seguintes assuntos são discutidos neste guia:

- **Padrões do agente** — Os padrões do agente são modelos de design reutilizáveis que descrevem a estrutura e o comportamento de agentes individuais. Isso inclui agentes de raciocínio, agentes de recuperação aumentada, agentes de codificação, interfaces de voz, orquestradores de fluxo de trabalho e sistemas multiagentes colaborativos. Cada padrão ilustra como os agentes percebem, raciocinam, agem e aprendem, mapeados. Serviços da AWS
- **Fluxos de trabalho do LLM** — Os fluxos de trabalho se concentram em como os agentes usam LLMs para raciocinar. Eles exploram estratégias de estímulo e mecanismos de planejamento e descrevem como LLMs são usados não apenas para gerar texto, mas também para impulsionar comportamentos estruturados, interpretáveis e confiáveis em um loop de agentes.
- **Padrões de fluxo de trabalho agentes** — Os padrões de fluxo de trabalho descrevem como vários agentes, ferramentas e ambientes interagem para formar sistemas autônomos. Isso inclui padrões para orquestração de tarefas, delegação de subagentes, coordenação baseada em eventos,

observabilidade e controle. Esses aspectos promovem arquiteturas de IA escaláveis, combináveis e auditáveis.

Sobre esta série de conteúdo

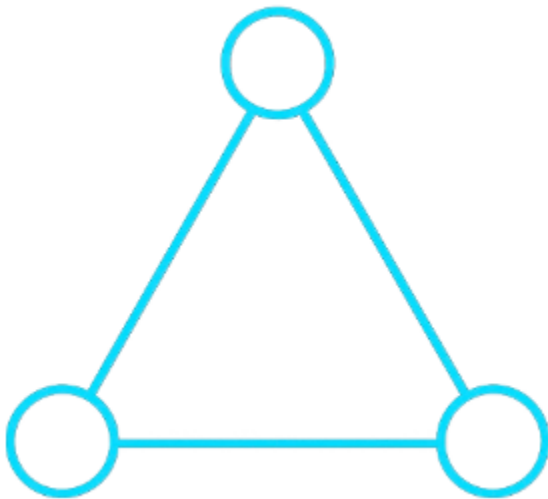
Este guia faz parte de uma série sobre IA agente em AWS. Para obter mais informações e ver os outros guias desta série, consulte [Agentic AI](#) no site de Orientação AWS Prescritiva.

Padrões de agentes

Os padrões de agentes são blocos de construção reutilizáveis e compostos que podem ser personalizados para domínios, casos de uso e níveis de complexidade específicos. Os sistemas Agentic diferem, no entanto, das aplicações tradicionais. No centro de todos os projetos de agentes de IA está um modelo conceitual ancorado nos três princípios fundamentais a seguir:

- Assíncrono — os agentes operam em ambientes pouco acoplados e ricos em eventos
- Autonomia — Os agentes agem de forma independente, sem controle humano ou externo
- Agência — Os agentes agem com um propósito, em nome de um usuário ou sistema, em direção a objetivos específicos

O triângulo no diagrama a seguir representa os principais blocos de construção de um agente de software: percepção, razão e ação. Isso permite que um sistema agente observe, tome decisões e aja dentro de seu ambiente.



Por design, os padrões agentes fornecem uma linguagem de design modular para a criação de sistemas de IA, o que significa que eles são acessíveis, operacionais, extensíveis e prontos para produção. O projeto desses sistemas requer atenção cuidadosa às três dimensões inter-relacionadas a seguir, que serão discutidas mais adiante neste guia.

Nesta seção

- [Agentes básicos de raciocínio](#)
- [Tool-based agentes para chamar funções](#)

- [Tool-based agentes para servidores](#)
- [Computer-use agentes](#)
- [Agentes de codificação](#)
- [Agentes de fala e voz](#)
- [Agentes de orquestração de fluxo de trabalho](#)
- [Memory-augmented agentes](#)
- [Agentes de simulação e de teste](#)
- [Agentes observadores e de monitoramento](#)
- [Multi-agent colaboração](#)

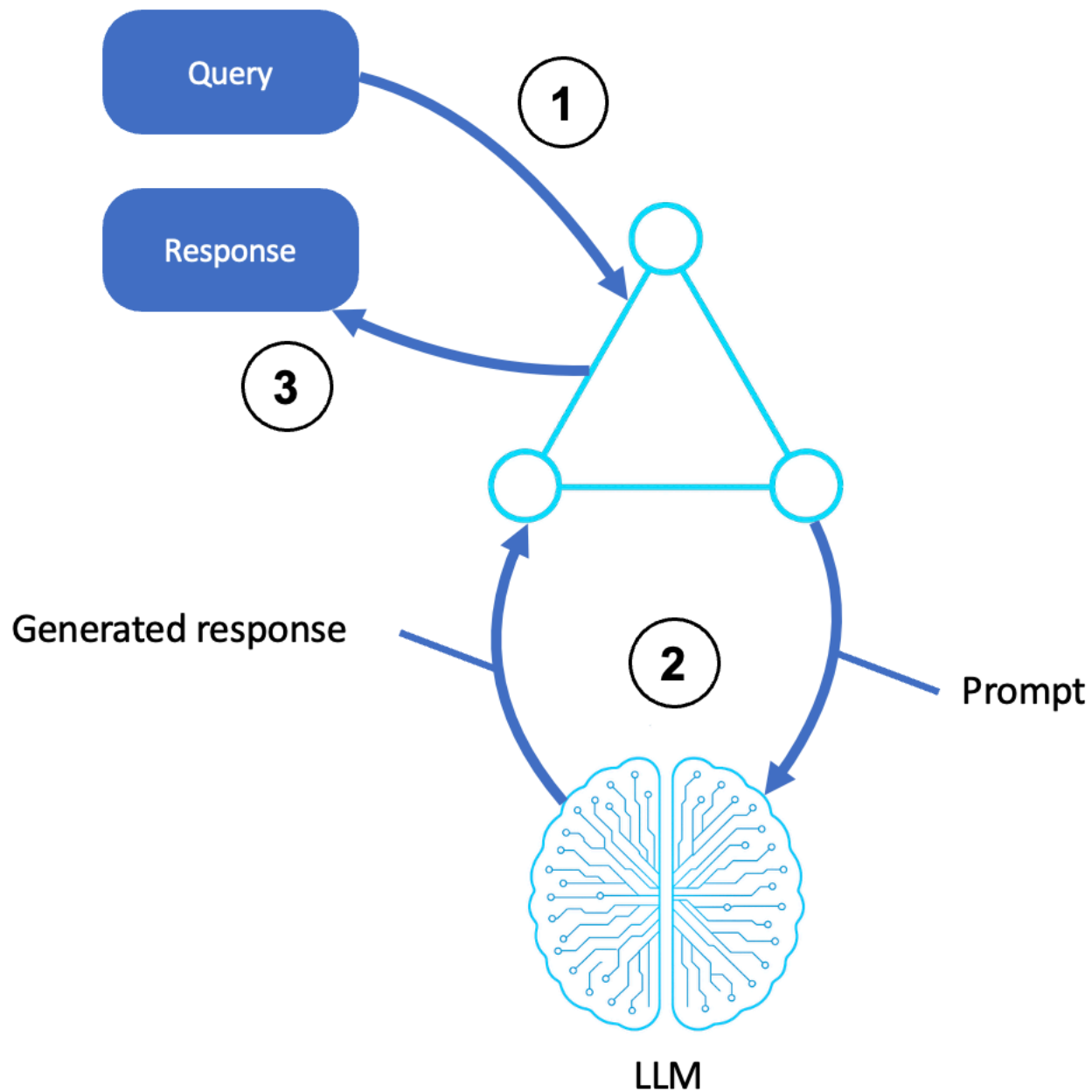
Agentes básicos de raciocínio

Um agente de raciocínio básico é a forma mais simples de IA agente que realiza inferência lógica ou tomada de decisão em resposta a uma consulta. Ele aceita a entrada de um usuário ou sistema e processa consultas e gera respostas usando prompts estruturados.

Esse padrão é útil para tarefas que exigem raciocínio, classificação ou resumo em uma única etapa com base em um determinado contexto. Ele não usa memória, ferramentas ou gerenciamento de estado, o que o torna sem estado, leve e altamente combinável em grandes fluxos de trabalho.

Arquitetura

O fluxo de um agente de raciocínio básico é mostrado no diagrama a seguir:



Description

1. Recebe uma entrada

- Um usuário, sistema ou agente upstream envia uma consulta ou instrução.
- A entrada é transferida para o shell do agente ou para a camada de orquestração.
- Essa etapa inclui qualquer pré-processamento, modelagem imediata e identificação de metas.

2. Invoca o LLM

- O agente transforma a consulta em um prompt estruturado e a envia para um LLM (por exemplo, por meio do Amazon Bedrock).
- O LLM gera uma resposta com base no prompt usando conhecimento e contexto pré-treinados.
- A saída gerada pode incluir etapas de raciocínio (cadeia de pensamento), respostas finais ou opções classificadas.

3. Retorna uma resposta

- A saída gerada é retransmitida para a interface do agente.
- Isso pode incluir formatação, pós-processamento ou uma resposta de API.

Capacidades

- Suporta linguagem natural ou entrada estruturada
- Usa engenharia rápida para orientar o comportamento
- Sem estado e escalável
- Pode ser incorporado à interface do usuário, CLI, APIs e pipelines

Limitações

- Sem memória ou consciência histórica
- Sem interação com ferramentas externas ou fontes de dados
- Limitado ao que o LLM sabe no momento da inferência

Casos de uso comuns

- Perguntas e respostas conversacionais
- Explicações e resumos de políticas
- Orientação para a tomada de decisões
- Fluxos de chatbot leves e automatizados
- Classificação, rotulagem e pontuação

Orientação para implementação

Você pode usar as seguintes ferramentas e serviços para criar um agente de raciocínio básico:

- Amazon Bedrock para invocação de LLM (Anthropic, AI21, Meta)
- Amazon API Gateway ou AWS Lambda para expô-lo como um microsserviço sem estado
- Modelos de prompt armazenados no Parameter Store ou como código AWS Secrets Manager

Resumo

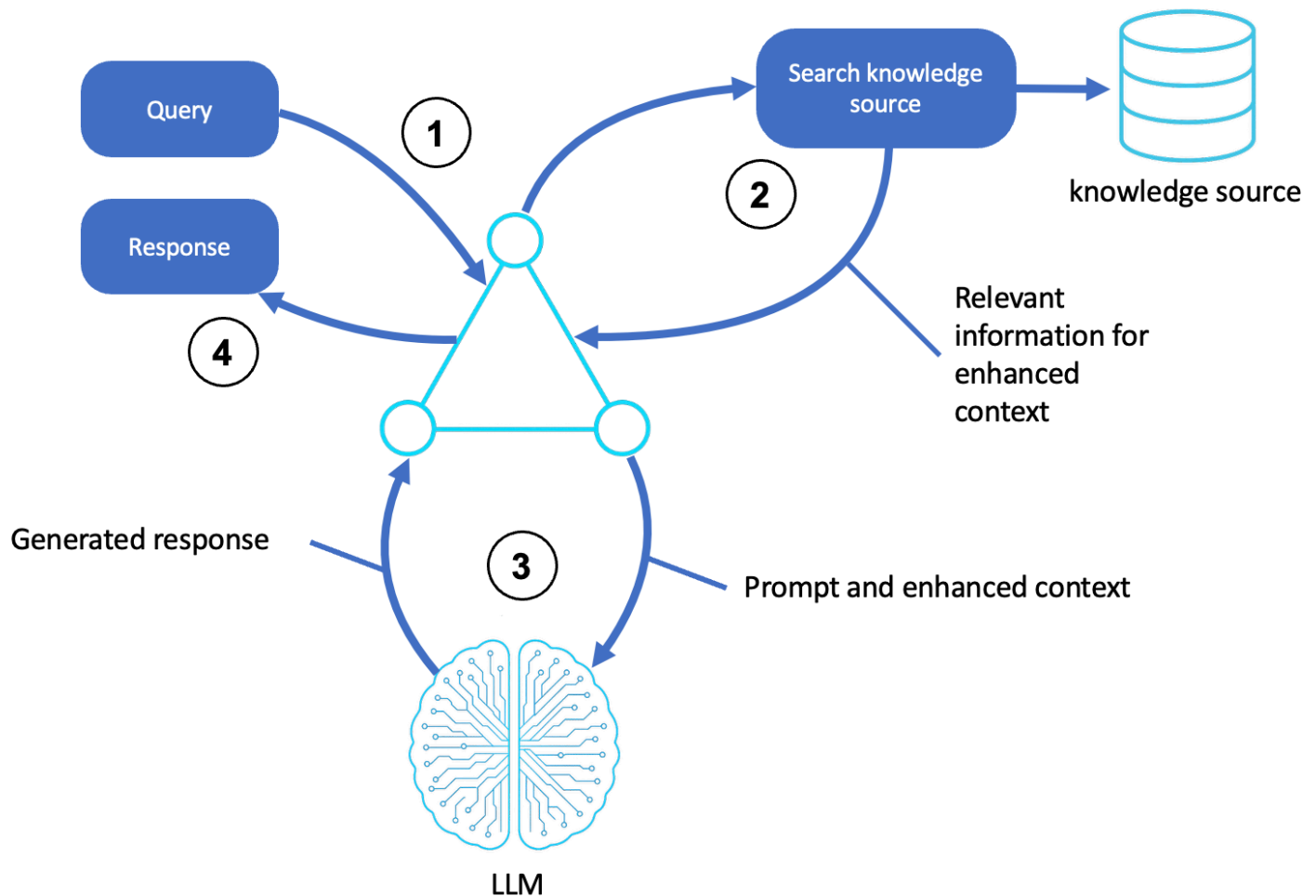
O agente de raciocínio básico é fundamental por causa de sua estrutura simples. Ele tem recursos essenciais que transformam metas em caminhos de raciocínio que levam a resultados inteligentes. Esse padrão geralmente é um ponto de partida para padrões avançados, como agentes baseados em ferramentas e agentes que usam geração aumentada de recuperação (RAG). Também é um componente confiável e modular de grandes fluxos de trabalho.

Agente RAG

Retrieval-augmented a geração (RAG) é uma técnica que combina recuperação de informações com geração de texto para criar respostas precisas e contextuais. O RAG permite que os agentes recuperem informações externas relevantes antes de contratar o LLM. Ele amplia a memória efetiva e a precisão do raciocínio de um agente ao basear suas decisões em informações atualizadas, factuais ou específicas do domínio. Em contraste com os LLMs sem estado que dependem exclusivamente de pesos pré-treinados, o RAG tem uma camada externa de pesquisa de conhecimento que aprimora dinamicamente as solicitações com o contexto.

Arquitetura

A lógica do padrão RAG é ilustrada no diagrama a seguir:



Description

1. Recebe uma consulta

- Um usuário ou sistema upstream envia uma consulta ou meta ao agente.
- O shell do agente aceita a solicitação e a formata como uma solicitação de raciocínio.

2. Pesquisa uma fonte externa

- O agente identifica os conceitos e a intenção da consulta.
- Ele consulta uma fonte de conhecimento, como um repositório vetorial, banco de dados ou índice de documentos usando pesquisa semântica ou correspondência de palavras-chave.
- As passagens, documentos ou entidades mais relevantes são recuperados para uso na próxima etapa.

3. Gera uma resposta contextual

- O agente aumenta o prompt com as informações recuperadas, formando uma entrada contextualizada para o LLM.
- O LLM processa qualquer entrada usando raciocínio generativo (por exemplo, cadeia de pensamento ou reflexão) para produzir uma resposta precisa.

4. Retorna a saída final

- O agente prepara a saída envolvendo-a em qualquer cabeçalho de comunicação ou na formatação necessária e, em seguida, a retorna ao usuário ou ao sistema de chamada.
- (Opcional) Os documentos recuperados e a saída do LLM podem ser registrados, pontuados e armazenados na memória para futuras consultas.

Capacidades

- Fact-grounded produção mesmo em domínios de longa duração ou específicos da empresa
- Extensão de memória sem ajustar o modelo
- Contexto dinâmico baseado em cada consulta e estado do usuário
- Totalmente compatível com bancos de dados vetoriais, índices semânticos e filtragem de metadados

Casos de uso comuns

- Assistentes de conhecimento corporativo
- Bots de conformidade regulatória
- Co-pilotos de suporte ao cliente
- Search-enhanced chatbots
- Agentes de documentação para desenvolvedores

Orientação para implementação

Use as seguintes ferramentas e serviços para criar um agente que usa o RAG:

- Amazon Bedrock para invocação de LLM
- Amazon Kendra OpenSearch ou Amazon Aurora para documentação ou pesquisa estruturada de dados

- Amazon Simple Storage Service (Amazon S3) (Amazon S3) para armazenamento de documentos
- AWS Lambda para orquestrar a pesquisa, o prompt e a inferência do LLM
- Knowledge-based integrações com agentes (usando plug-ins de memória, recuperadores semânticos ou Amazon Bedrock)

Resumo

O agente RAG conecta o raciocínio do modelo estático à inteligência dinâmica do mundo real. Ele capacita os agentes com a capacidade de pesquisar o que não sabem, sintetizar respostas a partir do conhecimento recuperado e produzir respostas auditáveis e de alta confiança.

Os padrões RAG são a base para a criação de agentes inteligentes que escalam o acesso ao conhecimento sem reciclagem. Geralmente, é um precursor de padrões de orquestração mais complexos que envolvem o uso de ferramentas, planejamento e memória de longo prazo.

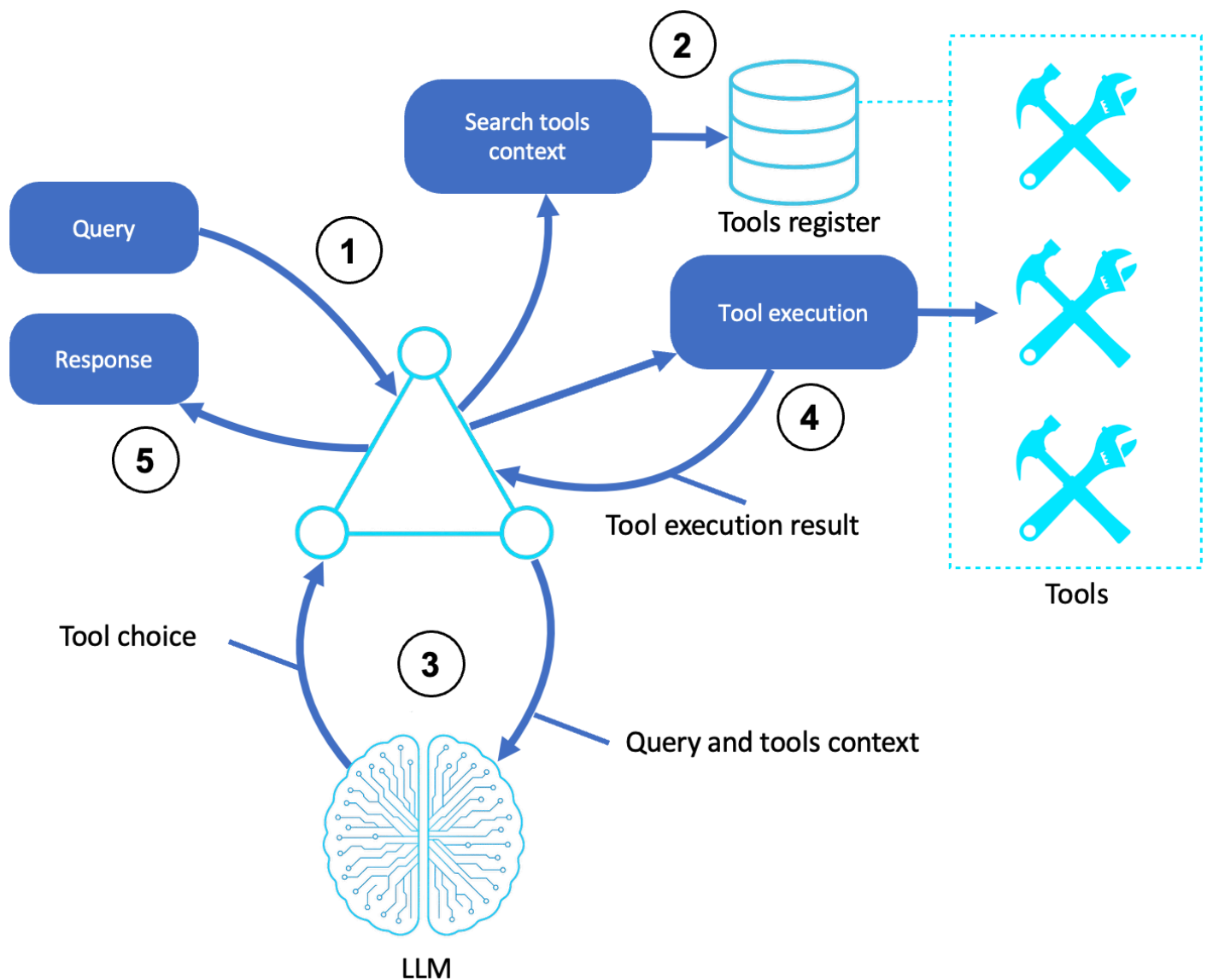
Tool-based agentes para chamar funções

Tool-based os agentes ampliam as capacidades dos agentes de raciocínio invocando funções externas ou APIs para concluir tarefas que vão além do raciocínio somente de linguagem. Esse padrão usa um LLM para decidir qual ferramenta usar e, em seguida, gera argumentos de chamada e incorpora a saída de uma ferramenta em seu ciclo de raciocínio.

Esse padrão permite que os agentes ajam em vez de apenas fornecer respostas. A interface da ferramenta representa qualquer recurso que possa ser chamado, desde cálculos aritméticos e pesquisas de banco de dados até APIs externas e serviços em nuvem.

Arquitetura

Um agente baseado em ferramentas para chamar funções é mostrado no diagrama a seguir:



Description

1. Recebe consulta

- O agente recebe uma consulta ou tarefa em linguagem natural do usuário ou do sistema de chamada.

2. Pesquisas por ferramentas

- O agente usa metadados internos ou um registro de ferramentas para pesquisar ferramentas, esquemas e recursos relevantes disponíveis.

3. Seleciona e invoca ferramentas

- O LLM recebe os metadados da consulta e da ferramenta (por exemplo, nomes de funções, tipos de entrada e descrições) em seu prompt.
 - Ele escolhe a ferramenta mais relevante, constrói argumentos de entrada e retorna uma chamada de função estruturada.
4. Executa a ferramenta escolhida
 - O shell do agente ou o executor de ferramentas executa a função selecionada e retorna o resultado (por exemplo, uma saída de API, valor de banco de dados ou computação).
 5. Retorna uma resposta
 - O LLM passa os resultados para o agente, diretamente ou como parte de um prompt atualizado. Em seguida, ele retorna um resultado em linguagem natural.

Capacidades

- Seleção dinâmica de ferramentas com base no contexto da tarefa
- Schema-based solicitação (OpenAPI, esquema AWS JSON, interface de função)
- Interpretação dos resultados e encadeamento dos resultados no raciocínio
- Operações sem estado ou com reconhecimento de sessão

Casos de uso comuns

- Assistentes virtuais com acesso externo a dados
- Calculadoras e estimadores financeiros
- API-based trabalhadores do conhecimento
- LLMs que invocam, SageMaker endpoints AWS Lambda da Amazon e serviços SaaS

Orientação para implementação

Use o seguinte para criar agentes baseados em ferramentas para chamar funções:

- Amazon Bedrock com suporte para chamadas funcionais (Anthropic Claude)
- AWS Lambda como um back-end de execução de ferramentas
- Amazon API Gateway ou AWS Step Functions para orquestração de ferramentas

- Amazon DynamoDB ou Amazon Relational Database Service (Amazon RDS) para metadados de ferramentas sensíveis ao contexto
- EventBridge Pipelines da Amazon ou AWS Step Functions que mapeiam estados para rotear saídas

Resumo

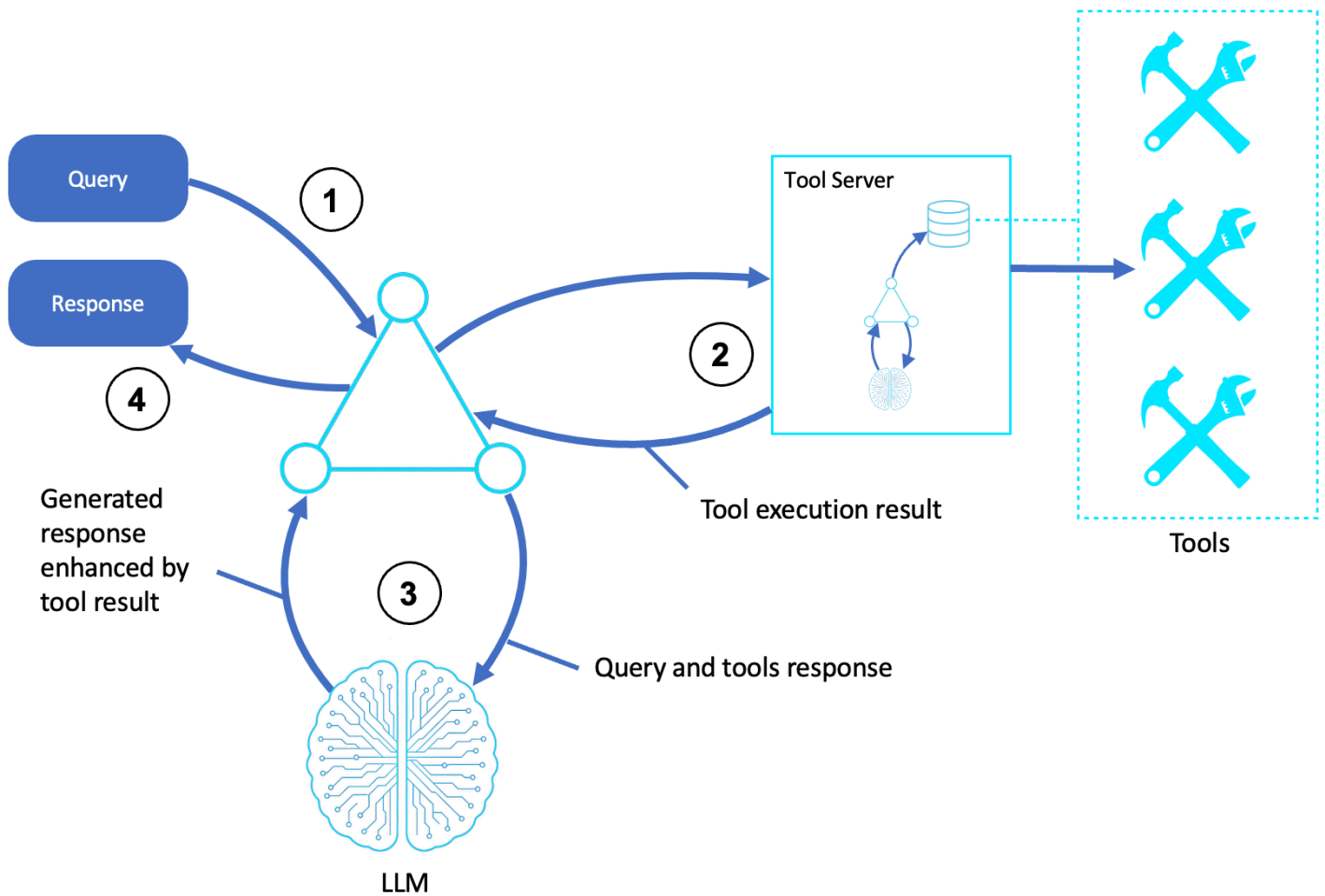
Tool-based agentes de chamada de funções representam uma mudança da compreensão da linguagem para a execução de ações. Esses agentes invocam ferramentas dinâmicas e sensíveis ao contexto, mantendo o raciocínio do LLM, transformando assistentes passivos em sistemas que concluem tarefas, acessam serviços e integram operações comerciais. Esse padrão é um componente importante da IA agente em ambientes corporativos, especialmente quando combinado com esquemas declarativos, estruturas de autorização e sistemas multiagentes.

Tool-based agentes para servidores

Tool-based agentes para servidores aprimoram os agentes de chamada de funções delegando a execução da ferramenta a um servidor externo que tem um ambiente de tempo de execução dedicado para ferramentas, scripts e agentes compostos. Diferentemente das chamadas de função em linha que o loop do agente seleciona e invoca, os agentes baseados em servidor terceirizam a lógica e o pipeline de execução para outros agentes ou sistemas. Isso fornece recursos avançados, como encadeamento de várias ferramentas, execução isolada e raciocínio especializado. Os servidores de ferramentas são ideais para ações complexas, com estado ou que consomem muitos recursos, nas quais as próprias ferramentas podem envolver modelos de IA, regras de negócios ou ambientes separados.

Arquitetura

O seguinte é um padrão para agentes baseados em ferramentas para servidores:



Description

1. Recebe consulta

- Um usuário ou sistema envia uma solicitação ao shell do agente.
- O agente interpreta a consulta e se prepara para enviá-la para um servidor de ferramentas.

2. Executa processos do servidor de ferramentas

- O agente envia a tarefa, junto com os parâmetros estruturados, para um servidor de ferramentas.
- O servidor de ferramentas pode então:
 - Execute scripts ou lógica em sistemas computacionais dedicados (por exemplo AWS Lambda, contêineres ou Amazon SageMaker)
 - Use seu próprio subagente com raciocínio LLM para selecionar e executar ferramentas
 - Gerencie dependências, novas tentativas ou fluxos de execução em várias etapas

- Resultados de saída para o agente primário quando a tarefa estiver concluída
3. Usa o raciocínio LLM com a saída da ferramenta
 - O agente invoca um LLM, transmitindo a consulta original e o resultado do servidor de ferramentas como parte do prompt.
 - O LLM sintetiza uma resposta que incorpora as informações recém-adquiridas.
 4. Retorna uma resposta
 - O agente retorna uma resposta estruturada ou em linguagem natural ao usuário ou ao sistema de chamada.
 - (Opcional) Os resultados podem ser armazenados na memória ou nos registros de auditoria.

Capacidades

- As ferramentas são invocadas fora do ciclo de execução do agente primário
- A execução da ferramenta pode envolver chamadas LLM, cadeias lógicas ou subagentes
- O agente atua como um controlador ou despachante, não apenas como um invólucro de ferramentas
- Permite composição, escalabilidade e isolamento da lógica

Casos de uso comuns

- Orquestrando cadeias de modelos (por exemplo, combinando LLM, visão e código)
- AI-driven tubulações de automação
- DevOps agentes assistentes com roteiristas
- Agentes complexos de computação financeira, simulação ou otimização
- Ferramentas multimodais (por exemplo, combinando áudio, documentação e ação)

Orientação para implementação

Você pode criar esse padrão usando o seguinte Serviços da AWS:

- Amazon Bedrock (agente, host e inferência de LLM)
- AWS Lambda, Amazon ECS ou SageMaker endpoints da Amazon como tempo de execução do servidor de ferramentas AWS Fargate

- Amazon API Gateway ou AWS App Runner para expor as APIs do servidor de ferramentas
- Amazon EventBridge para mensagens dissociadas de agente para ferramenta
- AWS Step Functions ou AWS AppFabric para compor lógica multiagente no servidor de ferramentas

Resumo

Tool-based os agentes que usam servidores são altamente modulares e escaláveis. Eles separam a lógica de decisão da execução, o que permite que o agente primário permaneça leve enquanto transfere ações complexas ou confidenciais para outros sistemas. Isso é importante para a IA agente de nível corporativo, especialmente em ambientes que exigem governança, observabilidade, isolamento, composição dinâmica ou qualquer combinação dos dois.

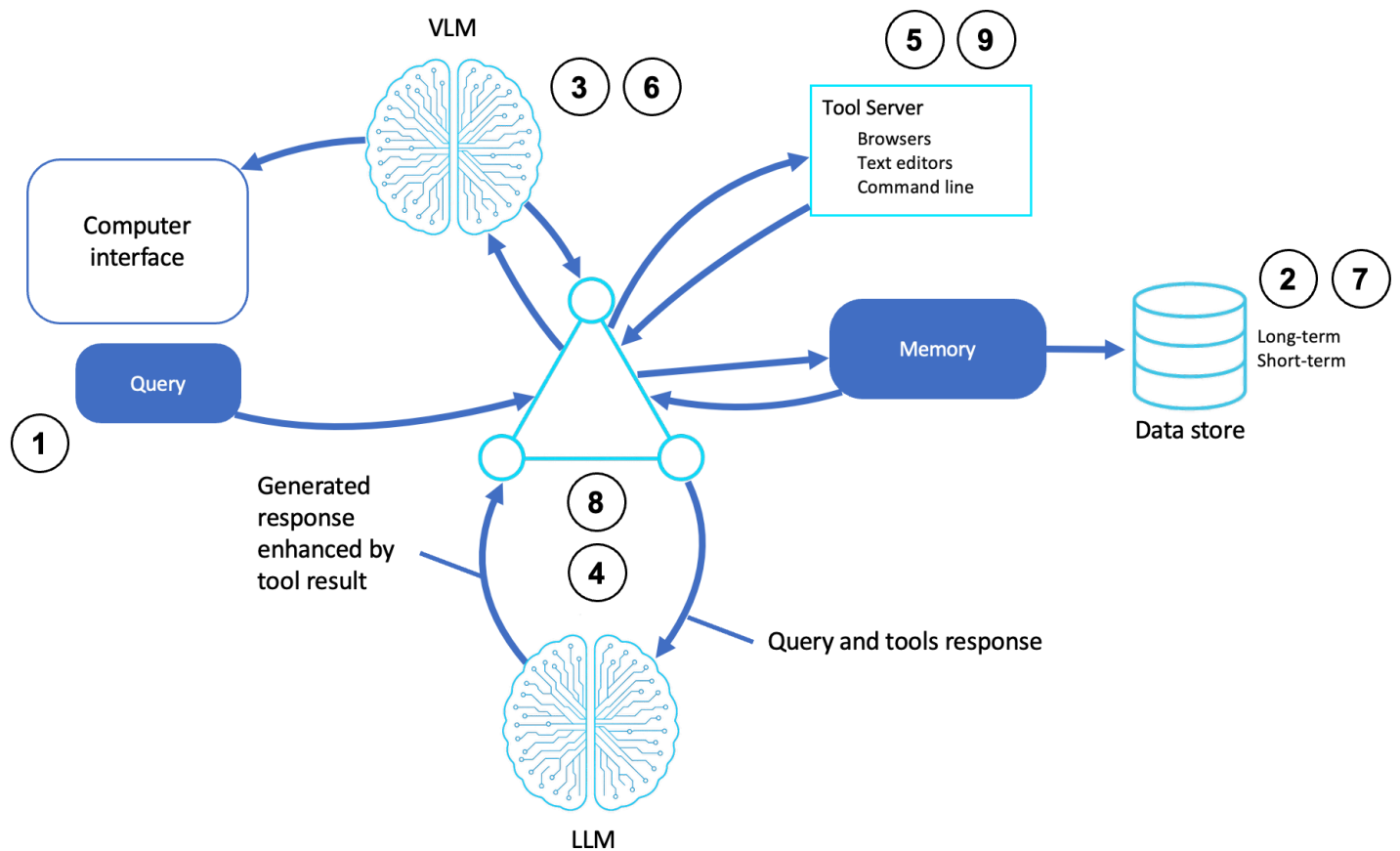
Computer-use agentes

Computer-use os agentes podem simular ou controlar ambientes digitais, como navegadores, terminais, sistemas de arquivos e aplicativos. Esses agentes interpretam a intenção do usuário, interagem com interfaces visuais e de texto e realizam ações direcionadas a objetivos combinando raciocínio LLM, modelos de linguagem visual (VLMs) e servidores de ferramentas que executam comandos ou simulam eventos de entrada.

Esse padrão é importante para automações práticas de IA, em que o agente funciona não apenas como um assistente, mas também como um proxy que executa ações como um humano faria, geralmente usando as mesmas ferramentas e ambientes.

Arquitetura

Um padrão de agente de uso do computador é mostrado no diagrama a seguir:



Description

1. Recebe uma consulta

- Uma tarefa ou solicitação é fornecida por meio de uma interface de usuário, API ou interface de linguagem natural.

2. Acessa a memória

- O agente recupera a memória de curto e longo prazo para relembrar comandos, metas e estados do sistema anteriores.

3. Analisa o contexto visual

- Um VLM observa a tela do computador, o estado do sistema ou os elementos da interface do usuário para entender um determinado contexto e identificar itens acionáveis.

4. Razões por meio de um LLM

- O LLM combina a consulta, o estado da memória, a ferramenta e a resposta do servidor para determinar a próxima ação.

5. Interage com o servidor de ferramentas

- O agente invoca ferramentas hospedadas em um servidor, que podem incluir o seguinte:
 - Navegadores (por exemplo, Chrome sem cabeçalho) e ambientes de shell
 - Editores de texto e código
 - Interfaces de script personalizadas
6. Atualiza as entradas visuais
- Se a interface do usuário do sistema mudar ou for necessária uma observação adicional, o VLM poderá reanalisar o estado da tela ou os buffers de texto.
7. Atualiza a memória
- Novos insights, estados do sistema ou feedback do usuário são gravados na memória de curto e longo prazo.
8. Formula decisões e explicações finais
- O LLM sintetiza resultados ou recomenda ações com base na consulta e na saída da ferramenta.
9. Retorna uma resposta
- O agente retorna os resultados para a interface (por exemplo, uma tarefa concluída, confirmação ou conteúdo gerado).

Capacidades

- Raciocínio multimodal com entradas visuais e textuais
- Controle sobre aplicativos por meio de simulações ou entradas API-driven
- Gerenciamento de memória para estado persistente
- Autonomia na execução de sequências (fluxos de várias etapas)

Casos de uso comuns

- Desenvolvedores de IA que escrevem e executam código em IDEs
- Computer-use agentes para fluxos de trabalho digitais repetitivos
- Usuários simulados para testes de software e garantia de qualidade
- Agentes de acessibilidade para navegar pelas interfaces de usuário por meio de instruções de voz ou de alto nível
- Automação inteligente de processos robóticos (RPA) aprimorada com o raciocínio

Orientação para implementação

- Você pode criar esse padrão usando o seguinte Serviços da AWS:
- Amazon Bedrock para LLM-based planejamento e raciocínio
- Amazon Elastic Compute Cloud (Amazon EC2) ou notebooks SageMaker Amazon para executar AWS Lambda servidores de ferramentas com ambientes de interface de usuário simulados
- Amazon Simple Storage Service (Amazon S3) ou Amazon DynamoDB para persistência de memória
- Amazon Rekognition (ou modelos personalizados) para análise de imagens de UI em cenários híbridos
- Amazon CloudWatch Logs ou AWS X-Ray para trilhas de observabilidade e auditoria

Resumo

Computer-use os agentes atuam como operadores digitais autônomos, preenchendo a lacuna entre as interações e ações humano-computador. AI-driven Ao incorporar memória, orquestração de ferramentas e VLMs, esses agentes podem interagir de forma adaptativa com sistemas projetados para humanos, executar ações, atualizar arquivos, navegar por menus e gerar respostas.

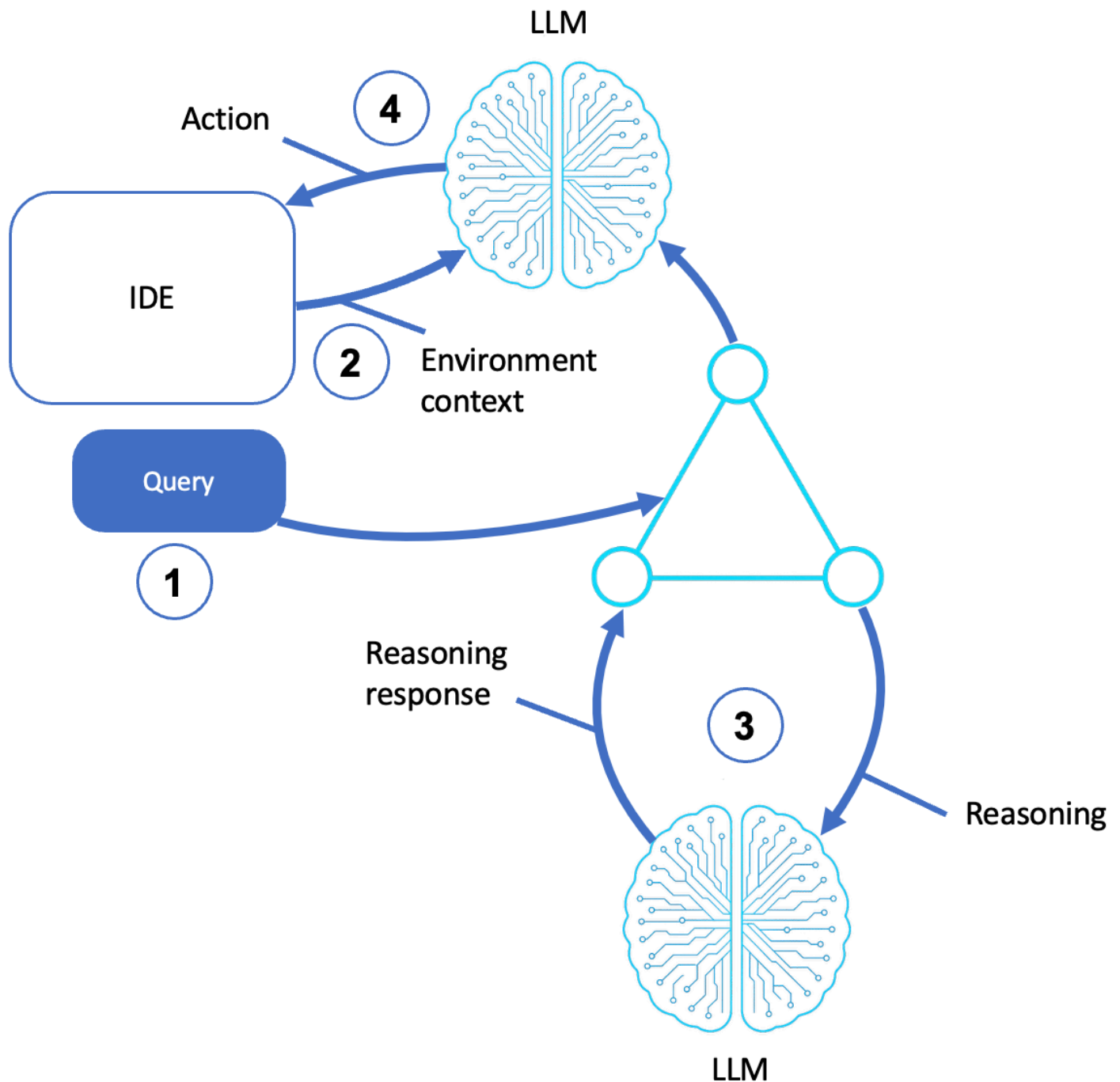
Agentes de codificação

Os agentes de codificação podem raciocinar sobre tarefas de programação, gerar ou modificar código e interagir com ambientes de desenvolvedores, como IDEs e CLIs. Esses agentes combinam a compreensão da linguagem natural com o raciocínio estruturado para auxiliar, aumentar e automatizar o desenvolvimento de software, desde a geração de funções até a correção de bugs e a criação de testes.

Diferentemente das ferramentas de preenchimento automático, os agentes de codificação interpretam ativamente as metas do usuário, consultam o ambiente de desenvolvimento em busca de contexto (por exemplo, ele abre arquivos e rastreia erros), identificam os requisitos e, em seguida, propõem e executam ações.

Arquitetura

Um padrão de agente codificador é mostrado no diagrama a seguir:



Description

1. Recebe consulta

- O usuário fornece instruções em linguagem natural por meio de uma paleta de comandos, janela de bate-papo ou CLI (por exemplo, “Adicionar registro a esta função” ou “Refatorar para facilitar a leitura”).

2. Extrai o contexto do ambiente

- O agente coleta o contexto do IDE, incluindo arquivos ativos, posição do cursor, trechos de código e tabelas de símbolos.
- Ele emite mensagens de erro, resultados de testes e saídas de outros agentes.

3. Raciocínio LLM

- O agente envia um prompt, incluindo a consulta e o contexto ambiental, para um LLM.
 - O LLM executa um raciocínio para determinar o seguinte:
 - O que precisa mudar
 - Como gerar uma solução
 - Qualquer etapa de refatoração, reescrita ou codificação

4. Executa ações

- O LLM retorna a saída para o agente e a importa para o ambiente IDE ou runtime.
- Isso pode incluir inserir ou modificar código, gerar comentários ou documentação e acionar tarefas posteriores de compilação, teste e linting.

Capacidades

- High-contextual reconhecimento (por exemplo, estado do IDE, cursor e árvore de sintaxe)
- Raciocínio iterativo de metas e feedback
- Planejamento de código opcional e separação de ações (por exemplo, primeiro motivo e depois ação)
- Funciona em fluxos de trabalho de desenvolvedores síncronos ou assíncronos

Casos de uso comuns

- Geração de código a partir de descrições de tarefas
- Refatoração e otimização de código
- Test-case geração e validação
- Explicações de erros e depuração
- Assistentes de documentação
- Co-pilotos de programação emparelhados

Orientação para implementação

- Você pode criar esse padrão usando as seguintes ferramentas e Serviços da AWS:
- Amazon Bedrock para LLM-driven geração e raciocínio
- Amazon Q Developer para sugestões e conclusões de codificação
- AWS Lambda ou Amazon Elastic Container Service (Amazon ECS) para executar e testar ambientes sandbox
- AWS Cloud9, extensões do VS Code ou integrações personalizadas de IDE para hospedar e avaliar o contexto
- Amazon Simple Storage Service (Amazon S3) (Amazon S3) para armazenar solicitações intermediárias, respostas e histórico de revisões

Resumo

Agentes de codificação são novas ferramentas de AI-powered desenvolvimento capazes de interpretar a linguagem natural, analisar o contexto, gerar alterações de código em várias etapas e integrar-se ao ciclo de vida do desenvolvimento de software.

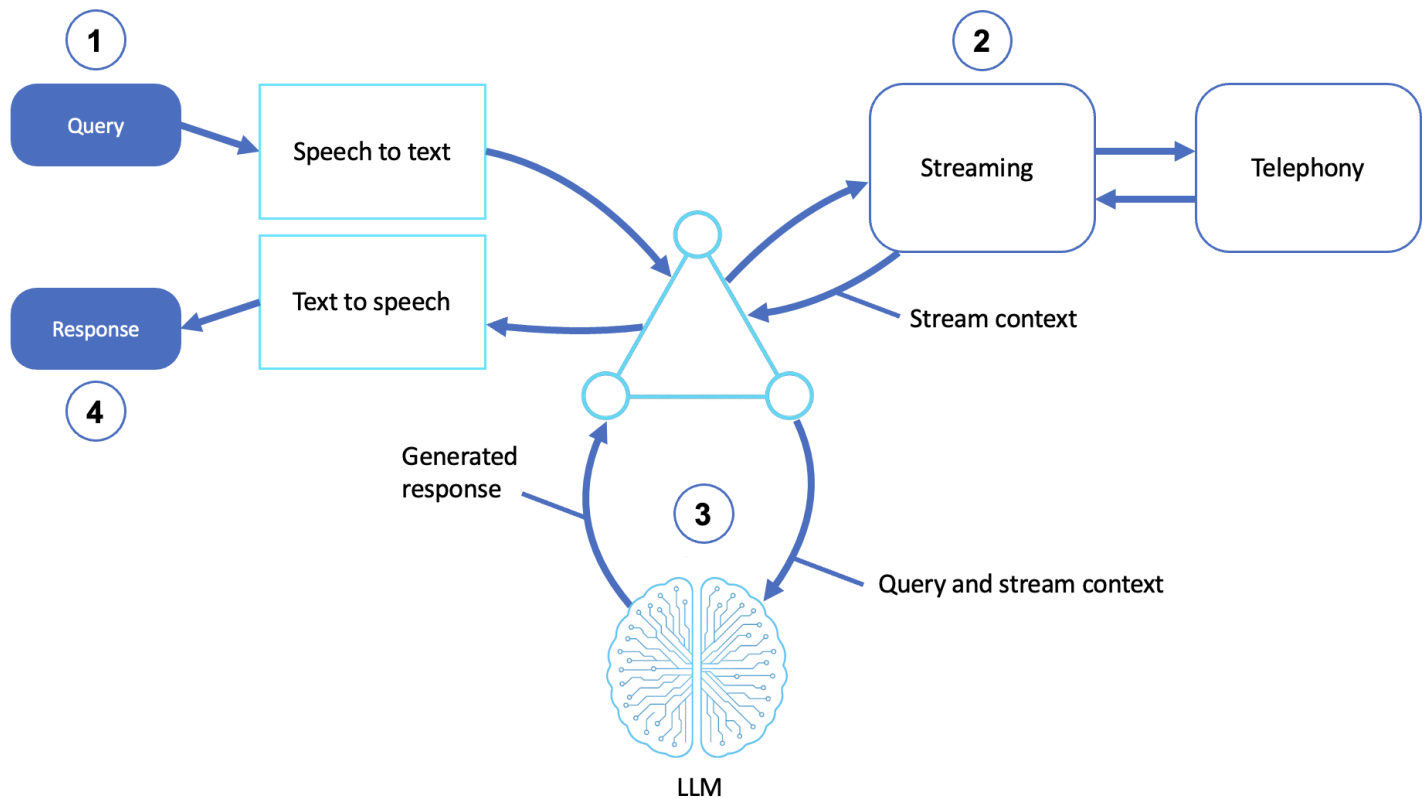
Agentes de fala e voz

Agentes de voz e voz interagem com os usuários por meio do diálogo falado. Esses agentes integram reconhecimento de fala, compreensão de linguagem natural e síntese de fala para permitir a IA conversacional em plataformas telefônicas, móveis, web e incorporadas.

Os agentes de voz são particularmente eficazes em ambientes sem usar as mãos, em tempo real ou orientados pela acessibilidade. Ao combinar interfaces de streaming com LLM-powered raciocínio, elas facilitam interações ricas e dinâmicas que parecem naturais para os usuários.

Arquitetura

Um agente de fala e voz é mostrado no diagrama a seguir:



Description

1. Recebe uma consulta de voz

- O usuário envia uma solicitação para um telefone, microfone ou sistema embarcado.
- Um módulo de fala para texto (STT) converte o áudio em texto.

2. Integra o contexto de streaming e telefonia

- O agente usa uma interface de streaming para gerenciar o áudio I/O em tempo real.
- Se for implantada em um contexto de contact center ou de telecomunicações, a integração telefônica gerencia o roteamento de sessões, a entrada multifrequência de dois tons (DTMF) e o transporte de mídia.

Nota: DTMF se refere aos tons gerados quando você pressiona os botões no teclado do telefone.

No contexto da integração do contexto de streaming e telefonia em agentes de voz, o DTMF é usado como um mecanismo de entrada de sinal durante uma chamada telefônica, especialmente em sistemas de resposta de voz interativa (IVR). As entradas DTMF permitem que o agente:

- Reconheça as seleções do menu (por exemplo, “Pressione 1 para faturar”. Pressione 2 para obter suporte.”)
- Colete entradas numéricas (por exemplo, números de conta, PINs e números de confirmação)
- Acione fluxos de trabalho ou transições de estado em fluxos de chamadas
- Reverta da fala para o tom de toque quando necessário

1. Razões por meio do contexto de fluxo do LLM

- A consulta é enviada ao agente, que a transmite, junto com qualquer metadado da sessão (por exemplo, ID do chamador, contexto anterior), para um LLM.
- O LLM gera uma resposta, possivelmente usando uma estratégia de cadeia de pensamento ou memória de múltiplas voltas se a interação for contínua.

2. Retorna uma resposta de voz

- O agente converte sua resposta em fala usando a conversão de texto em fala (TTS).
- Ele retorna o áudio para o usuário por meio de um canal de voz.

Capacidades

- Real-time compreensão e geração de fala
- Multilíngue I/O com suporte a STT e TTS
- Integração com APIs de telefonia ou streaming
- Reconhecimento da sessão e transferência de memória entre turnos

Casos de uso comuns

- Sistemas IVR conversacionais
- Recepcionistas virtuais e agendadores de consultas
- Voice-driven agentes de helpdesk
- Assistentes de voz vestíveis
- Interfaces de voz para casas inteligentes e ferramentas de acessibilidade

Orientação para implementação

Você pode criar esse padrão usando as seguintes ferramentas e Serviços da AWS:

- Amazon Lex V2 ou Amazon Transcribe para STT
- Amazon Polly para TTS
- Amazon Chime SDK, Amazon Connect Customer ou Amazon Interactive Video Service (Amazon IVS) para streaming e telefonia
- Amazon Bedrock para raciocinar com Anthropic, AI21 ou outros modelos de fundação
- AWS Lambda para conectar STT, LLM, TTS e contexto de sessão

(Opcional) Aprimoramentos adicionais podem incluir o seguinte:

- Amazon Kendra OpenSearch ou para RAG com reconhecimento de contexto
- Amazon DynamoDB para memória de sessão
- Amazon CloudWatch Logs e AWS X-Ray para rastreabilidade

Resumo

Os agentes de fala e voz são sistemas inteligentes que interagem por meio de conversas naturais. Ao integrar interfaces de fala com o raciocínio LLM e a infraestrutura de streaming em tempo real, os agentes de voz permitem interações perfeitas, acessíveis e escaláveis.

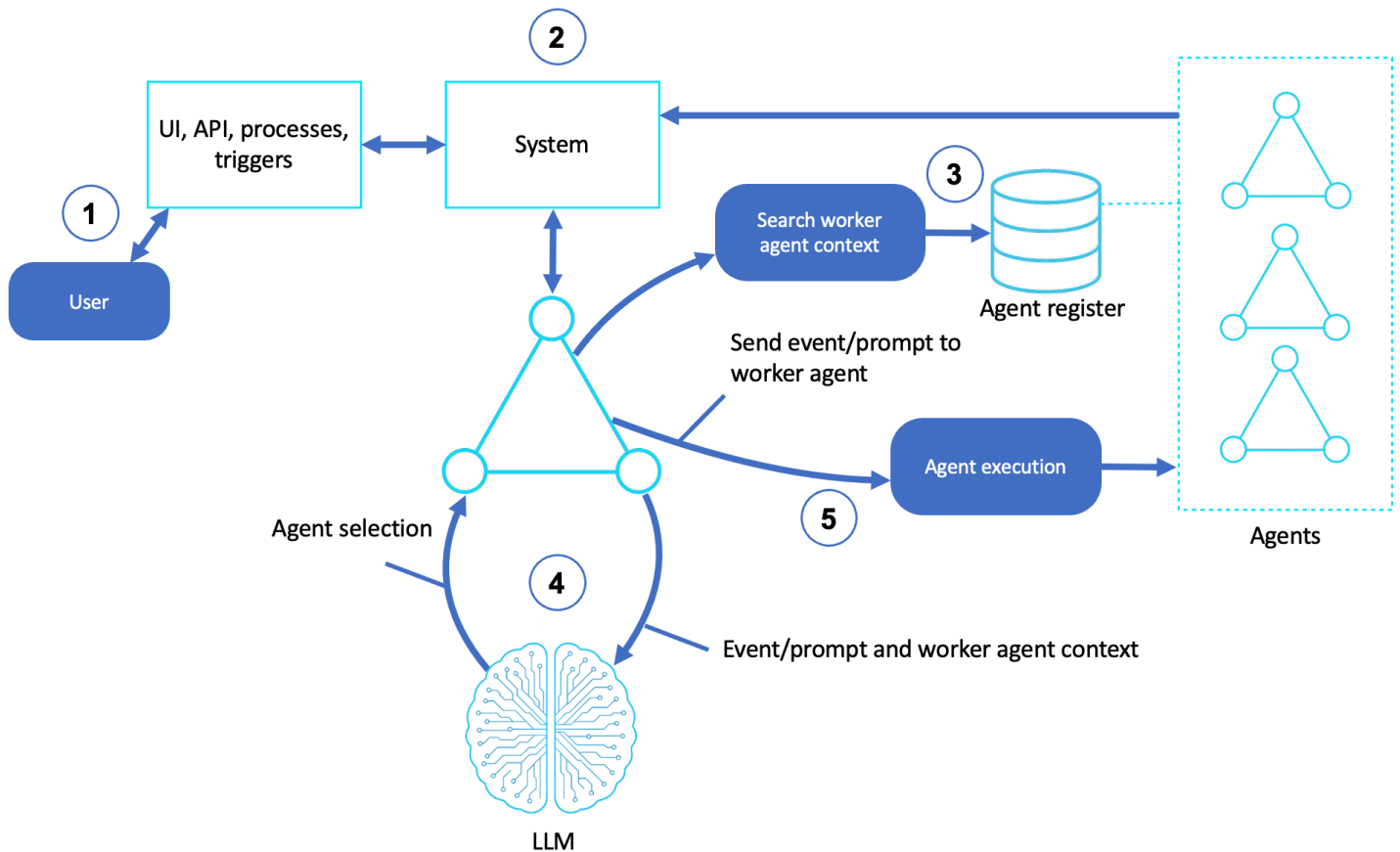
Agentes de orquestração de fluxo de trabalho

Agentes de orquestração de fluxo de trabalho gerenciam e coordenam tarefas, processos e serviços de várias etapas em sistemas distribuídos. Em vez de raciocinar e agir isoladamente, esses agentes delegam trabalho a subagentes ou outros sistemas, mantêm o contexto de execução e se adaptam com base em resultados intermediários.

Esses agentes são uma parte fundamental dos fluxos de automação. Eles são particularmente úteis ao lidar com tarefas de longa execução, composições de vários agentes e integrações entre domínios em que vários agentes e ferramentas devem ser chamados em sequência ou condicionalmente.

Arquitetura

Um agente de orquestração de fluxo de trabalho é mostrado no diagrama a seguir:



Description

1. Recebe a entrada do usuário

- Um usuário (ou acionador externo) inicia uma tarefa por meio de uma interface de usuário, API ou evento do sistema.

2. Lida com eventos do sistema

- Um componente do sistema recebe a solicitação e emite um evento ou comando que requer orquestração.

3. Recupera o contexto

- O agente do fluxo de trabalho consulta as bases de conhecimento e os registros de agentes para encontrar o agente de trabalho certo para a tarefa com base nos metadados, no domínio e na taxa de sucesso anterior.

4. Seleciona um agente LLM

- Um LLM ajuda a selecionar o melhor agente ou plano de fluxo de trabalho analisando a descrição da tarefa e as opções disponíveis.
- Também pode formular solicitações específicas de tarefas para enviar a um agente selecionado.

5. Delega e executa

- O agente de trabalho escolhido recebe o evento ou o prompt e começa a executar os comandos.
- Ele pode rastrear o estado de execução, tentar novamente em caso de falha e passar resultados intermediários para o próximo agente na sequência.

Capacidades

- Composição do agente (por exemplo, supervisores, agentes colaboradores e ferramentas)
- Event-driven ou execução programada
- Rastreamento de memória e estado ao longo do tempo
- Orquestração hierárquica ou paralela de tarefas (síncrona em comparação com fluxos de trabalho assíncronos)
- Seleção e encadeamento dinâmicos de agentes

Casos de uso comuns

- Automação em várias etapas (por exemplo, ingestão de dados e geração de relatórios)
- Roteamento e escalonamento do atendimento ao cliente (por exemplo, agente como coordenador)
- Agentes de IA coordenam com humanos e bots dentro do mesmo circuito
- Automatiza os processos corporativos usando LLM-powered a lógica
- Os sistemas híbridos combinam agentes de IA e ferramentas tradicionais de orquestração

Orientação para implementação

Você pode criar esse padrão usando as seguintes ferramentas e Serviços da AWS:

- Amazon Bedrock para raciocínio e seleção de agentes
- AWS Step Functions ou Amazon EventBridge para composição do fluxo de trabalho

- AWS Lambda como unidades de execução ou executores de tarefas
- Amazon DynamoDB, Amazon Simple Storage Service (Amazon S3) ou Amazon RDS para monitorar estados e resultados
- AWS AppFabric ou Amazon AppFlow para coordenação entre sistemas
- (Opcional) Use o agente de SageMaker execução da Amazon para hospedar agentes de trabalho específicos do domínio

Resumo

Agentes de fluxo de trabalho coordenam, adaptam e alinham metas em ambientes multiagentes. Isso significa que os agentes de IA podem colaborar, se adaptar às condições de tempo de execução e fornecer resultados complexos por meio de fluxos de trabalho modulares e explicáveis.

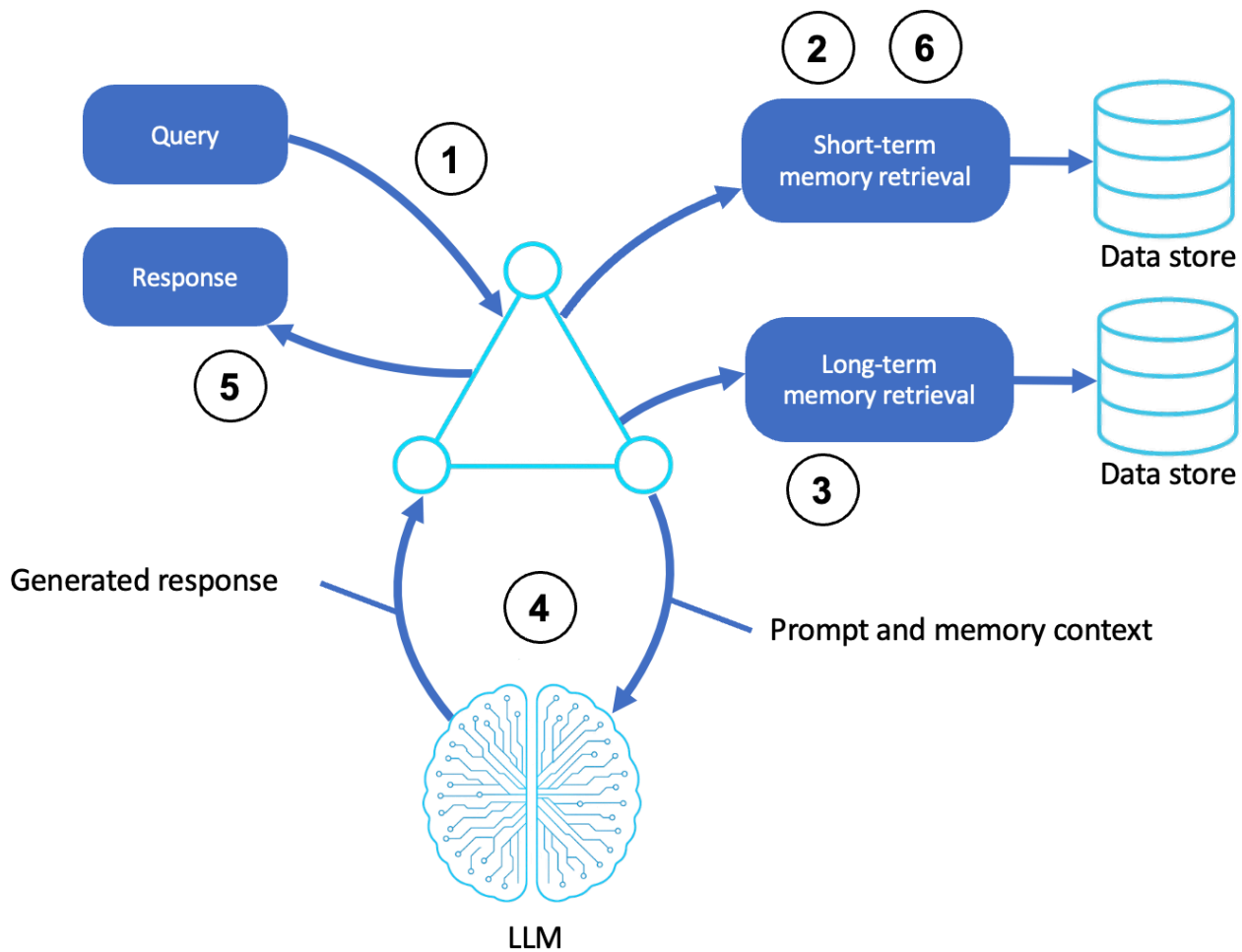
Memory-augmented agentes

Memory-augmented os agentes são aprimorados com a capacidade de armazenar, recuperar e raciocinar usando memória de curto e longo prazo. Isso permite que eles mantenham o contexto em várias tarefas, sessões e interações, o que produz respostas mais coerentes, personalizadas e estratégicas.

Diferentemente dos agentes apátridas, os agentes com memória aumentada se adaptam referenciando dados históricos, aprendem com resultados anteriores e tomam decisões alinhadas às metas, preferências e ambiente do usuário.

Arquitetura

Um agente com memória aumentada é mostrado no diagrama a seguir:



Description

1. Recebe informações ou eventos

- O agente recebe uma consulta do usuário ou um evento do sistema. Isso pode ser um texto, um gatilho de API ou uma mudança ambiental.

2. Recupera a memória de curto prazo

- O agente recupera o histórico recente da conversa, o contexto da tarefa ou o estado do sistema que é relevante para a sessão ou o fluxo de trabalho.

3. Recupera a memória de longo prazo

- O agente consulta a memória de longo prazo (por exemplo, bancos de dados vetoriais e armazenamentos de valores-chave) para obter informações históricas, como as seguintes:
 - Preferências do usuário
 - Decisões e resultados anteriores

- Conceitos, resumos ou experiências aprendidas

4. Razões por meio do LLM

- O contexto da memória é incorporado ao prompt do LLM, permitindo que o agente raciocine com base nas entradas atuais e no conhecimento prévio.

5. Gera saídas

- O agente produz uma resposta, um plano ou uma ação contextualmente consciente que é personalizada de acordo com o histórico da tarefa e as entradas do usuário.

6. Atualiza a memória

- Novas informações, como metas atualizadas, sinais de sucesso e fracasso e respostas estruturadas, são armazenadas para tarefas futuras.

Capacidades

- Continuidade da sessão em conversas ou eventos
- Persistência de metas ao longo do tempo
- Consciência contextual baseada em um estado em evolução
- Adaptabilidade informada por sucessos e fracassos anteriores
- Personalização alinhada às preferências e ao histórico do usuário

Casos de uso comuns

- Co-pilotos conversacionais que lembram as preferências do usuário
- Agentes de codificação que rastreiam alterações na base de código
- Agentes de fluxo de trabalho que se adaptam de acordo com o histórico de tarefas
- Gêmeos digitais que evoluem a partir do conhecimento do sistema
- Agentes de pesquisa que evitam recuperações redundantes

Implementação de agentes com memória aumentada

Use as seguintes ferramentas e Serviços da AWS para agentes com memória aumentada:

Camada de memória	AWS service (Serviço da AWS)	Finalidade
Short-term	Contexto do Amazon DynamoDB, Redis, Amazon Bedrock	Recuperação rápida de estados de interação recentes
Long-term (estruturado)	Amazon Aurora, Amazon DynamoDB, Amazon Neptune	Fatos, relacionamentos e registros
Long-term (semântico)	OpenSearch, PostgreSQL, Pinha	Embedding-based recuperação (ou seja, RAG)
Armazenamento	Amazon S3	Armazenamento de transcrições, memórias estruturadas e arquivos
Orquestração	AWS Lambda or AWS Step Functions	Gerenciando o ciclo de vida de injeção e atualização de memória
Reasoning	Amazon Bedrock	Claude antrópico ou Mistral com alertas de memória

Implementando a solicitação com injeção de memória

Para integrar a memória ao raciocínio do agente, use uma combinação de estado estruturado e injeção de contexto com recuperação aumentada:

- Inclua o estado mais recente do agente e o histórico recente de diálogos como entrada estruturada ao criar o prompt para o modelo de linguagem, para que ele possa raciocinar com todo o contexto.
- Use a geração aumentada de recuperação (RAG) para extrair documentos ou fatos relevantes da memória de longo prazo.
- Resuma os planos, o contexto e as interações anteriores para fins de compactação e relevância.
- Injete módulos de memória externa, como armazenamentos vetoriais ou registros estruturados, durante a inferência para orientar a tomada de decisões.

Resumo

Memory-augmented os agentes mantêm a continuidade do pensamento aprendendo com a experiência e lembrando o contexto do usuário. Esses agentes superam a inteligência reativa usando colaboração, personalização e raciocínio estratégico de longo prazo. Em termos de IA agente, a memória permite que os agentes se comportem mais como contrapartes digitais adaptáveis e menos como ferramentas sem estado.

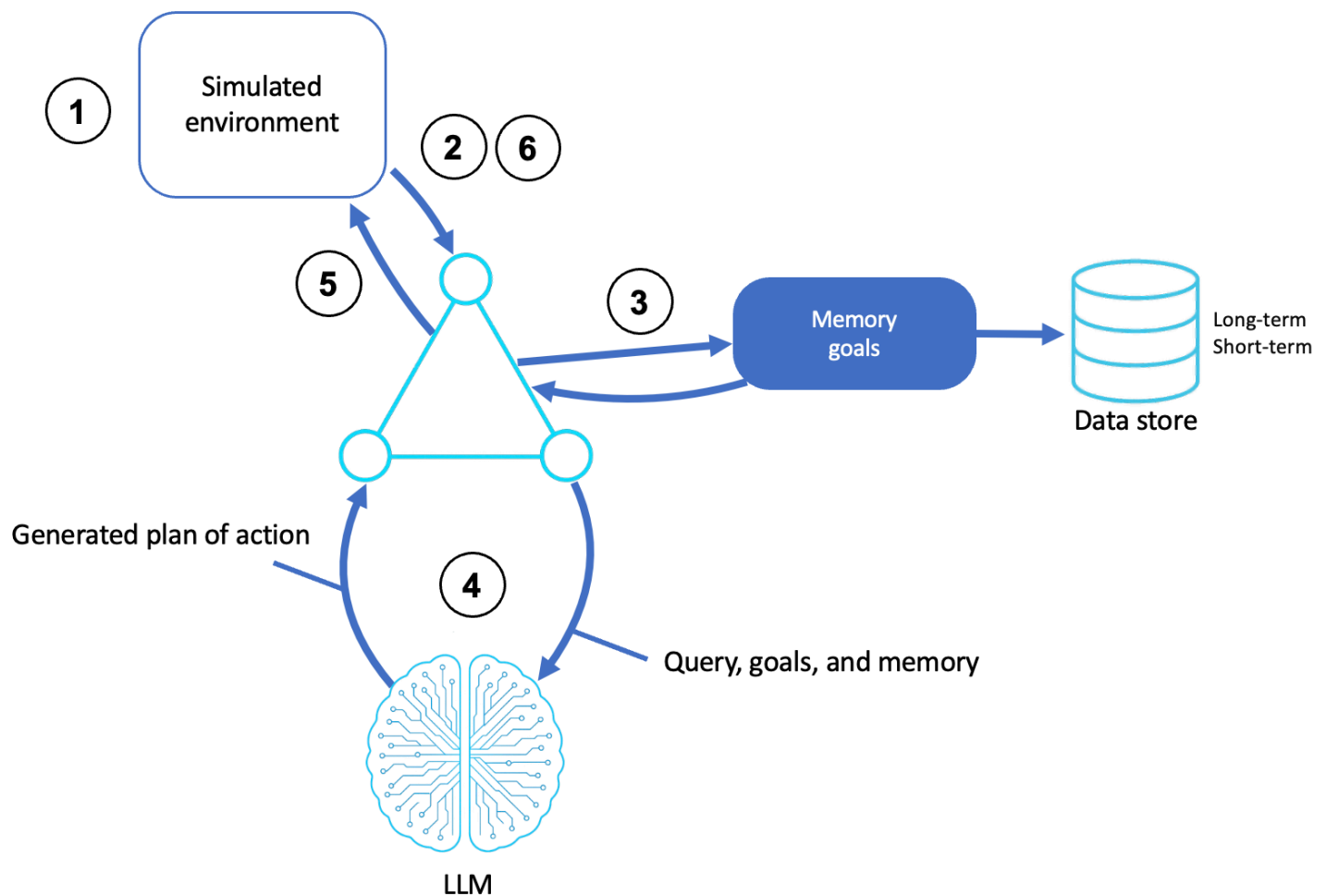
Agentes de simulação e de teste

Agentes de simulação e de teste operam em ambientes virtualizados ou controlados, onde raciocinam, agem e aprendem. Esses agentes simulam comportamento, modelam resultados e treinam estratégias em ambientes repetíveis antes de aplicá-las a ambientes do mundo real.

Esse padrão é útil para desenvolvimento iterativo, aprendizado por reforço (RL), avaliação autônoma de tomada de decisão e testes de comportamento emergentes. Os agentes de simulação geralmente operam em circuitos fechados, recebendo feedback de seu ambiente e ajustando seu comportamento de acordo, tornando-os essenciais para tarefas que envolvem raciocínio espacial, controle em tempo real ou dinâmica complexa do sistema.

Arquitetura

O diagrama a seguir mostra uma simulação ou um agente de teste:



Description

1. Inicia um ambiente

- O agente inicia um ambiente simulado (por exemplo, um mundo 3D, mecanismo de física, sandbox de CLI ou fluxo de dados sintético).
- O agente é carregado no ambiente com uma tarefa, meta ou política inicial.

2. Percebe o agente

- O agente percebe o estado atual por meio da telemetria de simulação (por exemplo, emulação de sensor, câmera virtual e registros estruturados).

3. Recupera objetivo e memória

- O agente recupera o objetivo atribuído, as instruções do cenário ou a meta contextual.
- Ele também pode recuperar a memória anterior, incluindo o seguinte:
 - Long-term estratégias ou políticas

- Mapas ambientais ou restrições conhecidas
- Sucessos ou fracassos anteriores de simulações semelhantes

4. Razões e planos

- Um LLM interpreta o estado simulado, os objetivos da tarefa e o conhecimento aprendido.
- Ele gera um plano de ação ou comando de controle.

5. Executa ações simuladas

- O agente executa o plano, modifica o estado, navega pelo espaço ou interage com entidades virtuais.

6. Aprende

- O agente avalia os resultados da ação
- Dependendo da configuração do agente, ele pode fazer o seguinte:
 - Execute RL
 - Registre os resultados para futuros ajustes
 - Adapte estratégias em tempo real

Capacidades

- Opera em ambientes sintéticos ou virtuais
- Oferece suporte ao aprendizado por tentativa e erro, ao refinamento de políticas e à modelagem do sistema
- Low-risk testes de comportamento, tratamento de falhas e casos extremos
- Permite a análise do comportamento de agentes emergentes em configurações de vários agentes
- Suporta controle de circuito fechado e exploração humana em circuito

Casos de uso comuns

- Aprendizado por reforço para robótica, drones e jogos
- Treinamento de veículos autônomos em estradas virtuais
- UIs ou CLIs simuladas para cenários DevOps de teste
- Experimentos de comportamento emergente em simulações sociais
- Validação de segurança da lógica de decisão antes da produção

Orientação para implementação

Você pode criar um agente de simulação e teste usando as seguintes ferramentas e: Serviços da AWS

Componente	AWS service (Serviço da AWS)	Finalidade
Environment	Amazon ECS, Amazon EC2 ou um simulador personalizado no SageMaker Amazon Studio Lab	Execute mundos virtuais (Gazebo, Unity, Unreal) ou CLIs de sandbox
Lógica do agente	Amazon Bedrock SageMaker, Amazon ou AWS Lambda	LLM-based planejadores ou agentes de RL
Ciclo de feedback	SageMaker Aprendizado por reforço da Amazon CloudWatch, Amazon ou registros personalizados	Rastreamento de recompensas, pontuação de resultados e registro de comportamento
Memória e repetição	Amazon S3, Amazon DynamoDB ou Amazon RDS	Estado persistente, histórico de episódios ou dados do cenário
Visualização	CloudWatch Painéis da Amazon ou cadernos da Amazon SageMaker	Observe as mudanças nas políticas, os resultados e as métricas de treinamento

A seguir estão os aplicativos adicionais:

- [AWS SimSpace Weaver](#) para simulações espaciais em grande escala
- [AWS IoT Core](#) para testar dispositivos de sombra
- [Amazon SageMaker Experiments](#) para avaliação e benchmarking de agentes

Resumo

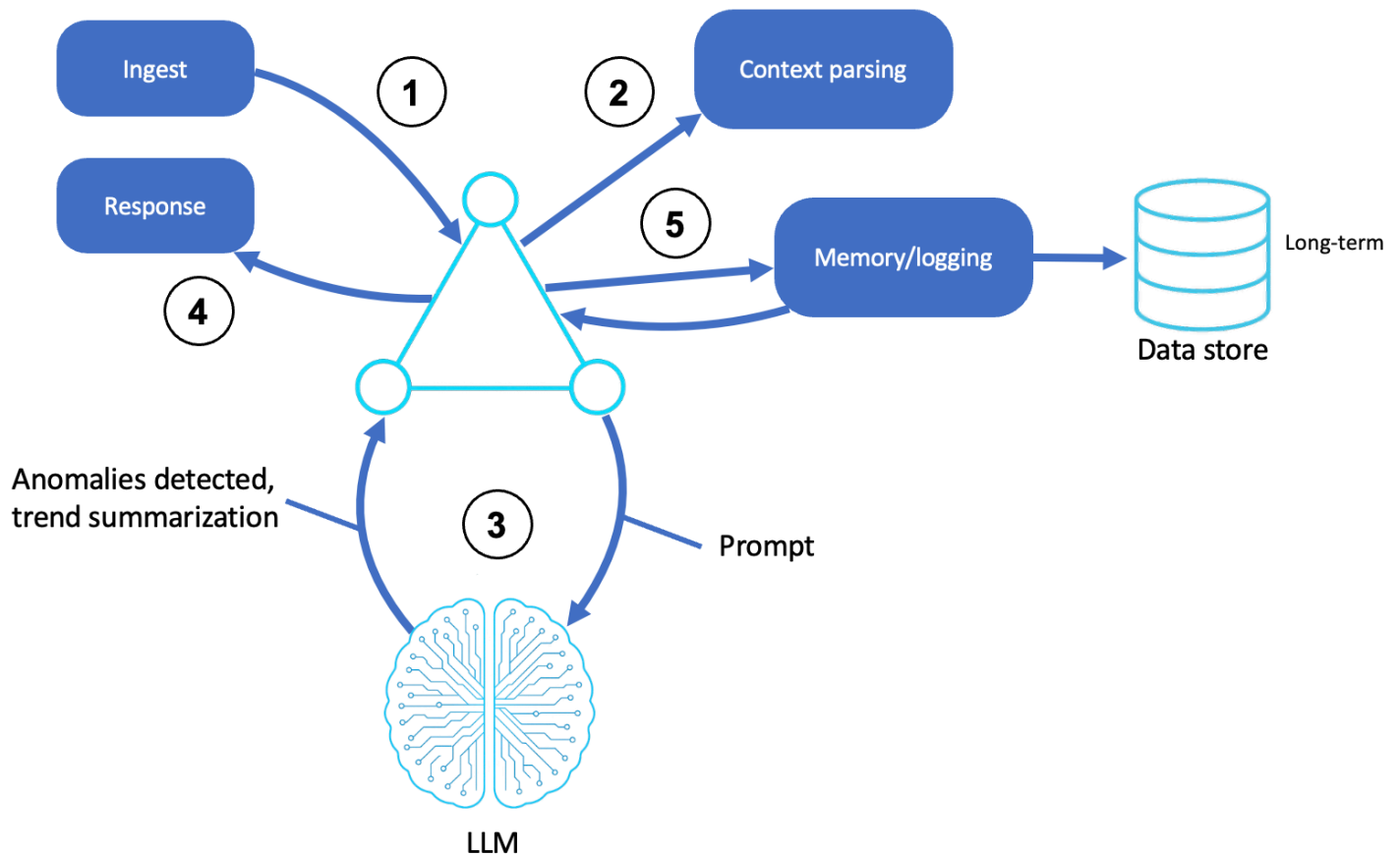
Os agentes de simulação e de teste são para exploração estruturada antes de serem implantados em sistemas de produção. Use esses agentes para treinar políticas de navegação autônoma, testar processos de negócios em ambientes sintéticos e avaliar padrões de coordenação em enxames.

Agentes observadores e de monitoramento

Agentes observadores e de monitoramento observam passivamente sistemas, ambientes e interações para detectar padrões, gerar insights e acionar ações. Como observadores inteligentes, eles aprimoram alertas, diagnósticos e auditorias sem iniciar diretamente o comportamento.

Esses agentes se destacam onde o monitoramento tradicional carece de adaptabilidade ou raciocínio, especialmente para AI-in-the-loop monitoramento, detecção de anomalias, supervisão de conformidade e inteligência de segurança. Agentes observadores são ouvintes de eventos que monitoram continuamente a telemetria do sistema e as interações do usuário. O agente depende da percepção, interpretação e escalonamento ou relatório condicional.

Arquitetura



Description

1. Telemetria de ingestão

- O agente recebe informações de uma ou mais fontes do sistema, como as seguintes:
 - Registros (aplicativo, infraestrutura, segurança)
 - Métricas (desempenho, latência, uso)
 - Eventos (chamadas de API, ações do usuário, dados do sensor)

2. Analise o contexto

- A entrada bruta é analisada, estruturada e enriquecida com metadados, como registro de data e hora, identidade do ator, estado do sistema e ID de rastreamento.

3. Razões para usar o LLM

- O agente usa um LLM ou módulo lógico para interpretar entradas analisadas identificando anomalias, resumindo tendências e correlacionando traços distribuídos ou janelas de tempo.

4. Classifique ou alerte

- O agente determina se o comportamento observado justifica o seguinte:
 - Um alerta ou escalonamento
 - Uma atualização de relatório ou painel
 - Um gatilho de resposta (por exemplo, remediação automática e aplicação de políticas)

5. Memória de log ou loops de feedback

- O agente armazena eventos e decisões para aprendizado de longo prazo, auditorias ou referência futura para outros agentes.

Capacidades

- Passivo e não invasivo (o agente não age diretamente)
- Altamente escalável e assíncrono
- AI-driven correlação entre sinais ruidosos ou distribuídos
- Oferece suporte a auditoria, conformidade e visão em tempo real
- Pode alimentar agentes posteriores ou fluxos de trabalho humanos

Casos de uso comuns

- AI-augmented observabilidade para microsserviços e APIs
- Monitoramento de desvios de modelos, violações de políticas ou comportamento fora da banda
- Análise da atividade do cliente ou resumos de interações
- Agentes de revisão de código que monitoram confirmações ou implantações
- Monitoramento de registros de segurança ou conformidade usando o raciocínio LLM

Orientação para implementação

Você pode criar um observador e um agente de monitoramento usando as seguintes ferramentas e Serviços da AWS:

Componente	AWS service (Serviço da AWS)	Finalidade
------------	------------------------------	------------

Ingestão de eventos	Amazon EventBridge, Amazon CloudWatch Logs, Amazon Kinesis, Amazon S3	Ingira telemetria estruturada e não estruturada
Pré-processamento	AWS Lambda, AWS Glue, AWS Step Functions	Transforme dados brutos em solicitações estruturadas
Motor de raciocínio	Amazon Bedrock, Amazônia SageMaker, AWS Lambda	Analise eventos, classifique comportamentos, gere insights
Armazenamento e memória	Amazon S3, Amazon DynamoDB, OpenSearch	Observações, resumos e resultados persistentes
Alertas e escalonamento	Amazon SNS, AWS AppFabric Amazon EventBridge	Acione sistemas ou agentes posteriores

A seguir estão os aplicativos adicionais:

- [AWS Security Hub CSPM](#) para monitoramento de registros de segurança
- [Amazon Quick](#) para visualizar as saídas do agente

Resumo

Agentes observadores e de monitoramento rastreiam sistemas e comportamentos em tempo real. Eles detectam anomalias, auditam a segurança e coletam inteligência operacional identificando padrões que humanos ou regras possam ignorar. Esse recurso ajuda a criar sistemas que podem se adaptar às mudanças nas condições e tomar decisões com base em uma análise abrangente de dados.

Multi-agent colaboração

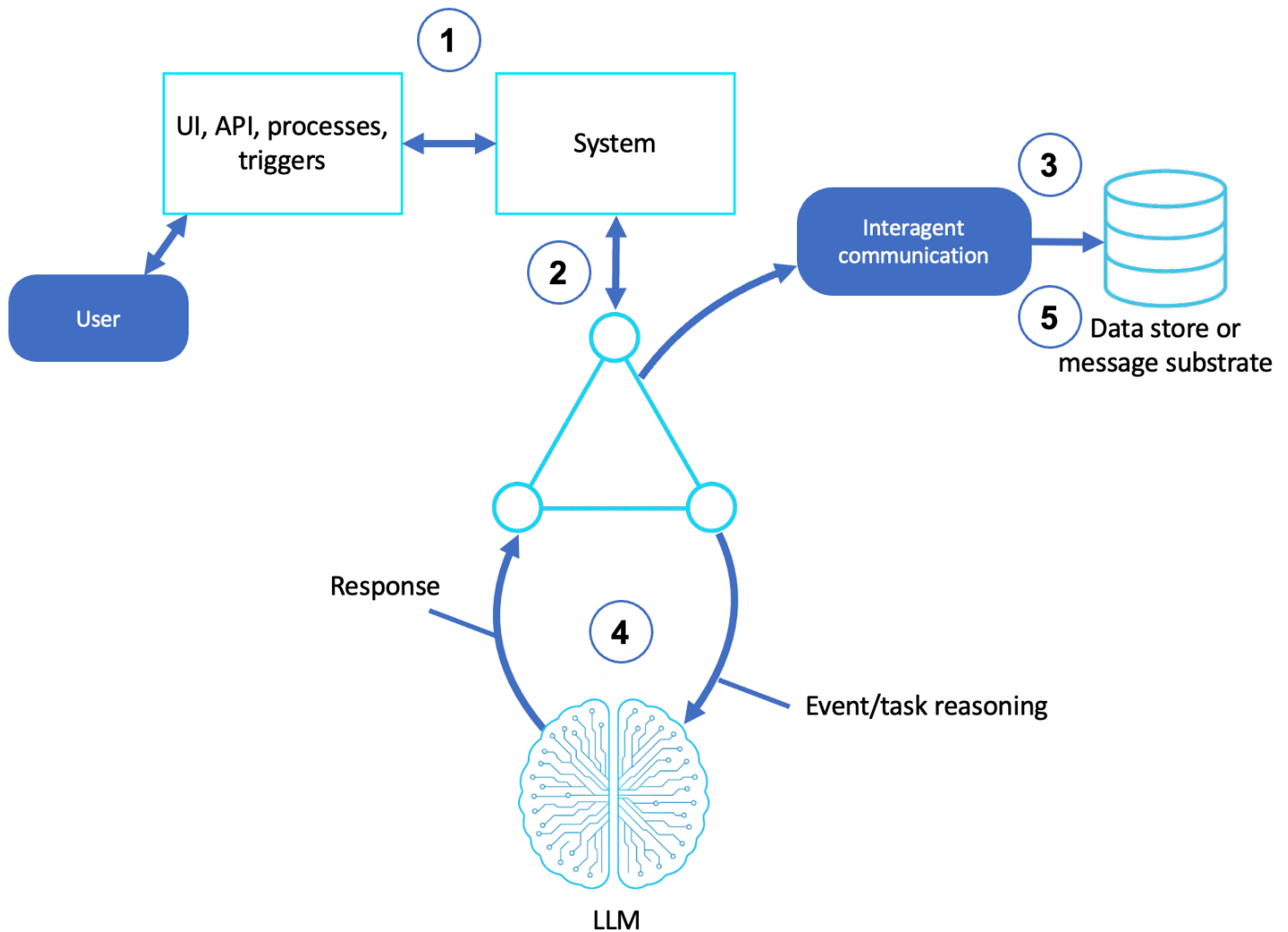
Multi-agent colaboração se refere a um padrão no qual vários agentes autônomos, cada um com uma função, especialização ou objetivo distinto, negociam para resolver tarefas complexas. Esses agentes podem operar de forma independente ou com outros agentes compartilhando informações, dividindo responsabilidades e raciocinando coletivamente em direção a uma meta.

Esse padrão difere dos agentes de fluxo de trabalho, que coordenam e delegam tarefas centralmente aos agentes subordinados em um fluxo estruturado. Em contraste, a colaboração multiagente enfatiza a coordenação ponto a ponto ou emergente, permitindo a adaptabilidade, o paralelismo e a divisão da cognição. A tabela a seguir compara a colaboração de vários agentes com agentes de fluxo de trabalho:

Recurso	agentes de fluxo de trabalho	Finalidade
Controle	Coordenador centralizado	Pares descentralizados, distribuídos ou baseados em funções
Interação	Um agente delega e monitora a execução	Vários agentes negociam, compartilham e se adaptam
Projeto	Sequência predefinida de tarefas	Distribuição de tarefas emergente e flexível
Coordenação	Orquestração processual	Interações cooperativas ou competitivas
Casos de uso	Automação de processos corporativos	Raciocínio complexo, exploração e estratégias emergentes

Arquitetura

O diagrama a seguir mostra a colaboração entre vários agentes:



Description

1. Inicia uma tarefa

- Um usuário ou sistema emite uma meta ou problema de alto nível.
- Um agente “gerente” ou contexto iniciador define o objetivo.

2. Atribui ou descobre funções

- Os agentes se autoatribuem (lógica simbólica ou raciocínio) ou são delegados (mediadores de eventos) a outras funções, como planejador, pesquisador, executor, crítico ou explicador.

3. Comunica-se com outros agentes

- Os agentes se comunicam por meio de memória compartilhada, filas de mensagens ou encadeamento de solicitações.
- Eles podem debater, questionar ou propor subtarefas uns aos outros.

4. Usa raciocínio especializado
- Cada agente usa seu próprio modelo ou lógica de domínio para resolver sua parte do problema.
 - Os agentes podem usar LLMs com prompts e memória específicos da função.
5. Coordena saídas ou metas
- Os agentes sintetizam as contribuições em uma resposta final, plano ou ação.
 - (Opcional) Um agente supervisor pode validar ou resumir a saída sintetizada.

Capacidades

- Peer-level agentes com funções ou habilidades especializadas
- Comportamento emergente por meio de comunicação ou negociação
- Processamento paralelo de problemas complexos ou multifacetados
- Suporta deliberação, autocorreção e iteração reflexiva
- Modele a dinâmica social, a colaboração científica ou as funções da equipe empresarial

Casos de uso comuns

- Equipes de pesquisa autônomas (agente de busca, resumido e validador)
- Desenvolvimento de software (planejador, codificador e testador)
- Modelagem de cenários de negócios (finanças, políticas e conformidade)
- Negociação, licitação ou raciocínio multipartidário
- Tarefas multimodais (imagem, texto e lógica)

Orientação para implementação

Você pode criar um sistema multiagente usando as seguintes ferramentas e Serviços da AWS:

Componente	AWS service (Serviço da AWS)	Finalidade
Hospedagem de agentes	Amazon Bedrock, Amazônia SageMaker, AWS Lambda	Hospede LLM-driven agentes individuais

Camada de comunicação	Amazon SQS, Amazon, EventBridge AWS AppFabric	Mensagens e coordenação entre agentes
Memória compartilhada	Amazon DynamoDB, Amazon S3 ou OpenSearch	Multi-agent sistema de memória ou quadro-negro
Camada de orquestração	AWS Step Functions, AWS Lambda oleodutos	Lógica de início, tempo limite, retorno e repetição
Identificação do agente	Agentes do Amazon Bedrock (definidos por função) e API converse do AWS AppConfig Amazon Bedrock (agentes fora do Amazon Bedrock)	Role-based invocação de ferramenta ou agente e aplicação de limites
Interação emergente	EventBridge Pipelines da Amazon ou registros de agentes	Habilite o roteamento ou escalonamento dinâmico de tarefas

Resumo

Multi-agent a colaboração distribui tarefas de solução de problemas entre agentes modulares e orientados por funções. Diferentemente da orquestração do fluxo de trabalho, os padrões de colaboração usam inteligência, resiliência e escalabilidade emergentes que refletem a forma como os humanos resolvem problemas. É especialmente valioso para domínios abertos, tarefas criativas, raciocínio multimodal e ambientes que se beneficiam de diversas perspectivas.

Conclusão

Os padrões discutidos anteriormente ilustram abordagens fundamentais para implementações reais de IA agente. Do raciocínio básico à inteligência aumentada pela memória, cada padrão é configurado exclusivamente para percepção, cognição e ação com base na autonomia, assincronia e agência.

Esses padrões compartilham vocabulários e planos técnicos para criar sistemas inteligentes e direcionados a objetivos. Se um padrão está incorporado em uma interface de usuário, orquestrado por meio de serviços em nuvem ou coordenado entre equipes de agentes, cada padrão é adaptável e modular.

Takeaways

- Os padrões do agente podem ser compostos — a maioria dos agentes do mundo real combina dois ou mais padrões (por exemplo, um agente de voz com raciocínio e memória baseados em ferramentas).
- O design do agente é contextual — escolha padrões com base na superfície de interação, na complexidade da tarefa, na tolerância à latência e nas restrições específicas do domínio.
- AWSa implementação nativa é possível — Com o Amazon Bedrock, o Amazon SageMaker, AWS Lambda AWS Step Functions, e as arquiteturas orientadas a eventos, cada padrão de agente pode ser entregue em grande escala.

Fluxos de trabalho do LLM

Em padrões de agentes, exploramos os padrões comuns de agentes de IA, cada um construído em torno de um conjunto de recursos modulares: percepção, ação, aprendizado e cognição. No centro do módulo cognitivo em muitos padrões de agentes está um grande modelo de linguagem (LLM) capaz de raciocinar, planejar e tomar decisões. No entanto, invocar um LLM sozinho não é suficiente para produzir um comportamento inteligente e direcionado a objetivos.

Para realizar tarefas complexas de forma confiável, os agentes devem incorporar o LLM em um fluxo de trabalho estruturado, onde os recursos do modelo são aumentados com ferramentas, memória, ciclos de planejamento e lógica de coordenação. Esses fluxos de trabalho de LLM permitem que um agente defina metas, encaminhe subtarefas, chame serviços externos, reflita sobre os resultados e se coordene com outros agentes.

Este capítulo apresenta os principais padrões de design para criar módulos LLM-driven cognitivos robustos, extensíveis e inteligentes, organizados em torno de fluxos de trabalho reutilizáveis.

Nesta seção

- [Visão geral da LLM-augmented cognição](#)
- [Fluxo de trabalho para encadeamento imediato](#)
- [Fluxo de trabalho para roteamento](#)
- [Fluxo de trabalho para paralelização](#)
- [Fluxo de trabalho para orquestração](#)
- [Fluxo de trabalho para avaliadores e ciclos de reflexão e refinamento](#)
- [Conclusão](#)

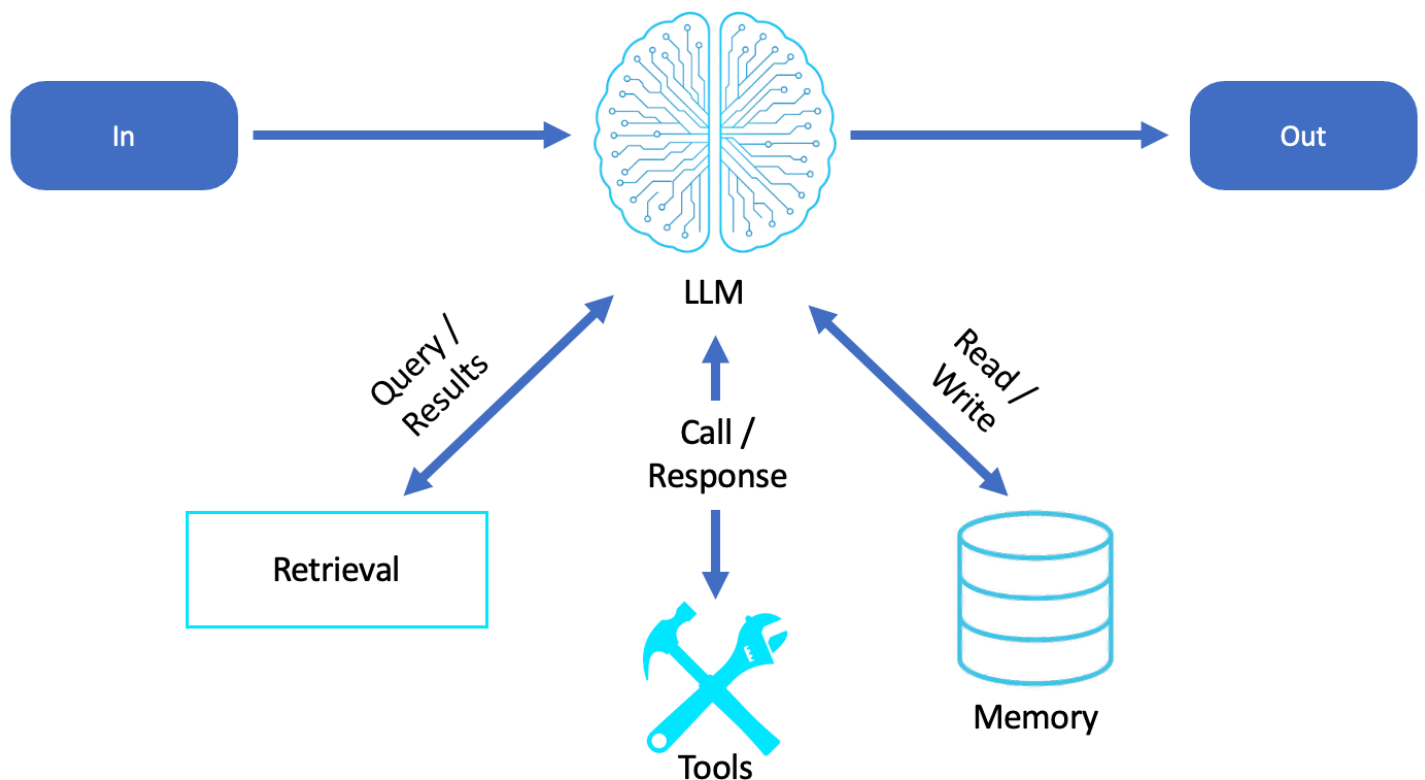
Visão geral da LLM-augmented cognição

Em sua essência, o módulo cognitivo de um agente de software pode ser visto como um LLM envolto em aprimoramentos. O agente pode usar os seguintes elementos básicos para raciocinar de forma eficaz em seu ambiente:

- Solicitação — enquadrando a entrada usando contexto, instruções, exemplos e memória

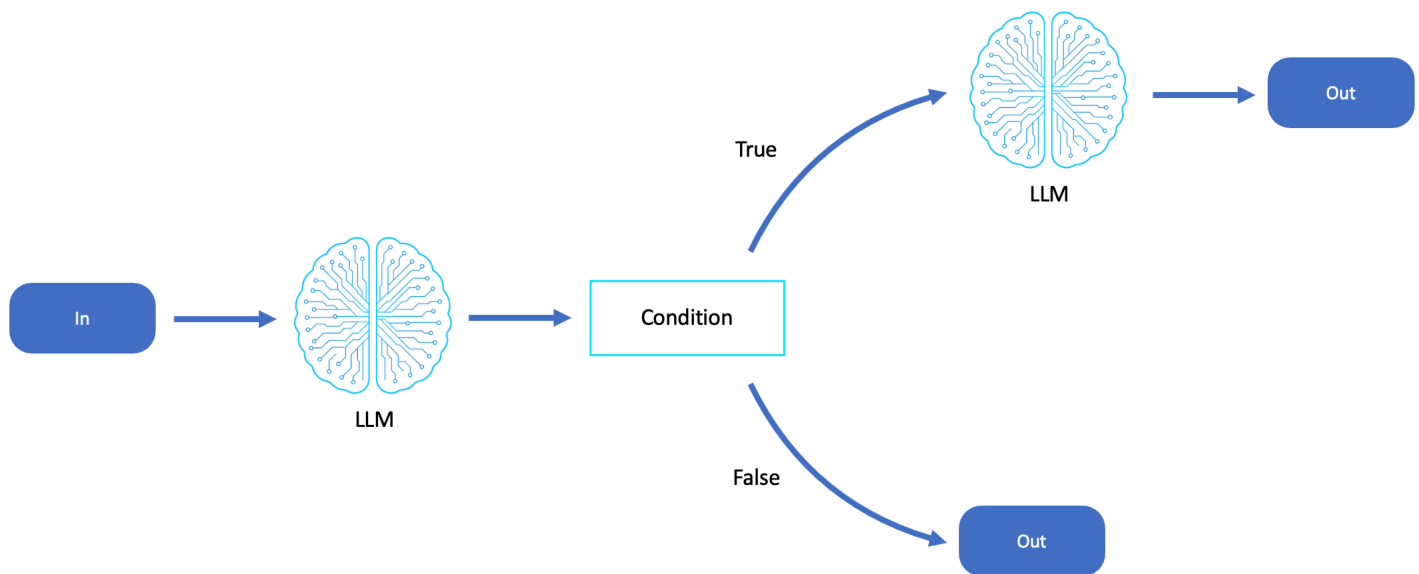
- Recuperação — Fornecendo conhecimento atualizado ou específico do domínio para o prompt do LLM por meio de pesquisa vetorial ou memória semântica, por exemplo, por meio de geração aumentada de recuperação (RAG)
- Uso da ferramenta — Permitindo que o LLM invoque APIs ou chame funções para recuperar ou agir sobre as informações
- Memória — Incorporar um estado persistente ou baseado em sessão ao ciclo de raciocínio, usando bancos de dados estruturados ou resumos contextuais

Esses aumentos são compostos por fluxos de trabalho que definem como o LLM é usado ao longo do tempo e em todas as tarefas, transformando-o de um mecanismo sem estado em um agente de raciocínio dinâmico.



Fluxo de trabalho para encadeamento imediato

O encadeamento imediato decompõe tarefas complexas em uma sequência de etapas, em que cada etapa é uma invocação LLM discreta que processa ou se baseia na saída da anterior.



O fluxo de trabalho de encadeamento imediato é adequado para cenários em que as tarefas podem ser divididas logicamente em etapas de raciocínio sequencial e em que os resultados intermediários informam o próximo estágio. Ele se destaca em fluxos de trabalho que exigem pensamento estruturado, transformação progressiva ou análise em camadas, como revisão de documentos, geração de código, extração de conhecimento e refinamento de conteúdo.

Description

- A complexidade da tarefa excede a janela de contexto ou a profundidade de raciocínio de uma única chamada LLM.
- Os resultados de uma etapa (por exemplo, análise, resumo ou planejamento) se tornam entradas para uma decisão de acompanhamento ou fase de geração.
- Você precisa de transparência e controle em todos os estágios de raciocínio (por exemplo, resultados intermediários auditáveis).
- Você deseja conectar a lógica externa de validação, filtragem ou enriquecimento entre as etapas.
- É ideal para agentes que operam em ciclos de raciocínio no estilo pipeline, como agentes de pesquisa, assistentes editoriais, sistemas de planejamento e copilotos de vários estágios.

Capacidades

- Cadeias lineares ou ramificadas de chamadas LLM

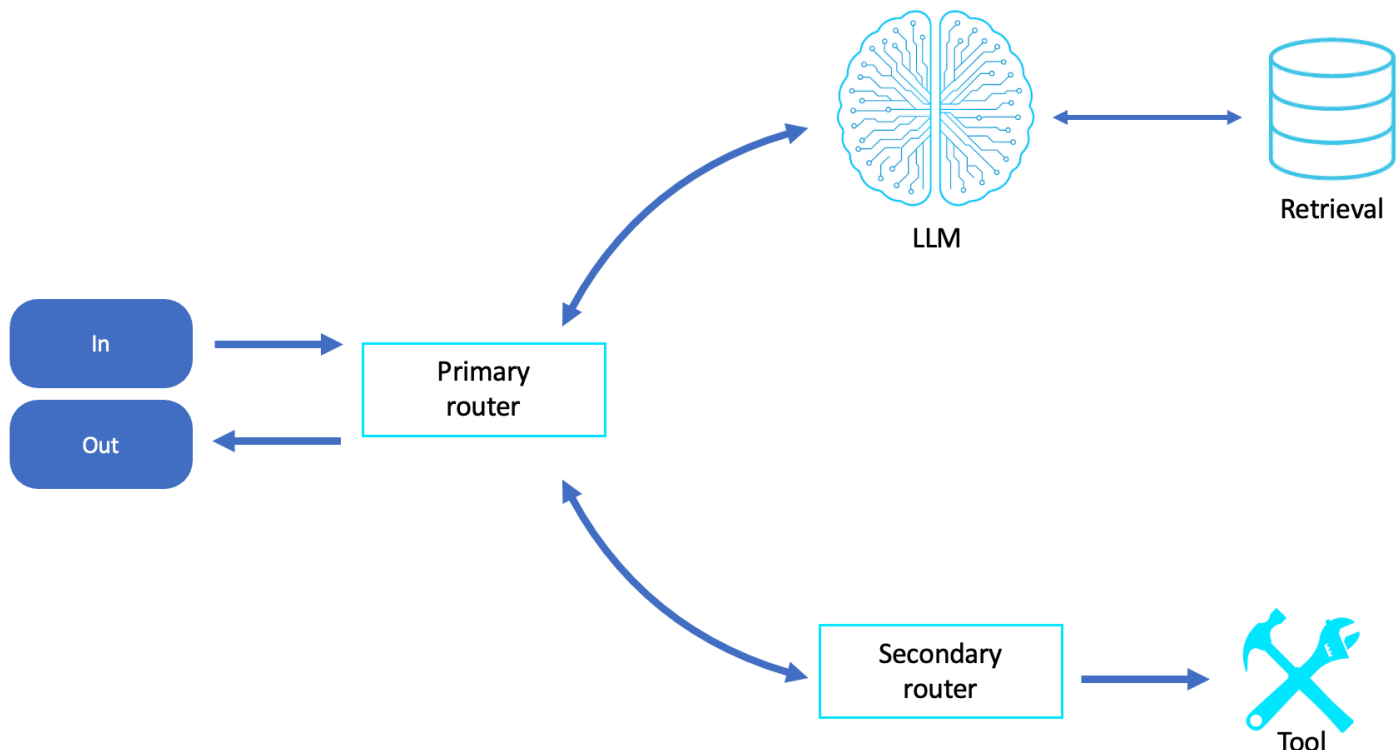
- Resultados intermediários passados como entrada estruturada ou incorporados em solicitações de acompanhamento
- Pode ser orquestrado com AWS Step Functions, ou com corretores AWS Lambda específicos do agente

Casos de uso comuns

- Tarefas de raciocínio em várias etapas (por exemplo, “resumir, reescrever uma crítica”)
- Assistentes de pesquisa sintetizando resultados em camadas (por exemplo, “pesquisar, extrair fatos, responder à pergunta”)
- Pipelines de geração de código (“gerar plano, escrever código de teste, explicar a saída”)

Fluxo de trabalho para roteamento

No padrão de roteamento, um classificador ou agente de roteador usa um LLM para interpretar a intenção ou a categoria de uma consulta e, em seguida, encaminha a entrada para uma tarefa ou agente downstream especializado.



O fluxo de trabalho de roteamento é usado em cenários em que um agente deve classificar rapidamente a intenção de entrada, o tipo de tarefa ou o domínio e, em seguida, delegar a solicitação a um subagente, ferramenta ou fluxo de trabalho especializado. É especialmente útil em agentes capacitados, como aqueles que atuam como assistentes gerais, portas de entrada para funções corporativas ou interfaces de IA voltadas para o usuário que abrangem domínios.

O roteamento é particularmente eficaz quando:

- Triagem de solicitações em uma variedade de tarefas (por exemplo, pesquisa, resumo, agendamento, cálculos).
- As entradas devem ser pré-processadas ou normalizadas antes de entrar em fluxos de trabalho mais especializados.
- Diferentes tipos de entrada (por exemplo, imagens versus texto, consultas estruturadas versus consultas não estruturadas) exigem tratamento personalizado.
- Um agente está atuando como uma central telefônica conversacional, delegando tarefas a agentes especializados ou microsserviços.
- Esse fluxo de trabalho é comum em copilotos específicos de domínio, bots de suporte ao cliente, roteadores de serviços corporativos e agentes multimodais, nos quais o despacho inteligente determina a qualidade e a eficiência do comportamento do agente.

Capacidades

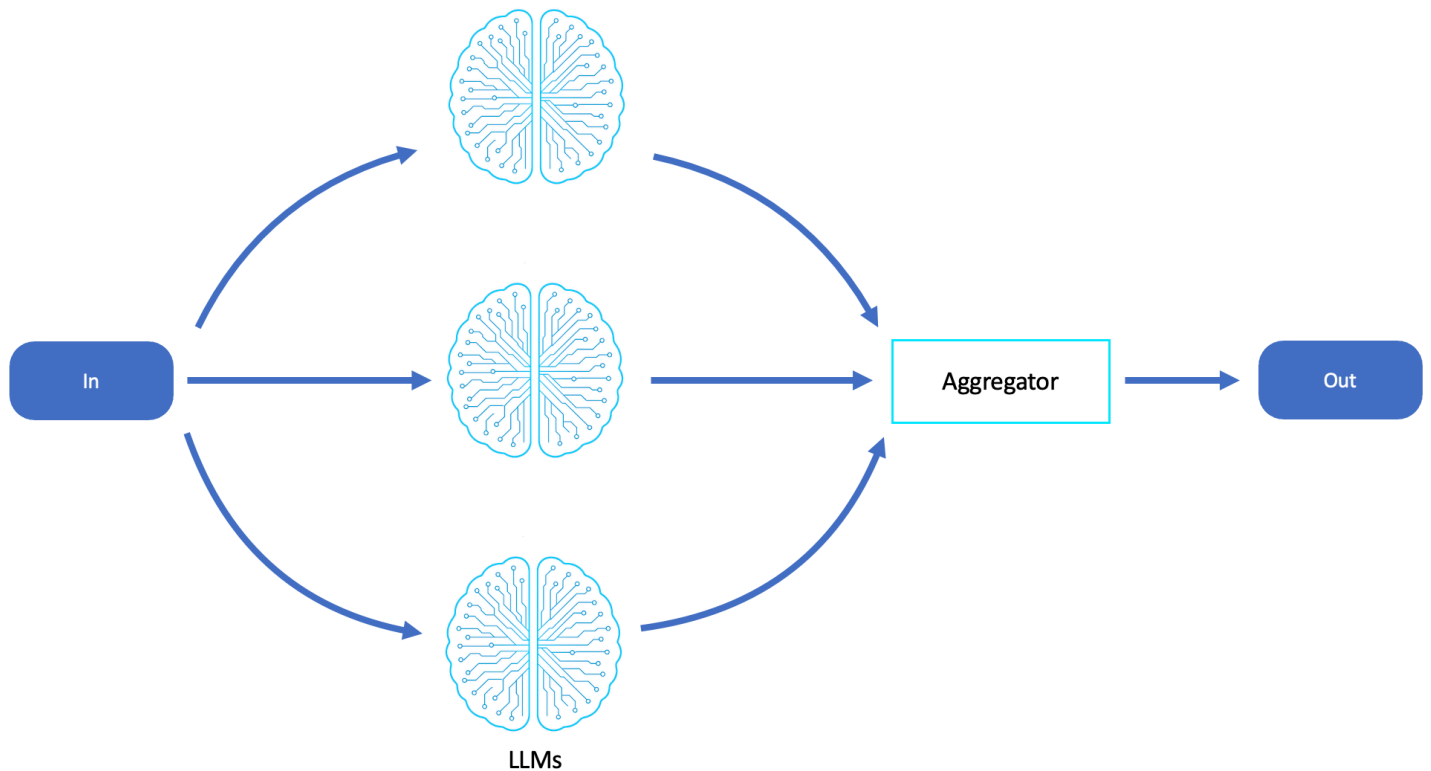
- Um LLM de primeira passagem atua como despachante
- As rotas podem invocar fluxos de trabalho distintos ou até mesmo outros padrões de agentes
- Oferece suporte à expansão modular dos recursos

Casos de uso comuns

- Assistentes de vários domínios (“essa é uma questão legal, médica ou financeira?”)
- Árvores de decisão aprimoradas com o raciocínio LLM
- Seleção dinâmica de ferramentas (por exemplo, pesquisa versus geração de código)

Fluxo de trabalho para paralelização

Esse fluxo de trabalho envolve dividir uma tarefa em subtarefas independentes que podem ser tratadas simultaneamente por várias chamadas ou agentes do LLM. As saídas são então agregadas programaticamente e sintetizadas em um resultado.



O fluxo de trabalho de paralelização é usado quando uma tarefa pode ser dividida em subtarefas independentes e não sequenciais que podem ser processadas simultaneamente, melhorando significativamente a eficiência, a produtividade e a escalabilidade. É especialmente poderoso em espaços problemáticos com muitos dados, orientados por lotes ou multiperspectivas, nos quais o agente deve analisar ou gerar conteúdo em várias entradas.

A paralelização é particularmente eficaz quando:

- As subtarefas não dependem dos resultados intermediários umas das outras, permitindo que elas sejam executadas paralelamente sem coordenação.
- Uma tarefa envolve repetir o mesmo processo de raciocínio em vários itens (por exemplo, resumir vários documentos ou avaliar uma lista de opções).
- Várias hipóteses ou perspectivas são exploradas paralelamente para promover diversidade, criatividade ou robustez.

- Você precisa reduzir a latência para solicitações de alto volume ou alta frequência por meio da execução simultânea do LLM.
- Esse fluxo de trabalho é comumente usado em agentes de processamento de documentos, mecanismos de pesquisa ou comparação, resumos de lotes, brainstormers multiagentes e tarefas escaláveis de classificação ou rotulagem, especialmente quando o raciocínio rápido e paralelo é uma vantagem de desempenho.

Capacidades

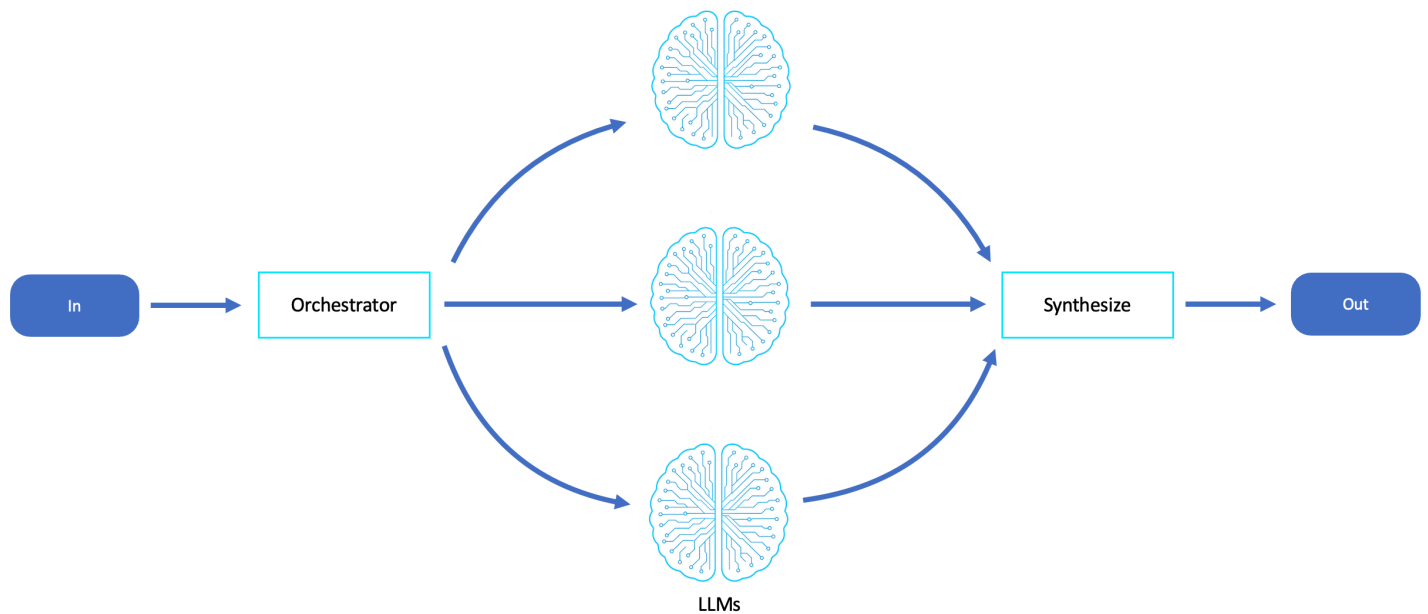
- Execução paralela de tarefas LLM (usando AWS Lambda, AWS Fargate, ou um estado do AWS Step Functions mapa)
- Requer alinhamento, validação ou deduplicação de resultados na fase de síntese
- Well-suited para circuitos de agentes apátridas

Casos de uso comuns

- Analisando vários documentos ou perspectivas em paralelo
- Gerando diversos rascunhos, resumos ou planos
- Acelerando a produtividade em trabalhos em lotes

Fluxo de trabalho para orquestração

Um agente orquestrador central usa um LLM para planejar, decompor e delegar subtarefas a agentes ou modelos de trabalhadores especializados, cada um com uma função ou domínio específico. Isso reflete as estruturas da equipe humana e apoia o comportamento emergente em vários agentes.



O fluxo de trabalho de orquestração é ideal para cenários complexos, hierárquicos ou multidisciplinares, que exigem decomposição estruturada e execução especializada. É particularmente adequado para tarefas que exigem divisão do trabalho, em que diferentes subcomponentes de uma tarefa são mais bem administrados por agentes com capacidades, conhecimentos ou conjuntos de ferramentas distintos.

Esse fluxo de trabalho é particularmente eficaz quando:

- As tarefas podem ser divididas em subtarefas que variam em escopo, tipo ou raciocínio (por exemplo, planejar, pesquisar, implementar e testar).
- Um LLM ou meta-agente deve coordenar outros agentes, monitorar o progresso e sintetizar os resultados.
- Você deseja modularizar as responsabilidades do agente, permitindo escalabilidade, reutilização e ajuste especializado.
- O sistema exige um comportamento baseado em funções, imitando como equipes humanas (por exemplo, gerentes de projeto, desenvolvedores e revisores) operam em colaboração.

A orquestração é ideal para agentes de planejamento de vários turnos, copilotos de desenvolvimento de software, agentes de processos corporativos e executores de projetos autônomos. É especialmente útil ao implementar sistemas multiagentes que exigem divisão centralizada de tarefas, mas lógica de execução distribuída, permitindo extensibilidade e comportamento mais explicável em todas as camadas do agente.

Capacidades

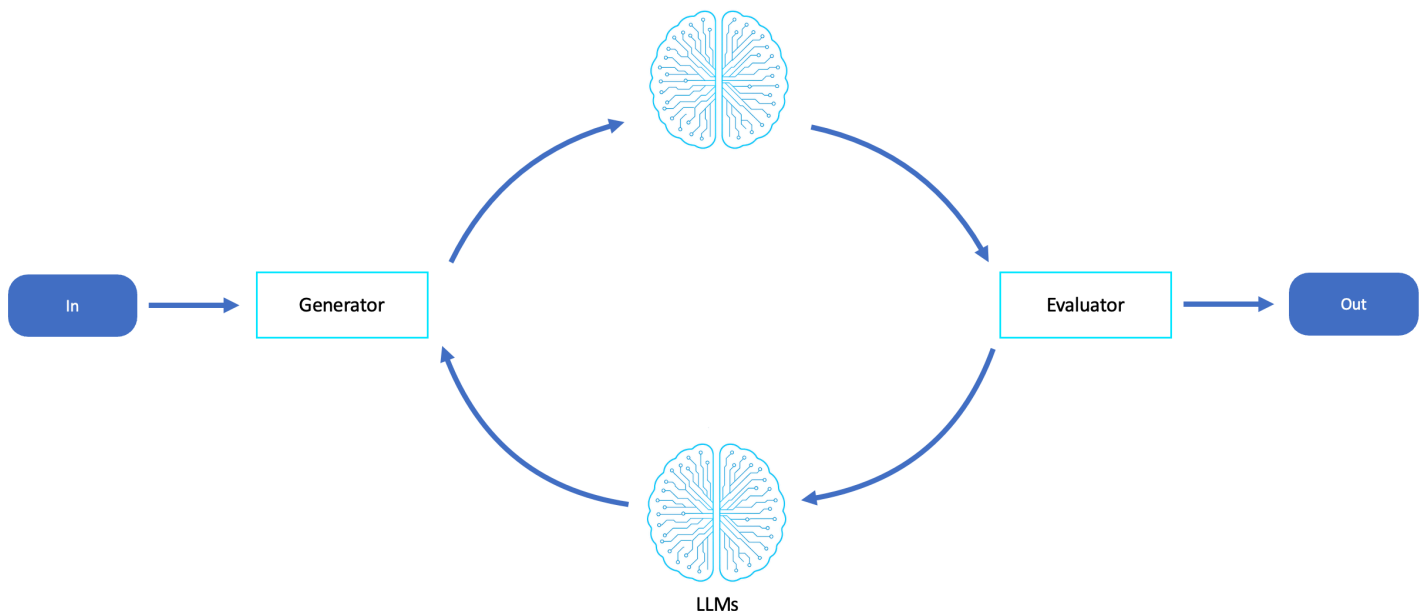
- O Orchestrator executa o metarraciocínio de metas
- Os agentes de trabalho podem incluir acesso à ferramenta, memória ou solicitação específica do domínio
- Pode ser hierárquico (ou seja, delegação de tarefas em vários níveis)

Casos de uso comuns

- Gerentes de projeto, pesquisadores coordenadores, escritores e agentes de garantia de qualidade
- Co-pilotos de codificação que combinam planejamento, execução e teste
- Agentes que supervisionam cadeias de ferramentas ou padrões de acesso à API

Fluxo de trabalho para avaliadores e ciclos de reflexão e refinamento

Esse fluxo de trabalho fornece um ciclo de feedback em que um LLM gera um resultado e outro avalia ou critica o resultado. Isso promove autorreflexão, otimização e melhorias iterativas.



O fluxo de trabalho do avaliador é ideal para cenários em que a qualidade, a precisão e o alinhamento da saída são importantes e em que a geração de passagem única não é confiável ou é

insuficiente. Esse fluxo de trabalho é excelente quando os agentes precisam fazer uma autocrítica, iterar e refinar seus resultados, seja para atender a um padrão mais alto de correção ou para explorar alternativas aprimoradas com base no feedback.

Esse fluxo de trabalho é particularmente eficaz quando:

- A saída envolve métricas de qualidade subjetivas (por exemplo, estilo, tom e legibilidade) ou critérios objetivos (por exemplo, correção, segurança e desempenho).
- O agente deve raciocinar por meio de compensações, avaliar restrições ou otimizar em direção a uma meta.
- Você precisa de redundância e garantia de qualidade integradas, especialmente em domínios regulamentados, voltados para o cliente ou criativos.
- Human-in-the-loop a revisão é cara ou não está disponível, e a validação autônoma é desejada.

Esse fluxo de trabalho é usado para geração de conteúdo, síntese e revisão de código, aplicação de políticas, verificação de alinhamento, ajuste de instruções e pós-processamento de RAG. Também é útil para agentes de autoaperfeiçoamento, onde o feedback contínuo ajuda a moldar respostas melhores ao longo do tempo para criar ciclos de decisão autônomos e confiáveis.

Casos de uso comuns

- Red-team agentes em comparação com agentes da equipe azul
- Agentes que geram, avaliam e revisam códigos ou planos
- Garantia de qualidade, detecção de alucinações e aplicação de estilo

Capacidades

- Suporta geração e avaliação dissociadas usando modelos diferentes (por exemplo, Claude para geração e Mistral para avaliação)
- O feedback é estruturado e usado para gerar resultados revisados
- Suporta várias iterações ou limites de convergência

Conclusão

Os LLMs fornecem o núcleo cognitivo dos agentes de software modernos, mas a invocação bruta do modelo não é suficiente para obter inteligência objetiva, robusta e controlável. Para passar da geração de resultados para o raciocínio estruturado e o comportamento alinhado a metas, os LLMs devem ser incorporados em padrões de fluxo de trabalho intencionais que definem como os modelos processam entradas, gerenciam contextos e coordenam ações.

Os fluxos de trabalho do LLM introduzem os fundamentos para criar o módulo cognitivo de um agente:

- O encadeamento imediato divide o raciocínio complexo em etapas modulares e auditáveis.
- O roteamento permite a classificação inteligente de tarefas e a delegação direcionada.
- A paralelização acelera a produtividade e promove um raciocínio diverso.
- A orquestração de agentes estrutura a colaboração entre vários agentes por meio da decomposição de tarefas e execução baseada em funções.
- O avaliador (loop reflect-refine) permite o autoaperfeiçoamento, o controle de qualidade e a verificação do alinhamento.

Cada fluxo de trabalho representa um padrão composto que pode ser adaptado às necessidades do agente, à complexidade da tarefa e às expectativas do usuário. Esses fluxos de trabalho não são mutuamente exclusivos. Eles são blocos de construção que geralmente são combinados em arquiteturas híbridas que oferecem suporte ao raciocínio dinâmico, à coordenação de vários agentes e à confiabilidade de nível corporativo.

À medida que você faz a transição para o próximo capítulo sobre padrões de fluxo de trabalho agentes, esses fluxos de trabalho de LLM reaparecerão como estruturas incorporadas em sistemas maiores, dando suporte à delegação de metas, orquestração de ferramentas, ciclos de decisão e autonomia do ciclo de vida. Dominar esses fluxos de trabalho de LLM é essencial para projetar agentes de software que não apenas prevejam texto, mas raciocinem, adaptem e ajam com propósito.

Padrões de fluxo de trabalho agentes

Os padrões de fluxo de trabalho agênticos integram agentes de software modulares com fluxos de trabalho estruturados de modelo de linguagem grande (LLM), permitindo raciocínio e ação autônomos. Embora inspirados nas arquiteturas tradicionais sem servidor e orientadas por eventos, esses padrões transferem a lógica central do código estático para os LLM-augmented agentes, proporcionando maior adaptabilidade e tomada de decisão contextual. Essa evolução transforma as arquiteturas de nuvem convencionais de sistemas determinísticos em sistemas capazes de interpretação dinâmica e aumento inteligente, mantendo os princípios fundamentais de escalabilidade e capacidade de resposta.

Nesta seção

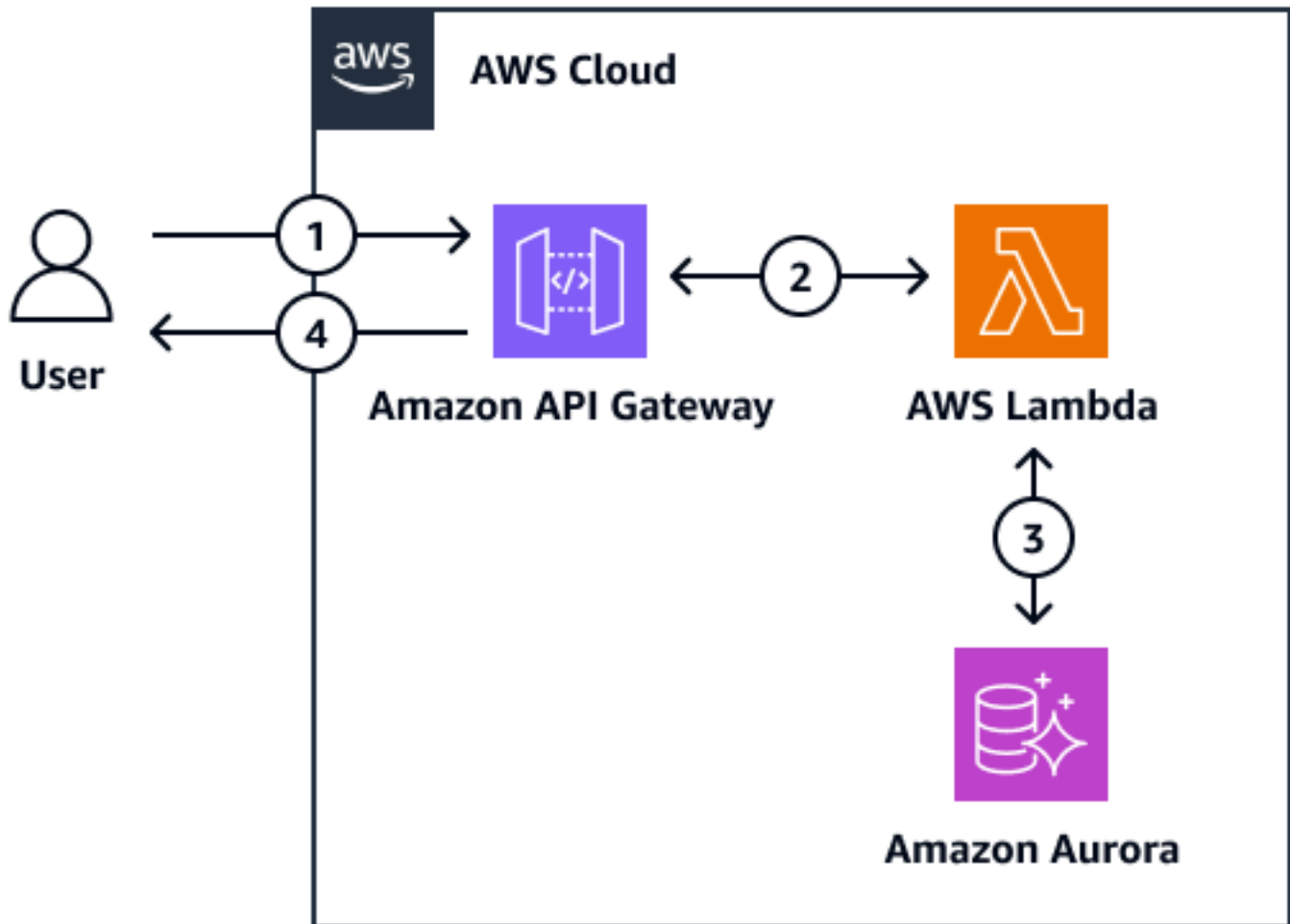
- [De sistemas orientados a eventos a sistemas aumentados por cognição](#)
- [Padrões de saga de encadeamento imediato](#)
- [Padrões de despacho dinâmico de roteamento](#)
- [Padrões de paralelização e coleta de dispersão](#)
- [Padrões de orquestração da saga](#)
- [O avaliador reflete e refina os padrões de loop](#)
- [Projetando fluxos de trabalho agentes em AWS](#)
- [Conclusão](#)

De sistemas orientados a eventos a sistemas aumentados por cognição

As arquiteturas de nuvem modernas, especialmente aquelas baseadas em princípios sem servidor e orientadas por eventos, tradicionalmente se baseiam em padrões como roteamento, distribuição e enriquecimento para criar sistemas responsivos e escaláveis. Os sistemas de IA da Agentix se baseiam nessas bases, ao mesmo tempo que as reformulam em torno do LLM-augmented raciocínio e da flexibilidade cognitiva. Essa abordagem permite recursos mais sofisticados de resolução de problemas e automação, potencialmente revolucionando a forma como tarefas complexas são tratadas em ambientes de nuvem.

Event-driven arquitetura

O diagrama a seguir mostra um sistema distribuído típico:

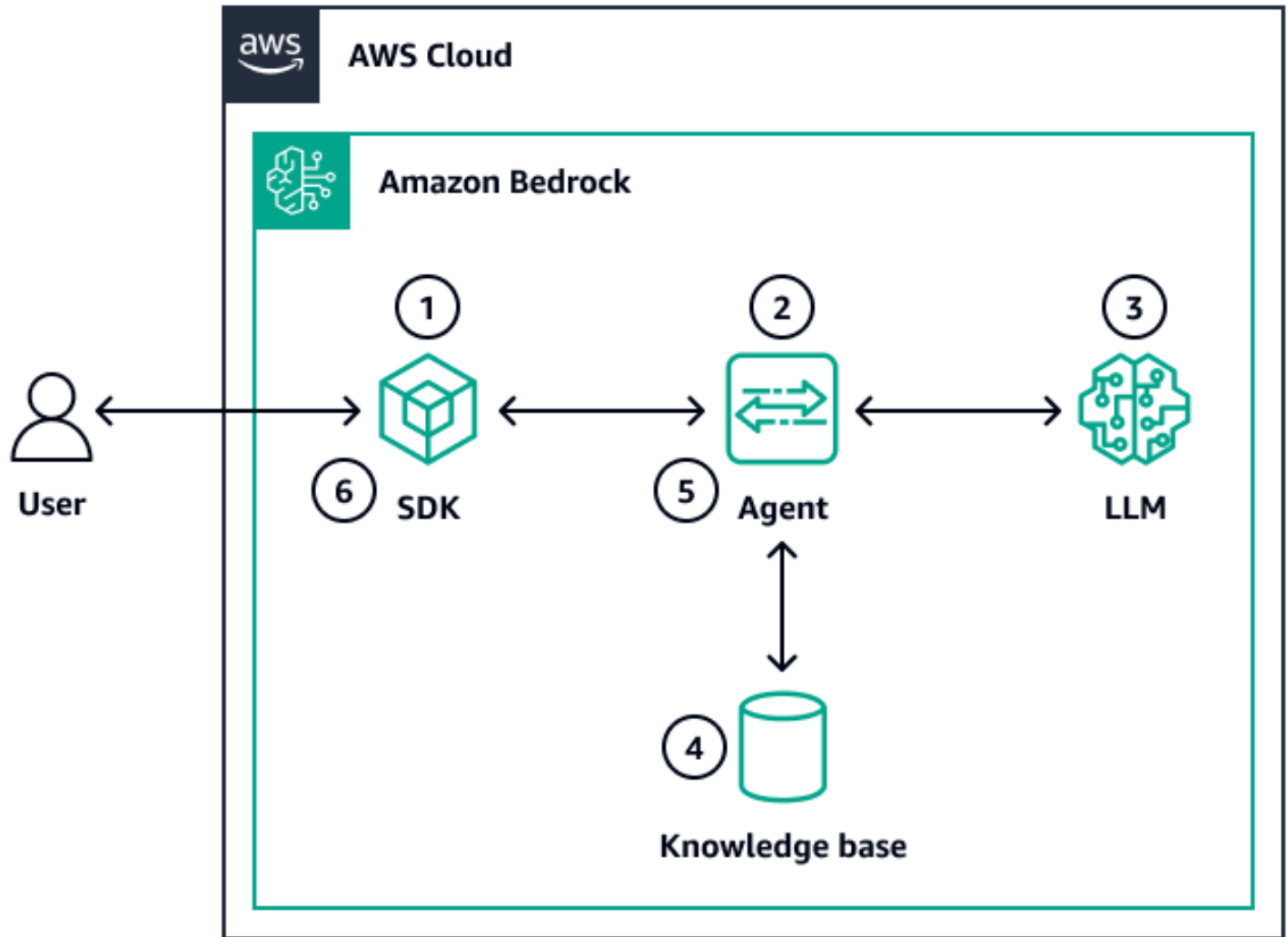


1. Um usuário envia uma solicitação para o Amazon API Gateway.
2. O Amazon API Gateway encaminha a solicitação para uma AWS Lambda função.
3. AWS Lambda realiza o enriquecimento de dados consultando um banco de dados Amazon Aurora
4. O Amazon API Gateway retorna a carga útil enriquecida para o chamador.

Essa estrutura é confiável e escalável, mas é fundamentalmente estática. As regras de negócios e os caminhos lógicos devem ser codificados explicitamente, e a adaptação a contextos em mudança ou a informações incompletas é limitada.

Cognition-augmented fluxos de trabalho

As arquiteturas agênticas adicionam aumento cognitivo a um sistema orientado por eventos. O diagrama a seguir mostra um equivalente agente:



1. Um usuário envia uma consulta por meio de uma chamada de SDK ou API.
2. Um agente do Amazon Bedrock recebe a consulta.
3. O agente interpreta a consulta invocando um LLM
4. O agente realiza o enriquecimento semântico pesquisando na base de conhecimento do Amazon Bedrock ou em outras fontes de dados externas.
5. O LLM sintetiza uma resposta rica em contexto e alinhada a objetivos.
6. O sistema retorna uma resposta sintetizada para o usuário.

Nesse fluxo, o LLM usa a lógica, entende a intenção, recupera e combina o contexto relevante e, em seguida, decide a melhor forma de responder. Esse padrão reflete o padrão tradicional de enriquecimento, em que as mensagens são aumentadas com dados externos antes de serem roteadas posteriormente. Em sistemas agênticos, entretanto, esse enriquecimento não é uma pesquisa estática. Em vez disso, o enriquecimento é dinâmico, guiado semanticamente e orientado por um propósito.

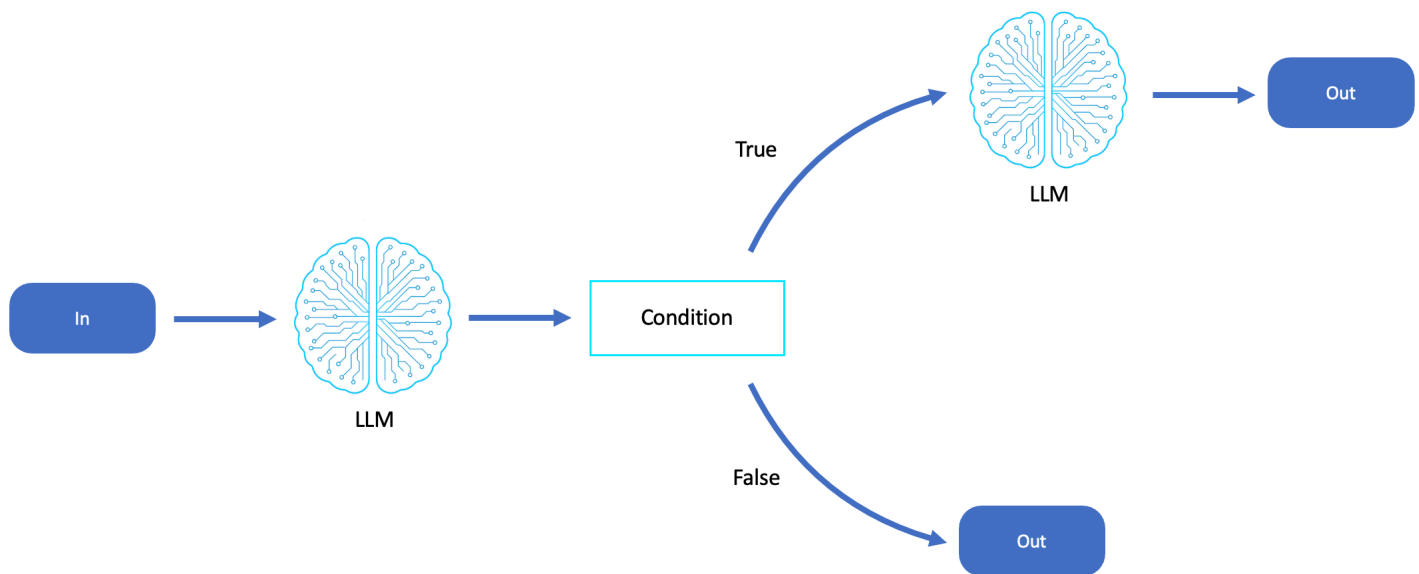
Insights principais

Cada fluxo de trabalho do LLM pode ser mapeado para um padrão de fluxo de trabalho agente, que reflete e desenvolve os estilos tradicionais de arquitetura orientada a eventos. Um alicerce básico dos fluxos de trabalho agentes é a capacidade de ampliar o contexto de um LLM com dados, ferramentas e memória. Isso cria um ciclo de raciocínio informado, adaptável e alinhado com a intenção do usuário. Enquanto os sistemas tradicionais enriquecem as mensagens com dados de pesquisa, os sistemas agentes permitem que o software atue menos como scripts e mais como colaboradores inteligentes.

Padrões de saga de encadeamento imediato

Ao reimaginar o encadeamento imediato do LLM como uma saga orientada por eventos, desbloqueamos um novo modelo operacional: os fluxos de trabalho se tornam distribuídos, recuperáveis e coordenados semanticamente entre agentes autônomos. Cada etapa de resposta rápida é reformulada como uma tarefa atômica, emitida como um evento, consumida por um agente dedicado e enriquecida com metadados contextuais.

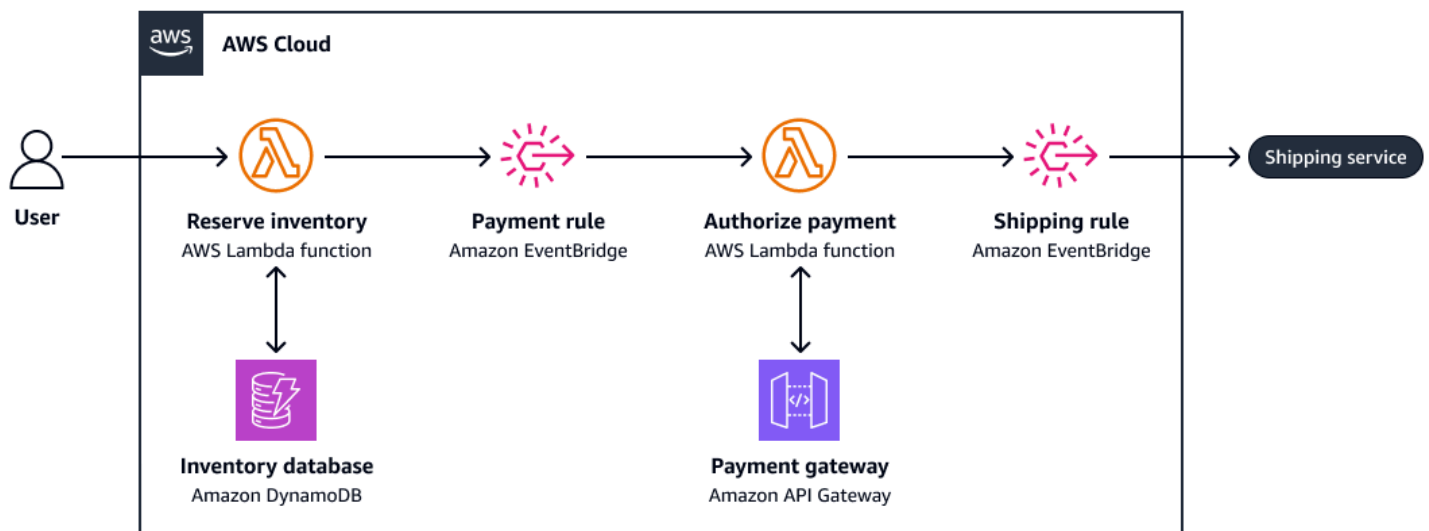
O diagrama a seguir é um exemplo do encadeamento de prompts do LLM:



Coreografia saga

O padrão coreográfico da saga é uma abordagem de implementação em sistemas distribuídos que não tem um coordenador central. Em vez disso, cada serviço ou componente publica eventos que acionam a próxima ação do fluxo de trabalho. Esse padrão é amplamente usado em sistemas distribuídos para gerenciar transações em vários serviços. Em uma saga, o sistema executa uma série de transações locais coordenadas. Se um falhar, o sistema aciona ações compensatórias para manter a consistência.

O diagrama a seguir é um exemplo de coreografia de saga:



1. Reservar inventário

2. Autorizar pagamento
3. Criar pedido de envio

Se a etapa 3 falhar, o sistema invoca ações compensatórias (por exemplo, cancelar um pagamento ou liberar o inventário).

Esse padrão é especialmente valioso em arquiteturas orientadas a eventos, nas quais os serviços são fracamente acoplados e os estados devem ser resolvidos de forma consistente ao longo do tempo, mesmo na presença de falha parcial.

Padrão de encadeamento imediato

O encadeamento imediato se assemelha ao padrão da saga, tanto na estrutura quanto no propósito. Ele executa uma série de etapas de raciocínio que são construídas sequencialmente, preservando o contexto e permitindo reversões e revisões.

Coreografia do agente

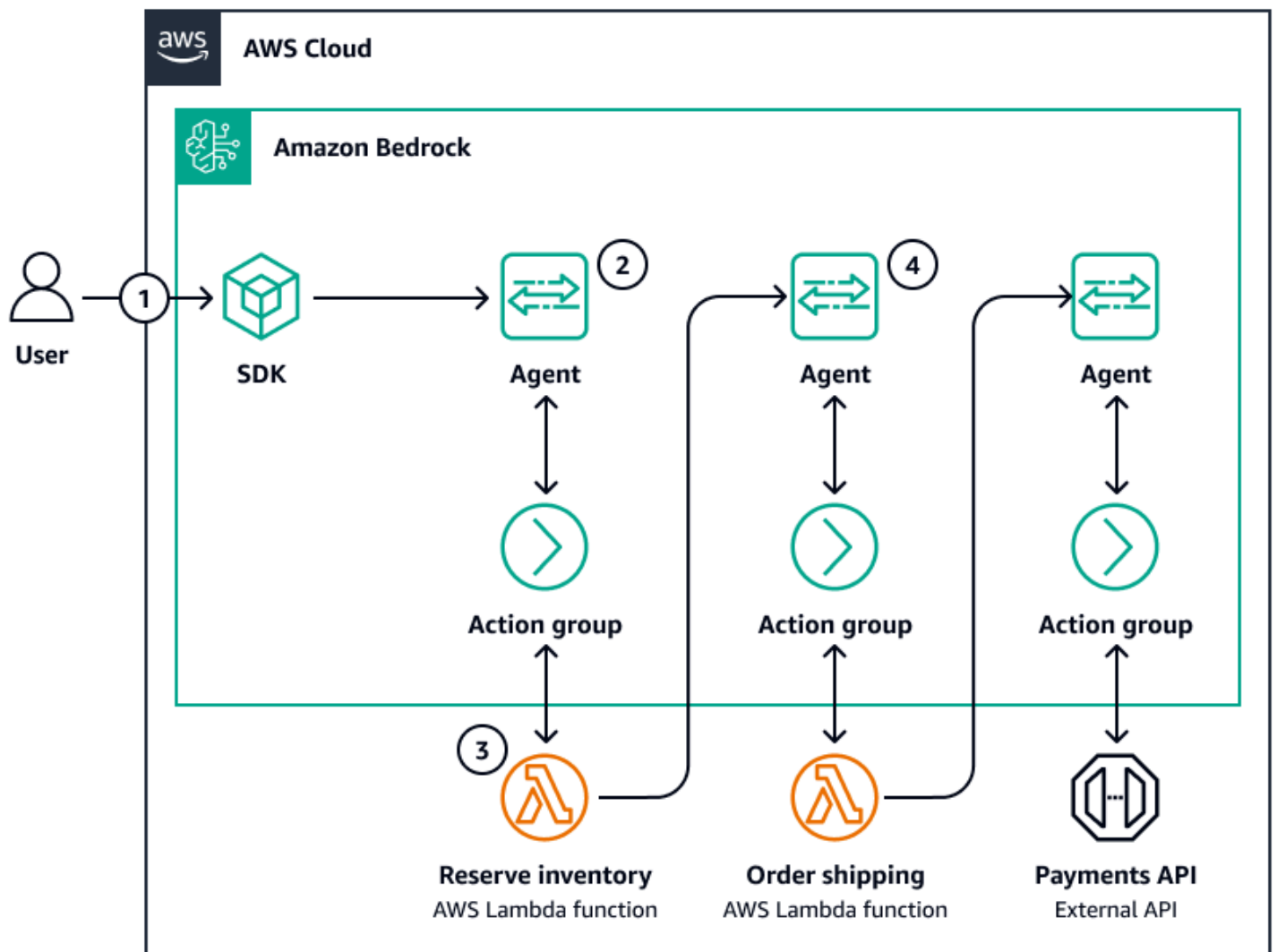
1. O LLM interpreta uma consulta complexa do usuário e gera uma hipótese
2. LLM elabora um plano para resolver a tarefa
3. O LLM executa uma subtarefa (por exemplo, usando uma chamada de ferramenta ou recuperando conhecimento)
4. O LLM refina a saída ou revisita uma etapa anterior se considerar um resultado insatisfatório

Se um resultado intermediário apresentar falhas, o sistema poderá fazer o seguinte:

- Repita as etapas usando uma abordagem diferente
- Reverta para uma solicitação anterior e replaneje
- Use um loop de avaliação (por exemplo, do padrão avaliador-otimizador) para detectar e corrigir falhas

Assim como o padrão da saga, o encadeamento imediato permite mecanismos parciais de progresso e reversão. Isso acontece por meio de refinamento e LLM-directed correção iterativos, em vez de compensar as transações do banco de dados.

O diagrama a seguir é um exemplo de coreografia de agentes:



1. Um usuário envia uma consulta por meio de um SDK.
2. Um agente do Amazon Bedrock orchestra o raciocínio por meio do seguinte:
 - Interpretação (LLM)
 - Planejamento (LLM)
 - Execução por meio de uma ferramenta ou base de conhecimento
 - Construção da resposta
3. Se uma ferramenta falhar ou retornar dados insuficientes, o agente poderá replanejar ou reformular dinamicamente a tarefa.
4. A memória (por exemplo, um armazenamento vetorial de curto prazo) pode preservar seu estado em todas as etapas

Takeaways

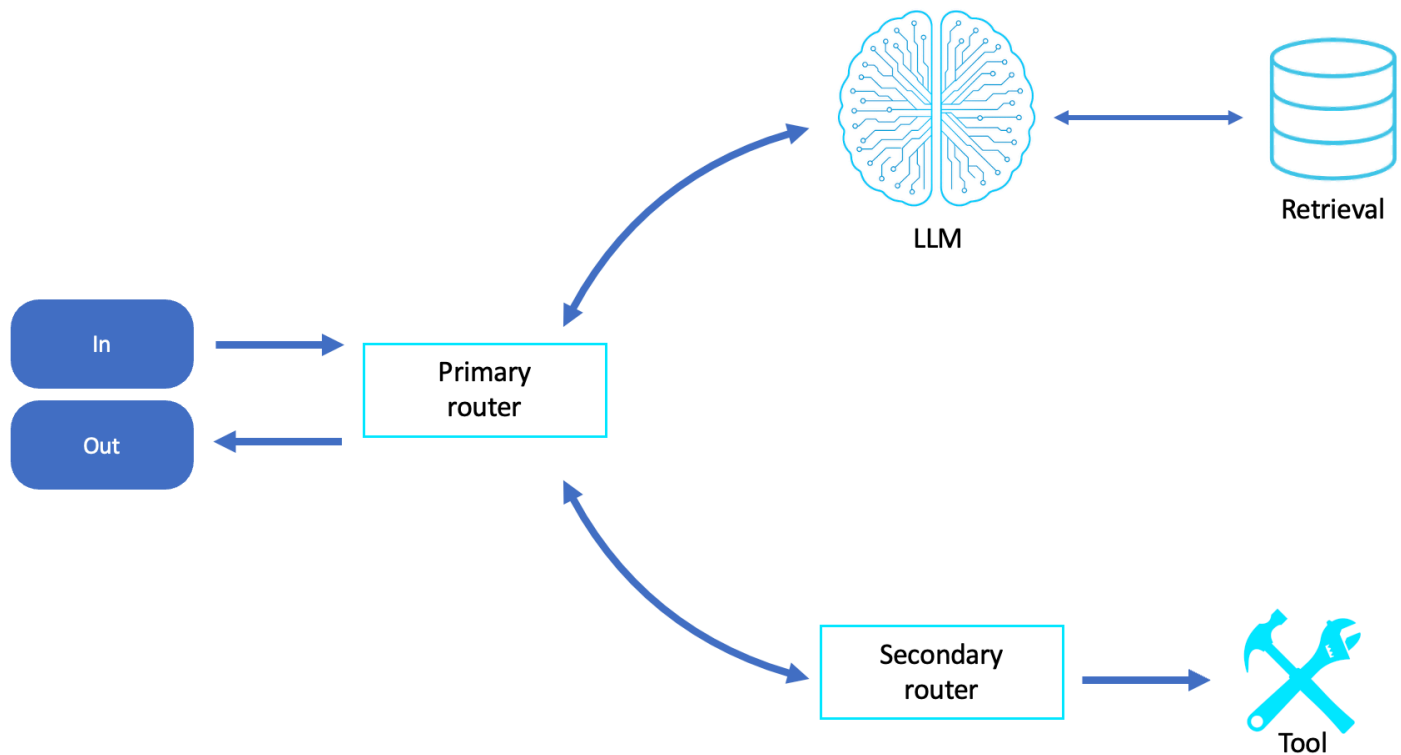
Enquanto o padrão da saga gerencia chamadas de serviço distribuídas com lógica de compensação, o encadeamento imediato gerencia as tarefas de raciocínio com sequenciamento reflexivo e replanejamento adaptativo. Ambos os sistemas permitem progresso incremental, pontos de decisão descentralizados e recuperação de falhas, e tudo isso por meio de raciocínio informado, em vez de reversão rígida.

O encadeamento imediato introduz o raciocínio transacional, que é o equivalente cognitivo das sagas. Ou seja, cada “pensamento” é reavaliado, revisado ou abandonado como parte de um diálogo mais amplo direcionado a um objetivo.

Padrões de despacho dinâmico de roteamento

Em sistemas agentes modernos, onde as tarefas variam da análise de documentos à geração autônoma de software, a capacidade de rotear solicitações dinamicamente para o modelo de linguagem grande (LLM) ou agente mais capaz se torna essencial. A lógica de roteamento estático, geralmente incorporada em scripts de orquestração ou camadas de API, carece da adaptabilidade necessária para ambientes em tempo real, com vários modelos e com vários recursos. Para resolver isso, os fluxos de trabalho de roteamento do LLM podem ser transformados em uma arquitetura orientada a eventos que aproveita um padrão de despacho dinâmico, transformando as chamadas do LLM em eventos roteados de forma inteligente e sensíveis ao contexto.

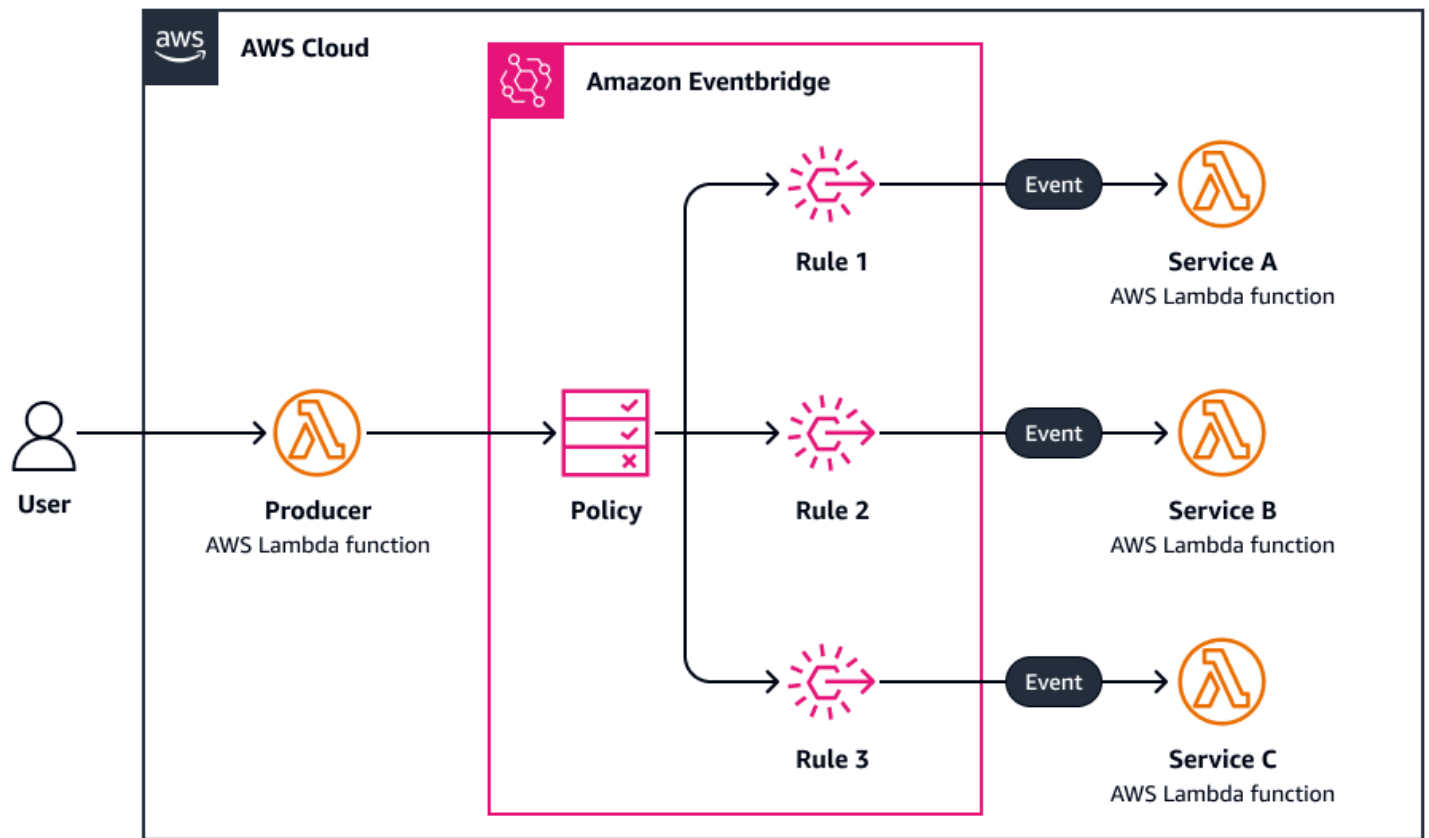
O diagrama a seguir é um exemplo de roteamento LLM:



Envio dinâmico

Em sistemas distribuídos tradicionais, o padrão de despacho dinâmico seleciona e invoca serviços específicos em tempo de execução com base nos atributos de eventos de entrada, como tipo de evento, origem e carga útil. Isso geralmente é implementado usando a Amazon EventBridge, que pode avaliar e rotear eventos recebidos para destinos apropriados (por exemplo AWS Step Functions, AWS Lambda funções ou tarefas do Amazon Elastic Container Service).

O diagrama a seguir é um exemplo de despacho dinâmico:



1. Um aplicativo emite um evento (por exemplo, `{"type": "orderCreated", "priority": "high"}`).
2. A Amazon EventBridge avalia o evento de acordo com suas regras de roteamento.
3. Com base nos atributos de um evento, o sistema despacha dinamicamente para o seguinte:
 - `HighPriorityOrderProcessor`(serviço A)
 - `StandardOrderProcessor`(serviço B)
 - `UpdateOrderProcessor`(serviço C)

Esse padrão oferece suporte a acoplamento frouxo, especialização baseada em domínio e extensibilidade de tempo de execução. Isso permite que os sistemas respondam de forma inteligente às mudanças nos requisitos e na semântica dos eventos.

LLM-based roteamento

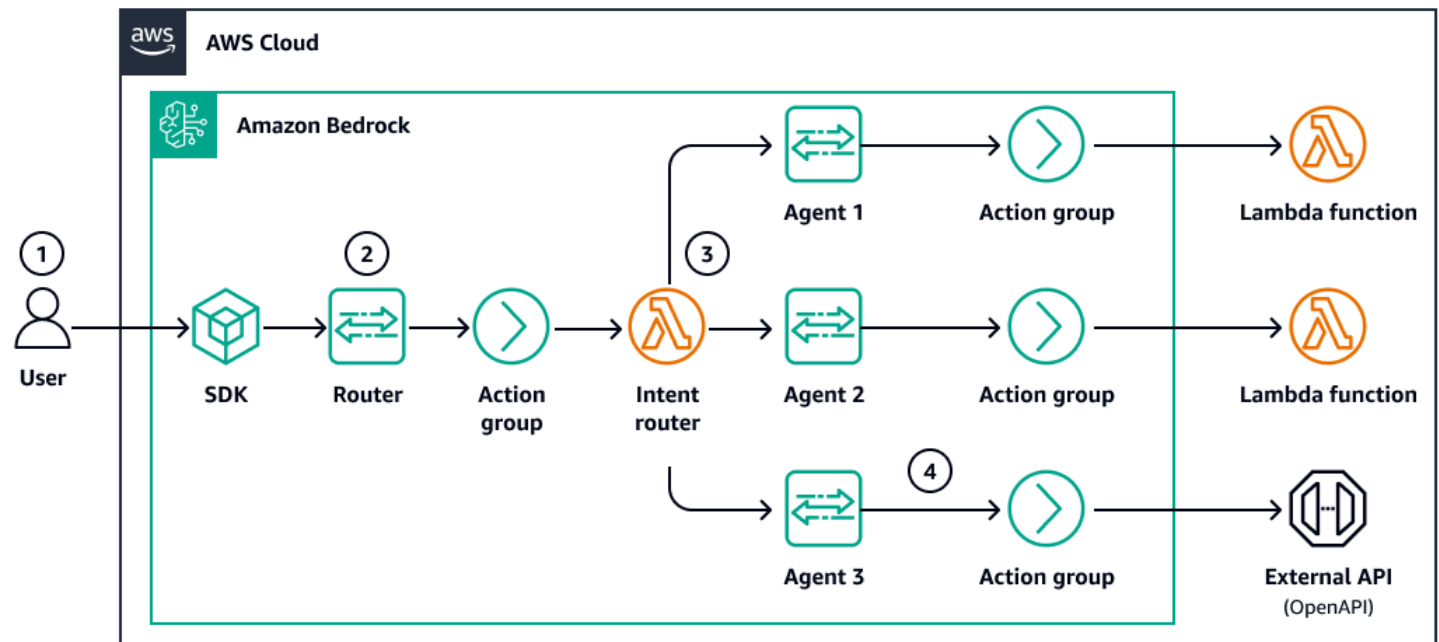
Em sistemas agentes, o roteamento também executa a delegação dinâmica de tarefas — mas, em vez EventBridge das regras ou filtros de metadados da Amazon, o LLM classifica e interpreta a intenção do usuário por meio da linguagem natural. O resultado é uma forma de despacho flexível, semântica e adaptável.

Roteador de agente

Essa arquitetura permite um envio rico baseado em intenção sem esquemas ou tipos de eventos predefinidos, o que é ideal para entradas não estruturadas e consultas complexas.

1. Um usuário envia a solicitação “Você pode me ajudar a revisar os termos do meu contrato?”
2. O LLM interpreta isso como uma tarefa de documento legal.
3. O agente encaminha a tarefa para uma ou mais das seguintes opções:
 - Modelo de solicitação de revisão de contrato
 - Subagente de raciocínio jurídico
 - Ferramenta de análise de documentos

O diagrama a seguir é um exemplo de um roteador de agente:



1. Um usuário envia uma solicitação de linguagem natural por meio de um SDK.
2. Um agente do Amazon Bedrock usa um LLM para classificar a tarefa (por exemplo, jurídica, técnica ou agendamento).
3. O agente roteia dinamicamente a tarefa por meio de um grupo de ações para invocar o agente necessário:
 - Domain-specific agente
 - Cadeia de ferramentas especializada

- Configuração de prompt personalizada
4. O manipulador selecionado processa a tarefa e retorna uma resposta personalizada.

Takeaways

Enquanto o despacho dinâmico tradicional usa EventBridge as regras da Amazon para roteamento com base em atributos de eventos estruturados, o roteamento agente usa LLMs para classificar e rotear semanticamente as tarefas com base no significado e na intenção. Isso expande a flexibilidade do sistema ao permitir o seguinte:

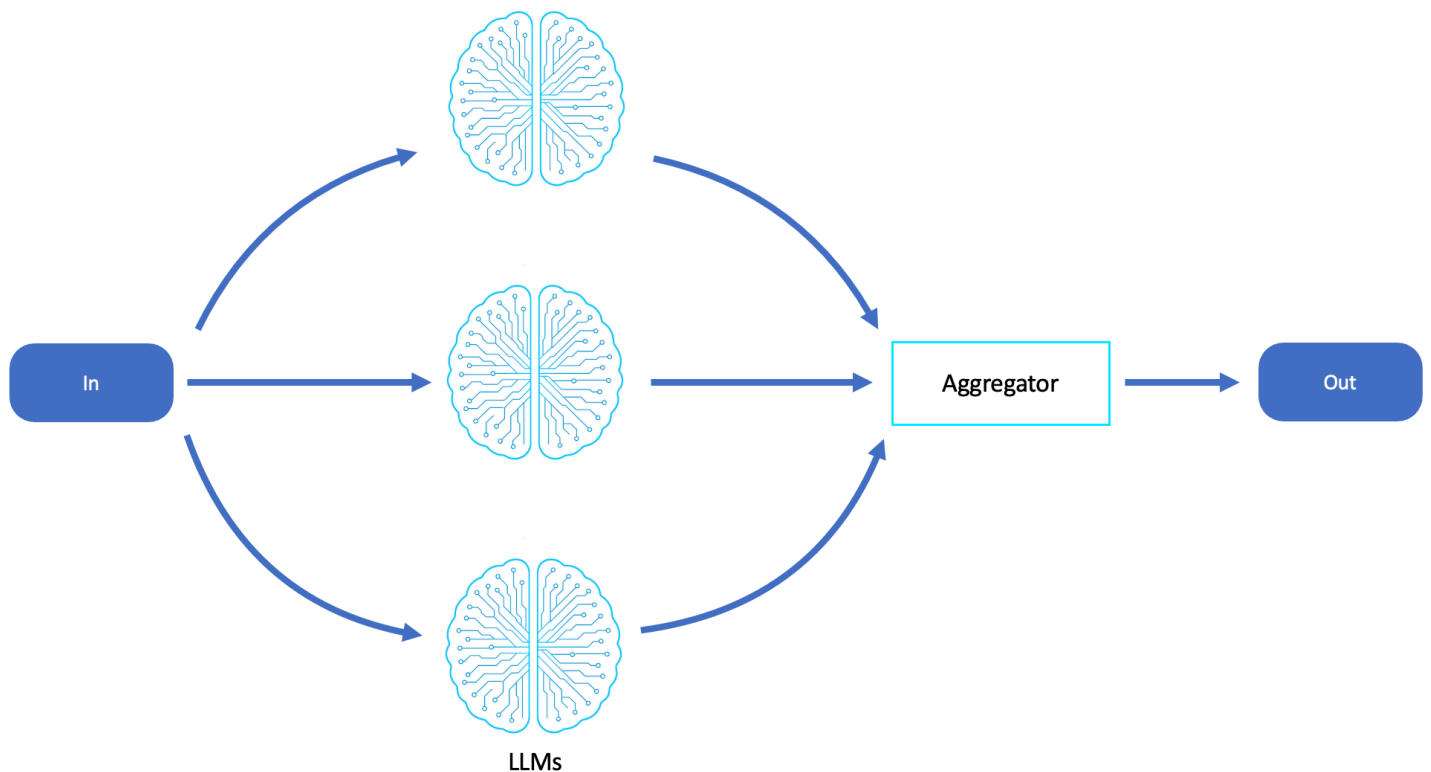
- Compreensão mais ampla de entradas
- Recuo inteligente e seleção de ferramentas
- Extensibilidade natural por meio de novas funções de agente ou estilos de solicitação

O roteamento agente substitui regras rígidas pelo despacho cognitivo dinâmico, o que permite que os sistemas evoluam com a linguagem e não com o código.

Padrões de paralelização e coleta de dispersão

Muitas tarefas avançadas de raciocínio e geração — como resumir documentos grandes, avaliar vários caminhos de solução ou comparar perspectivas diversas — se beneficiam da execução paralela de prompts. Os fluxos de trabalho sequenciais tradicionais são insuficientes quando são necessárias escalabilidade, capacidade de resposta e tolerância a falhas. Para superar isso, a LLM-based paralelização pode ser reimaginada usando um padrão de dispersão e coleta orientado por eventos, em que as tarefas são distribuídas dinamicamente para agentes autônomos e os resultados sintetizados de forma inteligente.

O diagrama a seguir é um exemplo de um fluxo de trabalho de paralelização do LLM:



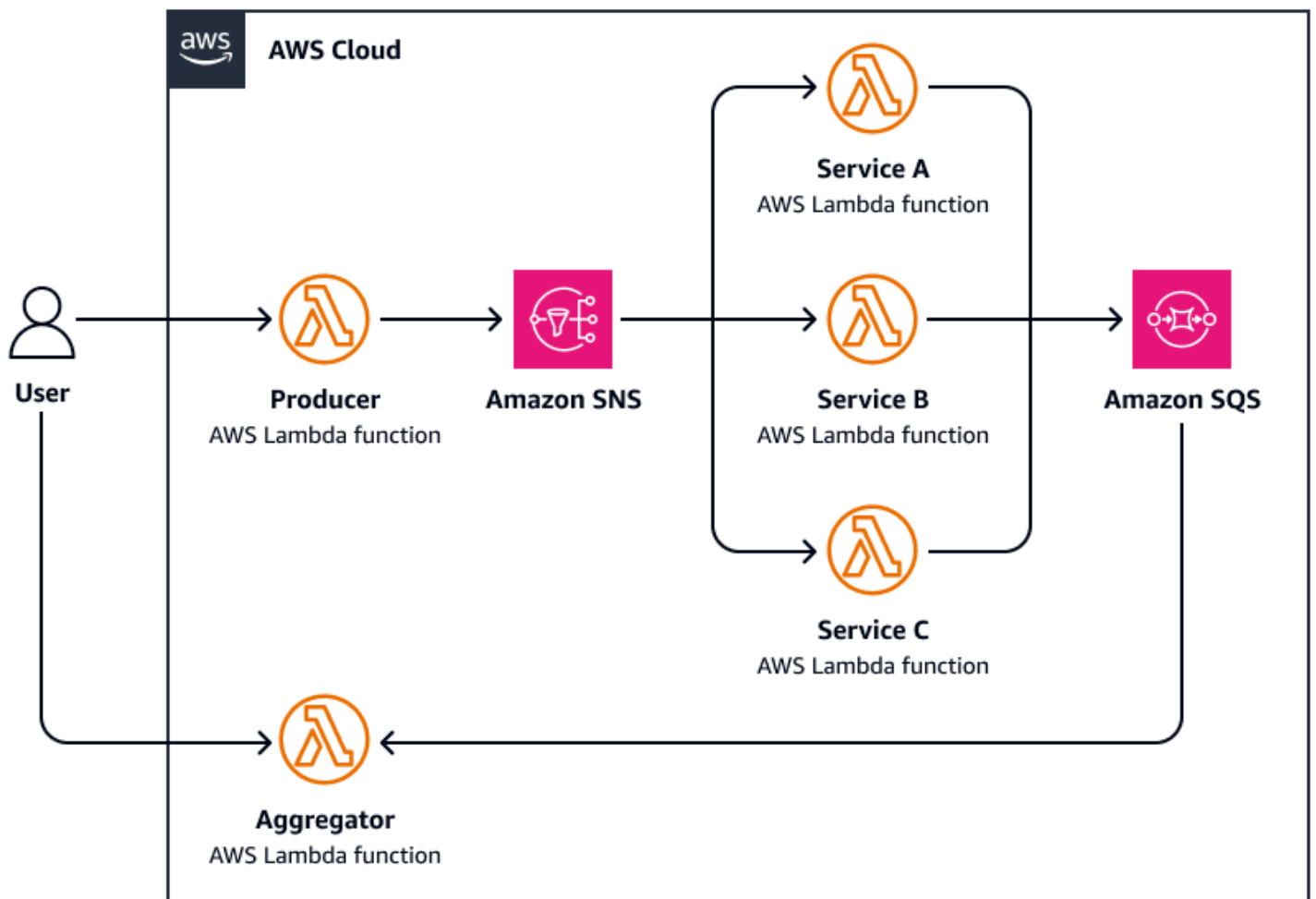
Scatter-gather

Em sistemas distribuídos, um padrão de dispersão envia tarefas para vários serviços ou unidades de processamento em paralelo, aguarda suas respostas e, em seguida, agrega os resultados em uma saída consolidada. Ao contrário do fan-out, o scatter-gather é coordenado porque espera respostas e geralmente aplica a lógica para combinar, comparar e selecionar resultados.

As implementações comuns para paralelização e coleta de dispersão incluem o seguinte:

- AWS Step Functions mapear um estado para execução paralela de tarefas
- AWS Lambda com simultaneidade, coordenando resultados de várias funções invocadas
- Amazon EventBridge com IDs de correlação e fluxos de trabalho de agregação
- Padrão de controlador personalizado para gerenciar o fan-out e coletar resultados usando o Amazon Simple Storage Service (Amazon S3), o Amazon DynamoDB ou filas

O diagrama a seguir é um exemplo de coleta dispersa:



1. Um usuário envia uma solicitação para uma função de coordenador central que dispersa a tarefa publicando mensagens paralelas em um tópico do Amazon Simple Notification Service (Amazon SNS).
2. Cada mensagem inclui metadados da tarefa e é encaminhada para um funcionário especializado. AWS Lambda
3. Cada funcionário processa de AWS Lambda forma independente sua subtarefa atribuída (por exemplo, consultar uma API externa, processar um documento e analisar dados).
4. Os resultados são gravados em uma camada de armazenamento comum, como o Amazon Simple Queue Service (Amazon SQS).
5. A função agregadora espera que todas as respostas sejam concluídas e, em seguida, faz o seguinte:
 - Reúne e agrega os resultados (por exemplo, mescla resumos, seleciona as melhores correspondências)

- Envia uma resposta final ou aciona um fluxo de trabalho posterior

Os casos de uso comuns para padrões de coleta de dispersão incluem o seguinte:

- Pesquisa federada
- Mecanismos de comparação de preços
- Análise de dados agregados
- Inferência multimodelo

LLM-based paralelização (cognição de dispersão e coleta)

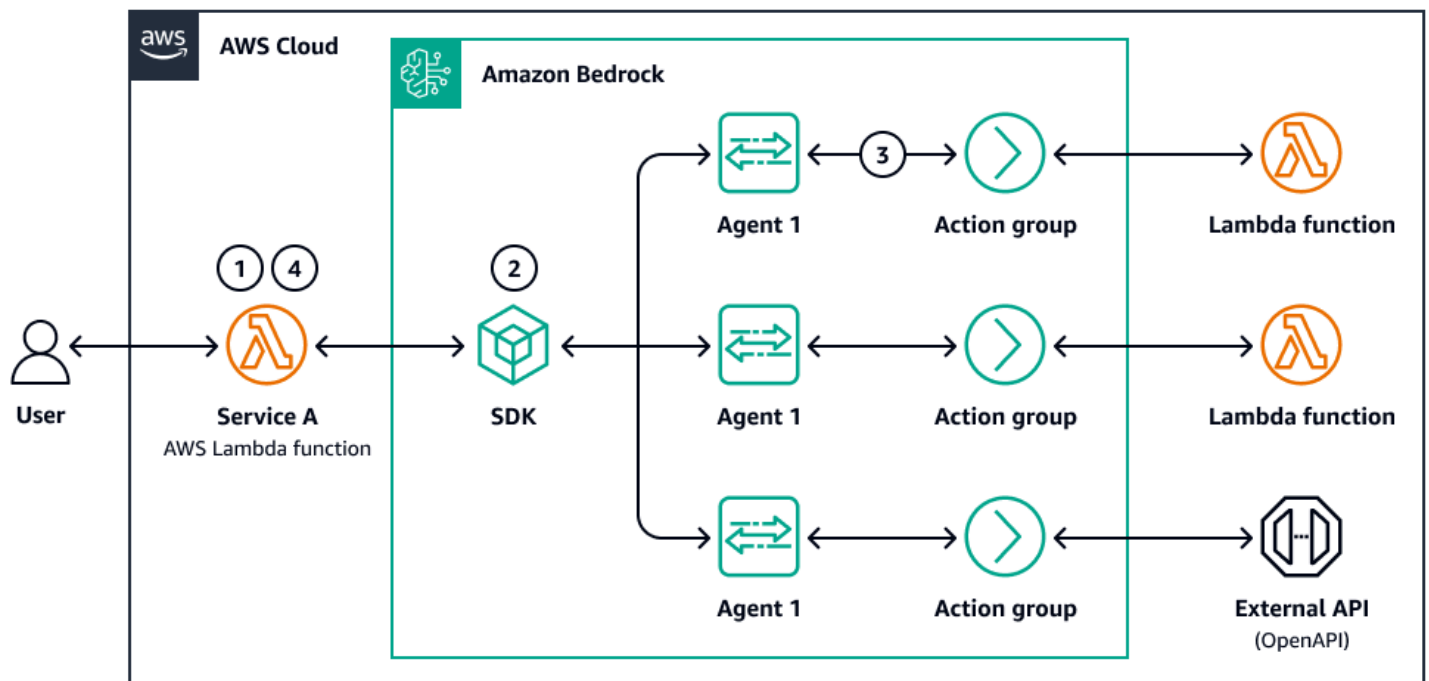
Em sistemas agentes, a paralelização reflete de perto a coleta dispersa, distribuindo subtarefas em várias chamadas ou agentes do LLM, cada um raciocinando de forma independente sobre uma parte do problema. Os resultados retornados são coletados e sintetizados por um processo de agregação, que geralmente é outro agente LLM ou controlador.

Paralelização de agentes

1. Um agente envia uma solicitação “Resuma os insights nesses 10 relatórios”.
2. Ele dispersa os relatórios em 10 tarefas paralelas de resumo do LLM.
3. Ao retornar todos os resumos, o agente faz o seguinte:
 - Agrega resumos em um briefing unificado
 - Identifica temas ou contradições
 - Envia a saída sintetizada para o usuário

Esse fluxo de trabalho agente permite um raciocínio paralelo escalável, modular e adaptável. Isso é ideal para casos de uso que exigem alto rendimento cognitivo.

O diagrama a seguir é um exemplo de paralelização de agentes:



1. Um usuário envia uma consulta com várias partes ou um conjunto de documentos.
2. Um controlador AWS Lambda ou função de etapa distribui as subtarefas. Cada tarefa invoca uma chamada ou subagente do Amazon Bedrock LLM com seu próprio prompt.
3. Quando as chamadas e subtarefas são concluídas, os resultados são armazenados (por exemplo, no Amazon S3 ou no armazenamento de memória) e uma etapa de agregação mescla, compara ou filtra as saídas.
4. O sistema retorna a resposta final ao usuário ou ao agente downstream.

Esse sistema tem um ciclo de raciocínio distribuído com rastreabilidade, tolerância a falhas e ponderação opcional de resultados ou lógica de seleção.

Takeaways

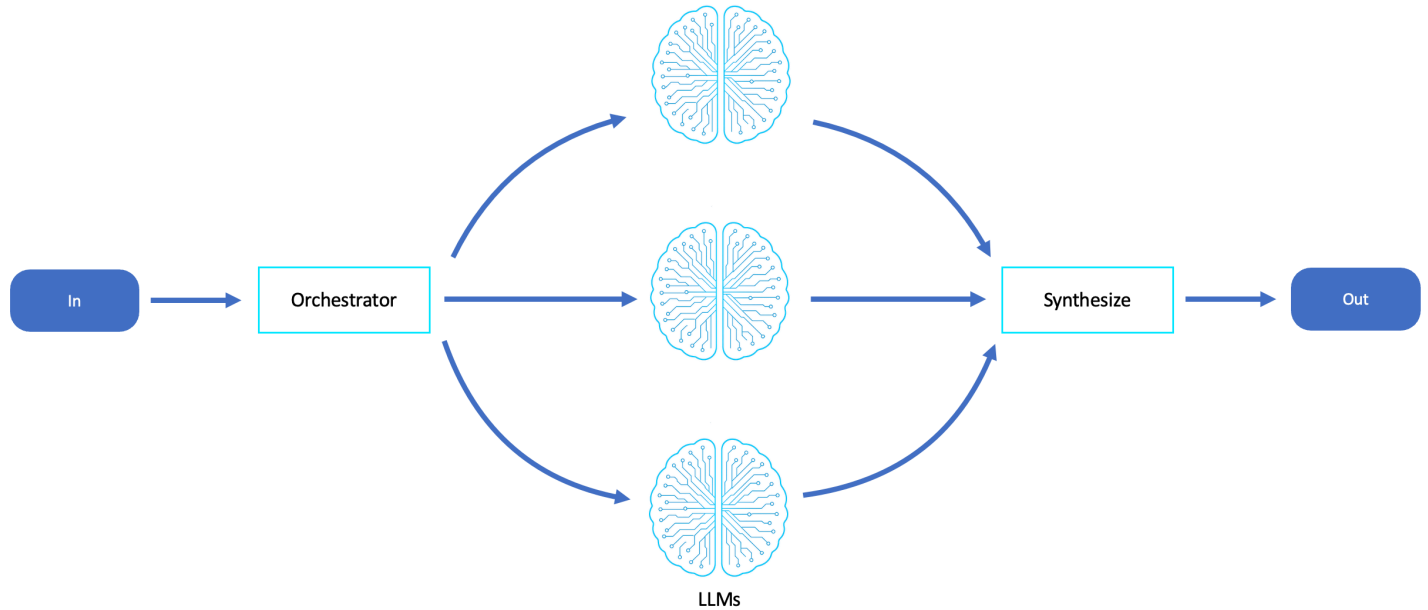
A paralelização agente usa padrões de coleta de dispersão para distribuir tarefas de LLM, permitindo processamento paralelo e síntese inteligente de resultados.

Padrões de orquestração da saga

À medida que os fluxos de trabalho conduzidos por LLMs se tornam cada vez mais complexos, abrangendo cadeias imediatas, etapas de processamento de dados, invocações de ferramentas e

colaboração de agentes, a necessidade de orquestração inteligente se torna essencial. Em vez de depender de scripts fortemente acoplados ou fluxos de execução estáticos predeterminados, esses fluxos de trabalho podem ser implementados como padrões de orquestração orientados por eventos, permitindo que os LLM-based sistemas coordenem, monitorem e adaptem dinamicamente tarefas de várias etapas em agentes autônomos.

O diagrama a seguir é um exemplo de um orquestrador:



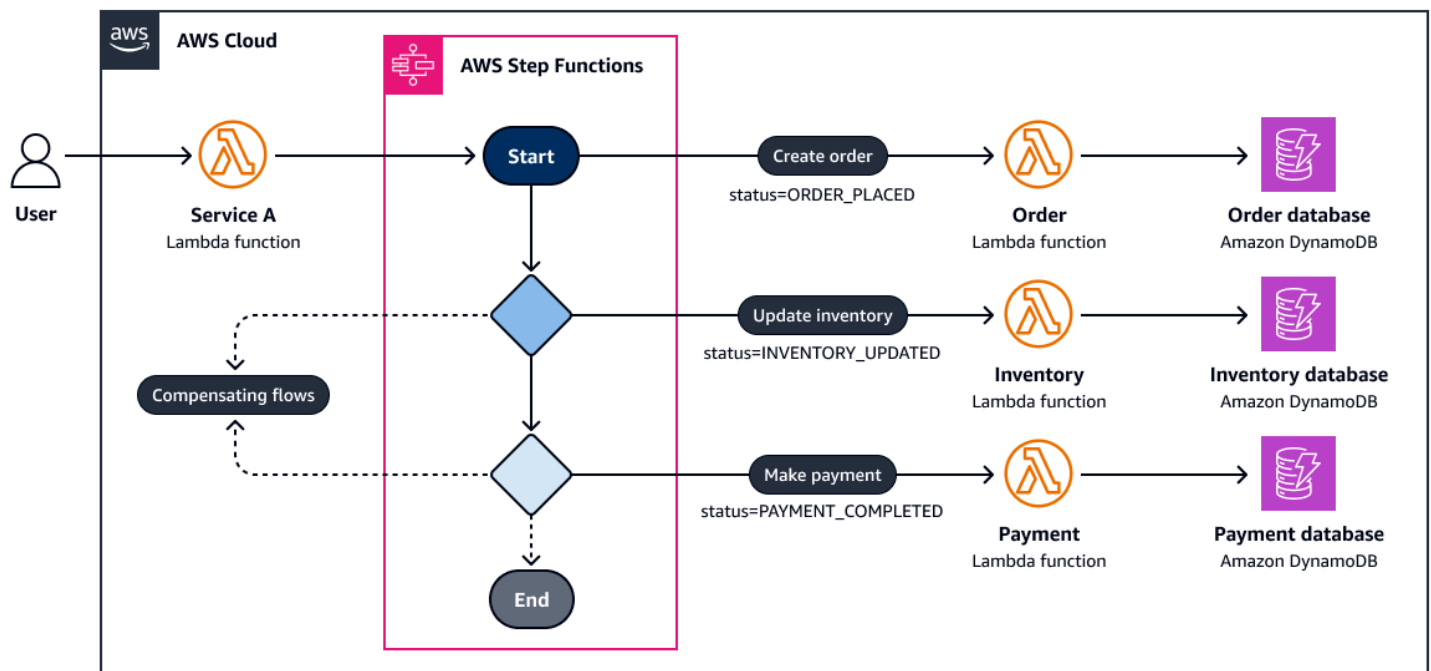
Orquestração de eventos

Em sistemas distribuídos tradicionais, a orquestração de eventos se refere a um padrão no qual um coordenador central gerencia um fluxo de trabalho complexo direcionando explicitamente o fluxo de controle em vários serviços ou tarefas. Diferentemente da coreografia de eventos (em que cada serviço reage de forma independente), a orquestração fornece lógica, visibilidade e controle centralizados sobre todo o processo.

Isso geralmente é implementado usando as seguintes ferramentas:

- AWS Step Functions— Definir e executar fluxos de trabalho com estado
- AWS Lambda— Execute tarefas discretas dentro do fluxo orquestrado
- Amazon SQS ou Amazon EventBridge — aciona etapas ou respostas assíncronas

O diagrama a seguir é um exemplo de orquestração de saga:



Um AWS Step Functions fluxo de trabalho gerencia o processo de pedido de um cliente:

1. Criar pedido (AWS Lambda)
2. Atualizar inventário (AWS Lambda)
3. Efetue o pagamento (AWS Lambda)

O orquestrador coordena cada etapa gerenciando novas tentativas, ramificações paralelas, tempos limite e falhas.

Role-based sistema de agentes (orquestrador)

Em sistemas agentes, o padrão do orquestrador reflete a orquestração de eventos, mas distribui a lógica entre vários agentes de raciocínio, cada um com uma função ou especialização definida. Um agente orquestrador central interpreta a tarefa geral, a decompõe em subtarefas e as delega aos agentes de trabalho, cada um otimizado para um domínio específico (por exemplo, pesquisa, codificação, resumo, revisão).

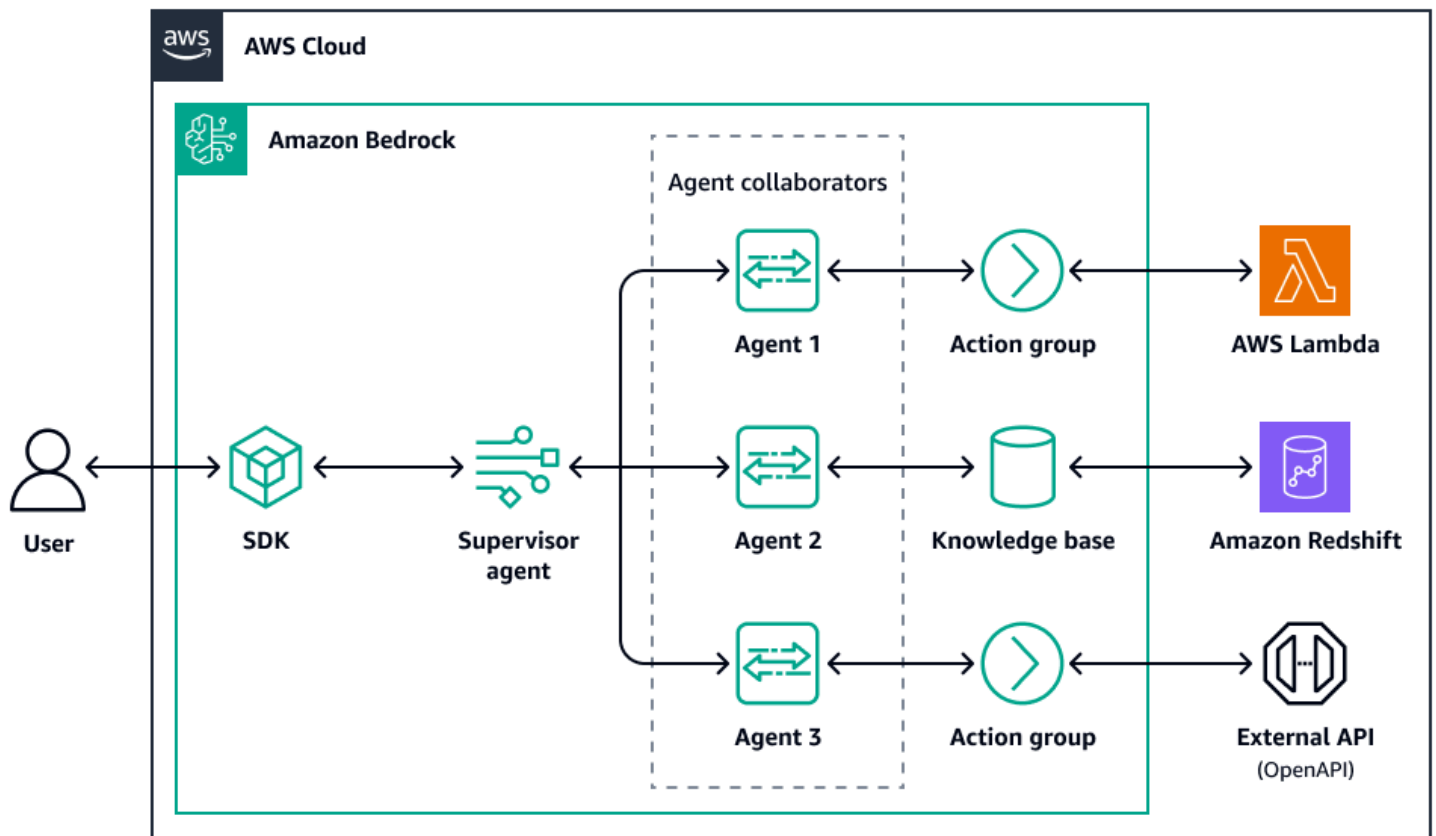
Supervisor

1. Um usuário envia a consulta “Crie um resumo do projeto e resuma os 5 principais concorrentes”.
2. O agente orquestrador faz o seguinte:

- Designa um agente de pesquisa para encontrar dados da concorrência
- Envia as descobertas brutas para um agente de sumarização
- Transmite os resultados para um agente redator de resumos
- Compila a saída final para o usuário

Cada agente opera de forma independente, mas o orquestrador coordena as tarefas. É como uma função Lambda que lida com tarefas de fluxo de trabalho.

O diagrama a seguir é um exemplo de um supervisor:



1. Um usuário envia uma tarefa para um agente supervisor do Amazon Bedrock.
2. O agente supervisor analisa a solicitação em subtarefas para cada agente colaborador.
3. Cada subtarefa é atribuída a um agente colaborador com solicitações ou conjuntos de ferramentas específicos da função.
4. Os agentes de trabalho chamam APIs ou ferramentas externas por meio de um grupo de ação.
5. Cada agente de trabalho retorna a saída em um formato estruturado.

6. Quando todos os trabalhadores retornam seus resultados, o supervisor avalia, sintetiza e retorna a resposta final.

Essa estrutura permite modularidade, adaptabilidade e introspecção em fluxos de trabalho complexos de agentes de várias etapas.

Takeaways

Onde a orquestração de eventos usa controle centralizado (por exemplo, AWS Step Functions) para direcionar a execução do serviço, os sistemas de agentes baseados em funções usam um agente LLM-powered orquestrador para raciocinar sobre a meta, delegar subtarefas aos agentes de trabalho e sintetizar a saída final.

Em ambos os paradigmas, o orquestrador faz o seguinte:

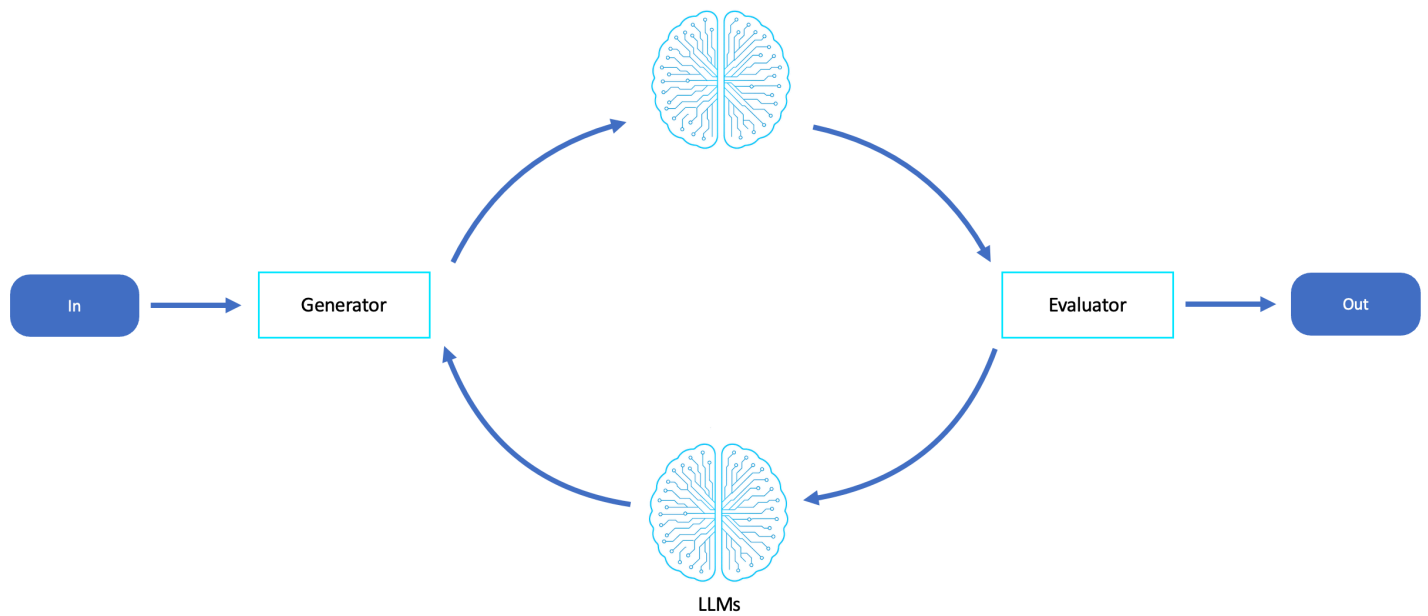
- Mantém o contexto e o fluxo de execução
- Lida com ramificação, sequenciamento e tratamento de erros
- Produz um resultado unificado a partir de componentes distribuídos

A orquestração agente, no entanto, acrescenta raciocínio, adaptabilidade e delegação semântica. Isso o torna adequado para tarefas abertas, ambíguas e em evolução.

O avaliador reflete e refina os padrões de loop

Tarefas como geração de código, resumo ou tomada de decisão autônoma se beneficiam muito do feedback de tempo de execução, permitindo que o sistema evolua por meio de observação e refinamento. Para operacionalizar isso, o ciclo reflect-refine pode ser implementado como um ciclo de controle de feedback orientado por eventos — um padrão inspirado na engenharia de sistemas, adaptado para fluxos de trabalho autônomos e inteligentes.

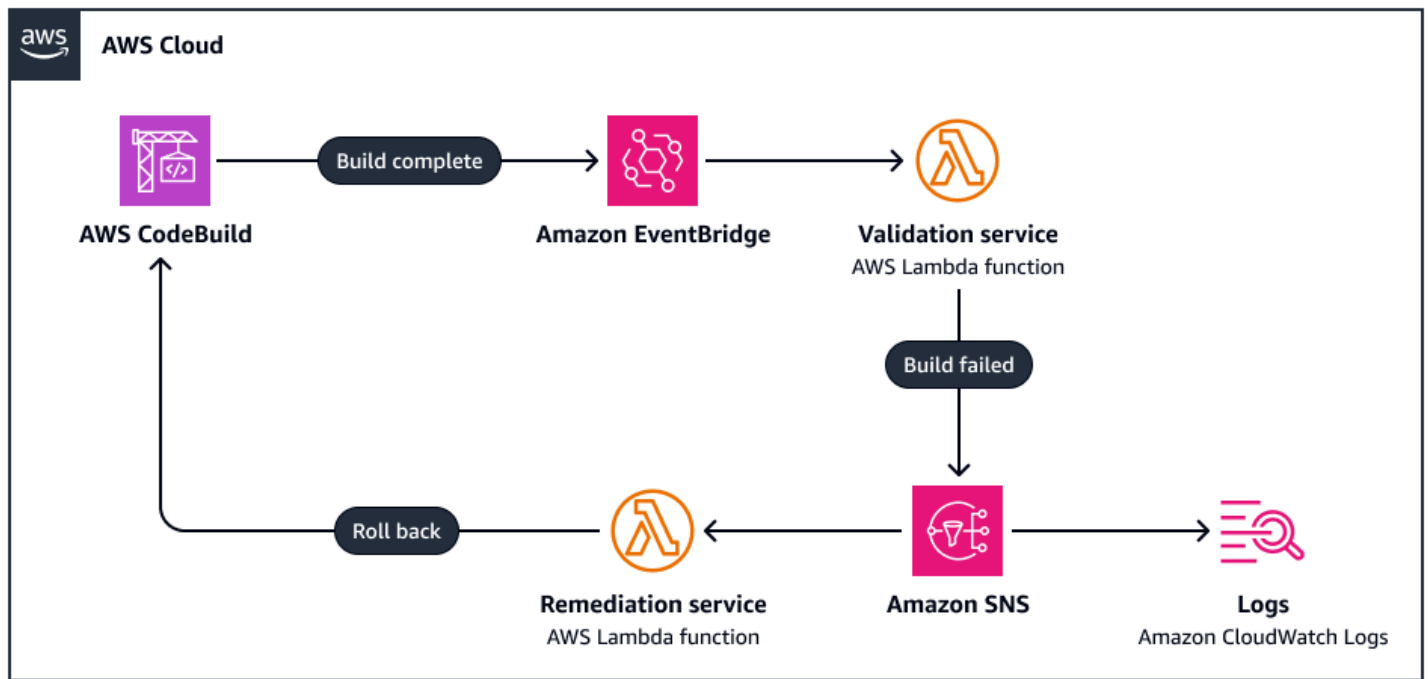
O diagrama a seguir é um exemplo de um ciclo de feedback de reflexão e refinamento do avaliador:



Circuito de controle de feedback

Um loop de controle de feedback é um padrão que monitora suas próprias saídas e comportamentos, os avalia em relação a critérios definidos ou a um estado desejado e, em seguida, ajusta suas ações de acordo. Essa arquitetura é inspirada na teoria de controle e é fundamental em domínios como automação, integração contínua e pipelines de entrega contínua (CI/CD) e operações de aprendizado de máquina.

O diagrama a seguir é um exemplo de um circuito de controle de feedback:



1. Um pipeline de implantação emite um evento BuildComplete.
2. O evento aciona um teste automatizado ou um trabalho de avaliação que valida a compilação.
3. Se a validação falhar (por exemplo, devido a falhas nos testes, problemas de segurança ou violação da política), o sistema:
 - Emite um evento BuildComplete
 - Registra o problema ou envia uma notificação
 - Aciona uma ação corretiva ou de remediação, como reversão, aplicação de patches ou nova tentativa

O ciclo continua até produzir um resultado ou escalonamento aceitável, ou até que ocorra um tempo limite. Esse padrão é comumente usado para o seguinte:

- EventBridge Regras da Amazon para encaminhar eventos para tarefas de avaliação ou remediação
- AWS Step Functions para lógica de repetição iterativa e ramificação dos resultados da avaliação
- Amazon Simple Notification Service (Amazon SNS) ou alarmes da Amazon para CloudWatch acionadores e alertas de feedback
- AWS Lambda funções ou trabalhadores em contêineres para aplicar ações corretivas

Circuito de controle de feedback (avaliador)

O fluxo de trabalho do avaliador é um ciclo de feedback cognitivo desenvolvido por LLMs ou agentes de raciocínio. O processo consiste no seguinte:

1. Um agente gerador ou LLM produz uma saída (por exemplo, um plano, uma resposta ou um rascunho).
2. Um agente avaliador analisa o resultado usando um aviso de crítica ou uma rubrica de avaliação.
3. Com base no feedback, o agente original ou um novo agente otimizador revisa a saída.

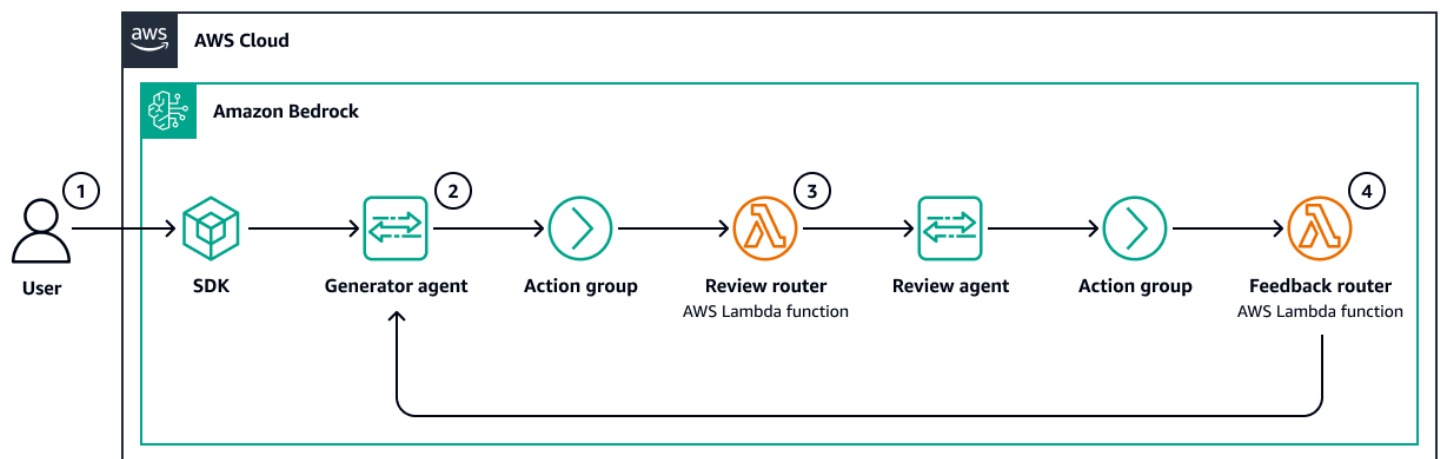
O loop se repete até que o resultado atenda a um conjunto de critérios, seja aprovado ou atinja um limite de repetição.

Avaliador

1. Um usuário pede que um agente escreva um resumo da política.
2. O agente gerador o desenha.
3. Um agente avaliador verifica a cobertura, o tom e a correção legal.
4. Se a resposta for inadequada, ela será refinada e reenviada até que o ciclo de feedback converja.

Isso permite a autoavaliação, o refinamento iterativo e o controle de saída adaptativo, tudo sem a intervenção humana.

O diagrama a seguir é um exemplo de um circuito de controle de feedback (avaliador):



1. Um usuário emite uma tarefa (por exemplo, elabore uma estratégia de negócios).

2. Um agente do Amazon Bedrock gera um rascunho inicial usando um LLM.
3. Um segundo agente (ou uma solicitação de acompanhamento) realiza uma avaliação estruturada (por exemplo, “classifique essa saída por clareza, integridade e tom”).
4. Se a classificação cair abaixo de um limite, a resposta será revisada por:
 - Reinvocando o gerador com uma crítica incorporada
 - Enviando o feedback para um agente refinador especializado
 - Iterando até que uma resposta aceitável seja alcançada

Componentes opcionais, como AWS Lambda controladores, AWS Step Functions podem gerenciar limites de feedback, novas tentativas e estratégias alternativas.

Takeaways

Enquanto os ciclos tradicionais de controle de feedback usam eventos, métricas e lógica de remediação para validar e ajustar o comportamento do sistema, os loops avaliadores agentes usam agentes de raciocínio para avaliar, refletir e revisar a saída dinamicamente.

Em ambos os paradigmas:

- A saída é avaliada após ser gerada
- Ações corretivas ou de refinamento são acionadas com base no feedback
- O sistema se adapta continuamente a uma meta ou qualidade alvo

A versão agente transforma a validação estática em reflexão semântica, permitindo que agentes de autoaperfeiçoamento avaliem sua própria eficácia.

Projetando fluxos de trabalho agentes em AWS

Cada padrão neste guia pode ser criado usando Serviços da AWS. Os agentes do Amazon Bedrock fornecem canais de orquestração, acesso a dados e interação.

Componente	AWS service (Serviço da AWS)	Finalidade
Raciocínio LLM	Amazon Bedrock	Lógica do agente, planejamento, uso de ferramentas

Execução da ferramenta	AWS Lambda, Amazon ECS, Amazon SageMaker	Hospede ferramentas externas para agentes
Memória e RAG	Base de conhecimento do Amazon Bedrock, Amazon S3, OpenSearch	Memória persistente e semântica
Orquestração	AWS Step Functions	Coordenação de tarefas e agentes em várias etapas
Roteamento de eventos	Amazon EventBridge, Amazon SQS	Mensagens interagentes desacopladas
Interface do usuário	Amazon API Gateway AWS AppSync, SDK	Pontos de entrada para aplicativos ou sistemas
Monitoramento	Amazon CloudWatch AWS X-Ray, AWS Distro para OpenTelemetry	Observabilidade e introspecção do agente

Conclusão

Os padrões de fluxo de trabalho agênticos são o próximo estágio evolutivo das arquiteturas orientadas a eventos, em que a lógica de negócios não é definida estaticamente, mas fundamentada dinamicamente por meio do uso de cognição aprimorada por meio do modelo de linguagem grande (LLM). Ao combinar primitivos nativos da nuvem tradicionais com fluxos de trabalho de LLM e padrões de design de agentes, as organizações podem criar sistemas adaptáveis, inteligentes e modulares que respondam com propósito e aprendam com a experiência.

Nesses padrões, o Amazon Bedrock é a porta de entrada para a cognição agente, permitindo que LLM-based os agentes acessem fluxos de trabalho de eventos, interajam com ferramentas e memória e forneçam resultados estruturados, rastreáveis e alinhados.

À medida que você projeta e implanta sistemas agentes, esses padrões de fluxo de trabalho fornecem planos para a criação de arquiteturas de IA autônomas e combináveis. Esses sistemas são baseados nas melhores práticas sem servidor e aumentados com modelos básicos inteligentes.

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	—	14 de julho de 2025

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.