Modèles et flux de travail d'IA agentic sur AWS

AWS Conseils prescriptifs

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

AWS Conseils prescriptifs: Modèles et flux de travail d'IA agentic sur AWS

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

Les marques et la présentation commerciale d’Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n’est pas d’Amazon, d’une manière susceptible de créer une confusion parmi les clients, ou d’une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Introduction 1
Public visé 1
Objectifs 1
À propos de cette série de contenus 2
Modèles d'agents 3
Agents de raisonnement de base 4
Architecture 4
Description 5
Fonctionnalités 6
Limitations 6
Cas d’utilisation courants 6
Directives d’implémentation 7
Résumé 7
Architecture 7
Description 8
Fonctionnalités 9
Cas d’utilisation courants 9
Directives d’implémentation 9
Résumé 10
Tool-based agents pour les fonctions d'appel 10
Architecture 10
Description 11
Fonctionnalités 12
Cas d’utilisation courants 12
Directives d’implémentation 12
Résumé 13
Tool-based agents pour serveurs 13
Architecture 13
Description 14
Fonctionnalités 15
Cas d’utilisation courants 15
Directives d’implémentation 15
Résumé 16
Computer-use agents 16

Architecture 16
Description 17
Fonctionnalités 18
Cas d’utilisation courants 18
Directives d’implémentation 19
Résumé 19
Agents de codage 19
Architecture 19
Description 20
Fonctionnalités 21
Cas d’utilisation courants 21
Directives d’implémentation 22
Résumé 22
Agents vocaux et vocaux 22
Architecture 22
Description 23
Fonctionnalités 24
Cas d’utilisation courants 24
Directives d’implémentation 25
Résumé 25
Agents d'orchestration des flux de travail 25
Architecture 26
Description 26
Fonctionnalités 27
Cas d’utilisation courants 27
Directives d’implémentation 27
Résumé 28
Memory-augmented agents 28
Architecture 28
Description 29
Fonctionnalités 30
Cas d’utilisation courants 30
Implémentation d'agents à mémoire augmentée 30
Implémentation de l'envoi de messages par injection de mémoire 31
Résumé 32
Agents de simulation et de banc d'essai 32

Architecture 32
Description 33
Fonctionnalités 34
Cas d’utilisation courants 34
Directives d’implémentation 35
Résumé 36
Observateurs et agents de surveillance 36
Architecture 37
Description 37
Fonctionnalités 38
Cas d’utilisation courants 38
Directives d’implémentation 39
Résumé 39
Multi-agent collaboration 40
Description 41
Fonctionnalités 42
Cas d’utilisation courants 42
Directives d’implémentation 42
Résumé 43
Conclusion 43
Plats à emporter 44
Flux de travail LLM 45
Vue d'ensemble de LLM-augmented la cognition 45
Flux de travail pour un chaînage rapide 46
Description 47
Fonctionnalités 48
Cas d’utilisation courants 48
Flux de travail pour le routage 48
Fonctionnalités 50
Cas d’utilisation courants 50
Workflow pour la parallélisation 50
Fonctionnalités 51
Cas d’utilisation courants 51
Flux de travail pour l'orchestration 52
Fonctionnalités 53
Cas d’utilisation courants 53

Flux de travail pour les évaluateurs et boucles d'affinage par réflexion 53
Cas d’utilisation courants 55
Fonctionnalités 55
Conclusion 55
Modèles de flux de travail agentic 57
Des systèmes pilotés par les événements aux systèmes augmentés par la cognition 57
Event-driven architecture 58
Cognition-augmented flux de travail 59
Informations de base 60
Motifs de saga à enchaînement rapide 60
Chorégraphie de saga 61
Schéma de chaînage rapide 62
Chorégraphie d'agent 62
Plats à emporter 64
Acheminement des modèles d'expédition dynamiques 64
Expédition dynamique 65
LLM-based routage 66
Routeur agent 67
Plats à emporter 68
Modèles de parallélisation et de diffusion 68
Scatter-gather 69
LLM-based parallélisation (cognition basée sur la diffusion et la collecte) 71
Parallélisation des agents 71
Plats à emporter 72
Modèles d'orchestration de Saga 72
Orchestration d'événements 73
Role-based système d'agents (orchestrateur) 74
Superviseur 74
Plats à emporter 76
L'évaluateur reflète et affine les modèles de boucles 76
Boucle de régulation du feedback 77
Boucle de régulation du feedback (évaluateur) 79
Évaluateur 79
Plats à emporter 80
Conception de flux de travail agentiques sur AWS 81
Conclusion 81

Historique du document 83
..... lxxxiv

Modèles et flux de travail d'IA agentic sur AWS

Aaron Sempf et Andrew Hooker, Amazon Web Services

Juillet 2025 (historique du document)

Organisations adoptent de grands modèles linguistiques (LLMs) et des agents logiciels pour résoudre des problèmes multidomaines dynamiques en utilisant une nouvelle discipline architecturale appelée modèles agentiques. Les modèles agentiques sont des modèles fondamentaux et des structures modulaires utilisés pour concevoir et orchestrer des agents d'IA orientés vers des objectifs dans de nombreux contextes.

Public visé

Ce guide est destiné aux architectes, aux développeurs et aux chefs de produit qui souhaitent créer des applications intelligentes allant au-delà de la logique statique, de la logique symbolique et de l'automatisation déterministe.

Objectifs

Ce guide fournit un cadre de conception et une approche de mise en œuvre pour les systèmes d'agents d'IA qui fonctionnent de manière autonome tout en restant contrôlables et alignés sur vos objectifs. Il relie les modèles architecturaux pilotés par les événements à diverses alternatives agentiques, démontrant ainsi comment créer des systèmes d'agents adaptés à la production à l'aide d'architectures cloud natives. Les sujets suivants sont abordés dans ce guide :

- Modèles d'agents : les modèles d'agents sont des modèles de conception réutilisables qui décrivent la structure et le comportement des agents individuels. Cela inclut les agents de raisonnement, les agents de récupération augmentée, les agents de codage, les interfaces vocales, les orchestrateurs de flux de travail et les systèmes multi-agents collaboratifs. Chaque modèle illustre la façon dont les agents perçoivent, raisonnent, agissent et apprennent, selon Services AWS une cartographie.
- Workflows LLM — Les flux de travail se concentrent sur la façon dont les agents les utilisent LLMs pour raisonner. Ils explorent les stratégies d'incitation et les mécanismes de planification, et décrivent comment ils LLMs sont utilisés non seulement pour générer du texte, mais également pour générer des comportements structurés, interprétables et fiables au sein d'une boucle d'agents.

- Modèles de flux de travail agentiques — Les modèles de flux de travail décrivent comment plusieurs agents, outils et environnements interagissent pour former des systèmes autonomes. Cela inclut les modèles d'orchestration des tâches, de délégation de sous-agents, de coordination basée sur les événements, d'observabilité et de contrôle. Ces aspects favorisent des architectures d'IA évolutives, composables et auditables.

À propos de cette série de contenus

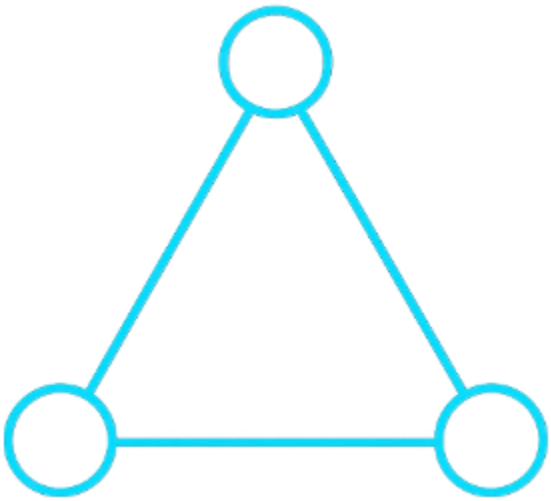
Ce guide fait partie d'une série sur l'IA agentique sur. AWS Pour plus d'informations et pour consulter les autres guides de cette série, consultez Agentic AI sur le site Web de AWS Prescriptive Guidance.

Modèles d'agents

Les modèles d'agents sont des éléments de base réutilisables et composables qui peuvent être adaptés à des domaines, à des cas d'utilisation et à des niveaux de complexité spécifiques.
Les systèmes Agentic diffèrent toutefois des applications traditionnelles. Au cœur de toutes les conceptions d'agents d'IA se trouve un modèle conceptuel fondé sur les trois principes fondamentaux suivants :

- Asynchrone : les agents fonctionnent dans des environnements faiblement couplés et riches en événements
- Autonomie — Les agents agissent de manière indépendante, sans contrôle humain ou externe
- Agence — Les agents agissent dans un but précis, au nom d'un utilisateur ou d'un système, dans le but d'atteindre des objectifs spécifiques

Le triangle du schéma suivant représente les éléments de base d'un agent logiciel : perception, raison et action. Cela permet à un système agentique d'observer, de prendre des décisions et d'agir dans son environnement.



De par leur conception, les modèles agentiques fournissent un langage de conception modulaire pour la création de systèmes d'IA, ce qui signifie qu'ils sont accessibles, opérationnels, extensibles et prêts pour la production. La conception de ces systèmes nécessite une attention particulière aux trois dimensions interdépendantes suivantes, qui sont abordées plus loin dans ce guide.

Dans cette section

- Agents de raisonnement de base
- Tool-based agents pour les fonctions d'appel
- Tool-based agents pour serveurs
- Computer-use agents
- Agents de codage
- Agents vocaux et vocaux
- Agents d'orchestration des flux de travail
- Memory-augmented agents
- Agents de simulation et de banc d'essai
- Observateurs et agents de surveillance
- Multi-agent collaboration

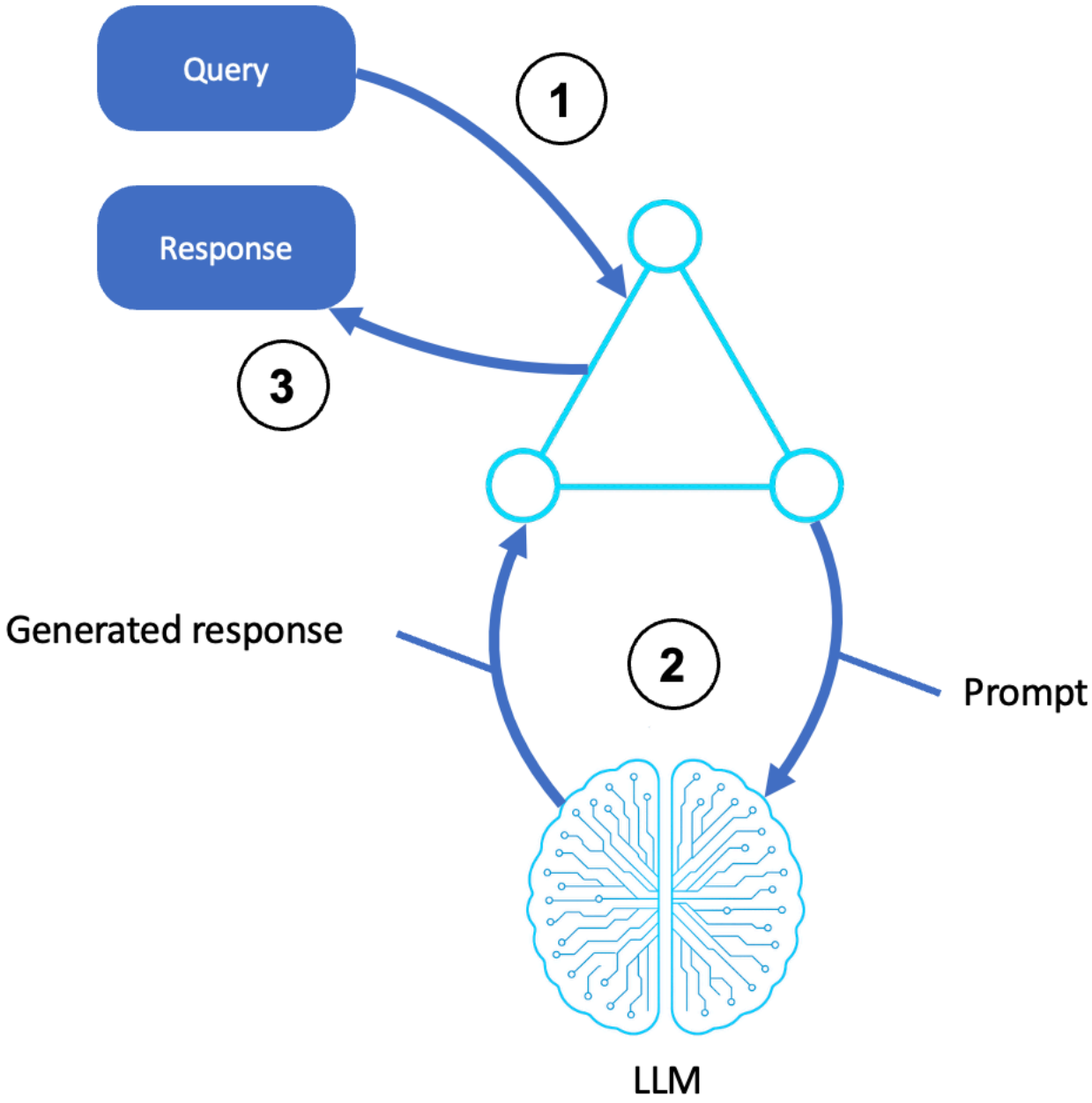
Agents de raisonnement de base

Un agent de raisonnement de base est la forme la plus simple d'IA agentique qui effectue une inférence logique ou prend des décisions en réponse à une requête. Il accepte les entrées d'un utilisateur ou d'un système, traite les requêtes et génère des réponses à l'aide d'instructions structurées.

Ce modèle est utile pour les tâches qui nécessitent un raisonnement, une classification ou une synthèse en une seule étape en fonction d'un contexte donné. Il n'utilise pas de mémoire, d'outils ou de gestion d'état, ce qui le rend apatride, léger et hautement composable dans les grands flux de travail.

Architecture

Le flux d'un agent de raisonnement de base est illustré dans le schéma suivant :



Description

1. Reçoit une entrée
   - Un utilisateur, un système ou un agent en amont soumet une requête ou une instruction.
   - L'entrée est transmise au shell de l'agent ou à la couche d'orchestration.
   - Cette étape inclut tout prétraitement, la création de modèles d'invite et l'identification des objectifs.

2. Invoque le LLM
   - L'agent transforme la requête en une invite structurée et l'envoie à un LLM (par exemple, via Amazon Bedrock).
   - Le LLM génère une réponse basée sur l'invite en utilisant des connaissances et un contexte préformés.
   - Le résultat généré peut inclure des étapes de raisonnement (chaîne de pensée), des réponses finales ou des options classées.
3. Renvoie une réponse
   - La sortie générée est transmise à l'interface de l'agent.
   - Cela peut inclure le formatage, le post-traitement ou une réponse d'API.

Fonctionnalités

- Supporte le langage naturel ou la saisie structurée
- Utilise une ingénierie rapide pour guider le comportement
- Apatride et évolutif
- Peut être intégré dans l'interface utilisateur, la CLI, les API et les pipelines

Limitations

- Absence de mémoire ou de conscience historique
- Aucune interaction avec des outils ou des sources de données externes
- Limité à ce que le LLM sait au moment de l'inférence

Cas d’utilisation courants

- Questions et réponses conversationnelles
- Explications et résumés des politiques
- Conseils pour la prise de décisions
- Des flux de chatbot légers et automatisés
- Classification, étiquetage et notation

Directives d’implémentation

Vous pouvez utiliser les outils et services suivants pour créer un agent de raisonnement de base :

- Amazon Bedrock pour l'invocation du LLM (Anthropic, AI21, Meta)
- Amazon API Gateway ou AWS Lambda pour l'exposer en tant que microservice apatride
- Modèles d'invite stockés dans le Parameter Store ou sous forme de code AWS Secrets Manager

Résumé

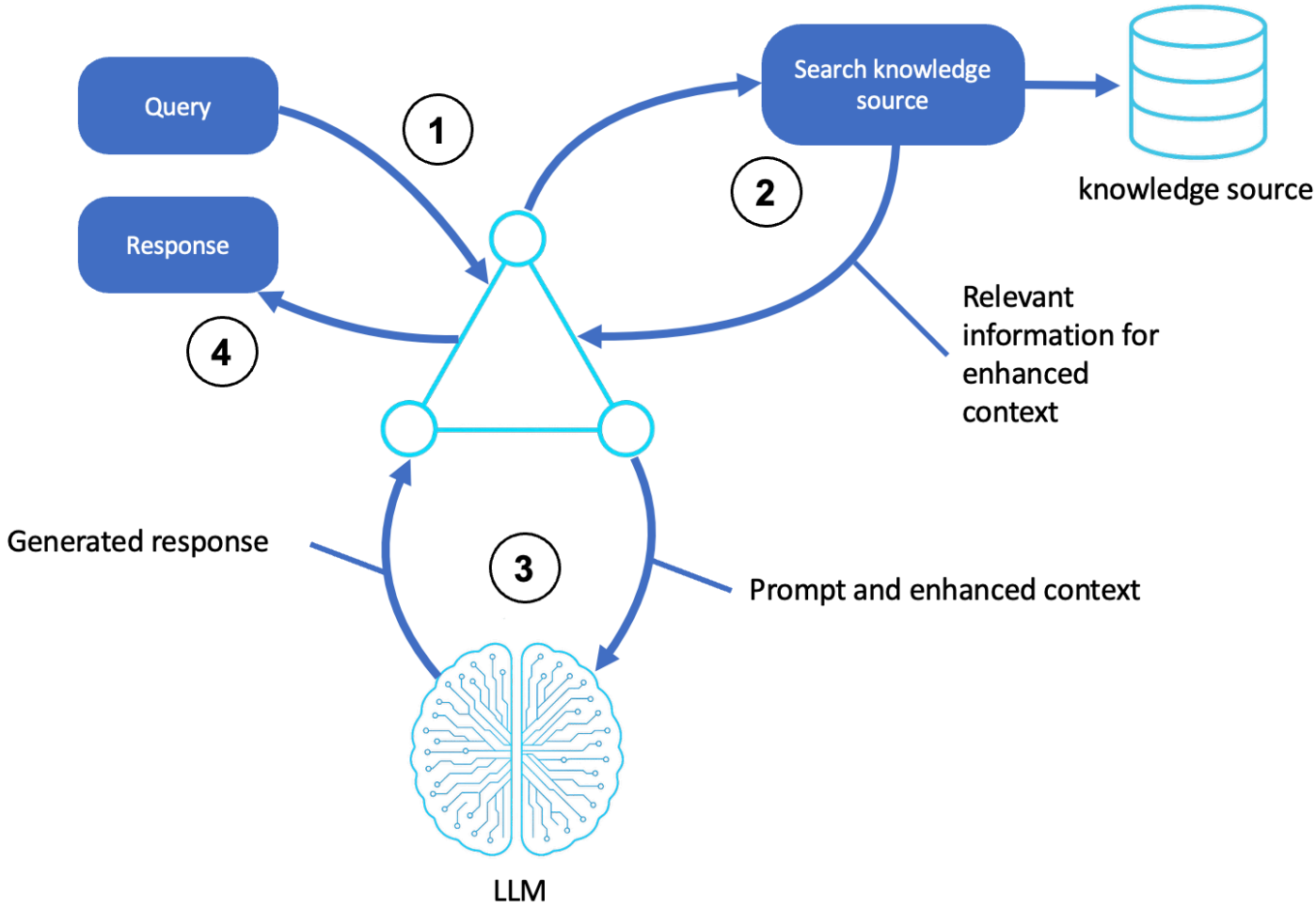
L'agent de raisonnement de base est fondamental en raison de sa structure simple. Il possède des capacités de base qui transforment les objectifs en voies de raisonnement menant à des résultats intelligents. Ce modèle est souvent le point de départ de modèles avancés, tels que les agents basés sur des outils et les agents utilisant la génération augmentée par récupération (RAG). C'est également un composant fiable et modulaire des grands flux de travail.

Agent RAG

Retrieval-augmented La génération (RAG) est une technique qui combine la récupération d'informations avec la génération de texte pour créer des réponses précises et contextuelles. RAG permet aux agents de récupérer des informations externes pertinentes avant d'engager le LLM. Il étend la mémoire effective et la précision du raisonnement d'un agent en fondant ses décisions sur des informations actualisées, factuelles ou spécifiques à un domaine. Contrairement aux LLM apatrides qui s'appuient uniquement sur des poids prédéterminés, RAG dispose d'une couche de recherche de connaissances externe qui améliore dynamiquement les instructions en fonction du contexte.

Architecture

La logique du modèle RAG est illustrée dans le schéma suivant :



Description

1. Reçoit une requête
   - Un utilisateur ou un système en amont soumet une requête ou un objectif à l'agent.
   - Le shell de l'agent accepte la demande et la met en forme comme une invite à raisonner.
2. Recherche dans une source externe
   - L'agent identifie les concepts et l'intention à partir de la requête.
   - Il interroge une source de connaissances, telle qu'un magasin de vecteurs, une base de données ou un index de documents, à l'aide d'une recherche sémantique ou d'une correspondance de mots clés.
   - Les passages, documents ou entités les plus pertinents sont récupérés pour être utilisés à l'étape suivante.
3. Génère une réponse contextuelle

- L'agent complète l'invite avec les informations récupérées, formant ainsi une entrée contextuelle améliorée pour le LLM.
- Le LLM traite toutes les entrées en utilisant un raisonnement génératif (par exemple, chaîne de pensée ou réflexion) pour produire une réponse précise.

4. Renvoie le résultat final
   - L'agent prépare le résultat en l'encapsulant dans n'importe quel en-tête de communication ou dans le formatage requis, puis le renvoie à l'utilisateur ou au système d'appel.
   - (Facultatif) Les documents récupérés et la sortie LLM peuvent être enregistrés, notés et stockés en mémoire pour de futures requêtes.

Fonctionnalités

- Fact-grounded production même dans des domaines à long terme ou spécifiques à l'entreprise
- Extension de mémoire sans réglage précis du modèle
- Contexte dynamique basé sur chaque requête et l'état de l'utilisateur
- Entièrement compatible avec les bases de données vectorielles, les index sémantiques et le filtrage des métadonnées

Cas d’utilisation courants

- Assistants du savoir d'entreprise
- Bots de conformité réglementaire
- Copilotes du support client
- Search-enhanced chatbots
- Agents de documentation pour développeurs

Directives d’implémentation

Utilisez les outils et services suivants pour créer un agent utilisant RAG :

- Amazon Bedrock pour l'invocation d'un LLM
- Amazon Kendra ou Amazon Aurora pour la documentation ou une recherche de données structurées OpenSearch

- Amazon Simple Storage Service (Amazon S3) pour le stockage de documents
- AWS Lambda pour orchestrer la recherche, l'invite et l'inférence LLM
- Knowledge-based intégrations avec des agents (en utilisant des plugins de mémoire, des récupérateurs sémantiques ou Amazon Bedrock)

Résumé

L'agent RAG associe le raisonnement du modèle statique à l'intelligence dynamique du monde réel. Il permet aux agents de rechercher ce qu'ils ne savent pas, de synthétiser les réponses à partir des connaissances récupérées et de produire des réponses vérifiables et fiables.

Les modèles RAG constituent la base de la création d'agents intelligents capables d'étendre l'accès aux connaissances sans avoir à se recycler. Il est souvent le précurseur de modèles d'orchestration plus complexes impliquant l'utilisation d'outils, la planification et la mémoire à long terme.

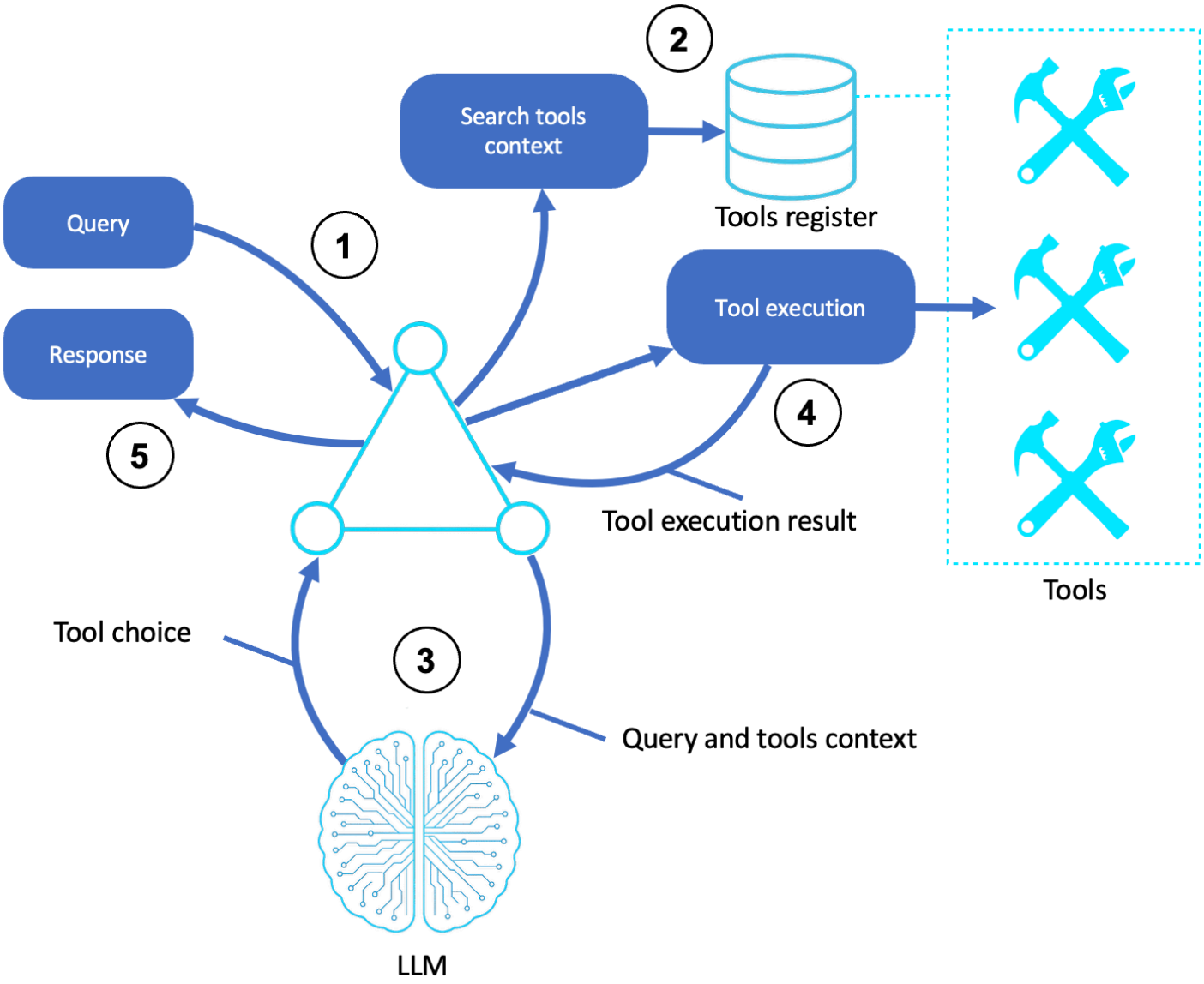
Tool-based agents pour les fonctions d'appel

Tool-based les agents étendent les capacités des agents de raisonnement en invoquant des fonctions externes ou des API pour effectuer des tâches qui vont au-delà du simple raisonnement linguistique. Ce modèle utilise un LLM pour décider de l'outil à utiliser, puis génère des arguments d'appel et intègre le résultat d'un outil dans sa boucle de raisonnement.

Ce modèle permet aux agents d'agir au lieu de simplement fournir des réponses. L'interface de l'outil représente toutes les capacités appelables, qu'il s'agisse de calculs arithmétiques, de recherches de bases de données, d'API externes ou de services cloud.

Architecture

Un agent basé sur un outil permettant d'appeler des fonctions est illustré dans le schéma suivant :



Description

1. Reçoit une requête
   - L'agent reçoit une requête ou une tâche en langage naturel de la part de l'utilisateur ou du système d'appel.
2. Recherches d'outils
   - L'agent utilise des métadonnées internes ou un registre d'outils pour rechercher les outils, les schémas et les fonctionnalités pertinentes disponibles.
3. Sélectionne et invoque des outils

- Le LLM reçoit les métadonnées de la requête et de l'outil (par exemple, les noms des fonctions, les types d'entrée et les descriptions) dans son invite.
- Il choisit l'outil le plus pertinent, construit des arguments d'entrée et renvoie un appel de fonction structuré.

4. Exécute l'outil choisi
   - Le shell de l'agent ou le lanceur d'outils exécute la fonction sélectionnée et renvoie le résultat (par exemple, une sortie d'API, une valeur de base de données ou un calcul).
5. Renvoie une réponse
   - Le LLM transmet les résultats à l'agent, soit directement, soit dans le cadre d'une invite mise à jour. Il renvoie ensuite un résultat en langage naturel.

Fonctionnalités

- Sélection d'outils dynamique en fonction du contexte de la tâche
- Schema-based invite (OpenAPI, schéma AWS JSON, interface de fonction)
- Interprétation des résultats et intégration des résultats dans le raisonnement
- Opérations apatrides ou liées à une session

Cas d’utilisation courants

- Assistants virtuels avec accès aux données externes
- Calculateurs et estimateurs financiers
- API-based travailleurs du savoir
- LLM qui invoquent AWS Lambda, SageMaker points de terminaison Amazon et services SaaS

Directives d’implémentation

Utilisez ce qui suit pour créer des agents basés sur des outils pour appeler des fonctions :

- Amazon Bedrock avec assistance pour les appels fonctionnels (Anthropic Claude)
- AWS Lambda en tant que backend d'exécution d'outils
- Amazon API Gateway ou AWS Step Functions pour l'orchestration d'outils

- Amazon DynamoDB ou Amazon Relational Database Service (Amazon RDS) pour les métadonnées des outils sensibles au contexte
- Amazon EventBridge Pipelines ou AWS Step Functions qui mappent les états pour acheminer les sorties

Résumé

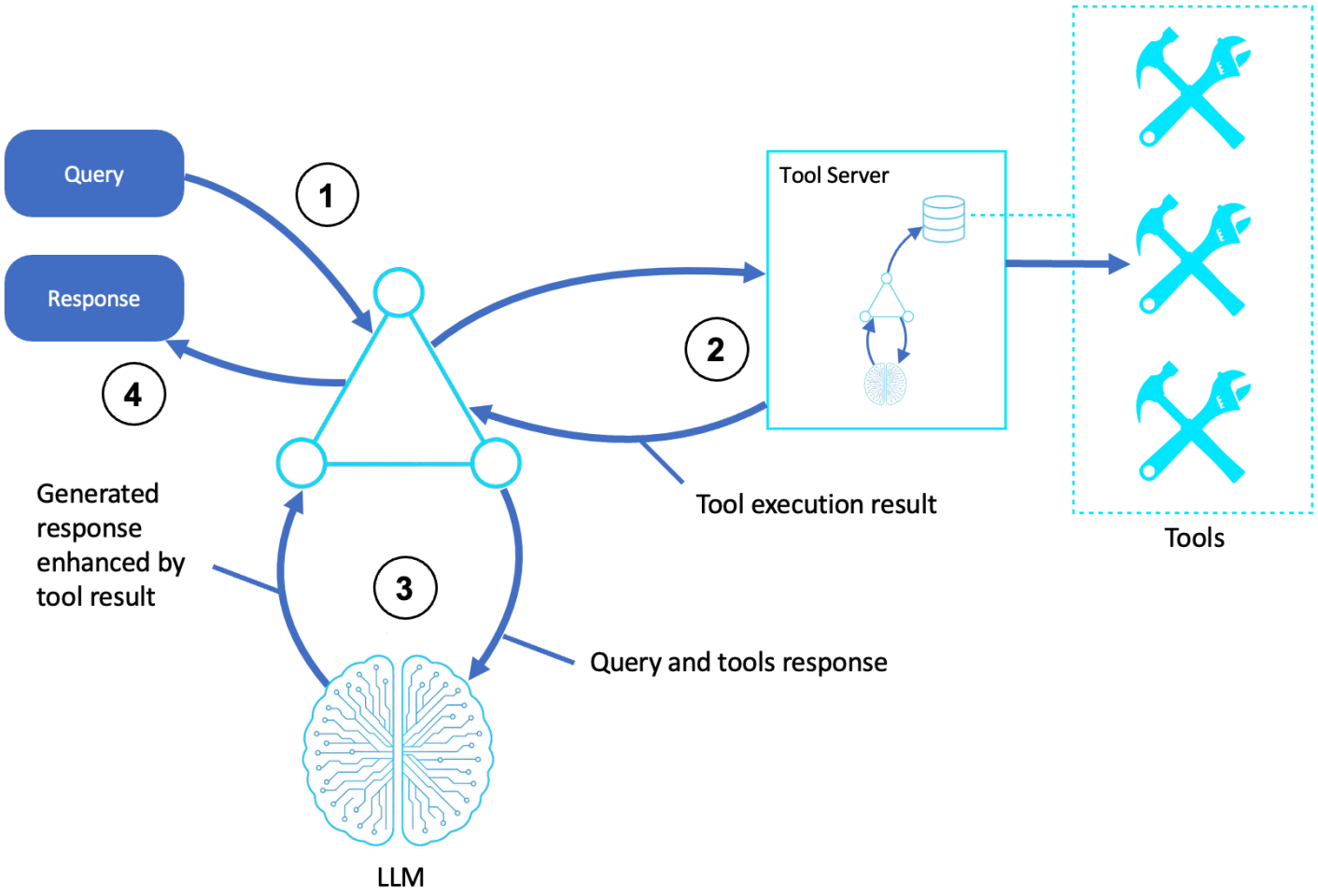
Tool-based les agents d'appel de fonctions représentent un passage de la compréhension du langage à l'exécution d'actions. Ces agents utilisent des outils dynamiques sensibles au contexte tout en maintenant le raisonnement LLM, transformant les assistants passifs en systèmes qui exécutent des tâches, accèdent aux services et intègrent les opérations commerciales. Ce modèle est un élément important de l'IA agentique dans les environnements d'entreprise, en particulier lorsqu'il est associé à des schémas déclaratifs, à des cadres d'autorisation et à des systèmes multi-agents.

Tool-based agents pour serveurs

Tool-based les agents pour serveurs améliorent les agents qui appellent des fonctions en déléguant l'exécution des outils à un serveur externe doté d'un environnement d'exécution dédié aux outils, aux scripts et aux agents composites. Contrairement aux appels de fonctions en ligne que la boucle d'agents sélectionne et invoque, les agents basés sur le serveur externalisent la logique et le pipeline d'exécution à d'autres agents ou systèmes. Cela fournit des fonctionnalités avancées telles que le chaînage multioutils, l'exécution isolée et le raisonnement spécialisé. Les serveurs d'outils sont idéaux pour les actions complexes, dynamiques ou gourmandes en ressources lorsque les outils eux-mêmes peuvent impliquer des modèles d'IA, des règles métier ou des environnements distincts.

Architecture

Voici un modèle d'agents basés sur des outils pour serveurs :



Description

1. Reçoit une requête
   - Un utilisateur ou un système envoie une demande à l'agent shell.
   - L'agent interprète la requête et prépare son envoi à un serveur d'outils.
2. Exécute les processus du serveur d'outils
   - L'agent envoie la tâche, ainsi que les paramètres structurés, à un serveur d'outils.
   - Le serveur d'outils peut alors :
     - Exécutez des scripts ou de la logique dans des systèmes informatiques dédiés (par exemple AWS Lambda, des conteneurs ou Amazon SageMaker)
     - Utiliser son propre sous-agent avec un raisonnement LLM pour sélectionner et exécuter des outils
     - Gérez les dépendances, les nouvelles tentatives ou les flux d'exécution en plusieurs étapes

    - Transmettre les résultats à l'agent principal lorsque la tâche est terminée
3. Utilise le raisonnement LLM avec les résultats de l'outil
    - L'agent invoque un LLM, en transmettant la requête d'origine et le résultat du serveur d'outils dans le cadre de l'invite.
    - Le LLM synthétise une réponse qui intègre les informations nouvellement acquises.
4. Renvoie une réponse
    - L'agent renvoie une réponse structurée ou en langage naturel à l'utilisateur ou au système d'appel.
    - (Facultatif) Les résultats peuvent être stockés dans la mémoire ou dans des journaux d'audit.

Fonctionnalités

- Les outils sont invoqués en dehors de la boucle d'exécution de l'agent principal
- L'exécution de l'outil peut impliquer des appels LLM, des chaînes logiques ou des sous-agents
- L'agent agit en tant que contrôleur ou répartiteur, et pas simplement en tant qu'emballage d'outils
- Permet la composabilité, l'évolutivité et l'isolation de la logique

Cas d’utilisation courants

- Orchestration de chaînes de modèles (par exemple, en combinant LLM, vision et code)
- AI-driven pipelines d'automatisation
- DevOps agents assistants avec exécuteurs de scripts
- Agents de calcul, de simulation ou d'optimisation financiers complexes
- Outils multimodaux (par exemple, en combinant audio, documentation et action)

Directives d’implémentation

Vous pouvez créer ce modèle en utilisant ce qui suit Services AWS :

- Amazon Bedrock (agent hôte et inférence LLM)
- AWS Lambda, Amazon ECS ou Amazon SageMaker endpoints en tant qu'environnement d'exécution du serveur d'outils AWS Fargate
- Amazon API Gateway ou AWS App Runner pour exposer les API du serveur d'outils

- Amazon EventBridge pour une messagerie agent-outil découplée
- AWS Step Functions ou AWS AppFabric pour composer une logique multi-agents sur le serveur d'outils

Résumé

Tool-based les agents qui utilisent des serveurs sont hautement modulaires et évolutifs. Ils dissocient la logique de décision de l'exécution, ce qui permet à l'agent principal de rester léger tout en déléguant des actions complexes ou sensibles à d'autres systèmes. Cela est important pour l'IA agentique de niveau entreprise, en particulier dans les environnements qui nécessitent une gouvernance, une observabilité, une isolation, une composition dynamique ou une combinaison des deux.

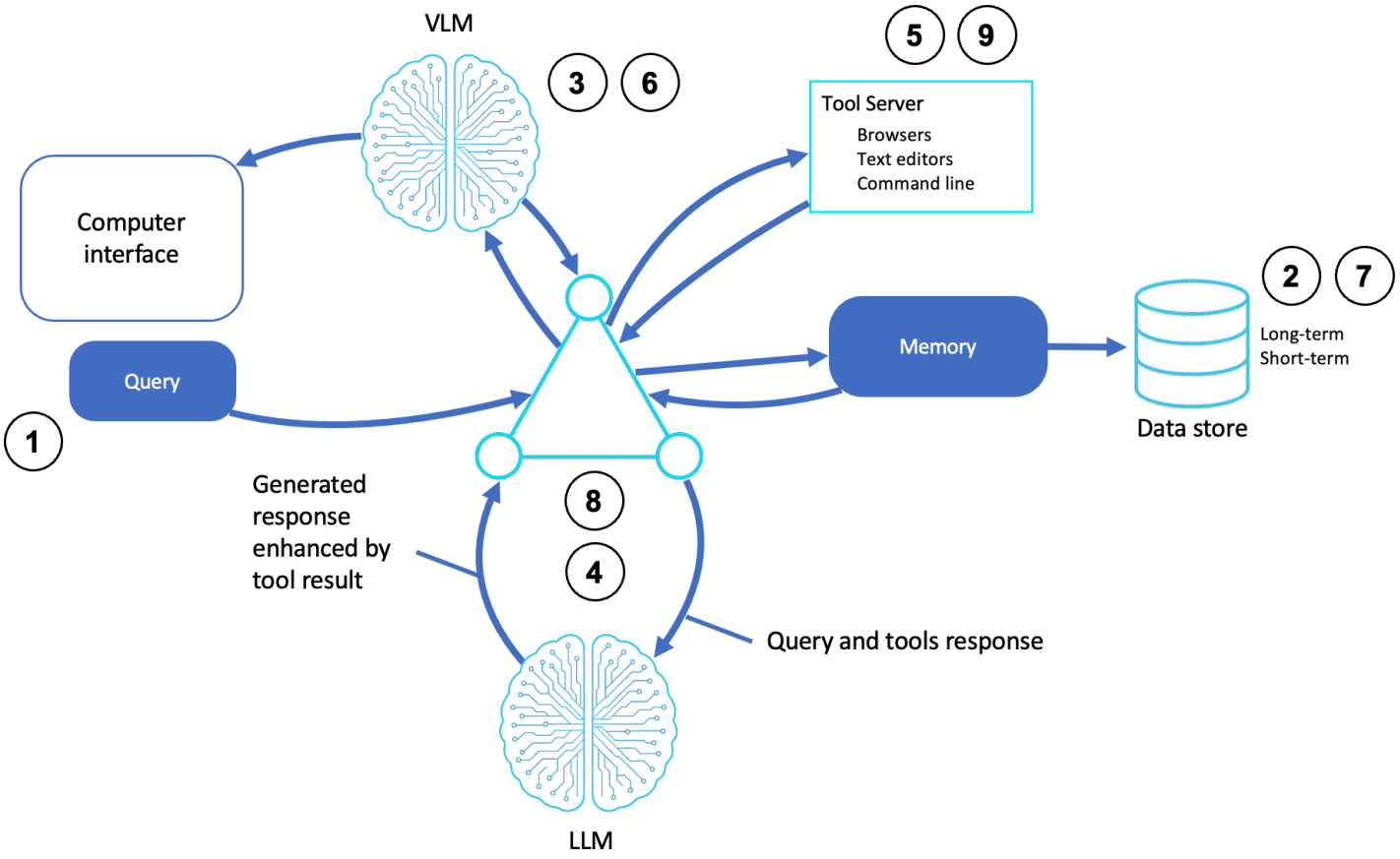
Computer-use agents

Computer-use les agents peuvent simuler ou contrôler des environnements numériques tels que les navigateurs, les terminaux, les systèmes de fichiers et les applications. Ces agents interprètent l'intention de l'utilisateur, interagissent avec les interfaces visuelles et textuelles et exécutent des actions ciblées en combinant le raisonnement LLM, les modèles de langage visuel (VLM) et les serveurs d'outils qui exécutent des commandes ou simulent des événements d'entrée.

Ce modèle est important pour les automatisations pratiques de l'IA, où l'agent fonctionne non seulement comme un assistant, mais également comme un proxy qui exécute des actions comme le ferait un humain, souvent en utilisant les mêmes outils et environnements.

Architecture

Un modèle d'agent utilisé par ordinateur est illustré dans le schéma suivant :



Description

1. Reçoit une requête
   - Une tâche ou une demande est fournie via une interface utilisateur, une API ou une interface en langage naturel.
2. Accède à la mémoire
   - L'agent récupère la mémoire à court et à long terme pour se souvenir des commandes passées, des objectifs et des états du système.
3. Analyse le contexte visuel
   - Un VLM observe l'écran de l'ordinateur, l'état du système ou les éléments de l'interface utilisateur pour comprendre un contexte donné et identifier les éléments exploitables.
4. Raisons grâce à un LLM
   - Le LLM combine la requête, l'état de la mémoire, l'outil et la réponse du serveur pour déterminer l'action suivante.
5. Interagit avec le serveur d'outils

- L'agent invoque des outils hébergés sur un serveur, notamment les suivants :
  - Navigateurs (Chrome sans en-tête, par exemple) et environnements shell
  - Éditeurs de texte et de code
  - Interfaces de script personnalisées

6. Met à jour les entrées visuelles
   - Si l'interface utilisateur du système change ou si une observation supplémentaire est nécessaire, le VLM peut réanalyser l'état de l'écran ou les tampons de texte.
7. Actualise la mémoire
   - Les nouvelles informations, l'état du système ou les commentaires des utilisateurs sont enregistrés dans la mémoire à court et à long terme.
8. Formulation des décisions finales et des explications
   - Le LLM synthétise les résultats ou recommande des actions en fonction de la requête et des résultats de l'outil.
9. Renvoie une réponse
   - L'agent renvoie les résultats à l'interface (par exemple, une tâche terminée, une confirmation ou du contenu généré).

Fonctionnalités

- Raisonnement multimodal avec entrées visuelles et textuelles
- Contrôle des applications par le biais de données simulées ou d'entrées API-driven
- Gestion de la mémoire pour un état persistant
- Autonomie dans l'exécution séquentielle (flux en plusieurs étapes)

Cas d’utilisation courants

- Développeurs d'IA qui écrivent et exécutent du code dans des IDE
- Computer-use agents pour les flux de travail numériques répétitifs
- Utilisateurs simulés pour les tests logiciels et l'assurance qualité
- Agents d'accessibilité pour naviguer dans les interfaces utilisateur par le biais d'instructions vocales ou de haut niveau
- Automatisation robotique intelligente des processus (RPA) améliorée par le raisonnement

Directives d’implémentation

- Vous pouvez créer ce modèle en utilisant ce qui suit Services AWS :
- Amazon Bedrock pour la LLM-based planification et le raisonnement
- Amazon Elastic Compute Cloud (Amazon EC2) ou ordinateurs portables SageMaker Amazon pour exécuter AWS Lambda des serveurs d'outils avec des environnements d'interface utilisateur simulés
- Amazon Simple Storage Service (Amazon S3) ou Amazon DynamoDB pour la persistance de la mémoire
- Amazon Rekognition (ou modèles personnalisés) pour l'analyse d'images d'interface utilisateur dans des scénarios hybrides
- Amazon CloudWatch Logs ou AWS X-Ray pour l'observabilité et les pistes d'audit

Résumé

Computer-use les agents agissent comme des opérateurs numériques autonomes, comblant le fossé entre les interactions homme-machine et AI-driven les actions. En incorporant de la mémoire, une orchestration d'outils et des VLM, ces agents peuvent interagir de manière adaptative avec des systèmes conçus pour les humains, exécuter des actions, mettre à jour des fichiers, naviguer dans les menus et générer des réponses.

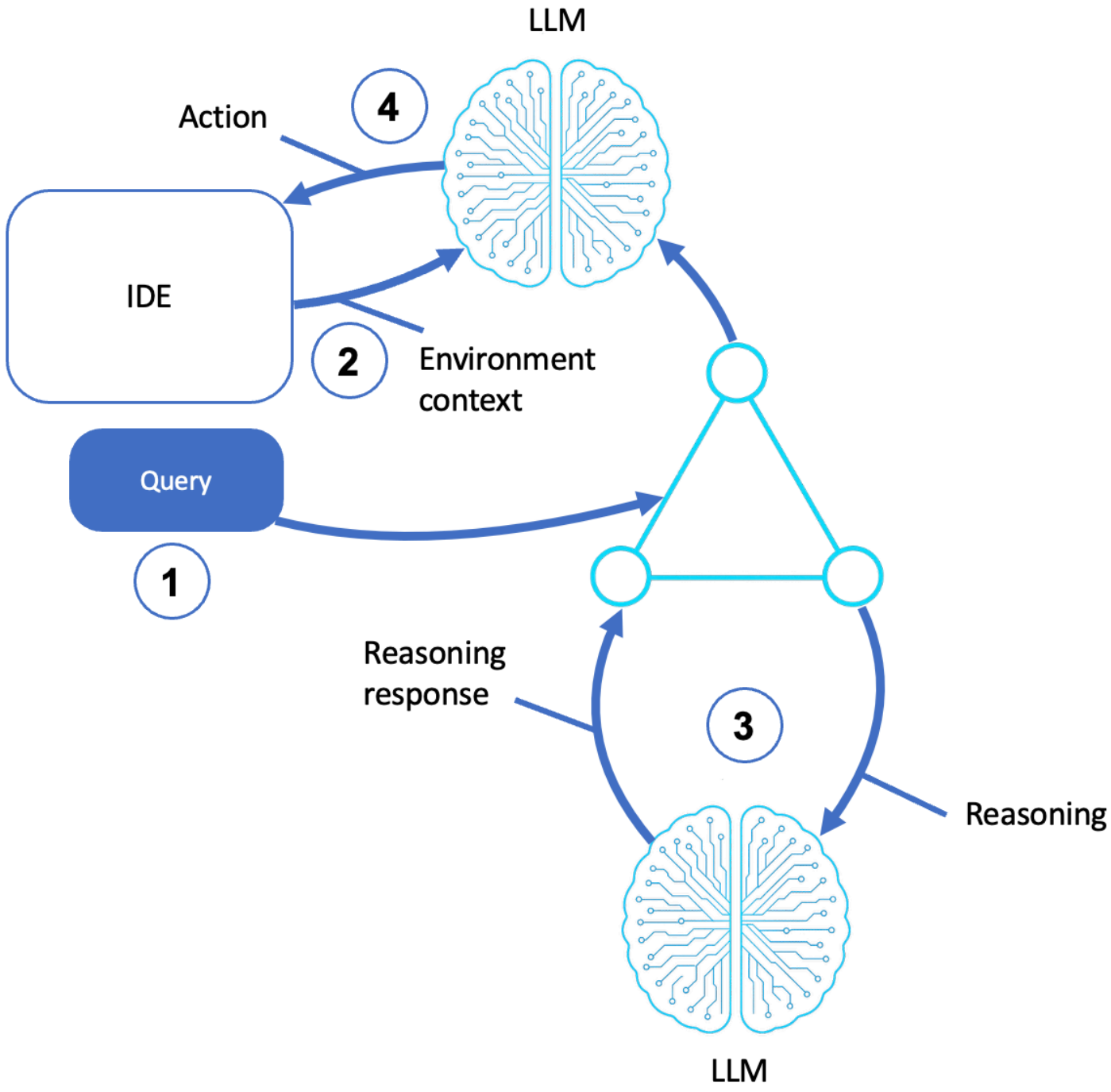
Agents de codage

Les agents de codage peuvent raisonner sur les tâches de programmation, générer ou modifier du code et interagir avec les environnements de développement, tels que les IDE et les CLI. Ces agents associent la compréhension du langage naturel à un raisonnement structuré pour aider, augmenter et automatiser le développement de logiciels, allant de la génération de fonctions à la correction de bogues et à la création de tests.

Contrairement aux outils de saisie semi-automatique, les agents de codage interprètent activement les objectifs des utilisateurs, interrogent l'environnement de développement pour obtenir du contexte (par exemple, il ouvre des fichiers et trace les erreurs), identifient les exigences, puis proposent et exécutent des actions.

Architecture

Un modèle d'agent de codage est illustré dans le schéma suivant :



Description

1. Reçoit une requête
   - L'utilisateur fournit des instructions en langage naturel par le biais d'une palette de commandes, d'une fenêtre de discussion ou d'une CLI (par exemple, « Ajouter une journalisation à cette fonction » ou « Refactoriser pour plus de lisibilité »).

2. Extrait le contexte de l'environnement
   - L'agent collecte le contexte de l'IDE, notamment les fichiers actifs, la position du curseur, les extraits de code et les tables de symboles.
   - Il produit des messages d'erreur, des résultats de test et des sorties provenant d'autres agents.
3. Raisonnement du LLM
   - L'agent envoie une invite, y compris la requête et le contexte environnemental, à un LLM.
     - Le LLM effectue une passe de raisonnement pour déterminer ce qui suit :
     - Ce qui doit changer
     - Comment générer une solution
     - Toutes les étapes de refactorisation, de réécriture ou de codage
4. Exécute des actions
   - Le LLM renvoie le résultat à l'agent et l'importe dans l'environnement d'exécution ou dans l'environnement d'exécution.
   - Cela peut inclure l'insertion ou la modification de code, la génération de commentaires ou de documentation, et le déclenchement de tâches de construction, de test et de linting en aval.

Fonctionnalités

- High-contextual prise de conscience (par exemple, état de l'IDE, curseur et arbre syntaxique)
- Raisonnement itératif des objectifs et du feedback
- Planification du code et séparation des actions facultatives (par exemple, d'abord raisonner puis agir)
- Fonctionne dans des flux de travail de développement synchrones ou asynchrones

Cas d’utilisation courants

- Génération de code à partir des descriptions de tâches
- Refactorisation et optimisation du code
- Test-case génération et validation
- Explications des erreurs et débogage
- Assistants de documentation
- Copilotes de programmation jumelés

Directives d’implémentation

- Vous pouvez créer ce modèle à l'aide des outils suivants et Services AWS :
- Amazon Bedrock pour la LLM-driven génération et le raisonnement
- Amazon Q Developer pour les suggestions de codage et les complétions
- AWS Lambda ou Amazon Elastic Container Service (Amazon ECS) pour exécuter et tester des environnements sandbox
- AWS Cloud9, des extensions VS Code ou des intégrations IDE personnalisées pour héberger et évaluer le contexte
- Amazon Simple Storage Service (Amazon S3) pour stocker les demandes intermédiaires, les réponses et l'historique des révisions

Résumé

Les agents de codage sont de nouveaux outils de AI-powered développement capables d'interpréter le langage naturel, d'analyser le contexte, de générer des modifications de code en plusieurs étapes et de s'intégrer au cycle de vie du développement logiciel.

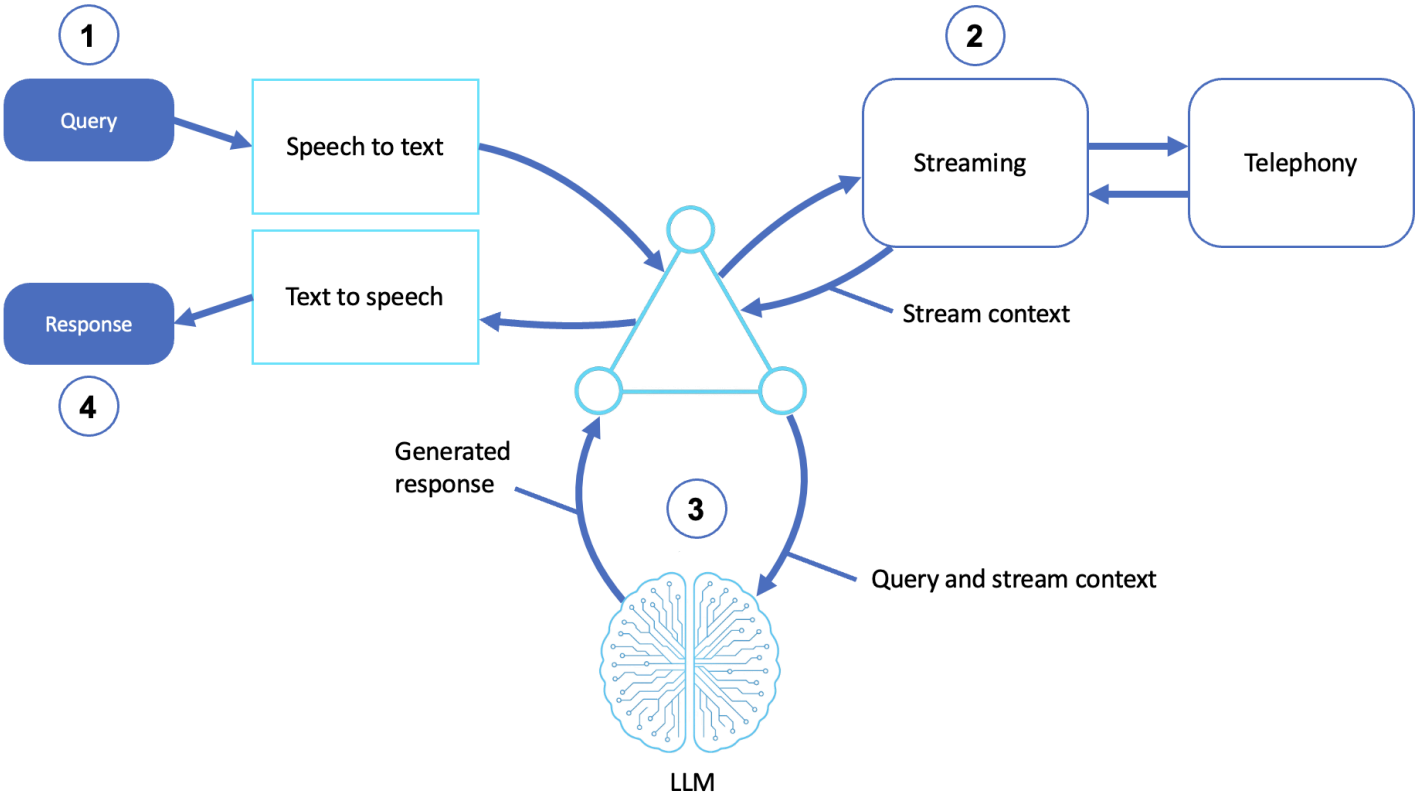
Agents vocaux et vocaux

Les agents vocaux et vocaux interagissent avec les utilisateurs par le biais de dialogues vocaux. Ces agents intègrent la reconnaissance vocale, la compréhension du langage naturel et la synthèse vocale pour permettre l'IA conversationnelle sur la téléphonie, le mobile, le Web et les plateformes intégrées.

Les agents vocaux sont particulièrement efficaces dans les environnements mains libres, en temps réel ou axés sur l'accessibilité. En combinant les interfaces de streaming avec LLM-powered le raisonnement, ils facilitent des interactions riches et dynamiques qui semblent naturelles pour les utilisateurs.

Architecture

Un agent vocal et vocal est illustré dans le schéma suivant :



Description

1. Reçoit une requête vocale
   - L'utilisateur adresse une demande à un téléphone, à un microphone ou à un système intégré.
   - Un module Speech-to-Text (STT) convertit le son en texte.
2. Intègre le contexte du streaming et de la téléphonie
   - L'agent utilise une interface de streaming pour gérer le son I/O en temps réel.
   - Si elle est déployée dans un centre d'appels ou dans un contexte de télécommunications, l'intégration de la téléphonie gère le routage des sessions, l'entrée multifréquence bicolore (DTMF) et le transport multimédia.

Remarque : Le DTMF fait référence aux tonalités générées lorsque vous appuyez sur les boutons du clavier d'un téléphone. Dans le contexte du streaming et de l'intégration du contexte téléphonique dans les agents vocaux, le DTMF est utilisé comme mécanisme d'entrée de signal lors d'un appel téléphonique, en particulier dans les systèmes de réponse vocale interactive (IVR). Les entrées DTMF permettent à l'agent de :

- Reconnaissez les sélections de menu (par exemple, « Appuyez sur 1 pour la facturation »). Appuyez sur 2 pour obtenir de l'aide. »)
- Collectez des entrées numériques (par exemple, des numéros de compte, des codes PIN et des numéros de confirmation)
- Déclenchez des workflows ou des transitions d'état dans les flux d'appels
- Passez de la parole à la tonalité tactile si nécessaire

1. Raisons liées au contexte du flux LLM
   - La requête est envoyée à l'agent, qui la transmet, avec toutes les métadonnées de session (par exemple, l'identifiant de l'appelant, le contexte antérieur), à un LLM.
   - Le LLM génère une réponse, éventuellement en utilisant une stratégie de chaîne de pensée ou une mémoire à plusieurs tours si l'interaction est en cours.
2. Renvoie une réponse vocale
   - L'agent convertit sa réponse en parole à l'aide de la synthèse vocale (TTS).
   - Il renvoie le son à l'utilisateur via un canal vocal.

Fonctionnalités

- Real-time compréhension et génération de la parole
- Multilingue I/O avec support STT et TTS
- Intégration aux API de téléphonie ou de streaming
- Sensibilisation aux sessions et transfert de mémoire entre les tours

Cas d’utilisation courants

- Systèmes IVR conversationnels
- Réceptionnistes et planificateurs de rendez-vous virtuels
- Voice-driven agents du service d'assistance
- Assistants vocaux portables
- Interfaces vocales pour maisons intelligentes et outils d'accessibilité

Directives d’implémentation

Vous pouvez créer ce modèle à l'aide des outils suivants et Services AWS :

- Amazon Lex V2 ou Amazon Transcribe pour STT
- Amazon Polly pour TTS
- Amazon Chime SDK, Amazon Connect Customer ou Amazon Interactive Video Service (Amazon IVS) pour le streaming et la téléphonie
- Amazon Bedrock pour raisonner avec Anthropic, AI21 ou d'autres modèles de base
- AWS Lambda pour connecter STT, LLM, TTS et le contexte de session

(Facultatif) Les améliorations supplémentaires peuvent inclure les suivantes :

- Amazon Kendra ou OpenSearch pour un RAG sensible au contexte
- Amazon DynamoDB pour la mémoire de session
- Amazon CloudWatch Logs et AWS X-Ray pour la traçabilité

Résumé

Les agents vocaux et vocaux sont des systèmes intelligents qui interagissent par le biais de conversations naturelles. En intégrant les interfaces vocales au raisonnement LLM et à l'infrastructure de streaming en temps réel, les agents vocaux permettent des interactions fluides, accessibles et évolutives.

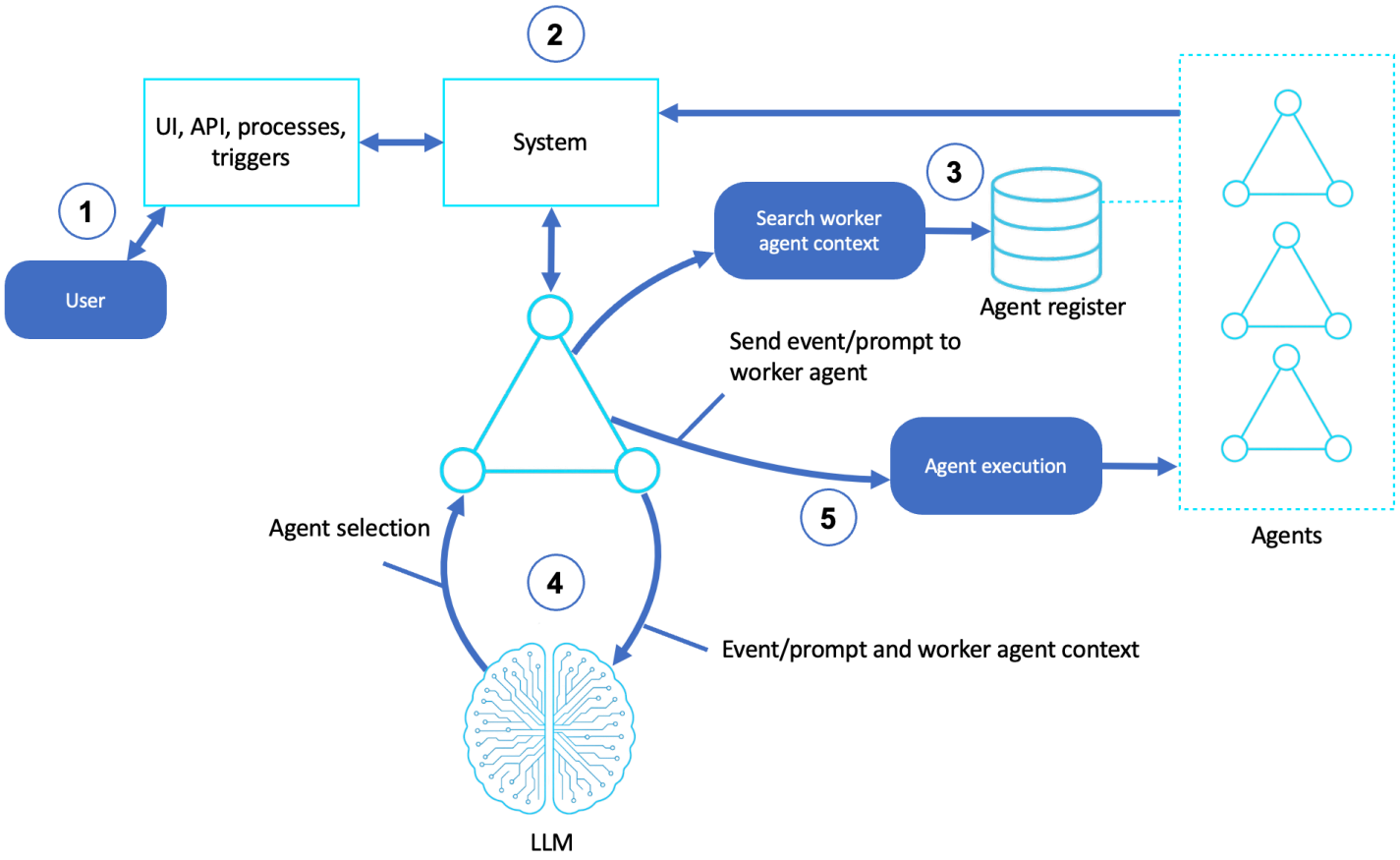
Agents d'orchestration des flux de travail

Les agents d'orchestration des flux de travail gèrent et coordonnent les tâches, les processus et les services en plusieurs étapes sur les systèmes distribués. Plutôt que de raisonner et d'agir de manière isolée, ces agents délèguent le travail à des sous-agents ou à d'autres systèmes, maintiennent le contexte d'exécution et s'adaptent en fonction des résultats intermédiaires.

Ces agents constituent un élément fondamental des flux d'automatisation. Ils sont particulièrement utiles lors de la gestion de tâches de longue durée, de compositions multi-agents et d'intégrations entre domaines où différents agents et outils doivent être appelés en séquence ou de manière conditionnelle.

Architecture

Un agent d'orchestration de flux de travail est illustré dans le schéma suivant :



Description

1. Reçoit les entrées utilisateur
   - Un utilisateur (ou un déclencheur externe) lance une tâche via une interface utilisateur, une API ou un événement système.
2. Gère les événements du système
   - Un composant du système reçoit la demande et émet un événement ou une commande nécessitant une orchestration.
3. Récupère le contexte
   - L'agent de flux de travail interroge les bases de connaissances et les registres d'agents afin de trouver l'agent de travail adapté à la tâche en fonction des métadonnées, du domaine et du taux de réussite antérieur.
4. Sélectionne un agent LLM

- Un LLM permet de sélectionner le meilleur agent ou le meilleur plan de flux de travail en analysant la description de la tâche et les options disponibles.
- Il peut également formuler des instructions spécifiques à une tâche à envoyer à un agent sélectionné.

5. Délégue et exécute
   - L'agent de travail choisi reçoit l'événement ou l'invite et commence à exécuter des commandes.
   - Il peut suivre l'état d'exécution, réessayer en cas d'échec et transmettre les résultats intermédiaires à l'agent suivant de la séquence.

Fonctionnalités

- Composition des agents (par exemple, superviseurs, agents collaborateurs et outils)
- Event-driven ou exécution planifiée
- Suivi de la mémoire et de l'état au fil du temps
- Orchestration hiérarchique ou parallèle des tâches (synchrone par rapport aux flux de travail asynchrones)
- Sélection et chaînage dynamiques des agents

Cas d’utilisation courants

- Automatisation en plusieurs étapes (par exemple, ingestion de données et création de rapports)
- Routage et escalade du service client (par exemple, agent-en-coordinateur)
- Les agents d'IA coordonnent leurs activités avec les humains et les robots au sein de la même boucle
- Automatise les processus d'entreprise à l'aide LLM-powered de la logique
- Les systèmes hybrides combinent des agents d'intelligence artificielle et des outils d'orchestration traditionnels

Directives d’implémentation

Vous pouvez créer ce modèle à l'aide des outils suivants et Services AWS :

- Amazon Bedrock pour le raisonnement et la sélection des agents

- AWS Step Functions ou Amazon EventBridge pour la composition du flux de travail
- AWS Lambda en tant qu'unités d'exécution ou exécuteurs de tâches
- Amazon DynamoDB, Amazon Simple Storage Service (Amazon S3) ou Amazon RDS pour suivre les états et les résultats
- AWS AppFabric ou Amazon AppFlow pour la coordination entre systèmes
- (Facultatif) Utilisez Amazon SageMaker Run Agent pour héberger des agents de travail spécifiques à un domaine

Résumé

Les agents de flux de travail coordonnent, adaptent et alignent les objectifs dans les environnements multi-agents. Cela signifie que les agents d'IA peuvent collaborer, s'adapter aux conditions d'exécution et obtenir des résultats complexes grâce à des flux de travail modulaires et explicables.

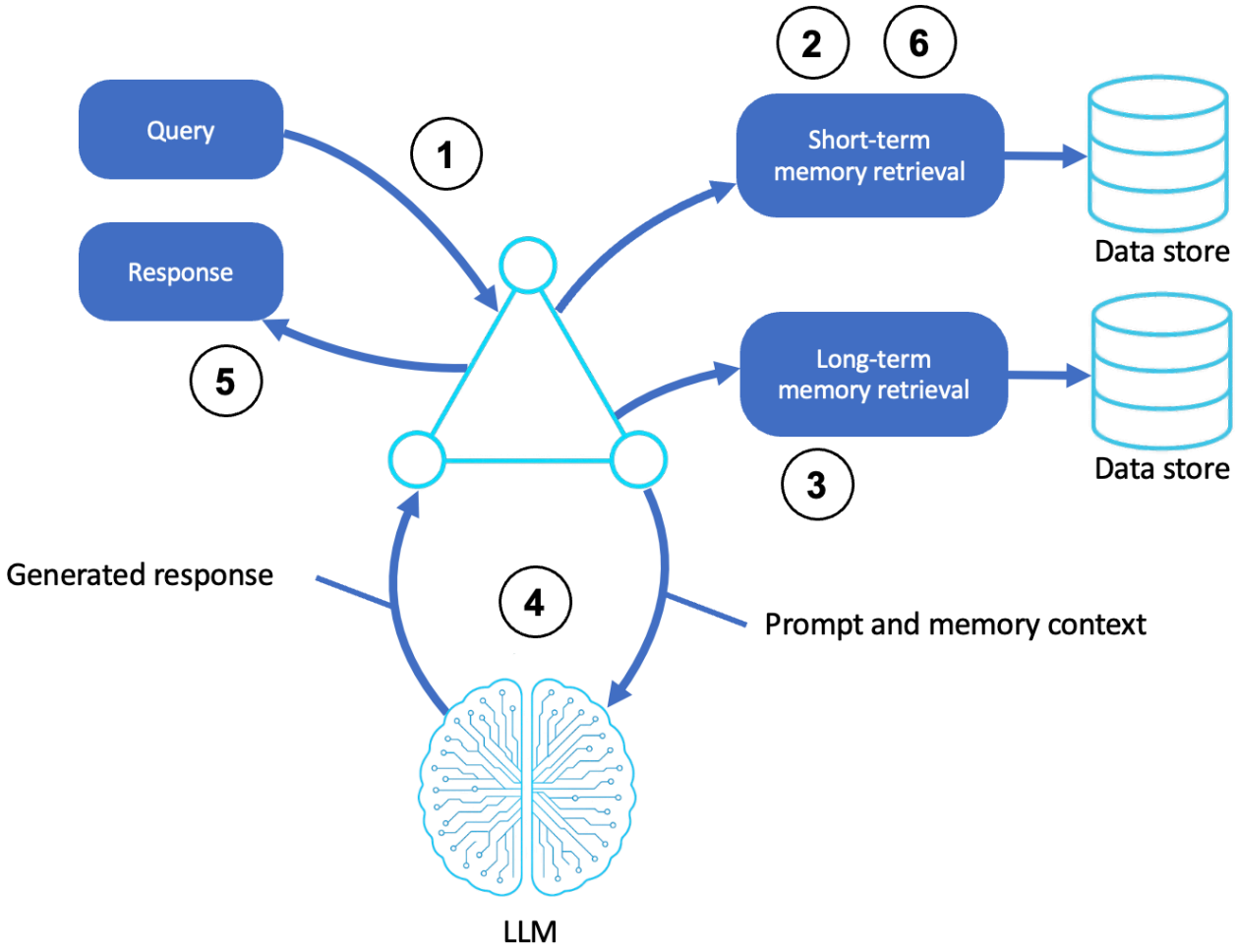
Memory-augmented agents

Memory-augmented les agents sont améliorés grâce à la capacité de stocker, de récupérer et de raisonner en utilisant la mémoire à court et à long terme. Cela leur permet de maintenir le contexte entre plusieurs tâches, sessions et interactions, ce qui produit des réponses plus cohérentes, personnalisées et stratégiques.

Contrairement aux agents apatrides, les agents à mémoire augmentée s'adaptent en faisant référence à des données historiques, en tirant des leçons des résultats antérieurs et en prenant des décisions conformes aux objectifs, aux préférences et à l'environnement de l'utilisateur.

Architecture

Un agent à mémoire augmentée est illustré dans le schéma suivant :



Description

1. Reçoit une entrée ou un événement
   - L'agent reçoit une requête utilisateur ou un événement système. Il peut s'agir d'un texte, d'un déclencheur d'API ou d'un changement environnemental.
2. Récupère la mémoire à court terme
   - L'agent récupère l'historique des conversations récentes, le contexte des tâches ou l'état du système correspondant à la session ou au flux de travail.
3. Récupère la mémoire à long terme
   - L'agent interroge la mémoire à long terme (par exemple, les bases de données vectorielles et les banques de valeurs clés) pour obtenir des informations historiques, telles que les suivantes :
     - Préférences de l'utilisateur
     - Décisions et résultats antérieurs

    - Concepts, résumés ou expériences apprises
4. Raisons motivées par le LLM
    - Le contexte de mémoire est intégré à l'invite LLM, ce qui permet à l'agent de raisonner en fonction à la fois des entrées actuelles et des connaissances antérieures.
5. Génère des sorties
    - L'agent produit une réponse, un plan ou une action adaptés au contexte qui sont personnalisés en fonction de l'historique des tâches et des entrées de l'utilisateur.
6. Actualise la mémoire
    - Les nouvelles informations, telles que les objectifs actualisés, les signaux de réussite et d'échec et les réponses structurées, sont stockées pour les tâches futures.

Fonctionnalités

- Continuité des sessions entre les conversations ou les événements
- Persistance des objectifs dans le temps
- Conscience contextuelle basée sur un état en évolution
- Adaptabilité éclairée par les succès et les échecs antérieurs
- Personnalisation alignée sur les préférences et l'historique de l'utilisateur

Cas d’utilisation courants

- Copilotes conversationnels qui mémorisent les préférences de l'utilisateur
- Agents de codage qui suivent les modifications de la base de code
- Des agents de flux de travail qui s'adaptent en fonction de l'historique des tâches
- Des jumeaux numériques qui évoluent à partir de la connaissance du système
- Des agents de recherche qui évitent les récupérations redondantes

Implémentation d'agents à mémoire augmentée

Utilisez les outils suivants et Services AWS pour les agents à mémoire augmentée :

Couche mémoire	Service AWS	Objectif

Short-term	Contexte Amazon DynamoDB, Redis, Amazon Bedrock	Récupération rapide des états d'interaction récents
Long-term (structuré)	Amazon Aurora, Amazon DynamoDB, Amazon Neptune	Faits, relations et journaux
Long-term (sémantique)	OpenSearch, PostgreSQL, Pinecone	Embedding-based récupérat ion (c'est-à-dire RAG)
Stockage	Amazon S3	Stockage de transcriptions, de mémoires structurées et de fichiers
Orchestration	AWS Lambda ou AWS Step Functions	Gestion de l'injection de mémoire et du cycle de mise à jour
Raisonnement	Amazon Bedrock	Claude ou Mistral anthropiq ue avec des instructions de mémorisation

Implémentation de l'envoi de messages par injection de mémoire

Pour intégrer la mémoire dans le raisonnement des agents, utilisez une combinaison d'états structurés et d'injection de contexte augmentée par extraction :

- Incluez l'état le plus récent de l'agent et l'historique récent des dialogues en tant qu'entrée structurée lors de la création de l'invite pour le modèle de langage, afin que celui-ci puisse raisonner en tenant compte du contexte complet.
- Utilisez la génération augmentée par extraction (RAG) pour extraire des documents ou des faits pertinents de la mémoire à long terme.
- Résumez les plans, le contexte et les interactions précédents à des fins de compression et de pertinence.
- Injectez des modules de mémoire externes, tels que des magasins de vecteurs ou des journaux structurés, lors de l'inférence pour guider la prise de décision.

Résumé

Memory-augmented les agents maintiennent une continuité de pensée en tirant des leçons de leur expérience et en mémorisant le contexte de l'utilisateur. Ces agents surpassent l'intelligence réactive en utilisant la collaboration à long terme, la personnalisation et le raisonnement stratégique. En termes d'IA agentique, la mémoire permet aux agents de se comporter davantage comme des homologues numériques adaptatifs que comme des outils apatrides.

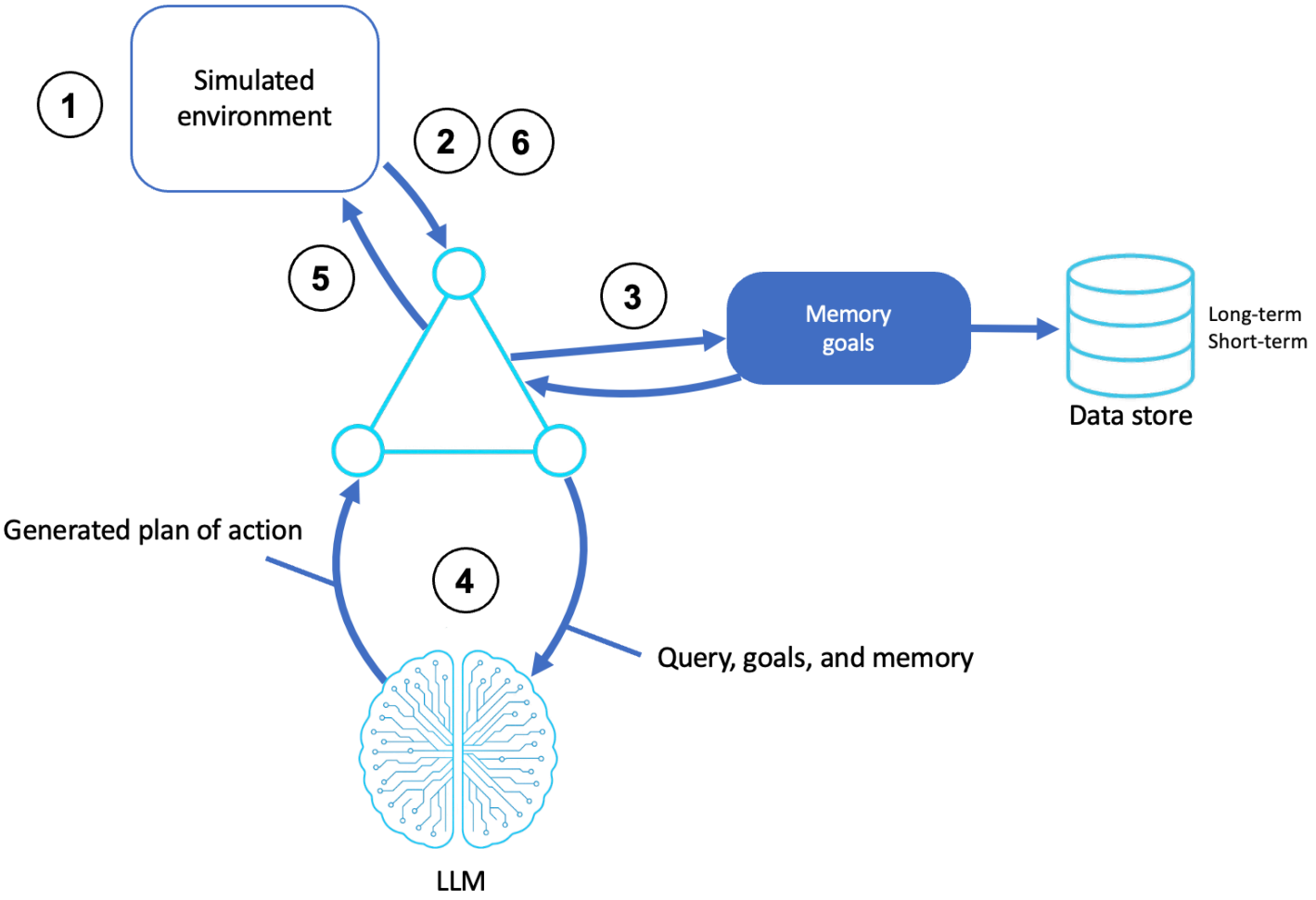
Agents de simulation et de banc d'essai

Les agents de simulation et de banc d'essai opèrent dans des environnements virtualisés ou contrôlés dans lesquels ils raisonnent, agissent et apprennent. Ces agents simulent le comportement, modélisent les résultats et entraînent des stratégies dans des contextes répétables avant de les appliquer à des environnements réels.

Ce modèle est utile pour le développement itératif, l'apprentissage par renforcement (RL), l'évaluation de la prise de décision autonome et les tests comportementaux émergents. Les agents de simulation fonctionnent souvent en boucle fermée, reçoivent des informations de leur environnement et ajustent leur comportement en conséquence, ce qui les rend essentiels pour les tâches impliquant un raisonnement spatial, un contrôle en temps réel ou une dynamique de système complexe.

Architecture

Le schéma suivant montre un agent de simulation ou un agent de banc d'essai :



Description

1. Lance un environnement
   - L'agent lance un environnement simulé (par exemple, un monde 3D, un moteur physique, un bac à sable CLI ou un flux de données synthétiques).
   - L'agent est chargé dans l'environnement avec une tâche, un objectif ou une politique initiale.
2. Perçoit l'agent
   - L'agent perçoit l'état actuel par le biais de la télémétrie de simulation (par exemple, émulation de capteurs, caméra virtuelle et journaux structurés).
3. Récupère l'objectif et la mémoire
   - L'agent récupère l'objectif, les instructions du scénario ou l'objectif contextuel qui lui ont été assignés.
   - Il peut également récupérer de la mémoire antérieure, notamment les éléments suivants :

    - Long-term stratégies ou politiques
    - Cartes environnementales ou contraintes connues
    - Succès ou échecs antérieurs issus de simulations similaires
4. Raisons et plans
    - Un LLM interprète l'état simulé, les objectifs de la tâche et les connaissances apprises.
    - Il génère un plan d'action ou une commande de contrôle.
5. Exécute des actions simulées
    - L'agent exécute le plan, modifie l'état, navigue dans l'espace ou interagit avec des entités virtuelles.
6. Apprend
    - L'agent évalue les résultats des actions
    - En fonction de la configuration de l'agent, celui-ci peut effectuer les opérations suivantes :
        - Exécuter une URL
        - Enregistrez les résultats pour les ajustements futurs
        - Adaptez les stratégies en temps réel

Fonctionnalités

- Fonctionne dans des environnements synthétiques ou virtuels
- Soutient l'apprentissage par essais et erreurs, l'affinement des politiques et la modélisation du système
- Low-risk tests de comportement, de gestion des défaillances et de cas extrêmes
- Permet l'analyse du comportement des agents émergents dans les configurations multi-agents
- Supporte à la fois le contrôle en boucle fermée et l'exploration par l'homme en boucle

Cas d’utilisation courants

- Apprentissage par renforcement pour la robotique, les drones et les jeux vidéo
- Formation aux véhicules autonomes sur des routes virtuelles
- UI ou CLI simulées pour des scénarios de banc DevOps d'essai
- Expériences comportementales émergentes dans le cadre de simulations sociales

- Validation de sécurité de la logique de décision avant la production

Directives d’implémentation

Vous pouvez créer un agent de simulation et de banc d'essai à l'aide des outils suivants et : Services AWS

Composant	Service AWS	Objectif
Environnement	Amazon ECS, Amazon EC2 ou un simulateur personnalisé dans le laboratoire Amazon Studio SageMaker	Exécutez des mondes virtuels (Gazebo, Unity, Unreal) ou des CLI sandbox
Logique de l'agent	Amazon Bedrock SageMaker, Amazon ou AWS Lambda	LLM-based planificateurs ou agents RL
Boucle de rétroaction	Amazon SageMaker Reinforcement Learning CloudWatch, Amazon ou journaux personnalisés	Suivi des récompenses, évaluation des résultats et enregistrement des comportements
Mémoire et rediffusion	Amazon S3, Amazon DynamoDB ou Amazon RDS	État persistant, historique des épisodes ou données de scénario
Visualisation	Tableaux de CloudWatch bord Amazon ou blocs-notes Amazon SageMaker	Observez les changements de politique, les résultats et les indicateurs de formation

Les applications supplémentaires suivantes sont les suivantes :

- AWS SimSpace Weaverpour des simulations spatiales à grande échelle
- AWS IoT Corepour tester les appareils fantômes
- Amazon SageMaker Experiments pour l'évaluation et l'analyse comparative des agents

Résumé

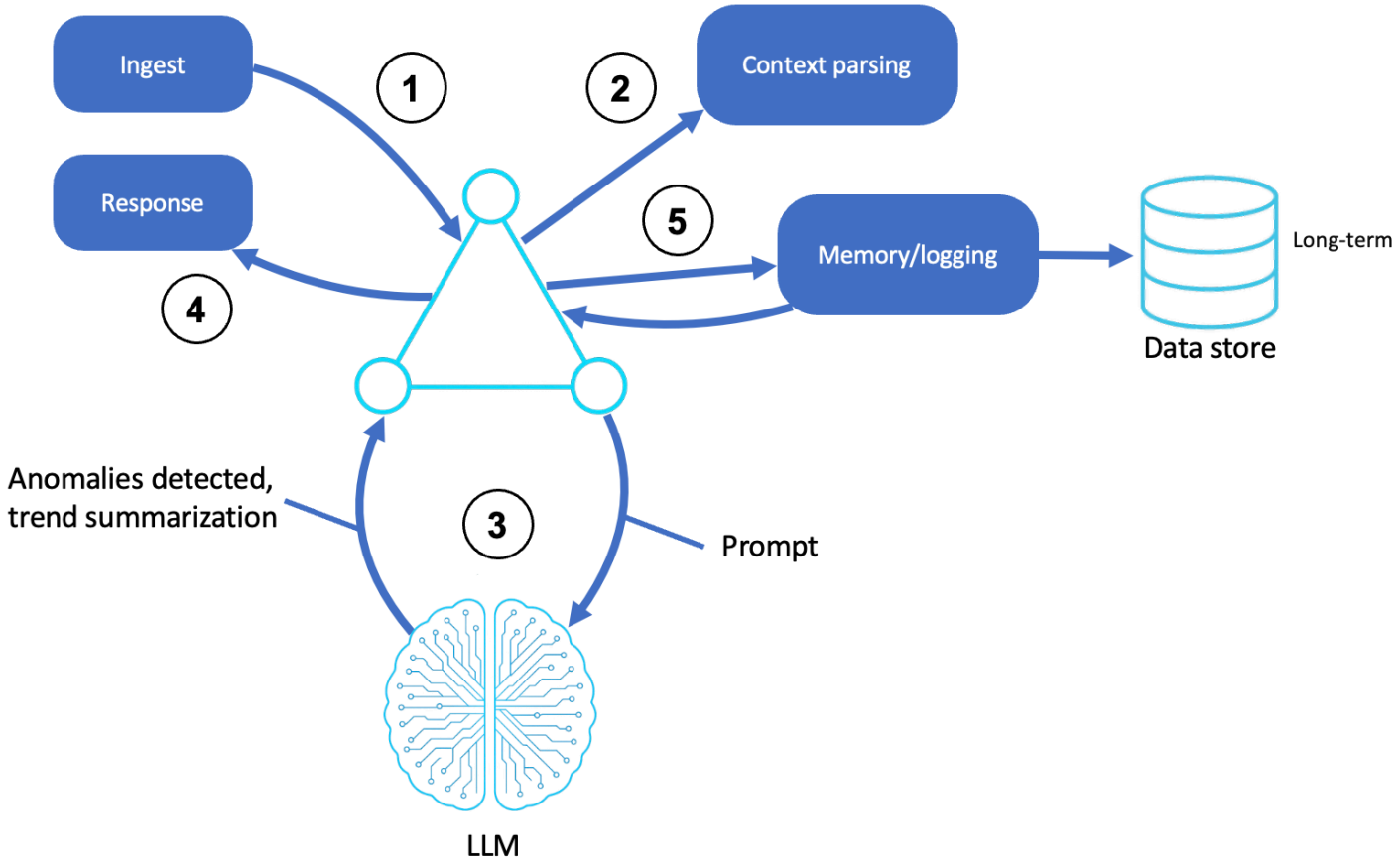
Les agents de simulation et de banc d'essai sont destinés à une exploration structurée avant d'être déployés dans les systèmes de production. Utilisez ces agents pour élaborer des politiques de navigation autonomes, tester des processus métier dans des environnements synthétiques et évaluer les essaims pour déterminer les modèles de coordination.

Observateurs et agents de surveillance

Les observateurs et les agents de surveillance observent passivement les systèmes, les environnements et les interactions afin de détecter des modèles, de générer des informations et de déclencher des actions. En tant qu'observateurs intelligents, ils améliorent les alertes, les diagnostics et les audits sans déclencher directement de comportement.

Ces agents excellent là où la surveillance traditionnelle manque d'adaptabilité ou de raisonnement, en particulier pour la AI-in-the-loop surveillance, la détection des anomalies, le contrôle de la conformité et le renseignement de sécurité. Les agents observateurs sont des auditeurs d'événements qui surveillent en permanence la télémétrie du système et les interactions avec les utilisateurs. L'agent dépend de la perception, de l'interprétation et de l'escalade conditionnelle ou du reporting.

Architecture



Description

1. Ingérer la télémétrie
   - L'agent reçoit des informations provenant d'une ou de plusieurs sources système, telles que les suivantes :
     - Logs (application, infrastructure, sécurité)
     - Métriques (performances, latence, utilisation)
     - Événements (appels d'API, actions des utilisateurs, données des capteurs)
2. Analyser le contexte
   - Les entrées brutes sont analysées, structurées et enrichies de métadonnées, telles que l'horodatage, l'identité de l'acteur, l'état du système et l'identifiant de trace.
3. Les raisons d'utiliser le LLM

- L'agent utilise un LLM ou un module logique pour interpréter les entrées analysées en identifiant les anomalies, en résumant les tendances et en établissant des corrélations entre des traces distribuées ou des fenêtres temporelles.

4. Classer ou alerter
   - L'agent détermine si le comportement observé justifie ce qui suit :
     - Une alerte ou une escalade
     - Mise à jour du rapport ou du tableau de bord
     - Un déclencheur de réponse (par exemple, correction automatique et application des politiques)
5. Mémoire journal ou boucles de rétroaction
   - L'agent enregistre les événements et les décisions à des fins d'apprentissage à long terme, d'audits ou de futures références pour d'autres agents.

Fonctionnalités

- Passif et non invasif (l'agent n'agit pas directement)
- Hautement évolutif et asynchrone
- AI-driven corrélation entre signaux bruyants ou distribués
- Soutient l'audit, la conformité et les informations en temps réel
- Peut alimenter les agents en aval ou les flux de travail humains

Cas d’utilisation courants

- AI-augmented observabilité pour les microservices et les API
- Surveillance de la dérive du modèle, de la violation des règles ou du comportement hors bande
- Analyse de l'activité des clients ou résumés des interactions
- Agents de révision du code qui surveillent les validations ou les déploiements
- Surveillance des journaux de sécurité ou de conformité à l'aide du raisonnement LLM

Directives d’implémentation

Vous pouvez créer un observateur et un agent de surveillance à l'aide des outils suivants et Services AWS :

Composant	Service AWS	Objectif
Ingestion d'événements	Amazon EventBridge, Amazon CloudWatch Logs, Amazon Kinesis, Amazon S3	Ingérer des données de télémétrie structurées et non structurées
Prétraitement	AWS Lambda, AWS Glue, AWS Step Functions	Transformez les données brutes en instructions structuré es
Moteur de raisonnement	Amazon Bedrock, Amazon SageMaker, AWS Lambda	Analyser les événements, classer les comportements, générer des informations
Stockage et mémoire	Amazon S3, Amazon DynamoDB, OpenSearch	Observations, résumés et résultats persistants
Alerte et escalade	Amazon SNS, AWS AppFabric Amazon EventBridge	Déclenchez des systèmes ou des agents en aval

Les applications supplémentaires suivantes sont les suivantes :

- AWS Security Hub CSPMpour la surveillance des journaux de sécurité
- Amazon Quick pour visualiser les résultats des agents

Résumé

Les observateurs et les agents de surveillance suivent les systèmes et les comportements en temps réel. Ils détectent les anomalies, auditent la sécurité et collectent des informations opérationnelles en identifiant les modèles que les humains ou les règles pourraient ignorer. Cette fonctionnalité permet de créer des systèmes capables de s'adapter aux conditions changeantes et de prendre des décisions basées sur une analyse complète des données.

Multi-agent collaboration

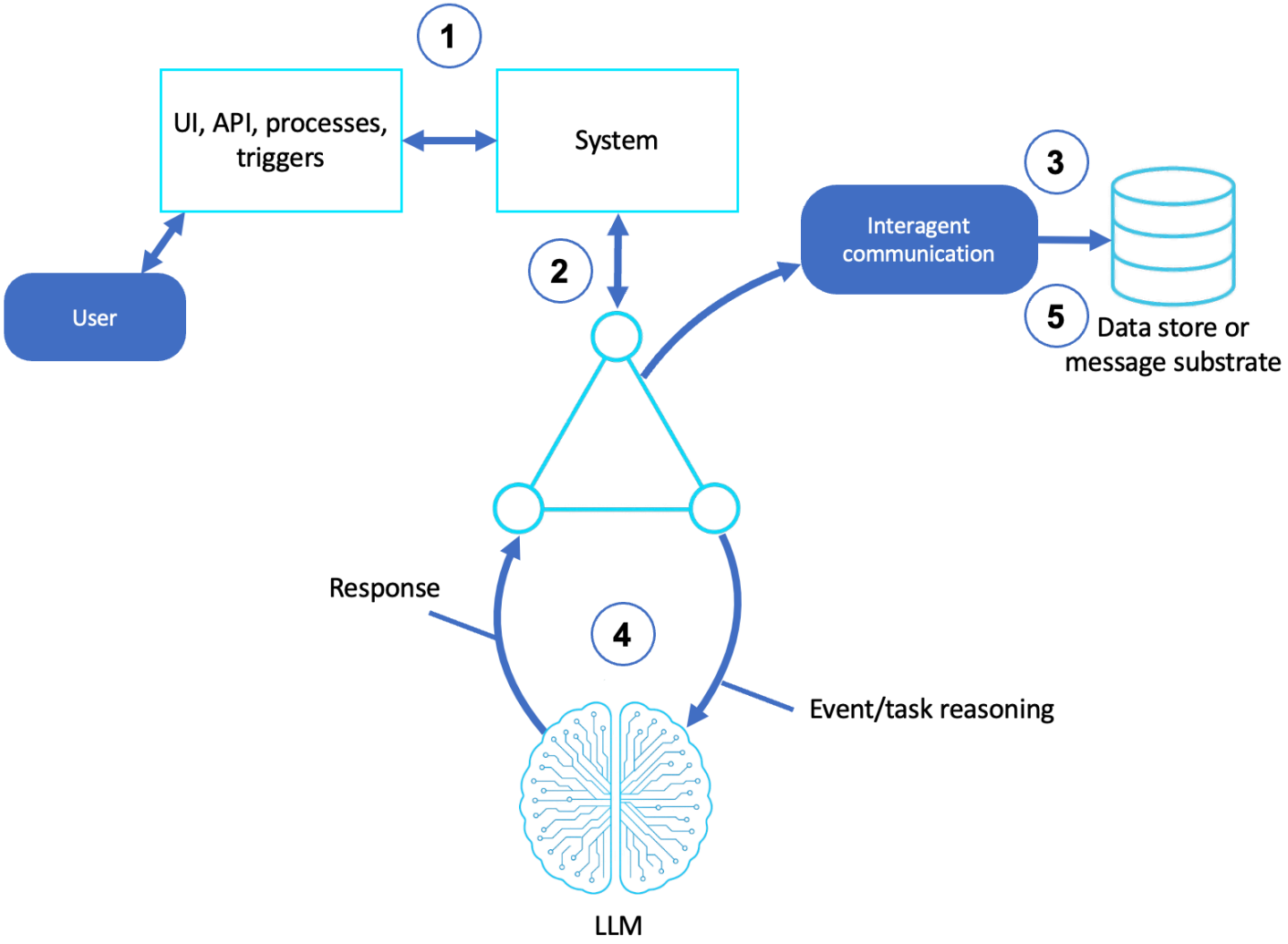
Multi-agent la collaboration fait référence à un modèle dans lequel plusieurs agents autonomes, chacun ayant un rôle, une spécialisation ou un objectif distinct, négocient pour résoudre des tâches complexes. Ces agents peuvent agir indépendamment ou avec d'autres agents en partageant des informations, en répartissant les responsabilités et en raisonnant collectivement en vue d'un objectif.

Ce modèle diffère de celui des agents de flux de travail, qui coordonnent et délèguent des tâches de manière centralisée aux agents subordonnés dans un flux structuré. En revanche, la collaboration multi-agents met l'accent sur la coordination entre pairs ou émergente en permettant l'adaptabilité, le parallélisme et la division de la cognition. Le tableau suivant compare la collaboration multi-agents avec les agents de flux de travail :

Fonctionnalité	Agents de workflow	Objectif
Contrôle	Coordinateur centralisé	Des pairs décentralisés, distribués ou basés sur les rôles
Interaction	Un agent délègue et suit l'exécution	Plusieurs agents négocient, partagent et s'adaptent
Conception	Séquence de tâches prédéfinie	Répartition des tâches innovante et flexible
Coordination	Orchestration procédurale	Interactions coopératives ou compétitives
Cas d’utilisation	Automatisation des processus d'entreprise	Raisonnement complexe, exploration et stratégies émergentes

Architecture

Le schéma suivant illustre la collaboration entre plusieurs agents :



Description

1. Lance une tâche
   - Un utilisateur ou un système émet un objectif ou un problème de haut niveau.
   - Un agent « manager » ou un contexte initiateur définit l'objectif.
2. Attribue ou découvre des rôles
   - Les agents s'attribuent eux-mêmes (logique symbolique ou raisonnement) ou sont délégués (courtier d'événements) à d'autres rôles, tels que ceux de planificateur, de chercheur, d'exécuteur testamentaire, de critique ou d'explicateur.
3. Communique avec les autres agents
   - Les agents communiquent via la mémoire partagée, les files d'attente de messagerie ou le chaînage rapide.

- Ils peuvent débattre, se poser des questions ou se proposer des sous-tâches.

4. Utilise un raisonnement spécialisé
   - Chaque agent utilise son propre modèle ou sa propre logique de domaine pour résoudre la partie du problème qui lui est propre.
   - Les agents peuvent utiliser des LLM avec des instructions et de la mémoire spécifiques aux rôles.
5. Coordonne les résultats ou les objectifs
   - Les agents synthétisent les contributions dans une réponse, un plan ou une action finale.
   - (Facultatif) Un agent de supervision peut valider ou résumer le résultat synthétisé.

Fonctionnalités

- Peer-level agents dotés de rôles ou de compétences spécialisés
- Comportement émergent par le biais de la communication ou de la négociation
- Traitement parallèle de problèmes complexes ou multidimensionnels
- Soutient la délibération, l'autocorrection et l'itération réflexive
- Modélisez la dynamique sociale, la collaboration scientifique ou les rôles des équipes d'entreprise

Cas d’utilisation courants

- Équipes de recherche autonomes (agent de recherche, synthétiseur et validateur)
- Développement de logiciels (planificateur, codeur et testeur)
- Modélisation de scénarios commerciaux (finances, politiques et conformité)
- Négociation, appel d'offres ou raisonnement multipartite
- Tâches multimodales (image, texte et logique)

Directives d’implémentation

Vous pouvez créer un système multi-agents à l'aide des outils suivants et Services AWS :

Composant	Service AWS	Objectif

Hébergement d'agents	Amazon Bedrock, Amazon SageMaker, AWS Lambda	Héberger des LLM-driven agents individuels
Couche de communication	Amazon SQS, Amazon, EventBridge AWS AppFabric	Messagerie et coordination entre les agents
Mémoire partagée	Amazon DynamoDB, Amazon S3 ou OpenSearch	Multi-agent système de mémoire ou de tableau noir
Couche d'orchestration	AWS Step Functions, AWS Lambda oléoducs	Logique de lancement, de délai d'expiration, de repli et de nouvelle tentative
Identification de l'agent	Agents Amazon Bedrock (définis par rôle) et API AWS AppConfig Amazon Bedrock Converse (agents extérieurs à Amazon Bedrock)	Role-based Invocation d'un outil ou d'un agent et applicati on des limites
Interaction émergente	EventBridge Pipelines ou registres d'agents Amazon	Activer le routage ou l'escalad e dynamique des tâches

Résumé

Multi-agent la collaboration répartit les tâches de résolution de problèmes entre des agents modulaires et axés sur les rôles. Contrairement à l'orchestration des flux de travail, les modèles de collaboration utilisent l'intelligence émergente, la résilience et l'évolutivité qui reflètent la façon dont les humains résolvent les problèmes. Il est particulièrement utile pour les domaines ouverts, les tâches créatives, le raisonnement multimodal et les environnements qui bénéficient de perspectives diverses.

Conclusion

Les modèles évoqués précédemment illustrent les approches fondamentales des implémentations réelles de l'IA agentique. Du raisonnement de base à l'intelligence augmentée par la mémoire, chaque modèle est configuré de manière unique pour la perception, la cognition et l'action, sur la base de l'autonomie, de l'asynchronie et du libre arbitre.

Ces modèles partagent des vocabulaires et des plans techniques pour créer des systèmes intelligents orientés vers des objectifs. Qu'un modèle soit intégré dans une interface utilisateur, orchestré via des services cloud ou coordonné entre des équipes d'agents, chaque modèle est adaptable et modulaire.

Plats à emporter

- Les modèles d'agents sont composables : la plupart des agents du monde réel combinent deux modèles ou plus (par exemple, un agent vocal doté d'un raisonnement et d'une mémoire basés sur des outils).
- La conception de l'agent est contextuelle : choisissez des modèles en fonction de la surface d'interaction, de la complexité des tâches, de la tolérance à la latence et des contraintes spécifiques au domaine.
- AWSla mise en œuvre native est réalisable — Avec Amazon Bedrock SageMaker, Amazon,, AWS Lambda AWS Step Functions, et les architectures pilotées par les événements, chaque modèle d'agent peut être fourni à grande échelle.

Flux de travail LLM

En ce qui concerne les modèles d'agents, nous avons exploré les modèles d'agents d'IA courants, chacun étant construit autour d'un ensemble de capacités modulaires : perception, action, apprentissage et cognition. Au cœur du module cognitif de nombreux modèles d'agents se trouve un grand modèle de langage (LLM) capable de raisonner, de planifier et de prendre des décisions. Cependant, l'invocation d'un LLM à elle seule n'est pas suffisante pour produire un comportement intelligent et ciblé.

Pour effectuer des tâches complexes de manière fiable, les agents doivent intégrer le LLM dans un flux de travail structuré, dans lequel les capacités du modèle sont augmentées grâce à des outils, à de la mémoire, à des boucles de planification et à une logique de coordination. Ces flux de travail LLM permettent à un agent de définir des objectifs, d'acheminer des sous-tâches, d'appeler des services externes, de réfléchir aux résultats et de se coordonner avec d'autres agents.

Ce chapitre présente les modèles de conception de base pour créer des modules LLM-driven cognitifs robustes, extensibles et intelligents, organisés autour de flux de travail réutilisables.

Dans cette section

- Vue d'ensemble de LLM-augmented la cognition
- Flux de travail pour un chaînage rapide
- Flux de travail pour le routage
- Workflow pour la parallélisation
- Flux de travail pour l'orchestration
- Flux de travail pour les évaluateurs et boucles d'affinage par réflexion
- Conclusion

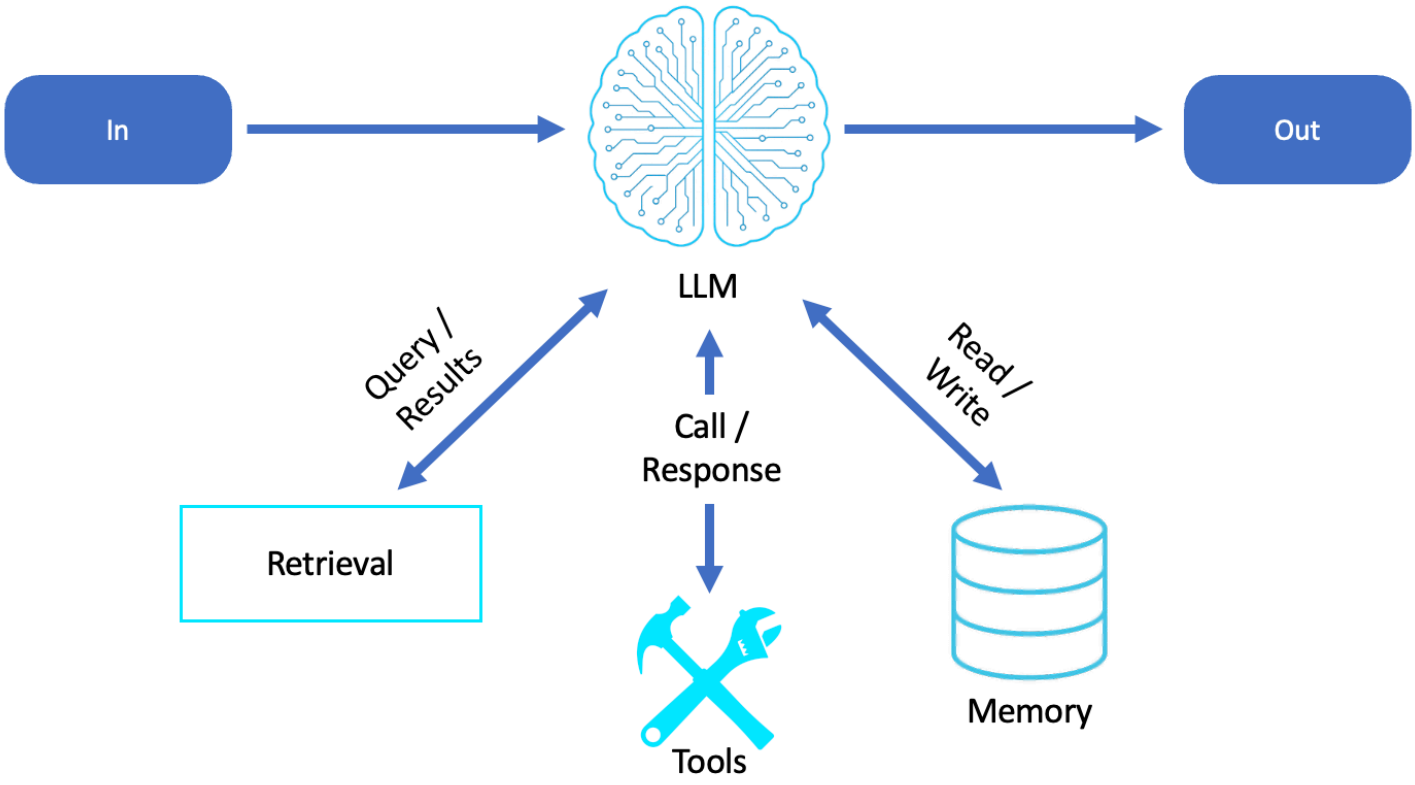
Vue d'ensemble de LLM-augmented la cognition

À la base, le module cognitif d'un agent logiciel peut être considéré comme un LLM assorti d'augmentations. L'agent peut utiliser les éléments de base suivants pour raisonner efficacement dans son environnement :

- Invitation — Cadrage de la saisie à l'aide du contexte, des instructions, des exemples et de la mémoire

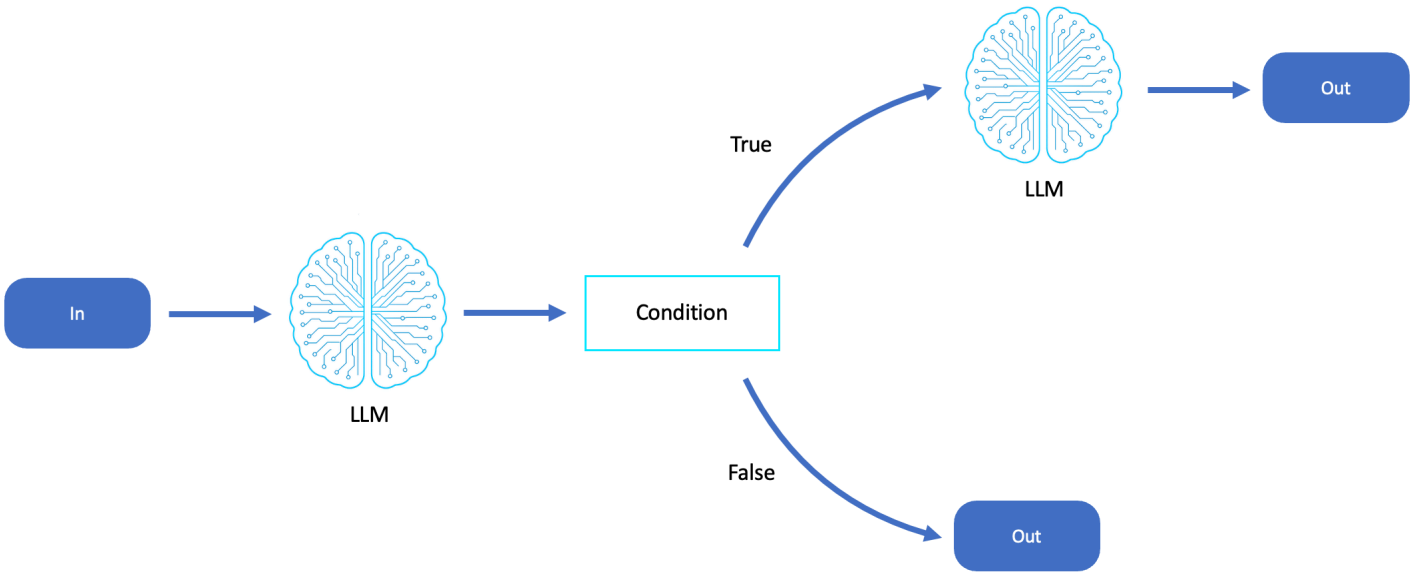
- Récupération — Fournir des connaissances actualisées ou spécifiques à un domaine à l'invite LLM par le biais de la recherche vectorielle ou de la mémoire sémantique, par exemple, par le biais de la génération augmentée par extraction (RAG)
- Utilisation de l'outil — Permettre au LLM d'invoquer des API ou d'appeler des fonctions pour récupérer des informations ou agir sur celles-ci
- Mémoire — Incorporation d'un état persistant ou d'un état basé sur une session dans la boucle de raisonnement, soit en utilisant des bases de données structurées, soit des résumés contextuels

Ces augmentations sont composées de flux de travail qui définissent la manière dont le LLM est utilisé au fil du temps et entre les tâches, le transformant d'un moteur apatride en un agent de raisonnement dynamique.



Flux de travail pour un chaînage rapide

Le chaînage rapide décompose les tâches complexes en une séquence d'étapes, chaque étape étant un appel LLM discret qui traite ou s'appuie sur le résultat de la précédente.



Le flux de travail de chaînage rapide convient aux scénarios dans lesquels les tâches peuvent être logiquement divisées en étapes de raisonnement séquentielles, et dans lesquels les résultats intermédiaires indiquent l'étape suivante. Il excelle dans les flux de travail qui nécessitent une réflexion structurée, une transformation progressive ou une analyse en couches, tels que la révision de documents, la génération de code, l'extraction de connaissances et le raffinement du contenu.

Description

- La complexité de la tâche dépasse la fenêtre contextuelle ou la profondeur de raisonnement d'un seul appel LLM.
- Les résultats d'une étape (par exemple, analyse, synthèse ou planification) deviennent des entrées pour une décision de suivi ou une phase de génération.
- Vous avez besoin de transparence et de contrôle à chaque étape du raisonnement (par exemple, résultats intermédiaires vérifiables).
- Vous souhaitez intégrer une logique de validation, de filtrage ou d'enrichissement externe entre les étapes.
- Il est idéal pour les agents opérant dans des boucles de raisonnement de type pipeline, tels que les agents de recherche, les assistants éditoriaux, les systèmes de planification et les copilotes en plusieurs étapes.

Fonctionnalités

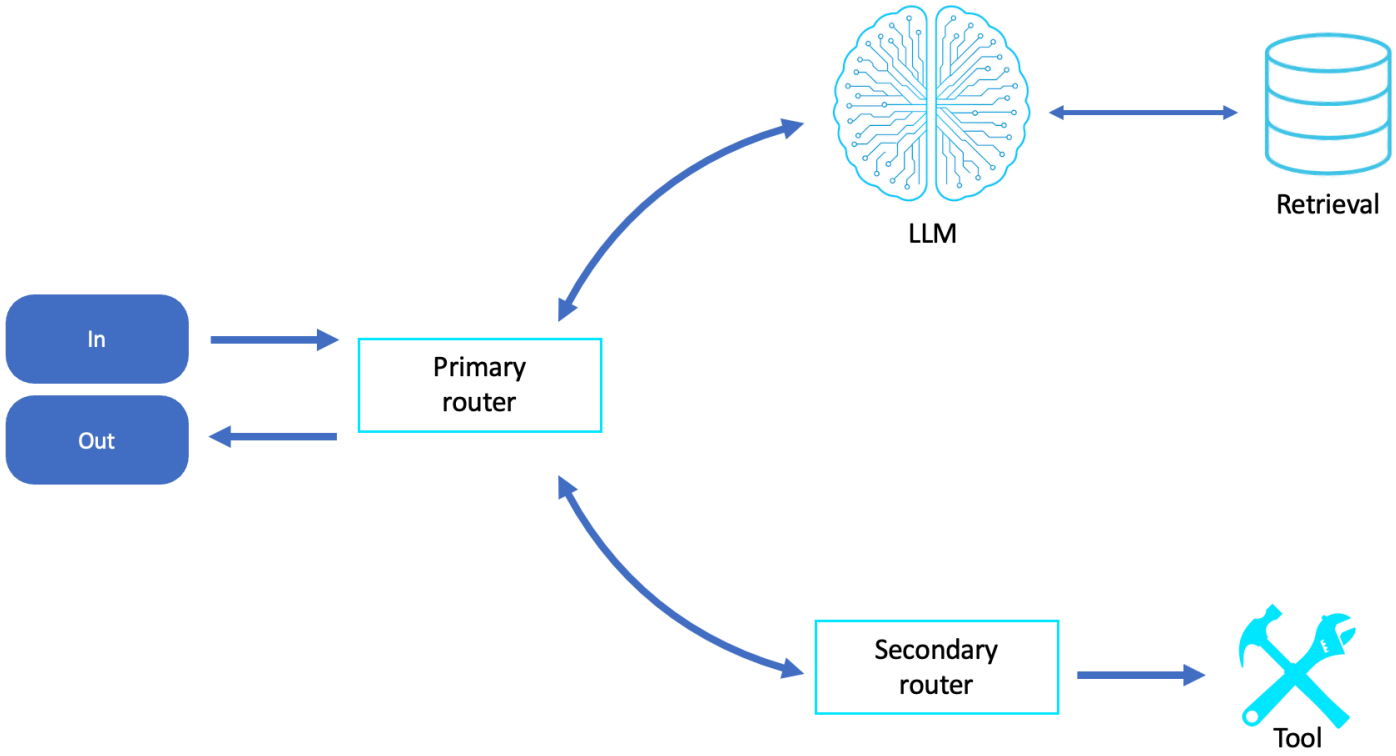
- Chaînes linéaires ou ramifiées d'appels LLM
- Résultats intermédiaires transmis sous forme de saisie structurée ou intégrés dans des instructions de suivi
- Peut être orchestré avec ou avec AWS Step Functions des coureurs AWS Lambda spécifiques à un agent

Cas d’utilisation courants

- Tâches de raisonnement en plusieurs étapes (par exemple, « réécriture d'une critique résumée »)
- Assistants de recherche synthétisant des résultats en couches (par exemple, « rechercher, extraire des faits, répondre à une question »)
- Pipelines de génération de code (« générer un plan, écrire du code, tester le code, expliquer la sortie »)

Flux de travail pour le routage

Dans le schéma de routage, un classificateur ou un agent de routeur utilise un LLM pour interpréter l'intention ou la catégorie d'une requête, puis achemine l'entrée vers une tâche ou un agent spécialisé en aval.



Le flux de travail de routage est utilisé dans les scénarios où un agent doit rapidement classer l'intention d'entrée, le type de tâche ou le domaine, puis déléguer la demande à un sous-agent, un outil ou un flux de travail spécialisé. Il est particulièrement utile pour les agents de capacité, tels que ceux qui font office d'assistants généraux, de portes d'entrée des fonctions de l'entreprise ou d'interfaces d'IA orientées utilisateur couvrant plusieurs domaines.

Le routage est particulièrement efficace lorsque :

- Triage des demandes selon diverses tâches (par exemple, recherche, synthèse, réservation, calculs).
- Les entrées doivent être prétraitées ou normalisées avant d'entrer dans des flux de travail plus spécialisés.
- Les différents types de saisie (par exemple, images par rapport au texte, requêtes structurées par rapport aux requêtes non structurées) nécessitent un traitement personnalisé.
- Un agent agit comme un central conversationnel, déléguant des tâches à des agents spécialisés ou à des microservices.
- Ce flux de travail est courant dans les copilotes spécifiques à un domaine, les robots de support client, les routeurs de services d'entreprise et les agents multimodaux, où le dispatching intelligent détermine à la fois la qualité et l'efficacité du comportement des agents.

Fonctionnalités

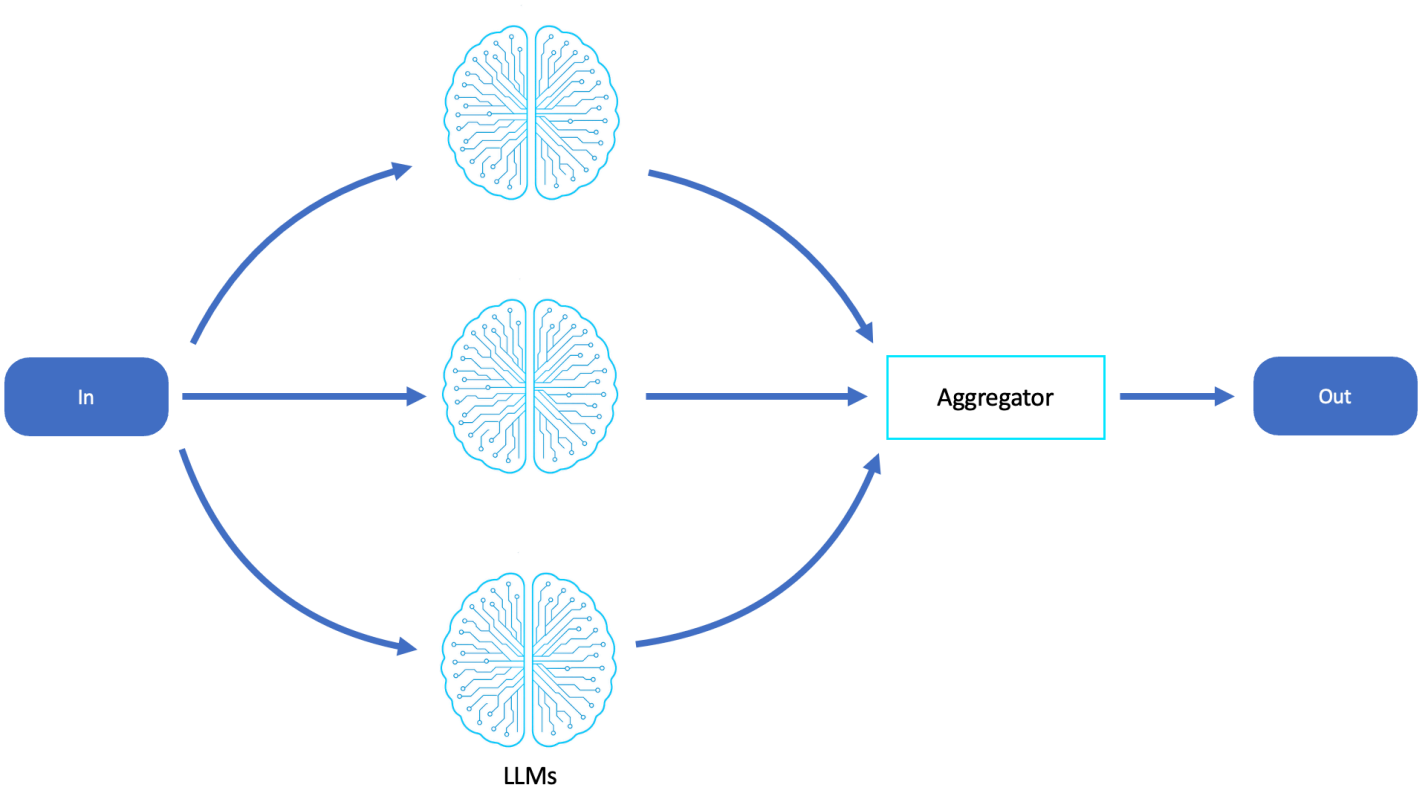
- Un LLM de premier passage agit en tant que répartiteur
- Les itinéraires peuvent invoquer des flux de travail distincts ou même d'autres modèles d'agents
- Prend en charge l'extension modulaire des capacités

Cas d’utilisation courants

- Assistants multidomaines (« s'agit-il d'une question juridique, médicale ou financière ? »)
- Arbres de décision améliorés grâce au raisonnement LLM
- Sélection d'outils dynamique (par exemple, recherche ou génération de code)

Workflow pour la parallélisation

Ce flux de travail consiste à décomposer une tâche en sous-tâches indépendantes qui peuvent être gérées simultanément par plusieurs appels ou agents LLM. Les résultats sont ensuite agrégés par programmation et synthétisés en un résultat.



Le flux de travail de parallélisation est utilisé lorsqu'une tâche peut être divisée en sous-tâches indépendantes et non séquentielles pouvant être traitées simultanément, ce qui améliore considérablement l'efficacité, le débit et l'évolutivité. Il est particulièrement puissant dans les espaces problématiques riches en données, orientés par lots ou à perspectives multiples où l'agent doit analyser ou générer du contenu à partir de plusieurs entrées.

La parallélisation est particulièrement efficace lorsque :

- Les sous-tâches ne dépendent pas des résultats intermédiaires des unes et des autres, ce qui leur permet de s'exécuter en parallèle sans coordination.
- Une tâche implique de répéter le même processus de raisonnement pour de nombreux éléments (par exemple, résumer plusieurs documents ou évaluer une liste d'options).
- De multiples hypothèses ou perspectives sont explorées en parallèle pour promouvoir la diversité, la créativité ou la robustesse.
- Vous devez réduire la latence pour les demandes à volume élevé ou à haute fréquence grâce à l'exécution simultanée du LLM.
- Ce flux de travail est couramment utilisé dans les agents de traitement de documents, les moteurs d'enquête ou de comparaison, les synthétiseurs par lots, les brainstormers multi-agents et les tâches de classification ou d'étiquetage évolutives, en particulier lorsque le raisonnement rapide et parallèle constitue un avantage en termes de performances.

Fonctionnalités

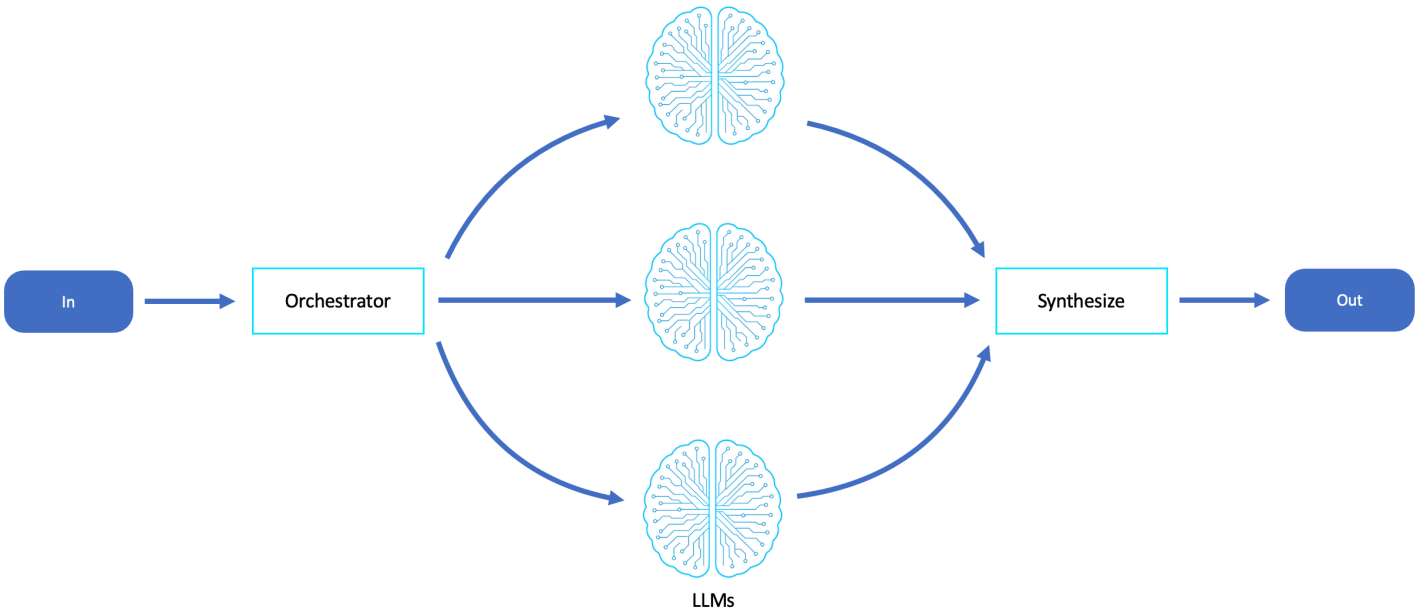
- Exécution parallèle de tâches LLM (en utilisant AWS Lambda AWS Fargate, ou un état de AWS Step Functions carte)
- Nécessite l'alignement, la validation ou la déduplication des résultats au stade de la synthèse
- Well-suited pour les boucles d'agents apatrides

Cas d’utilisation courants

- Analyse de plusieurs documents ou perspectives en parallèle
- Génération de brouillons, de résumés ou de plans variés
- Accélérer le débit entre les tâches par lots

Flux de travail pour l'orchestration

Un agent d'orchestration central utilise un LLM pour planifier, décomposer et déléguer des sous-tâches à des agents ou modèles spécialisés, chacun ayant un rôle ou une expertise dans un domaine spécifique. Cela reflète les structures de l'équipe humaine et soutient les comportements émergents de plusieurs agents.



Le flux de travail d'orchestration est idéal pour les scénarios complexes, hiérarchiques ou multidisciplinaires, nécessitant une décomposition structurée et une exécution spécialisée. Il est particulièrement adapté aux tâches qui nécessitent une division du travail, où les différents sous-composants d'une tâche sont mieux gérés par des agents dotés de capacités, de connaissances ou d'ensembles d'outils distincts.

Ce flux de travail est particulièrement efficace lorsque :

- Les tâches peuvent être divisées en sous-tâches dont la portée, le type ou le raisonnement varient (par exemple, planifier, rechercher, implémenter et tester).
- Un LLM ou un méta-agent doit coordonner les autres agents, suivre les progrès et synthétiser les résultats.
- Vous souhaitez modulariser les responsabilités des agents, en permettant l'évolutivité, la réutilisation et un réglage spécialisé.
- Le système nécessite un comportement basé sur les rôles, imitant la façon dont les équipes humaines (par exemple, les chefs de projet, les développeurs et les réviseurs) opèrent en collaboration.

L'orchestration est idéale pour les agents de planification multitours, les copilotes de développement logiciel, les agents de processus d'entreprise et les exécuteurs de projets autonomes. Il est particulièrement utile lors de la mise en œuvre de systèmes multi-agents qui nécessitent une répartition centralisée des tâches mais une logique d'exécution distribuée, permettant une extensibilité et un comportement plus explicable entre les couches d'agents.

Fonctionnalités

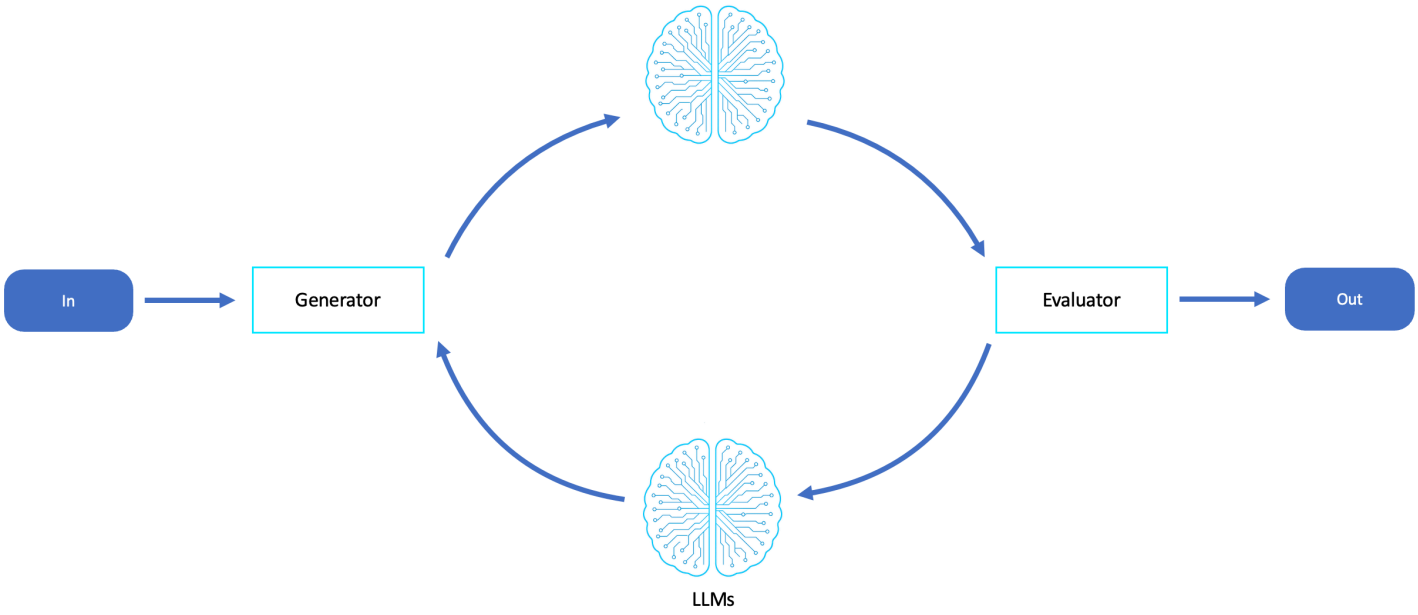
- Orchestrator exécute un méta-raisonnement sur les objectifs
- Les agents de travail peuvent inclure l'accès aux outils, la mémoire ou des instructions spécifiques au domaine
- Peut être hiérarchique (c'est-à-dire délégation de tâches à plusieurs niveaux)

Cas d’utilisation courants

- Chefs de projet, chercheurs coordinateurs, rédacteurs et agents d'assurance qualité
- Copilotes de codage qui combinent la planification, l'exécution et les tests
- Agents qui supervisent les chaînes d'outils ou les modèles d'accès aux API

Flux de travail pour les évaluateurs et boucles d'affinage par réflexion

Ce flux de travail fournit une boucle de rétroaction dans laquelle un LLM génère un résultat et un autre évalue ou critique le résultat. Cela favorise l'autoréflexion, l'optimisation et les améliorations itératives.



Le flux de travail de l'évaluateur est idéal pour les scénarios dans lesquels la qualité, la précision et l'alignement des résultats sont importants et dans lesquels la génération en un seul passage est peu fiable ou insuffisante. Ce flux de travail excelle lorsque les agents doivent s'autocritiquer, itérer et affiner leurs résultats, soit pour répondre à des normes d'exactitude plus élevées, soit pour explorer des alternatives améliorées en fonction des commentaires.

Ce flux de travail est particulièrement efficace lorsque :

- Le résultat implique des mesures de qualité subjectives (par exemple, le style, le ton et la lisibilité) ou des critères objectifs (par exemple, exactitude, sécurité et performance).
- L'agent doit raisonner en faisant des compromis, évaluer les contraintes ou optimiser pour atteindre un objectif.
- Vous avez besoin d'une redondance et d'une assurance qualité intégrées, en particulier dans les domaines réglementés, destinés aux clients ou créatifs.
- Human-in-the-loop la révision coûte cher ou n'est pas disponible, et une validation autonome est souhaitée.

Ce flux de travail est utilisé pour la génération de contenu, la synthèse et la révision du code, l'application des politiques, le contrôle de l'alignement, le réglage des instructions et le post-traitement RAG. Il est également utile pour les agents qui s'améliorent eux-mêmes, où le feedback continu contribue à façonner de meilleures réponses au fil du temps afin de créer des boucles de décision fiables et autonomes.

Cas d’utilisation courants

- Red-team agents comparés aux agents de l'équipe bleue
- Agents qui génèrent, évaluent et révisent le code ou les plans
- Assurance qualité, détection des hallucinations et application du style

Fonctionnalités

- Supporte la génération et l'évaluation découplées à l'aide de différents modèles (par exemple, Claude pour la génération et Mistral pour l'évaluation)
- Les commentaires sont structurés et utilisés pour demander des résultats révisés
- Supporte plusieurs itérations ou seuils de convergence

Conclusion

Les LLM constituent le cœur cognitif des agents logiciels modernes, mais l'invocation de modèles bruts ne suffit pas pour obtenir une intelligence ciblée, robuste et contrôlable. Pour passer de la génération de résultats à un raisonnement structuré et à un comportement aligné sur les objectifs, les LLM doivent être intégrés dans des modèles de flux de travail intentionnels qui définissent la manière dont les modèles traitent les entrées, gèrent les contextes et coordonnent les actions.

Les flux de travail LLM introduisent les bases nécessaires à la création du module cognitif d'un agent :

- Le chaînage rapide décompose le raisonnement complexe en étapes modulaires et contrôlables.
- Le routage permet une classification intelligente des tâches et une délégation ciblée.
- La parallélisation accélère le débit et favorise la diversité des raisonnements.
- L'orchestration des agents structure la collaboration multi-agents par le biais de la décomposition des tâches et de l'exécution basée sur les rôles.
- L'évaluateur (boucle reflect-refine) permet l'auto-amélioration, le contrôle de la qualité et le contrôle de l'alignement.

Chaque flux de travail représente un modèle composable qui peut être adapté aux besoins de l'agent, à la complexité de la tâche et aux attentes de l'utilisateur. Ces flux de travail ne s'excluent pas mutuellement. Ce sont des éléments de base souvent combinés dans des architectures hybrides

qui prennent en charge le raisonnement dynamique, la coordination multi-agents et une fiabilité de niveau entreprise.

Au fur et à mesure que vous passerez au chapitre suivant sur les modèles de flux de travail agentiques, ces flux de travail LLM réapparaîtront sous forme de structures intégrées dans des systèmes plus importants, prenant en charge la délégation d'objectifs, l'orchestration des outils, les boucles de décision et l'autonomie du cycle de vie. La maîtrise de ces flux de travail LLM est essentielle pour concevoir des agents logiciels qui ne se contentent pas de prédire le texte, mais aussi de raisonner, de s'adapter et d'agir de manière ciblée.

Modèles de flux de travail agentic

Les modèles de flux de travail agentic intègrent des agents logiciels modulaires à des flux de travail structurés basés sur des modèles de langage large (LLM), permettant ainsi un raisonnement et une action autonomes. Tout en s'inspirant des architectures traditionnelles sans serveur et pilotées par les événements, ces modèles font passer la logique de base du code statique aux LLM-augmented agents, offrant ainsi une meilleure adaptabilité et une prise de décision contextuelle améliorée. Cette évolution transforme les architectures cloud classiques des systèmes déterministes en systèmes capables d'interprétation dynamique et d'augmentation intelligente, tout en maintenant les principes fondamentaux d'évolutivité et de réactivité.

Dans cette section

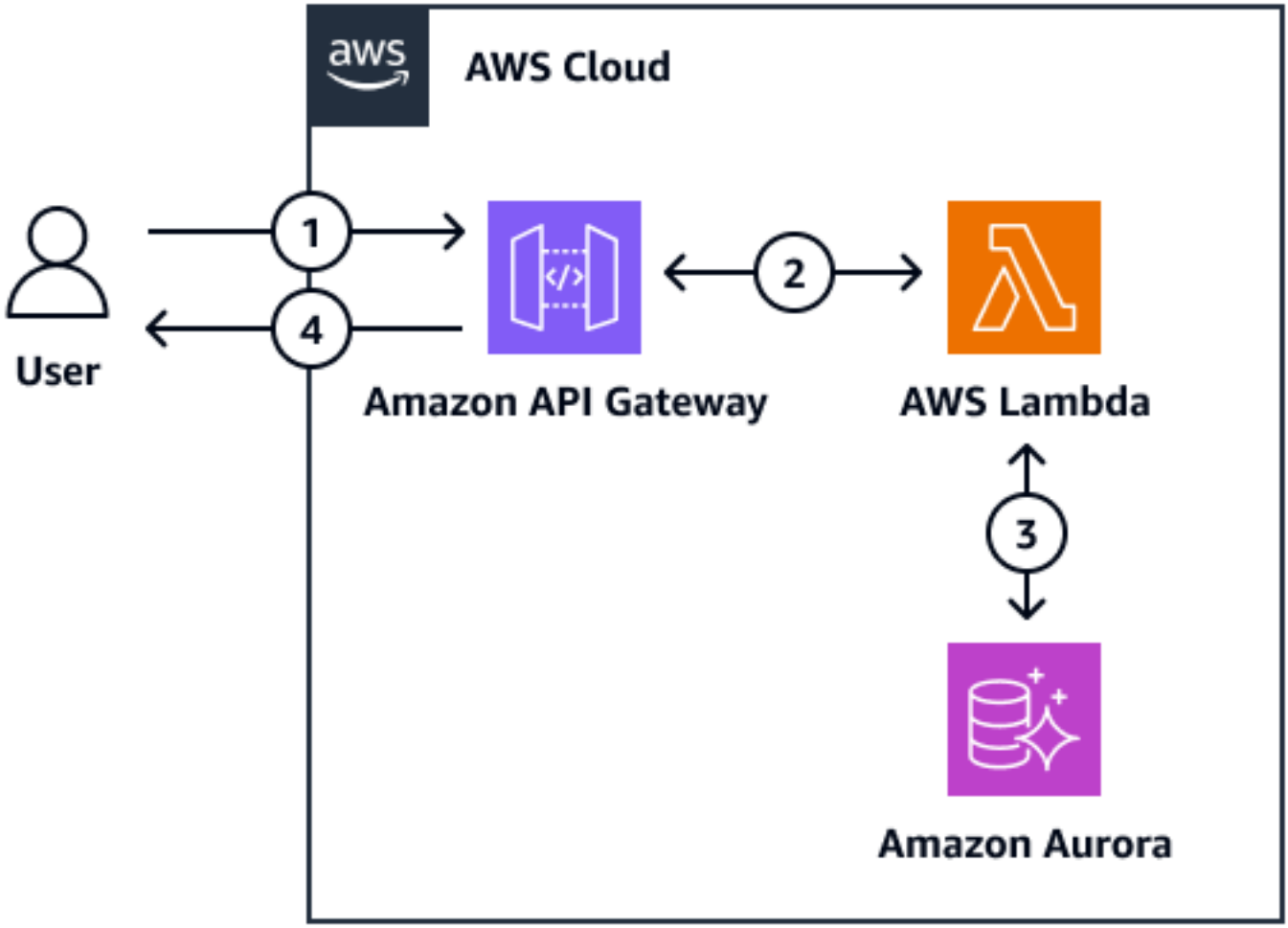
- Des systèmes pilotés par les événements aux systèmes augmentés par la cognition
- Motifs de saga à enchaînement rapide
- Acheminement des modèles d'expédition dynamiques
- Modèles de parallélisation et de diffusion
- Modèles d'orchestration de Saga
- L'évaluateur reflète et affine les modèles de boucles
- Conception de flux de travail agentiques sur AWS
- Conclusion

Des systèmes pilotés par les événements aux systèmes augmentés par la cognition

Les architectures cloud modernes, en particulier celles basées sur des principes sans serveur et pilotés par les événements, se sont traditionnellement appuyées sur des modèles tels que le routage, le fan-out et l'enrichissement pour créer des systèmes réactifs et évolutifs. Les systèmes d'IA agentic s'appuient sur ces fondements tout en les recadrant autour du LLM-augmented raisonnement et de la flexibilité cognitive. Cette approche permet des capacités de résolution de problèmes et d'automatisation plus sophistiquées, susceptibles de révolutionner la manière dont les tâches complexes sont gérées dans les environnements cloud.

Event-driven architecture

Le schéma suivant montre un système distribué typique :

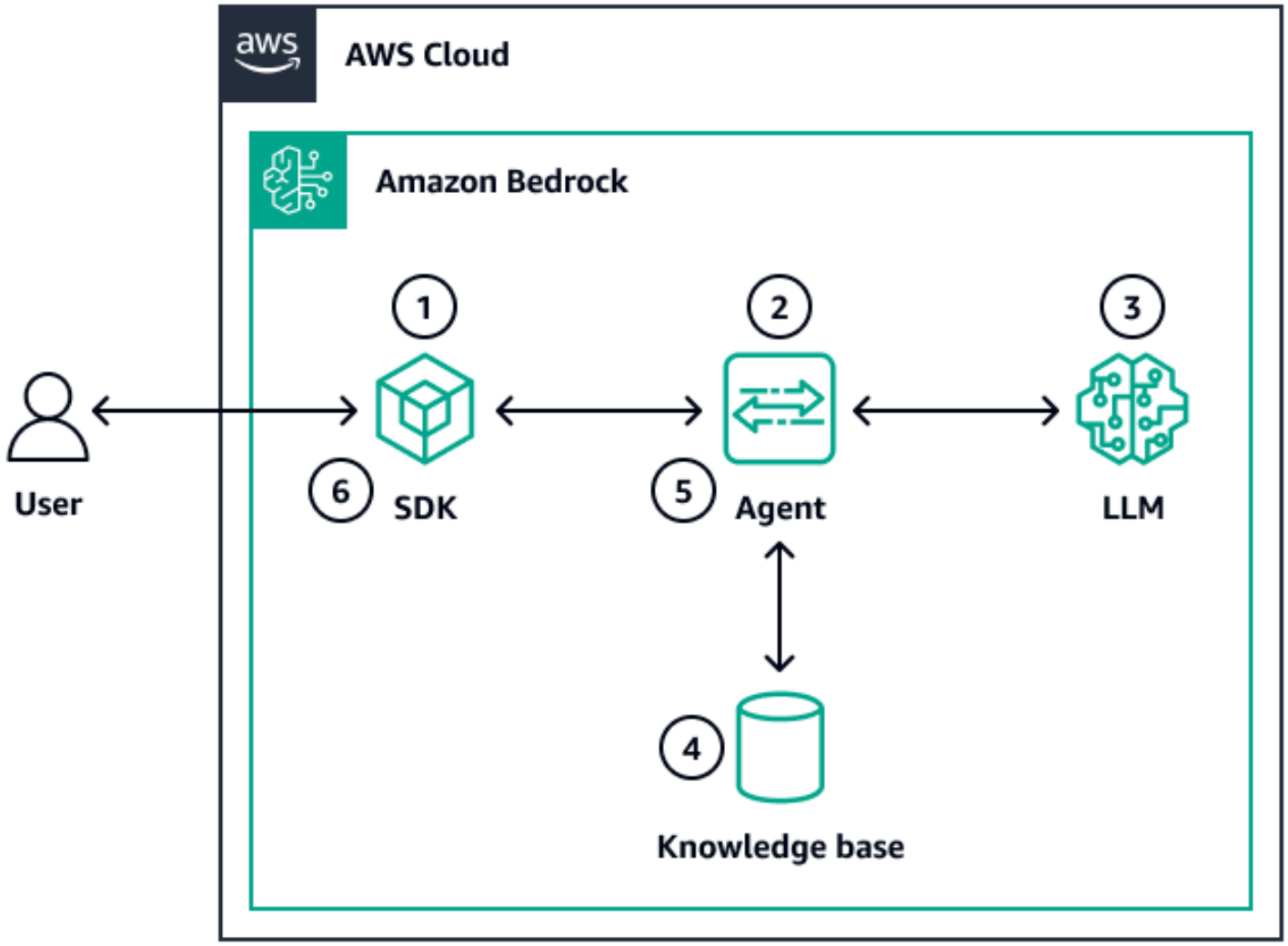


1. Un utilisateur envoie une demande à Amazon API Gateway.
2. Amazon API Gateway achemine la demande vers une AWS Lambda fonction.
3. AWS Lambda enrichit les données en interrogeant une base de données Amazon Aurora
4. Amazon API Gateway renvoie la charge utile enrichie à l'appelant.

Cette structure est à la fois fiable et évolutive, mais elle est fondamentalement statique. Les règles métier et les chemins logiques doivent être explicitement codés, et l'adaptation à des contextes changeants ou à des informations incomplètes est limitée.

Cognition-augmented flux de travail

Les architectures agentic ajoutent une augmentation cognitive à un système piloté par les événements. Le schéma suivant montre un équivalent agentique :



1. Un utilisateur soumet une requête via un SDK ou un appel d'API.
2. Un agent Amazon Bedrock reçoit la requête.
3. L'agent interprète la requête en invoquant un LLM
4. L'agent effectue un enrichissement sémantique en effectuant des recherches dans la base de connaissances Amazon Bedrock ou dans d'autres sources de données externes.
5. Le LLM synthétise une réponse riche en contexte et alignée sur les objectifs.
6. Le système renvoie une réponse synthétisée à l'utilisateur.

Dans ce flux, le LLM utilise la logique, comprend l'intention, récupère et combine le contexte pertinent, puis décide de la meilleure façon de répondre. Ce modèle reflète le modèle d'enrichissement traditionnel, dans lequel les messages sont complétés par des données externes avant d'être acheminés ultérieurement. Dans les systèmes agentiques, toutefois, cet enrichissement n'est pas une recherche statique. L'enrichissement est plutôt dynamique, guidé sémantiquement et motivé par un objectif.

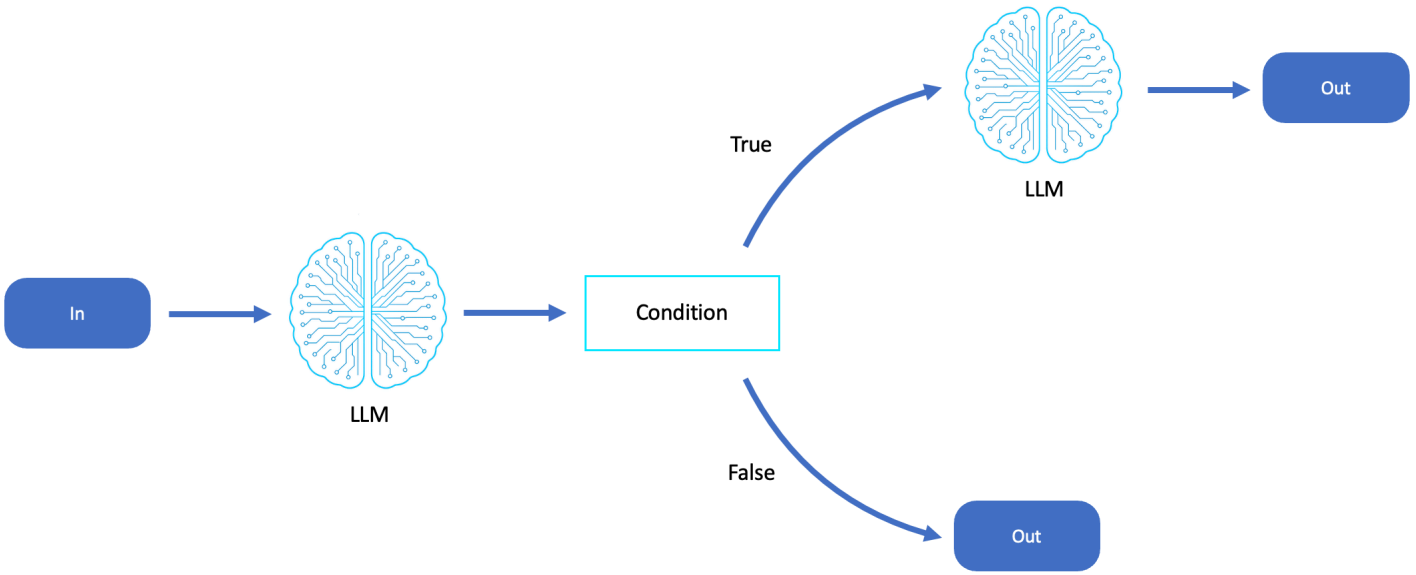
Informations de base

Chaque flux de travail LLM peut être mappé à un modèle de flux de travail agentic, qui reflète et fait évoluer les styles d'architecture traditionnels axés sur les événements. L'un des éléments de base des flux de travail agentiques est la capacité d'enrichir le contexte d'un LLM avec des données, des outils et de la mémoire. Cela crée une boucle de raisonnement informée, adaptative et alignée sur l'intention de l'utilisateur. Alors que les systèmes traditionnels enrichissent les messages avec des données de recherche, les systèmes agentic permettent aux logiciels d'agir moins comme des scripts que comme des collaborateurs intelligents.

Motifs de saga à enchaînement rapide

En réinventant le chaînage rapide LLM sous la forme d'une saga axée sur les événements, nous ouvrons la voie à un nouveau modèle opérationnel : les flux de travail deviennent distribués, récupérables et coordonnés sémantiquement entre les agents autonomes. Chaque étape de réponse rapide est redéfinie comme une tâche atomique, émise en tant qu'événement, consommée par un agent dédié et enrichie de métadonnées contextuelles.

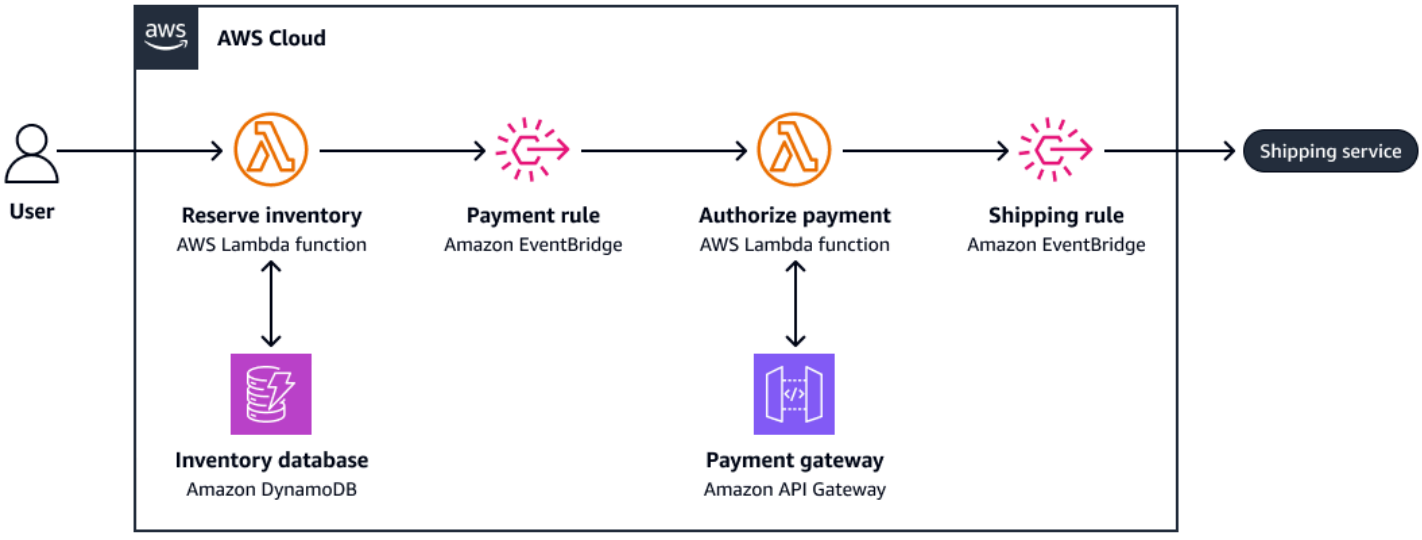
Le schéma suivant est un exemple de chaînage d'invite LLM :



Chorégraphie de saga

Le schéma chorégraphique de la saga est une approche de mise en œuvre dans les systèmes distribués qui n'a pas de coordinateur central. Au lieu de cela, chaque service ou composant publie des événements qui déclenchent l'action suivante du flux de travail. Ce modèle est largement utilisé dans les systèmes distribués pour gérer les transactions entre plusieurs services. Dans une saga, le système exécute une série de transactions locales coordonnées. En cas d'échec, le système déclenche des actions de compensation pour maintenir la cohérence.

Le schéma suivant est un exemple de chorégraphie de saga :



1. Inventaire de réserve

2. Autoriser le paiement
3. Créer un bon de commande d'expédition

Si l'étape 3 échoue, le système invoque des actions compensatoires (par exemple, annuler un paiement ou libérer un inventaire).

Ce modèle est particulièrement utile dans les architectures axées sur les événements où les services sont faiblement couplés et où les états doivent être résolus de manière cohérente au fil du temps, même en cas de défaillance partielle.

Schéma de chaînage rapide

Le chaînage rapide ressemble au schéma de la saga, à la fois en termes de structure et d'objectif. Il exécute une série d'étapes de raisonnement qui se développent de manière séquentielle tout en préservant le contexte et en autorisant les annulations et les révisions.

Chorégraphie d'agent

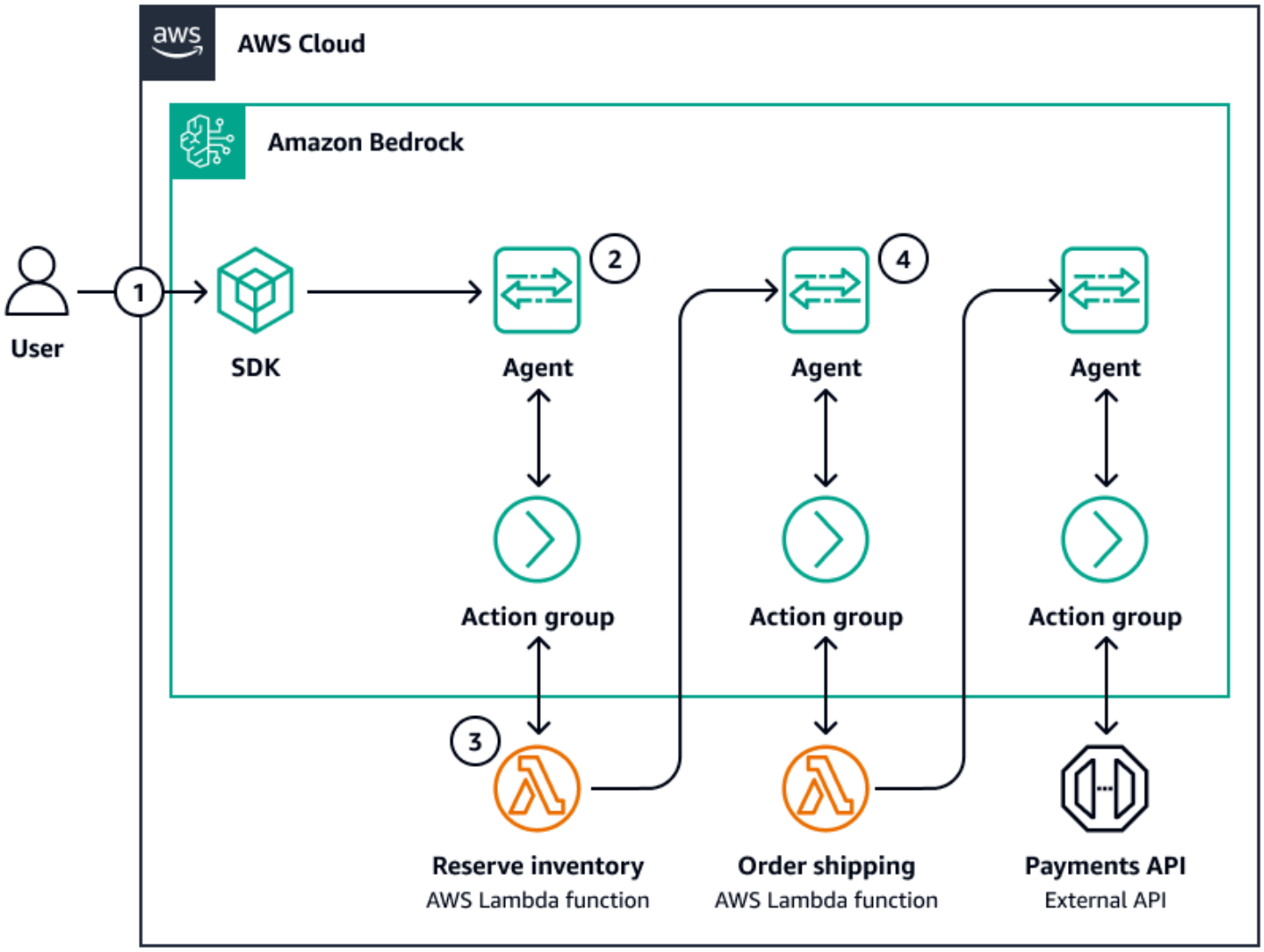
1. LLM interprète une requête utilisateur complexe et génère une hypothèse
2. Le LLM élabore un plan pour résoudre la tâche
3. LLM exécute une sous-tâche (par exemple, en utilisant un appel d'outil ou en récupérant des connaissances)
4. Le LLM affine le résultat ou revoit une étape précédente s'il juge le résultat insatisfaisant

Si un résultat intermédiaire est erroné, le système peut effectuer l'une des opérations suivantes :

- Réessayez les étapes en utilisant une approche différente
- Revenir à une demande précédente et replanifier
- Utiliser une boucle d'évaluateur (par exemple, à partir du modèle évaluateur-optimiseur) pour détecter et corriger les défaillances

À l'instar du modèle de saga, le chaînage rapide permet des mécanismes de progression partielle et de retour en arrière. Cela se fait par le biais d'un raffinement et d' LLM-directed une correction itératifs plutôt que par la compensation des transactions de base de données.

Le schéma suivant est un exemple de chorégraphie d'agent :



1. Un utilisateur soumet une requête par le biais d'un SDK.
2. Un agent Amazon Bedrock orchestre le raisonnement de la manière suivante :
   - Interprétation (LLM)
   - Planification (LLM)
   - Exécution via un outil ou une base de connaissances
   - Construction de la réponse
3. Si un outil échoue ou ne renvoie pas suffisamment de données, l'agent peut replanifier ou reformuler la tâche de manière dynamique.
4. La mémoire (par exemple, un magasin de vecteurs à court terme) peut conserver son état d'une étape à l'autre

Plats à emporter

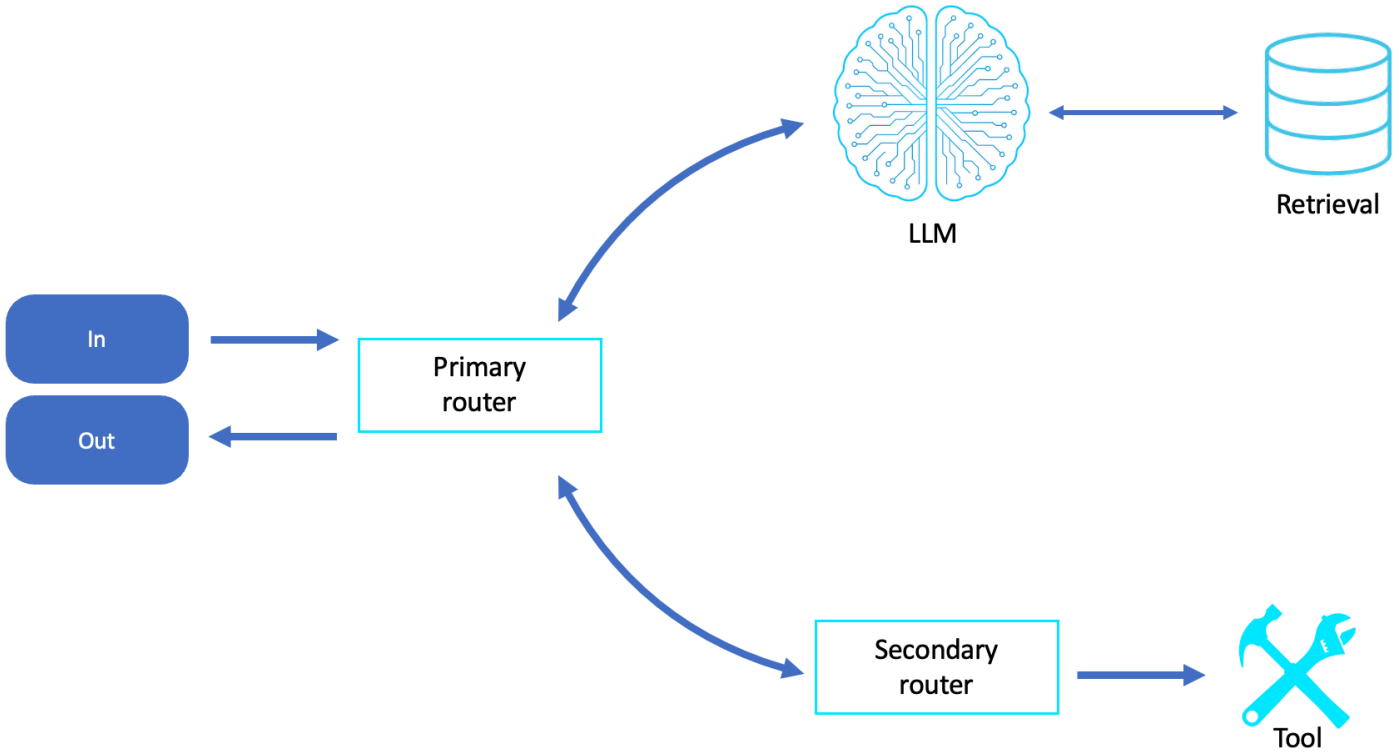
Alors que le modèle en saga gère les appels de service distribués selon une logique de compensation, le chaînage rapide gère les tâches de raisonnement grâce à un séquençage réflexif et à une replanification adaptative. Les deux systèmes permettent de réaliser des progrès progressifs, de décentraliser les points de décision et de remédier aux défaillances, et tout cela grâce à un raisonnement éclairé plutôt qu'à un retour en arrière rigide.

Le chaînage rapide introduit le raisonnement transactionnel, qui est l'équivalent cognitif des sagas. C'est-à-dire que chaque « pensée » est réévaluée, révisée ou abandonnée dans le cadre d'un dialogue plus large axé sur les objectifs.

Acheminement des modèles d'expédition dynamiques

Dans les systèmes agentiques modernes, où les tâches vont de l'analyse de documents à la génération autonome de logiciels, la capacité d'acheminer dynamiquement les demandes vers le modèle ou l'agent de grande langue (LLM) le plus performant devient essentielle. La logique de routage statique, souvent intégrée dans des scripts d'orchestration ou des couches d'API, ne possède pas l'adaptabilité requise pour les environnements multi-modèles et multi-capacités en temps réel. Pour y remédier, les flux de travail de routage LLM peuvent être transformés en une architecture axée sur les événements qui tire parti d'un modèle d'expédition dynamique, transformant les appels LLM en événements routés intelligemment et sensibles au contexte.

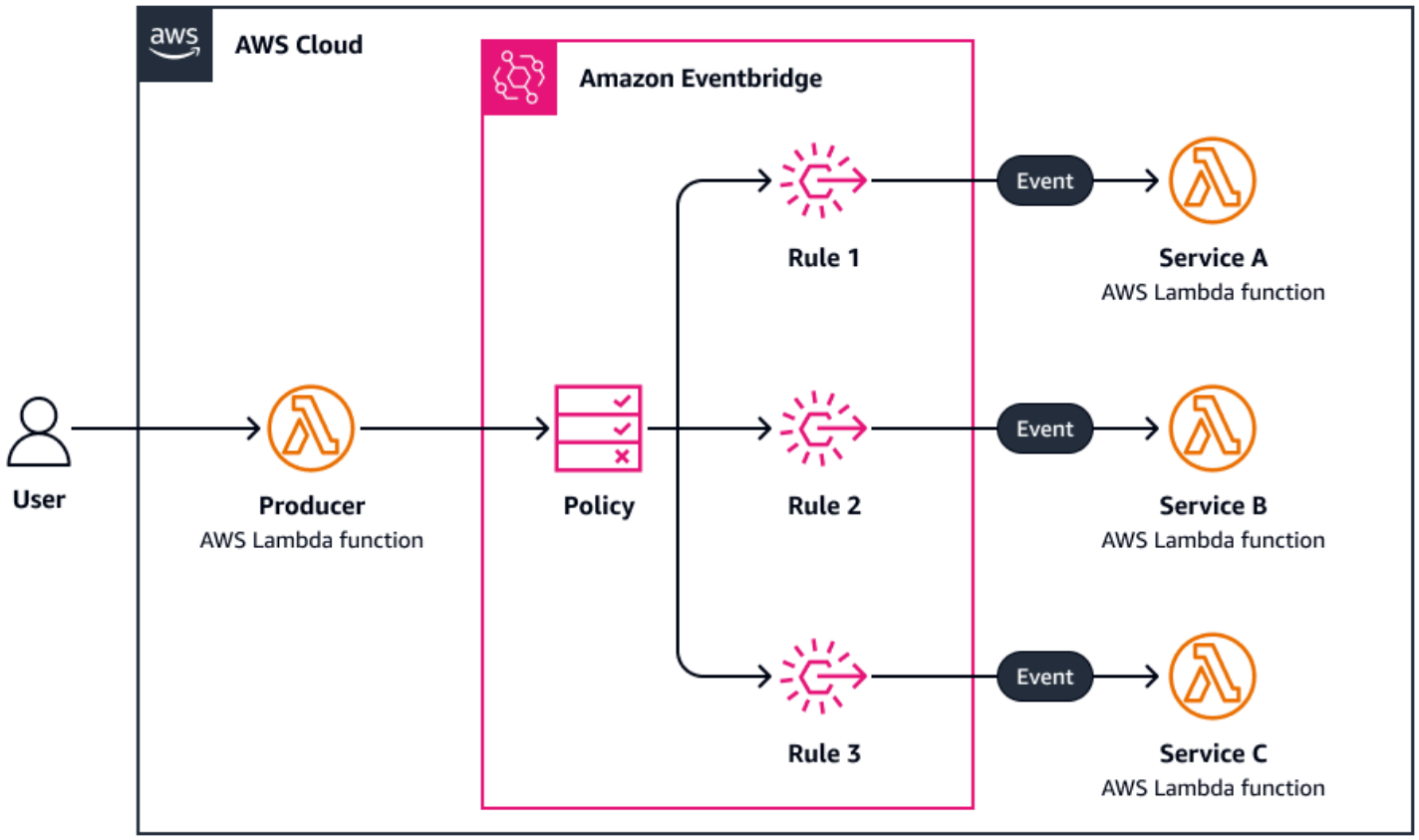
Le schéma suivant est un exemple de routage LLM :



Expédition dynamique

Dans les systèmes distribués traditionnels, le modèle de distribution dynamique sélectionne et invoque des services spécifiques au moment de l'exécution en fonction des attributs des événements entrants, tels que le type d'événement, la source et la charge utile. Ceci est généralement mis en œuvre à l'aide d'Amazon EventBridge, qui peut évaluer et acheminer les événements entrants vers des cibles appropriées (par exemple, des AWS Lambda fonctions AWS Step Functions ou des tâches Amazon Elastic Container Service).

Le schéma suivant est un exemple de répartition dynamique :



1. Une application émet un événement (par exemple, {"type » : « OrderCreated », « priority » : « high"}).
2. Amazon EventBridge évalue l'événement par rapport à ses règles de routage.
3. Sur la base des attributs d'un événement, le système effectue des envois dynamiques vers les sites suivants :
   - `HighPriorityOrderProcessor`(service A)
   - `StandardOrderProcessor`(service B)
   - `UpdateOrderProcessor`(service C)

Ce modèle prend en charge le couplage souple, la spécialisation basée sur le domaine et l'extensibilité du temps d'exécution. Cela permet aux systèmes de répondre intelligemment à l'évolution des exigences et de la sémantique des événements.

LLM-based routage

Dans les systèmes agentic, le routage effectue également une délégation dynamique des tâches, mais au lieu des EventBridge règles Amazon ou des filtres de métadonnées, le LLM classe et

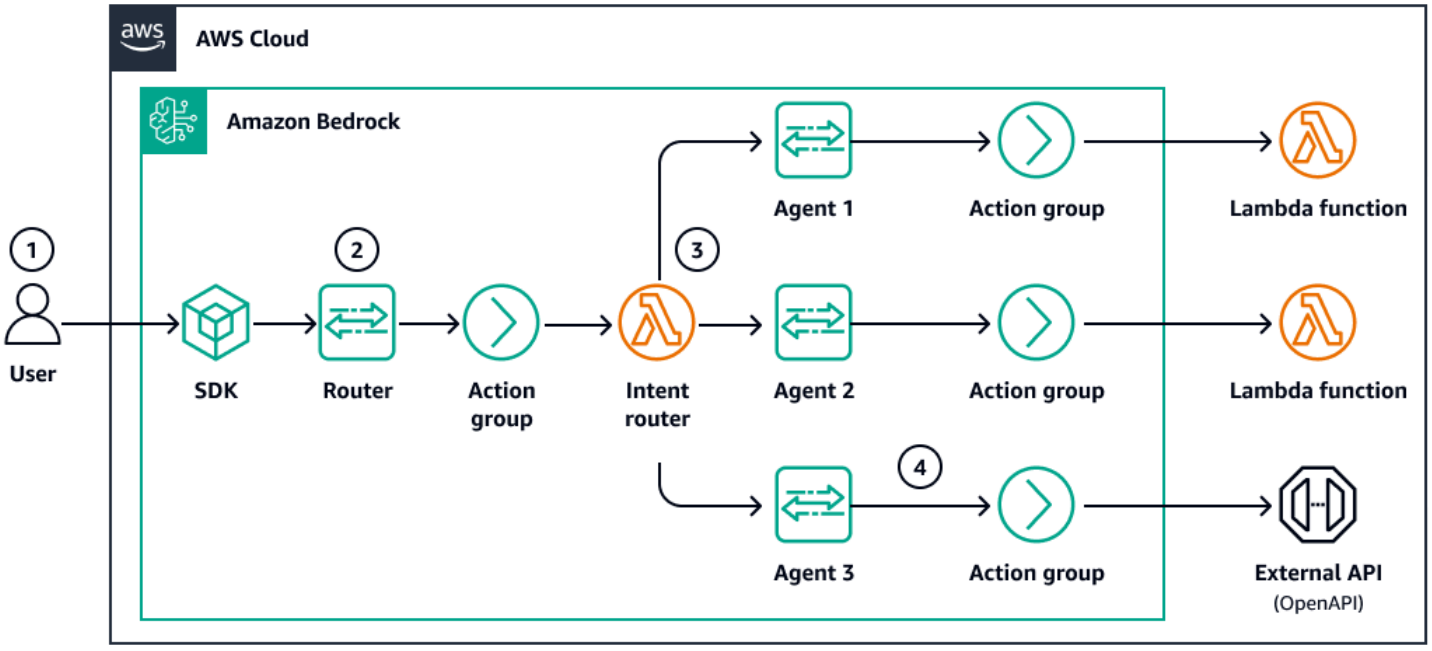
interprète l'intention de l'utilisateur en langage naturel. Le résultat est une forme de répartition flexible, sémantique et adaptative.

Routeur agent

Cette architecture permet une distribution riche basée sur l'intention sans schémas ni types d'événements prédéfinis, ce qui est idéal pour les entrées non structurées et les requêtes complexes.

1. Un utilisateur soumet la demande « Pouvez-vous m'aider à revoir les termes de mon contrat ? »
2. Le LLM interprète cela comme une tâche de documentation juridique.
3. L'agent achemine la tâche vers un ou plusieurs des sites suivants :
   - Modèle de demande de révision de contrat
   - Sous-agent de raisonnement juridique
   - outil d'analyse de documents

Le schéma suivant est un exemple d'agent-routeur :



1. Un utilisateur soumet une demande en langage naturel par le biais d'un SDK.
2. Un agent Amazon Bedrock utilise un LLM pour classer la tâche (par exemple, juridique, technique ou de planification).
3. L'agent achemine dynamiquement la tâche via un groupe d'action pour appeler l'agent requis :

- Domain-specific agent
- Chaîne d'outils spécialisée
- Configuration rapide personnalisée

4. Le gestionnaire sélectionné traite la tâche et renvoie une réponse personnalisée.

Plats à emporter

Alors que l'expédition dynamique traditionnelle utilise EventBridge les règles Amazon pour le routage en fonction d'attributs d'événements structurés, le routage agentic utilise des LLM pour classer et acheminer sémantiquement les tâches en fonction de leur signification et de leur intention. Cela augmente la flexibilité du système en permettant ce qui suit :

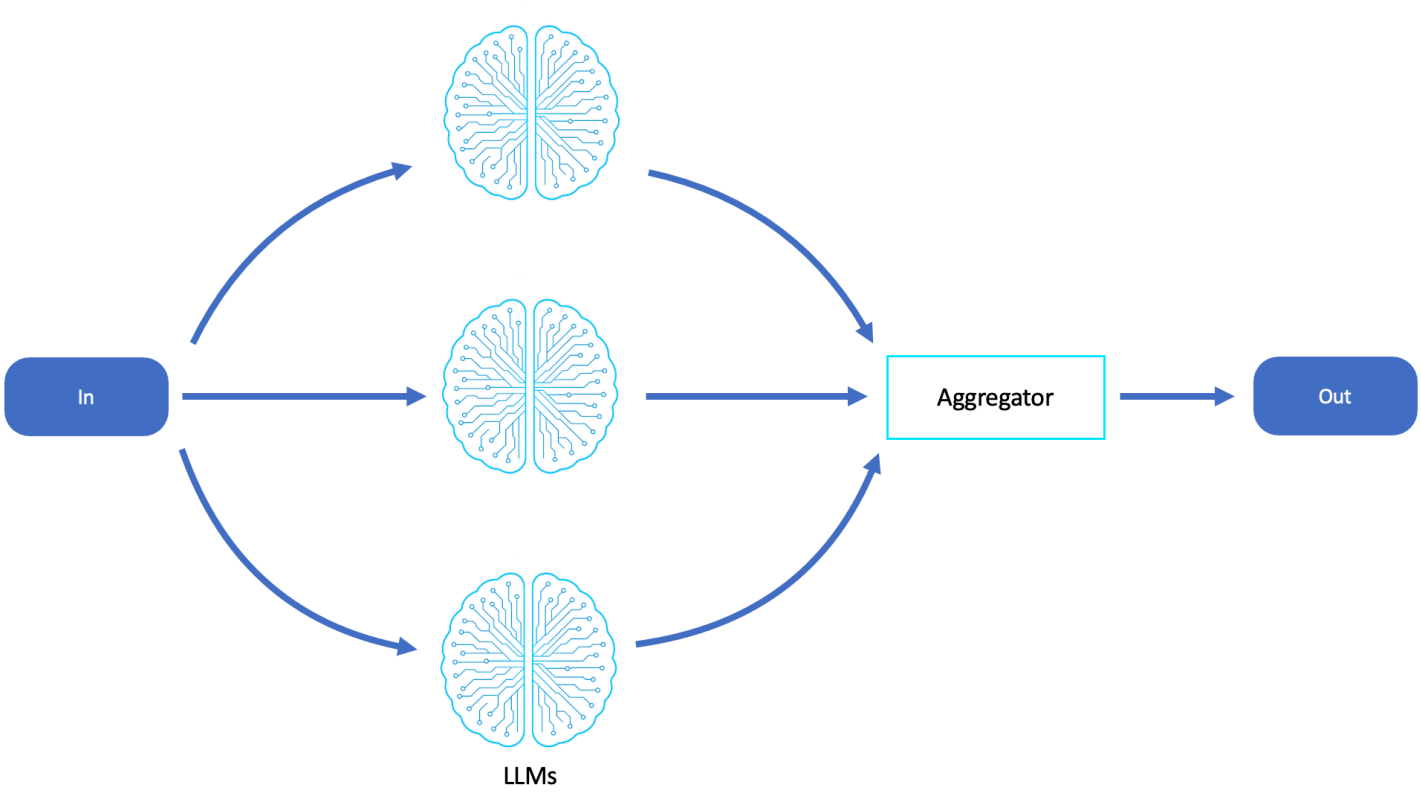
- Compréhension plus large des entrées
- Solution de repli intelligente et sélection d'outils
- Extensibilité naturelle grâce à de nouveaux rôles d'agent ou à des styles rapides

Le routage agentic remplace les règles rigides par un dispatching cognitif dynamique, qui permet aux systèmes d'évoluer en fonction du langage plutôt que du code.

Modèles de parallélisation et de diffusion

De nombreuses tâches avancées de raisonnement et de génération, telles que la synthèse de documents volumineux, l'évaluation de plusieurs solutions ou la comparaison de points de vue variés, tirent parti de l'exécution parallèle des instructions. Les flux de travail séquentiels traditionnels sont insuffisants lorsque l'évolutivité, la réactivité et la tolérance aux pannes sont requises. Pour surmonter ce problème, LLM-based la parallélisation peut être réinventée à l'aide d'un modèle de collecte par dispersion piloté par des événements, dans lequel les tâches sont réparties dynamiquement vers des agents autonomes et les résultats synthétisés de manière intelligente.

Le schéma suivant est un exemple de flux de travail de parallélisation LLM :



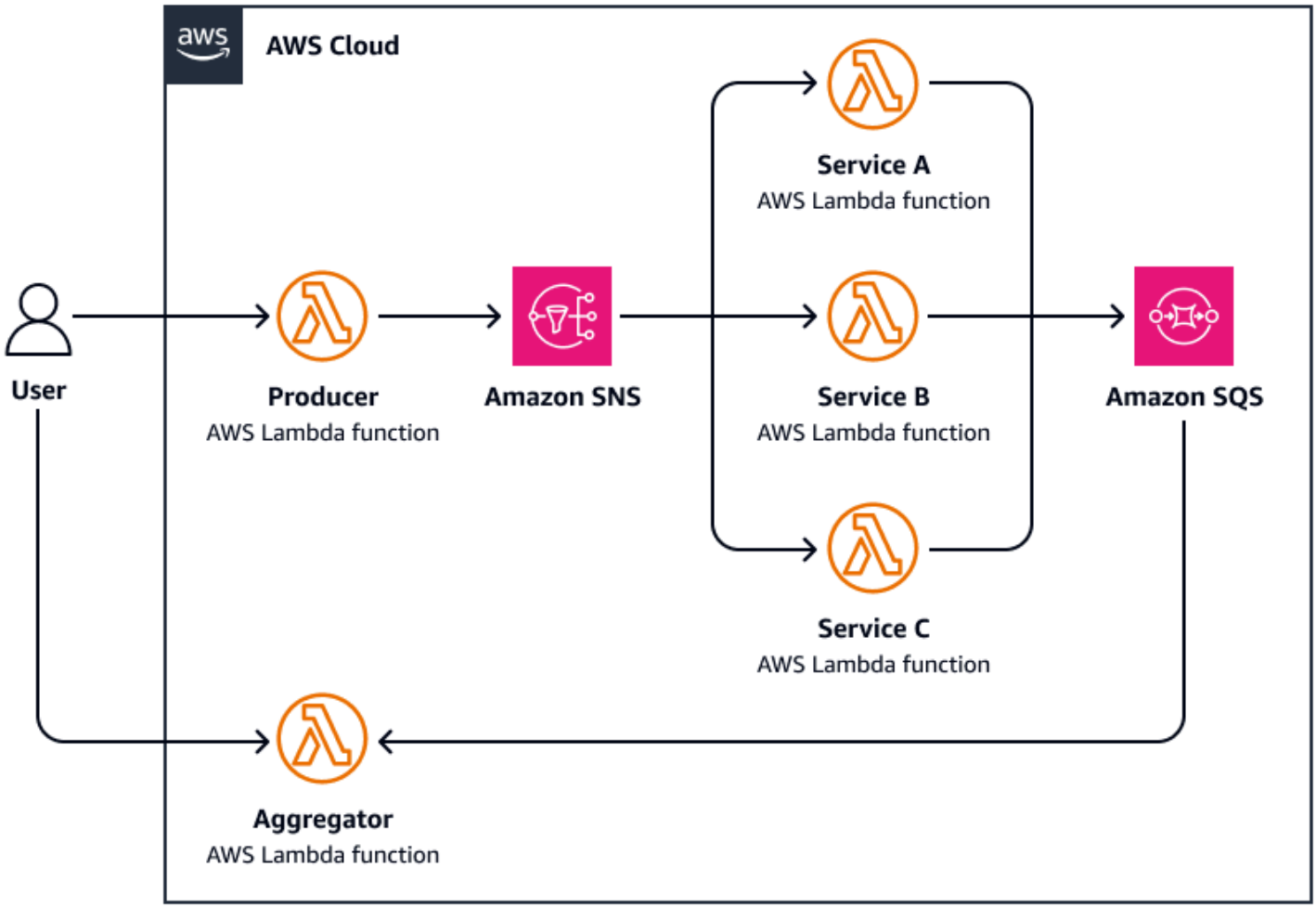
Scatter-gather

Dans les systèmes distribués, un modèle de collecte par dispersion envoie des tâches à plusieurs services ou unités de traitement en parallèle, attend leurs réponses, puis agrège les résultats dans une sortie consolidée. Contrairement au fan-out, le scatter-gather est coordonné car il attend des réponses et applique généralement une logique pour combiner, comparer et sélectionner les résultats.

Les implémentations courantes pour la parallélisation et le scatter-gather sont les suivantes :

- AWS Step Functions mapper un état pour l'exécution de tâches en parallèle
- AWS Lambda avec simultanéité, coordination des résultats de plusieurs fonctions invoquées
- Amazon EventBridge avec des identifiants de corrélation et des flux de travail d'agrégation
- Modèle de contrôleur personnalisé pour gérer le fan-out et recueillir des résultats à l'aide d'Amazon Simple Storage Service (Amazon S3), d'Amazon DynamoDB ou de files d'attente

Le schéma suivant est un exemple de scatter-gather :



1. Un utilisateur envoie une demande à une fonction de coordination centrale qui répartit la tâche en publiant des messages parallèles sur une rubrique Amazon Simple Notification Service (Amazon SNS).
2. Chaque message inclut les métadonnées des tâches et est acheminé vers un travailleur AWS Lambda spécialisé.
3. Chaque collaborateur traite AWS Lambda indépendamment la sous-tâche qui lui est assignée (par exemple, interroger une API externe, traiter un document et analyser des données).
4. Les résultats sont écrits sur une couche de stockage commune, telle qu'Amazon Simple Queue Service (Amazon SQS).
5. La fonction d'agrégation attend que toutes les réponses soient terminées, puis elle effectue les opérations suivantes :
   - Rassemble et agrège les résultats (par exemple, fusionne les résumés, sélectionne les meilleures correspondances)

- Envoie une réponse finale ou déclenche un flux de travail en aval

Les cas d'utilisation courants des modèles de dispersion sont les suivants :

- Recherche fédérée
- Moteurs de comparaison de prix
- Analyse des données agrégées
- Inférence multimodèle

LLM-based parallélisation (cognition basée sur la diffusion et la collecte)

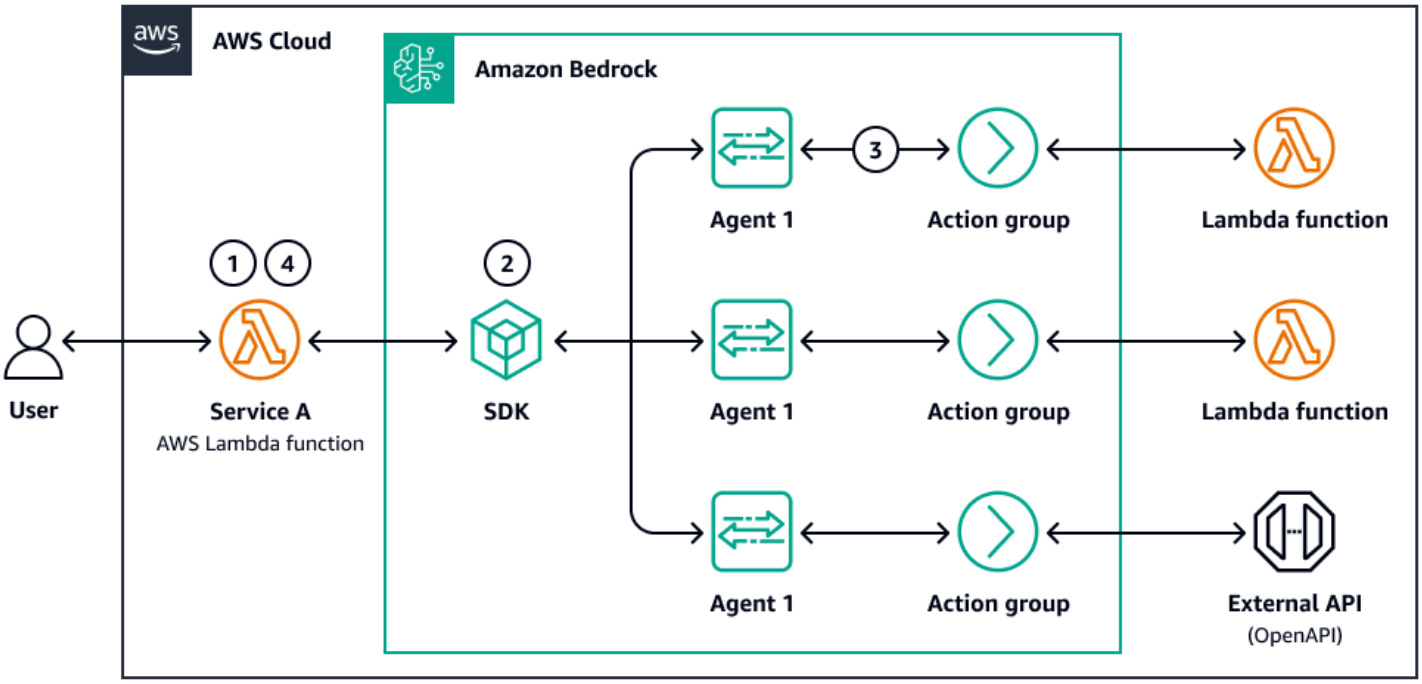
Dans les systèmes agentiques, la parallélisation reflète étroitement la diffusion en répartissant les sous-tâches entre plusieurs appels ou agents LLM, chacun abordant indépendamment une partie du problème. Les résultats renvoyés sont collectés et synthétisés par un processus d'agrégation, qui est souvent un autre LLM ou agent de contrôle.

Parallélisation des agents

1. Un agent soumet une demande « Résumez les informations issues de ces 10 rapports ».
2. Il répartit les rapports en 10 tâches de synthèse LLM parallèles.
3. Lorsqu'il renvoie tous les résumés, l'agent effectue les opérations suivantes :
   - Regroupe les résumés dans un briefing unifié
   - Identifie les thèmes ou les contradictions
   - Envoie la sortie synthétisée à l'utilisateur

Ce flux de travail agentique permet un raisonnement parallèle évolutif, modulaire et adaptatif. C'est idéal pour les cas d'utilisation qui nécessitent un débit cognitif élevé.

Le schéma suivant est un exemple de parallélisation d'agents :



1. Un utilisateur soumet une requête ou un ensemble de documents en plusieurs parties.
2. Un contrôleur AWS Lambda ou une fonction d'étape distribue les sous-tâches. Chaque tâche appelle un appel ou un sous-agent Amazon Bedrock LLM avec sa propre invite.
3. Lorsque les appels et les sous-tâches sont terminés, les résultats sont stockés (par exemple, dans Amazon S3 ou dans une mémoire), et une étape d'agrégation fusionne, compare ou filtre les sorties.
4. Le système renvoie la réponse finale à l'utilisateur ou à l'agent en aval.

Ce système comporte une boucle de raisonnement distribuée avec traçabilité, tolérance aux pannes et logique optionnelle de pondération ou de sélection des résultats.

Plats à emporter

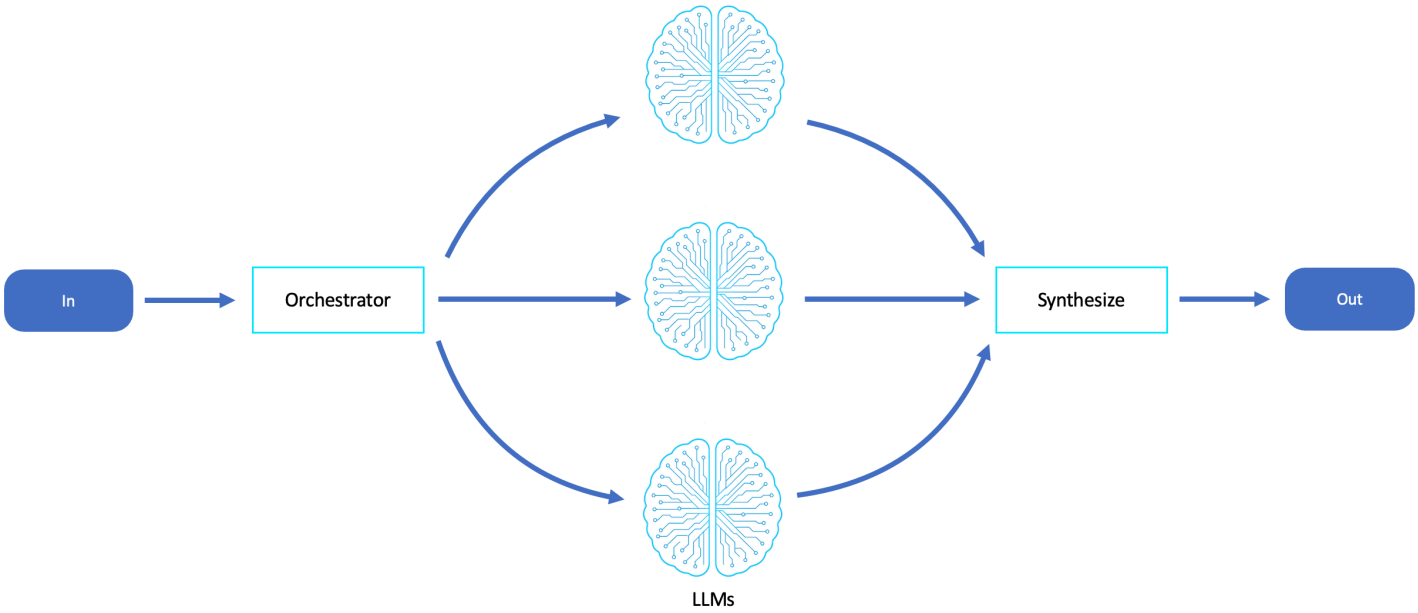
La parallélisation agentique utilise des modèles de dispersion pour distribuer les tâches LLM, permettant ainsi un traitement parallèle et une synthèse intelligente des résultats.

Modèles d'orchestration de Saga

Alors que les flux de travail pilotés par les LLM deviennent de plus en plus complexes, impliquant des chaînes rapides, des étapes de traitement des données, des invocations d'outils et la collaboration

des agents, le besoin d'une orchestration intelligente devient essentiel. Plutôt que de s'appuyer sur des scripts étroitement couplés ou des flux d'exécution statiques prédéterminés, ces flux de travail peuvent être mis en œuvre sous forme de modèles d'orchestration pilotés par des événements, permettant aux LLM-based systèmes de coordonner, de surveiller et d'adapter de manière dynamique des tâches en plusieurs étapes entre des agents autonomes.

Le schéma suivant est un exemple d'orchestrateur :



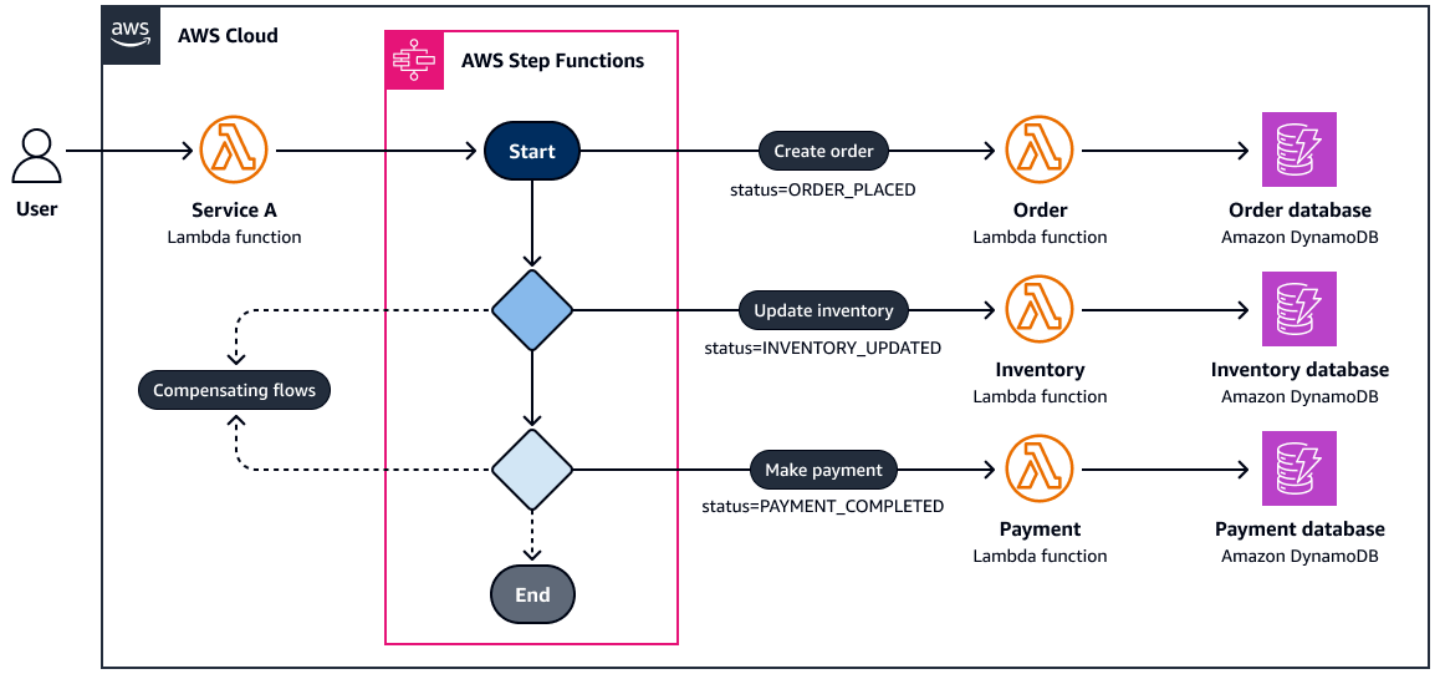
Orchestration d'événements

Dans les systèmes distribués traditionnels, l'orchestration d'événements fait référence à un modèle dans lequel un coordinateur central gère un flux de travail complexe en dirigeant explicitement le flux de contrôle entre plusieurs services ou tâches. Contrairement à la chorégraphie événementielle (où chaque service réagit indépendamment), l'orchestration fournit une logique, une visibilité et un contrôle centralisés sur l'ensemble du processus.

Ceci est généralement mis en œuvre à l'aide des outils suivants :

- AWS Step Functions— Définissez et exécutez des flux de travail dynamiques
- AWS Lambda— Effectuez des tâches discrètes dans le cadre du flux orchestré
- Amazon SQS ou Amazon EventBridge — Déclenche des étapes ou des réponses asynchrones

Le schéma suivant est un exemple d'orchestration de saga :



Un AWS Step Functions flux de travail gère le processus de commande d'un client :

1. Créer une commande (AWS Lambda)
2. Mettre à jour l'inventaire (AWS Lambda)
3. Effectuer le paiement (AWS Lambda)

L'orchestrateur coordonne chaque étape en gérant les nouvelles tentatives, les branches parallèles, les délais d'attente et les échecs.

Role-based système d'agents (orchestrateur)

Dans les systèmes agentiques, le modèle d'orchestrateur reflète l'orchestration des événements mais répartit la logique entre plusieurs agents de raisonnement, chacun ayant un rôle ou une spécialisation définis. Un agent orchestrateur central interprète l'ensemble de la tâche, la décompose en sous-tâches et les délègue aux agents de travail, chacune étant optimisée pour un domaine particulier (par exemple, recherche, codage, synthèse, révision).

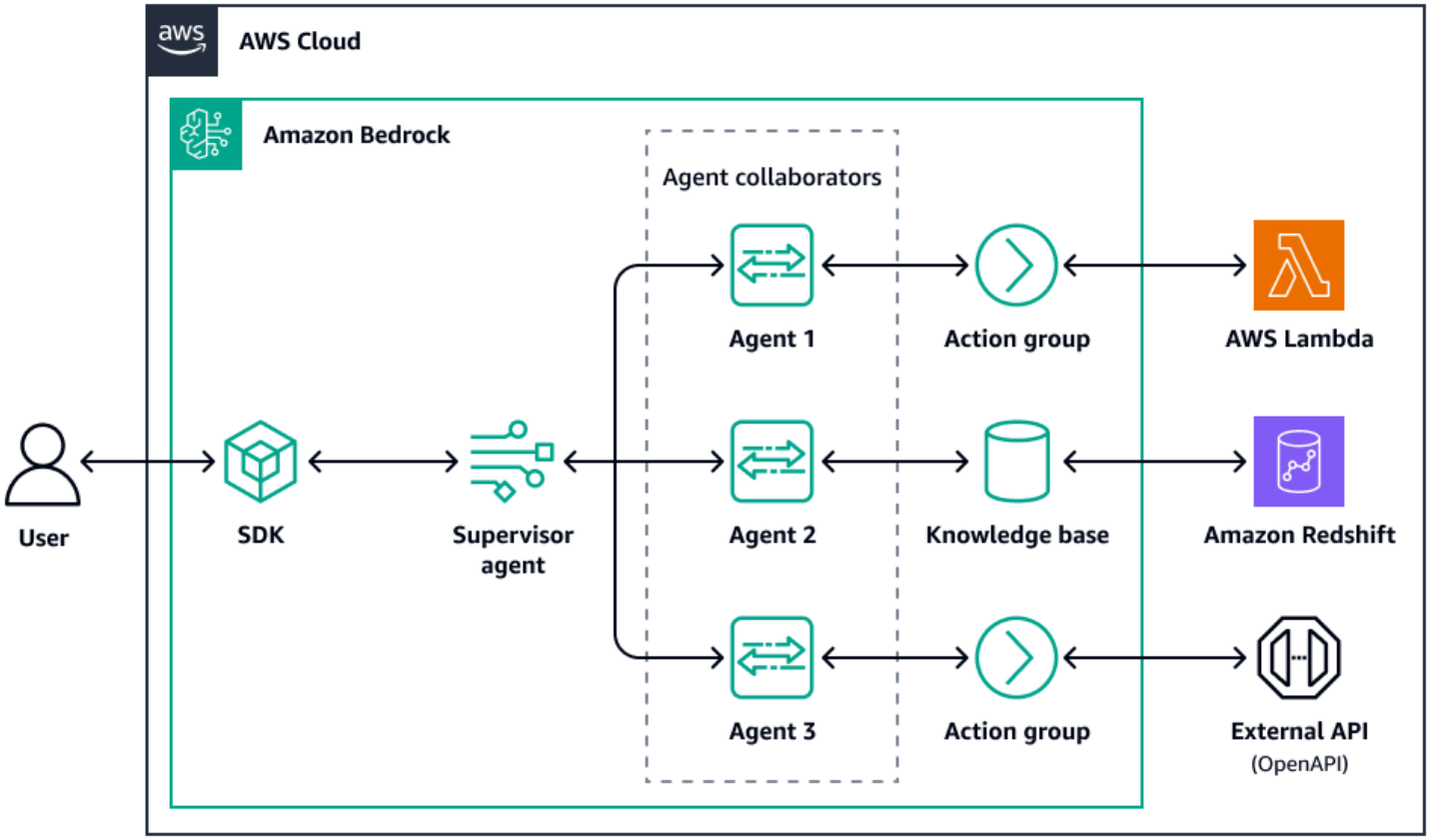
Superviseur

1. Un utilisateur soumet la requête « Créez un brief de projet et résumez les 5 principaux concurrents ».
2. L'agent Orchestrator effectue les opérations suivantes :

- Désigne un agent de recherche pour trouver les données des concurrents
- Envoie les résultats bruts à un agent de synthèse
- Transmet les résultats à un agent rédacteur
- Compile le résultat final pour l'utilisateur

Chaque agent fonctionne indépendamment, mais l'orchestrateur coordonne les tâches. Cela ressemble à une fonction Lambda qui gère les tâches du flux de travail.

Le schéma suivant est un exemple de superviseur :



1. Un utilisateur soumet une tâche à un agent de supervision Amazon Bedrock.
2. L'agent superviseur analyse la demande en sous-tâches pour chaque agent collaborateur.
3. Chaque sous-tâche est attribuée à un agent collaborateur avec des instructions ou des chaînes d'outils spécifiques au rôle.
4. Les agents de travail appellent des API ou des outils externes par le biais d'un groupe d'action.
5. Chaque agent de travail renvoie le résultat dans un format structuré.

6. Lorsque tous les travailleurs renvoient leurs résultats, le superviseur évalue, synthétise et renvoie la réponse finale.

Cette structure permet la modularité, l'adaptabilité et l'introspection dans les flux de travail d'agents complexes en plusieurs étapes.

Plats à emporter

Lorsque l'orchestration d'événements utilise un contrôle centralisé (par exemple AWS Step Functions) pour diriger l'exécution des services, les systèmes d'agents basés sur les rôles utilisent un agent LLM-powered orchestrateur pour définir l'objectif, déléguer des sous-tâches aux agents ouvriers et synthétiser le résultat final.

Dans les deux paradigmes, l'orchestrateur effectue les opérations suivantes :

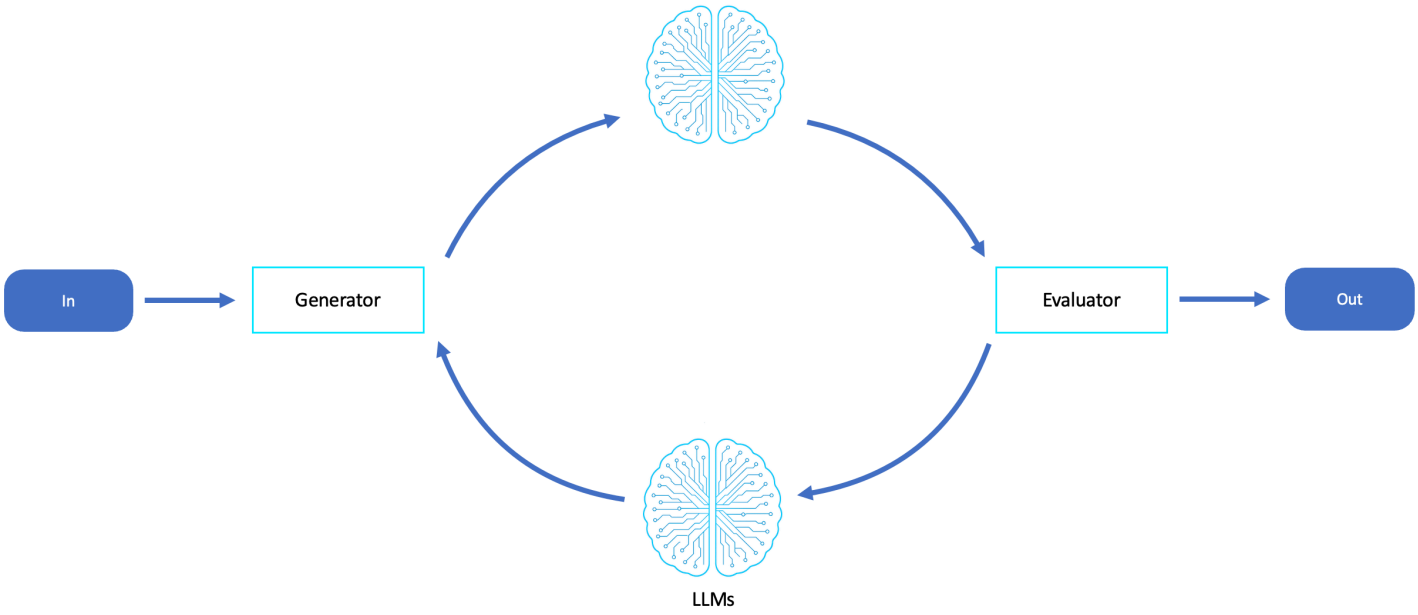
- Maintient le contexte et le flux d'exécution
- Gère le branchement, le séquençage et la gestion des erreurs
- Produit un résultat unifié à partir de composants distribués

L'orchestration agentique ajoute toutefois du raisonnement, de l'adaptabilité et de la délégation sémantique. Cela le rend parfaitement adapté aux tâches ouvertes, ambiguës et évolutives.

L'évaluateur reflète et affine les modèles de boucles

Les tâches telles que la génération de code, la synthèse ou la prise de décision autonome bénéficient grandement du feedback d'exécution, permettant au système d'évoluer grâce à l'observation et au raffinement. Pour rendre cela opérationnel, le cycle reflète-affiner peut être mis en œuvre sous la forme d'une boucle de contrôle de rétroaction pilotée par les événements, un modèle inspiré de l'ingénierie des systèmes, adapté aux flux de travail autonomes et intelligents.

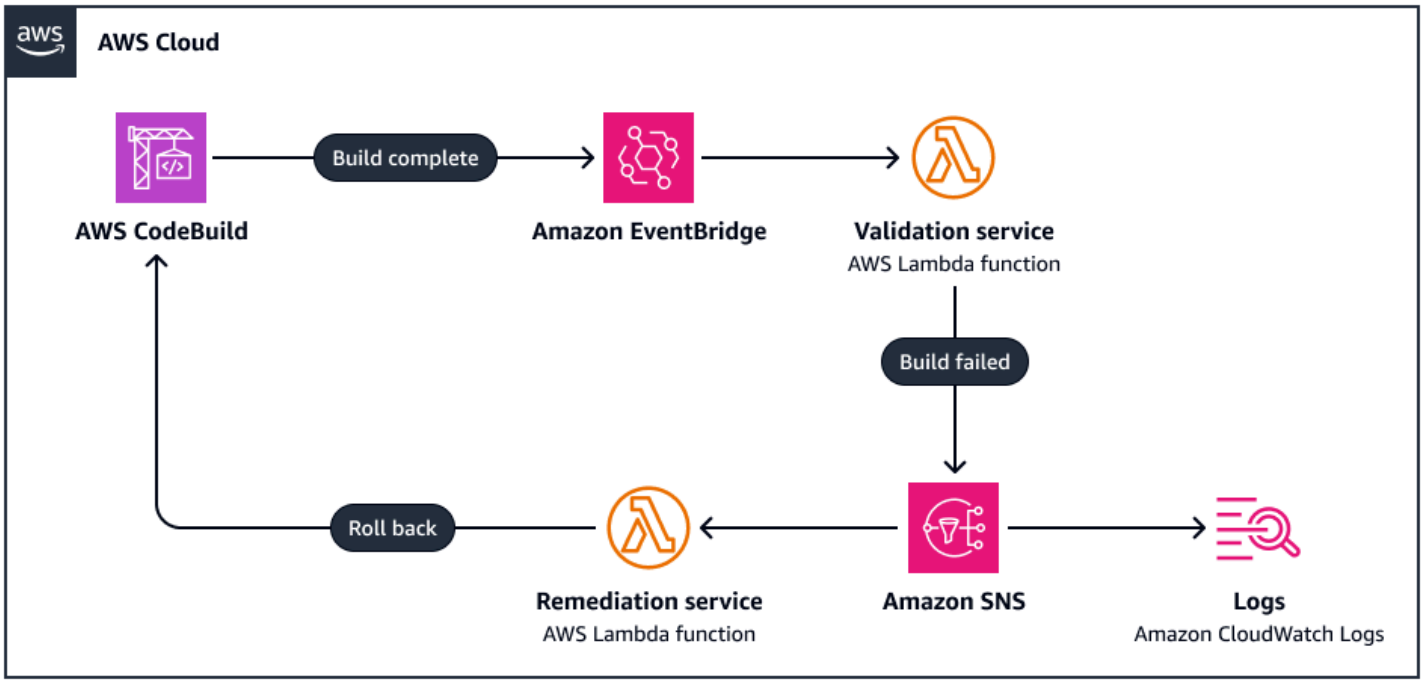
Le schéma suivant est un exemple de boucle de rétroaction basée sur l'affinement du reflet par un évaluateur :



Boucle de régulation du feedback

Une boucle de contrôle par rétroaction est un modèle qui surveille ses propres sorties et comportements, les évalue par rapport à des critères définis ou à un état souhaité, puis ajuste ses actions en conséquence. Cette architecture s'inspire de la théorie du contrôle et est fondamentale dans des domaines tels que l'automatisation, l'intégration continue et les pipelines de livraison continue (CI/CD), ainsi que les opérations d'apprentissage automatique.

Le schéma suivant est un exemple de boucle de régulation de rétroaction :



1. Un pipeline de déploiement émet un événement BuildComplete.
2. L'événement déclenche un test ou une tâche d'évaluation automatique qui valide le build.
3. En cas d'échec de la validation (par exemple, en raison de l'échec des tests, de problèmes de sécurité ou d'une violation des règles), le système :
   - Émet un événement BuildComplete
   - Enregistre le problème ou envoie une notification
   - Déclenche une correction ou une action corrective, telle que l'annulation, l'application de correctifs ou une nouvelle tentative

La boucle continue jusqu'à ce qu'elle produise un résultat acceptable ou une escalade, ou jusqu'à ce qu'un délai d'attente se produise. Ce modèle est couramment utilisé dans les cas suivants :

- EventBridge Règles Amazon pour acheminer les événements vers des tâches d'évaluation ou de correction
- AWS Step Functions pour la logique itérative des nouvelles tentatives et le branchement sur les résultats de l'évaluation
- Amazon Simple Notification Service (Amazon SNS) ou alarmes Amazon pour les CloudWatch déclencheurs de commentaires et les alertes
- AWS Lambda fonctions ou travailleurs conteneurisés pour appliquer des mesures correctives

Boucle de régulation du feedback (évaluateur)

Un flux de travail d'évaluateur est une boucle de rétroaction cognitive alimentée par des LLM ou des agents de raisonnement. Le processus comprend les étapes suivantes :

1. Un agent générateur ou LLM produit un résultat (par exemple, un plan, une réponse ou un brouillon).
2. Un agent évaluateur examine le résultat à l'aide d'une invite de critique ou d'une rubrique d'évaluation.
3. Sur la base des commentaires, l'agent d'origine ou un nouvel agent d'optimisation révise le résultat.

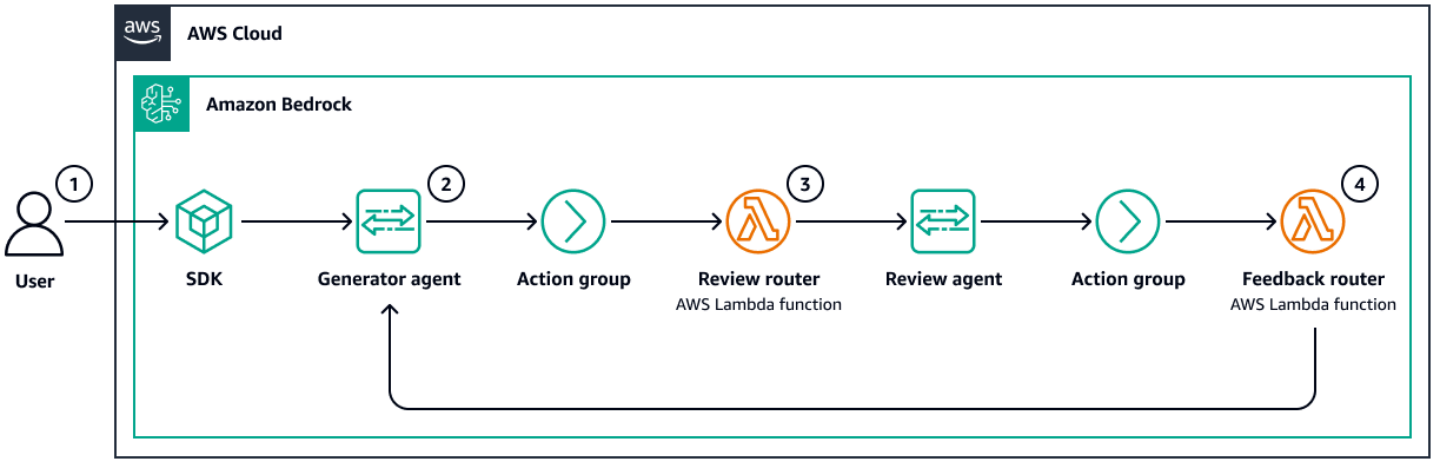
La boucle se répète jusqu'à ce que le résultat réponde à un ensemble de critères, soit approuvé ou atteigne une limite de tentatives.

Évaluateur

1. Un utilisateur demande à un agent de rédiger un résumé de la politique.
2. L'agent générateur le rédige.
3. Un agent évaluateur vérifie la couverture, le ton et l'exactitude juridique.
4. Si la réponse est inadéquate, elle est affinée et soumise à nouveau jusqu'à ce que la boucle de rétroaction converge.

Cela permet une auto-évaluation, un raffinement itératif et un contrôle adaptatif des sorties, le tout sans intervention humaine.

Le schéma suivant est un exemple de boucle de régulation de rétroaction (évaluateur) :



1. Un utilisateur émet une tâche (par exemple, rédiger une stratégie commerciale).
2. Un agent Amazon Bedrock génère un brouillon initial à l'aide d'un LLM.
3. Un deuxième agent (ou une invite de suivi) effectue une évaluation structurée (par exemple, « évaluez ce résultat en fonction de sa clarté, de son exhaustivité et de son ton »).
4. Si la note tombe en dessous d'un seuil, la réponse est révisée comme suit :
   - Réinvoquer le générateur avec une critique intégrée
   - Envoi du feedback à un agent de raffinage spécialisé
   - Itérer jusqu'à ce qu'une réponse acceptable soit atteinte

Les composants optionnels tels que AWS Lambda les contrôleurs AWS Step Functions peuvent gérer les seuils de feedback, les nouvelles tentatives et les stratégies de repli.

Plats à emporter

Alors que les boucles de contrôle par rétroaction traditionnelles utilisent des événements, des métriques et une logique de correction pour valider et ajuster le comportement du système, les boucles d'évaluation agentic utilisent des agents de raisonnement pour évaluer, refléter et réviser les résultats de manière dynamique.

Dans les deux paradigmes :

- La sortie est évaluée après sa génération
- Des actions correctives ou d'affinage sont déclenchées en fonction des commentaires
- Le système s'adapte en permanence à une cible, à une qualité ou à un objectif

La version agentic transforme la validation statique en réflexion sémantique, permettant aux agents de s'améliorer eux-mêmes d'évaluer leur propre efficacité.

Conception de flux de travail agentiques sur AWS

Chaque modèle de ce guide peut être créé à l'aide de Services AWS. Les agents Amazon Bedrock fournissent des canaux d'orchestration, d'accès aux données et d'interaction.

Composant	Service AWS	Objectif
Raisonnement du LLM	Amazon Bedrock	Logique des agents, planification, utilisation des outils
Exécution de l'outil	AWS Lambda, Amazon ECS, Amazon SageMaker	Héberger des outils externes pour les agents
Mémoire et RAG	Base de connaissances Amazon Bedrock, Amazon S3, OpenSearch	Mémoire persistante et sémantique
Orchestration	AWS Step Functions	Coordination des tâches et des agents en plusieurs étapes
Routage des événements	Amazon EventBridge, Amazon SQS	Messagerie interagent découplée
Interface utilisateur	Amazon API Gateway AWS AppSync, SDK	Points d'entrée pour les applications ou les systèmes
Contrôle	Amazon CloudWatch AWS X-Ray, AWS Distro pour OpenTelemetry	Observabilité et introspection des agents

Conclusion

Les modèles de flux de travail agentiques constituent la prochaine étape évolutive des architectures axées sur les événements, dans lesquelles la logique métier n'est pas définie de manière statique

mais raisonnée dynamiquement à l'aide d'un large modèle de langage (LLM), c'est-à-dire une cognition améliorée. En combinant des primitives natives du cloud traditionnelles avec des flux de travail LLM et des modèles de conception d'agents, les entreprises peuvent créer des systèmes adaptatifs, intelligents et modulaires qui répondent à des objectifs précis et tirent les leçons de l'expérience.

Dans ces modèles, Amazon Bedrock est la passerelle vers la cognition agentique, permettant aux LLM-based agents d'accéder aux flux de travail liés aux événements, d'interagir avec les outils et la mémoire, et de fournir des résultats structurés, traçables et alignés.

Lorsque vous concevez et déployez des systèmes agentiques, ces modèles de flux de travail fournissent des modèles pour la création d'architectures d'IA autonomes et composables. Ces systèmes sont basés sur les meilleures pratiques sans serveur et complétés par des modèles de base intelligents.

Historique du document

Le tableau suivant décrit les modifications importantes apportées à ce guide. Pour être averti des mises à jour à venir, abonnez-vous à un fil RSS.

Modification	Description	Date
Publication initiale	—	14 juillet 2025

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.