Alliance Conflent et Google Cloud – Architecture Agentique Unifiée

André-Guy Bruneau M.Sc. IT – Août 2025

Abstract

Cet essai propose une analyse architecturale de l'alliance stratégique entre Google Cloud et Confluent, visant à résoudre le problème critique du cloisonnement des agents d'intelligence artificielle en entreprise. En nous appuyant sur les annonces officielles, nous examinons comment l'intégration du protocole A2A (Agent-to-Agent) de Google avec les nouvelles capacités des agents de diffusion en continu (Streaming Agents) de Confluent Cloud établit une nouvelle norme pour les systèmes agentiques. L'analyse détaille la proposition de valeur de cette offre conjointe : le protocole A2A fournit un standard de communication gouverné, tandis que Confluent Cloud, propulsé par Apache Kafka, agit comme l'infrastructure de communication évolutive et en temps réel. Une étude de cas, fondée sur la démonstration de l'alignement automatisé des prix, illustre l'application pratique de fonctionnalités clés telles que l'intégration de serveurs MCP distants et l'enrichissement contextuel en temps réel. En conclusion, nous évaluons les implications stratégiques de cette offre pour les architectes de solutions, en positionnant cette alliance comme un catalyseur pour la transition d'expérimentations d'IA isolées vers des écosystèmes d'entreprise intelligents, connectés et réactifs.

1.0 Contexte et Enjeux Stratégiques

1.1 L'état actuel de l'IA agentique : Le défi des « îlots d'agents » en entreprise

L'adoption de l'intelligence artificielle (IA) agentique en entreprise progresse rapidement, mais cette croissance s'accompagne d'un défi structurel fondamental : la prolifération d'« îlots d'agents ». Ces agents, souvent développés au sein de systèmes SaaS spécifiques comme Oracle Cloud ERP, Salesforce ou ServiceNow, fonctionnent en vase clos, créant des silos technologiques et de données qui limitent leur potentiel.[1] Ce phénomène n'est pas simplement un problème technique, mais un obstacle stratégique majeur à la réalisation de la promesse de l'IA.

Les symptômes de ce cloisonnement sont multiples. Chaque agent, confiné à son domaine, ne dispose que d'une vue partielle des données de l'entreprise. Par exemple, un agent dans Salesforce a accès aux données client, mais ignore les perturbations de la chaîne d'approvisionnement gérées dans Oracle ERP.[1] Cette fragmentation des données conduit inévitablement à des aperçus incomplets et à des actions sous-optimales. L'impact sur la prise de décision est direct : sans vision holistique, les agents peuvent générer des recommandations contradictoires. Un agent ERP pourrait suggérer des réductions de coûts, tandis qu'un agent CRM préconise une augmentation des dépenses marketing, créant un désalignement opérationnel qui nuit aux objectifs globaux de l'organisation.[1]

Sur le plan commercial, ce cloisonnement constitue une responsabilité stratégique qui freine la croissance, érode la productivité et compromet l'expérience client.[2] Le problème transcende la simple fragmentation des données pour inclure l'isolement des connaissances, où les meilleures pratiques et les leçons apprises restent piégées au sein des équipes, et l'incapacité à automatiser les flux de travail interfonctionnels complexes.[3] Cette situation a engendré un paradoxe, identifié par McKinsey : alors que les copilotes « horizontaux » à l'échelle de l'entreprise se déploient rapidement avec des gains diffus, environ 90 % des cas d'usage « verticaux » à fort impact restent bloqués en phase pilote. Cette stagnation est due à des initiatives fragmentées, des limitations technologiques et des équipes d'IA elles-mêmes cloisonnées.[4] Le défi s'étend même à la sécurité et à la gestion des identités. Les agents opérant dans des environnements hybrides (cloud, sur site, en périphérie) manquent d'une couche d'orchestration d'identité unifiée, ce qui rend la traçabilité, l'application des politiques et la gouvernance de la sécurité presque impossibles à l'échelle.[5]

De manière ironique, les agents, conçus pour briser les silos, deviennent eux-mêmes victimes du problème qu'ils sont censés résoudre.[2] Cette situation n'est pas nouvelle ; elle est la réincarnation, à un niveau d'abstraction supérieur, du problème classique des silos informatiques qui a opposé les systèmes ERP et CRM pendant des décennies. L'industrie de l'IA, dans sa course à l'adoption, risque de répéter les erreurs du passé en construisant des solutions verticalement intégrées sans une fondation horizontale commune.

1.2 L'annonce stratégique : La réponse conjointe de Google Cloud et Confluent

Face à ce défi systémique, Google Cloud et Confluent ont formé une alliance stratégique pour fournir une réponse architecturale directe. Leur offre conjointe vise à créer le « tissu conjonctif » manquant qui peut unifier ces îlots d'agents en un écosystème cohérent et réactif.[3] Cette collaboration s'articule autour de deux piliers technologiques complémentaires. Le premier est le protocole Agent-to-Agent (A2A) de Google, un standard de communication ouvert conçu pour permettre aux agents de dialoguer et de collaborer, indépendamment de leur origine ou de leur technologie sous-jacente.[7] Le second pilier est Confluent Cloud, une plateforme de streaming de données basée sur Apache Kafka, qui sert d'infrastructure de communication en temps réel, évolutive et durable pour ces interactions.[8]

1.3 Thèse : L'offre intégrée comme fondation d'une nouvelle génération d'architectures agentiques

La synergie entre un protocole d'interaction standardisé (A2A) et une épine dorsale de communication découplée et événementielle (Confluent/Kafka) constitue un changement de paradigme architectural. Cette approche intégrée permet de passer de systèmes d'agents rigides, basés sur des intégrations point à point fragiles, à des écosystèmes d'agents collaboratifs, évolutifs et réactifs. En résolvant simultanément les problèmes d'interopérabilité sémantique via A2A et de communication en temps réel à grande échelle via Confluent, cette alliance fournit une fondation robuste. Elle offre aux entreprises une voie pour surmonter le stade du pilote et industrialiser leurs initiatives d'IA les plus prometteuses, brisant ainsi le cycle récurrent de fragmentation technologique avant que les silos agentiques ne se pétrifient.[4]

2.0 Pilier 1 : Le Protocole A2A de Google Cloud comme Standard d'Interaction

Le premier pilier de cette architecture unifiée est le protocole Agent-to-Agent (A2A) de Google. Il ne s'agit pas d'une simple API, mais d'une grammaire de gouvernance conçue pour orchestrer la collaboration entre des agents hétérogènes, établissant ainsi un langage commun indispensable à la création d'un véritable écosystème.

2.1 Analyse de l'offre : Positionnement du protocole A2A dans l'écosystème Vertex AI

L'objectif principal du protocole A2A est de fournir un standard de communication ouvert qui permet aux agents, quel que soit leur framework de développement (Agent Development Kit de Google, LangGraph, Crew.ai, etc.) ou leur fournisseur, de communiquer, d'échanger des informations en toute sécurité et de coordonner des actions.[7]
Il est essentiel de distinguer l'A2A du Model Context Protocol (MCP). Alors que le MCP se concentre sur la connexion des agents aux *outils* et aux sources de données, l'A2A est spécifiquement conçu pour la collaboration entre *agents*. La recommandation officielle est donc d'utiliser le MCP pour l'accès aux outils et l'A2A pour les interactions inter-agents.[11] Google a profondément intégré le support natif de l'A2A dans son écosystème Vertex AI, notamment dans l'Agent Development Kit (ADK), la plateforme de création d'agents no-code Agentspace, et l'environnement d'exécution géré Agent Engine. Cette intégration facilite le déploiement d'agents conformes à l'A2A sur des services comme Cloud Run ou Google Kubernetes Engine (GKE).[14]

2.2 Spécifications et fonctionnalités : Gouvernance, description des capacités et négociation

Le protocole A2A est bâti sur des standards web éprouvés, ce qui garantit une adoption et une intégration aisées dans les infrastructures informatiques existantes. Il utilise HTTP/S pour le transport, JSON-RPC 2.0 pour le formatage des messages, et Server-Sent Events (SSE) pour les communications en streaming.[7] La version 0.3 de la spécification a également introduit le support de gRPC pour les cas d'usage nécessitant une haute performance.[14]
Le mécanisme de découverte des capacités est l'une des caractéristiques les plus innovantes du protocole. Il repose sur l'« Agent Card », un fichier de métadonnées au format JSON, accessible via un endpoint standard (/.well-known/agent.json).[12] Cette « carte de visite numérique » permet à un agent de publier ses informations : nom, description, endpoint A2A, exigences d'authentification et, surtout, les compétences (skills) spécifiques

qu'il propose. Cela permet une découverte et une composition dynamiques des services agentiques.[13]
La communication elle-même est structurée autour de « Tâches » (Tasks). Un agent client formule une tâche et l'envoie à un agent distant via une requête JSON-RPC. Le résultat de cette tâche est un « Artefact » (Artifact), qui peut être composé de plusieurs « Parties » (Parts) de différents types (texte, fichier, données structurées), offrant une grande flexibilité dans les échanges.[13] Le protocole est également conçu pour gérer des tâches asynchrones de longue durée, pouvant s'étendre sur des heures ou des jours, avec un système de notifications push via des webhooks pour suivre leur progression en temps réel.[7]

2.3 Valeur ajoutée pour l'architecte : Assurer l'interopérabilité et la sécurité des communications hétérogènes

Pour un architecte de solutions, la valeur de l'A2A réside dans sa capacité à remplacer les intégrations point à point, fragiles et coûteuses à maintenir, par un langage commun et standardisé. Cela permet de concevoir des systèmes modulaires où les agents peuvent être ajoutés, retirés ou remplacés sans nécessiter une refonte de la logique d'intégration.[18]
La sécurité et la gouvernance sont intégrées au cœur du protocole. Il prend en charge des mécanismes d'authentification robustes comme OAuth 2.0, le contrôle d'accès basé sur les rôles (RBAC), et le chiffrement de bout en bout des communications.[18] Les mises à jour récentes de la spécification ont encore renforcé la gestion des privilèges délégués et des agents authentifiés et non authentifiés, répondant ainsi aux exigences de sécurité des entreprises.[14]
Enfin, le protocole favorise un découplage fort et une abstraction. Les agents collaborent via l'interface A2A standardisée sans avoir besoin d'exposer leur mémoire interne ou les outils spécifiques qu'ils utilisent. Ce principe d'encapsulation est fondamental pour construire des systèmes complexes, maintenables et sécurisés.[17]

Caractéristique	Intégration API REST ad hoc	Bus de messages traditionnel (p. ex., RabbitMQ)	Protocole A2A
Découverte des capacités	Manuelle (documentation) ou via des registres externes	Aucune (dépend de la convention de nommage des files/topics)	Native et standardisée (Agent Card)
Négociation des tâches	Logique personnalisée dans chaque service	Non applicable	Intégrée au protocole
Gestion des tâches asynchrones	Requiert des mécanismes personnalisés (webhooks, polling)	Modèle de base (mise en file d'attente)	Support natif pour les tâches de longue durée avec notifications push
Standardisation des charges utiles	Spécifique à chaque API (p. ex., OpenAPI)	Format de message générique	Format standardisé (Task, Artifact, Part)
Sécurité et identité de l'agent	Gérée au niveau du transport (p. ex., OAuth 2.0)	Gérée au niveau de la connexion au bus	Intégrée au protocole (authentification, RBAC)
Support multi-modalité	Géré via des types MIME personnalisés	Géré via la sérialisation du corps du message	Natif via la structure Part

Complexité de l'intégration	Élevée (N2 intégrations pour N agents)	Modérée (dépend de la complexité du routage)	Faible (chaque agent implémente un seul standard)

3.0 Pilier 2 : Confluent Cloud comme Infrastructure de Communication en Temps Réel

Si le protocole A2A fournit la grammaire sémantique pour la collaboration entre agents, Confluent Cloud, propulsé par Apache Kafka, constitue l'infrastructure physique et logique qui donne vie à cette communication à l'échelle de l'entreprise. Il agit comme le système circulatoire qui transporte les messages A2A de manière fiable, durable et en temps réel.

3.1 Le rôle d'Apache Kafka : De la messagerie à l'épine dorsale des interactions agentiques

Apache Kafka a évolué bien au-delà d'un simple bus de messages pour devenir une plateforme de streaming d'événements, agissant comme le « système nerveux central » des entreprises modernes.[21] Dans un contexte agentique, Kafka devient plus qu'un simple tuyau ; il se transforme en une mémoire partagée, durable et rejouable pour toutes les interactions entre agents. Chaque décision, chaque requête et chaque réponse peut être enregistrée comme un événement immuable dans un log Kafka.[8]
Cette architecture événementielle offre un découplage temporel et spatial fondamental. Un agent peut publier un événement sans savoir quels autres agents le consommeront, ni même s'ils sont en ligne à ce moment-là. Cette approche asynchrone est intrinsèquement plus résiliente et évolutive que les appels API synchrones point à point, qui créent des dépendances rigides entre les services.[22] De plus, la nature de log de Kafka, combinée à des outils de gouvernance comme Confluent Schema Registry, assure une traçabilité complète des flux de communication. Cela permet de savoir quel agent a dit quoi et quand, une capacité essentielle pour l'audit, le débogage et la sécurité.[8]

3.2 L'offre « Streaming Agents » sur Confluent Cloud

Confluent a récemment dévoilé son offre « Streaming Agents », une innovation qui fusionne le monde du traitement de flux et celui de l'IA agentique. Ces agents sont décrits comme des « microservices événementiels dotés d'un cerveau », construits nativement sur Apache Flink et Apache Kafka.[8] Leur principale caractéristique est d'être « toujours actifs » (always on), capables de percevoir, de raisonner et d'agir en continu sur les flux de données en temps réel, sans attendre un déclencheur externe.[8] La proposition de valeur fondamentale est d'intégrer le raisonnement de l'IA directement dans les pipelines de traitement de flux à l'aide d'API Flink familières. Cela transforme l'agent d'IA d'une application externe qui consomme des données en une entité de calcul de première classe intégrée au flux de données lui-même. Cette fusion réduit la latence, simplifie radicalement la pile technologique et démocratise le développement d'agents, le rendant accessible aux ingénieurs de données familiers avec Flink SQL.[8]

3.2.1 Intégration de serveurs MCP distants (p. ex., Zapier) pour l'extension des capacités

Pour agir sur le monde, les agents ont besoin d'outils. Les Streaming Agents intègrent nativement le support du Model Context Protocol (MCP) pour l'invocation d'outils externes.[8] Cela permet à un agent, en fonction du contexte d'un événement en temps réel, de sélectionner et d'appeler dynamiquement le bon outil, qu'il s'agisse d'une base de données, d'une API SaaS ou d'un autre service. Un exemple puissant de cette capacité est l'intégration avec Zapier, qui expose son vaste écosystème de plus de 7 000 applications et 30 000 actions via un serveur MCP.[27] Un Streaming Agent sur Confluent peut ainsi invoquer une action Zapier (comme « envoyer un e-mail via Gmail » ou « créer une tâche dans Jira ») comme s'il s'agissait d'un outil natif, étendant de manière exponentielle sa capacité à interagir avec des systèmes tiers sans nécessiter de code d'intégration personnalisé.[28]

3.2.2 Support natif pour la recherche vectorielle (p. ex., Pinecone, MongoDB)

Les agents d'IA, en particulier ceux basés sur les grands modèles de langage (LLM), ont besoin d'un contexte à jour pour fournir des réponses précises et éviter les hallucinations. Confluent Cloud répond à ce besoin en s'intégrant étroitement avec des bases de données vectorielles de premier plan comme Pinecone et MongoDB Atlas Vector Search.[30] L'architecture permet de construire des pipelines de Retrieval-Augmented Generation (RAG) en streaming. Les Streaming Agents peuvent utiliser Flink SQL pour intercepter des données non structurées à la volée (p. ex., des documents, des logs clients), les transformer en embeddings vectoriels en temps réel, et les synchroniser en continu avec la base de données vectorielle. Cela garantit que la base de connaissances sur laquelle l'agent s'appuie est toujours fraîche et reflète l'état le plus récent de l'entreprise.[8]

3.2.3 Enrichissement contextuel en temps réel avec les bases de données externes (p. ex., MySQL, Postgres)

Une décision intelligente nécessite un contexte complet. La fonctionnalité « External Tables » des Streaming Agents permet d'utiliser Flink SQL pour joindre des flux d'événements en temps réel avec des données provenant de sources externes non-Kafka, telles que des bases de données relationnelles (MySQL, Postgres) ou des API REST.[8] Cette capacité d'enrichissement à la volée est cruciale. Par exemple, un événement de transaction contenant un ID client et un montant peut être instantanément enrichi avec le profil complet du client (historique d'achat, statut de fidélité) depuis une base de données CRM. L'agent qui reçoit cet événement enrichi dispose alors de tout le contexte nécessaire pour prendre une décision plus pertinente et personnalisée.[26]

3.3 Avantages architecturaux : Évolutivité, durabilité et observabilité des flux de communication

L'utilisation de Confluent Cloud comme infrastructure de communication offre des avantages architecturaux décisifs. L'architecture découplée de Kafka permet une mise à l'échelle indépendante des producteurs et des consommateurs d'événements (les agents), ce qui permet de gérer des volumes massifs d'interactions sans goulots d'étranglement.[23] La durabilité des messages, persistés dans le log Kafka, est une caractéristique fondamentale. Elle garantit non seulement qu'aucune interaction n'est perdue, mais elle permet également de « rejouer » l'historique des événements. Cette capacité de rejouabilité est inestimable pour le débogage, les tests (A/B testing, lancements "dark"), la récupération après sinistre et l'audit rigoureux des décisions prises par les agents.[8] Enfin, la plateforme offre une observabilité complète sur les flux de données, permettant de suivre la lignée des informations et de surveiller la santé des communications entre les différents agents de l'écosystème.[8]

4.0 Architecture de Référence et Cas d'Usage Appliqué

Pour concrétiser les concepts des deux piliers, cette section propose une architecture de référence unifiée et l'illustre à travers un cas d'usage métier tangible : l'automatisation de l'alignement des prix dans le secteur du e-commerce.

4.1 Conception d'une architecture de solution intégrant les offres Google Cloud et Confluent

L'architecture de référence positionne Confluent Cloud comme le « système nerveux central » événementiel de l'entreprise. Les différents agents, qu'ils soient déployés sur Vertex AI Agent Engine, Cloud Run ou d'autres plateformes, agissent comme des producteurs et des consommateurs de topics Kafka.[22] La communication entre les agents n'est pas directe (point à point), mais médiatisée par Kafka. Un agent A publie une requête, potentiellement formatée selon le protocole A2A, dans un topic Kafka. L'agent B, abonné à ce topic, consomme le message, le traite, et publie une réponse dans un autre topic. Cette chorégraphie événementielle assure un découplage maximal.[22]

Dans cette architecture, les Streaming Agents de Confluent jouent un rôle spécialisé. Intégrés directement au flux de données, ils peuvent enrichir, filtrer ou transformer les messages à la volée, ou encore orchestrer des workflows complexes en réagissant à des séquences d'événements.[8] Les agents construits sur l'écosystème Vertex AI, comme ceux utilisant l'ADK, interagissent avec cette épine dorsale via les connecteurs Confluent. Ainsi, Vertex AI Agent Engine fournit l'environnement d'exécution pour la logique de l'agent (modèle, outils,

mémoire), tandis que Confluent gère la communication inter-agents de manière asynchrone et évolutive.[21]

4.2 Étude de cas détaillée : L'automatisation de l'alignement des prix

L'objectif métier est de permettre à une entreprise de e-commerce de maintenir des prix compétitifs en ajustant automatiquement ses propres prix en fonction des changements tarifaires de ses concurrents, et d'en informer les clients pertinents.

4.2.1 Séquençage des interactions : De l'extraction de données (scraping) à la notification client

Le processus se déroule en plusieurs étapes orchestrées par des événements :

1. **Perception :** Un « Agent Scraper », déployé sur Cloud Run, surveille les sites des concurrents. Lorsqu'un changement de prix est détecté pour un produit suivi, il publie un événement PriceChangeDetected dans un topic Kafka.
2. **Traitement et Enrichissement :** Un « Streaming Agent » Confluent consomme cet événement. Il utilise une table externe Flink pour joindre l'événement en temps réel avec des données internes sur le produit (marge bénéficiaire, niveau de stock, etc.) provenant d'une base de données MySQL.[26]
3. **Raisonnement :** L'événement, désormais enrichi, est publié par le Streaming Agent dans un topic PriceAnalysisRequired. Un « Agent d'Analyse de Prix », hébergé sur Vertex AI Agent Engine, consomme cet événement. Il utilise un LLM (Gemini) pour analyser le contexte complet et décider de la stratégie de prix à adopter (s'aligner, s'aligner avec une marge de X%, ne rien faire) en fonction des règles métier.[34]
4. **Action :** L'Agent d'Analyse de Prix publie sa décision sous la forme d'un événement PriceUpdateAction (contenant le nouveau prix) dans un topic Kafka. Cet événement est ensuite consommé en parallèle par deux agents distincts.
   - Un « Agent de Mise à Jour de Produit » qui appelle l'API interne du catalogue de produits pour effectuer le changement de prix.
   - Un « Agent de Notification » chargé d'informer les clients.
5. **Notification Externe :** L'Agent de Notification, via une invocation d'outil MCP, appelle un serveur MCP exposé par Zapier. Cela déclenche un « Zap » préconfiguré qui envoie un e-mail via Gmail aux clients qui avaient placé ce produit dans leur liste de favoris.[8]

4.2.2 Rôle de chaque composant : Agent Confluent, LLM, Webhook Zapier, et Gmail

Composant	Technologie sous-jacente	Rôle principal	Données en entrée	Données en sortie/Action
Agent Scraper	Python (sur Cloud Run)	Perception	URL du concurrent	Événement PriceChangeDetected
Streaming Agent Confluent	Confluent Cloud (Flink SQL)	Enrichissement & Routage	Événement PriceChangeDetected, Données produit (MySQL)	Événement PriceAnalysisRequired
Agent d'Analyse de Prix	Vertex AI Agent Engine (Gemini)	Raisonnement & Décision	Événement PriceAnalysisRequired	Événement PriceUpdateAction
Agent de Mise à Jour de Produit	Microservice (sur Cloud Run)	Action interne	Événement PriceUpdateAction	Appel API au catalogue produit
Agent de	Vertex AI	Orchestration	Événement	Appel d'outil MCP

Notification	Agent Engine	d'action externe	PriceUpdateAction	
Serveur MCP Zapier	Zapier MCP	Abstraction d'API externe	Requête MCP	Déclenchement d'un Zap (envoi d'e-mail)

Le **Streaming Agent de Confluent** agit comme le chef d'orchestre du workflow, assurant la transformation et l'enrichissement des données en temps réel pour fournir un contexte complet à l'agent de raisonnement.[8] Le **LLM (Gemini)** est le cerveau décisionnel, appliquant une logique métier complexe sur des données déjà contextualisées.[34] Le **serveur MCP Zapier** sert de passerelle vers le monde extérieur, abstrayant la complexité de l'API de Gmail et permettant à l'agent de simplement demander l'envoi d'un e-mail.[28]

4.2.3 Analyse des flux de données et des appels de service via Kafka et le protocole A2A

Dans ce scénario, les données circulent sous forme d'événements immuables à travers une série de topics Kafka (price-changes, enriched-price-changes, price-update-actions). Cette architecture de chorégraphie événementielle est extrêmement flexible ; on pourrait, par exemple, ajouter un nouvel « Agent d'Audit » qui consomme les événements PriceUpdateAction pour la conformité, sans modifier aucun des agents existants.[22] Bien que ce cas d'usage soit principalement une chorégraphie, le protocole A2A serait utilisé pour des interactions plus directes, de type requête-réponse, au sein de ce cadre événementiel. Si l'Agent d'Analyse de Prix avait besoin de l'approbation d'un « Agent de Marge » spécialisé avant de valider un changement de prix important, il publierait une requête formatée selon le protocole A2A dans un topic margin-approval-requests. L'Agent de Marge consommerait cette requête, effectuerait sa validation, et publierait une réponse, également au format A2A, dans un topic margin-approval-responses. Le protocole A2A garantirait que le format et la sémantique de cette interaction sont standardisés, clairs et gouvernés.

5.0 Implications pour l'Architecture d'Entreprise

L'adoption d'une architecture agentique unifiée basée sur l'alliance Google-Confluent a des implications profondes pour l'architecture d'entreprise. Elle nécessite une réflexion sur la maturité des systèmes, la gouvernance et le positionnement par rapport aux approches alternatives.

5.1 Modèle de maturité : De l'automatisation de tâches à l'orchestration de processus complexes

Pour aider les organisations à évaluer leur parcours, un modèle de maturité pour les systèmes multi-agents peut être défini. Ce modèle, inspiré par la recherche académique et les analyses de l'industrie, décrit une progression depuis des automatisations simples jusqu'à des écosystèmes autonomes.[37]

Niveau de Maturité	Description	Caractéristiques architecturales	Rôle humain
1. Automatisé	Agents isolés exécutant des tâches répétitives.	Communication ad hoc via des API point à point ; "îlots d'agents".	Déclencheur et superviseur de tâches.
2. Coordonné	Des groupes d'agents collaborent sur des workflows simples.	Bus de messages simple ou orchestration centralisée rigide.	Concepteur et mainteneur des workflows.
3. Collaboratif	Agents hétérogènes communiquant via un protocole standard (A2A) sur une infrastructure partagée (Kafka).	Architecture événementielle, protocole de communication standardisé.	Superviseur de l'écosystème, gestionnaire d'exceptions.

4. Adaptatif	L'écosystème peut se reconfigurer dynamiquement ; les agents découvrent et négocient des tâches de manière autonome.	Découverte dynamique de services (Agent Cards), mécanismes de négociation.	Définisseur d'objectifs et de contraintes stratégiques.
5. Autonome	Systèmes multi-agents auto-organisés optimisant des processus métier complexes avec une supervision humaine stratégique.	Gouvernance décentralisée, apprentissage collectif.	Garant du contrôle humain significatif (Meaningful Human Control).

L'architecture proposée par Google et Confluent est un catalyseur majeur, permettant aux entreprises de passer du Niveau 2 (Coordonné) au Niveau 3 (Collaboratif), qui est une étape fondamentale et une condition préalable pour atteindre les niveaux supérieurs d'adaptation et d'autonomie.

5.2 Considérations pour l'intégration : Sécurité, gestion du cycle de vie des agents et gouvernance des données

Le déploiement d'un tel écosystème nécessite une gouvernance rigoureuse. Les meilleures pratiques en matière de gouvernance de l'IA doivent être appliquées, notamment l'établissement d'une propriété claire des agents et des données (modèle RACI), la classification des données sensibles circulant dans les topics Kafka, et la mise en place d'une observabilité continue des interactions entre agents.[41]
Un aspect critique est la gestion des identités non humaines (Non-Human Identities - NHI). Chaque agent doit être traité comme une identité de première classe, avec ses propres informations d'identification, des permissions strictement délimitées (principe du moindre privilège) et une piste d'audit complète pour chaque action entreprise.[40] Le protocole A2A, avec ses mécanismes d'authentification intégrés, et Kafka, avec ses listes de contrôle d'accès (ACLs), fournissent les briques technologiques essentielles à cette gouvernance.
Enfin, l'architecture doit prendre en charge le cycle de vie complet des agents : développement, déploiement (sur Agent Engine ou Cloud Run), évaluation, surveillance (via les logs et traces de Vertex AI) et retrait.[15] La capacité de rejouabilité offerte par Kafka est ici un atout majeur, permettant de réaliser des tests de régression et des évaluations de performance en utilisant des données de production réelles sans impacter le système live.[8]

5.3 Analyse comparative : Positionnement de l'offre Google/Confluent face aux approches alternatives

Le choix d'une infrastructure de communication pour les agents est une décision architecturale fondamentale. L'approche Google/Confluent doit être évaluée par rapport à plusieurs alternatives.

Critère	API REST point à point	RabbitMQ (Bus de messages)	Apache Kafka (Streaming d'événements)	Apache Pulsar (Streaming d'événements)
Modèle de communication	Synchrone, requête-réponse	Asynchrone, mise en file d'attente (AMQP)	Asynchrone, log d'événements	Asynchrone, unifié (file d'attente + log)
Découplage	Faible (couplage fort)	Élevé	Très élevé (découplage temporel)	Très élevé (découplage temporel)
Évolutivité	Limitée par les endpoints	Élevée, mais peut devenir un goulot	Extrêmement élevée (scaling	Extrêmement élevée (scaling

		d'étranglement	horizontal)	indépendant)
Persistance/Rejouabilité	Aucune	Limitée (messages en file)	Native et à long terme	Native et à long terme (tiered storage)
Traitement de flux intégré	Non	Non	Oui (Kafka Streams, ksqlDB, Flink)	Oui (Pulsar Functions)
Multi-location	Gérée par l'application	Limitée (vhosts)	Limitée (nécessite des outils)	Native et robuste
Complexité opérationnelle	Élevée à l'échelle (N2)	Modérée	Modérée à élevée	Élevée (plus de composants)

L'approche basée sur Kafka se distingue par son écosystème de traitement de flux mature et profondément intégré (notamment avec Flink), qui est au cœur de l'offre des Streaming Agents de Confluent. Son concurrent le plus direct, Apache Pulsar, offre des avantages architecturaux notables comme la séparation du calcul et du stockage, une meilleure multi-location native et une plus grande flexibilité dans les modèles de messagerie.[44] Cependant, Kafka bénéficie d'une adoption plus large et d'un écosystème plus vaste, ce qui peut être un facteur décisif pour de nombreuses entreprises. Le choix entre Kafka et Pulsar dépendra des priorités spécifiques de l'organisation : la maturité et la richesse de l'écosystème pour Kafka, contre la flexibilité architecturale et la multi-location pour Pulsar.

6.0 Conclusion et Recommandations Stratégiques

6.1 Synthèse : La valeur synergique de l'alliance Google Cloud et Confluent

L'analyse de l'alliance stratégique entre Google Cloud et Confluent révèle que sa véritable innovation ne réside pas dans ses composants pris isolément, mais dans leur puissante synergie. La combinaison d'un *standard de gouvernance sémantique* (le protocole A2A) et d'une *infrastructure de transport découplée et évolutive* (Confluent Cloud sur Kafka) crée une solution complète qui répond de front au problème des « îlots d'agents ». L'un sans l'autre serait incomplet. Le protocole A2A, sans une épine dorsale industrielle comme Kafka, manquerait de la robustesse, de la durabilité et de l'évolutivité nécessaires pour les déploiements d'entreprise. Inversement, Kafka, sans un langage standardisé comme A2A, risquerait de n'être qu'un simple tuyau transportant des messages propriétaires et non interopérables, perpétuant le cloisonnement à un autre niveau. Ensemble, ils fournissent une plateforme industrialisée qui offre une voie de sortie au « paradoxe de l'IA générative ».[4] Ils permettent aux cas d'usage verticaux, riches en valeur, de s'intégrer, de collaborer et de s'étendre à l'échelle de l'entreprise, transformant des expériences isolées en un écosystème intelligent et connecté.

6.2 Recommandations pour les architectes de solutions : Quand et comment adopter cette architecture

Cette architecture est particulièrement indiquée pour les entreprises qui se trouvent à un point d'inflexion stratégique. Son adoption devrait être sérieusement considérée lorsque l'organisation :

1. Prévoit de déployer plus qu'une poignée d'agents et anticipe des besoins de collaboration entre différents domaines métier (p. ex., ventes, logistique, finance).
2. Opère dans des environnements hautement événementiels où la réactivité en temps réel constitue un avantage concurrentiel direct.
3. Possède un paysage applicatif hétérogène (multi-cloud, sur site, SaaS tiers) qui exige une interopérabilité robuste.

4. Est soumise à des exigences réglementaires ou de conformité strictes nécessitant un audit, une gouvernance et une traçabilité sans faille des décisions automatisées.

Pour une adoption réussie, une approche par étapes est recommandée :

1. **Phase 1 (Fondation) :** Établir Confluent Cloud comme le système nerveux central pour les flux d'événements métier les plus critiques, créant ainsi une source de vérité en temps réel.
2. **Phase 2 (Premiers agents) :** Développer un premier cas d'usage à fort impact avec un ou deux agents qui communiquent via des événements sur Kafka, en utilisant des schémas de données bien définis et gouvernés.
3. **Phase 3 (Standardisation) :** Introduire le protocole A2A pour formaliser les messages échangés entre les agents, en commençant par les nouvelles implémentations pour garantir l'interopérabilité future.
4. **Phase 4 (Expansion) :** Tirer parti de l'écosystème A2A pour intégrer des agents de tiers et étendre progressivement l'écosystème d'agents collaboratifs à l'ensemble de l'entreprise.

6.3 Perspectives : L'avenir des écosystèmes d'agents autonomes et interconnectés en entreprise

À plus long terme, des protocoles comme A2A et des infrastructures comme Confluent jettent les bases de concepts encore plus avancés, tels que des « marchés » internes d'agents. Dans une telle vision, les agents pourraient dynamiquement offrir et consommer des services cognitifs, négociant des tâches et des ressources en temps réel pour optimiser les processus métier.

La perspective finale est celle d'une transformation de l'entreprise, qui passerait d'une collection rigide d'applications et de processus à un système adaptatif complexe. Cet organisme numérique, composé d'agents intelligents et collaboratifs, serait capable de percevoir, de raisonner et d'agir collectivement pour s'adapter dynamiquement et en temps réel aux conditions du marché. L'alliance entre Google Cloud et Confluent ne représente pas l'aboutissement de cette vision, mais elle constitue une étape fondamentale, pragmatique et architecturale solide sur la voie de sa réalisation.

Ouvrages cités

1. Challenges of Siloed AI Agents in Enterprise SaaS - Jade Global, dernier accès : août 20, 2025, https://www.jadeglobal.com/blog/challenges-siloed-ai-agents-built-enterprise-saas-providers
2. Even AI agents aren't immune to silos - TechRadar, dernier accès : août 20, 2025, https://www.techradar.com/pro/even-ai-agents-arent-immune-to-silos
3. Breaking the Walls: How Agentic AI Is Dismantling Silos in Global Enterprises – Part I, dernier accès : août 20, 2025, https://customerthink.com/breaking-the-walls-how-agentic-ai-is-dismantling-silos-in-global-enterprises-part-i/
4. Seizing the agentic AI advantage - McKinsey, dernier accès : août 20, 2025, https://www.mckinsey.com/capabilities/quantumblack/our-insights/seizing-the-agentic-ai-advantage
5. The 6 identity problems blocking AI agent adoption in hybrid environments - Strata.io, dernier accès : août 20, 2025, https://www.strata.io/blog/agentic-identity/6-identity-problems-ai-agent-adoption-3a/
6. Confluent Cloud Joins AWS Marketplace's New AI Agents and Tools Category, dernier accès : août 20, 2025, https://enterpriseitworldmea.com/confluent-cloud-joins-aws-marketplaces-new-ai-agents-and-tools-category/
7. Announcing the Agent2Agent Protocol (A2A) - Google for Developers Blog, dernier accès : août 20, 2025, https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/
8. Introducing Streaming Agents on Confluent Cloud, dernier accès : août 20, 2025, https://www.confluent.io/blog/introducing-streaming-agents/
9. Real-Time AI Agents Powered by Apache Kafka, Apache Flink, and Google Cloud, dernier accès : août 20, 2025, https://systemsdigest.com/videos/real-time-ai-agents-powered-apache-kafka-apache-flink-and-google-cloud
10. Vertex AI Agent Builder | Google Cloud, dernier accès : août 20, 2025, https://cloud.google.com/products/agent-builder
11. dernier accès : décembre 31, 1969, https://www.jadeglobal.com/blog/challenges-siloed-ai-agents-built-enterprise-saas-providers/
12. Getting Started with Agent-to-Agent (A2A) Protocol: A Purchasing Concierge and Remote Seller Agent

Interactions with Gemini on Cloud Run and Agent Engine | Google Codelabs, dernier accès : août 20, 2025, https://codelabs.developers.google.com/intro-a2a-purchasing-concierge
13. How Google A2A Protocol Works: Key Insights - Trickle AI, dernier accès : août 20, 2025, https://content.trickle.so/blog/how-google-a2a-protocol-actually-works
14. Google Brings the A2A Protocol to More of Its Cloud - The New Stack, dernier accès : août 20, 2025, https://thenewstack.io/google-brings-the-a2a-protocol-to-more-of-its-cloud/
15. Agent2Agent protocol (A2A) is getting an upgrade | Google Cloud Blog, dernier accès : août 20, 2025, https://cloud.google.com/blog/products/ai-machine-learning/agent2agent-protocol-is-getting-an-upgrade
16. Using Vertex AI to evaluate an example A2A Agent - Google Developer forums, dernier accès : août 20, 2025, https://discuss.google.dev/t/using-vertex-ai-to-evaluate-an-example-a2a-agent/194032
17. Exploring Agent2Agent (A2A) Protocol with Purchasing Concierge ..., dernier accès : août 20, 2025, https://medium.com/google-cloud/exploring-agent2agent-a2a-protocol-with-purchasing-concierge-use-case-on-cloud-run-36f4b896eadf
18. Google A2A Protocol Technical Documentation Guide - BytePlus, dernier accès : août 20, 2025, https://www.byteplus.com/en/topic/551076
19. Google A2A Protocol Developer Resources | Complete Guide - BytePlus, dernier accès : août 20, 2025, https://www.byteplus.com/en/topic/551317
20. Google A2A Protocol Hybrid Cloud Implementation Guide - BytePlus, dernier accès : août 20, 2025, https://www.byteplus.com/en/topic/551359
21. Confluent brings real-time capabilities to Google Cloud gen AI, dernier accès : août 20, 2025, https://cloud.google.com/blog/topics/partners/confluent-brings-real-time-capabilities-to-google-cloud-gen-ai
22. Agentic AI with the Agent2Agent Protocol (A2A) and MCP using Apache Kafka as Event Broker - Kai Waehner, dernier accès : août 20, 2025, https://www.kai-waehner.de/blog/2025/05/26/agentic-ai-with-the-agent2agent-protocol-a2a-and-mcp-using-apache-kafka-as-event-broker/
23. Google Cloud Technology Partner of the Year | Confluent, dernier accès : août 20, 2025, https://www.confluent.io/partner/google-cloud/
24. Confluent's Real-Time Agents Build on Kafka Streaming Data - The New Stack, dernier accès : août 20, 2025, https://thenewstack.io/confluents-real-time-agents-build-on-kafka-streaming-data/
25. Confluent Unlocks Scalable Real-Time Agentic AI With Streaming Agents - AiThority, dernier accès : août 20, 2025, https://aithority.com/machine-learning/confluent-unlocks-scalable-real-time-agentic-ai-with-streaming-agents/
26. Confluent Unlocks Scalable Real-Time Agentic AI With Streaming Agents, dernier accès : août 20, 2025, https://investors.confluent.io/news-releases/news-release-details/confluent-unlocks-scalable-real-time-agentic-ai-streaming-agents
27. Zapier MCP—Connect your AI to any app instantly, dernier accès : août 20, 2025, https://zapier.com/mcp
28. Confluence Server MCP AI | Zapier, dernier accès : août 20, 2025, https://zapier.com/mcp/confluence
29. Zapier MCP: Perform tens of thousands of actions in your AI tool, dernier accès : août 20, 2025, https://zapier.com/blog/zapier-mcp-guide/
30. Confluent Integration | Pinecone, dernier accès : août 20, 2025, https://www.pinecone.io/confluent-integration/
31. Generative Artificial Intelligence (GenAI) - Confluent, dernier accès : août 20, 2025, https://www.confluent.io/generative-ai/
32. New MongoDB Atlas Vector Search Capabilities Help Developers Build And Scale AI Applications, dernier accès : août 20, 2025, https://www.mongodb.com/company/newsroom/press-releases/new-mongodb-atlas-vector-search-capabilities-help-developers-build-and-scale-ai-applications
33. Building a Real-Time Vector Database for RAG made easy with Confluent Cloud and MongoDB Atlas - YouTube, dernier accès : août 20, 2025, https://www.youtube.com/watch?v=Sv_59o66hRE
34. Vertex AI Agent Engine - Google Cloud - Medium, dernier accès : août 20, 2025, https://medium.com/google-cloud/ai-agents-8eb2b6edea9b
35. Integrating Confluent and Vertex AI with LLMs | Google Cloud Blog, dernier accès : août 20, 2025, https://cloud.google.com/blog/topics/partners/integrating-confluent-and-vertex-ai-with-llms
36. How to Rank Salesforce Leads Using Flink with the Google Vertex AI, dernier accès : août 20, 2025, https://www.confluent.io/blog/supercharge-lead-scoring-with-apache-flink-and-google-cloud-vertex-ai/

37. A Maturity Model for Collaborative Agents in Human-AI Ecosystems - ResearchGate, dernier accès : août 20, 2025, https://www.researchgate.net/publication/374010348_A_Maturity_Model_for_Collaborative_Agents_in_Human-AI_Ecosystems
38. Definition of Multiagent Systems - IT Glossary - Gartner, dernier accès : août 20, 2025, https://www.gartner.com/en/information-technology/glossary/multiagent-systems
39. How to Build a Multi-Agent AI System : In-Depth Guide, dernier accès : août 20, 2025, https://www.aalpha.net/blog/how-to-build-multi-agent-ai-system/
40. The AI Security Guide - Non-Human Identity Management Group - NHI Forum, dernier accès : août 20, 2025, https://nhimg.org/community/agentic-ai-and-nhis/token-security-announces-new-ai-security-guide/
41. 10 AI Governance Best Practices for Enterprise Teams - Knostic AI, dernier accès : août 20, 2025, https://www.knostic.ai/blog/ai-governance-best-practices
42. AI Governance Best Practices: How to Balance Security with Innovation - DTEX Systems, dernier accès : août 20, 2025, https://www.dtexsystems.com/blog/ai-governance-best-practices/
43. Case Study: From NHI Security to AI Agent Governance - Non-Human Identity Management Group, dernier accès : août 20, 2025, https://nhimg.org/community/agentic-ai-and-nhis/case-study-from-nhi-security-to-ai-agent-governance/
44. How is Apache Pulsar different from Apache Kafka? - Milvus, dernier accès : août 20, 2025, https://milvus.io/ai-quick-reference/how-is-apache-pulsar-different-from-apache-kafka
45. Apache Kafka vs. Apache Pulsar: Differences & Comparison - GitHub, dernier accès : août 20, 2025, https://github.com/AutoMQ/automq/wiki/Apache-Kafka-vs.-Apache-Pulsar:-Differences-&-Comparison
46. Kafka vs. Pulsar : Pick the Right one for your Business - XenonStack, dernier accès : août 20, 2025, https://www.xenonstack.com/blog/kafka-vs-pulsar