Spécifications et évolutions en avril 2025

Le Model Context Protocol (MCP) a une spécification publique tenue à jour par Anthropic et la communauté, la révision la plus récente étant datée du 26 mars 2025 (Key Changes - Model Context Protocol). Au début d'avril 2025, aucune nouvelle version officielle de la spécification n'a été publiée, mais plusieurs discussions étaient en cours autour des prochaines évolutions. La révision *2025-03-26* avait introduit des changements majeurs, notamment un cadre d'**authentification OAuth 2.1** pour sécuriser l'accès aux outils (Key Changes - Model Context Protocol), un transport HTTP **streamable** plus flexible en remplacement du transport HTTP+SSE précédent (Key Changes - Model Context Protocol), la prise en charge du **JSON-RPC batch** pour envoyer des requêtes groupées (Key Changes - Model Context Protocol), ainsi que de nouveaux **métadonnées d'outils** (annotations indiquant si un outil est en lecture seule, potentiellement destructif, etc.) (Key Changes - Model Context Protocol). Des ajouts mineurs au schéma JSON ont également été faits, par exemple un champ `message` dans les notifications de progression et le support des données audio en plus du texte et des images (Key Changes - Model Context Protocol). Ces mises à jour reflètent l'effort constant pour enrichir le protocole avant sa standardisation formelle.

En avril 2025, la communauté travaille déjà sur la prochaine mouture de MCP. Le *roadmap* officiel (publié le 27 mars 2025) mentionne plusieurs axes de développement, tels que l'amélioration du **streaming** (messages par morceaux et communication bidirectionnelle pour plus d'interactivité), le support de nouvelles **modalités** (vidéo, etc.) et la notion de **graphs d'agents** permettant à des agents multiples de collaborer via MCP (Roadmap - Model Context Protocol) (Roadmap - Model Context Protocol). Une attention particulière est portée à la **gouvernance** et à la standardisation transparente du protocole, avec l'idée de le faire évoluer de manière communautaire et possiblement via des instances de normalisation officielles (Roadmap - Model Context Protocol). Par ailleurs, des initiatives parallèles suggèrent l'intérêt de standardiser l'interopérabilité des agents IA : par exemple, un *Internet-Draft* indépendant publié le 21 avril 2025 propose un schéma d'URI `agent://` pour adresser et invoquer des agents de façon générique ([

```
        draft-narvaneni-agent-uri-01
](https://datatracker.ietf.org/doc/html/draft-narvaneni-agent-uri-01#:~:text=Internet,Expi
        draft-narvaneni-agent-uri-01
](https://datatracker.ietf.org/doc/html/draft-narvaneni-agent-uri-01#:~:text=The%20agent%3
```

Nouvelles implémentations (serveurs, clients, hosts)

Plusieurs outils et plateformes ont ajouté ou étendu leur support de MCP durant le mois d'avril 2025 :

- **Visual Studio Code (Agent Mode)** : Le 7 avril, l'éditeur VS Code a annoncé le déploiement généralisé du *Copilot Agent Mode*, qui utilise MCP pour étendre les capacités de l'agent IA intégré (Agent mode: available to all users and supports MCP) (Agent mode: available to all users and supports MCP). Concrètement, VS Code agit comme **hôte MCP** (client) pouvant se connecter à des serveurs MCP externes. L'agent Copilot peut désormais utiliser des « outils » fournis soit par des **extensions VS Code**, soit par des **serveurs MCP** tiers, afin d'accomplir des tâches complexes : naviguer dans un codebase, exécuter des commandes shell, requêter une base de données, interagir avec GitHub, etc. (Agent mode: available to all users and supports MCP) (Agent mode: available to all users and supports MCP). Cette intégration s'inspire explicitement du succès du Language Server Protocol (LSP) pour les IDE, en visant une standardisation analogue pour les outils à destination des LLM (Agent mode: available to all users and supports MCP). L'Agent Mode dans VS Code permet ainsi à l'IA d'accéder à *n'importe quel contexte ou service* via MCP, plutôt que d'être limitée aux données locales (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog). *Exemple concret* : un développeur peut connecter l'agent Copilot à un serveur MCP donnant accès à son dépôt GitHub. L'agent sera alors capable, sur une simple requête en langage naturel, de retrouver le titre d'une Pull Request assignée la veille et de mettre à jour le profil GitHub en conséquence, en enchaînant automatiquement les appels d'API requis (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog) (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog). VS Code supporte les transports MCP en **STDIO** (pour serveurs locaux) et **Server-Sent Events (SSE)** pour les serveurs distants (Agent mode: available to all users and supports MCP), et fournit une interface de configuration simplifiée (fichier `mcp.json` ou via l'UI) pour enregistrer de nouveaux serveurs MCP dans l'éditeur (Agent mode: available to all users and supports MCP).

- **GitHub – Serveur MCP officiel** : GitHub a publié son **propre serveur MCP open-source** début avril, en *public preview*. Annoncé le 4 avril lors d'un billet sur le blog GitHub (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog), le *GitHub MCP Server* vise à permettre à toute application compatible MCP d'interagir avec les fonctionnalités de GitHub (issues, PR, recherche de code, etc.). Ce serveur, développé en Go en collaboration avec Anthropic, remplace la référence précédemment fournie par Anthropic (GitHub Announces Public Preview of GitHub MCP Server - InfoQ) (GitHub Announces Public Preview of GitHub MCP Server - InfoQ). Il apporte des outils spécialisés, par exemple la recherche sur l'ensemble des dépôts, la gestion des issues/PR, l'analyse de code via CodeQL, ainsi qu'une fonction `get_me` facilitant les interactions en langage naturel avec des dépôts privés (GitHub Announces Public Preview of GitHub MCP Server - InfoQ) (GitHub Announces Public Preview of GitHub MCP Server - InfoQ). En fournissant ce serveur clé en main, GitHub *« dote l'Agent Copilot de pouvoirs d'utilisateur GitHub »*, selon l'expression de leurs équipes (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog) (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog). L'impact a été immédiat dans la communauté : ce serveur GitHub est désormais nativement pris en charge dans VS Code (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog), et son dépôt sur GitHub a rapidement attiré des milliers d'étoiles. Cette initiative s'inscrit dans la stratégie plus large de

GitHub et Microsoft de rendre Copilot plus *agentique*, capable d'actions autonomes multi-étapes et connecté à l'écosystème du développeur (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog) (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog).

- **Microsoft Copilot Studio & Power Platform** : Le 28 avril, Microsoft a dévoilé un **laboratoire d'intégration MCP** pour Copilot Studio, illustrant comment relier les agents "Copilot" à des serveurs MCP personnalisés (Visual Studio Magazine: Contact Information, Journalists, and Overview | Muck Rack) (Visual Studio Magazine: Contact Information, Journalists, and Overview | Muck Rack). Copilot Studio est la plateforme low-code de Microsoft pour créer des agents IA personnalisés ; grâce à MCP, ces agents peuvent dépasser leurs capacités natives en consommant des outils externes. Microsoft a démontré qu'un serveur MCP peut être exposé dans Copilot Studio via l'infrastructure des **connecteurs Power Platform** existants (Microsoft Copilot Studio ❤️ MCP | Power Platform Developer Blog). En pratique, cela signifie qu'un serveur MCP apparait comme un connecteur sécurisé dans l'agent : il hérite de toutes les garanties d'**authentification**, de **DLP (Data Loss Prevention)** et d'**intégration réseau** offertes par la plateforme Power Platform (Microsoft Copilot Studio ❤️ MCP | Power Platform Developer Blog). Microsoft insiste sur le fait que MCP et les connecteurs traditionnels sont *« meilleurs ensemble »* : MCP apporte flexibilité et standardisation, pendant que les connecteurs assurent la robustesse et la gouvernance d'entreprise (Microsoft Copilot Studio ❤️ MCP | Power Platform Developer Blog). Le lab publié (dépôt `microsoft/mcsmcp`) guide les développeurs pour : 1) **déployer un serveur MCP** (exemple : un serveur de blagues), 2) **créer un connecteur personnalisé** pointant vers ce serveur, 3) **intégrer ce connecteur MCP** comme action dans un agent Copilot Studio (Microsoft Copilot Studio ❤️ MCP | Power Platform Developer Blog) (Microsoft Copilot Studio ❤️ MCP | Power Platform Developer Blog). Ainsi armé, un agent Copilot Studio peut interroger en temps réel une source de données tierce (par ex. raconter une blague fournie par le serveur MCP) au cours d'une conversation (Microsoft Copilot Studio ❤️ MCP | Power Platform Developer Blog). Cette démarche illustre l'intégration de MCP dans un contexte *low-code* business, en exploitant l'écosystème de connecteurs déjà en place pour les questions de sécurité et conformité. Enfin, Microsoft a publié en parallèle un **tutoriel vidéo** présentant l'intégration MCP dans Copilot Studio (Microsoft Copilot Studio ❤️ MCP | Power Platform Developer Blog), soulignant l'importance stratégique de MCP dans sa suite d'outils professionnels (#CopilotStudio).

- **Amazon Q Developer CLI** : Amazon a annoncé le 29 avril la prise en charge de MCP dans **Amazon Q** (son assistant IA pour développeurs) via la CLI Q Developer (Amazon Q Developer CLI now supports Model Context Protocol (MCP) - AWS). Auparavant, Q CLI ne pouvait utiliser que les outils intégrés fournis par AWS, mais avec MCP, les développeurs peuvent désormais **brancher des serveurs MCP externes** pour enrichir les capacités de Q CLI (Amazon Q Developer CLI now supports Model Context Protocol (MCP) - AWS). Amazon souligne que cela ouvre l'accès à de nombreuses intégrations *préexistantes* d'AWS (via des MCP servers compatibles) ainsi qu'à des serveurs communautaires, notamment via le transport **stdio** supporté nativement (Amazon Q Developer CLI now supports Model Context Protocol (MCP) - AWS). Un billet technique d'AWS détaille un cas d'usage parlant : intégrer un serveur MCP PostgreSQL pour que Q CLI connaisse le schéma d'une base de données métier (Extend the Amazon Q Developer CLI with Model Context

Protocol (MCP) for Richer Context | AWS DevOps & Developer Productivity Blog) (Extend the Amazon Q Developer CLI with Model Context Protocol (MCP) for Richer Context | AWS DevOps & Developer Productivity Blog). En configurant un fichier `mcp.json` local avec le lancement du serveur Postgres (via la commande NPM `@modelcontextprotocol/server-postgres`), l'assistant Q est devenu capable d'énumérer les tables de la base et de générer des requêtes SQL précises avec le contexte du schéma (Extend the Amazon Q Developer CLI with Model Context Protocol (MCP) for Richer Context | AWS DevOps & Developer Productivity Blog) (Extend the Amazon Q Developer CLI with Model Context Protocol (MCP) for Richer Context | AWS DevOps & Developer Productivity Blog). Ainsi, une demande du type *« liste-moi les tables de la BD »* ou *« écris la requête SQL pour les crédits par étudiant »* donne dorénavant des résultats corrects basés sur les tables réelles, là où auparavant l'IA ne pouvait proposer qu'une requête générique faute de contexte (Extend the Amazon Q Developer CLI with Model Context Protocol (MCP) for Richer Context | AWS DevOps & Developer Productivity Blog) (Extend the Amazon Q Developer CLI with Model Context Protocol (MCP) for Richer Context | AWS DevOps & Developer Productivity Blog). Amazon indique travailler à étendre cette fonctionnalité MCP à ses **plugins IDE** (par ex. pour VS Code ou Cloud9) dans les semaines suivant l'annonce (Extend the Amazon Q Developer CLI with Model Context Protocol (MCP) for Richer Context | AWS DevOps & Developer Productivity Blog).

- **Autres éditeurs et outils** : L'adoption de MCP s'est propagée chez divers éditeurs d'IDE et d'assistants de code en avril. L'éditeur **Zed** (outil de code collaboratif) avait intégré MCP plus tôt dans l'année et a livré en avril des mises à jour corrigeant des bugs de son agent MCP (gestion de l'activation/désactivation des outils) (Preview Release - Zed). **Sourcegraph Cody** et **Replit Ghostwriter** faisaient partie des premiers partenaires annoncés dès novembre 2024 (Introducing the Model Context Protocol \ Anthropic), et on observe qu'en avril leurs solutions poursuivent l'intégration de MCP (par ex. Replit a collaboré avec Anthropic sur des serveurs contextuels de documentation de code). De même, **Claude Desktop** d'Anthropic supporte depuis son lancement les serveurs MCP locaux (permettant à Claude d'utiliser des outils locaux sur la machine de l'utilisateur) (Introducing the Model Context Protocol \ Anthropic). Enfin, **Codeium**, un assistant IA open source, a indiqué travailler avec le protocole pour élargir ses connecteurs. Cette effervescence se voit aussi dans la communauté open-source : un registre non-officiel comme *Smithery* recense **plus de 4 000 outils MCP** disponibles en avril 2025 (4 431 exactement début avril) (Accelerating LLM-Powered Apps with MCP and A2A Protocols | by Roberto Infante | Apr, 2025 | Medium), couvrant un spectre allant de la recherche web aux assistants de codage. En somme, le mois d'avril a marqué une montée en puissance notable du nombre de **hosts MCP** (VS Code, Q CLI, Copilot Studio, etc.) et de **serveurs MCP spécialisés**, consolidant l'écosystème.

SDK et intégrations tierces (frameworks, bibliothèques)

L'écosystème MCP ne se limite pas aux IDE : durant cette période, de nombreux frameworks et bibliothèques tiers ont ajouté un support MCP pour faciliter l'intégration dans des applications IA personnalisées.

- **SDK officiels (Python, TS, etc.)** : Les SDK fournis par l'organisation *modelcontextprotocol* sur GitHub ont suivi l'évolution du protocole. Une version 1.6.0 du SDK Python est sortie fin mars

2025 (GitHub - modelcontextprotocol/python-sdk: The official Python SDK for Model Context Protocol servers and clients) pour aligner le schéma avec la spec MCP 2025-03-26. En avril, ces SDK ont été adoptés par plusieurs projets : par exemple, **LangChain**, le framework populaire d'orchestration d'agents, a introduit un adaptateur MCP permettant de connecter un agent LangChain à n'importe quel serveur MCP en quelques lignes de code (What Is MCP?. MCP (Model Context Protocol) is a new… | by Ronny Roeller | NEXT Engineering | Mar, 2025 | Medium) (What Is MCP?. MCP (Model Context Protocol) is a new… | by Ronny Roeller | NEXT Engineering | Mar, 2025 | Medium). Cela signifie qu'on peut, via LangChain, brancher des outils MCP existants (Google Drive, Weather API, etc.) sans écrire de logique spécifique, l'agent choisissant dynamiquement quel outil invoquer en fonction du prompt utilisateur (What Is MCP?. MCP (Model Context Protocol) is a new… | by Ronny Roeller | NEXT Engineering | Mar, 2025 | Medium). De même, le **framework *Semantic Kernel*** de Microsoft a ajouté un support complet MCP dans sa version Python 1.28.1 (17 avril) (Semantic Kernel adds Model Context Protocol (MCP) support for Python | Semantic Kernel). Semantic Kernel (SK) peut désormais agir comme **client MCP** pour consommer des serveurs (en local via STDIO ou à distance via SSE/WebSocket), et aussi s'exposer lui-même comme **serveur MCP** fournissant des fonctions ou des "skills" SK à d'autres agents (Semantic Kernel adds Model Context Protocol (MCP) support for Python | Semantic Kernel) (Semantic Kernel adds Model Context Protocol (MCP) support for Python | Semantic Kernel). Microsoft fournit des exemples illustrant la connexion d'un plugin SK à un serveur MCP local via stdio ou à un serveur distant SSE en une poignée de lignes Python (Semantic Kernel adds Model Context Protocol (MCP) support for Python | Semantic Kernel) (Semantic Kernel adds Model Context Protocol (MCP) support for Python | Semantic Kernel). Cette convergence permet d'orchestrer plusieurs agents SK et outils en chaîne, renforçant l'interopérabilité.

- **Intégration dans des plateformes d'intégration** : En avril, MuleSoft (Salesforce) a lancé en bêta la **MuleSoft MCP Support** dans son produit d'intégration Anypoint Platform (MuleSoft MCP Support: Getting Started With MCP | MuleSoft Blog). L'objectif est de permettre aux flux Mule (flows d'intégration) d'agir en tant que serveurs MCP ou clients MCP. Concrètement, MuleSoft a créé des composants de configuration *Mule* pour définir un endpoint SSE MCP dans une application Mule et y associer des "tools" – par exemple un flow Mule réalisant une requête dans SAP peut être exposé comme un outil MCP utilisable par un agent IA (MuleSoft MCP Support: Getting Started With MCP | MuleSoft Blog) (MuleSoft MCP Support: Getting Started With MCP | MuleSoft Blog). Inversement, un flow Mule peut se comporter en client MCP et invoquer un outil distant via une étape de son pipeline. Ce support MCP pour MuleSoft (nom de code *Agentforce*) vise à *« connecter les agents IA aux systèmes d'entreprise »*, exploitant la richesse de l'écosystème MuleSoft (plus de 200 connecteurs API) via MCP (MuleSoft MCP Support: Getting Started With MCP | MuleSoft Blog) (MuleSoft MCP Support: Getting Started With MCP | MuleSoft Blog). Cette annonce est importante car elle amène MCP dans le monde de l'intégration d'entreprise et des ESB, domaines où la gouvernance et la fiabilité priment.

- **Autres frameworks et services** : Fin mars et début avril, on a vu apparaître des connecteurs MCP dans d'autres bibliothèques. **LangChain**, déjà mentionné, et d'autres frameworks d'agents open-source comme Haystack ou LlamaIndex ont expérimenté l'utilisation de MCP pour remplacer des *tools* propriétaires. Du côté cloud, **Cloudflare** a publié un guide pour déployer facilement des

serveurs MCP distants sur sa Workers Platform (Will Model Context Protocol (MCP) Become the Standard for Agentic ...) (Build and deploy Remote Model Context Protocol (MCP) servers to ...), en tirant parti de l'élasticité du cloud (on compare *« le support MCP distant à la transition du logiciel desktop vers le SaaS »*, soulignant l'importance de fournir le contexte à l'IA même hors du poste de travail) (Will Model Context Protocol (MCP) Become the Standard for Agentic ...). Signalons aussi **PayPal** qui, d'après InfoQ, a adopté MCP dans certaines de ses solutions internes ou API (GitHub Announces Public Preview of GitHub MCP Server - InfoQ). L'entreprise Descope a noté dans un article que *« plusieurs éditeurs d'IDE ont adopté le protocole, y compris Cursor IDE »* et mentionne la compatibilité de frameworks comme LangChain et même **HuggingFace Transformers** avec MCP pour l'accès aux fonctions avancées (via l'API de `tools` ou de génération de code) (Model Context Protocol (MCP) Explained - by Nir Diamant - DiamantAI). L'ensemble de ces SDK et intégrations tierces démontre que MCP est en train de devenir un **standard de facto** : les développeurs peuvent l'utiliser au sein de leurs propres applications (via LangChain, SK, etc.) tout en bénéficiant d'une **bibliothèque croissante de serveurs** prêts à l'emploi.

Retours d'expérience, usages et limites constatés

Plusieurs articles de blog, présentations et retours d'utilisateurs ont été publiés en avril pour partager les **bénéfices concrets** de MCP et aussi souligner ses limites ou points d'attention.

- **Productivité et cas d'usage concrets** : Un article sur The New Stack qualifie MCP de *« nouveau gold standard pour intégrer des ressources externes dans les workflows agentiques »*, arguant qu'il élimine la nécessité de coder des intégrations sur mesure pour chaque outil (Model Context Protocol: A Primer for the Developers - The New Stack). De fait, les témoignages soulignent un **gain de temps significatif** pour les développeurs. Par exemple, chez AWS, l'ajout d'un serveur MCP Postgres dans Q CLI a permis de générer automatiquement du code SQL exact et documenté, là où auparavant le développeur devait manuellement fournir le schéma de la base ou écrire la requête lui-même (Extend the Amazon Q Developer CLI with Model Context Protocol (MCP) for Richer Context | AWS DevOps & Developer Productivity Blog) (Extend the Amazon Q Developer CLI with Model Context Protocol (MCP) for Richer Context | AWS DevOps & Developer Productivity Blog). De même, des utilisateurs de VS Code Insiders ayant activé l'Agent Mode dès février ont partagé sur les réseaux sociaux des scénarios réussis : génération automatique d'applications web complètes (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog), corrections de bugs « en boucle fermée » (l'agent code puis exécute les tests et corrige jusqu'à réussite) (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog), etc. Ces *success stories* attribuent en partie leur réussite à MCP qui permet à l'agent d'accéder aux bonnes données ou outils au bon moment. Un autre exemple concret partagé sur Reddit : un développeur a branché un serveur MCP "Jira" et "Confluence" à son assistant ; il a pu en langage naturel demander à l'IA de collecter des tickets liés à un bug et de rédiger une synthèse dans Confluence, tâche accomplie en quelques minutes là où il aurait fallu naviguer manuellement entre les applications (témoignage indirect via un meetup Slack, avril 2025).

- **Écosystème et communauté** : Le rapide enrichissement de l'écosystème MCP a été noté par plusieurs blogueurs. Roberto Infante décrit comment la communauté a résolu le *« problème M×N des intégrations »* : traditionnellement, brancher *chaque* LLM sur *chaque* source de données nécessitait N connecteurs × M modèles, alors que **MCP standardise une interface unique** (Google Adopts Anthropic's MCP: Pioneering AI Data Connectivity). Ainsi, au lieu que chaque fournisseur propose son plugin propriétaire (fragmentation), un serveur MCP peut fonctionner avec n'importe quel client compatible. Au 15 avril, on recensait plus de **4 400 outils MCP publics** (Accelerating LLM-Powered Apps with MCP and A2A Protocols | by Roberto Infante | Apr, 2025 | Medium) (chiffre issu de Smithery.ai) – un chiffre en forte croissance qui illustre l'engouement. Parmi ces serveurs figurent des connecteurs pour la majorité des services cloud (Google Drive, Slack, Notion, GitHub, bases SQL, mais aussi des outils plus originaux comme des solveurs de Sudoku ou des "outils de pensée séquentielle" pour aider l'IA à structurer ses raisonnements (Smithery - Model Context Protocol Registry) (Smithery - Model Context Protocol Registry)). Des *meetups* et *webinaires* se sont multipliés : Anthropic a organisé une session de Q/R sur Discord mi-avril où des early adopters (de Block, Replit, etc.) ont partagé leurs retours. On a vu aussi apparaître un sous-reddit dédié (r/MCP) où des employés de Microsoft et GitHub sont venus répondre aux questions de la communauté lors du lancement du serveur GitHub (GitHub Announces Public Preview of GitHub MCP Server - InfoQ) (GitHub Announces Public Preview of GitHub MCP Server - InfoQ). Ce dialogue ouvert a rassuré de nombreux développeurs sur la pérennité du protocole.

- **Bénéfices constatés** : Les bénéfices le plus souvent cités sont : (1) **Réduction de l'isolation des IA** : MCP permet à un agent d'*« sortir de sa bulle » et d'accéder à de l'information à jour, ce qui améliore la pertinence des réponses (Introducing the Model Context Protocol \ Anthropic) (Introducing the Model Context Protocol \ Anthropic). (2) **Moins de code "glue"** : là où avant chaque intégration demandait du code spécifique, les développeurs peuvent réutiliser des serveurs existants ou n'écrire qu'une fine couche pour exposer un système interne via MCP (Introducing the Model Context Protocol \ Anthropic) (Introducing the Model Context Protocol \ Anthropic). (3) **Interopérabilité multi-outils** : grâce à la standardisation JSON-RPC, un même agent peut enchaîner des appels à divers services hétérogènes dans un seul flux cohérent. Par exemple, un article Medium démontre un agent qui, en traitant une requête utilisateur, va successivement : faire une recherche web (via un serveur MCP de recherche), extraire du texte d'une page (via un MCP "Browser"), puis utiliser une API d'envoi d'email – trois actions très différentes, orchestrées uniformément grâce à MCP (Google Adopts Anthropic's MCP: Pioneering AI Data Connectivity) (Google Adopts Anthropic's MCP: Pioneering AI Data Connectivity). (4) **Maintien du contexte sur la durée** : MCP étant conçu pour des sessions stateful, plusieurs retours notent que l'IA parvient mieux à chaîner des tâches car le protocole lui permet de **garder en mémoire les interactions** passées avec les outils (par ex. se souvenir des données chargées précédemment) (Accelerating LLM-Powered Apps with MCP and A2A Protocols | by Roberto Infante | Apr, 2025 | Medium). Cette continuité contextuelle, combinée à la capacité de *découvrir dynamiquement* de nouveaux outils disponibles, donne aux agents une plus grande autonomie.

- **Limites et défis identifiés** : Malgré son potentiel, MCP a aussi ses limites pratiques en 2025. D'abord, plusieurs développeurs soulignent la **courbe d'apprentissage** pour composer

efficacement avec des dizaines d'outils : un agent non guidé peut essayer trop d'outils inutiles (*tool overload*), ce qui alourdit les échanges. Des benchmarks informels montrent qu'au-delà de ~10 outils disponibles, les modèles actuels peuvent avoir du mal à choisir pertinemment, d'où la recommandation de n'activer que les outils nécessaires pour une tâche donnée (principe appliqué dans VS Code en incitant l'utilisateur à filtrer les outils actifs) (Agent mode: available to all users and supports MCP) (Agent mode: available to all users and supports MCP). Ensuite, la question de la **latence** est posée : un article sur BigDataWire cite des ingénieurs de Cloudflare notant qu'utiliser un serveur MCP distant ajoute de la latence réseau et de la complexité (dépendance à un service externe), par rapport à des fonctions intégrées locales (Will Model Context Protocol (MCP) Become the Standard for Agentic ...) (Build and deploy Remote Model Context Protocol (MCP) servers to ...). Toutefois, ils comparent cela à la transition du local vers le cloud – inévitable pour bénéficier de données live. Autre limite, plus conceptuelle : MCP ne résout pas tout le problème de coordination entre agents multiples. Google ayant introduit en avril un protocole complémentaire, Agent-2-Agent (A2A), pour la communication directe entre *agents IA* eux-mêmes (MCP vs A2A: Which Protocol Is Better For AI Agents? [2025] | Blott Studio) (MCP vs A2A: Which Protocol Is Better For AI Agents? [2025] | Blott Studio), certains analystes voient MCP et A2A comme deux briques adressant des besoins différents mais adjacents (MCP pour agent↔outil, A2A pour agent↔agent). Des présentations en meetup ont discuté comment combiner les deux – par exemple un agent principal qui utilise MCP pour les outils et A2A pour déléguer subtasks à d'autres agents. Enfin, sur les limites **cognitives** : Andrej Karpathy (ex-directeur IA de Tesla) a plaisanté lors d'une conf fin avril que les nouveaux agents "foufous" dopés au MCP sont *« comme des stagiaires surdoués mais hyperactifs »* et qu'il faut *les tenir en laisse courte* pour filtrer leurs idées farfelues (Visual Studio Magazine: Contact Information, Journalists, and Overview | Muck Rack). Cela pointe la nécessité de conserver **un contrôle humain** et des garde-fous, car un agent pouvant toucher à de nombreux systèmes peut aussi faire des erreurs en cascade s'il interprète mal une instruction.

En synthèse, les retours d'expérience en avril 2025 sont globalement enthousiastes quant à la capacité de MCP à **accélérer les workflows IA** et à **décloisonner les données** pour les modèles, tout en reconnaissant qu'une adoption maîtrisée et progressive est nécessaire pour en tirer le meilleur parti.

Sécurité, permissionnement et gouvernance

La question de la **sécurité** et de la **gouvernance** autour de MCP a été centrale dans les discussions d'avril, d'autant que le protocole permet à une IA d'effectuer potentiellement des actions à large portée. Voici les principaux éléments abordés :

- **Authentification et autorisations** : La spécification MCP intègre désormais un véritable cadre d'authentification OAuth 2.1 pour les interactions client-serveur (Key Changes - Model Context Protocol). En pratique, cela signifie que les serveurs MCP peuvent exiger des jetons d'accès et vérifier les permissions de l'agent appelant avant d'exécuter une action. Par exemple, le serveur GitHub officiel requiert un **GitHub Personal Access Token** pour pouvoir effectuer des opérations sur les repos privés de l'utilisateur (GitHub Announces Public Preview of GitHub MCP Server - InfoQ). Cette approche évite de devoir partager des identifiants ou API keys directement avec le

modèle : le client (hôte) détient le token et s'authentifie auprès du serveur MCP, qui ensuite ne fournit à l'IA que les données autorisées. Microsoft, de son côté, exploite son système de **connecteurs** : lorsqu'un serveur MCP est intégré via Power Platform, il bénéficie automatiquement du système OAuth2 des connecteurs (Azure AD, etc.) et du stockage sécurisé des credentials dans la plateforme (Microsoft Copilot Studio ❤️ MCP | Power Platform Developer Blog). Autrement dit, l'IA n'a jamais directement accès aux secrets, elle ne voit que le résultat des appels autorisés. C'est un modèle de **confiance intermédiaire** : on fait confiance au client (l'hôte) pour contrôler l'authentification, et le serveur MCP applique les permissions en amont.

- **Permissionnement granulaire des actions** : Du côté *host*, les implémenteurs ont ajouté des gardes fous pour que l'agent IA ne prenne pas de décisions potentiellement destructrices sans validation. Visual Studio Code, par exemple, **requiert l'approbation explicite de l'utilisateur** pour chaque invocation d'outil MCP qui n'est pas en lecture seule (Agent mode: available to all users and supports MCP). Lorsqu'un agent Copilot veut appeler un outil (par ex. supprimer un fichier via un MCP server), l'interface VS Code affiche la demande et attend que l'utilisateur la valide pour la session courante, le workspace ou définitivement (Agent mode: available to all users and supports MCP). Par défaut, aucun outil potentiellement critique n'est exécuté de façon automatique en arrière-plan. Cette transparence se double d'une journalisation : toutes les actions de l'agent sont visibles dans l'UI, ce qui crée un **audit trail** des opérations IA. D'autres clients adoptent des approches similaires : Claude Desktop demande confirmation quand un outil veut envoyer un email ou poster un message Slack via MCP, etc. L'idée directrice est de **garder l'humain dans la boucle** pour les opérations ayant un impact externe, afin d'éviter toute dérive ou abus si le modèle venait à mal comprendre une instruction.

- **Isolation et sandboxing** : Pour limiter les dégâts potentiels d'un agent qui ferait une action imprévue, les développeurs recommandent d'isoler l'environnement d'exécution. VS Code suggère par exemple d'utiliser des **Dev Containers** Docker pour exécuter l'agent : ainsi, si l'IA supprime un fichier ou installe un package, ces changements restent confinés dans le conteneur et n'affectent pas l'OS hôte (Agent mode: available to all users and supports MCP). De même, GitHub a souligné que son agent Copilot ne peut pas, par conception, pousser du code sur un dépôt sans approbation, même si techniquement le serveur MCP GitHub le pourrait. Des limites sont codées côté client. Une discussion publique autour d'un éventuel **mode "no sandbox"** (où l'agent aurait carte blanche) a eu lieu sur le subreddit /mcp, mais la majorité estime que ce serait trop risqué en environnement non contrôlé. **Conseil** : en entreprise, dédier des environnements de test pour les agents, avec des comptes limités, avant de les brancher sur des comptes de production.

- **Gouvernance des accès et conformité** : Les entreprises adoptant MCP veulent s'assurer que cela respecte leurs politiques de sécurité. Là, l'approche de Microsoft via Power Platform est notable : en faisant transiter les appels MCP par les connecteurs, toutes les règles de **DLP (Data Loss Prevention)** s'appliquent. Par exemple, un administrateur peut interdire qu'un agent Copilot (via MCP) transfère des données d'un système classifié "confidentiel" vers un autre classé "public", comme il le ferait pour un flux d'intégration classique. MuleSoft mise sur une stratégie similaire, proposant d'encapsuler les outils MCP dans des flows Mule où l'on peut appliquer des politiques

(par ex. chiffrer/déchiffrer certaines données sensibles entrant ou sortant du MCP). **Audit** : les serveurs MCP eux-mêmes peuvent implémenter des logs d'activité et même des quotas. Rien n'empêche un serveur de limiter le nombre de requêtes d'un client par minute ou de désactiver certains outils en fonction du profil de l'utilisateur. Ces considérations commencent à émerger : aucune faille de sécurité (CVE) publique n'a été reportée sur MCP en avril, mais les experts en sécurité anticipent que des audits approfondis seront nécessaires, notamment sur la **surface d'attaque** qu'ouvre un agent intelligent (par ex. un agent connectant un outil web pourrait être amené à cliquer sur des liens malveillants). Des discussions sont en cours pour définir des **lignes directrices de sécurité** dans la documentation MCP, incluant la nécessité de valider/censurer les contenus ramenés par un outil externe (éviter injections via réponses d'API, etc.).

- **Retours et débats publics** : Un sujet de débat concerne la **confiance dans les serveurs MCP tiers**. Puisque tout développeur peut publier un serveur MCP, comment s'assurer qu'il est sûr ? Une proposition discutée est de mettre en place une **registry avec vérification**. Anthropic a mentionné travailler sur un *MCP Registry* centralisé pour cataloguer les serveurs avec métadonnées (Roadmap - Model Context Protocol). En attendant, des initiatives communautaires comme *Smithery* permettent aux utilisateurs de noter et commenter les serveurs. L'idée d'une **signature numérique** des serveurs (pour attester qu'un binaire d'outil n'a pas été altéré) a aussi été évoquée. Par exemple, Cloudflare propose d'héberger des serveurs MCP sur son infra avec un sceau de confiance. Enfin, un autre débat touche aux **performances** : la couche MCP ajoute un overhead JSON-RPC, mais jugé acceptable étant donné les bénéfices. Certains envisagent d'éventuelles *optimisations* (compression, protocoles binaires) si MCP devait prendre un rôle de très bas niveau, mais ce n'est pas prioritaire pour l'instant – la flexibilité et la lisibilité JSON étant privilégiées en phase d'adoption.

En résumé, la sécurité de MCP repose sur une défense en profondeur : authentification robuste, approbation humaine, isolation technique, et gouvernance via les outils existants. En avril 2025, ces mécanismes ont permis de **lancer MCP en preview sans incident majeur**, tout en sensibilisant la communauté aux bonnes pratiques pour éviter les pièges d'une IA trop « libre ».

Références et ressources clés

Afin d'approfondir la veille MCP d'avril 2025, voici une liste de références importantes, incluant dépôts GitHub, articles officiels et ressources communautaires :

- **Spécification officielle et schéma JSON** : Le site **modelcontextprotocol.io** héberge la spécification formelle du protocole (version *2025-03-26*) ainsi que le schéma TypeScript/JSON correspondant (Specification - Model Context Protocol) (Specification - Model Context Protocol). On y trouve également le changelog détaillé (Key Changes - Model Context Protocol) (Key Changes - Model Context Protocol) et un guide d'utilisation. Le dépôt GitHub associé, **modelcontextprotocol/schema** (Specification and documentation for the Model Context Protocol), contient le JSON schema et permet de suivre les propositions d'évolutions (issues & pull requests du groupe de standardisation).

- **Dépôts GitHub officiels (organisation *modelcontextprotocol*)** : Anthropic et ses partenaires ont publié plusieurs dépôts de référence sous l'organisation *modelcontextprotocol*. Notamment :
  - **SDKs officiels** : dépôts *python-sdk*, *typescript-sdk*, *java-sdk*, *kotlin-sdk*, *csharp-sdk* (tous MIT). Ils contiennent le code source pour implémenter facilement des clients et serveurs MCP dans ces langages. Ex : le README du python-sdk explique comment démarrer un client en quelques lignes (Accelerating LLM-Powered Apps with MCP and A2A Protocols | by Roberto Infante | Apr, 2025 | Medium).
  - **Servers examples** : dépôt *modelcontextprotocol/servers* (modelcontextprotocol/servers: Model Context Protocol ... - GitHub), rassemblant des implémentations open-source d'exemples de serveurs MCP (Google Drive, Slack, Git, GitHub, Postgres, Puppeteer, etc. (Introducing the Model Context Protocol \ Anthropic)). Ce dépôt communautaire a servi de base à de nombreuses intégrations et a connu des forks actifs (par ex. le fork *cyanheads/github-mcp-server* qui a ensuite été rendu obsolète par la version officielle de GitHub (cyanheads/github-mcp-server)).
  - **Outils** : on trouve aussi des utilitaires comme *mcp-inspector* (outil GUI pour tester un serveur MCP localement) mentionné dans certains tutoriels (Accelerating LLM-Powered Apps with MCP and A2A Protocols | by Roberto Infante | Apr, 2025 | Medium).
- **GitHub MCP Server (dépôt officiel)** : Publié sur **github.com/github/github-mcp-server**, c'est le serveur MCP maintenu par GitHub pour l'intégration de sa plateforme (mcp-server · GitHub Topics). Écrit en Go, il est fourni avec une documentation complète pour l'auto-héberger via Docker et un PAT GitHub (GitHub Announces Public Preview of GitHub MCP Server - InfoQ). Le dépôt inclut également une **collection d'exemples d'utilisation** (fichier README avec cas concrets d'appels JSON-RPC aux outils de repo, d'issues, etc.). Ce projet est open-source (licence MIT) et ouvert aux contributions. *(À noter : le dépôt initial modelcontextprotocol/server-github a été migré ici, comme confirmé par un PM de GitHub sur Reddit (GitHub Announces Public Preview of GitHub MCP Server - InfoQ).)*
- **Autres dépôts et projets notables** :
  - **Awesome MCP Servers** : liste communautaire (ex : *punkpeye/awesome-mcp-servers* sur GitHub) recense les serveurs MCP populaires et innovants (punkpeye/awesome-mcp-servers - GitHub). Utile pour découvrir des outils non officiels (ex : connecteur Linear, serveur Weather, etc.).
  - **LangChain MCP** : le support MCP de LangChain est évoqué dans leur documentation (Module Tools/Agents). Un *pull request* notable de mars 2025 ajoute un wrapper MCP tool, permettant aux agents LangChain de charger un serveur via son endpoint SSE.
  - **MuleSoft MCP (Beta)** : disponible dans le repo Anypoint Exchange de MuleSoft (section Connectors > MCP Connector). Ce n'est pas sur GitHub mais référencé via le blog MuleSoft (MuleSoft MCP Support: Getting Started With MCP | MuleSoft Blog).
- **Blogs officiels et annonces presse** :

- **Anthropic** : *Introducing the Model Context Protocol* (25 nov 2024) (Introducing the Model Context Protocol \ Anthropic) (Introducing the Model Context Protocol \ Anthropic) annonce l'open-source de MCP et liste déjà des intégrations (Block, Replit, etc.). Bonne introduction aux motivations du protocole.
- **GitHub Blog** : *"Vibe coding with GitHub Copilot: Agent mode and MCP support…"* (Thomas Dohmke, 4 avril 2025) (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog) (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog) présente l'arrivée de MCP dans Copilot et le lancement du serveur GitHub. C'est une source primaire sur les fonctionnalités d'Agent Mode, multi-modèles, etc.
- **Microsoft Dev Blogs** : article *"Microsoft Copilot Studio ❤️ MCP"* (A. Dunnam & D. Laskewitz, 28 avril) détaillant l'intégration avec Power Platform (Microsoft Copilot Studio ❤️ MCP | Power Platform Developer Blog) (Microsoft Copilot Studio ❤️ MCP | Power Platform Developer Blog). Sur le blog Semantic Kernel, l'article *"SK adds MCP support for Python"* (17 avril) est également très instructif techniquement (Semantic Kernel adds Model Context Protocol (MCP) support for Python | Semantic Kernel) (Semantic Kernel adds Model Context Protocol (MCP) support for Python | Semantic Kernel).
- **AWS Blog** : *"Extend the Amazon Q Developer CLI with MCP for richer context"* (29 avril) qui illustre avec code et images l'exemple de la base de données LMS (Extend the Amazon Q Developer CLI with Model Context Protocol (MCP) for Richer Context | AWS DevOps & Developer Productivity Blog) (Extend the Amazon Q Developer CLI with Model Context Protocol (MCP) for Richer Context | AWS DevOps & Developer Productivity Blog).
- **InfoQ (press)** : *"GitHub Announces Public Preview of GitHub MCP Server"* (A. Kulkarni, InfoQ, 29 avril 2025) offre un résumé succinct des annonces GitHub/Microsoft (GitHub Announces Public Preview of GitHub MCP Server - InfoQ) (GitHub Announces Public Preview of GitHub MCP Server - InfoQ), utile pour un aperçu rapide. De plus, InfoQ a publié une interview "Creators of MCP" début mars (Latent Space) commentant l'adoption par OpenAI et Google (The Creators of Model Context Protocol - Latent.Space).
- **The New Stack** : *"Model Context Protocol: A Primer for Developers"* (avril 2025) – un article pédagogique qui positionne MCP comme un équivalent du "USB-C de l'IA" pour brancher des outils (GitHub Announces Public Preview of GitHub MCP Server - InfoQ). Il reprend la métaphore utilisée par Cassidy Williams de GitHub sur la standardisation apportée par MCP.
- **Forbes** : *"Why You Need to Know About MCP"* (avril 2025) – article grand public listant les entreprises impliquées (Block, Apollo, etc.) et le bénéfice pour la productivité des développeurs (Why You Need To Know About The Model Context Protocol (MCP)).
- **Medium & blogs tech** : De nombreux billets indépendants ont fleuri : citons *"MCP vs A2A: which protocol for AI agents?"* (B. Yang, Blott Studio) qui compare MCP à la proposition de Google (MCP vs A2A: Which Protocol Is Better For AI Agents? [2025] | Blott Studio) (MCP vs A2A: Which Protocol Is Better For AI Agents? [2025] | Blott Studio), ou encore le billet *"Agentic AI and the Model Context Protocol: A New Era"* (sur LinkedIn Pulse, 12 avril) qui rapporte l'annonce de Demis Hassabis sur l'adoption de MCP dans Google Gemini (Google

Adopts Anthropic's MCP: Pioneering AI Data Connectivity) (Google Adopts Anthropic's MCP: Pioneering AI Data Connectivity).

- **Vidéos et conférences** :
  - Une présentation remarquée est la vidéo *"MCP Explained in 20 minutes"* sur YouTube (Model Context Protocol (MCP) Explained in 20 Minutes - YouTube), qui fait un tour d'horizon pratique (on y voit la création d'un serveur MCP "Cursor" pour un IDE).
  - La conférence en ligne **"Agents Bar"** du 15 avril a consacré un épisode entier à MCP, avec des démos en live de Zed et Replit intégrant des serveurs (replay disponible sur YouTube).
  - Daniel Laskewitz (Power Platform Advocate) a publié une **démo vidéo** accompagnant le lab Copilot Studio (Microsoft Copilot Studio ❤️ MCP | Power Platform Developer Blog).
  - Côté événements, **Build 2025** (début mai) et **Google I/O 2025** ont tous deux mentionné MCP : chez Microsoft, une session sur l'extension de Copilot aux données locales via MCP, et chez Google, une annonce de la compatibilité prochaine de *Gemini* (modèle multi-modal de Google) avec MCP pour le branchement aux Google API (Google Adopts Anthropic's MCP: Pioneering AI Data Connectivity) (Demis Hassabis on X).
- **Communauté et veille** :
  - Le subreddit **r/mcp** et les GitHub Discussions de l'org modelcontextprotocol sont des lieux d'échange actifs (Roadmap - Model Context Protocol). On y trouve des *roundups* mensuels (notamment un *MCP April 2025 Roundup* épinglé résumant toutes les sorties majeures).
  - Des newsletters spécialisées IA, comme *Latent Space* ou *Pragmatic Engineer*, ont abordé MCP. Le Pragmatic Engineer du 30 avril mentionne par exemple *« Zed a ajouté le support serveur MCP – fun fact : ses fondateurs ont contribué au client MCP dans Zed »* (MCP Protocol: a new AI dev tools building block).
  - Enfin, le site communautaire **smithery.ai** sert de *hub* pour découvrir et tester des serveurs (avec recherche par catégorie, ex : "Web search", "CRM tools") (Smithery - Model Context Protocol Registry) (Smithery - Model Context Protocol Registry). C'est une ressource en constante évolution qui témoigne de la vitalité de l'écosystème MCP.

En résumé, la documentation et les ressources sur MCP sont abondantes et en grande partie ouvertes. Les dépôts GitHub officiels et les blogs des grands acteurs (Anthropic, GitHub, Microsoft, AWS) constituent la base incontournable. La communauté prend le relais avec des inventaires d'outils, des tutoriels et un partage d'expériences qui enrichissent la veille technologique autour de MCP.

Conseils pratiques et enseignements tirés

Au vu de cette veille, voici quelques **recommandations concrètes** pour les équipes souhaitant adopter MCP, en tirer profit en développement, tout en assurant sécurité et interopérabilité :

- **Commencer petit avec des cas d'usage ciblés** : Il est tentant de brancher un grand nombre d'outils MCP dès le départ, mais l'expérience montre qu'il vaut mieux ajouter progressivement les

serveurs pertinents pour un scénario donné. Identifiez une ou deux intégrations critiques (par ex. accès à votre base documentaire interne via MCP) et implémentez-les d'abord. Vous pourrez ensuite élargir la panoplie. Cette démarche incrémentale facilite le contrôle et la compréhension du comportement de l'agent IA.

- **Tirer parti des serveurs existants** : Avant de développer un serveur MCP custom, vérifiez si la communauté n'a pas déjà quelque chose d'approchant. L'**inventaire Smithery** ou l'**Awesome MCP** list (punkpeye/awesome-mcp-servers - GitHub) sont de bons points de départ. Réutiliser un composant éprouvé vous fait gagner du temps et assure une certaine robustesse (les bugs courants ont souvent été corrigés). Par exemple, plutôt que de coder de zéro un connecteur Trello, on peut utiliser un serveur "Generic REST API" en configurant l'API Trello dessus. Cependant, n'hésitez pas à forker un serveur existant pour l'adapter à vos besoins internes si nécessaire – la licence MIT de la plupart des exemples le permet.

- **Investir dans la sécurité dès la conception** : Intégrez dès le début les mécanismes d'authentification et de permissions. Si vous exposez un système interne via un serveur MCP, implémentez le **OAuth2 Device Code** ou un échange de token JWT pour ne fournir que des accès limités à l'agent. Côté client, configurez votre application pour exiger des confirmations utilisateur sur les actions sensibles. Comme le souligne GitHub, MCP doit être vu comme un *« port USB »* branché à votre IA : on n'y connecte pas une clé inconnue sans précaution (GitHub Announces Public Preview of GitHub MCP Server - InfoQ). Documentez en interne quelles classes d'outils MCP sont autorisées (liste blanche) et lesquels sont bannis ou restreints (par ex. pas d'outil "exécuter code arbitraire" en prod sans validation).

- **Utiliser les environnements de test et sandbox** : Avant de déployer un agent MCP en production, faites-le tourner sur des données factices ou en environnement isolé. Profitez des fonctionnalités de **"dry-run"** si disponibles (certains serveurs ont un mode simulation). Sur VS Code, testez l'Agent Mode dans un container éphémère. Cela vous permettra d'observer le comportement de l'agent quand il enchaîne des outils, sans risquer d'affecter de vraies ressources. De plus, monitorer l'agent dans un contexte fermé peut révéler des appels inattendus ou des boucles inutiles qu'on pourra corriger en ajustant les prompt templates ou en supprimant des outils redondants.

- **Surveiller les performances et l'expérience utilisateur** : Dans un workflow MCP, l'**expérience** est aussi importante que la fonction. Mesurez le temps de réponse de vos outils MCP et l'impact sur l'utilisateur final. Si une action via MCP prend trop de temps, envisagez d'optimiser le serveur (mise en cache, chargement paresseux des données, etc.) ou de repenser la manière dont l'agent utilise l'outil (peut-être faut-il d'abord filtrer la requête de l'utilisateur). N'hésitez pas à tirer parti du support du **streaming** : MCP permet d'envoyer des réponses partielles progressives, ce qui améliore la réactivité perçue pour l'utilisateur (par exemple, un serveur de recherche pourrait streamer les résultats trouvés un par un au lieu d'attendre d'avoir tout). Le streaming SSE a été amélioré dans la version récente du protocole (Key Changes - Model Context Protocol), profitez-en si vos outils s'y prêtent.

- **Participer à la communauté et rester à jour** : MCP évolue vite. Il est crucial de suivre les annonces (par ex. rejoignez la discussion *GitHub Discussions* (Roadmap - Model Context Protocol) ou le Discord Anthropic si disponible) pour anticiper les changements. Participez si possible aux **groupes de travail** ou aux forums : vous pourrez y partager vos retours (peut-être que la fonctionnalité que vous souhaitez est déjà en draft dans la prochaine version). La roadmap indique que la communauté sera impliquée dans la gouvernance du standard (Roadmap - Model Context Protocol) – c'est l'occasion de peser sur les orientations qui comptent pour vous (ex: support de nouveaux langages, standardisation de tel ou tel outil). De plus, en cas de découverte de faille ou de souci de sécurité, remontez-le rapidement sur les canaux officiels (une adresse security@modelcontextprotocol.io est fournie dans le dépôt (GitHub - microsoft/mcsmcp: Lab for creating an MCP Server and using it in Microsoft Copilot Studio.) (GitHub - microsoft/mcsmcp: Lab for creating an MCP Server and using it in Microsoft Copilot Studio.)). Être proactif vous assure aussi de bénéficier rapidement des patchs ou contournements proposés.

- **Interopérabilité et standards multiples** : Enfin, gardez en tête que MCP s'inscrit dans un écosystème plus large de normes émergentes pour agents IA. Suivre MCP ne signifie pas ignorer les autres : surveillez aussi des initiatives comme le protocole **A2A de Google** (agent-à-agent) ou l'**IETF agent://**. Il est probable que ces standards seront complémentaires. Par exemple, on pourrait voir un futur où un agent utilise MCP pour ses outils et A2A pour coordonner avec un autre agent d'une organisation partenaire – le tout de façon orchestrée. En anticipant cela, concevez vos agents de façon modulaire pour pouvoir brancher de nouveaux protocoles si besoin. **Conseil pratique** : structurez votre code d'agent en isolant la couche "intégration MCP" (ou autre) de la logique cœur de l'agent. Ainsi, si vous décidez d'adopter un autre protocole ou d'utiliser une surcouche (comme un orchestrateur multi-agents), vous n'aurez pas à réécrire toute votre application.

En appliquant ces conseils, une organisation pourra **adopter MCP de manière maîtrisée**. L'objectif est de maximiser les gains (automatisation, pertinence accrue grâce au contexte) tout en minimisant les risques (erreurs ou actions non désirées). Comme pour toute nouvelle technologie, l'accompagnement des utilisateurs et développeurs est crucial : formez vos équipes aux bonnes pratiques de prompt design pour guider l'agent, expliquez aux utilisateurs finaux les capacités et limites de l'IA outillée par MCP (afin qu'ils aient des attentes réalistes et continuent de superviser les résultats). MCP est un puissant levier – à nous de l'utiliser intelligemment.

Impacts attendus et perspectives

Les avancées du mois d'avril 2025 autour du Model Context Protocol laissent entrevoir des **impacts majeurs dans les mois à venir** sur la manière dont les agents IA s'intègrent aux outils et sur la productivité des développeurs :

- **Vers des agents réellement polyvalents** : MCP pourrait concrétiser la vision d'*« agents généraux » capables de naviguer entre de multiples systèmes. À mesure que l'adoption s'étend, on peut s'attendre à ce que les principaux environnements de travail des développeurs supportent nativement MCP. Après VS Code, on peut imaginer IntelliJ, Eclipse ou d'autres IDE intégrant des

clients MCP pour permettre aux IA d'interagir avec les bases de code, le suivi de tickets, les bases de données de l'entreprise, tout cela sans plugin propriétaire spécifique. Un agent IA pourrait ainsi *"voyager"* avec son utilisateur : commencer une tâche dans l'IDE, la poursuivre dans un outil de gestion de projet, puis dans un client mail – en gardant son contexte et en utilisant les connecteurs MCP appropriés à chaque étape. Cela changerait le paradigme : l'IA ne serait plus cantonnée à un silo (un chat dans telle application) mais deviendrait un **assistant transversal**, cohérent d'une application à l'autre.

- **Gain de productivité des développeurs** : En libérant l'IA des silos, MCP promet de réduire le temps perdu dans les changements d'outils. Par exemple, aujourd'hui un développeur interrompt souvent son flux pour aller chercher une information dans Jira ou dans la doc interne. Demain, avec MCP, il pourra poser la question directement à son assistant dans l'IDE, qui ira récupérer l'info via le serveur MCP adéquat (Jira, Confluence…) et la ramènera dans le contexte du code (Google Adopts Anthropic's MCP: Pioneering AI Data Connectivity) (Google Adopts Anthropic's MCP: Pioneering AI Data Connectivity). On anticipe ainsi un **décloisonnement de la connaissance** : la documentation, les tickets, les logs, les métriques… tout peut devenir accessible à la volée par langage naturel. Les études internes menées par des early adopters (annoncées lors de GitHub Universe fin 2024) montraient déjà une diminution du « temps de bascule » entre tâches. Sur les prochains mois, si ces gains se confirment à plus grande échelle, on pourrait voir une **amélioration significative de la vélocité** des équipes de développement, avec moins de contexte perdu entre les outils. Un développeur pourra se concentrer sur la logique métier pendant que son "copilote" enrichi via MCP gère les interactions secondaires.

- **Évolution des rôles et workflows** : L'introduction d'agents capables d'actions autonomes va probablement transformer certains workflows. Par exemple en **DevOps**, on commence à voir des agents (comme l'agent Code Review de GitHub Copilot, GA en avril (Vibe coding with GitHub Copilot: Agent mode and MCP support rolling out to all VS Code users - The GitHub Blog)) qui ouvrent des PR de correction de bugs ou qui ajustent la configuration CI/CD. MCP va amplifier cela en donnant accès aux outils d'infrastructure : on peut imaginer un agent SRE qui, recevant une alerte monitoring, utilise MCP pour aller chercher des données complémentaires (logs via un serveur ElasticSearch, métriques via un serveur Prometheus) puis propose directement un correctif sur une config Kubernetes. Ce type d'automatisation intelligente pourrait réduire le MTTR (temps de résolution des incidents). Cependant, il faudra cadrer ces agents "Ops" par des *policies* (ex: qu'ils ne puissent pas déployer sans approbation humaine). En somme, certains rôles verront leur nature changer : plus de **supervision/validation** de l'IA et moins de recherche manuelle.

- **Standardisation et interopérabilité industrielle** : Avec l'adoption annoncée par de grands acteurs, MCP a de fortes chances de s'imposer comme standard industriel pour la connexion des IA aux outils. Demis Hassabis (CEO de Google DeepMind) a publiquement soutenu MCP en avril en le qualifiant de *« bon protocole, en passe de devenir un standard ouvert de l'ère des agents IA »* (Demis Hassabis on X). Google a annoncé intégrer MCP dans son SDK pour les modèles *Gemini* (Google Adopts Anthropic's MCP: Pioneering AI Data Connectivity), et OpenAI aurait également indiqué lors d'une réunion développeurs fin mars son intention de rendre ChatGPT compatible

MCP pour accéder à des données entreprises de façon standard (Google Adopts Anthropic's MCP: Pioneering AI Data Connectivity). Si ces promesses se réalisent dans les prochains mois, cela signifie qu'indépendamment du fournisseur de modèle (OpenAI, Google, Anthropic, etc.), **une entreprise pourra brancher les mêmes serveurs MCP** à son agent IA. Ce serait un énorme progrès d'interopérabilité, là où jusque-là chaque écosystème proposait son format (plugins OpenAI, tools LangChain, etc.). On peut s'attendre à ce que des consortiums ou alliances se forment pour faire évoluer MCP vers une standardisation formelle (peut-être via l'ITU ou l'ISO si l'IETF ne se saisit pas du sujet). L'enjeu sera de préserver l'ouverture du protocole tout en assurant sa stabilité pour un usage critique.

- **Innovation et nouveaux usages** : En rendant l'accès aux capacités externalisées plus facile, MCP ouvre la voie à des **usages IA inédits**. Par exemple, l'essor des **IA agents autonomes** (*Auto-GPT* et consorts) pourrait bénéficier d'un protocole comme MCP pour accomplir des tâches complexes sur le web et les systèmes d'information. Plutôt que de *câbler* ces agents avec des hacks web-scraping, ils pourraient utiliser proprement des serveurs MCP "Browser" (Smithery - Model Context Protocol Registry) (Smithery - Model Context Protocol Registry), "Search", "Shopping" (imaginez un agent achetant des fournitures en ligne via MCP!). De même, dans le domaine de la **data science**, un agent scientifique pourrait utiliser MCP pour interroger des bases de données, lancer des jobs Spark, puis récupérer les résultats, le tout en langage naturel. Ces scénarios deviennent plus réalistes à mesure que le catalogue d'outils MCP s'étoffe. On voit déjà apparaître des *MCP servers* pour des tâches spécialisées (ex : un serveur "ThinkTool" pour aider l'agent à structurer un raisonnement complexe étape par étape (Smithery - Model Context Protocol Registry)). Cela préfigure des agents plus intelligents, capables de s'auto-améliorer en convoquant des outils d'analyse ou de calcul pendant qu'ils réfléchissent.

- **Défis à venir** : Bien sûr, quelques défis accompagneront ces impacts. La **charge cognitive** pour l'utilisateur de faire confiance à une IA aux longs tentacules en est un : il faudra que les interfaces utilisateur évoluent pour donner de la **visibilité sur le contexte** que l'IA utilise. Microsoft planche déjà sur des *UX* montrant à l'utilisateur quelles données l'IA a consultées via MCP durant une session, afin d'expliquer ses réponses (ex: "Copilot a utilisé vos agendas et emails pour proposer ce planning"). La **cohabitation de multiples protocoles** (MCP, A2A, etc.) devra aussi être harmonisée, peut-être via des passerelles ou des protocoles unifiés plus tard. Enfin, la **montée en charge** sera un enjeu si des milliers d'agents utilisent MCP simultanément : des solutions de cache, de load-balancing pour les MCP servers populaires devront être mises en place (on peut imaginer un *MCP Registry* faisant office de CDN pour certains outils publics).

En conclusion, l'impression générale qui se dégage en avril 2025 est que MCP est bien parti pour **transformer l'écosystème des agents conversationnels et assistants de développement**. Les mois à venir verront sans doute une consolidation : plus d'outils, plus de retours terrain, et possiblement les premières versions stables (la spec pourrait passer en version 1.0 finalisée d'ici la fin 2025). Pour les développeurs et organisations, il s'agit d'une opportunité de gagner en efficacité et en intelligence outillée, à condition d'embrasser ce changement de paradigme. MCP dessine un futur où les **IA sont pleinement connectées** – un futur qui, de l'avis de beaucoup en cette fin avril, commence à se réaliser

dès maintenant (GitHub Announces Public Preview of GitHub MCP Server - InfoQ) (GitHub Announces Public Preview of GitHub MCP Server - InfoQ).