Checklist gratuite tirée d'Agents en production

Checklist de mise en production d'un agent

59 contrôles testables pour faire passer un agent IA du pilote à la production, dont les 19 à valider avant le canary

Écrit par des agents IA (Claude), sous la direction de Yann

Édition 2026 à 2027

Comment utiliser cette checklist

La plupart des projets d'agents n'échouent pas à la démo. Ils échouent dans les semaines qui suivent : une boucle que personne n'a plafonnée, un outil qui renvoie « aucun résultat » quand le backend a expiré, une facture que personne n'a modélisée, un connecteur qui a plus de droits que la personne qui l'a installé.

Cette checklist est le point de contrôle que tout agent devrait franchir avant de toucher de vrais utilisateurs. Elle est organisée dans l'ordre où les problèmes ont tendance à apparaître, du périmètre au *rollout* (déploiement progressif).

- **Avant de construire :** passez la section périmètre et conception. Si vous ne pouvez pas la cocher, il vous faut probablement un *workflow* (enchaînement d'étapes prédéfinies dans le code), pas un agent.
- **Avant le déploiement *canary* (une petite part du trafic réel passe d'abord sur la nouvelle version) :** chaque point marqué **(obligatoire)** devrait être coché, avec une preuve (un test, un tableau de bord, un document) plutôt qu'une promesse.
- **Chaque trimestre :** repassez toute la liste. Les modèles, les frameworks et les serveurs MCP changent sous vos pieds.

Chaque point tient en une ligne et se teste. Si un point ne s'applique pas, notez pourquoi. Cette note vaut plus que la case cochée.

Cette checklist est l'annexe A d'**Agents en production**, le guide pratique des frameworks agentiques, de MCP, des skills et de l'automatisation par l'IA (édition 2026 à 2027).

Écrit par des agents IA (Claude, d'Anthropic) sous la direction de Yann. Chaque contrôle renvoie à un chapitre sourcé du livre.

A

Checklist de mise en production d'un agent

Cinquante-neuf contrôles à passer avant qu'un agent reçoive du trafic réel, puis chaque trimestre

Tech Manager Solo No-code Claude

Une démo prouve qu'un agent fonctionne une fois ; la production s'intéresse à la millième exécution, à la mise à niveau du modèle, à la page piégée et au responsable parti. Chaque point est un oui ou un non que vous pouvez prouver.

Comment utiliser cette checklist

Passez-la en revue de conception (périmètre, outils, sécurité), en entier avant le déploiement *canary* (une petite part du trafic réel passe d'abord sur la nouvelle version), puis chaque trimestre, et repassez les sections concernées après un changement de modèle, d'outil, de données ou d'utilisateurs, ou après un incident. Liez une preuve versionnée à chaque point coché ; une case cochée de mémoire vaut non. Sur les plateformes no-code, demandez par écrit à l'éditeur ce que vous ne pouvez pas voir, et utilisez le tableau « le harness sur un canevas » du chapitre 13 (le *harness* est le code qui encadre la boucle du modèle) : la fiche de test valide les points sur les *evals* (évaluations automatisées : jeux de tâches avec résultat attendu et notation) ; les exécutions enregistrées avec identifiant d'exécution, version du prompt et modèle valident le point sur les traces ; la protection contre le rejeu valide le point sur l'idempotence ; et un filtre qui oriente une équipe ou une part des déclencheurs vers la nouvelle version du flux, en conservant l'ancienne, valide le point sur le canary.



> **La préparation se prouve, elle ne se décrète pas.** Passez la checklist trois fois ; les points (obligatoire) conditionnent le canary, et une couverture de 90 % conditionne le rollout complet.

Règle de décision **Notation**

Le canary exige que chaque point **(obligatoire)** soit coché ; le *rollout* (déploiement progressif) complet exige 90 % de tous les points, ceux restés ouverts ayant un responsable et une date. Tout point **(obligatoire)** ouvert en sécurité ou en fiabilité est un arrêt. **Non applicable** exige une raison d'une ligne ; un point (obligatoire) n'est non applicable que si sa condition préalable est absente (pas d'outils d'écriture, pas de récupération). « Trop tôt » n'est jamais une raison.

1. Périmètre et conception (chapitres 1, 2, 14)

- [ ] **(obligatoire)** La tâche tient en un paragraphe, avec des exemples dans et hors périmètre.
- [ ] Un *workflow* (enchaînement d'étapes prédéfinies dans le code) a été écarté avant de choisir une boucle d'agent.
- [ ] **(obligatoire)** Le succès est un événement observable dans les traces (par exemple, « brouillon envoyé sans modification »).

- [ ] Le niveau d'autonomie est précisé : nombre maximal d'étapes avant un humain, et actions qui exigent une approbation.
- [ ] La valeur et le coût complet par succès sont estimés dans une feuille de calcul du retour sur investissement.
- [ ] **(obligatoire)** Un responsable de l'agent et un pilote de processus nommés ont validé le périmètre.

2. Outils et MCP (chapitres 3, 4, annexe B en ligne)

- [ ] Chaque outil a un nom avec espace de noms, une description destinée au modèle et des paramètres sans ambiguïté.
- [ ] Les outils renvoient une sortie à fort signal, avec pagination et une taille plafonnée par défaut.
- [ ] Les erreurs d'outil indiquent au modèle comment corriger l'appel ; les erreurs de validation reviennent comme erreurs d'outil, pas comme plantages.
- [ ] **(obligatoire)** Chaque outil qui modifie un état est idempotent ou accepte une clé d'idempotence.
- [ ] **(obligatoire)** Chaque outil est marqué lecture seule ou écriture, et les écritures exigent une approbation là où le périmètre le prévoit.
- [ ] La configuration MCP est gérée de façon centralisée et en lecture seule depuis l'environnement de l'agent.
- [ ] Chaque serveur MCP tiers est figé sur une version, et ses descriptions d'outils sont comparées (diff) à chaque mise à jour.
- [ ] Les serveurs MCP distants utilisent HTTPS et OAuth avec des *tokens* d'accès (identifiants délivrés par OAuth) à portée limitée et liés à leur audience (RFC 8707, exigée par la spécification d'autorisation MCP), pas des secrets partagés.
- [ ] L'agent ne voit que les outils nécessaires à cette tâche (liste d'autorisation, *allowlist*, par agent).

3. Contexte et données (chapitre 5)

- [ ] Le prompt système, la liste d'outils et les instructions stables viennent en premier, dans un ordre fixe, pour que la mise en cache fonctionne.
- [ ] **(obligatoire)** La récupération respecte les droits de l'utilisateur final : l'agent ne peut pas lire ce que l'utilisateur ne peut pas lire.
- [ ] Les documents récupérés portent une source et une date de fraîcheur ; les documents périmés sont signalés ou exclus.
- [ ] Les longues sessions utilisent la compaction ou des notes, et les contraintes clés sont réinjectées après compaction.
- [ ] Les réponses factuelles citent leurs sources, et un test vérifie que les citations renvoient à un texte réellement récupéré.
- [ ] Les données personnelles envoyées au modèle sont minimisées et leurs catégories recensées.

4. Fiabilité (chapitre 8)

- [ ] **(obligatoire)** La boucle a un plafond d'itérations strict et un budget de *tokens* (unités de texte facturées par le modèle) ou de coût par exécution.
- [ ] **(obligatoire)** Les appels au modèle et aux outils ont des *timeouts* (délais d'expiration) et des nouvelles tentatives en nombre limité, avec temporisation croissante.
- [ ] Les exécutions longues ou qui touchent à de l'argent tournent dans une couche de workflow durable qui survit à un crash.
- [ ] Un chemin de *fallback* (solution de repli : modèle plus petit, workflow fixe ou humain) existe en cas de défaillance de l'agent ou du fournisseur.
- [ ] **(obligatoire)** Un *kill switch* (coupe-circuit qui arrête l'agent immédiatement) arrête l'agent, ou ses outils d'écriture, en quelques minutes et sans déploiement.
- [ ] Les approbations humaines sont attendues de façon asynchrone, sans bloquer un exécutant ni perdre l'état.

5. Evals (chapitre 9)

- [ ] **(obligatoire)** Un jeu d'évaluation hors ligne de cas réels étiquetés existe, sous la responsabilité du pilote de processus.
- [ ] **(obligatoire)** Les evals tournent en CI à chaque changement de prompt, d'outil, de modèle ou de framework, avec un point de contrôle de régression.
- [ ] Les métriques d'eval incluent la réussite de la tâche, le coût modèle par succès, les appels d'outils et les escalades.
- [ ] Les correcteurs, les corrigés et les *holdouts* (jeux de test réservés) sont hors de portée de l'agent ; les evals utilisent le harness de production.
- [ ] Tout LLM juge (*LLM-as-judge*) est calibré sur des étiquettes humaines, et la version du modèle juge est figée.
- [ ] Des cas adverses (injection de prompt ou *prompt injection*, sortie d'outil malformée, données manquantes) figurent dans le jeu d'évaluation.
- [ ] Les evals en ligne notent un échantillon de traces de production ; les échecs rejoignent le jeu hors ligne.

6. Observabilité et coût (chapitre 10)

- [ ] **(obligatoire)** Chaque exécution produit une trace avec des *spans* (segments d'une trace) pour chaque appel au modèle et chaque appel d'outil.
- [ ] Les spans suivent les conventions sémantiques GenAI d'OpenTelemetry, avec une version des conventions sémantiques figée.
- [ ] L'usage de tokens, les lectures du cache et le coût par exécution sont enregistrés et agrégés par agent.
- [ ] Un tableau de bord montre le taux de réussite, le coût modèle par succès, la latence, les erreurs d'outils et les escalades.
- [ ] **(obligatoire)** Les dépenses sont plafonnées en dehors de l'agent (passerelle, compte ou limite du fournisseur) ; les clés facturées à l'usage ne sont pas lisibles depuis l'environnement de l'agent ; les alertes se déclenchent avant le plafond.

- [ ] Les journaux et les traces masquent ou pseudonymisent les données personnelles, avec une durée de conservation documentée.

7. Sécurité (chapitre 11, annexe C en ligne)

- [ ] **(obligatoire)** Une modélisation des menaces existe et couvre l'injection de prompt directe et indirecte.
- [ ] **(obligatoire)** Aucune session ne combine données privées, contenu non fiable et communication externe sans approbation humaine ou conception séparée.
- [ ] **(obligatoire)** Les identifiants suivent le moindre privilège par outil ; pas de portées génériques, pas de transmission directe de tokens d'accès.
- [ ] L'exécution de code et les serveurs MCP locaux tournent dans un bac à sable (*sandbox*) avec des listes d'autorisation pour le système de fichiers et le réseau.
- [ ] Le trafic sortant est limité à des destinations connues via un proxy ou une liste d'autorisation.
- [ ] **(obligatoire)** Aucun secret n'apparaît dans les prompts, les fichiers de *skills* (dossiers avec un `SKILL.md`, chargés à la demande) ni en clair dans la configuration MCP.
- [ ] Les *hooks* (fonctions déclenchées à un moment du cycle de vie de l'agent) de politique ou les garde-fous bloquent en cas d'échec ou d'expiration.
- [ ] Une suite de tests d'injection tourne en CI contre les outils de l'agent et ses sources de récupération.

8. Gouvernance (chapitre 12)

- [ ] **(obligatoire)** L'agent figure dans un registre avec responsable de l'agent, pilote de processus, objectif, outils, catégories de données, modèle et circuit d'astreinte.
- [ ] Les changements de prompt, d'outil, de modèle, d'effort et de limites sont versionnés comme un seul *bundle* et relus comme du code.

- [ ] Les journaux d'audit enregistrent qui a approuvé quelle action, et sont conservés selon la politique en vigueur.
- [ ] Les utilisateurs sont informés qu'ils interagissent avec un système d'IA, sauf si c'est évident (article 50 du règlement européen sur l'IA, applicable depuis le 2 août 2026).
- [ ] Le cas d'usage a été comparé aux domaines à haut risque de l'annexe III du règlement européen sur l'IA (les règles à haut risque s'appliquent à partir du 2 décembre 2027).
- [ ] Chaque fournisseur de modèle et d'outil a passé une revue fournisseur sur le traitement et la localisation des données.

9. Rollout (chapitres 8, 14, 16)

- [ ] L'agent a tourné en *shadow mode* (l'agent tourne sur des cas réels pendant que les humains font encore le travail) sur des entrées réelles, comparées aux résultats humains.
- [ ] **(obligatoire)** Le rollout utilise un *feature flag* (interrupteur de fonctionnalité) et une part de trafic en canary, avec un *rollback* (retour arrière vers la version précédente) documenté.
- [ ] Les critères de passage et d'arrêt de chaque étape sont écrits avant qu'elle commence.
- [ ] L'équipe concernée a été formée à approuver, corriger et escalader le travail de l'agent.
- [ ] Une date de revue trimestrielle est fixée pour revérifier le coût, la qualité, l'adoption et la responsabilité.

L'annexe A en une page

Checklist de mise en production d'un agent

Chaque contrôle est un oui ou un non que vous pouvez prouver ; si vous ne pouvez pas lier la preuve, c'est un non.

À retenir

- 59 contrôles en neuf domaines, dont 19 marqués (obligatoire)
- Tous les points (obligatoire) avant le canary ; 90 % de tous les points avant le rollout complet
- Tout point (obligatoire) ouvert en sécurité ou en fiabilité est un arrêt

À faire dès lundi

- Passez les sections périmètre et sécurité sur votre prochain agent
- Demandez à chaque responsable d'agent le lien de preuve derrière trois points cochés tirés au hasard
- Évaluez cette semaine votre agent en production le plus risqué au regard des points (obligatoire)

59 contrôles

19 points (obligatoire)

Pour aller plus loin : le livre complet

Ceci était un extrait gratuit d'**Agents en production**, un guide pratique pour faire passer des agents du pilote à la production et les y maintenir. Le livre complet comprend un prologue, 21 chapitres en neuf parties, une postface sur la façon dont le livre a été fait et la checklist de mise en production, plus un cahier d'exercices compagnon et des compléments gratuits en ligne.

Le livre complet couvre :

- **Modèles mentaux :** ce qu'est vraiment un agent, et quand un simple *workflow* (enchaînement d'étapes prédéfinies dans le code) vaut mieux.

- **Briques de base :** des outils que les agents peuvent vraiment utiliser, MCP après la révision 2026-07-28, les *skills* (dossiers avec un `SKILL.md`, chargés à la demande), les sous-agents et les plugins, l'ingénierie du contexte et la mémoire.
- **Frameworks et architecture :** une carte indépendante des éditeurs du paysage des frameworks en 2026, un guide de décision et des architectures de référence pour les agents pilotés par les événements et durables.
- **Production :** des *harnesses* (le code qui encadre la boucle du modèle) de fiabilité, des *evals* (évaluations automatisées) en CI, l'observabilité et l'économie unitaire, la sécurité des agents et de MCP, la gouvernance et le règlement européen sur l'IA.
- **Business :** des recettes d'automatisation no-code, des guides pratiques pour indépendants qui font gagner ou économiser de l'argent, et un plan construire, acheter et recruter pour les managers.
- **L'usine logicielle agentique :** des agents qui mènent le logiciel de l'idée à la *pull request* (demande de fusion) fusionnée, le testent de bout en bout et continuent de l'améliorer, avec un projet de synthèse qui fait passer un système par chaque poste.
- **La suite :** un atlas des cas d'usage, les preuves sur la productivité, l'emploi, le coût et l'énergie, et des prévisions sourcées.
- **Annexe :** la checklist de mise en production (59 contrôles, dont 19 obligatoires). En ligne et gratuits : une checklist de conception des outils et serveurs MCP, un modèle de modélisation des menaces, les notes de terrain 2026, un glossaire et les consignes de chaque agent qui a écrit le livre. Le cahier d'exercices compagnon contient les exercices et leurs corrigés.

Quatre parcours de lecture (Tech, Manager, Solo, No-code) vous disent exactement quels chapitres lire et lesquels sauter. Les exercices, le projet fil rouge et les autoévaluations sont fournis avec leurs corrigés complets.

Écrit par des agents IA (Claude, un modèle conçu par Anthropic) sous la direction de Yann.

Disponible sur {{STORE_URL}}