APOLLO
CODING LIFE, LIVING CODE

LIVRE BLANC · ÉDITION 2026

RUN APPLICATIF · IA GÉNÉRATIVE · APOLLO CODING LIFE

Run applicatif et IA générative 5 cas d'usage chez Apollo.

Comment intégrer l'IA dans le maintien en condition opérationnelle de vos applications métier — sans perdre la maîtrise.

Un livre blanc destiné aux **DSI** qui veulent piloter le Run par la valeur, et aux **responsables Run et architectes** qui veulent intégrer l'IA dans leurs workflows quotidiens sans naïveté ni dogmatisme.

Apollo Coding Life · Paris · Lyon · Grenoble

v1.0 — Mai 2026

contact@apollossc.com

SOMMAIRE

Ce que vous allez lire

Un parcours en cinq parties et quatre annexes, conçu pour être lu in extenso ou par chapitre selon votre rôle. Comptez 25 à 30 minutes en lecture complète.

POUR LE DSI

Lecture rapide : avant-propos, parties 1, 5, 6 et annexes A & D. 12 minutes.

POUR LE RESPONSABLE RUN

Lecture complète recommandée. Focus particulier sur les parties 3, 4 et 5. 25 minutes.

AVANT-PROPOS

Le Run n'est pas le parent pauvre. C'est le terrain le plus exigeant.

Quand une conversation sur l'IA générative dans l'ingénierie logicielle s'engage, elle se concentre presque toujours sur la phase de Build. C'est compréhensible : un développeur qui code plus vite, c'est visible, mesurable, vendable. L'IA y trouve un terrain de démonstration immédiat.

Mais 80 % du coût total d'une application se joue après sa mise en production. Et c'est précisément dans cette phase — le Run — que la pression est la plus forte : disponibilité 24/7, dette technique qui s'accumule, équipes parfois changeantes, documentation périmée, budget contraint, satisfaction métier mesurée.

Le Run, c'est l'endroit où l'on récolte les conséquences des décisions prises ailleurs. C'est aussi l'endroit où une IA bien encadrée peut produire la plus grande valeur — à condition de ne jamais oublier qu'un Run mal maîtrisé devient une dette dont vos utilisateurs paieront le prix.

Chez Apollo, l'IA n'est pas un produit qu'on vend. C'est une discipline qu'on encadre.

Ce livre blanc présente comment nous intégrons concrètement l'IA — et notamment Claude Code — dans nos missions de Run. Cinq cas d'usage, sept garde-fous, six indicateurs de valeur, une grille de maturité. Pas de promesse magique. Pas de chiffrage de gains de productivité à 70 % qu'aucun retour terrain ne corrobore. Juste ce que nous observons réellement, ce que nous refusons de faire, et ce que nous engageons sur la durée.

Bonne lecture.

L'équipe Apollo Coding Life
Lyon · Paris · Grenoble — Mai 2026

PARTIE 1

01 Pourquoi parler d'IA dans le Run ?

1.1 — Le Run, parent pauvre de la conversation IA

L'écosystème de l'IA générative s'est construit autour du développeur en phase de construction. GitHub Copilot, Cursor, Claude Code, Codeium : tous ces outils visent la productivité de l'écriture de code. Naturel : c'est là que la promesse est la plus visible et la plus facile à démontrer.

Pourtant, la moitié des consultants ESN n'écrit pas du code neuf : ils maintiennent, corrigent, font évoluer, documentent et exploitent l'existant. Le Run mobilise probablement plus de jours-hommes en France que le Build, mais reste très peu adressé par les outils IA actuels.

POUR LE DSI

Si votre stratégie IA se concentre sur "augmenter la vélocité des projets", vous laissez sur la table 60 à 80 % de votre budget applicatif. Le levier ROI le plus rapide est souvent le Run, pas le Build.

1.2 — Trois idées reçues à déconstruire

« L'IA va remplacer les équipes de Run »

Non. L'IA accélère certaines tâches du Run — diagnostic, génération de tests, documentation — mais elle ne porte ni la responsabilité contractuelle, ni la connaissance produit, ni la relation client. Un Run sans humain est un Run sans engagement.

Ce que l'on observe sur le terrain, c'est plutôt une **redistribution du temps** : moins de tâches répétitives, plus de temps consacré à la compréhension fine du métier et à la réduction proactive de la dette.

« L'IA va décider à notre place »

Non plus. Dans tous nos cas d'usage Run, l'IA *propose* ; un humain *arbitre*. Aucune action de production n'est exécutée sans validation explicite d'un consultant senior. C'est une règle absolue chez Apollo, et c'est aussi la condition pour rester en conformité avec l'EU AI Act dans la plupart des contextes métiers.

« On va perdre le contrôle de notre code »

Cette inquiétude est légitime — et elle se traite techniquement. Politique de sandboxing, MCP locaux pour limiter le contexte exposé, chiffrement des secrets, désactivation du fine-tuning sur vos données, audit log complet. La question n'est pas "IA ou pas IA" mais "avec quelle gouvernance".

POUR L'OPÉRATIONNEL

Avant d'introduire l'IA dans un workflow Run, posez-vous trois questions : (1) qu'est-ce qui est exposé au modèle ? (2) qui valide la sortie ? (3) que se passe-t-il si la sortie est fausse ?

1.3 — Le bon niveau de promesse

Apollo ne promet pas que l'IA va diviser par 10 le coût de votre Run. Nous observons, sur nos missions, des gains plus mesurés mais plus solides :

- Diagnostic d'incident P1 — temps moyen **réduit de 40 à 60 %** sur les périmètres où l'IA est déployée avec un bon corpus de logs et d'historique.
- Couverture de tests de non-régression — passage typique de **15-25 % à 55-70 %** sur 3 à 6 mois.
- Temps d'onboarding d'un nouveau consultant sur une application reprise — **réduit de moitié** lorsque la documentation a été refondue avec assistance IA.

Ces gains sont mesurés et tracés. Ils ne sont pas universels : ils dépendent fortement de la qualité du contexte fourni à l'IA, de la maturité de l'observabilité, et de l'expertise des humains qui valident.

L'IA dans le Run, ce n'est pas une révolution. C'est une discipline d'industrialisation supplémentaire — qui ne fonctionne que si les fondamentaux sont déjà solides.

PARTIE 2

02 Le cadre Apollo : IA augmentée, contrôle humain.

Notre cadre d'usage de l'IA dans le Run repose sur quatre principes simples, énoncés dès la phase contractuelle et appliqués sans exception.

2.1 — Les quatre principes fondateurs

1

Revue humaine systématique

Aucune sortie IA n'est appliquée en production sans validation explicite par un développeur senior ou un tech lead. Cette règle est inscrite dans nos contrats. Elle vaut pour le code généré, les recommandations de remédiation, les suggestions de refactoring et la documentation.

2

Traçabilité totale

Chaque usage de l'IA dans un livrable est tracé : prompt utilisé, modèle invoqué, sortie brute, modifications humaines apportées. Cet audit log est mis à disposition du client à la demande, et systématiquement remis lors d'une réversibilité.

3

Aucune décision IA non auditée

L'IA peut proposer, classifier, prioriser, suggérer. Elle ne peut pas, dans nos workflows Run, déclencher seule une action de production : restart, déploiement, rollback, fermeture de ticket critique. Le geste reste humain. Toujours.

4

Sécurisation du contexte

Avant d'exposer un corpus à un modèle, nous appliquons une politique de filtrage : secrets purgés, PII masquées, données client compartimentées. L'usage de MCP (Model Context Protocol) locaux nous permet de limiter au strict nécessaire ce qui sort de votre périmètre.

2.2 — Le rôle de Claude Code dans le workflow Run

Apollo a choisi Claude Code comme outil principal d'intégration de l'IA dans ses workflows Run, pour trois raisons :

- **Maîtrise du contexte.** Claude Code est conçu pour opérer dans un répertoire de travail délimité, avec un contrôle fin sur ce qui est lu et ce qui est partagé avec le modèle.
- **Capacité d'orchestration.** L'outil sait enchaîner des actions de lecture, d'analyse, d'écriture et de test — utile pour les workflows de diagnostic et de refactoring.
- **Auditabilité.** Chaque session laisse une trace exploitable, indispensable dans un cadre contractuel de Run.

Cela ne signifie pas que nous écartons les autres outils : pour certains usages (chat de support, classification), des modèles plus légers ou des services métiers font tout aussi bien. Le choix est dicté par le cas d'usage, pas par l'outil.

2.3 — Le workflow type "diagnostic incident assisté IA"

1. **Détection.** L'incident est levé par la supervision (Datadog, Dynatrace, ELK ou équivalent).
2. **Préparation du contexte.** Le consultant Run collecte les logs pertinents, les métriques d'observabilité et l'historique récent du périmètre.
3. **Invocation IA.** Claude Code reçoit ce contexte avec un prompt cadré : "analyse ces logs, propose trois hypothèses de cause racine classées par probabilité, et indique pour chacune les vérifications à effectuer".
4. **Validation senior.** Le consultant senior arbitre, écarte les hypothèses peu plausibles, choisit la stratégie de vérification.
5. **Action humaine.** Le geste de remédiation reste humain. L'IA peut éventuellement aider à rédiger le commit ou le post-mortem.
6. **Capitalisation.** Le ticket, le diagnostic et la résolution sont indexés pour enrichir le contexte des prochains incidents.

POUR LE DSI

Ce workflow ne nécessite pas de transformation profonde de votre SI. Il s'insère dans des outils que vos équipes utilisent déjà : ITSM, supervision, dépôts de code. L'investissement principal est *méthodologique*, pas technique.

PARTIE 3

03 Cinq cas d'usage Apollo.

Cinq situations concrètes rencontrées par nos consultants Run, dans lesquelles l'IA générative apporte une valeur mesurable — sous garde-fous Apollo. Les chiffres présentés correspondent à ce que nous observons typiquement sur nos missions, pas à des moyennes universelles.

CAS D'USAGE N°1

Diagnostic d'incident accéléré

CONTEXTE

Une application métier critique (back-office logistique, ~150 000 transactions/jour) lève une alerte P1 le mardi 14h : taux d'erreur API en pic, lent et progressif. Cinq composants en jeu, des dépendances externes, et un historique de releases sur les dix derniers jours qui complique l'analyse.

AVANT IA

Un développeur senior épluche manuellement les logs des 5 services concernés. Croisement avec les métriques. Vérification des releases récentes. Hypothèses construites au fil de l'eau, sans formalisme.

Durée typique : 4 à 6 heures avant identification fiable de la cause racine.

AVEC ASSISTANCE IA

Claude Code reçoit les logs corrélés, les métriques, le diff des dernières releases et l'historique des incidents passés sur ce périmètre. Il produit trois hypothèses de cause racine classées par probabilité, chacune assortie d'une stratégie de vérification.

Le consultant senior arbitre, écarte deux hypothèses, vérifie la troisième. Cause identifiée.

RÉSULTATS OBSERVÉS

- MTTR P1 sur ce type d'incident **réduit de 4-6h à 1h30-2h** en moyenne.
- Qualité des post-mortems améliorée : plus de contexte capté, moins de "trous" dans la timeline.

- Effet d'apprentissage : le corpus de diagnostics indexé enrichit les analyses suivantes.

GARDE-FOUS

- L'IA ne déclenche jamais d'action corrective elle-même (redémarrage, rollback, scaling).
- Les logs sont préalablement filtrés pour exclure les secrets et données personnelles non strictement nécessaires.
- Le tech lead valide la stratégie de remédiation avant exécution.

POUR LE DSI

Gain SLA mesurable et reportable. Un MTTR P1 divisé par 2 sur un parc critique se traduit directement en disponibilité client et en pénalités évitées. Le ROI est typiquement atteint en moins de 6 mois pour les parcs à forte criticité.

POUR L'OPÉRATIONNEL

Prompt type : "Voici les logs des services A, B, C entre 13h40 et 14h10. Voici la timeline des derniers déploiements. Identifie trois hypothèses de cause racine, classe-les par probabilité, et propose une stratégie de vérification pour chacune. Ne propose aucune action en production."

CAS D'USAGE N°2

Génération de tests de non-régression

CONTEXTE

Une application Java de 12 ans, périmètre métier facturation, ~280 000 lignes de code. Couverture de tests : 14 %. Chaque évolution génère une peur de régression. Le Build embarqué dans le Run ralentit, le coût de chaque correctif augmente.

AVANT IA

Le client refuse les "projets de test rétroactifs", trop chers et sans valeur métier visible. La couverture stagne. Les régressions se découvrent en production, ce qui dégrade la satisfaction métier et alourdit le Run.

AVEC ASSISTANCE IA

Sur chaque ticket d'évolution ou de correctif, Claude Code analyse la zone de code modifiée, le comportement observé via les logs récents, et propose une batterie de tests unitaires et d'intégration. Le développeur senior revoit, ajuste, complète.

RÉSULTATS OBSERVÉS

- Couverture passée de **14 % à 62 %** en 6 mois, sans projet de test dédié.
- Régressions détectées en pré-production multipliées par 3, régressions atteignant la production divisées par 4.
- Coût marginal négligeable : les tests sont générés "au fil de l'eau" des évolutions, pas dans un projet à part.

GARDE-FOUS

- Tous les tests sont revus, exécutés et stabilisés par un développeur senior avant intégration au pipeline.
- Tests "flaky" rejetés systématiquement — un faux positif fait plus de dégâts qu'une lacune assumée.
- La génération couvre le comportement attendu, pas la spécification métier — qui reste à valider avec le référent.

POUR LE DSI

Action structurelle sur la dette : la couverture de tests est l'un des indicateurs de patrimoine les plus fiables. Sa progression continue, financée par le forfait Run, transforme votre application legacy en application gérable sans dépendance d'expert unique.

POUR L'OPÉRATIONNEL

Bonne pratique : adosser cette génération à chaque pull request via un hook CI. Le développeur reçoit une suggestion de tests à revoir avant merge. La friction est minimale, l'effet cumulé sur 6 mois est très significatif.

CAS D'USAGE N°3

Refactoring guidé de la dette technique

CONTEXTE

Application .NET d'une ETI industrielle, ~95 000 lignes. Score SonarQube en dégradation continue depuis 2 ans. Code dupliqué, méthodes de plus de 200 lignes, couplage fort entre couches. La dette est connue mais "non priorisable" dans le contexte business.

AVANT IA

Le refactoring "big-bang" est repoussé indéfiniment. Les développeurs corrigent localement, sans toucher à la structure. La dette continue de s'accumuler.

AVEC ASSISTANCE IA

À chaque ticket touchant une zone "dette", Claude Code propose un refactoring ciblé et limité (extraction de méthode, suppression de duplication, simplification de conditionnels). Le tech lead arbitre la portée. Les tests générés (cf. cas 2) protègent l'opération.

RÉSULTATS OBSERVÉS

- Score SonarQube en amélioration continue (~30 % de dette mesurée en moins sur 6 mois).
- Aucune régression majeure attribuable au refactoring sur la période.
- Effet bonus : les développeurs deviennent plus exigeants sur la qualité, l'IA jouant un rôle de "veille permanente".

GARDE-FOUS

- **Jamais de big-bang.** Les refactorings sont locaux, incrémentaux, et toujours adossés à un ticket fonctionnel.
- Batterie de tests obligatoire avant tout refactoring (cf. cas 2).
- Une PR de refactoring "pur" ne peut pas dépasser un seuil de complexité défini (ex. 200 lignes touchées).
- Le client est informé via le tableau de pilotage mensuel : transparence sur la dette traitée.

POUR LE DSI

C'est ici que l'on transforme une **dette accumulée** en **dette pilotée**. La différence n'est pas comptable : c'est une différence de stratégie. Une dette pilotée se mesure, se prévoit, se finance — comme une charge récurrente de maintenance immobilière.

POUR L'OPÉRATIONNEL

Convention Apollo : 15 % du forfait Run est consacré à des actions structurantes (réduction de dette, génération de tests, doc). Ce pourcentage est non négociable côté budget mais arbitré côté périmètre avec le client.

CAS D'USAGE N°4

Documentation vivante du patrimoine

CONTEXTE

Reprise par Apollo d'un portefeuille de 7 applications historiquement opérées par un autre prestataire. Documentation présente mais largement obsolète, runbooks d'astreinte incomplets, schémas d'architecture datant de 4 ans. Risque opérationnel élevé pendant la phase de transition.

AVANT IA

Reconstitution manuelle, longue, fastidieuse. Onboarding de chaque consultant : 3 à 4 semaines. Documentation produite "à la fin", rarement maintenue ensuite.

AVEC ASSISTANCE IA

Claude Code analyse le code, les configurations, les workflows CI/CD, l'historique des incidents et des conversations Slack/Teams (sur consentement et purge). Génère des projets de docs : ADR rétroactifs, schémas d'archi, runbooks d'astreinte, glossaire métier. Le SDM et le tech lead révisent.

RÉSULTATS OBSERVÉS

- Temps d'onboarding consultant **réduit de 4 semaines à 2 semaines** en moyenne.
- Time-to-fix sur les incidents nocturnes amélioré (runbooks plus fiables).
- La documentation reste à jour : à chaque release significative, l'IA propose les évolutions de docs associées.

GARDE-FOUS

- Toute documentation générée est revue humainement avant publication. La source de vérité reste le code, jamais la doc.
- Les conversations confidentielles sont purgées du corpus d'apprentissage.
- La documentation est versionnée dans le dépôt code, pas dans un outil externe susceptible de divergence.

POUR LE DSI

Réduction du risque de dépendance fournisseur. Un patrimoine bien documenté est un patrimoine que vous pouvez confier — ou reprendre — à coût raisonnable. C'est aussi la condition concrète de la réversibilité contractuelle.

POUR L'OPÉRATIONNEL

Astuce : générer en priorité les runbooks d'incidents les plus fréquents (top 10). C'est là que la documentation a la plus grande valeur immédiate, et c'est là que les équipes d'astreinte gagnent le plus de sérénité.

CAS D'USAGE N°5

Analyse et priorisation des tickets

CONTEXTE

Portefeuille applicatif d'un acteur du retail : 4 applications, ~300 tickets ouverts en permanence, ~80 nouveaux par mois. Le tri manuel mange le temps du SDM. Doublons, mauvais aiguillages, priorités contestées entre métiers.

AVANT IA

Le SDM passe 6 à 8 heures par semaine à trier, dispatcher, requalifier les tickets. Les patterns récurrents sont identifiés tard. Certains tickets dorment plusieurs semaines avant d'être correctement traités.

AVEC ASSISTANCE IA

À l'ouverture d'un ticket, l'IA propose une classification (catégorie, application, criticité estimée), détecte les doublons potentiels avec des tickets existants, et suggère une priorité. Le SDM valide ou ajuste en quelques clics.

Bonus : analyse trimestrielle des patterns récurrents pour identifier les zones de fragilité à traiter en priorité.

RÉSULTATS OBSERVÉS

- Temps de tri ramené de **6-8h/semaine à 1h30-2h/semaine**.
- ~12 % de doublons détectés et fusionnés en amont, qui auparavant passaient en double traitement.
- Identification proactive de patterns : ex. "23 % des tickets de la dernière saison concernaient le module facturation des retours" → arbitrage de refonte ciblée plus facile à défendre.

GARDE-FOUS

- L'IA ne ferme jamais un ticket toute seule, même les doublons "évidents" : la fusion est validée par le SDM.
- Aucune communication client n'est générée et envoyée automatiquement.
- Les classifications sont auditables ; le client peut demander à voir la justification de chaque suggestion.

POUR LE DSI

Levier FinOps et gouvernance Run. Le pilotage des tickets devient une source d'information stratégique : où concentrer les évolutions, quelles applications sont fragiles, quels modules méritent une refonte. C'est aussi un excellent outil de pilotage du contrat.

POUR L'OPÉRATIONNEL

Bien intégrer l'IA à votre ITSM existant (Jira Service Management, ServiceNow, ZenDesk, etc.) via les webhooks d'ouverture de ticket. Pas besoin de migrer d'outil. Garder le ticketing comme source de vérité.

PARTIE 4

04 Garde-fous et bonnes pratiques.

Sept garde-fous structurent l'usage de l'IA dans nos missions Run. Aucune exception, même sur les demandes urgentes. Ces principes sont contractuellement opposables.

1

Pas de production sans humain

Aucune action sur un environnement de production n'est déclenchée par une IA sans validation humaine explicite. Ni redémarrage de service, ni rollback, ni modification de configuration, ni envoi d'alerte client. C'est la ligne rouge.

2

Sécurisation des secrets et données sensibles

Aucun secret, aucune clé d'API, aucune donnée à caractère personnel non strictement nécessaire ne transite par un appel modèle. Outillage : pre-commit hooks, scans Gitleaks, MCP locaux, anonymisation des logs en amont.

3

Sandbox de test obligatoire

Toute proposition IA (correctif, refactoring, génération de tests) passe d'abord par un environnement de test isolé. Validation fonctionnelle complète avant remontée vers la pré-prod. Pas de "petite correction directe en prod" assistée par IA.

4

Audit trail complet

Chaque session IA est tracée : prompt initial, contexte fourni, sortie brute, modifications humaines, action finale exécutée. Cet audit log est conservé 12 mois minimum et remis au client à la fin du contrat ou en cas de réversibilité.

5

Formation continue des consultants

Tous nos consultants Run suivent un cursus annuel sur l'usage encadré de l'IA générative : reconnaissance des hallucinations, bonnes pratiques de prompt, garde-fous éthiques, conformité RGPD et EU AI Act. Pas de "freestyle".

6

Les patterns d'usage de l'IA sont partagés et challengés en interne via notre communauté technique. Une "biblio de prompts validés" est maintenue pour les cas d'usage récurrents. Pas d'usage solitaire non documenté.

7

Politique d'usage IA contractualisée

Notre politique d'usage IA est annexée à chaque contrat Run. Vous savez exactement ce que nous faisons, avec quels outils, avec quelles données, et avec quelles limites. Aucune surprise, aucune ambiguïté.

4.1 — Conformité réglementaire

Notre cadre d'usage de l'IA est conçu pour rester conforme aux deux principaux référentiels applicables :

RGPD

Tout traitement de données personnelles est documenté dans un registre dédié au contrat. Anonymisation systématique avant tout appel modèle. Sous-traitance des fournisseurs IA encadrée par DPA. Droit à l'effacement honoré dans les délais légaux.

EU AI Act

Nos usages relèvent majoritairement de la catégorie "risque limité" (assistance au développement, analyse de logs, génération de documentation). Pour les usages susceptibles d'être qualifiés "haut risque" (ex. classification automatique de tickets dans un contexte santé ou bancaire critique), nous appliquons les exigences additionnelles : documentation technique, contrôle humain renforcé, évaluation des biais, journalisation étendue.

4.2 — Risques connus et mitigations

RISQUE	DESCRIPTION	MITIGATION APOLLO
Hallucinations	Le modèle invente une dépendance, une fonction, une API qui n'existe pas	Validation systématique par exécution (tests, compilation, recherche dans le code source)
Drift	Évolution silencieuse du modèle entre versions, sorties différentes pour un même prompt	Versionnage du modèle utilisé, prompts archivés, contrats avec engagement de version stable
Biais	Suggestions de classification ou de priorisation reflétant des biais d'entraînement	Calibration sur historique client, revue d'échantillons trimestrielle, justification accessible
Fuite de contexte	Données sensibles incluses involontairement dans un prompt	Filtrage en amont, MCP locaux, audit log, formation des consultants
Sur-confiance		

RISQUE	DESCRIPTION	MITIGATION APOLLO
	Validation humaine devenue tamponnée, perte d'esprit critique	Politique de rotation, revue croisée, indicateurs de qualité des sorties
Coût	Explosion des coûts d'appel modèle sur certains usages volumiques	Quotas par mission, suivi mensuel, arbitrages cas par cas modèle léger vs lourd

Un risque que vous voyez se traite. Un risque que vous ignorez vous fragilise.

PARTIE 5

05 Mesurer la valeur : six KPI.

Si vous n'avez pas mesuré la valeur, vous n'en avez pas eu. C'est aussi vrai pour l'IA dans le Run que pour n'importe quelle transformation. Voici les six indicateurs que nous suivons systématiquement.

KPI	DÉFINITION	CIBLE APOLLO (TYPIQUE)
1. Gain de temps moyen par tâche	% de réduction du temps consacré à une tâche type (diagnostic, génération de tests, doc)	30 à 60 %
2. Taux de suggestion IA acceptée	% de suggestions IA retenues par le consultant senior après revue	50 à 70 %
3. Réduction du MTTR	Évolution du temps moyen de résolution incident, périmètre comparable	– 30 à – 50 %
4. Réduction de la dette technique	Score qualité code (Sonar ou équivalent), évolution sur 6 mois glissants	Stable ou ↘
5. Couverture de tests	% de lignes / branches couvertes par tests automatisés	+ 10 pts en 6 mois
6. Satisfaction consultant	NPS interne des consultants sur l'usage des outils IA Apollo	≥ 8 / 10

5.1 — Pourquoi ces six-là et pas d'autres

Les KPI 1, 2 et 3 mesurent la **valeur opérationnelle immédiate**. Ce sont ceux que regarde le responsable Run.

Les KPI 4 et 5 mesurent la **valeur patrimoniale long terme**. Ce sont ceux que regarde le DSI.

Le KPI 6, la satisfaction consultant, n'est pas un luxe : un consultant frustré par un outil mal calibré dégrade la qualité de service. Et inversement, un consultant qui se sent augmenté par ses outils reste plus longtemps. Le turnover est l'ennemi mortel du Run.

POUR LE DSI

Ces six KPI sont intégrés à notre tableau de pilotage Run mensuel. Vous les recevez sans demande spécifique. Ils complètent — sans remplacer — les SLA contractuels de niveau de service.

5.2 — Ce qu'il ne faut pas mesurer

Quelques fausses bonnes idées qu'on rencontre fréquemment dans les benchmarks IA :

- **Le nombre de lignes de code générées par IA.** Métrique sans signification : générer beaucoup de code médiocre n'est pas un succès.
- **Le pourcentage de code "écrit par IA".** Ambiguë et facilement détournée : un développeur peut adopter une suggestion sans en comprendre la portée.
- **Le coût absolu d'appel IA.** À regarder uniquement rapporté à un usage métier (€/incident, €/ticket résolu), jamais en valeur brute.

Mesurez ce qui change pour vos utilisateurs et pour votre patrimoine — pas ce qui flatte le bilan IA.

ANNEXE A

Grille de maturité IA dans le Run.

Quatre niveaux pour situer votre organisation et tracer une trajectoire d'évolution. Cette grille s'utilise comme miroir, pas comme objectif imposé : chaque niveau peut être une cible suffisante selon votre contexte.

N0 AD HOC	Usage individuel et non documenté de l'IA par certains consultants. Pas de cadre, pas de partage, pas de mesure. Risque maîtrisé tant que personne ne pousse en production sans relecture, mais peu de valeur capturée. *État de départ pour la majorité des contrats Run actuels.*
N1 OUTILLÉ	L'équipe dispose des outils IA partagés (Claude Code ou équivalent), avec une politique d'usage écrite. Quelques cas d'usage récurrents sont identifiés (typiquement diagnostic d'incident, génération de docs). Pas encore de KPI ni de traçabilité systématique. *Maturité visée à 3-6 mois d'un démarrage de contrat.*
N2 INDUSTRIALISÉ	Les cinq cas d'usage (ou un sous-ensemble cohérent) sont intégrés aux workflows quotidiens. Audit log activé. KPI suivis mensuellement. Communauté technique active. Les gains sont mesurables et reportables au client. *Standard cible Apollo sur un contrat Run mature.*
N3 DIFFÉRENCIANT	L'usage de l'IA dans le Run devient un actif stratégique : modèles de prompts spécifiques au client, corpus de diagnostic enrichi en continu, MCP métiers développés sur mesure. Réversibilité documentée même sur la couche IA. *Maturité observée sur quelques contrats long terme stratégiques.*

Important : passer du N0 au N3 n'est pas un objectif universel. Sur un parc applicatif peu critique avec faible volumétrie, rester au N1 peut être tout à fait suffisant. La grille sert à **choisir consciemment**, pas à courir après une maturité dont vous n'auriez pas besoin.

ANNEXE B

Checklist d'auto-évaluation.

Quinze questions à vous poser avant d'intégrer l'IA dans votre Run, ou pour challenger votre dispositif actuel. Une majorité de "non" indique un terrain à préparer avant d'aller plus loin.

PRÉPARATION DU TERRAIN

- [ ] Disposez-vous d'une politique d'usage IA écrite, approuvée par votre DSI et votre direction juridique ?
- [ ] Vos contrats Run actuels comportent-ils une clause encadrant l'usage de l'IA par le prestataire ?
- [ ] Avez-vous identifié les données sensibles à exclure des prompts (PII, secrets, données métier confidentielles) ?
- [ ] Disposez-vous d'un environnement de test isolé suffisamment représentatif de la production ?

WORKFLOW ET GOUVERNANCE

- [ ] Avez-vous défini qui valide les sorties IA, et selon quels critères ?
- [ ] Disposez-vous d'un audit log capable de retracer les usages IA dans vos workflows Run ?
- [ ] Vos consultants (internes ou prestataires) ont-ils reçu une formation à l'usage encadré de l'IA générative ?
- [ ] Existe-t-il un mécanisme de peer review sur les usages IA récurrents ?

CAS D'USAGE ET MESURE

- [ ] Avez-vous identifié au moins un cas d'usage prioritaire à fort potentiel de valeur (parmi les cinq présentés) ?
- [ ] Disposez-vous d'une mesure baseline (avant IA) pour comparer les gains ?
- [ ] Avez-vous défini les KPI que vous suivrez ?

- [ ] Avez-vous une cadence de revue (mensuelle/trimestrielle) des résultats IA ?

RÉVERSIBILITÉ ET CONFORMITÉ

- [ ] Si votre prestataire actuel quittait demain, récupéreriez-vous l'audit log de ses usages IA ?
- [ ] Vos usages IA ont-ils été évalués au regard de l'EU AI Act (catégorie de risque) ?
- [ ] Avez-vous identifié un référent IA interne capable d'arbitrer les questions sensibles ?

*Vous pouvez nous solliciter pour conduire cette auto-évaluation lors de notre **atelier de cadrage** de 30 minutes — sans engagement.*

ANNEXE C

Glossaire.

Les termes employés dans ce document, expliqués sans condescendance — chacun sait certains, ignore d'autres. Volontairement court : un glossaire trop long n'est pas lu.

Agent (IA)

Programme combinant un modèle de langage et des outils (lecture/écriture de fichiers, exécution de commandes, appels API) capable d'enchaîner des actions pour atteindre un objectif. Claude Code est un agent.

Audit log (IA)

Journal qui trace les prompts envoyés, le modèle utilisé, les sorties produites, et les actions humaines associées. Indispensable pour la traçabilité et la conformité.

Claude Code

Outil d'Anthropic combinant le modèle Claude et un agent d'exécution capable d'opérer dans un répertoire de code délimité. Choix principal d'Apollo pour l'IA dans le Run, pour ses capacités de contrôle de contexte et d'audit.

EU AI Act

Règlement européen sur l'intelligence artificielle, applicable progressivement à partir de 2025-2027. Classe les usages IA en quatre catégories de risque (interdit, haut risque, risque limité, risque minimal) avec des exigences associées.

Fine-tuning

Procédure d'ajustement d'un modèle de base sur un corpus spécifique. Apollo ne pratique pas le fine-tuning sur les données clients : le contexte est apporté à chaque session, pas appris durablement.

Hallucination

Production par un modèle d'une information fausse mais formulée avec assurance (fonction inexistante, dépendance inventée, fait erroné). Risque majeur du Run, traité par la validation systématique d'exécution.

LLM (Large Language Model)

Modèle de langage de grande taille, entraîné sur un corpus massif, capable de générer du texte cohérent. Claude, GPT, Gemini, Llama sont des LLM.

MCP (Model Context Protocol)

Protocole standard pour exposer des outils, des sources de données ou des actions à un modèle d'IA, de manière sécurisée et auditable. Apollo utilise des MCP locaux pour limiter au strict nécessaire ce qui sort du périmètre client.

MTTR (Mean Time To Resolve / Recover)

Temps moyen de résolution d'un incident. Indicateur central du Run. Distinguer MTTR P1 (critique), P2 (majeur), P3 (mineur).

Prompt

Instruction donnée à un modèle d'IA. Sa qualité (cadre, contexte, contraintes, format attendu) conditionne fortement la qualité de la sortie.

RAG (Retrieval-Augmented Generation)

Technique consistant à enrichir un prompt avec des extraits de documents pertinents récupérés dynamiquement, plutôt que de tout apprendre dans le modèle. Très utile pour exposer une base de connaissance Run sans fine-tuning.

Sandbox

Environnement isolé dans lequel on peut exécuter du code ou tester une modification sans risque pour la production. Obligatoire pour toute validation de sortie IA.

SDM (Service Delivery Manager)

Responsable de la livraison du service Run côté Apollo. Point de contact unique du client, garant des SLA, animateur du tableau de pilotage mensuel.

TMA (Tierce Maintenance Applicative)

Maintien en condition opérationnelle d'une application par un prestataire tiers. Corrective, évolutive, préventive. Cœur historique de l'offre Run Apollo.

ANNEXE D

À propos d'Apollo.

Apollo est une ESN française spécialisée dans la conception, le développement et l'exploitation d'applications métier stratégiques. Trois agences (Paris, Lyon, Grenoble), une expertise revendiquée sur trois stacks (.NET, Java, JavaScript), une posture de partenaire de cycle de vie complet — Design, Build, Run.

21+ années d'expertise	200+ projets livrés	+100 consultants certifiés
98 % satisfaction client	7 ans d'expérience moyenne	100 % made in France

Nos engagements

- **Microsoft Gold Partner** sur l'écosystème .NET / Azure.
- **UN Global Compact** — engagements en matière de droits humains, normes du travail, environnement et anti-corruption.
- **Ecovadis Silver** — performance RSE.
- **Label Égalité Professionnelle**.
- **Conformité RGPD** dans toutes nos missions.

Notre offre Run applicatif

Trois modèles cumulables selon la maturité et la gouvernance souhaitées :

- **TMA classique** — maintenance corrective, évolutive et préventive, au forfait, avec engagement SLA.
- **Run capacitaire** — Run et petites évolutions portés par la même équipe, sans rupture build/run.
- **Outsourcing applicatif** — centre de services externalisé sur un portefeuille applicatif complet.

Et après ce livre blanc ?

Si ce document vous a intéressé et que vous souhaitez creuser, nous proposons un **atelier de cadrage de 30 minutes**, gratuit et sans engagement, avec un Service Delivery Manager Apollo. Objectif : comprendre votre contexte, identifier 1 à 2 cas d'usage prioritaires, et vous remettre un premier diagnostic.

Nous contacter

Téléphone — +33 (0)4 78 35 45 70

Email — contact@apollossc.com

Agences — Paris · Lyon · Grenoble

Web — apollossc.com/run-applicatif