LIVRE BLANC

De l’ambition IA au système en production

Relier agent engineering, infrastructure d’inférence, opérations et valeur métier

UNE GRILLE DE LECTURE SOCIO-TECHNIQUE ET UN GUIDE DE PASSAGE À L’ÉCHELLE

Amine Essafoui | WeSwell

Version 1.0 — août 2026

Note au lecteur

Ce livre blanc poursuit un double objectif. Il propose aux dirigeants, responsables produit, architectes et équipes plateforme une grille de lecture pour transformer une ambition ou un prototype d'intelligence artificielle en service réellement utilisable. Il sert aussi de parcours d'apprentissage structuré pour celles et ceux qui veulent comprendre l'écosystème qui relie les modèles, l'inférence, les agents, les opérations et la valeur métier.

Il ne présente ni recette universelle ni catalogue d'outils. Les technologies changent vite ; les contraintes fondamentales changent moins vite. Un service d'IA doit produire un résultat utile, sous des contraintes de coût, de délai, de qualité, de sécurité et de responsabilité. La difficulté n'est donc pas seulement de choisir un modèle. Elle consiste à concevoir un système socio-technique capable d'apprendre de sa propre production.

Statut des affirmations

Les faits techniques et les descriptions de produits sont reliés à des sources primaires. Les propositions d'intervention et les grilles de décision sont des synthèses de l'auteur. Les scénarios de mission sont des cas-types, pas des références clients réalisées.

À propos de l'auteur

Amine Essafoui se définit comme ingénieur des systèmes socio-techniques. Son travail porte sur les flux, les dépendances, les décisions, les interfaces, les responsabilités et les conditions qui permettent à une organisation de créer de la valeur sans user inutilement les personnes qui la produisent. Ce livre marque l'extension de cette pratique aux systèmes d'IA agentiques et à leur passage en production.

Comment lire ce document

- Décideur : lire la synthèse, les chapitres 1, 5, 6 et 8.
- Responsable technique : lire les chapitres 2 à 7, puis les annexes A et B.
- Praticien en apprentissage : suivre l'intégralité du document et exécuter le lab de 30 jours en annexe C.

Sommaire

Synthèse exécutive

L'IA générative est souvent traitée comme un composant logiciel supplémentaire : une API entre, une réponse sort. Cette représentation suffit pour une démonstration. Elle devient dangereuse en production. Un agent combine un modèle probabiliste, des instructions, des outils, des données, des permissions, de la mémoire, des boucles de contrôle et des infrastructures dont les comportements varient avec la charge et le contexte. Il agit dans une organisation qui doit pouvoir l'observer, le corriger et assumer ses effets.

La thèse de ce livre est simple : le passage à la production n'est pas un problème de modèle, mais un problème de système. Il faut relier trois couches techniques à une quatrième couche organisationnelle.

Couche	Question structurante	Échec typique
1. Modèle	Quelle capacité cognitive et quel comportement de base ?	Choisir le benchmark plutôt que le besoin.
2. Inférence	Comment servir cette capacité avec un SLO et une économie soutenables ?	Ignorer latence, débit, mémoire et coûts réels.
3. Agent	Comment composer contexte, outils et contrôle pour accomplir une tâche ?	Confondre démonstration et comportement fiable.
4. Système opérant	Qui décide, mesure, surveille, corrige et assume ?	Ajouter de la technologie sans boucle d'apprentissage.

Conviction centrale

La valeur n'apparaît pas quand un agent "fonctionne". Elle apparaît quand une organisation peut lui confier une tranche de travail précise, mesurer le résultat, maîtriser les risques et améliorer le système à partir des traces réelles.

Les travaux californiens étudiés insistent sur la composition des agents, l'observabilité, les évaluations et la discipline de production. Les travaux chinois étudiés rendent visible l'autre moitié du problème : la mémoire, le cache KV, l'ordonnancement, la désagrégation du prefill et du decode, l'utilisation des accélérateurs et l'économie du token. L'entreprise européenne qui veut industrialiser l'IA doit savoir faire travailler ensemble ces deux rationalités.

La trajectoire proposée commence par un point d'appui étroit : un flux métier à valeur mesurable, une architecture minimale, des évaluations avant déploiement, une observabilité de bout en bout et une décision explicite d'élargir, pivoter ou arrêter. La preuve précède l'échelle.

1. Pourquoi les prototypes IA se grippent

Un prototype optimise la vitesse d'apprentissage initiale. Un service optimise un compromis durable : utilité, fiabilité, délai, coût, sécurité et capacité d'évolution. Les deux objets n'ont ni les mêmes exigences ni les mêmes acteurs. Le problème commence lorsque l'organisation attend du premier qu'il se transforme mécaniquement en second.

1.1 Le prototype masque les interfaces

Dans une démonstration, les entrées sont choisies, l'utilisateur est patient, les permissions sont larges et l'erreur est interprétée avec bienveillance. En production, les entrées sont ambiguës, les outils tombent en panne, les données sont incomplètes, les utilisateurs contournent les règles et chaque appel a un coût. Les interfaces invisibles deviennent alors le système réel.

- Interface métier : qu'est-ce qu'un résultat acceptable, utile et vérifiable ?
- Interface agent-outil : que se passe-t-il si un outil répond lentement, partiellement ou deux fois ?
- Interface agent-modèle : quel comportement change avec le modèle, le prompt ou le contexte ?
- Interface service-infrastructure : comment les files, caches et limites de capacité affectent-elles l'expérience ?
- Interface organisationnelle : qui possède la qualité, le coût, l'incident et l'amélioration ?

1.2 L'activité est confondue avec l'impact

Nombre de prompts, volume de tokens, taux d'utilisation ou quantité de POC décrivent une activité. Ils ne prouvent pas la valeur. Une mesure utile relie au moins quatre dimensions : outcome métier, qualité du comportement, performance du service et soutenabilité opérationnelle.

Dimension	Exemples de mesure	Question
Valeur	temps de traitement évité, conversion, résolution, coût du délai	Le travail produit-il un effet utile ?
Qualité	exactitude, trajectoire, bonne sélection d'outil, taux d'escalade	Le résultat est-il digne de confiance ?
Service	latence, disponibilité, débit, erreurs, saturation	L'expérience tient-elle sous charge ?
Économie	coût par tâche réussie, tokens utiles, GPU-hours	La valeur reste-t-elle soutenable ?

Dimension	Exemples de mesure	Question
Humain	charge de revue, interruptions, capacité de reprise	Le système réduit-il ou déplace-t-il la friction ?

1.3 La dette se déplace

L'agent peut accélérer une étape tout en augmentant la charge ailleurs : davantage de revue, d'exceptions, d'investigation ou de coordination. Le bon indicateur n'est donc pas le coût par requête, mais le coût par résultat accepté, incluant la reprise humaine et les effets en aval.

2. Cartographier l'écosystème sans se perdre

La carte en quatre couches évite deux réductions symétriques : croire que l'IA est uniquement un problème d'application, ou croire qu'elle est uniquement un problème de calcul. Chaque couche possède ses objets, ses métriques et ses modes de défaillance.

2.1 Couche 1 — le modèle

Le modèle fournit une capacité générale ou spécialisée : génération, raisonnement, code, multimodalité, usage d'outils. Le choix ne peut pas reposer sur un classement global. Il dépend d'une distribution de tâches, de langues, de contraintes de contexte, de politiques de données et d'une enveloppe économique.

2.2 Couche 2 — l'inférence

Servir un modèle transforme des poids en service. Cette couche gère notamment le placement, le batching, l'ordonnancement, le cache KV, la quantification, la parallélisation, le routage et les objectifs de service. Elle arbitre en permanence entre time-to-first-token, vitesse de génération, débit, utilisation mémoire et coût.

Pourquoi le cache KV est structurant

À chaque token, le modèle réutilise des représentations des tokens précédents. Ce cache peut devenir la ressource dominante. Le gérer comme une mémoire paginée, partagée ou hiérarchisée modifie directement la capacité, la latence et l'économie du service.

2.3 Couche 3 — l'agent

Un agent est un programme dont une partie du contrôle est déléguée au modèle. Il combine un état, des règles, des outils et une boucle d'exécution. La sophistication n'est pas une vertu : chaque boucle, sous-agent ou mémoire supplémentaire accroît l'espace des comportements possibles et donc la surface à évaluer.

2.4 Couche 4 — le système opérant

Cette couche réunit produit, plateforme, sécurité, exploitation, juridique et métier. Elle transforme les traces en décisions : quels incidents comptent, quels exemples rejoignent le jeu d'évaluation, quelle version est promue, quelle action requiert une approbation humaine et qui peut arrêter le système.

2.5 L'interface critique entre agent et inférence

Un choix applicatif change la charge d'inférence. Un long contexte augmente la mémoire et le délai. Un agent parallèle augmente la concurrence. Une boucle de réflexion augmente les tokens de sortie. Un cache de préfixe peut rendre une architecture économique ; une variation de prompt peut annuler ce bénéfice. La séparation des équipes ne doit donc pas devenir une séparation des modèles mentaux.

Décision agentique	Effet probable sur l'inférence	Question de conception
Contexte plus long	plus de prefill, mémoire KV accrue	Quelle information influence réellement la décision ?
Agents en parallèle	pics de concurrence et fan-out	Quel gain de délai compense le surcoût ?
Réessais automatiques	charge amplifiée lors des pannes	Le retry est-il borné, observable et idempotent ?
Sortie structurée	décodage contraint	Le schéma améliore-t-il l'intégration et l'évaluation ?
Mémoire conversationnelle	croissance du contexte et risques de contamination	Que retenir, résumer, expirer ou isoler ?

3. Concevoir des agents fiables

3.1 Commencer par le workflow déterministe

La littérature d'ingénierie des agents converge sur une règle pragmatique : utiliser la structure la plus simple qui accomplit la tâche. Un workflow explicite est plus facile à tester et à opérer qu'une autonomie ouverte. L'agent devient pertinent quand la variété des situations rend impossible l'écriture préalable de chaque chemin.

3.2 L'évaluation est une spécification exécutable

Un système probabiliste ne peut pas être protégé uniquement par des tests d'égalité. Il faut décomposer la qualité : résultat final, étapes intermédiaires, sélection d'outil, arguments, respect des politiques, latence et coût. LangSmith distingue l'évaluation hors ligne — avant livraison — et l'évaluation en ligne sur les traces réelles. Anthropic décrit les évaluations comme un moyen de rendre visibles les défaillances et changements de comportement avant qu'ils n'atteignent les utilisateurs.

- Construire 5 à 10 exemples représentatifs par composant critique, puis enrichir le corpus avec les échecs réels.
- Combiner règles déterministes, revue humaine, comparaison par paires et LLM-as-judge.
- Évaluer la trajectoire lorsque le chemin compte : outil choisi, ordre des actions, appels interdits.

- Versionner données, instructions, outils, modèles et évaluateurs afin d'expliquer une régression.

3.3 L'observabilité doit raconter une tâche

Les métriques d'infrastructure ne suffisent pas. Une trace utile relie l'intention utilisateur, la version du système, les décisions de l'agent, les appels d'outils, les réponses, les tokens, les délais et le feedback. Elle permet de passer d'un incident — "la réponse était mauvaise" — à une hypothèse testable.

3.4 Les permissions font partie de l'architecture

Un agent qui agit doit être traité comme un acteur du système : identité, droits minimaux, séparation des environnements, journal d'audit, secrets gérés et approbation pour les actions irréversibles. Le contrôle humain n'est pas un bouton ajouté après coup ; il est un embranchement du workflow.

3.5 Concevoir la récupération, pas seulement le chemin heureux

Chaque étape doit préciser son comportement en cas de timeout, réponse partielle, duplication, indisponibilité ou résultat ambigu. L'idempotence, les limites de retry, les checkpoints et les files de reprise deviennent des propriétés produit, car ils déterminent ce que vit l'utilisateur.

4. Servir les modèles : la discipline cachée

Le niveau d'inférence est souvent invisible aux équipes produit parce qu'une API en masque la complexité. Pourtant, lorsque le volume, la confidentialité ou la différenciation augmentent, cette couche détermine directement l'expérience et la marge.

4.1 Quatre métriques à tenir ensemble

Métrique	Définition opérationnelle	Tension principale
TTFT	temps avant le premier token	préfill, file, cache, charge
ITL / TPOT	intervalle entre tokens	decode, contention, kernels
Débit	tokens ou requêtes par unité de temps	batching et utilisation GPU
Coût par tâche réussie	coût total rapporté à l'outcome accepté	qualité, longueur, retries, exploitation

4.2 PagedAttention : traiter la mémoire comme un système

Le papier vLLM introduit PagedAttention pour réduire la fragmentation du cache KV et permettre un partage plus efficace. Le principe est moins important que la leçon : dans l'inférence, la gestion de mémoire n'est pas une optimisation marginale ; elle structure la capacité du service.

4.3 RadixAttention : réutiliser les préfixes

SGLang exploite les préfixes communs entre requêtes à travers RadixAttention et accélère aussi les sorties structurées. Pour un système agentique qui réutilise de longues instructions ou des historiques proches, la stabilité du préfixe devient une décision économique.

4.4 Désagréger prefill et decode

Le prefill traite le contexte en parallèle et mobilise fortement le calcul ; le decode génère token après token et est souvent limité par la mémoire. Mooncake, issu de Moonshot AI, propose une architecture KV-centric désagrégée, avec stockage hiérarchique entre VRAM, DRAM et NVMe. Cette séparation permet d'optimiser et d'élasticiser différemment les deux phases.

4.5 L'économie ne se résume pas au prix du token

DeepSeek a publié un aperçu de son système V3/R1 : nœuds de prefill et de decode séparés, cache KV sur disque et données de charge et de coût. Ces chiffres ne sont pas un tarif transposable ; ils montrent la nature industrielle du problème. Le coût dépend du taux d'utilisation, des longueurs d'entrée et de sortie, des hits de cache, du routage et de la capacité à absorber les pointes.

Unité économique recommandée

Mesurer le coût par tâche métier acceptée. Cette unité absorbe les tokens, les retries, les appels d'outils, les échecs, la revue humaine et l'infrastructure. Elle rapproche la plateforme du langage du sponsor.

5. Californie et Chine : deux écoles complémentaires

Parler d'"écoles" est ici une grille de lecture, non une frontière géographique absolue. Les projets sont internationaux et open source. Mais les corpus étudiés révèlent deux centres de gravité utiles.

Centre de gravité	Californie / écosystème agent	Chine / écosystème open inference
Objet visible	composer, évaluer et opérer des agents	maximiser la performance d'inférence
Abstraction	graphe, trace, dataset, outil, garde-fou	token, cache KV, batch, nœud, topologie
Question	comment rendre le comportement fiable ?	comment servir à grande échelle et coût maîtrisé ?
Forces	boucles produit, observabilité, intégration entreprise	efficacité système, désagrégation, open weights
Risque si isolé	ignorer la physique et l'économie du serving	optimiser des tokens sans outcome métier

5.1 Leçons du corpus californien

- Anthropic : privilégier les patterns simples, rendre les évaluations concrètes et adaptées au domaine.
- LangChain/LangSmith : relier orchestration, traces, évaluations hors ligne et surveillance en production.
- OpenAI AgentKit : rapprocher construction, déploiement, connecteurs, datasets et optimisation dans un cycle continu.
- vLLM et SGLang, nés dans l'écosystème Berkeley : faire du runtime une discipline d'ingénierie ouverte.

5.2 Leçons du corpus chinois

- DeepSeek : rendre publics des éléments de l'architecture et de l'économie de production.
- Moonshot/Mooncake : concevoir autour du cache KV et du mouvement des données, jusqu'à désagréger les phases de calcul.
- Qwen : publier modèles, variantes de déploiement et compatibilités avec plusieurs runtimes.
- Culture d'ingénierie visible : quantification, parallélisme, utilisation des accélérateurs et optimisation de bout en bout.

5.3 Le point d'intégration européen

L'opportunité n'est pas de copier un camp. Elle consiste à intégrer les deux : agent engineering, économie du serving, exigences de souveraineté et contraintes d'organisation. Cette compétence d'interface est rare parce qu'elle traverse produit, logiciel, infrastructure, sécurité et métier.

Hypothèse stratégique

La différenciation durable se situera moins dans la connaissance d'un framework que dans la capacité à concevoir et opérer la boucle complète : intention métier → agent → inférence → traces → évaluation → décision d'amélioration.

6. Le système opérant : produit, plateforme et responsabilité

6.1 Une équipe n'"adopte" pas un agent : elle redistribue du travail

Introduire un agent déplace des décisions, des exceptions et des responsabilités. Il faut cartographier le flux avant et après : qui initie, qui valide, qui reprend, qui apprend et qui assume. Sans cette cartographie, l'automatisation locale peut dégrader la chaîne de valeur.

6.2 La plateforme comme produit

Une plateforme IA interne est pertinente lorsque plusieurs équipes portent la même complexité : accès aux modèles, secrets, observabilité, évaluations, déploiement, politiques ou maîtrise des coûts. Son rôle est d'offrir des chemins sûrs et simples, pas d'imposer un contrôle abstrait. Elle doit mesurer l'adoption, le temps économisé et la qualité obtenue par ses utilisateurs internes.

6.3 Build, buy, partner : décider par capacité distinctive

Choix	Pertinent lorsque…	Signal d'alerte
Acheter	la capacité est standard et la vitesse prime	dépendance sans stratégie de sortie
Construire	le comportement ou l'économie est différenciant	plateforme générique sans utilisateurs
Open source opéré	contrôle et portabilité sont stratégiques	coût d'exploitation sous-estimé
Partenariat	la compétence doit être transférée par le terrain	délégation permanente de la connaissance

6.4 Gouverner par les preuves

La gouvernance utile ne demande pas uniquement des documents. Elle définit des portes de décision alimentées par des preuves : qualité sur dataset, comportement en canary, incidents, coût par outcome,

capacité de reprise et acceptation métier. Une équipe conserve de l'autonomie tant qu'elle rend ces preuves visibles et assume les responsabilités associées.

7. Une méthode de passage à la production

La méthode suivante transpose neuf principes d'intervention socio-technique au contexte IA. Elle ne promet pas une transformation globale. Elle cherche un point d'appui qui permette au système d'apprendre honnêtement.

Étape	Travail	Preuve produite
1. Mandat	formuler le problème, l'outcome, le sponsor et les contraintes	fiche mandat et critères de succès
2. Flux actuel	observer le travail, les interfaces, exceptions et coûts	carte du flux vécu
3. Tranche de valeur	choisir une tâche étroite et fréquente	périmètre testable et baseline
4. Architecture	dessiner agent, outils, données, permissions et serving	architecture et registre de risques
5. Évaluation	construire exemples, règles et revue humaine	dataset v0 et seuils
6. Prototype opérable	instrumenter dès le premier chemin utile	traces et dashboard minimal
7. Expérience terrain	canary, shadow mode ou population limitée	résultats comparés à la baseline
8. Décision	continuer, élargir, pivoter ou arrêter	preuve de valeur et prochain pari

7.1 Les neuf principes traduits en IA

Principe	Traduction
Lire le système vécu	observer les tâches, contournements et reprises, pas seulement le process déclaré.
Lire culture et pouvoir	voir qui peut interrompre, contester ou refuser une décision automatisée.
Ne pas labourer l'océan	partir d'une tranche de valeur étroite.
Cartographier les interfaces	inclure métier, données, outils, sécurité, plateforme et exploitation.
Juste assez de représentation	aligner sur les critères nécessaires pour agir.
Désaccords = informations	transformer les divergences sur la qualité en exemples d'évaluation.
Diagnostic → hypothèse	définir une amélioration testable, pas une solution totale.

Principe	Traduction
Mesurer sans abîmer	équilibrer valeur, qualité, vitesse, coût et charge humaine.
Protéger les conditions de vérité	traces, sponsor, droit d'alerte et seuils d'arrêt.

7.2 Format d'intervention candidat

Un premier engagement peut tenir en six semaines si un sponsor actif, une équipe cœur et l'accès au terrain sont réunis. La durée n'est pas une promesse commerciale figée : elle matérialise une boucle complète, du mandat à la décision.

- Semaine 1 : mandat, flux actuel, baseline, interfaces et risques.
- Semaine 2 : tranche de valeur, architecture, jeu d'évaluation initial.
- Semaines 3-4 : construction instrumentée, tests, revues et scénarios de panne.
- Semaine 5 : expérimentation limitée, observation et collecte de traces.
- Semaine 6 : mesure, transmission, décision et backlog d'apprentissage.

8. Trois scénarios de mission

8.1 Industrialiser un agent métier prometteur

Situation : un prototype répond bien en démonstration mais échoue sur les cas ambigus et n'a ni propriétaire de qualité ni trajectoire de déploiement. Point d'appui : une tâche à volume suffisant, dont la qualité peut être jugée. Résultat attendu : architecture opérable, dataset d'évaluation, traces, garde-fous et preuve comparée à la baseline.

8.2 Construire le premier chemin de plateforme agentique

Situation : plusieurs équipes créent leurs propres wrappers, prompts, observabilité et accès modèles. Point d'appui : un golden path choisi avec deux équipes clientes internes. Résultat attendu : délai de mise en service réduit, politiques explicites, composants mutualisés et mesure d'adoption.

8.3 Arbitrer une stratégie d'inférence

Situation : l'entreprise hésite entre API propriétaire, service spécialisé, open source opéré ou infrastructure interne. Point d'appui : une charge représentative et des SLO explicites. Résultat attendu : benchmark reproductible, coût par tâche réussie, risques de souveraineté et décision build/buy/partner.

Scénario	Sponsor naturel	Équipe cœur	Preuve principale
Agent métier	direction produit ou métier	produit, engineering, expert métier	qualité et valeur sur cas réels
Plateforme	CTO / plateforme	plateforme + équipes pilotes	temps de mise en service et adoption
Inférence	CTO / infrastructure	ML systems, FinOps, sécurité	SLO, coût, portabilité

9. Ce que cette lecture change pour les décideurs

- Financer une boucle d'apprentissage, pas seulement un POC.
- Nommer un sponsor responsable de l'outcome et des conditions d'accès au terrain.
- Exiger un coût par tâche réussie et non un prix de token isolé.
- Traiter les évaluations et les traces comme des actifs produit.
- Décider explicitement quelles actions exigent un humain et pourquoi.
- Faire de la plateforme un produit interne avec des clients et des métriques.
- Élargir uniquement lorsqu'une preuve de valeur et un prochain point d'impact existent.

Question de comité d'investissement

Quelle nouvelle capacité de l'organisation sera démontrée, sur quel flux réel, avec quels critères de succès et quelle décision sera prise à la fin de l'expérience ?

Conclusion — de la démonstration au système apprenant

Le modèle impressionne ; le système crée la valeur. Entre les deux se trouvent des files, des caches, des outils, des permissions, des exceptions, des équipes et des décisions. Les rendre visibles n'alourdit pas nécessairement le projet. Cela évite de découvrir trop tard que la qualité n'a pas de définition, que le coût n'a pas d'unité métier ou que personne ne possède la reprise.

Le passage à la production peut être pensé comme une ingénierie de la représentation : construire juste assez de compréhension commune pour agir sur le bon problème, instrumenter le comportement réel et transformer chaque désaccord ou incident en objet d'apprentissage. La finalité n'est pas une architecture élégante. C'est une amélioration mesurable de la chaîne de valeur, soutenable pour celles et ceux qui la produisent.

La boussole finale tient en une phrase : partir d'un problème réel, choisir une tranche de valeur, relier agent et inférence, mesurer le comportement et l'économie, puis n'élargir qu'à partir d'une preuve.

Annexe A — Architecture de référence

Cette architecture logique n'impose aucun fournisseur. Elle sert à poser les interfaces et responsabilités avant le choix des produits.

Plan	Composants	Responsabilité
Expérience	canal, identité utilisateur, feedback	exprimer l'intention et rendre l'état visible
Agent	graphe, état, règles, mémoire, outils	orchestrer la tâche et borner le comportement
Contrôle	politiques, approbations, garde-fous	autoriser, refuser, interrompre
Évaluation	datasets, evaluators, seuils, comparaison	définir et mesurer la qualité
Observabilité	traces, métriques, coûts, événements	expliquer le comportement réel
Modèle / inférence	routage, runtime, cache, SLO	fournir une capacité sous contraintes
Intégration	API, files, connecteurs, données	fiabiliser les effets sur le SI
Opérations	déploiement, incident, rollback, FinOps	maintenir et améliorer le service

Flux de feedback minimal

Intention métier → exécution tracée → résultat et effet → feedback humain/métier → exemple d'évaluation → comparaison de version → promotion contrôlée.

Annexe B — Matrice de maturité

Domaine	Exploration	Opérable	Apprenant
Outcome	cas d'usage formulé	baseline et critères	impact suivi et arbitrages
Agent	prompt / démo	workflow borné et reprise	patterns adaptés par données
Évaluation	tests manuels	dataset versionné et seuils	production → dataset continu
Observabilité	logs techniques	traces de bout en bout	détection et diagnostic agrégés
Inférence	API choisie	SLO et coûts mesurés	routage et capacité optimisés

Domaine	Exploration	Opérable	Apprenant
Sécurité	revue ponctuelle	identité, droits, audit	politiques et signaux continus
Organisation	champion isolé	équipe cœur et sponsor	ownership distribué et plateforme produit

Une organisation ne doit pas viser le niveau “apprenant” partout. La maturité utile dépend du risque, du volume, de la différenciation et de la réversibilité du cas d’usage.

Annexe C — Lab de crédibilité en 30 jours

Le lab vise une preuve technique et narrative crédible avant l’activation d’un contact chez LangChain. Il ne doit pas démontrer que l’auteur “maîtrise toute l’IA”, mais qu’il sait transformer une tâche en système agentique instrumenté et expliquer ses arbitrages.

Semaine	Objectif	Livrable public
1 — cadrer	choisir un flux, une baseline et 20 cas	README problème/outcome + dataset v0
2 — construire	workflow LangGraph simple, outils réels ou simulés	agent exécutable + architecture
3 — fiabiliser	traces, évaluations, retries, HITL, pannes	rapport d’évaluation et runbook
4 — opérer	déployer, mesurer latence/coût, tester 2 modèles	démo, dashboard, postmortem et décision

Sujet de lab recommandé

Un agent de qualification d’incident socio-technique : il collecte des signaux, interroge une base documentaire, propose des hypothèses, choisit les outils autorisés et prépare un dossier d’escalade. Il n’exécute aucune action irréversible sans approbation. Ce sujet relie naturellement expérience infrastructure, flux, interfaces, responsabilités, charge cognitive et valeur.

Critères de sortie avant activation du contact

- Le dépôt s’installe et se lance avec une procédure courte.
- Le système contient au moins 30 cas d’évaluation, dont des échecs et attaques simples.
- Chaque exécution produit une trace exploitable et un coût estimé.
- Un changement de modèle ou de prompt peut être comparé objectivement.
- Deux pannes d’outil et un timeout ont un comportement de reprise démontré.
- Une vidéo de 5 minutes raconte problème, architecture, preuves et limites.

- Le README distingue clairement ce qui est appris, démontré et encore inconnu.

Définition de crédibilité

Crédible ne veut pas dire complet

Être crédible pour une première mission de Deployed Engineer signifie savoir apprendre vite, construire avec le client, instrumenter le réel, opérer ce que l'on livre et demander de l'aide avec précision. Le lab doit montrer ces comportements.

Annexe D — Curriculum ciblé

Bloc 1 — Agent engineering

- Workflows et agents ; état, outils, mémoire, human-in-the-loop.
- LangGraph : graphes, persistance, streaming, déploiement.
- LangSmith : traces, datasets, évaluations offline/online.
- Sécurité des outils, permissions, idempotence et reprise.

Bloc 2 — Inference engineering

- Anatomie transformer et cache KV ; prefill vs decode.
- Batching continu, PagedAttention, prefix caching et structured decoding.
- Parallélismes tensor/pipeline/expert ; quantification.
- SLO : TTFT, inter-token latency, throughput, tail latency et capacité.
- vLLM, SGLang, Mooncake ; benchmark reproductible et profiling.

Bloc 3 — Production socio-technique

- Outcome, baseline et coût par tâche réussie.
- Value stream du travail humain-agent et charge de reprise.
- Plateforme comme produit, golden paths et expérience développeur.
- Incident, audit, gouvernance par preuve et transmission de capacité.

Lectures indispensables dans l'ordre

Ordre	Lecture	Question à laquelle répondre
1	Anthropic — Building Effective Agents	Quelle structure minimale choisir ?

Ordre	Lecture	Question à laquelle répondre
2	Anthropic — Demystifying evals for AI agents	Comment rendre la qualité observable ?
3	LangSmith — Evaluation & Observability docs	Comment fermer la boucle production-évaluation ?
4	vLLM — PagedAttention paper	Pourquoi la mémoire dicte-t-elle le serving ?
5	SGLang paper	Comment réutiliser les préfixes et structurer l'exécution ?
6	Mooncake paper/repository	Pourquoi désagréger et hiérarchiser le cache KV ?
7	DeepSeek inference overview	À quoi ressemble un système industriel réel ?
8	OCTO Culture DevOps, vol. 1-3	Comment relier organisation, automatisation et qualité ?

Annexe E — Sources et notes

Sources consultées en juillet-août 2026. Les pages produit décrivent les capacités annoncées par leurs éditeurs ; les papiers et dépôts techniques documentent des architectures et résultats dans leurs conditions propres. Aucun benchmark ne doit être transposé sans reproduction sur une charge représentative.

Anthropic. Building effective agents. https://www.anthropic.com/engineering/building-effective-agents

Anthropic. Demystifying evals for AI agents. https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents

LangChain. Documentation : agent engineering, observability and evaluation. https://docs.langchain.com/

LangChain. LangSmith Evaluation. https://docs.langchain.com/langsmith/evaluation

LangChain. Self-hosted LangSmith architectures. https://docs.langchain.com/langsmith/self-hosted

LangChain. Deployed Engineer — description du rôle. https://jobs.ashbyhq.com/langchain/2d920c9c-abf2-4c46-a01b-03e61a03c8e6

OpenAI. Introducing AgentKit. https://openai.com/index/introducing-agentkit/

Kwon et al.. Efficient Memory Management for Large Language Model Serving with PagedAttention. https://arxiv.org/abs/2309.06180

Zheng et al.. SGLang: Efficient Execution of Structured Language Model Programs. https://arxiv.org/abs/2312.07104

Moonshot AI / kvcache-ai. Mooncake repository and technical documentation. https://github.com/kvcache-ai/Mooncake

DeepSeek. V3/R1 inference system overview. https://github.com/deepseek-ai/open-infra-index/blob/main/202502OpenSourceWeek/day_6_one_more_thing_deepseekV3R1_inference_system_overview.md

Qwen. Qwen3 repository and deployment guidance. https://github.com/QwenLM/Qwen3

Umans AI. Open Models Inference for coding. https://app.umans.ai/landing

OCTO Technology. Culture DevOps — volumes 1, 2 et 3. https://publication.octo.com/fr/telechargement-livre-blanc-culture-devops-vol-1

OCTO Technology. Culture Flow. https://publication.octo.com/culture-flow-livre-blanc-octo-technology

À retenir

La formule

Problème réel + tranche de valeur + architecture agent/inférence + évaluations + traces + responsabilité + décision = passage à la production.

Ce document est une version 1.0 destinée à être éprouvée par la pratique, enrichie par les retours de pairs et mise à jour à partir de missions et de benchmarks reproductibles.