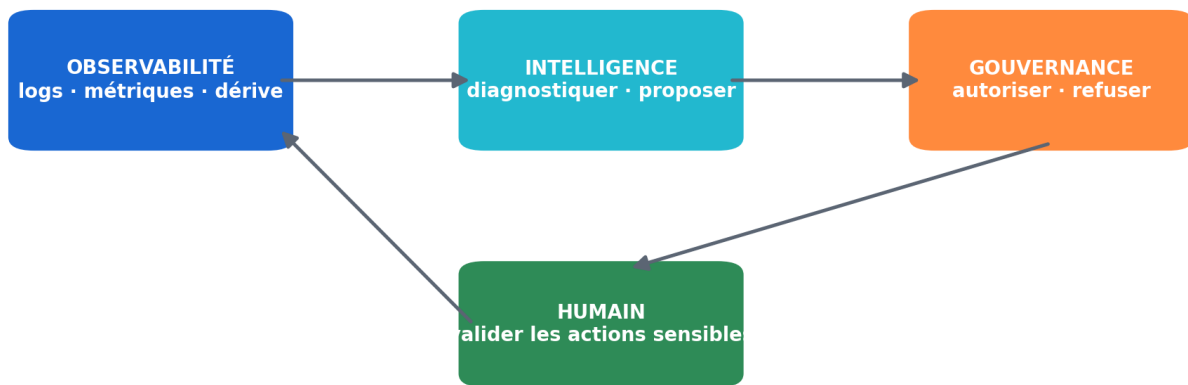


Agentic-Ops

Rendre l'IA utile en exploitation sans lui abandonner le contrôle

Conception d'un plan de contrôle MLOps gouverné, branché sur la plateforme cible CAC_MLOPS

De l'alerte à la décision sûre



Principe directeur : l'IA éclaire ; les règles et l'humain gardent la maîtrise.

Illustration de couverture : la trajectoire d'une alerte vers une décision gouvernée.

Auteur : Noël Ching

Version de soutenance • 20 octobre 2026

Dépôts de référence : github.com/jakatt/cac_mlops et github.com/noll77/agentic-ops

Résumé exécutif

Les modèles de machine learning ne tombent pas toujours en panne de manière visible. Ils peuvent continuer à répondre tout en devenant progressivement moins pertinents, parce que les données changent, qu'un pipeline se dégrade ou qu'un composant d'infrastructure se fragilise. C'est ce décalage entre un service apparemment disponible et une décision devenue moins fiable qui constitue le point de départ du projet.

Agentic-Ops a été conçu comme un plan de contrôle externe à la plateforme MLOps. Sa mission n'est pas de remplacer les outils d'exploitation, ni de donner les pleins pouvoirs à un modèle de langage. Il collecte des signaux, organise l'enquête, formule une proposition, puis soumet cette proposition à des règles déterministes. Les actions sensibles restent placées sous validation humaine.

Le message central peut se résumer ainsi : l'IA propose, le code autorise ou refuse, et l'humain garde la maîtrise des décisions à fort impact. Cette séparation des pouvoirs transforme l'IA en assistant d'investigation plutôt qu'en administrateur autonome.

Contributions principales

- Une architecture séparant clairement le terrain MLOps et le plan de contrôle.
- Une chaîne de traitement explicite : signal, faits, diagnostic, proposition, politique, décision, preuve.
- Une gouvernance fail-closed : en cas de doute ou d'information manquante, le système bloque.
- Un mécanisme human-in-the-loop pour les actions mutatrices.
- Une traçabilité pensée pour expliquer après coup ce qui a été observé, proposé, autorisé et exécuté.
- Une démarche d'honnêteté technique distinguant prototype, simulation, qualification locale et production.

Comment lire ce rapport

Le premier rapport était techniquement riche, mais il décrivait surtout les pièces de la machine. Cette version raconte davantage le problème, la décision d'architecture et le chemin parcouru. Les composants techniques restent présents, mais ils apparaissent lorsqu'ils deviennent utiles à l'histoire.

Partie	Question à laquelle elle répond
Partie I	Pourquoi la supervision MLOps classique ne suffit-elle pas toujours ?
Partie II	Comment Agentic-Ops se branche-t-il sur CAC_MLOPS ?
Partie III	Comment une alerte devient-elle une décision gouvernée ?
Partie IV	Comment la sécurité, l'humain et la preuve encadrent-ils l'action ?
Partie V	Que démontrent les scénarios, et quelles limites restent ouvertes ?

Convention de vérité

Fait vérifiable publiquement : le dépôt CAC_MLOPS se présente comme un point de départ pour des projets MLOps autour des accidents routiers et expose une structure data science classique. Le dépôt Agentic-Ops se présente comme un moteur autonome de politiques MLOps et un runtime de contrôle IA.

Éléments issus du dossier fourni : les métriques, nombres de tests, latences, scénarios et détails d'implémentation sont repris comme résultats déclarés dans l'extraction technique transmise. Ils devront être accompagnés des journaux de test ou reproduits en démonstration lorsqu'ils sont présentés comme mesures.

Positionnement : ce rapport ne revendique ni certification réglementaire, ni qualification de production complète. Il décrit des mécanismes contribuant à l'auditabilité et à la maîtrise du risque.

Sommaire

Prologue - Le jour où le modèle répond encore, mais moins bien	5
1. Le problème : exploiter un modèle vivant	7
2. Deux systèmes complémentaires	11
3. L'architecture d'Agentic-Ops	15
4. L'enquête multi-spécialiste	20
5. Gouverner avant d'exécuter	24
6. Garder l'humain dans la boucle	29
7. Prouver ce qui s'est passé	32
8. Démontrer par les scénarios	35
9. Résultats, limites et honnêteté technique	39
10. Feuille de route et conclusion	43

Prologue — Le modèle répond encore, mais moins bien

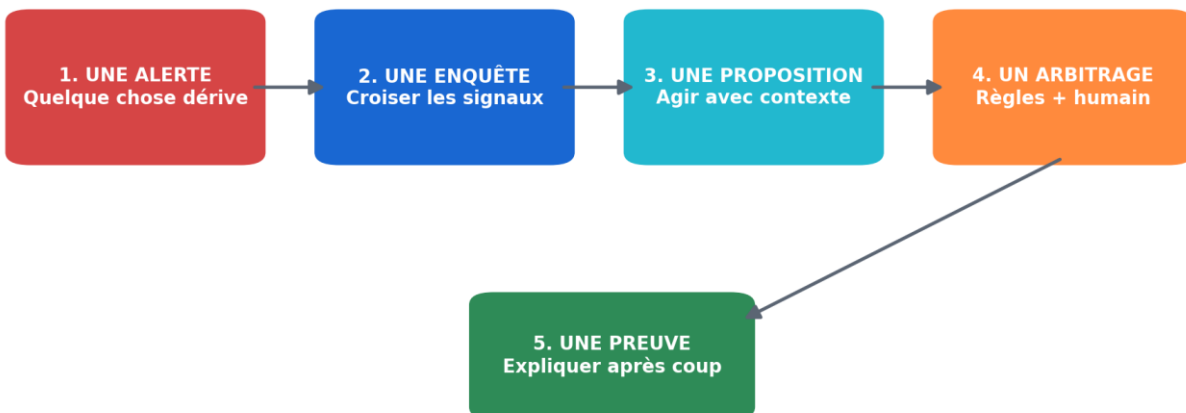
À 8 h 17, aucun serveur n'est déclaré indisponible. Pourtant, quelque chose a changé.

Le modèle de prédiction continue de répondre. L'API renvoie un statut normal, les conteneurs sont actifs et les tableaux de bord ne montrent pas de panne franche. Mais la performance sur une catégorie critique se dégrade. Ce type d'incident est difficile parce qu'il ne ressemble pas à une panne informatique traditionnelle.

L'ingénieur d'astreinte doit répondre à plusieurs questions en même temps : les données d'entrée ont-elles changé ? Le modèle a-t-il vieilli ? Le pipeline a-t-il perdu une partie des enregistrements ? L'infrastructure est-elle réellement en cause ou ne montre-t-elle qu'un symptôme ? Chaque réponse possible conduit à une action différente.

Agentic-Ops naît dans cet espace d'incertitude. Il ne cherche pas à agir plus vite à tout prix. Il cherche d'abord à reconstruire une situation fiable, à séparer les faits des hypothèses, puis à empêcher qu'une action plausible mais dangereuse soit exécutée sans contrôle.

Le fil narratif du projet



La soutenance suit un incident de bout en bout plutôt qu'un catalogue de composants.

Figure 1 — Le fil narratif retenu pour la soutenance.

1.1 — Le MLOps est un système vivant

Un logiciel classique change surtout lorsque son code change. Un système ML peut changer parce que le monde change.

Une application déterministe exécute des règles explicites. Un modèle apprend une relation statistique à partir d'un passé observé. Lorsque les comportements, les conditions météorologiques, les capteurs ou les pratiques de saisie évoluent, le modèle peut rester techniquement disponible tout en devenant moins pertinent.

Cette réalité introduit une double exploitation : surveiller le service informatique et surveiller la validité statistique de ce qu'il produit. La disponibilité, la latence et la mémoire ne suffisent plus. Il faut aussi regarder la distribution des données, la qualité du pipeline, les performances du modèle et les écarts entre entraînement et production.

Le défi de l'exploitation MLOps est donc moins de collecter beaucoup d'indicateurs que de les relier dans une histoire causale utilisable.

- Disponibilité : le service répond-il ?
- Intégrité : les données sont-elles complètes et cohérentes ?
- Pertinence : le modèle reste-t-il adapté au contexte ?
- Gouvernance : une action proposée est-elle autorisée et proportionnée ?

1.2 — Le piège du premier symptôme

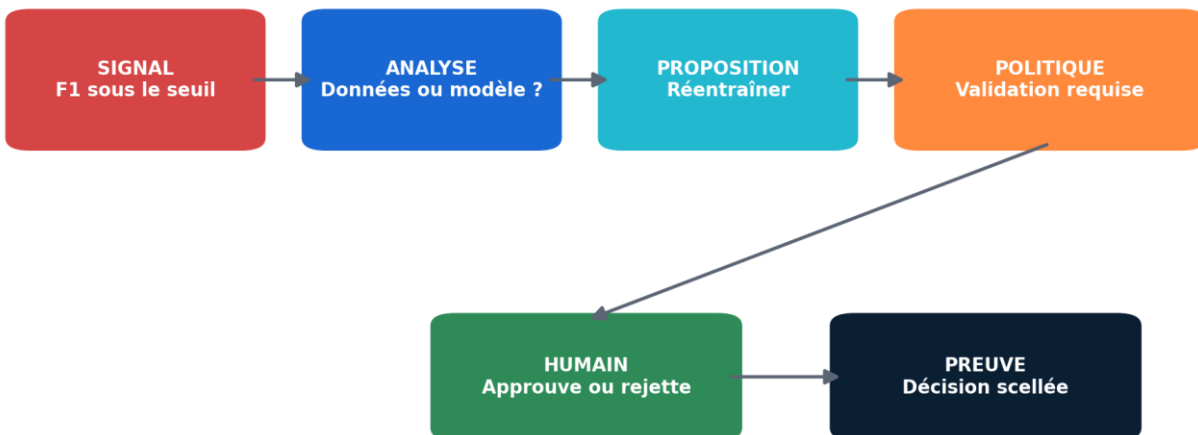
Le premier signal visible n'est pas toujours la cause racine.

Un pic CPU peut être provoqué par une charge légitime, une attaque, un lot de données anormal ou une boucle de calcul. Un échec d'entraînement peut provenir d'un manque de mémoire, mais aussi d'un schéma de données modifié ou d'un volume multiplié accidentellement.

Dans ces situations, le réflexe classique consiste à redémarrer, augmenter la capacité ou relancer le pipeline. Ces gestes peuvent restaurer temporairement le service tout en masquant la cause réelle. Ils peuvent aussi amplifier l'incident, par exemple en réentraînant un modèle sur un jeu de données incomplet.

La première décision d'Agentic-Ops est donc méthodologique : aucune action de remédiation ne doit être évaluée isolément de son contexte amont.

Scénario fil rouge : une dérive du modèle



Le système ne confond pas « action plausible » et « action autorisée ».

Figure 2 — Un scénario fil rouge où réentraîner peut être légitime, prématuré ou dangereux.

1.3 — Pourquoi ne pas donner les clés au LLM ?

Un modèle de langage sait proposer. Il ne fournit pas, à lui seul, une garantie d'autorisation.

Les LLM sont utiles pour synthétiser des signaux hétérogènes, rapprocher des observations et formuler des hypothèses. Mais leur fonctionnement probabiliste les rend inadaptés à la décision finale de sécurité. Une formulation convaincante n'est pas une preuve, et une commande plausible n'est pas nécessairement une commande permise.

Trois risques motivent la séparation des pouvoirs : l'hallucination d'une cible ou d'une action, l'influence de données hostiles présentes dans les logs, et l'impossibilité de garantir qu'une même situation produira toujours le même verdict.

La réponse d'architecture n'est pas de supprimer l'IA. Elle consiste à limiter son mandat : interpréter, expliquer et proposer. La décision d'autoriser reste confiée à des règles explicites, testables et auditable.

Capacité	LLM	Moteur déterministe
Interpréter un contexte	Fort	Limité
Appliquer une règle stable	Non garanti	Fort
Expliquer une hypothèse	Fort	Factuel
Autoriser une mutation sensible	Non	Oui, selon politique

1.4 — La promesse du projet

Réduire l'incertitude sans automatiser l'imprudence.

Agentic-Ops ne promet pas de résoudre seul tous les incidents. Il propose une discipline d'exploitation : collecter passivement, normaliser les faits, confronter plusieurs diagnostics, calculer le risque, appliquer une politique, demander une validation lorsque nécessaire, puis conserver la preuve de la décision.

Cette promesse est volontairement plus sobre qu'un agent autonome capable de tout réparer. Elle est aussi plus crédible pour des environnements où la sécurité, la continuité de service et la responsabilité humaine comptent davantage que la vitesse brute.

La valeur attendue est double : aider l'opérateur à comprendre plus vite et empêcher le système d'agir au-delà de son mandat.

- Comprendre avant d'agir.
- Séparer proposition et autorisation.
- Refuser par défaut quand une information critique manque.
- Conserver une trace exploitable par un humain.

2.1 — CAC_MLOPS : le terrain métier

La plateforme cible porte le cycle de vie du modèle ; Agentic-Ops se branche au-dessus d'elle.

Le dépôt public CAC_MLOPS est présenté comme un socle de projet autour de la prédiction d'accidents routiers. Il organise les données brutes, les transformations, les fonctionnalités, l'entraînement, la prédiction, les modèles et les rapports. Cette structure constitue le terrain métier sur lequel les signaux de qualité et d'exploitation prennent sens.

Le dossier technique fourni décrit une cible enrichie par des services d'observabilité et d'orchestration. Ces services sont traités dans le rapport comme l'architecture de démonstration annoncée, à distinguer de ce qui est directement visible dans la page publique du dépôt.

Le point important pour la maintenance n'est pas le nom des outils, mais leur rôle : produire des traces permettant de savoir ce qui s'est passé dans les données, le modèle, les workflows et l'infrastructure.

Couche	Responsabilité
Données	Importer, transformer et préparer
Modèle	Entraîner, versionner et prédire
Orchestration	Planifier et suivre les traitements
Observabilité	Exposer logs, métriques et dérive

2.2 — Agentic-Ops : le plan de contrôle

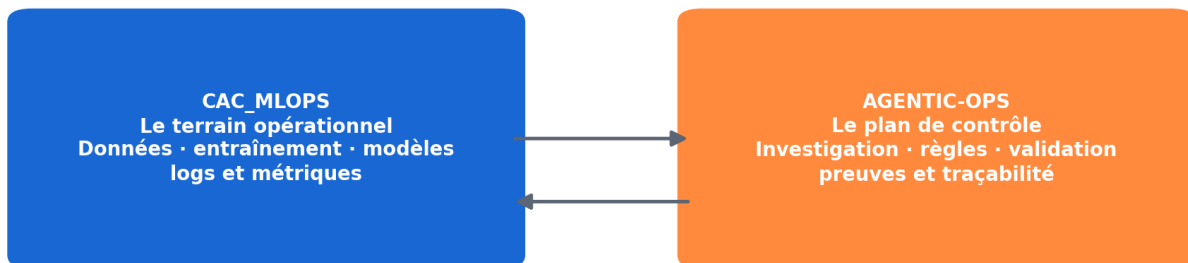
Agentic-Ops n'est pas un second pipeline ML. Il est le système qui questionne, encadre et documente.

Le dépôt public Agentic-Ops se présente comme un moteur de politiques MLOps et un runtime de contrôle IA. Son positionnement complète la plateforme cible : il ne crée pas lui-même la vérité métier, mais transforme les signaux disponibles en une décision gouvernée.

Cette séparation évite un couplage excessif. CAC_MLOPS peut faire évoluer ses méthodes d'entraînement ; Agentic-Ops doit surtout conserver des contrats stables pour recevoir des faits, diagnostiquer et évaluer une proposition.

Le système est donc pensé comme un plan de contrôle externe : une position utile pour observer sans devenir un nouveau point unique de défaillance dans la chaîne de production.

Deux projets, deux responsabilités complémentaires



La plateforme cible produit et expose des signaux. Agentic-Ops les interprète sans devenir propriétaire du modèle métier.

Figure 3 — CAC_MLOPS et Agentic-Ops : un terrain opérationnel et un plan de contrôle.

2.3 — Une connexion d’abord passive

Avant de savoir agir, le système doit savoir regarder sans nuire.

Le connecteur décrit dans le dossier technique adopte un principe de collecte en lecture seule pour les cycles réguliers de supervision. Les journaux, métriques, états de workflows, informations de registre de modèles et rapports de dérive sont interrogés afin de produire un instantané cohérent.

Ce choix réduit le rayon d’impact d’un défaut du collecteur. Une boucle défaillante peut générer une charge de lecture, mais elle ne doit pas pouvoir modifier directement un modèle, relancer un entraînement ou redimensionner un service.

La séparation entre observer et agir devient ainsi visible jusque dans les protocoles : les flux de collecte sont distincts des éventuels drivers d’exécution, eux-mêmes gardés par des politiques et des autorisations.

- Collecte régulière : lecture seule.
- Actions passives : investigation, ticket ou rapport.
- Actions mutatrices : voie séparée, gouvernée et désactivable.
- Coupure distante : état explicite plutôt que crash en cascade.

2.4 — La connectivité n'est pas la qualification

Un tunnel chiffré permet de joindre une cible. Il ne prouve pas que l'ensemble est prêt pour la production.

Le dossier décrit un lien inter-cloud fondé sur Tailscale et WireGuard, avec un fonctionnement userspace adapté à un environnement conteneurisé. Cette solution répond au besoin de ne pas exposer directement les interfaces d'administration sur Internet.

Pour autant, la présence d'un tunnel ne valide ni les droits, ni la disponibilité, ni la supervision de bout en bout. La qualification doit encore couvrir la rotation des secrets, la perte de connectivité, les délais d'expiration, la saturation du proxy, la reprise après redémarrage et la conformité des routes autorisées.

Cette distinction est importante dans une soutenance : montrer une connexion réussie démontre une faisabilité. Revendiquer une exploitation qualifiée exige des preuves répétables.

Démonstration	Qualification
Un nœud rejoint le maillage	Identité et secrets gérés dans la durée
Une requête atteint la cible	SLO, timeouts et limites de débit définis
Une coupure est absorbée	Tests de reprise et procédures documentées
Le trafic est chiffré	Règles d'accès et audit validés

3.1 — Voir l'ensemble sans noyer l'opérateur

L'architecture simplifie la décision en séparant les responsabilités.

Le système reçoit des signaux issus de plusieurs sources. Ces signaux sont convertis en faits structurés, puis confiés à des spécialistes. Une proposition d'action est ensuite évaluée par le moteur de politiques. Selon le verdict, elle est refusée, simulée, autorisée ou suspendue dans l'attente d'une validation humaine.

Cette chaîne n'est pas une succession de microservices pour le plaisir de distribuer le système. Chaque frontière matérialise une responsabilité et réduit une classe de risque : le collecteur ne décide pas, le spécialiste n'autorise pas, l'exécuteur n'invente pas, et le registre ne modifie pas l'histoire.

La lisibilité de l'architecture devient alors un mécanisme de sûreté.

Architecture fonctionnelle simplifiée

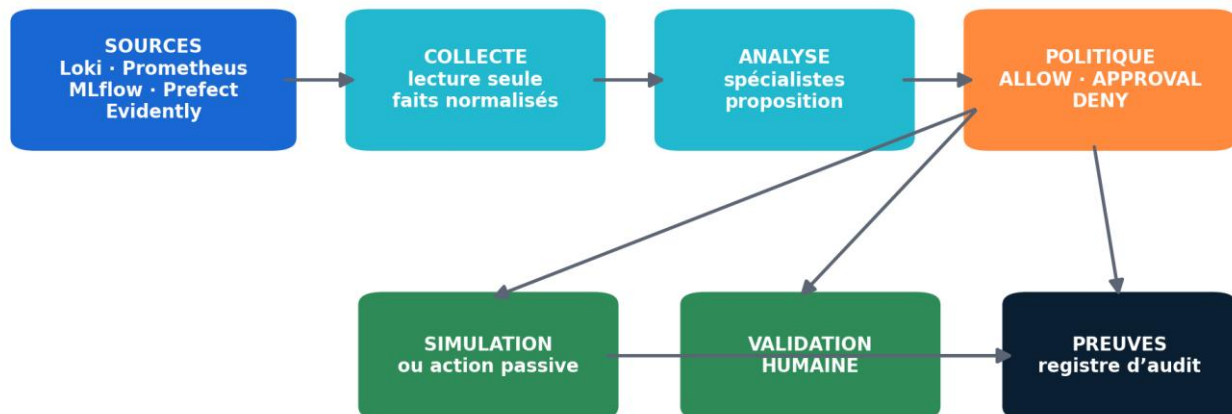


Schéma conceptuel. Les capacités réelles dépendent de la configuration, des droits et de la qualification de la cible.

Figure 4 — Architecture fonctionnelle simplifiée d'Agentic-Ops.

3.2 — Des contrats plutôt que des phrases libres

Une proposition exploitable doit être plus précise qu'un conseil en langage naturel.

Entre l'espace cognitif et l'espace déterministe, Agentic-Ops utilise des objets structurés : signal, diagnostic, proposition d'action, décision, résultat d'exécution et épisode. Le typage impose des champs attendus, des valeurs autorisées et une forme commune pour tous les composants.

Cette approche limite les ambiguïtés. Dire « il faudrait peut-être redémarrer » ne suffit pas. Le système doit préciser l'action, la cible, le motif, la réversibilité déclarée et le rayon d'impact attendu. Le moteur de politiques peut ensuite comparer ces déclarations au catalogue autorisé et à l'inventaire.

Le contrat est donc le passage obligé entre une hypothèse produite par l'IA et une décision mécanisable.

Objet	Question
Signal	Qu'a-t-on observé et quand ?
Diagnostic	Quelle couche semble en cause et avec quelle confiance ?
Proposition	Quelle action, sur quelle cible, pour quelle raison ?
Décision	Quelle règle autorise, suspend ou refuse ?
Épisode	Comment reconstituer toute l'histoire ?

3.3 — L’orchestrateur comme chef de circulation

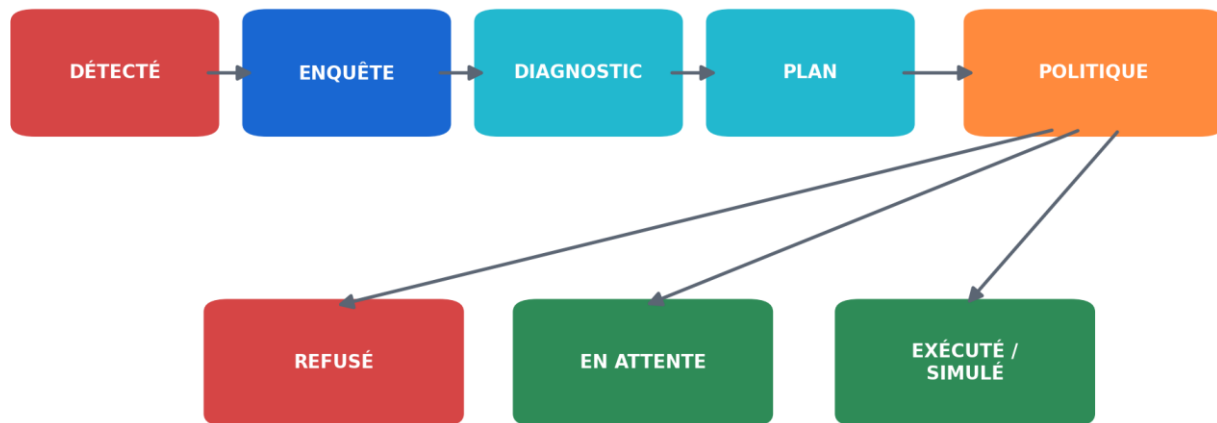
Il ne remplace pas les spécialistes ; il garantit l’ordre des étapes.

L’orchestrateur pilote le cycle de vie d’un incident. Il s’assure qu’un épisode est détecté, investigué, diagnostiqué et planifié avant d’être soumis à la politique. Il gère aussi les chemins de refus, d’attente et de reprise après approbation.

Cette machine à états empêche deux erreurs fréquentes : exécuter une action avant que le diagnostic soit stabilisé, et reprendre un incident sans vérifier que la proposition approuvée est toujours la même.

Dans l’architecture fournie, l’idempotence et les verrous participent également à la maîtrise des doublons. L’objectif est d’éviter que plusieurs workers traitent le même incident comme s’il s’agissait de décisions indépendantes.

Un incident avance par étapes contrôlées



Aucun saut d'étape : en cas d'information manquante ou de règle violée, le parcours se ferme de manière sûre.

Figure 5 — Cycle de vie déterministe d'un épisode.

3.4 — Le mode dégradé comme comportement normal

Une architecture distribuée doit savoir dire : la cible est indisponible, je n'agis pas.

Lorsque la plateforme distante ne répond plus, le comportement attendu n'est pas de transformer une absence de données en diagnostic certain. Le collecteur doit signaler l'indisponibilité, préserver le fonctionnement local utile et empêcher toute remédiation nécessitant des faits récents.

Ce choix exprime le principe fail-closed : si l'état de la cible ne peut pas être confirmé, l'autorisation se ferme. C'est une règle plus conservatrice, mais plus cohérente pour une solution qui intervient potentiellement sur des composants de production.

Le mode dégradé inclut aussi la possibilité de fournir une documentation ou un diagnostic limité à partir de connaissances locales, sans prétendre remplacer la télémétrie absente.

- Signaler clairement l'indisponibilité.
- Ne pas fabriquer de faits manquants.
- Maintenir les fonctions locales non dangereuses.
- Bloquer les mutations dépendantes d'un état non vérifié.

3.5 — Le RAG déterministe : un choix de contrainte

Dans ce projet, retrouver exactement une fiche de risque compte parfois plus que rechercher une proximité sémantique.

Le dossier décrit un moteur de recherche local indexant une taxonomie MLOps et des scénarios. Il privilégie les identifiants exacts, les filtres et les tables de hachage plutôt qu'un stockage vectoriel lourd.

Ce choix répond à trois contraintes annoncées : une faible enveloppe mémoire, un démarrage rapide et la nécessité de retrouver sans ambiguïté une référence telle que DATA-042. Il fournit aussi un repli lorsque les fournisseurs LLM sont indisponibles.

Il faut néanmoins préserver la nuance : ce composant est un moteur de recherche structuré complété par une réponse hors ligne. Il ne possède pas la souplesse d'une recherche sémantique générale et dépend fortement de la qualité des fiches préparées.

Approche	Atout	Limite
Index déterministe	Exact, léger, reproductible	Moins souple lexicalement
Recherche vectorielle	Rappel sémantique	Coût, mémoire, faux voisins
LLM seul	Synthèse riche	Dépendance, variabilité, hallucination

4.1 — Plusieurs spécialistes, une seule situation

Les incidents MLOps traversent souvent les frontières entre données, modèle et infrastructure.

Agentic-Ops distingue des rôles cognitifs spécialisés. Le spécialiste données cherche les ruptures de schéma, les pertes de volume et les dérives. Le spécialiste modèle examine les performances et les versions. Le spécialiste infrastructure s'intéresse aux ressources, aux erreurs d'exécution et aux services.

Cette spécialisation évite qu'un unique agent applique toujours le même raisonnement. Elle ne crée toutefois pas trois vérités indépendantes. Les diagnostics doivent être réunis par un rôle de remédiation qui recherche la cause dominante, les contradictions et les informations manquantes.

Le bénéfice recherché est une enquête contradictoire : chaque angle enrichit la situation, mais aucun spécialiste n'obtient seul le pouvoir d'agir.

Rôle	Question dominante
Données	Les entrées restent-elles fiables ?
Modèle	La performance et la version sont-elles cohérentes ?
Infrastructure	Le service dispose-t-il des ressources nécessaires ?
Remédiation	Quelle proposition respecte le contexte global ?

4.2 — Ne pas confondre confiance et vérité

Un score de confiance décrit une estimation ; il ne transforme pas une hypothèse en fait.

Chaque spécialiste peut produire un diagnostic, un ensemble d'observations et un niveau de confiance. Pour rester utile, le rapport doit présenter ce score comme une information de décision, non comme une certitude mathématique universelle.

La politique ne devrait donc pas autoriser une action sensible sur la seule base d'un score élevé. Elle doit vérifier la cible, le catalogue d'actions, le niveau d'autonomie, l'intégrité des faits et les risques connus.

Cette distinction organise le langage de soutenance : les faits viennent de la télémétrie ; l'interprétation vient de l'analyse ; l'autorisation vient de la politique ; la responsabilité finale peut rester humaine.

- Fait : une métrique mesurée ou un état lu.
- Interprétation : une cause possible reliée aux faits.
- Hypothèse : une piste encore non confirmée.
- Décision : un verdict gouverné et traçable.

4.3 — Résister aux pannes des fournisseurs LLM

La supervision ne doit pas s'arrêter parce qu'un modèle externe renvoie une erreur de quota.

Le registre multi-fournisseurs décrit dans le dossier vise à sélectionner un modèle selon le rôle, puis à basculer vers des alternatives lorsque le fournisseur principal est indisponible. Cette stratégie protège la continuité de l'analyse cognitive.

Cependant, multiplier les modèles multiplie aussi les comportements, les coûts potentiels, les conditions d'usage et les surfaces de configuration. Une chaîne de repli doit donc être observée elle aussi : quel modèle a répondu, avec quelle latence, quelle version et quel budget ?

Le dernier niveau de repli reste local et déterministe. Il ne cherche pas à simuler un raisonnement complet, mais à fournir des fiches de référence et à maintenir une capacité minimale d'assistance.

- Tracer le fournisseur réellement utilisé.
- Masquer les secrets dans les journaux.
- Définir un plafond de coût et de latence.
- Tester les erreurs 429, 500 et 503.
- Prévoir un fonctionnement utile sans LLM.

4.4 — Protéger le raisonnement des données hostiles

Un log est une donnée observée, jamais une autorité.

Les journaux et runbooks peuvent contenir des chaînes ressemblant à des instructions. Un attaquant peut exploiter cette proximité pour tenter d'influencer un modèle de langage. Le dossier prévoit une mise en quarantaine des contenus non fiables, un nettoyage des caractères de contrôle et une limitation de taille.

Cette protection réduit le risque d'interprétation abusive, mais elle ne doit pas être présentée comme une défense absolue. La barrière la plus robuste reste structurelle : le LLM n'a pas le droit d'autoriser l'action qu'une donnée hostile lui suggère.

Autrement dit, la sécurité ne dépend pas seulement d'un bon prompt. Elle dépend surtout de l'absence de chemin direct entre le texte observé et l'exécution.

- Données externes étiquetées comme non fiables.
- Sortie de l'IA limitée à un contrat.
- Cibles vérifiées dans un inventaire.
- Actions comparées à un catalogue.
- Décision finale indépendante du texte des logs.

5.1 — Le moteur de politiques : la constitution du système

Là où l'IA formule une possibilité, la politique formule une limite.

Le moteur détermine si une proposition appartient au catalogue, si sa cible est autorisée, si son rayon d'impact est acceptable et si le contexte impose une validation humaine. Il peut conclure par trois verdicts : autoriser, demander une approbation ou refuser.

Le caractère déterministe signifie qu'une même proposition, évaluée avec le même contexte et la même version de politique, doit produire le même verdict. Cette reproductibilité est essentielle pour tester, auditer et expliquer.

La politique est déclarative afin de rendre visibles les règles d'autonomie, le catalogue d'actions et le registre de risques. Mais cette lisibilité crée aussi une obligation : gérer les versions et prouver quelles règles étaient actives au moment de la décision.

Verdict	Sens
ALLOW	Action autorisée dans le cadre défini
REQUIRE_APPROVAL	Action suspendue jusqu'à décision humaine
DENY	Action interdite ou contexte insuffisant

5.2 — Le fail-closed, une règle de prudence

Quand une condition de sécurité ne peut pas être vérifiée, le système refuse plutôt qu'il n'improvise.

Le fonctionnement fail-closed concerne plusieurs situations : une cible absente de l'inventaire, une action inconnue, un rayon d'impact excessif, des faits incomplets, un verrou indisponible ou une politique incohérente.

Cette posture réduit le risque d'une action non maîtrisée, mais augmente le nombre de situations nécessitant une intervention humaine. C'est un compromis assumé. Dans un plan de contrôle, une indisponibilité temporaire de l'automatisation est généralement préférable à une mutation irréversible mal fondée.

La maturité du projet se mesurera donc aussi au taux de refus utiles et au taux de faux blocages, pas seulement au nombre d'actions automatisées.

- Sécurité : refuser l'inconnu.
- Exploitation : expliquer clairement le motif.
- Amélioration continue : analyser les refus récurrents.
- Gouvernance : faire évoluer la politique de manière contrôlée.

5.3 — Le veto amont

Une anomalie de données peut rendre dangereuse une action pourtant raisonnable sur l'infrastructure.

Le veto amont cherche à empêcher un redémarrage ou un redimensionnement de service de masquer une cause de données plus importante. Si la couche amont présente une anomalie de sévérité comparable ou supérieure, les mutations de service peuvent être bloquées.

Cette règle traduit une connaissance opérationnelle en décision explicite. Elle illustre la valeur du projet : transformer un principe d'expert, souvent appliqué intuitivement, en contrôle reproductible.

Comme toute règle causale, elle doit être confrontée à des incidents réels. Une règle trop stricte peut retarder une action nécessaire ; une règle trop permissive peut laisser passer une remédiation superficielle. Les scénarios et GameDays servent précisément à ajuster ce seuil.

Situation	Réponse prudente
Perte massive de données + OOM	Bloquer le redémarrage automatique
Fuite mémoire confirmée, données saines	Demander validation pour redémarrer
Cible inconnue	Refuser
Simple investigation	Autoriser si non mutatrice

5.4 — Inventaire, catalogue et rayon d'impact

Trois barrières empêchent l'action imaginaire : la cible doit exister, l'action doit être connue, l'impact doit rester borné.

L'inventaire local protège contre les ressources inventées ou mal orthographiées. Le catalogue limite les verbes disponibles. Le rayon d'impact mesure l'étendue prévisible de la modification.

Cette trilogie est efficace à condition que l'inventaire reste à jour. Le dossier reconnaît ici une limite importante : un fichier statique n'est pas une source autoritaire synchronisée. En production, la dérive entre l'inventaire local et la réalité peut provoquer des refus injustifiés ou, plus grave, conserver une cible devenue obsolète.

La feuille de route doit donc relier l'inventaire à des sources d'autorité tout en conservant une validation avant activation de nouvelles cibles.

- Action connue.
- Cible existante et approuvée.
- Paramètres conformes.
- Impact maximal respecté.
- Quota d'actions non dépassé.

5.5 — Les niveaux d'autonomie

Toutes les actions ne méritent pas le même degré de liberté.

Les actions passives, comme créer un ticket ou approfondir une investigation, peuvent être autorisées avec un niveau d'autonomie faible si elles ne modifient pas la cible. Les actions mutatrices, comme redémarrer, redimensionner, réentraîner ou promouvoir un modèle, nécessitent une validation plus forte.

Ce classement doit dépendre du contexte et non seulement du verbe. Créer une issue peut être peu risqué, mais une tempête de tickets peut tout de même perturber l'exploitation. De même, un redémarrage en bac à sable n'a pas le même impact qu'en production.

L'autonomie est donc une combinaison : nature de l'action, environnement, cible, rayon d'impact, réversibilité et qualité des faits disponibles.

Niveau	Exemples	Contrôle
L1	Investigation, ticket	Autorisation encadrée
L2	Redémarrage limité, réentraînement	Approbation humaine
L3	Promotion, rollback, mise à l'arrêt	Quorum ou procédure renforcée

6.1 — Pourquoi l’humain reste indispensable

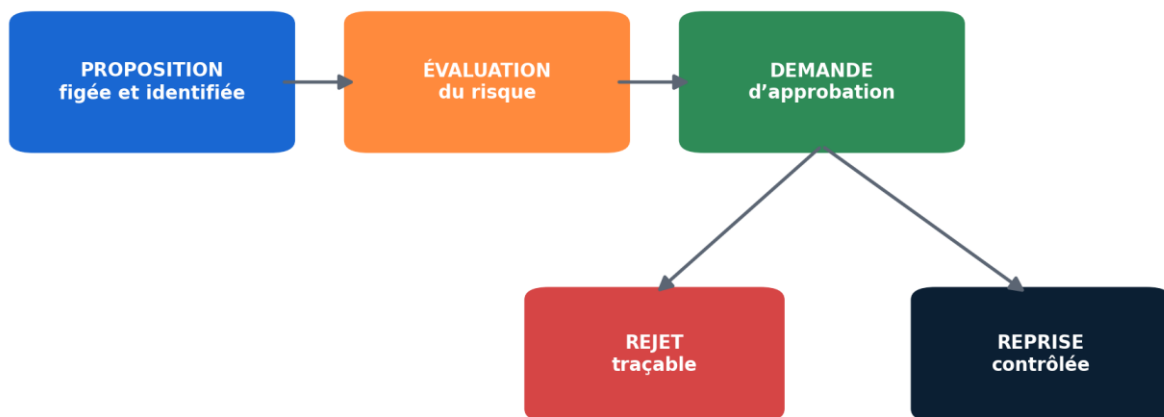
Le contrôle humain n’est pas un aveu d’échec de l’IA ; c’est une décision d’architecture.

Lorsqu’une action peut interrompre un service, modifier un modèle ou toucher plusieurs composants, le système suspend l’épisode. L’opérateur reçoit le diagnostic, les faits, l’action exacte, la cible et le motif de la politique.

L’humain peut alors approuver ou rejeter avec un contexte supérieur à celui du système : calendrier métier, incident en cours, changement non encore inventorié, risque juridique ou priorité opérationnelle.

Pour être réel, ce contrôle doit laisser le temps de comprendre, empêcher l’auto-approbation et garantir que la proposition approuvée n’a pas changé entre la demande et l’exécution.

Human-in-the-loop : ralentir au bon moment



L’approbation humaine n’est pas un bouton décoratif : elle sépare l’analyse de l’autorisation.

Figure 6 — L’approbation humaine comme frontière entre proposition et action.

6.2 — Sécuriser l’approbation

Un bouton d’approbation est une surface de sécurité à part entière.

Le dossier décrit une intégration Slack dans laquelle les requêtes interactives sont vérifiées par signature HMAC et limitées par une fenêtre temporelle. La proposition est figée et identifiée avant d’être présentée à l’opérateur.

Ces mécanismes visent à empêcher trois attaques : fabriquer une fausse approbation, rejouer une ancienne approbation et substituer une action différente après validation.

Une mise en production devra compléter ce dispositif par la gestion des identités, des rôles, du quorum, de la révocation et des journaux d’accès. La qualité du contrôle humain dépend autant de l’interface que du protocole cryptographique.

- Authentifier la source de la décision.
- Vérifier la fraîcheur du message.
- Lier l’approbation à une proposition immuable.
- Empêcher l’auteur de s’auto-approuver si la politique l’interdit.
- Tracer le décideur et le résultat.

6.3 — Simulation avant exécution réelle

La soutenance doit montrer une chaîne crédible sans transformer la démonstration en prise de risque.

Le mode simulation est activé par défaut dans l'exécuteur décrit. Les propositions suivent le parcours complet, mais le résultat d'exécution reste local et simulé tant que les préconditions matérielles et les secrets nécessaires ne sont pas réunis.

Cette barrière protège la cible pendant les tests et la soutenance. Elle permet d'observer les décisions du moteur, les refus et les demandes d'approbation sans envoyer de mutation réelle.

La limite doit être dite clairement : un driver présent dans le code n'est pas un driver qualifié en production. Le passage au réel nécessite des tests contrôlés, des environnements dédiés et des preuves de reprise.

Étape	Preuve attendue
Simulation	Scénario reproductible et résultat traçable
Bac à sable	Action réelle sur cible non critique
Préproduction	GameDays, rollback et supervision
Production	Autorisation formelle, SLO et astreinte

7.1 — La preuve comme produit du système

Une bonne décision ne suffit pas. Il faut pouvoir expliquer comment elle a été prise.

Chaque épisode agrège les faits, les diagnostics, la proposition, le verdict de politique, l'approbation éventuelle et le résultat. Le registre d'évidence conserve cette histoire dans l'ordre.

L'objectif n'est pas de rendre le système mystérieusement inviolable. Il est de rendre les altérations détectables et les décisions reconstituables. Une chaîne de hachage relie les enregistrements ; une structure de Merkle permet de prouver l'inclusion d'un épisode sans transmettre tout le registre.

Pour un jury non spécialiste, l'image utile est celle d'un cahier dont chaque page porte l'empreinte de la précédente : modifier une page ancienne casse la continuité visible.

- Qui a observé quoi ?
- Quelle hypothèse a été formulée ?
- Quelle règle a été appliquée ?
- Qui a approuvé ?
- Quelle action a réellement eu lieu ?

7.2 — WORM et Merkle sans jargon inutile

Deux mécanismes répondent à deux besoins : conserver l'ordre et prouver l'appartenance.

Le registre append-only évite de réécrire silencieusement le passé. Chaque bloc contient une référence cryptographique au précédent. L'arbre de Merkle produit ensuite une racine représentant un ensemble d'épisodes et une preuve courte pour un épisode donné.

Ces mécanismes renforcent l'auditabilité, mais leur portée dépend de la protection des clés, du stockage, de l'horodatage et des procédures de sauvegarde. Ils ne transforment pas à eux seuls un fichier local en archive réglementaire certifiée.

La présentation honnête consiste donc à parler de mécanismes de détection d'altération et de preuve d'inclusion, pas d'immuabilité absolue indépendante de l'infrastructure.

Mécanisme	Réponse
Chaîne de hachage	Le passé a-t-il été modifié ou réordonné ?
Signature	Le bloc a-t-il été produit avec la clé attendue ?
Racine de Merkle	Quel est le résumé de l'ensemble ?
Preuve d'inclusion	Cet épisode appartient-il à cet ensemble ?

7.3 — Sauvegarder l'état, pas seulement le code

Après un sinistre, il faut retrouver les approbations, les quotas et les preuves, pas uniquement redéployer l'application.

Le dossier décrit des instantanés SQLite cohérents, une vérification d'intégrité et un manifeste de hachages. Les bases de runtime et le répertoire d'évidence sont sauvegardés comme un ensemble, puis vérifiés avant restauration.

Cette approche traite un risque souvent oublié dans les prototypes : une application peut redémarrer alors que l'histoire des décisions a disparu ou que des approbations en attente ont été perdues.

Les objectifs RPO et RTO mentionnés dans le dossier doivent être présentés comme des cibles déclarées tant qu'ils ne sont pas démontrés par un test chronométré et documenté sur l'environnement réel.

- Sauvegarde cohérente à chaud.
- Vérification des bases.
- Manifeste des empreintes.
- Contrôle de la chaîne d'évidence.
- Restauration atomique testée.

8.1 — Démontrer avec un incident, pas avec un inventaire

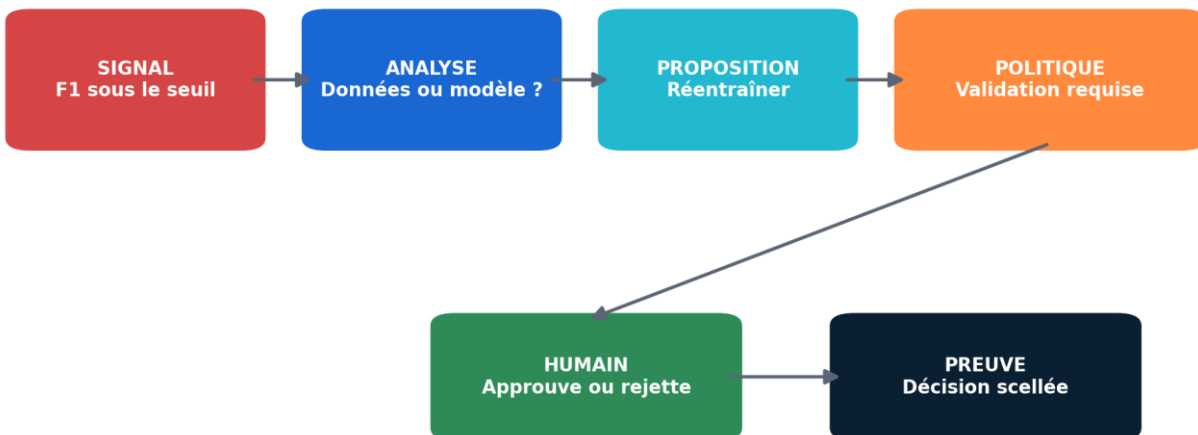
Le jury retiendra mieux une décision difficile qu’une liste de 40 classes Python.

La démonstration recommandée suit une dérive de performance. Une métrique franchit un seuil ; les spécialistes cherchent si la cause vient des données, du modèle ou de l’infrastructure ; le système propose un réentraînement ; la politique exige une validation ; l’opérateur approuve ou rejette ; l’épisode est scellé.

Ce scénario révèle toutes les idées essentielles sans parcourir chaque fichier : multi-source, diagnostic, règle déterministe, human-in-the-loop, simulation et preuve.

Les composants plus avancés, notamment la passerelle MCP ou les différents fournisseurs LLM, deviennent alors des extensions du parcours principal plutôt que de nouveaux sujets isolés.

Scénario fil rouge : une dérive du modèle



Le système ne confond pas « action plausible » et « action autorisée ».

Figure 7 — Scénario de démonstration conseillé pour la soutenance.

8.2 — Les scénarios contradictoires

Un système de gouvernance se juge surtout sur ce qu’il refuse.

Le dossier présente des scénarios où une action apparemment logique doit être bloquée : réentraîner sur un jeu tronqué, revenir vers une version incompatible, redémarrer face à une attaque, accepter une cible injectée ou suivre une instruction cachée dans un log.

Ces tests contradictoires démontrent la différence entre intelligence et gouvernance. Le LLM peut proposer une action dangereuse ; la réussite du système consiste alors à l’empêcher d’atteindre l’exécuteur.

Pour soutenir les résultats quantitatifs annoncés, la démonstration doit conserver les sorties de test, la version du code, les données de scénario et la version du bundle de politiques.

Scénario	Piège	Barrière
Données tronquées	Réentraîner trop vite	Veto amont
Rollback incompatible	Revenir à tort	Registre de risques
Cible injectée	Agir sur l’inconnu	Inventaire
Log hostile	Obéir au texte	Quarantaine + politique

8.3 — La passerelle MCP : ouvrir sans contourner

L'interopérabilité est utile uniquement si elle respecte les mêmes règles que l'interface principale.

La passerelle MCP décrite expose des outils de diagnostic, d'évaluation de politique et d'inspection des preuves. L'intérêt est de permettre à un IDE ou à un autre agent d'interroger le plan de contrôle à travers une interface standardisée.

Le risque serait de créer une entrée secondaire moins protégée. Les outils MCP doivent donc appliquer les mêmes contrats, les mêmes contrôles d'identité et les mêmes barrières d'exécution que le reste du système.

Dans la soutenance, MCP doit être présenté comme une capacité d'intégration. Il ne constitue pas le cœur de la contribution, qui reste la séparation entre raisonnement, politique et exécution.

- Pas de raccourci autour du PolicyEngine.
- Métadonnées explicites sur les outils destructifs.
- Authentification des routes distantes.
- Validation des schémas d'entrée.
- Journalisation des appels.

8.4 — Ce que les tests doivent prouver

Un grand nombre de tests impressionne ; une matrice de risques couverts convainc.

Le dossier annonce une suite de tests couvrant collecteurs, registre LLM, politiques, sécurité, scénarios et preuves cryptographiques. Pour le jury, il faut relier ces tests aux invariants du système.

Les invariants les plus importants sont : aucune action mutatrice ne contourne la politique ; une cible inconnue est refusée ; une approbation expirée est rejetée ; une altération du registre est détectée ; une panne distante ne provoque pas une action aveugle.

La preuve la plus forte est reproductible : commande de test, version exacte, sortie conservée et scénario identifiable.

Invariant	Famille de test
Aucun contournement de politique	Tests d'orchestration et d'exécuteur
Aucune cible inconnue	Tests d'inventaire
Approbation authentique	Tests Slack et identité
Preuve altérée détectée	Tests WORM et Merkle
Panne absorbée	Tests de collecteur et mode offline

9.1 — Résultats annoncés et discipline de preuve

Les chiffres sont utiles lorsqu'ils sont reliés à un protocole reproductible.

L'extraction fournie annonce notamment une suite de 231 tests, dix scénarios principaux, aucun passage d'action interdite dans les scénarios concernés et une faible latence d'arbitrage déterministe. Ces résultats donnent une image encourageante de la couverture locale.

Ils doivent toutefois être présentés avec leur contexte : environnement matériel, version du dépôt, configuration, jeux de test et date d'exécution. Sans ces éléments, ils restent des chiffres déclarés plutôt que des mesures auditable.

Le rapport retient donc une formulation prudente : les tests locaux fournis soutiennent la conception ; ils ne remplacent pas une qualification de production, des tests de charge ou des exercices de reprise.

- Conserver le rapport pytest complet.
- Identifier le commit testé.
- Séparer tests simulés et appels réels.
- Documenter les conditions de mesure.
- Ne pas extrapoler vers la production sans GameDay.

9.2 — Ce qui paraît solide

La force du projet réside moins dans un algorithme isolé que dans l'assemblage de garde-fous cohérents.

La séparation proposition-autorisation est claire. Les contrats structurent les échanges. Le moteur de politiques porte la décision. Le mode simulation limite le risque de démonstration. Le registre d'évidence organise la reconstitution. Enfin, les scénarios contradictoires testent les chemins de refus.

Ces éléments forment un socle crédible pour poursuivre l'industrialisation. Ils donnent également une identité forte au projet : Agentic-Ops n'est pas seulement un agent qui analyse des alertes, c'est un système qui délimite ce qu'un agent a le droit de faire.

Cette cohérence architecturale constitue le principal résultat de la soutenance.

- Architecture lisible.
- Garde-fous indépendants du LLM.
- Séparation entre observation et mutation.
- Chemins de refus explicites.
- Traçabilité intégrée au cycle de vie.

9.3 — Ce qui reste un prototype

La maturité technique exige de nommer les frontières non encore franchies.

Le dossier identifie plusieurs limites : stockage local mono-nœud, inventaire statique, haute disponibilité non démontrée et actions mutatrices conservées en simulation par défaut. Ces limites ne diminuent pas la valeur du prototype ; elles définissent le travail restant.

D'autres points méritent une qualification : comportement sous charge, disponibilité du maillage, rotation des clés, reprise après panne, gestion des versions de politiques, dérive des coûts LLM et synchronisation de l'identité des approubateurs.

La bonne posture consiste à ne pas qualifier de 'prêt pour la production' un composant simplement bien testé localement. La production est une propriété de l'ensemble déployé, opéré et surveillé.

Acquis	À qualifier
Moteur de règles local	Comportement distribué et charge
Chaîne d'évidence	Stockage durable et gouvernance des clés
Connecteur lecture seule	SLO réseau et reprise
Drivers présents	GameDays et rollback réel
Inventaire local	Synchronisation autoritative

9.4 — Ce que le projet ne doit pas revendiquer

La crédibilité augmente lorsque la conclusion reste plus précise que spectaculaire.

Les mécanismes d’audit et de contrôle contribuent à répondre à des exigences de traçabilité, mais ils ne constituent pas une certification de conformité à l’AI Act. De même, l’usage de HMAC, de SHA-256 ou de Merkle ne suffit pas à certifier une archive réglementaire.

Le système ne garantit pas non plus qu’un diagnostic produit par un LLM est vrai. Il garantit surtout que ce diagnostic ne devient pas automatiquement une autorisation d’agir.

Enfin, l’architecture ne prouve pas une portabilité universelle vers toute stack MLOps. Les adaptateurs rendent cette portabilité plausible, mais chaque nouvelle cible demande des contrats, des droits, des tests et une qualification.

- Dire « contribue à l’auditabilité », pas « certifié conforme ».
- Dire « testé localement », pas « industrialisé » sans preuve.
- Dire « connecteur d’exemple », pas « plug-and-play universel ».
- Dire « action simulée », pas « remédiation production » si elle n’a pas eu lieu.

10.1 — Feuille de route d'industrialisation

La priorité n'est pas d'ajouter davantage d'agents. Elle est de qualifier la chaîne existante.

La première étape consiste à figer un scénario de référence reproductible et à publier les preuves de test. La deuxième vise à durcir l'identité, les secrets et le stockage. La troisième synchronise l'inventaire et formalise les contrats avec la cible. La quatrième organise des GameDays et mesure les SLO de bout en bout.

Cette progression réduit le risque de construire une façade riche sur un socle opérationnel encore fragile. Elle prépare également une discussion plus concrète avec les responsables d'infrastructure et de sécurité.

Chaque jalon doit produire une preuve vérifiable : rapport de test, procédure de récupération, résultat de GameDay ou décision d'architecture.

Passer du prototype gouverné à l'exploitation qualifiée



La feuille de route priorise la preuve opérationnelle avant la multiplication des fonctions.

Figure 8 — Roadmap centrée sur la qualification, pas sur l'empilement fonctionnel.

10.2 — Protocole de soutenance conseillé

Une soutenance réussie doit raconter un problème, montrer une décision et finir sur une preuve.

Commencer par le modèle qui répond encore mais dérive. Présenter ensuite les deux systèmes, CAC_MLOPS comme terrain et Agentic-Ops comme plan de contrôle. Faire vivre un incident de bout en bout. Montrer enfin ce que le système refuse et ce qu'il conserve comme preuve.

Les détails cryptographiques, les ports, les classes et les fournisseurs LLM doivent rester disponibles en annexe ou en réponse aux questions. Ils ne doivent pas casser le récit principal.

La phrase de conclusion peut rester simple : le projet ne cherche pas à rendre l'IA toute-puissante ; il cherche à la rendre utile, contrôlable et explicable dans un contexte d'exploitation.

Temps indicatif	Séquence
2 min	Problème et incident fil rouge
3 min	Deux plateformes, une architecture
5 min	Démonstration du cycle gouverné
3 min	Sécurité, humain et preuve
2 min	Résultats, limites, roadmap

10.3 — Conclusion générale

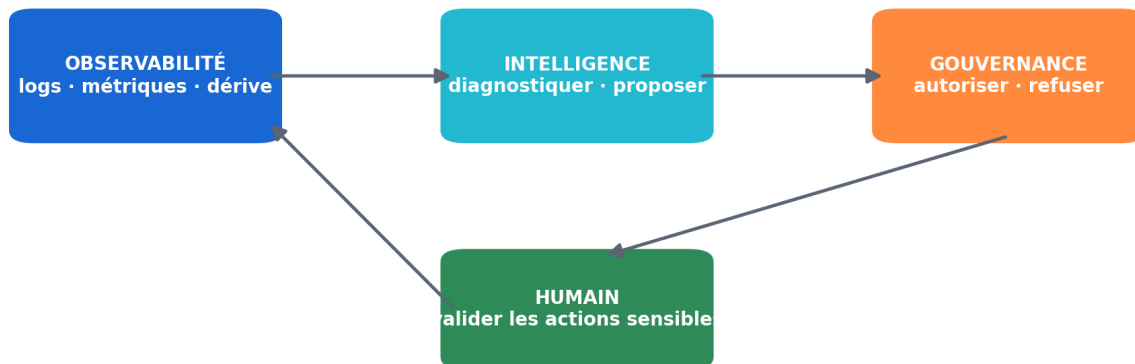
Agentic-Ops répond à une question simple mais exigeante : comment bénéficier de l'analyse d'une IA sans lui confier seule le pouvoir d'agir ?

La réponse proposée repose sur une séparation des pouvoirs. Les sources observent. Les spécialistes interprètent. L'agent de remédiation propose. Le moteur de politiques décide selon des règles explicites. L'humain arbitre les actions sensibles. Le registre conserve l'histoire.

Cette architecture ne supprime ni l'incertitude du machine learning, ni les pannes, ni la responsabilité opérationnelle. Elle les rend plus visibles et plus gouvernables. C'est précisément là que se situe la contribution : transformer une promesse d'autonomie en une chaîne de décision maîtrisée.

La suite du projet ne doit pas chercher d'abord plus de sophistication cognitive. Elle doit rechercher davantage de preuves opérationnelles : tests reproductibles, inventaire fiable, identité durcie, stockage durable, GameDays et qualification progressive des actions réelles.

De l'alerte à la décision sûre



Principe directeur : l'IA éclaire ; les règles et l'humain gardent la maîtrise.

Figure 9 — L'IA éclaire la décision ; la gouvernance en fixe les limites.

Annexe A — Cartographie synthétique des composants

Bloc	Composants cités dans le dossier	Rôle
Collecte	CacPullCollector, adaptateurs	Transformer les sources en faits
Cognition	Spécialistes Data, Model, Infra, Remediation	Analyser et proposer
Orchestration	Orchestrator, FSM, idempotence	Garantir l'ordre et l'unicité
Gouvernance	PolicyEngine, catalogue, registre de risques	Autoriser, suspendre ou refuser
Exécution	ActionExecutor, drivers	Appliquer uniquement une décision valide
Preuve	EvidenceStore, Merkle, sauvegarde	Reconstituer et vérifier
Interfaces	Web, Slack, MCP	Dialogue, approbation, intégration

Lecture recommandée

Cette annexe sert de repère technique. Elle ne remplace pas le parcours narratif du rapport.

Annexe B — Matrice des principaux risques

Risque	Conséquence	Barrière proposée	Limite
Hallucination de cible	Mutation d'une ressource incorrecte	Inventaire exact	Inventaire statique à synchroniser
Action trop large	Impact excessif	Rayon d'impact	Qualité du catalogue
Injection dans les logs	Influence du diagnostic	Quarantaine + politique	Défense à tester continuellement
Rejeu d'approbation	Exécution non consentie	Signature + fenêtre temporelle	Gestion d'identité à durcir
Perte de réseau	Décision sur faits périmés	Fail-closed	Disponibilité réduite
Altération du journal	Audit trompeur	Chaîne de hachage + Merkle	Dépend de la protection des clés

Annexe C — Glossaire de soutenance

Terme	Explication simple
Blast radius	Étendue maximale des effets d'une action.
Fail-closed	Refuser lorsque les conditions de sécurité ne peuvent pas être prouvées.
FSM	Machine à états qui impose l'ordre des étapes.
HITL	Validation humaine intégrée au cycle de décision.
MLOps	Pratiques d'industrialisation et d'exploitation du machine learning.
Policy Engine	Moteur déterministe qui applique les règles d'autorisation.
RAG	Recherche de connaissances venant étayer une réponse.
WORM	Principe d'écriture append-only visant à empêcher la réécriture silencieuse.
Merkle	Arbre de hachage permettant des preuves d'inclusion compactes.
MCP	Protocole d'exposition d'outils et de ressources à des clients IA.

Annexe D — Sources et traçabilité

Sources publiques consultées le 20 octobre 2026 :

- Dépôt CAC_MLOPS : https://github.com/jakatt/cac_mlops
- Dépôt Agentic-Ops : <https://github.com/noll77/agentik-ops>

Source principale de rédaction : extraction technique et premier rapport fournis par l’auteur dans la demande. Les nombres de tests, scénarios, latences, seuils et détails internes repris dans le présent document proviennent de cette source et sont formulés comme résultats déclarés lorsqu’ils ne sont pas directement visibles dans les dépôts publics.

Note de sécurité rédactionnelle : les adresses privées, secrets, clés, instructions contenues dans des exemples de logs et détails inutiles à la compréhension ont été minimisés ou omis.

Annexe E — Checklist de preuve avant soutenance

- Capturer le commit exact d'Agentic-Ops utilisé.
- Générer le rapport pytest complet et conserver la commande.
- Exécuter le scénario fil rouge en mode simulation.
- Montrer un refus de cible inconnue.
- Montrer une demande d'approbation et un rejet de requête invalide.
- Vérifier la chaîne d'évidence avant et après une altération contrôlée.
- Distinguer à l'oral les résultats simulés, locaux et réellement intégrés.
- Préparer une slide « limites et prochain jalon ».

Message final à préserver : la valeur du projet n'est pas de tout automatiser, mais de rendre l'automatisation gouvernable.