MSE | MASTER OF SCIENCE IN ENGINEERING

Déploiement IT en « vibe-coding »

Chaos ou automatisation : l'IA générative face aux défis du déploiement IT

Management des Projets IT
MSE

Eva Ray, Samuel Roland, Massimo Stefani

3 Janvier 2026

Hes·so

Table des matières

Resume 2
Introduction 3
Contexte 3
Objectif du rapport 4
Problématique 4
Méthodologie 5
Approche pédagogique 5
Outils et technologies utilises 5
OpenDidac 5
Kubernetes 6
Terraform 6
Windsurf 6
Modèle GPT-5 (medium reasoning) 8
Participants 8
Collecte de données 9
Critères d'evaluation 9
Références 9
Résultats et analyses 9
Phase de déploiement 9
Synthèse des resultats 9
Discussion 9
Bénéfices observés 9
Défis rencontrés 9
Retour d'expérience du groupe 9
Comparaison avec d'autres approches ou pratiques 9
Conclusion 10
Recommandations 10
Références 10
Bibliographie 10
Liens additionnels 11
Annexes 11

Resume

Introduction

Contexte

La gestion de projets informatiques nécessite des méthodologies structurées pour organiser le développement et le déploiement de systèmes d'information. Le Systems Development Life Cycle (SDLC), située dans la phase « Execute and Control Project » du Project Life-Cyle (PLC), représente le cycle de vie produit le plus utilisé dans le domaine des technologies de l'information. Ce cycle établit une séquence logique d'activités de développement organisées en phases distinctes : la planification (planning), l'analyse (analysis), la conception (design), l'implémentation (implementation), et la maintenance et support (maintenance and support).

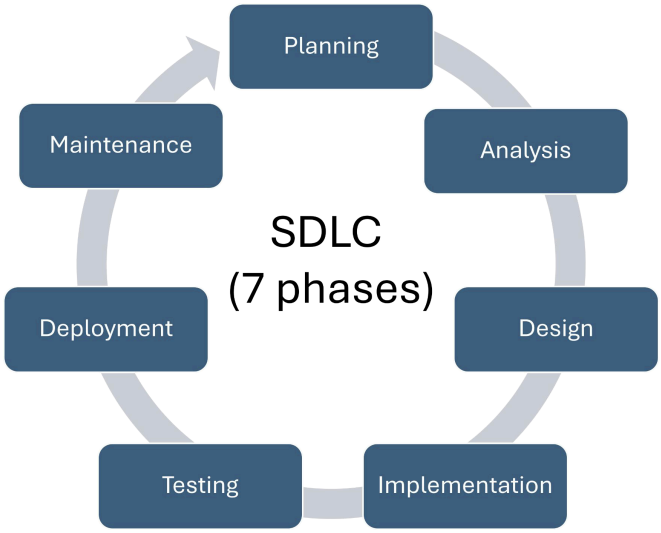


Fig. 1. – SDLC en 7 phases

Chaque phase du SDLC présente des enjeux spécifiques.

- La phase de planification se caractérise par sa forte dépendance aux livrables de la phase précédente du PLC et peut se décliner selon trois situations : une planification complète et imposée par le client (date, budget et fonctionnalités), une absence de planification préalable nécessitant une proposition complète du prestataire IT, ou une situation intermédiaire où seul le périmètre fonctionnel est défini. Cette phase distingue les projets feature-driven, typiquement internes avec flexibilité sur les délais, des projets date-driven, généralement externalisés avec flexibilité sur le périmètre fonctionnel.
- L'analyse englobe l'analyse du business, des utilisateurs, des technologies, des travaux antérieurs et de la faisabilité du projet. Cette phase produit la Software Requirements Specification (SRS), document décrivant le système logiciel à développer en détaillant les exigences fonctionnelles et non-fonctionnelles, pouvant inclure des cas d'utilisation, la définition des utilisateurs, des activités utilisateur et des contraintes métier.
- La conception élabore l'architecture des différentes parties du système selon une approche hiérarchique de décomposition en composants et modules. Cette phase définit la conception système, les interfaces, les bases de données et fichiers, ainsi que la conception des programmes, en sélectionnant la pile technologique (technology stack) appropriée. Le résultat est une description formelle et une représentation du système complet consignée dans un document de spécification détaillé.
- L'implémentation concrétise cette architecture par le développement, l'intégration, les tests, l'installation du système, ainsi que la formation et la documentation nécessaires à son utilisation.
- La phase de maintenance et support assure la pérennité du système en production, gérant les corrections d'erreurs, les améliorations fonctionnelles et l'adaptation aux évolutions de l'environnement organisationnel.

La maîtrise de ces phases et de leurs interdépendances constitue un facteur déterminant pour la gestion de projet IT, permettant d'assurer une progression ordonnée depuis l'identification d'un besoin jusqu'à la mise en production et le maintien opérationnel d'un système d'information.

Objectif du rapport

L'intégration de l'IA générative dans le processus de déploiement peut offrir plusieurs avantages.

- **Automatisation de la création des fichiers de déploiement**: Beaucoup d'outils permettant de déployer des applications nécessitent la création de fichiers de configuration spécifiques (docker-compose.yml, Kubernetes YAML, etc.). L'IA générative peut automatiser la création de ces fichiers en fonction des spécifications du projet, réduisant ainsi le temps et les erreurs humaines.
- **Analyse automatique des environnements**: Le déploiement varie en fonction des providers cloud, des configurations réseau et des contraintes de sécurité. L'IA générative peut analyser automatiquement l'environnement cible et adapter les fichiers ou scripts de déploiement en conséquence.
- **Assistance à la résolution des problèmes**: Lors du déploiement, des problèmes techniques peuvent survenir. L'IA générative peut fournir une assistance en temps réel pour diagnostiquer et résoudre ces problèmes, en suggérant des solutions basées sur des bases de données de connaissances.
- **Automatisation de la documentation**: La documentation est essentielle pour le déploiement et la maintenance des systèmes. L'IA générative peut automatiser la création de documentation technique, facilitant ainsi la compréhension et l'utilisation du système par les équipes de développement et d'exploitation.
- **Optimisation des processus de déploiement**: L'IA peut analyser les processus de déploiement existants et suggérer des améliorations pour les rendre plus efficaces, en identifiant les goulots d'étranglement et en proposant des stratégies d'optimisation. L'IA peut aussi aider à concevoir des stratégies de déploiement minimiser le temps d'indisponibilité.

Dans ce projet, nous avons choisi de nous concentrer sur le déploiement cloud sur AWS d'une application web déjà existante en utilisant Kubernetes et Terraform. Ainsi, nous avons surtout exploré comment l'IA générative peut assister dans la création des fichiers de déploiement, l'adaptation aux environnements cloud, ainsi que l'aide à la résolution des problèmes.

Problématique

Le déploiement est une étape dans le SDLC qui peut être complexe et où plusieurs défis peuvent compromettre la mise en production.

- **Complexité des environnements hétérogènes**: Les systèmes sont souvent déployés sur des infrastructures variées (cloud multi-fournisseurs, hybride) avec des configurations et contraintes différentes, augmentant le risque d'incompatibilités et d'erreurs. L'IA générative peut analyser automatiquement les caractéristiques de chaque environnement cible et générer des scripts ou fichiers de déploiement adaptés, incluant la détection et la résolution de conflits de dépendances.
- **Gestion des fichiers de configuration**: La création et la gestion des fichiers de configuration peut être fastidieuses, chronophages et sujettes aux erreurs. L'IA générative peut automatiser la génération de ces fichiers en fonction des besoins spécifiques de chaque environnement, assurant ainsi une cohérence et une réduction des erreurs humaines.
- **Gestion des dépendances et des versions**: Les systèmes modernes comportent de nombreuses dépendances (bibliothèques, frameworks, services externes) dont les incompatibilités de versions peuvent provoquer des échecs de déploiement difficiles à diagnostiquer. L'IA générative peut aider à identifier les dépendances nécessaires, à gérer les versions compatibles et à générer des scripts d'installation automatisés.
- **Gestion des erreurs**: Le déploiement peut échouer pour diverses raisons, et la détection rapide des erreurs est cruciale. L'IA générative peut analyser les logs de déploiement en temps réel,

identifier les erreurs potentielles et proposer des solutions correctives basées sur des modèles d'apprentissage automatique.
- **Interruption de service**: Le déploiement nécessite souvent des interruptions de service impactant la disponibilité du système, particulièrement critique pour les systèmes en production 24/7. L'IA générative peut concevoir des stratégies de déploiement intelligentes (blue-green deployment, rolling updates) optimisant la séquence des opérations pour minimiser l'indisponibilité.

Défis du déploiement	**Solutions potentielles avec l'IA générative**
Complexité des environnements hétérogènes	Analyse automatique des environnements et génération de fichiers ou scripts adaptés.
Gestion des fichiers de configuration	Automatisation de la création et mise à jour des fichiers de configuration.
Gestion des erreurs et diagnostics	Analyse des logs, détection des erreurs et suggestion de corrections.
Documentation	Génération automatique de documentation technique pour faciliter la maintenance et l'exploitation du système.
Interruption de service	Conception de stratégies de déploiement pour minimiser les temps d'indisponibilité.

Méthodologie

Approche pédagogique

Dans le cadre de ce projet, l'approche pédagogique adoptée est celle de la classe inversée. Cette méthode consiste à inverser les rôles traditionnels des étudiants et des enseignants. En effet, les étudiants se voient attribuer un thème lié au SDLC ou au PLC qu'ils doivent étudier afin de le présenter à leurs pairs et aux enseignants lors de sessions dédiées. En particulier, pour ce projet, les étudiants sont chargés d'explorer l'intégration de l'IA générative dans le processus de déploiement des systèmes informatiques. Le but est de tester et évaluer comment l'IA générative permet d'améliorer l'efficacité, la qualité et la gestion des projets IT à chaque étape de leur cycle de vie. Le côté pratique est mis en avant à travers des démonstrations des résultats obtenus, des analyses critiques et des discussions sur les bénéfices et défis rencontrés.

Outils et technologies utilises

OpenDidac

Notre but étant de s'intéresser à la phase de déploiement, nous avons tout d'abord sélectionné une application web à déployer. Nous avons choisi l'application web `OpenDidac`, qui est une plateforme open-source de gestion d'évaluation en ligne développée par l'HEIG-VD. En particulier, cette application permet aux enseignants de concevoir des questionnaires, aux étudiants de se connecter et de répondre aux évaluations, et enfin de récupérer les réponses aux questions pour analyse. Les questionnaires permettent plusieurs types de questions: vrai/faux, QCM, questions ouvertes, questions avec exécution de code ou encore des requêtes de bases de données. L'architecture de l'application est la suivante :
- **Frontend**: Développé en Typescript avec le framework Next.js 14 pour React.
- **Backend**: Développé en Typescript avec le framework Next.js API Routes.
- **Base de données**: PostgreSQL avec modélisation gérée par l'ORM Prisma.
- **Authentification**: Gestion des utilisateurs et des sessions avec NextAuth.js et keycloak.
- **Conteneurisation**: Utilisation de Docker pour la création d'images et la gestion des conteneurs.

Le but de ce projet est donc de déployer cette application web en utilisant différentes technologies d'infrastructure, en s'appuyant sur l'IA générative pour automatiser le processus de déploiement.

Kubernetes

Terraform

Windsurf

Pour intégrer l'IA générative dans notre processus de déploiement, nous avons utilisé Windsurf. Windsurf est un environnement de développement intégré (IDE) de nouvelle génération intégrant nativement des capacités d'intelligence artificielle générative pour assister les développeurs tout au long du cycle de développement logiciel. En particulier, la fonctionnalité distinctive de Windsurf réside dans son mode Agent (appelé Code), qui permet à l'IA d'agir de manière autonome sur le code et le projet. En mode Agent, l'IA ne se limite pas à suggérer du code ou répondre à des questions, elle peut analyser l'architecture du projet, proposer des modifications dans les fichiers, créer des fichiers, refactoriser du code existant, générer des tests, créer de la documentation ou encore exécuter des commandes dans le terminal intégré à l'IDE.

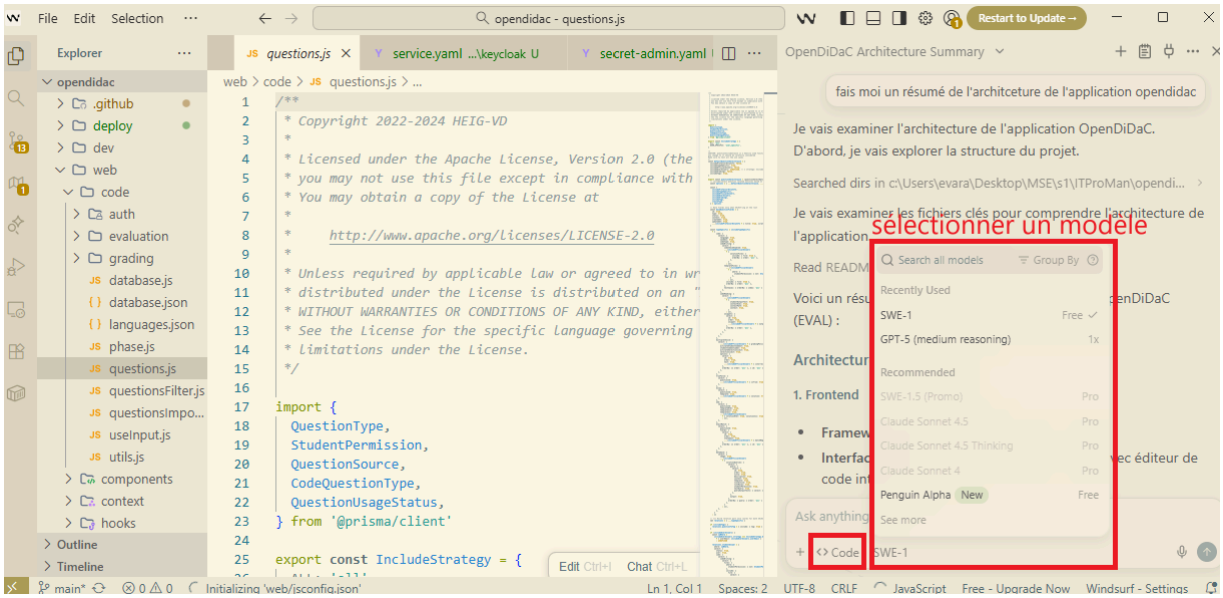


Fig. 2. – Aperçu de l'interface de Windsurf avec les fichiers et le code à gauche, et la conversation et les configurations à droite

Windsurf offre l'accès à plusieurs modèles d'IA avec des capacités et des coûts variables :

- **Modèle de base (SWE-1)**: Un modèle gratuit et illimité adapté aux tâches courantes de développement, permettant une utilisation quotidienne sans consommation de crédits.
- **Modèles premium**: Ces modèles avancés offrent des performances supérieures en termes de compréhension contextuelle, de génération de code complexe et de raisonnement. Chaque requête utilisant un modèle premium consomme 1 crédit.

Un plan gratuit incluant 25 crédits par mois est proposé, permettant aux développeurs de tester les modèles premium pour des tâches nécessitant des capacités avancées ou pour comparer les performances entre les différents modèles. Des plans payants sont également disponibles pour les utilisateurs ayant des besoins plus importants en crédits. Le tarif d'entrée pour 500 crédits par mois pour un utilisateur est de 15 USD.

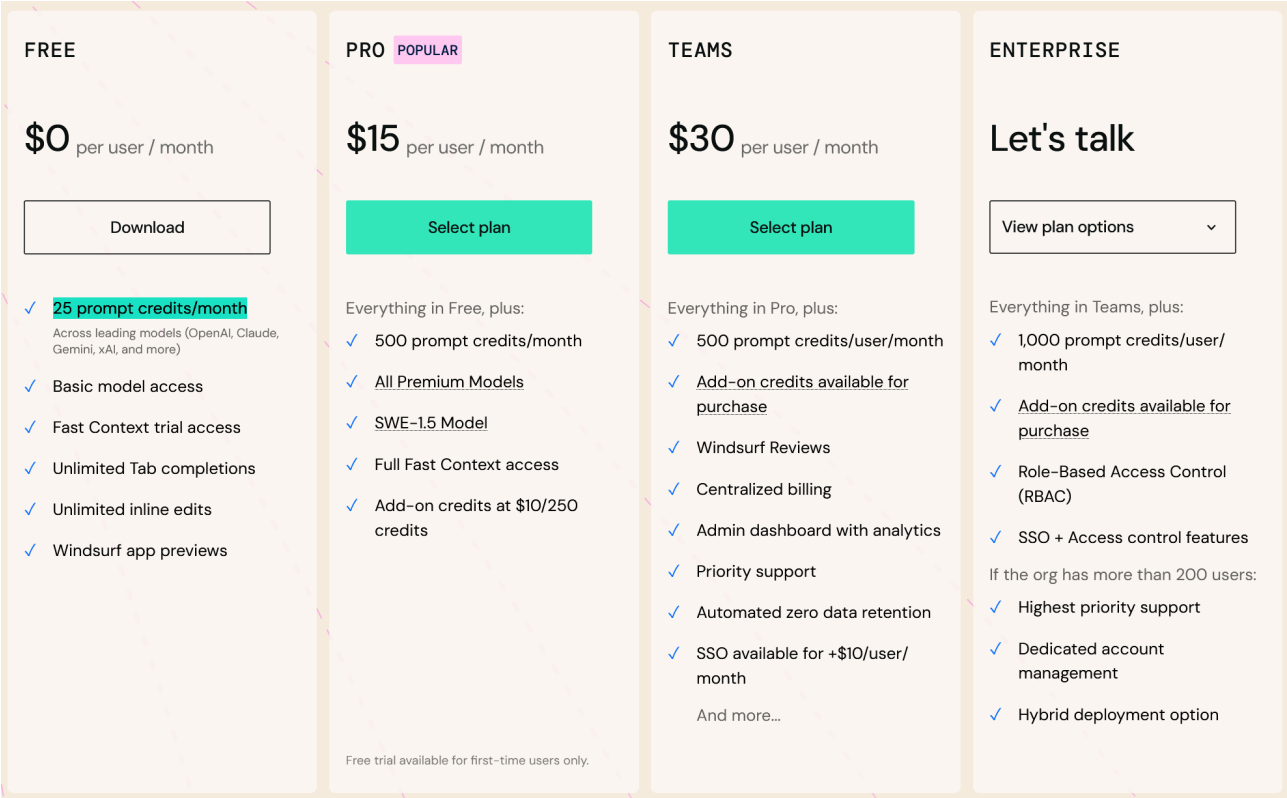


Fig. 3. – Plans de paiements de Windsurf, en décembre 2025

Sans contraintes, l'IA peut produire du code non conforme aux standards de l'entreprise, introduire des vulnérabilités de sécurité ou utiliser des patterns incompatibles avec l'architecture existante. Face aux défis de qualité et de cohérence du code généré par l'IA, Windsurf propose deux mécanismes de contrôle : les rules et les workflows. Les rules garantissent que le code généré respecte les conventions établies, tandis que les workflows assurent la reproductibilité et la fiabilité des processus critiques comme les déploiements. Ces outils permettent ainsi de maintenir un contrôle qualité tout en bénéficiant de la productivité apportée par l'IA.

Les rules permettent de définir des directives que l'IA doit suivre lors de la génération de code. Ces règles peuvent être configurées à deux niveaux :

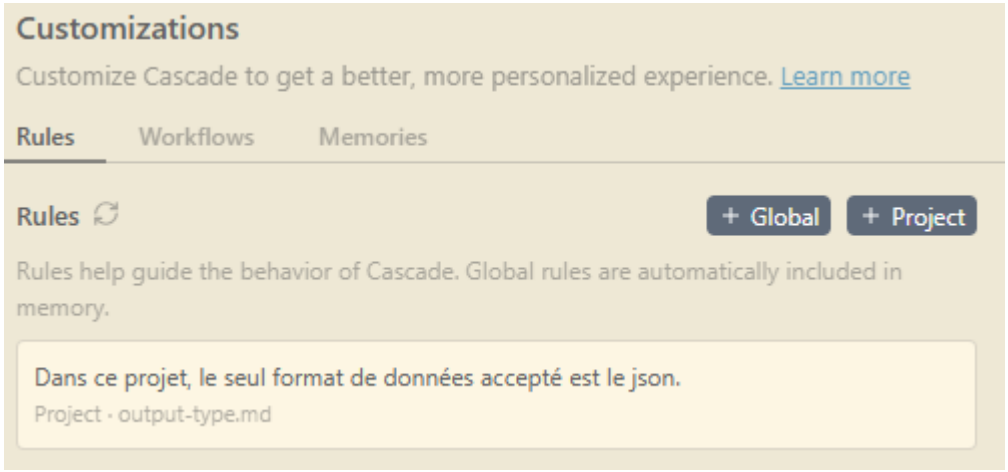
- **Global Rules**: Applicables à tous les projets de l'utilisateur, elles définissent des préférences personnelles comme le style de code ou les conventions de nommage.
- **Project Rules**: Spécifiques à un projet, elles permettent d'établir des guidelines organisationnelles, comme des standards d'architecture, des patterns de sécurité ou des contraintes techniques propres à l'entreprise. Idéalement, chaque entreprise utilisant windsurf devrait définir un ensemble de Project Rules pour garantir la cohérence et la qualité du code généré par l'IA au sein de ses projets.

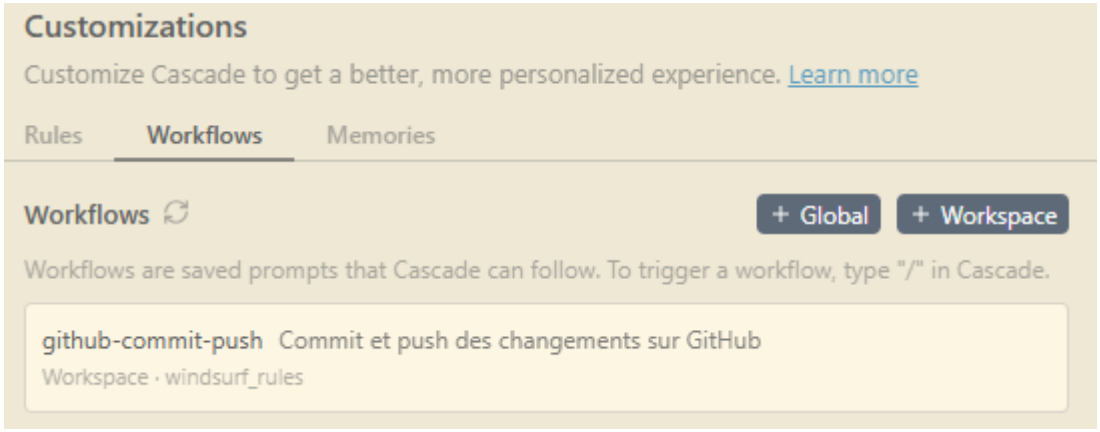
Windsurf plusieurs modes d'application des rules, dont :

- **Mode Always On**: Les rules sont systématiquement appliquées à chaque interaction avec l'IA, garantissant une conformité constante.
- **Mode Model Decision**: L'IA décide de manière autonome quand appliquer les rules en fonction du contexte de la requête, offrant plus de flexibilité.

Les workflows, quant à eux, permettent d'automatiser des tâches répétitives en créant des séquences d'actions guidées. Par exemple, un workflow de déploiement peut enchaîner automatiquement la vérification des tests, la construction de l'application, la génération de documentation et le déploiement sur l'environnement cible. L'utilisateur invoque le workflow comme une commande, et Windsurf guide

ensuite le processus étape par étape, en demandant les paramètres nécessaires et en exécutant les actions définies.





Modèle GPT-5 (medium reasoning)

- https://www.vellum.ai/best-llm-for-coding
- https://www.swebench.com/index.html
- https://epoch.ai/benchmarks/gpqa-diamond

Participants

Ce projet a été réalisé par trois étudiants de l'orientation Computer Science du Master of Science in Engineering (MSE) de la HES-SO. Les membres sont tous des ingénieurs détenteurs d'un Bachelor of Science en informatique et systèmes de communication avec une orientation en informatique logicielle, délivré par la HEIG-VD. Les participants sont Eva Ray, Samuel Roland et Massimo Stefani.

Les trois membres ont suivi un cours de cloud computing durant leur Bachelor et suivent actuellement une version avancée de ce cours dans le cadre de leur Master. Ainsi, ils possèdent tous des connaissances de base en déploiement d'applications cloud. Cependant, Samuel et Eva sont débutants en Kubernetes et Terraform, tandis que Massimo a une solide expérience de ces outils qu'ils a utilisés dans des projets professionnels antérieurs, ainsi que son travail de Bachelor.

Pour ce projet, le travail a été divisé en trois parties, de la manière suivante :

- Eva Ray : Déploiement Kubernetes de la base de données PostgreSQL et de keycloak
- Samuel Roland : Déploiement Kubernetes de l'application web OpenDidac
- Massimo Stefani : Déploiement Terraform de l'infrastructure Kubernetes sur AWS

Collecte de données

Critères d'evaluation

Références

Résultats et analyses

Phase de déploiement

Synthèse des resultats

Discussion

Bénéfices observés

Défis rencontrés

Retour d'expérience du groupe

Comparaison avec d'autres approches ou pratiques

Comme le montre la Fig. 4, nous avons défini 6 niveaux d'adoption de l'IA, pour cette comparaison. Tout à gauche, le niveau d'adoption zéro, sans aucune aide d'intelligence artificielle. Ensuite, nous avons les chatbot en ligne qui regroupent ChatGPT, Copilat Chat, et beaucoup d'autres. Ils sont accessibles via des sites web et permettent souvent de choisir entre différents LLM mises à disposition. Ces chatbots en ligne n'ont comme contexte que l'historique de conversation des messages fournis, et peuvent parfois faire également des recherches sur le web. Leur réponse ne peut être donnée quand dans l'interface de chat, leur changements doivent donc être intégrés à la main. Dans le contexte d'une base de code existante comme Opendidac, il aurait fallu lui donner le contenu de certains fichiers qui nous semblent pertinents, pour qu'il puisse comprendre la structure de l'application.

Pour donner par défaut accès à suffisament de contexte, nous avons choisi l'option des IDE dédié, comme Cursor et Windsurf. En effet, le mode agent de Windsurf peut de lui-même décider d'aller lire les fichiers du projets qui lui sont utiles, afin de comprendre le projet ouvert dans l'IDE. Une fois l'architecture identifiée, il peut modifier des fichiers dans n'importe quelle partie du projet. Un panneau de conversation est intégré et permet de demander des modifications.

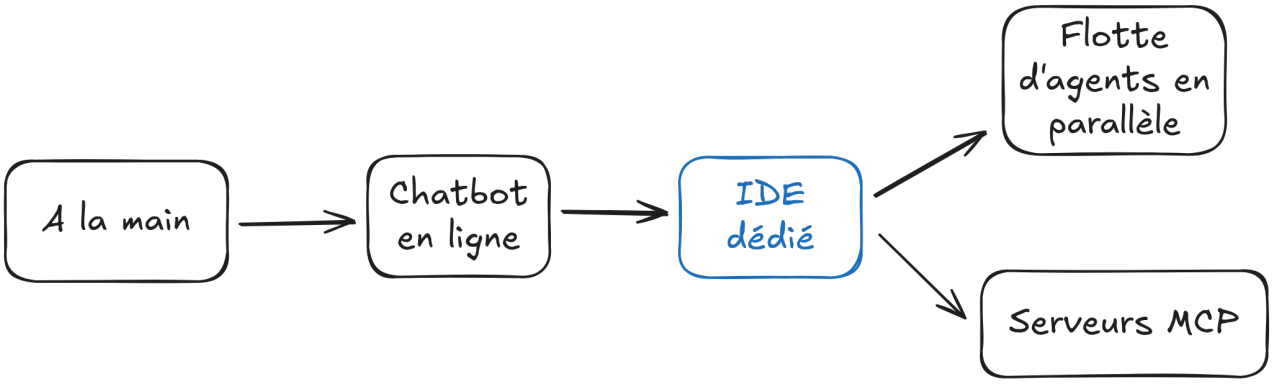


Fig. 4. – Niveaux d'adoption de l'IA, avec notre approche en bleu

Les dernières approches qui poussent encore plus loin l'intégration sont les flottes d'agents en parallèle et les serveurs MCP. Nous n'avons exploré pas ces outils, pour limiter le temps de configuration et d'apprentissage, mais ils pourraient fournir une meilleure expérience. Au lieu d'avoir un seul agent, pourquoi ne pas en avoir plusieurs, avec différents rôles, pour gérer les étapes, découper en tâches,

coder, revoir le résultat, tester le déploiement… Ces systèmes de flottes permettent de simuler les rôles d'une équipe de développement, pour résoudre des tâches plus complexes et une meilleure qualité finale.[1]

L'autre option qui n'est pas incompatible avec la précédente, consiste à donner accès à des outils ou ressources avancées via des serveurs MCP (Model Context Protocol) [1]. Nous aurions pu utiliser un serveur MCP pour Kubernetes, permettant à notre LLM d'accéder à l'état du cluster et de créer des ressources, sans avoir de CLIs installés sur nos machines. Au lieu de nous donner des commandes `kubectl` pour lancer les services, pods et autres fichiers définis, une intégration d'un serveur MCP aurait pu permettre de lancer ces actions tout seul (après certaines approbations).[2]

Conclusion

Recommandations

Références

Bibliographie

[1] « Understanding MCP servers - Model Context Protocol ». Consulté le: 2 janvier 2026. [En ligne]. Disponible sur: https://modelcontextprotocol.io/docs/learn/server-concepts

[2] J. Davison, « What is SDLC? A Simple Guide For Business Professionals - People Helping Machines Helping People ». Consulté le: 6 décembre 2025. [En ligne]. Disponible sur: https://content.startuplandia.io/thoughts/software-development-lifecycle/

[3] A. Talreja, « Deployment Phase in SDLC: Strategies, CI/CD & Best Practices (2026) ». Consulté le: 6 décembre 2025. [En ligne]. Disponible sur: https://teachingagile.com/sdlc/deployment

[4] H. Team, « The Seven Phases of the Software Development Life Cycle ». Consulté le: 9 décembre 2025. [En ligne]. Disponible sur: https://www.harness.io/blog/software-development-life-cycle-phases

[5] Wahengchang, « Mastering Windsurf: Restricting AI Output with Windsurf Rules ». Consulté le: 6 décembre 2025. [En ligne]. Disponible sur: https://medium.com/@wahengchang2024/mastering-windsurf-restricting-ai-output-with-windsurf-rules-d7e429654db2

[6] P. Duvall, « Windsurf Rules & Workflows: AI-Driven Software Delivery Best Practices ». Consulté le: 6 décembre 2025. [En ligne]. Disponible sur: https://www.paulmduvall.com/using-windsurf-rules-workflows-and-memories/

[7] S.-b. Team, « SWE-bench Leaderboards ». Consulté le: 6 décembre 2025. [En ligne]. Disponible sur: https://www.swebench.com/

[8] P. Gauthier, « Aider LLM Leaderboards | aider ». Consulté le: 6 décembre 2025. [En ligne]. Disponible sur: https://aider.chat/docs/leaderboards/

[9] Vellum, « Best LLM for Coding ». Consulté le: 6 décembre 2025. [En ligne]. Disponible sur: https://www.vellum.ai/best-llm-for-coding

[10] Z. Z. Z. X. X. S. Tianyi Zhang Shidong Pan, « Deployability-Centric Infrastructure-as-Code Generation: An LLM-based Iterative Framework ». Consulté le: 9 décembre 2025. [En ligne]. Disponible sur: https://arxiv.org/abs/2506.05623

[1] Voir par exemple CrewAI: https://www.crewai.com/

[2] https://github.com/containers/kubernetes-mcp-server/

[11] J.-H. C. A. B. Rana Nameer Hussain Khan Dawood Wasif, « Multi-Agent Code-Orchestrated Generation for Reliable Infrastructure-as-Code ». Consulté le: 9 décembre 2025. [En ligne]. Disponible sur: https://arxiv.org/abs/2510.03902

Liens additionnels

- Repository Git de Opendidac: https://github.com/opendidac/opendidac
- Windsurf website: https://windsurf.com/
- Windsurf options d'abonnements: https://windsurf.com/pricing
- Serveur MCP pour Kubernetes: https://github.com/containers/kubernetes-mcp-server/

Annexes