Blue AcademIA
Blue AcademIA est une marque de BlueMarketing
contact@blueacademia.fr
www.blueacademia.fr

Programme de formation : Intégrer les agents IA dans votre workflow de développement

Vos développeurs utilisent déjà l'IA, chacun à sa façon. Les résultats sont inégaux, rien n'est outillé en équipe. Ce parcours de 10 jours vous fait passer du prompt occasionnel à un poste de travail réellement augmenté. V ous maîtrisez Claude Code en profondeur, vous le connectez à vos outils et vos données via MCP et les Skills, vous construisez vos propres agents avec l' API Anthropic, puis vous les intégrez aux pratiques d'équipe : tests,
documentation, analyse de code legacy, pipelines CI/CD.
60 % du temps se passe en pratique, sur du code réel, sous supervision humaine à chaque étape. Vous repartez avec des agents de développement fonctionnels et un pipeline CI/CD agentique complet, reproductibles sur vos propres dépôts.

Durée : 70.00 heures ou 10.00 jour(s) si formation en présentiel.

Lieu : À distance ou en présentiel

Profils des stagiaires

- Développeurs et développeuses (Python, JavaScript, Java…)
- Ingénieurs DevOps et CI/CD
- Architectes logiciels
- Tech leads
- Toute personne ayant des bases solides de programmation

Prérequis

- Avoir les bases d'un langage de programmation (Python recommandé, les labs sont en Python et JavaScript),
- Maîtriser l'utilisation du terminal (bash/zsh) et de Git (commit, push, branches).
- Avoir un éditeur de code installé (VS Code ou Cursor) et des comptes créés en amont : GitHub, Anthropic (Claude API), Hugging Face.
- Aucune expérience préalable de l'IA générative n'est requise.
- Les pré-requis sont validés lors d'un échange avec le commanditaire du projet de formation et retranscrits dans le document de recueil des besoins. Il s'agit d'un document collaboratif partagé systématiquement avec le commanditaire à l'issue du premier entretien pour co-construire l'action de formation. Par ailleurs, nous adressons à chaque apprenant, 2 semaines avant le début de la formation, un questionnaire de préparation de la formation pour adapter au mieux la formation à ses besoins.

Objectifs pédagogiques

- Cartographier l'écosystème des outils IA pour le développement (extensions IDE, agents CLI, APIs, agents autonomes) et choisir le bon outil selon le contexte
- Maîtriser Claude Code en profondeur : configuration avancée, contexte projet (CLAUDE.md), permissions, mode interactif et mode headless

- Connecter l'agent aux outils et données de l'entreprise via le Model Context Protocol (MCP) et étendre ses capacités avec les Skills
- Déployer des workflows d'équipe : tâches longues déléguées en arrière-plan (BMAD), hooks, conventions partagées et versionnées dans Git
- Construire un agent de code from scratch avec l'API Anthropic : Tool Use, cycle ReAct, outils de lecture, d'écriture et d'exécution
- Générer automatiquement tests (unitaires, intégration, mutation testing) et documentation (docstrings, OpenAPI, ADR, README) par agents
- Cartographier et indexer une codebase (repo-map, embeddings, recherche sémantique) pour détecter les bugs et refactoriser du legacy
- Intégrer revue de code et tests par l'IA dans les pipelines CI/CD (GitLab CI, GitHub Actions) avec supervision humaine et sécurisation des agents
- Livrer en projet final un agent autonome de développement — fonctionnel, testé, documenté — et le soutenir devant le groupe

Contenu de la formation

- Panorama de l'IA pour le code (J1, 7 h)
    - Panorama des outils : assistants de code, agents CLI (Claude Code, Aider, Codex), APIs LLM - la différence fondamentale entre autocomplétion, génération et agent autonome
    - Architecture des LLM de code : pré-entraînement sur code, Fill-in-the-Middle, fenêtre de contexte ; modèles clés et benchmarks (HumanEval, SWE-Bench)
    - Critères de choix : latence, confidentialité, coût, qualité selon les langages cibles
    - Installation et prise en main : Claude Code en CLI, extension VS Code, connexion aux APIs
    - Cas pratique appliqué : Lab comparatif : le même exercice de génération de code mené sur plusieurs outils, mesure de la pertinence, de la vitesse et de la complétude, puis débrief collectif pour identifier quand utiliser quoi.
- Claude Code, maîtrise avancée (J2 et J3, 14 h)
    - Fonctionnalités avancées : édition multi-fichiers, terminal intégré, création de PR, mode interactif vs headless, personnalisation via CLAUDE.md
    - Prompting efficace pour le code : conventions, exemples few-shot, gestion des hallucinations, patterns Explain / Refactor / Optimize / Test
    - API Anthropic et Tool Use : SDK Python/Node.js, function calling, streaming, gestion du contexte long
    - Model Context Protocol (MCP) : architecture (serveurs, clients, transports), MCP intégrés (filesystem, git, GitHub, bases de données), création et sécurisation d'un MCP custom
    - Skills : définir un SKILL.md, skills prédéfinis (tests, doc, refactoring, sécurité), créer, partager et composer des skills d'équipe versionnés dans Git
    - Cas pratique appliqué : Refactoring guidé par IA d'un projet Python legacy (identifier, tester, refactoriser, documenter), puis atelier MCP + Skills : connecter Claude Code à une base PostgreSQL via un MCP custom et créer un Skill d'audit de schéma.
- Workflows équipe & automatisation (J4, 7 h)
    - Déléguer des tâches longues à l'agent en arrière-plan (BMAD) : sessions persistantes, migrations de codebase, génération de tests massive, checkpoints de validation humaine
    - CLAUDE.md projet : conventions, architecture, commandes, règles de style - versionné dans Git pour un onboarding instantané de l'agent
    - Hooks personnalisés : déclencher des actions avant et après chaque action de l'agent
    - Intégration dans les PR GitHub/GitLab : revue automatisée et suggestions
    - Sécurité d'équipe : journalisation des actions, audit des permissions, gestion des secrets

    - Cas pratique appliqué : Setup d'équipe complet : configurer un dépôt avec CLAUDE.md partagé, lancer une tâche de fond de refactoring (migration Python 2→3), valider les étapes, puis simuler l'onboarding d'un autre participant sur le projet.
- Agents de code, architecture & premiers agents (J5, 7 h)
    - Anatomie d'un agent codeur : ce qui le distingue d'un simple appel LLM, cycle ReAct appliqué au code (lire → analyser → planifier → écrire → exécuter → observer)
    - Les outils canoniques d'un agent : lecture et écriture de fichiers, exécution de commandes, recherche dans la codebase, opérations Git ; gestion du contexte long (repo-map)
    - Tool Use avec l'API Anthropic : définir des outils en JSON Schema, boucle d'exécution, gestion de la récursion et des erreurs
    - Agents CLI open source (Aider…) : comparaison des architectures et cas d'usage, bonnes pratiques partagées : commits atomiques, validation humaine
    - Cas pratique appliqué : Lab agent lecteur/modifieur : construire from scratch un agent Python avec l'API Anthropic qui reçoit une tâche, planifie, modifie les fichiers et exécute les tests puis le comparer avec les agents du marché.
- Génération de tests & documentation automatique (J6-J7, 14 h)
    - Génération de tests unitaires : stratégies (analyser la fonction → cas limites → assertions), frameworks cibles (pytest, Jest, JUnit), prompts optimaux, couverture vs qualité réelle
    - Tests d'intégration : génération à partir d'une architecture en couches, mocking et fixtures générés par IA
    - Mutation testing assisté : détecter les tests faibles ; agent de test en boucle : générer, exécuter, corriger, régénérer
    - Documentation automatique : docstrings (Google, NumPy, reST), OpenAPI/Swagger depuis le code, ADR et README générés, synchronisation doc-code par hook pre-commit
    - Agent de revue de code : lire, analyser, évaluer (style, logique, sécurité, performance), commenter ; intégration git diff, sortie SARIF/JSON, commentaires inline GitLab/GitHub
    - Cas pratique appliqué : Cas pratique sur une API REST Flask avec base de données, puis lab Agent TDD complet : de la user story aux tests validés par le développeur, à l'implémentation par l'agent et à la correction : un workflow TDD IA-assisté de bout en bout.
- Cartographie et indexation de codebase (J8, 7 h)
    - Techniques de repo-map : tree-sitter pour l'AST, extraction de signatures
    - Chunking intelligent du code (par fichier, classe, fonction) et embeddings de code
    - Stockage vectoriel et requêtes sémantiques : retrouver la logique métier (« où est la gestion des erreurs HTTP? »)
    - Agent de détection de bugs : analyser les stack traces, corréler avec le code, proposer un correctif
    - Refactoring multi-fichiers supervisé : renommage, extraction de module, injection de dépendances et validation par les tests avant/après, conflits de merge assistés
    - Cas pratique appliqué : Lab agent d'analyse de legacy : sur un projet Python/JS fourni, l'agent produit cartographie, liste des problèmes (bugs, dette technique, failles), plan de refactoring priorisé et correctifs pour les trois premiers, revue humaine et validation.
- CI/CD agentique & pipelines IA (J9, 7 h)
    - Architecture d'un pipeline IA : étapes déterministes + étapes agentiques ; que déléguer à l'IA — revue de code, tests manquants, sécurité (SAST), changelog, vérification de la documentation
    - Implémentation GitLab CI et GitHub Actions : job de revue IA déclenché sur merge request, job de génération de tests, Claude Code en mode headless dans un job CI
    - Conditions d'échec : bloquer le merge si la qualité est insuffisante ; sécurisation des clés API, runner self-hosted pour maîtriser les coûts
    - Human-in-the-loop et sécurité des agents CI : approbation humaine avant merge, permissions lecture vs écriture, journalisation des décisions, gestion des faux positifs et des secrets
    - Cas pratique appliqué : Lab pipeline CI/CD complet : lint → tests → agent IA (revue + sécurité + changelog) → build → déploiement en staging. L'agent poste ses retours en commentaire de merge request ; le merge est conditionné à l'approbation humaine du rapport.
- Projet final, agent autonome de développement (J10, 7 h)

- Déploiement et bonnes pratiques : packaging d'un agent (Docker, variables d'environnement, secrets), monitoring (tokens, taux de succès, latence), gestion des coûts et budgets, versioning du prompt système, onboarding de l'agent dans une équipe
- Implémentation du projet : chaque participant ou groupe construit un agent codeur sur un sujet au choix — migration de code, génération d'API REST depuis un schéma, agent de sécurité (scan + correction), traduction de stack
- Objectif livrable : un agent fonctionnel, testé, documenté
- Soutenance et bilan : démonstration devant le groupe, retours du formateur, bilan des 10 jours, roadmap personnelle et ressources pour continuer
- Cas pratique appliqué : Projet final avec soutenance : implémentation d'un agent autonome de développement, démontré en direct devant le groupe et évalué sur son fonctionnement réel.

Organisation de la formation

Equipe pédagogique

Formation animée par Christophe Ancey, formateur praticien qui utilise quotidiennement les agents de coding sur des projets réels, du prototype à la production. Il a conçu ce parcours à partir de sa propre pratique : mêmes outils, mêmes pipelines, mêmes garde-fous que ceux qu'il déploie en mission.

Moyens pédagogiques et techniques

- Supports pédagogiques commentés et partagés à distance
- Apports méthodologiques structurés
- Templates prêts à l'emploi
- Cas pratiques appliqués à l'entreprise
- Etudes de cas concrets
- Quiz
- Mise à disposition en ligne de documents supports à la suite de la formation
- Une semaine avant le début de la formation, nous donnons, à chaque participant, un accès sécurisé à notre plate-forme de formation en ligne (LMS : Learning Management System). L'apprenant accède via son extranet au livret d'accueil, au tuto de l'espace apprenant et au règlement intérieur.

Dispositif de suivi de l'exécution de l'évaluation des résultats de la formation

- Feuilles de présence
- QCM et questions orales ou écrites
- Évaluation continue par le formateur sur les cas pratiques, exercices, TP et mises en situation
- Projet final avec soutenance
- Formulaires d'évaluation de la formation
- Certificat de réalisation de l'action de formation transmis à la fin de la formation
- 10 jours avant l'entrée en formation : Nous transmettons à chaque stagiaire un questionnaire de préparation de la formation permettant d'adapter au mieux la formation à ses besoins.
- Pendant la formation : Le contrôle de connaissances et des acquis se font de façon continue via des quiz en ligne avec notation instantanée et des cas pratiques à réaliser. En fin de formation, les stagiaires restituent leurs travaux avec feedback du formateur.
- A l'issue de la formation : Nous adressons aux stagiaires un questionnaire de satisfaction à chaud et d'auto-diagnostic. Nous mesurons ainsi leur satisfaction et constatons directement l'évolution des compétences en comparaison avec les données recueillies avant la formation. Nous adressons un questionnaire d'évaluation au formateur.
- 15 jours après la fin de la formation : Nous adressons un questionnaire d'évaluation au financeur (OPCO).
- Un mois après la formation : Nous adressons un questionnaire d'évaluation aux différentes parties prenantes : commanditaire /manager.

- 3 mois après la formation : Nous transmettons au stagiaire un questionnaire d'évaluation à froid pour évaluer l'impact de la formation dans son activité professionnelle.

Tarif de la formation

À partir de 15400.00 €HT
TVA non applicable, art. 261-4-4 du CGI