EduGroupe
Accompagner pour réussir

Claude Code pour les développeurs

Référence : **IA036** Durée : **2 jours (14 heures)** Certification : **Aucune**

CONNAISSANCES PRÉALABLE

- Avoir une expérience significative dans un langage de programmation et savoir prompter une IA Générative

PROFIL DES STAGIAIRES

- Développeurs, DevOps, architectes logiciels et chefs de projets techniques souhaitant utiliser Claude Code comme assistant de développement

OBJECTIFS

- À l'issue de cette formation, les participants seront capables de :
- Installer et configurer Claude Code
- Utiliser efficacement l'assistant en ligne de commande
- Automatiser des tâches de développement complexes
- Configurer des hooks et des workflows personnalisés
- Intégrer Claude Code dans votre environnement de développement

CERTIFICATION PRÉPARÉE

- Aucune

MÉTHODES PÉDAGOGIQUES

- Mise à disposition d'un poste de travail par stagiaire
- Remise d'une documentation pédagogique papier ou numérique pendant le stage
- La formation est constituée d'apports théoriques, d'exercices pratiques et de réflexions
- Le suivi de cette formation donne lieu à la signature d'une feuille d'émargement

FORMATEUR

- Consultant-formateur expert de Claude Code

MÉTHODES D'ÉVALUATION DES ACQUIS

- Auto-évaluation des acquis par le stagiaire via un questionnaire
- Attestation des compétences acquises envoyée au stagiaire
- Attestation de fin de stage adressée avec la facture

ACCESSIBILITÉ DE LA FORMATION

- EduGroupe met en place un ensemble de dispositifs pour accueillir, accompagner et adapter ses formations aux personnes en situation de handicap (PSH).
- Notre référent(e) handicap se tient à votre disposition au 01.71.19.70.30 ou par mail à referent.handicap@edugroupe.com pour tout besoin d'aménagement, afin de vous offrir la meilleure expérience possible.

CONTENU DU COURS

1. Jour 1 - Matin

2. Introduction à Claude Code

- Présentation de Claude et des modèles Anthropic
- Installation et configuration initiale
- Interface en ligne de commande
- Fichiers CLAUDE.md et configuration projet
- Gestion des permissions et sécurité
- 💡 *Exemples de travaux pratiques (à titre indicatif) : Installer Claude Code CLI, s'authentifier, vérifier claude --version et le modèle actif ; Rédiger un CLAUDE.md projet : stack, conventions de code, branche par défaut, règles critiques ; Ajouter un CLAUDE.md utilisateur avec préférences personnelles ; Configurer settings.json : les permissions (allow/ask/deny), les outils autorisés ; Tester une requête sur le codebase et vérifier que les règles du CLAUDE.md sont respectées*

3. Jour 1 - Après-midi

4. Utilisation quotidienne

- Navigation dans le codebase
- Lecture et compréhension de code
- Génération et modification de fichiers
- Commandes Git intégrées
- Exécution de commandes shell
- Gestion du contexte (/compact, /clear, /context) et mémoire
- 💡 *Exemples de travaux pratiques (à titre indicatif) : Utiliser Claude pour explorer le codebase (Glob, Grep, Read) sans éditer ; Demander un diagnostic : reproduction du bug, identification de la cause racine ; Laisser Claude proposer un correctif ciblé, le relire, ajuster manuellement si besoin ; Utiliser /compact à mi-session, puis /clear avant une seconde tâche*

5. Jour 2 - Matin

6. Fonctionnalités avancées

- Hooks : pre-tool-use, post-tool-use, notification
- Slash commands personnalisées (/commit, /review...)
- Mode plan pour les tâches complexes
- Agents spécialisés et task tool
- Intégration MCP (Model Context Protocol)
- Configuration des serveurs MCP
- 💡 *Exemples de travaux pratiques (à titre indicatif): Créer une slash command /review qui lance une revue de code sur les fichiers modifiés ; Créer une slash command /changelog qui met à jour CHANGELOG.md à partir du git diff ; Configurer un hook PostToolUse sur Edit|Write qui lance un linter sur les fichiers modifiés ; Configurer un hook UserPromptSubmit qui injecte la date du jour dans le contexte ; Installer un serveur MCP et utiliser une ressource MCP dans une requête (ex : lister des fichiers distants via le serveur)*

7. Jour 2 - Après-midi

8. Workflow Explore, Plan, Code, Commit

- Le cycle agentique : explore > plan > code > commit > PR/MR
- Debugging assisté par Claude
- Refactoring de code legacy
- Génération de tests
- Documentation automatique
- Intégration IDE (VS Code, JetBrains)
- Limites, coûts et optimisation des tokens
- Sécurité : secrets, permissions, sandbox
- 💡 *Exemples de travaux pratiques (à titre indicatif) : Utiliser le workflow Explore, Plan, Code, Commit sur une feature (ex : "ajouter pagination à un endpoint REST" ou "convertir un test Jest en Vitest") ; Faire générer les tests manquants et les exécuter*