Anthropic Academy

Résumé complet et détaillé de tous les cours

Cours	Statut	Leçons	Pages
1. Claude Code in Action	**Complété**	21/21	2-8
2. Claude 101	**Complété**	14/14	9-16
3. Introduction to Claude Cowork	**Complété**	11/11	17-23
4. AI Fluency: Framework & Foundations	**Complété**	15/15	24-32
5. Building with the Claude API	**En cours (61%)**	52/85	33-44

Apprenant : Yoann Boulch | **Date :** 25 mars 2026 | **Plateforme :** anthropic.skilljar.com

Chapitre 1 — Claude Code in Action

Intégrer Claude Code dans votre workflow de développement (21 leçons, 100%)

1.1 Qu'est-ce que Claude Code ?

Claude Code est un outil de codage agentique en ligne de commande qui permet aux développeurs de déléguer des tâches de programmation à Claude directement depuis leur terminal. Contrairement aux assistants de code classiques intégrés dans un IDE (type Copilot), Claude Code fonctionne de manière autonome : il lit les fichiers de votre projet, comprend l'architecture, exécute des commandes, et effectue des modifications multi-fichiers en toute autonomie.

Architecture d'un assistant de codage

Un assistant de codage IA repose sur une architecture en boucle : l'utilisateur formule une demande, le modèle analyse le contexte disponible (fichiers, historique, documentation), génère un plan d'action, puis exécute les modifications nécessaires via des outils intégrés. Claude Code se distingue par son approche agentique : il ne se contente pas de générer du texte, il **agit** sur votre codebase.

Figure 1.1 — Flux d'exécution de Claude Code : de la requête utilisateur à la modification du code

Différences avec les assistants intégrés à l'IDE

Caractéristique	Assistant IDE (ex: Copilot)	Claude Code
Mode d'interaction	Complétion inline dans l'éditeur	Terminal en ligne de commande
Portée des modifications	Fichier actif uniquement	Multi-fichiers, projet complet
Autonomie	Suggestion passive	Agent autonome avec planification
Utilisation d'outils	Limitée (complétion)	Lecture/écriture fichiers, shell, git, MCP
Gestion du contexte	Fenêtre de contexte limitée	Fichier CLAUDE.md + navigation intelligente
Intégration CI/CD	Non	Oui (GitHub Actions, hooks)

Tableau 1.1 — Comparaison entre un assistant IDE classique et Claude Code

1.2 Installation et configuration

L'installation de Claude Code est rapide et compatible macOS, Linux et WSL :

```
npm install -g @anthropic-ai/claude-code
```

Après installation, lancez `claude` dans votre terminal. Lors du premier lancement, vous serez invité à vous authentifier via votre compte Anthropic. Claude Code supporte également AWS Bedrock et Google Cloud Vertex AI pour les déploiements enterprise.

Configuration du projet

Une fois dans un répertoire de projet, Claude Code analyse automatiquement la structure. Le fichier `CLAUDE.md` à la racine du projet sert de mémoire persistante : il contient les conventions du projet, l'architecture, les commandes de build, et toute information que Claude doit connaître pour travailler efficacement.

Exemple de fichier CLAUDE.md :

```
# Projet: API de gestion de commandes ## Architecture - Backend: Python FastAPI - Base de données: PostgreSQL avec SQLAlchemy - Tests: pytest avec fixtures ## Commandes - Build: make build - Tests: pytest tests/ -v - Lint: ruff check .
```

1.3 Ajouter du contexte et effectuer des modifications

Claude Code utilise plusieurs sources de contexte pour comprendre votre projet. La gestion intelligente du contexte est cruciale car la fenêtre de contexte du modèle est limitée. Claude Code charge sélectivement les fichiers pertinents plutôt que l'ensemble du projet.

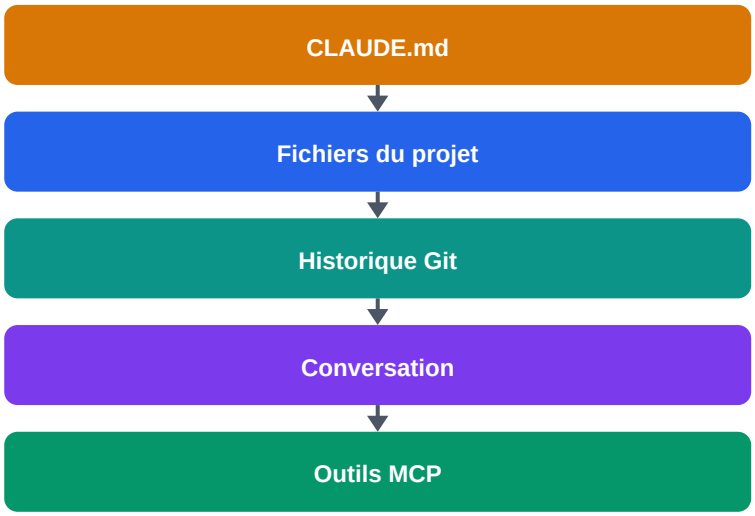

Figure 1.2 — Sources de contexte exploitées par Claude Code

Contrôler le contexte

Plusieurs mécanismes permettent de contrôler le contexte :

- **Commande /compact :** Résume la conversation pour libérer de l'espace dans la fenêtre de contexte.
- **Mentions @ :** Référencez explicitement un fichier avec @nom_fichier pour le charger dans le contexte.
- **Fichiers .claudeignore :** Similaire à .gitignore, exclut des fichiers/dossiers du contexte.
- **Navigation intelligente :** Claude Code explore le projet à la demande via grep, find, et lecture de fichiers.

1.4 Commandes personnalisées (Custom Commands)

Les commandes personnalisées permettent de créer des raccourcis réutilisables pour des workflows récurrents. Elles sont définies dans le dossier `.claude/commands/` sous forme de fichiers Markdown.

Exemple : commande de revue de code

```
# .claude/commands/review.md Analyse le diff git entre la branche actuelle et main. Pour chaque fichier modifié : 1. Vérifie la cohérence avec l'architecture du projet 2. Identifie les bugs potentiels 3. Suggère des améliorations de performance 4. Vérifie la couverture de tests Produis un rapport structuré avec priorité (critique/important/mineur).
```

Une fois créée, cette commande s'exécute via `/review` dans le terminal Claude Code.

1.5 Serveurs MCP (Model Context Protocol)

Le Model Context Protocol (MCP) est un standard ouvert qui permet à Claude Code de se connecter à des outils et services externes. Chaque serveur MCP expose des outils que Claude peut invoquer de manière autonome pendant son travail.

Exemples de serveurs MCP courants :

Serveur MCP	Fonction	Cas d'usage
Playwright	Automatisation navigateur	Tests E2E, scraping, vérification visuelle
PostgreSQL	Accès base de données	Requêtes SQL, migrations, exploration de schéma
GitHub	Gestion de dépôts	Création de PR, revue, gestion d'issues
Filesystem	Accès fichiers avancé	Opérations sur fichiers hors projet
Slack	Messagerie	Envoi de notifications, lecture de canaux

Tableau 1.2 — Serveurs MCP fréquemment utilisés avec Claude Code

1.6 Intégration GitHub

Claude Code s'intègre nativement avec GitHub pour automatiser les workflows de développement. Il peut créer des branches, commiter du code, ouvrir des pull requests, et même effectuer des revues de code automatisées dans le cadre de GitHub Actions.

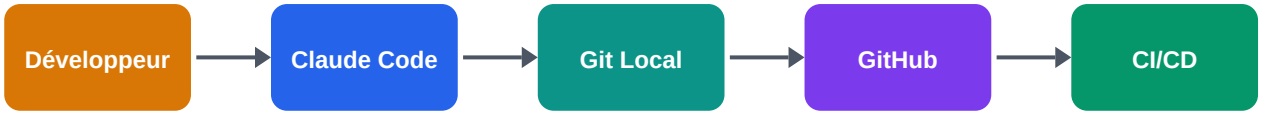

Figure 1.3 — Pipeline d'intégration Claude Code + GitHub

1.7 Les Hooks : automatisation événementielle

Les hooks sont des scripts qui s'exécutent automatiquement en réponse à des événements dans Claude Code. Ils permettent d'ajouter des validations, des transformations, ou des notifications à chaque action sans intervention manuelle.

Types de hooks disponibles :

Type de Hook	Déclencheur	Cas d'usage typique
PreToolUse	Avant l'exécution d'un outil	Valider les commandes avant exécution
PostToolUse	Après l'exécution d'un outil	Logger les résultats, formater la sortie
Notification	Messages importants	Alertes Slack/email lors d'erreurs
Stop	Fin de session	Nettoyage, rapport de session

Tableau 1.3 — Types de hooks dans Claude Code

Exemple de hook PreToolUse :

```
{ "hooks": { "PreToolUse": [{ "matcher": "Bash", "hooks": [{ "type": "command", "command": "echo \"Commande détectée: $TOOL_INPUT\"" }] }] } }
```

Ce hook intercepte chaque commande Bash avant son exécution, permettant une validation ou un logging.

Pièges courants (Gotchas)

- **Timeouts :** Les hooks ont un délai d'exécution limité. Les scripts longs doivent être asynchrones.
- **Boucles infinies :** Un hook qui déclenche un outil qui déclenche le même hook = boucle. Utilisez des conditions de garde.
- **Permissions :** Les hooks héritent des permissions de Claude Code. Attention aux opérations destructives.
- **Ordre d'exécution :** Quand plusieurs hooks matchent, ils s'exécutent dans l'ordre de déclaration.

1.8 Le SDK Claude Code

Le SDK Claude Code permet d'utiliser Claude Code de manière programmatique dans vos propres scripts et applications. Il expose une API TypeScript/JavaScript qui donne accès à toute la puissance de Claude Code : exécution de tâches, gestion de conversations, et orchestration multi-agents.

Exemple d'utilisation du SDK :

```
import { claude } from "@anthropic-ai/claude-code"; const result = await claude({ prompt: "Ajoute des tests unitaires pour le module auth", options: { maxTurns: 10, model: "claude-sonnet-4-6" } }); console.log(result.output);
```

Points clés à retenir

Claude Code transforme le développement logiciel en passant d'un paradigme d'assistance passive (complétion de code) à un paradigme d'agent autonome capable de comprendre, planifier et exécuter des tâches complexes de bout en bout. La maîtrise du contexte (CLAUDE.md), des commandes personnalisées, des hooks et du SDK ouvre la voie à des workflows hautement automatisés.

Chapitre 2 — Claude 101

Apprendre à utiliser Claude pour le travail quotidien (14 leçons, 100%)

2.1 Qu'est-ce que Claude ?

Claude est un assistant IA développé par Anthropic, conçu pour être **utile, inoffensif et honnête** (helpful, harmless, honest). Contrairement à un simple chatbot, Claude est un partenaire de réflexion capable d'une grande variété de tâches conversationnelles et de traitement de texte tout en maintenant la nuance et la sécurité.

Principes fondamentaux

- **Helpful (Utile) :** Claude cherche à fournir des réponses pertinentes, détaillées et actionnables.
- **Harmless (Inoffensif) :** Claude est guidé par des principes qui lui font éviter les contenus toxiques ou discriminatoires.
- **Honest (Honnête) :** Claude reconnaît ses limites et ne prétend pas savoir ce qu'il ne sait pas.

Capacités principales

Capacité	Description	Exemple
Rédaction	Contenu créatif, emails, rapports	Rédiger un email de suivi client professionnel
Recherche et analyse	Explorer des angles, compiler des résultats	Analyser les tendances d'un marché
Assistance au code	Génération, débogage, explication	Trouver un bug dans une fonction Python
Résolution de problèmes	Raisonnement complexe, stratégie	Élaborer un plan de migration de données
Apprentissage	Exploration de domaines, tutoriels	Expliquer les concepts de Kubernetes

Tableau 2.1 — Capacités principales de Claude

Moyens d'accès à Claude

Plateforme	Type	Cas d'usage
Claude.ai (web/mobile/desktop)	Interface conversationnelle	Usage quotidien, tous profils
Claude Code	Terminal en ligne de commande	Développeurs, ingénierie
Claude + Slack	Intégration messagerie	Collaboration d'équipe
Claude pour Excel	Sidebar dans Excel	Analyse de données, tableurs
API Anthropic	Intégration programmatique	Applications sur mesure

Tableau 2.2 — Plateformes d'accès à Claude

2.2 Votre première conversation avec Claude

Interagir avec Claude se fait comme avec un collègue compétent : de manière naturelle, concise et contextuelle. La qualité de la réponse dépend directement de la qualité du prompt.

Le framework de prompting en 3 étapes

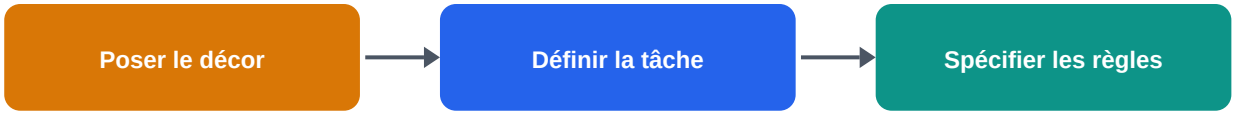

Figure 2.1 — Les trois étapes d'un prompt efficace

Étape	Question à se poser	Exemple
1. Poser le décor (Setting the stage)	Quel est votre rôle ? Quel contexte Claude doit-il connaître ?	"Je suis responsable marketing d'une startup de streaming"
2. Définir la tâche (Defining the task)	Quelle action Claude doit-il accomplir ?	"Recherche le marché du streaming : tendances, concurrents, opportunités"
3. Spécifier les règles (Specifying rules)	Quel format, ton, contraintes ?	"Utilise des sources web récentes, formate en rapport professionnel"

Tableau 2.3 — Framework de prompting en 3 étapes

Exemple complet de prompt :

```
"Je suis le responsable marketing d'une startup de streaming indépendant, et nous
préparons un pitch deck pour les investisseurs. Effectue une recherche sur le marché du
streaming : tendances actuelles, principaux concurrents, et opportunités pour les
plateformes de niche. Utilise des sources web actuelles avec citations, et structure le
résultat en rapport professionnel."
```

2.3 Obtenir de meilleurs résultats

Défis courants et solutions

Défi	Cause	Solution
Réponse trop générique	Pas assez de contexte	Ajoutez des détails sur votre audience, rôle, contraintes
Réponse trop longue/courte	Claude devine la longueur	Soyez explicite : "en 2 paragraphes" ou "moins de 100 mots"
Format incorrect	Claude comprend le quoi mais pas le comment	Montrez un exemple du format souhaité
Informations incorrectes	Hallucination sur des faits précis	Demandez des citations, vérifiez indépendamment
Ton inadapté	Claude par défaut est professionnel et neutre	Décrivez le ton : "conversationnel", "autoritaire"

Tableau 2.4 — Défis courants et stratégies de résolution

L'état d'esprit itératif

Le premier prompt produit rarement le résultat parfait. Les utilisateurs efficaces de Claude :

- **Traitent les premiers résultats comme des brouillons** et les affinent progressivement.
- **Donnent un feedback spécifique** : "Raccourcis les 2 premiers paragraphes et rends la conclusion plus orientée action" est plus utile que "fais mieux".

• **Savent quand recommencer** : si la conversation dérive, ouvrir un nouveau chat avec un prompt plus clair est souvent plus rapide.

Le framework 4D pour l'AI Fluency

Le cours introduit le framework 4D (détaillé au chapitre 4) : Delegation, Description, Discernment, Diligence. Ce cadre méthodologique structure la collaboration humain-IA de manière efficace, éthique et sécurisée.

Évaluations (Evals)

Pour intégrer Claude dans des workflows récurrents, les évaluations systématiques sont essentielles :

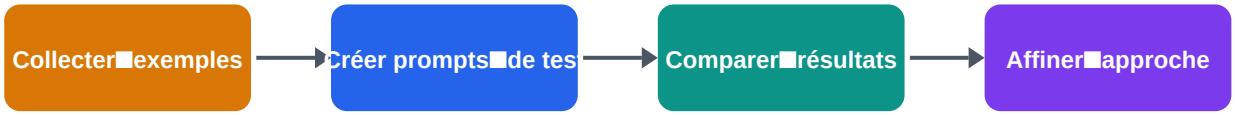

Figure 2.2 — Workflow d'évaluation simple en 4 étapes

2.4 L'application Claude Desktop : Chat, Cowork, Code

L'application Claude Desktop offre trois modes d'interaction distincts, chacun adapté à un type de travail différent :

Mode	Public cible	Type d'interaction	Exemple
Chat	Tous	Conversationnel, Q&A	Brainstorming, rédaction, analyse
Cowork	Knowledge workers	Tâches sur fichiers, agentique	Créer un deck, analyser des données
Code	Développeurs	Terminal, coding assisté	Refactoring, tests, déploiement

Tableau 2.5 — Les trois modes de Claude Desktop

2.5 Les Projets : espaces de travail persistants

Les projets dans Claude sont des espaces de travail autonomes avec leur propre mémoire, historique de conversations, base de connaissances et instructions personnalisées. Ils fonctionnent comme des espaces de travail dédiés où Claude conserve le contexte entre les sessions.

Création d'un projet en 3 étapes

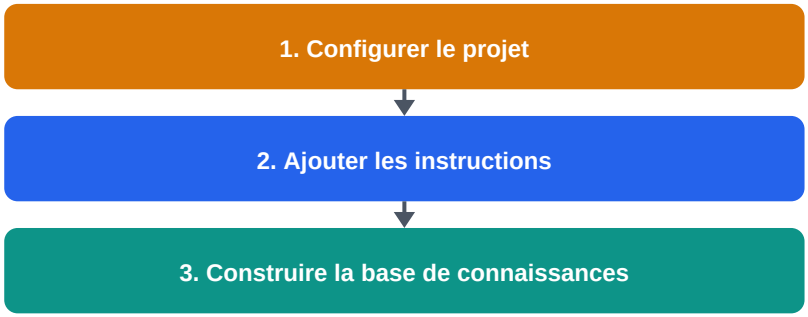

Figure 2.3 — Les 3 étapes de création d'un projet Claude

Instructions de projet efficaces :

- **Contexte :** "Ce projet concerne la création de contenu marketing pour notre logiciel B2B."
- **Processus :** "D'abord propose une structure de blog engageante, puis rédige le brouillon."
- **Ton :** "Utilise un ton professionnel mais conversationnel. Évite le jargon."
- **Contraintes :** "Toujours inclure un appel à l'action en fin de contenu marketing."

Gestion de la scalabilité

Lorsque la base de connaissances approche les limites de la fenêtre de contexte, Claude active automatiquement le mode RAG (Retrieval Augmented Generation). Ce mécanisme permet à Claude de rechercher et récupérer sélectivement les informations les plus pertinentes parmi tous les documents du projet, sans dégradation de performance.

2.6 Artifacts, Skills, Connectors et Research

Artifacts

Les Artifacts sont des créations interactives générées par Claude : applications React, visualisations de données, documents formatés, diagrammes. Ils apparaissent dans un panneau dédié à droite de la conversation et sont directement utilisables, modifiables et partageables.

Skills

Les Skills sont des instructions réutilisables en format Markdown qui personnalisent le comportement de Claude. Elles peuvent être créées manuellement ou installées depuis la communauté. Exemples : style de rédaction spécifique, workflow d'analyse de données, processus de revue de code.

Connectors (MCP)

Les connecteurs permettent à Claude d'accéder à vos outils externes : Google Drive, Slack, Jira, bases de données, etc. Ils utilisent le protocole MCP (Model Context Protocol) pour une intégration sécurisée et standardisée.

Research Mode

Le mode Recherche permet à Claude de parcourir le web en profondeur pour répondre à des questions complexes nécessitant des données actuelles. Claude explore plusieurs sources, croise les informations et produit un rapport structuré avec citations.

> Claude 101 fournit les bases essentielles pour utiliser Claude au quotidien. La maîtrise du prompting structuré (3 étapes), de l'itération, des projets et des outils avancés (Artifacts, Skills, Connectors) permet de transformer Claude d'un simple chatbot en un véritable collaborateur adapté à votre contexte professionnel.

Chapitre 3 — Introduction to Claude Cowork

Travailler directement sur vos fichiers et projets (11 leçons, 100%)

3.1 Qu'est-ce que Cowork ?

Cowork est le mode agentique de Claude Desktop qui change le paradigme d'interaction : au lieu d'un échange conversationnel (question-réponse), vous **déléguez une tâche complète**. Cowork planifie, exécute et produit des livrables concrets en travaillant directement sur vos fichiers.

Cowork est construit sur la même architecture que Claude Code — le système agentique utilisé pour écrire et livrer du code en production. Cette capacité est désormais accessible pour tout type de travail de connaissance.

Chat vs Cowork

Aspect	Mode Chat	Mode Cowork
Paradigme	Conversation	Délégation de tâche
Input	Question ou demande	Description d'un résultat attendu
Output	Texte dans la conversation	Fichiers réels (PPTX, DOCX, XLSX...)
Durée	Réponse instantanée	Minutes (tâches longues possibles)
Accès fichiers	Upload/download	Lecture/écriture directe sur votre disque
Outils externes	Non	Oui via connecteurs MCP
Planification	Non	Plan visible avant exécution

Tableau 3.1 — Comparaison entre le mode Chat et le mode Cowork

Les 3 piliers de Cowork

Figure 3.1 — Les trois piliers de Cowork : Planifier, Exécuter, Connecter

- **Plan :** Pour les tâches multi-étapes, Claude montre son approche avant de commencer. Vous le validez, l'ajustez, puis donnez le feu vert.
- **Execute :** Le travail s'exécute dans un environnement isolé sur votre machine. Création de fichiers, analyse, calculs — tout est automatisé.
- **Connect :** Cowork accède à vos outils existants : email, drives partagés, outils de productivité via les connecteurs MCP.

6 capacités fondamentales

Capacité	Description
Connecteurs	Accès aux outils existants (email, drives, messagerie)
Opérations sur fichiers	Lire, éditer, créer des fichiers réels (PPTX, XLSX, DOCX, PDF)
Plugins	Expertise spécialisée par domaine (ventes, data, légal...)
Tâches planifiées	Automatisation récurrente (daily/weekly/cron)
Sous-agents	Parallélisation des tâches indépendantes
Calcul local	Exécution de code directement sur vos fichiers

Tableau 3.2 — Les 6 capacités fondamentales de Cowork

3.2 La boucle de tâche (Task Loop)

Le cœur de Cowork est une boucle en 4 étapes que vous répéterez pour chaque tâche :

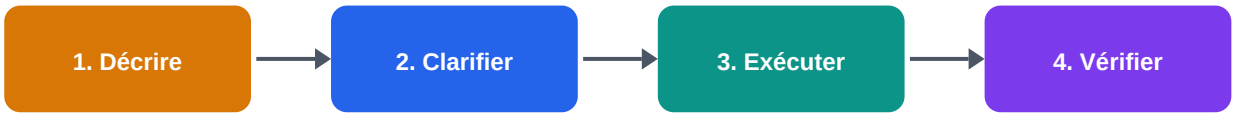

Figure 3.2 — La boucle de tâche Cowork en 4 étapes

Étape 1 : Décrire ce que vous voulez

Un bon prompt Cowork donne trois informations : **ce qu'il faut regarder** (vos fichiers sources), **ce que vous voulez en retour** (le livrable), et **où le placer** (le dossier de destination).

```
"Trois de nos entreprises suivies ont publié leurs résultats cette semaine. Les
transcripts sont dans mon dossier avec les notes des analystes. Crée un résumé exécutif
d'une page pour chacune, mettant en avant les surprises et les changements de guidance.
Sauvegarde dans le dossier Earnings."
```

Étape 2 : Répondre aux questions de clarification

Cowork pose quelques questions pour affiner le résultat — quel format, quel niveau de détail, quelles priorités. C'est l'équivalent du brief qu'un collègue demanderait avant de se lancer.

Étape 3 : Laisser travailler (ou intervenir)

Un panneau de progression montre chaque étape : quels fichiers sont lus, ce qui est construit. Pour les tâches longues, Cowork peut lancer des **sous-agents** qui travaillent en parallèle. Vous pouvez intervenir à tout moment en tapant dans le chat pour rediriger.

Étape 4 : Ouvrir et vérifier le travail fini

Le résultat est un vrai fichier dans votre dossier. Traitez-le comme un premier brouillon d'un collègue compétent : la structure sera bonne, le contenu solide, mais il pourra nécessiter votre touche personnelle.

3.3 Plugins : Cowork spécialisé

Les plugins ajoutent une expertise de domaine à Cowork. Chaque plugin combine des skills (instructions spécialisées), des connecteurs MCP, et des workflows pré-configurés.

Exemples de plugins :

- **Plugin Data Analysis :** Analyse de données avec Python, visualisations, dashboards interactifs.
- **Plugin Sales :** Préparation de rendez-vous, analyse de pipeline CRM, rapports d'activité.
- **Plugin Legal :** Analyse de contrats, comparaison de clauses, résumés juridiques.
- **Plugin Content :** Création de contenu multi-canal, adaptation de ton, planification éditoriale.

3.4 Tâches planifiées (Scheduled Tasks)

Les tâches planifiées permettent d'automatiser des workflows récurrents. Elles s'exécutent automatiquement selon un calendrier défini (cron) et peuvent produire des rapports, effectuer des analyses régulières, ou synchroniser des données.

Exemple : rapport hebdomadaire automatique

```
Tâche : "Chaque lundi à 9h, analyse les données de vente de la semaine précédente dans le
dossier Sales/. Génère un rapport PPTX avec les KPIs principaux, les tendances et les
alertes. Sauvegarde dans Reports/."
```

3.5 Cas pratiques

Gestion de fichiers et documents

Cowork excelle dans les tâches documentaires : transformer un dossier de notes désorganisées en présentation structurée, convertir des données brutes en rapports formatés, créer des documents professionnels (PPTX, DOCX, XLSX) à partir de sources multiples.

Recherche et analyse à grande échelle

Grâce aux sous-agents et à la recherche web, Cowork peut mener des analyses approfondies en parallélisant le travail. Par exemple, analyser 50 transcripts d'appels de résultats financiers simultanément et produire un rapport consolidé.

3.6 Permissions, usage et choix de modèle

Niveau	Actions autorisées
Lecture seule	Lire les fichiers, analyser le contenu
Lecture + écriture	Créer et modifier des fichiers dans le dossier sélectionné
Suppression	Supprimer des fichiers (demande confirmation explicite)
Outils externes	Accéder aux connecteurs MCP (selon configuration)

Tableau 3.3 — Niveaux de permissions dans Cowork

Choix du modèle

Modèle	Cas d'usage	Coût relatif
Claude Haiku 4.5	Tâches simples, rapides, volume élevé	Faible
Claude Sonnet 4.6	Équilibre performance/coût, usage quotidien	Moyen
Claude Opus 4.6	Tâches complexes, raisonnement avancé	Élevé

Tableau 3.4 — Guide de sélection de modèle dans Cowork

Cowork transforme Claude d'un assistant conversationnel en un véritable agent de travail qui manipule vos fichiers, se connecte à vos outils, et produit des livrables réels. La boucle Décrire → Clarifier → Exécuter → Vérifier, combinée aux plugins et tâches planifiées, permet d'automatiser une grande partie du travail de connaissance.

Chapitre 4 — AI Fluency: Framework & Foundations

Collaborer efficacement, éthiquement et en sécurité avec l'IA (15 leçons, 100%)

4.1 Introduction à l'AI Fluency

L'AI Fluency ne se limite pas à savoir utiliser un outil IA — c'est la capacité à collaborer de manière **efficace, efficiente, éthique et sûre** avec les systèmes d'intelligence artificielle. Ce cours, développé par les professeurs Rick Dakan (Ringling College) et Joseph Feller (University College Cork) en partenariat avec Anthropic, propose un cadre structuré : le **framework 4D**.

Pourquoi l'AI Fluency est essentielle

Trois modes de collaboration avec l'IA émergent, chacun nécessitant des compétences différentes :

Mode	Description	Rôle de l'humain	Exemple
Automation	L'IA exécute des tâches répétitives de façon autonome	Supervision, evalidation	Tri automatique d'emails
Augmentation	L'IA amplifie les capacités humaines	Direction, jugement	Analyse assistée de données
Agency	L'IA prend des décisions dans un cadre défini	Cadrage, audit	Agent de recherche autonome

Tableau 4.1 — Les trois modes de collaboration humain-IA

4.2 Le Framework 4D

Le framework 4D est le cœur méthodologique du cours. Il définit quatre compétences interdépendantes pour une collaboration IA efficace :

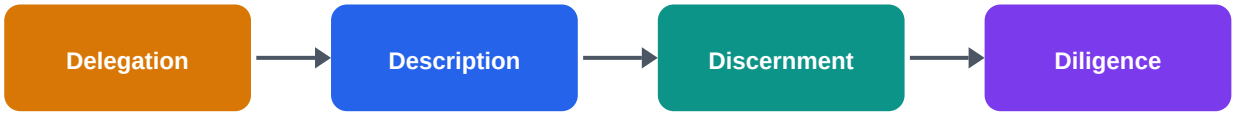

Figure 4.1 — Le Framework 4D pour l'AI Fluency

Compétence	Définition	Question clé
Delegation	Décider quelles tâches confier à l'IA vs garder pour l'humain	"Cette tâche est-elle adaptée à l'IA ?"
Description	Communiquer efficacement avec l'IA (prompting)	"Comment formuler ma demande clairement ?"
Discernment	Évaluer de manière critique les résultats de l'IA	"Ce résultat est-il fiable et complet ?"

Diligence	Utiliser l'IA de manière éthique et responsable	"Est-ce que j'utilise l'IA de façon éthique ?"

Tableau 4.2 — Les 4 compétences du framework

4.3 Fondamentaux de l'IA Générative

Avant d'appliquer le framework 4D, il est essentiel de comprendre comment fonctionne l'IA générative et quelles sont ses capacités et limitations.

Comment fonctionne un LLM

Un Large Language Model (LLM) comme Claude est un réseau de neurones entraîné sur de vastes corpus de textes. Il prédit le token (mot/sous-mot) le plus probable en fonction du contexte précédent. Mais grâce à l'entraînement par renforcement à partir du feedback humain (RLHF), Claude va au-delà de la simple prédiction statistique pour produire des réponses utiles, pertinentes et sûres.

Capacités et limitations

Capacités	Limitations
Rédaction, résumé, reformulation	Peut halluciner des faits ou des citations
Analyse et raisonnement	Connaissance limitée par la date de coupure
Code, calcul, logique	Pas de compréhension "réelle" du monde
Adaptation au ton et au contexte	Peut être influencé par les biais des données
Multilingue	Performances variables selon les langues
Créativité et brainstorming	Pas de mémoire entre conversations (sauf projets)

Tableau 4.3 — Capacités vs limitations des LLMs

4.4 Delegation : savoir quoi confier à l'IA

La Delegation est la première compétence du framework. Elle repose sur trois axes de conscience :

- **Conscience du problème (Problem Awareness) :** Comprendre la nature de la tâche — est-elle bien définie ? Quels sont les critères de réussite ?
- **Conscience de la plateforme (Platform Awareness) :** Connaître les forces et faiblesses de l'outil IA — Claude excelle-t-il dans ce type de tâche ?
- **Conscience de la tâche (Task Awareness) :** Décomposer le travail en sous-tâches et décider lesquelles confier à l'IA vs garder pour soi.

Matrice de décision de delegation

	IA performante	IA moins performante
Tâche bien définie	Déléguer à l'IA (automatisation)	Humain principal, IA en support
Tâche ambiguë	IA en exploration, humain en validation	Garder pour l'humain (créativité, jugement)

Tableau 4.4 — Matrice de décision pour la delegation de tâches

4.5 Description : communiquer efficacement avec l'IA

La Description concerne la qualité de la communication avec l'IA. Elle se décompose en trois dimensions :

Dimension	Objet	Exemple de question
Description du produit (Product)	Définir le résultat attendu	"Quel format, ton, longueur, public cible ?"
Description du processus (Process)	Guider les étapes de travail	"D'abord analyse, puis structure, enfin rédige"
Description de la performance (Performance)	Définir les critères de succès	"Le résultat doit être factuellement exact et concis"

Tableau 4.5 — Les trois dimensions de la Description

6 techniques de prompting efficaces

Technique	Description	Exemple
Clarté et directivité	Être explicite et sans ambiguïté	"Résume en 3 bullet points"
Spécificité	Fournir des détails précis	"Pour un public de CTOs"
Tags XML	Structurer les inputs complexes	"<context>...</context>"
Exemples (few-shot)	Montrer le format attendu	"Voici un exemple de sortie..."
Rôle/persona	Définir l'expertise de Claude	"Tu es un expert en data engineering"
Chain-of-thought	Demander un raisonnement étape par étape	"Réfléchis étape par étape"

Tableau 4.6 — Les 6 techniques fondamentales de prompting

4.6 Discernment : évaluer les résultats de l'IA

Le Discernment est la capacité à évaluer de manière critique ce que l'IA produit. Comme pour la Description, il se décompose en trois dimensions :

- **Discernment du produit :** Le résultat est-il exact, complet, pertinent ? Y a-t-il des hallucinations ou des omissions ?
- **Discernment du processus :** L'IA a-t-elle suivi le bon raisonnement ? Les étapes intermédiaires sont-elles logiques ?
- **Discernment de la performance :** Le résultat atteint-il les critères de qualité définis ? Est-il exploitable en l'état ?

La boucle Description-Discernment

Figure 4.2 — La boucle itérative Description-Discernment

Cette boucle est le mécanisme central de travail avec l'IA. Chaque interaction affine le résultat : vous décrivez, l'IA produit, vous évaluez, vous fournissez du feedback, et le cycle recommence jusqu'à obtenir un résultat satisfaisant.

4.7 Diligence : utiliser l'IA de manière éthique et responsable

La Diligence couvre les aspects éthiques, légaux et sociaux de l'utilisation de l'IA. Elle se décline en trois axes :

Axe	Enjeu	Action recommandée
Diligence de création (Creation)	D'où viennent les données d'entraînement ?	Vérifier les sources, respecter les droits d'auteur
Diligence de transparence (Transparency)	Comment le résultat a-t-il été produit ?	Indiquer quand l'IA est utilisée, documenter le processus
Diligence de déploiement (Deployment)	Quels impacts sur les utilisateurs finaux ?	Tester les biais, prévoir les garde-fous

Tableau 4.7 — Les trois axes de la Diligence

Le framework 4D — Delegation, Description, Discernment, Diligence — fournit un cadre complet et structuré pour collaborer avec l'IA. Ces compétences ne sont pas techniques mais méthodologiques : elles restent pertinentes quel que soit l'outil IA utilisé et quelle que soit l'évolution de la technologie.

Chapitre 5 — Building with the Claude API

Développer avec l'API Anthropic — En cours (52/85 leçons, 61%)

5.1 Accès à l'API et premiers pas

L'API Claude permet d'intégrer l'intelligence de Claude dans vos applications. Le cours couvre l'ensemble du spectre, de l'authentification à la construction de systèmes agentiques.

Architecture d'une requête API

Figure 5.1 — Cycle de vie d'une requête à l'API Claude

Exemple de requête de base :

```
import anthropic client = anthropic.Anthropic() # API key from env message =
client.messages.create( model="claude-sonnet-4-6", max_tokens=1024, messages=[ {"role":
"user", "content": "Explique le CAP theorem"} ] ) print(message.content[0].text)
```

Conversations multi-tours

Les conversations multi-tours nécessitent de renvoyer l'historique complet à chaque requête. Le modèle est stateless — il ne conserve pas de mémoire entre les appels.

```
messages = [ {"role": "user", "content": "Qu'est-ce que Kafka ?"}, {"role": "assistant",
"content": "Apache Kafka est..."}, {"role": "user", "content": "Comment le configurer en
Python ?"} ] response = client.messages.create( model="claude-sonnet-4-6",
max_tokens=2048, system="Tu es un expert en data engineering.", messages=messages )
```

System Prompts et Température

Paramètre	Rôle	Valeurs typiques
system	Définir le comportement global	Instructions de rôle, ton, contraintes
temperature	Contrôler la créativité	0.0 (déterministe) à 1.0 (créatif)
max_tokens	Limiter la longueur de réponse	256 (court) à 8192 (long)
top_p	Filtrage par probabilité cumulative	0.9 (défaut) — rarement modifié

Tableau 5.1 — Paramètres clés de l'API

5.2 Évaluation de prompts (Prompt Evals)

L'évaluation systématique des prompts est essentielle en production. Le cours présente un workflow complet de test et d'amélioration des prompts.

Figure 5.2 — Workflow d'évaluation de prompts

Deux méthodes de grading

Méthode	Principe	Quand l'utiliser
Grading par code	Règles programmatiques (regex, contains, structure)	Réponses structurées, format strict
Grading par modèle	Un LLM évalue la réponse d'un autre LLM	Réponses ouvertes, qualité subjective

Tableau 5.2 — Méthodes de grading pour les évaluations

5.3 Techniques de Prompt Engineering

Le prompt engineering est l'art de formuler des instructions qui maximisent la qualité des réponses du modèle. Le cours couvre 5 techniques fondamentales :

Technique	Principe	Exemple API
Clarté et directivité	Instructions explicites sans ambiguïté	"Extrais les noms et dates. Retourne un JSON."
Spécificité	Détails précis sur le contexte et le format	"Pour un ingénieur data senior chez un éditeur SaaS"
Structure XML	Balises pour séparer les sections du prompt	"<context>...</context> <task>...</task>"
Exemples (few-shot)	Montrer input/output attendus	"Input: X -> Output: Y Input: A -> Output: ?"
Préfilling	Pré-remplir le début de la réponse	assistant: {"results": [

Tableau 5.3 — Techniques de prompt engineering pour l'API

Exemple complet avec tags XML :

```
system_prompt = """ Tu es un assistant d'analyse de données. Réponds toujours en JSON
structuré. """ user_prompt = """ Dataset: ventes Q4 2025, 15000 lignes Colonnes: date,
produit, montant, region Génère une requête SQL pour le top 10 des produits par chiffre
d'affaires, groupé par région. {"sql": "...", "explanation": "..."} """
```

5.4 Tool Use (Utilisation d'outils)

Le Tool Use est l'une des fonctionnalités les plus puissantes de l'API Claude. Il permet au modèle d'invoquer des fonctions externes que vous définissez, créant ainsi un pont entre le raisonnement du LLM et des actions concrètes dans le monde réel.

Architecture du Tool Use

Figure 5.3 — Flux complet du Tool Use en 6 étapes

Définition d'un outil (Tool Schema) :

```
tools = [{ "name": "get_weather", "description": "Récupère la météo actuelle d'une ville",
"input_schema": { "type": "object", "properties": { "city": { "type": "string",
"description": "Nom de la ville" } }, "required": ["city"] } }]
```

Outils spéciaux intégrés

Outil	Fonction	Cas d'usage
Text Edit Tool	Modification de texte structurée (insert, replace)	Édition de documents, refactoring de code
Web Search Tool	Recherche web intégrée directement dans l'API	Données actuelles, fact-checking
Computer Use	Contrôle d'interface graphique (clavier, souris)	Automatisation UI, tests d'interface

Tableau 5.4 — Outils spéciaux intégrés à l'API Claude

5.5 RAG et Recherche Agentique

Le Retrieval Augmented Generation (RAG) permet d'enrichir les réponses de Claude avec des données externes en temps réel, résolvant ainsi le problème de la connaissance limitée par la date de coupure.

Pipeline RAG complet

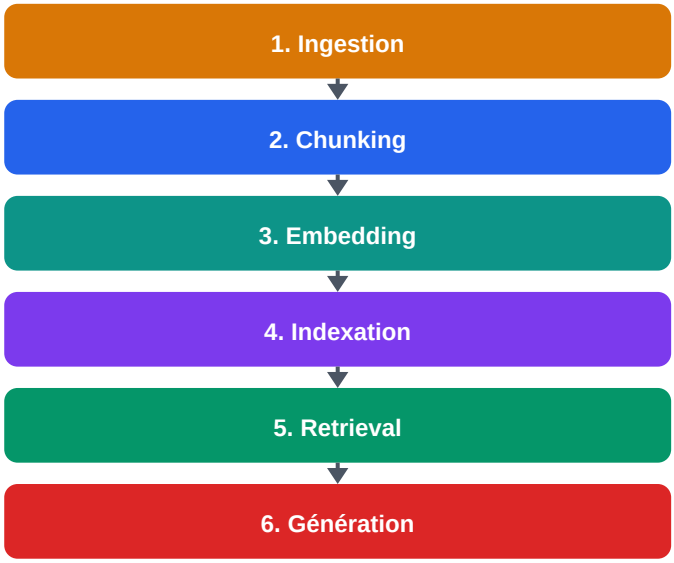

Figure 5.4 — Pipeline RAG en 6 étapes

Étape	Technique	Détail

Chunking	Découpage du texte	Par paragraphes, tokens fixes, ou sémantique
Embedding	Vectorisation	Conversion texte -> vecteur dense (ex: Voyage AI)
Indexation	Stockage vectoriel	Base vectorielle (Pinecone, Chroma, FAISS)
Retrieval sémantique	Similarité cosinus	Top-K documents les plus similaires à la query
BM25 lexical	Correspondance de mots	Recherche par termes exacts (complémentaire)
Multi-Index	Hybride	Combine sémantique + lexical pour meilleure précision

Tableau 5.5 — Détail des techniques RAG couvertes dans le cours

5.6 Fonctionnalités avancées (complétées)

Extended Thinking

L'Extended Thinking permet à Claude de "réfléchir" plus longuement avant de répondre, améliorant significativement la qualité des réponses pour les problèmes complexes (mathématiques, code, raisonnement multi-étapes). Le budget de tokens de réflexion est configurable.

```
response = client.messages.create( model="claude-sonnet-4-6", max_tokens=16000, thinking={
"type": "enabled", "budget_tokens": 10000 }, messages=[{"role": "user", "content": "..."}]
)
```

Support d'images

Claude accepte les images en entrée (base64 ou URL) et peut les analyser, décrire, extraire des données de graphiques, lire du texte dans des captures d'écran, et comparer visuellement des éléments.

5.7 Modules restants à compléter

Module	Leçons restantes	Sujets clés
Features of Claude	7 leçons	PDF, citations, prompt caching, code execution, Files API
Model Context Protocol	11 leçons	Serveurs MCP, clients, outils, ressources, prompts
Claude Code & Computer Use	4 leçons	Applications Anthropic, setup, MCP servers
Agents et Workflows	8 leçons	Parallélisation, chaînage, routage, agents autonomes
Évaluation finale	2 leçons	Quiz final et conclusion

Tableau 5.6 — Modules restants du cours Building with the Claude API

Aperçu des patterns d'agents et workflows

Les modules non complétés couvrent les patterns architecturaux avancés pour la construction de systèmes IA en production :

Pattern	Description	Cas d'usage
Parallélisation	Exécuter des tâches indépendantes en parallèle	Analyse de N documents simultanément

Chaînage (Chaining)	Sortie d'une étape = entrée de la suivante	ETL: extraction -> transformation -> chargement
Routage	Diriger vers le bon traitement selon le type	Classifier puis traiter différemment par catégorie
Agent autonome	Boucle outil + décision jusqu'à résolution	Agent de recherche qui explore et synthétise

Tableau 5.7 — Patterns d'agents et workflows (aperçu des modules restants)

Le cours Building with the Claude API est le plus technique et le plus complet. Les modules déjà complétés couvrent les fondations essentielles : API, prompting, tool use et RAG. Les modules restants (MCP, agents, workflows) représentent les patterns avancés pour construire des systèmes IA de production. En tant qu'expert en data engineering, ces modules sur le RAG multi-index et les pipelines agentiques sont particulièrement pertinents pour votre activité.