

AI-NATIVE SDLC • MANUAL → AGENTIC → ORCHESTRATED

Advanced Claude & AI-Native SDLC.

A 16-hour intensive, hands-on program that moves engineers from "Ask AI" to "Build with AI" — orchestrating Claude Code, Cowork, Subagents, Skills, Hooks, MCP and Plugins into one governed, repository-aware AI-native SDLC.

16h

LIVE HOURS • 2 DAYS

10

MODULES

70%

PRACTITIONER LABS

5

AGENT CAPSTONE PIPELINE

01 — PROGRAM PHILOSOPHY

This is not a Claude introduction. It's an engineering orchestration program.

Casual prompting is a temporary hack — most engineers plateau there because they never learn how **agent.md**, **SKILL.md** and **MCP hooks** actually connect. This program builds the discipline to move from isolated prompts and code generation toward repository-aware, reusable, governed engineering workflows across the full delivery lifecycle.

Manual SDLC → AI-Assisted Development → Agentic Engineering → AI-Native SDLC Orchestration

THE OLD HABIT

"Ask AI" — isolated prompts, one file at a time, no repository memory.

THE AIXL SHIFT

"Build with AI" — repository-aware, agent-orchestrated, gated at every checkpoint.

TESTERS

From manual scripts to autonomous validation agents.

DEVELOPERS

From file editing to repository-wide reasoning.

ARCHITECTS

From diagramming to AI-native lifecycle design.

DEVOPS & DBA

From manual config to agentic infrastructure & data ops.

02 — PROGRAM AT A GLANCE

DURATION

FORMAT

SCHEDULE

16 hours · 2 days

Live, hands-on intensive

Weekend cohort

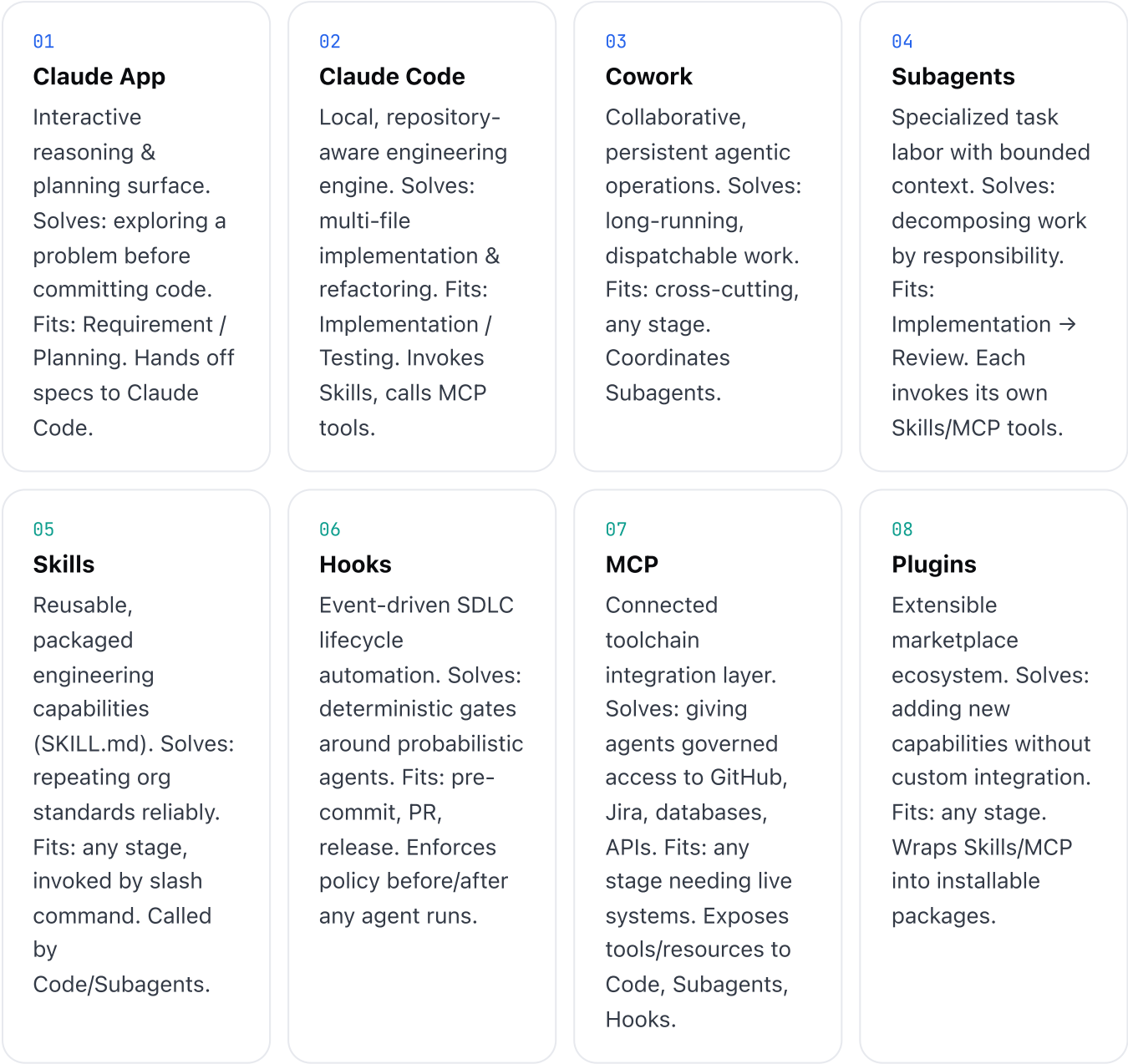
METHODOLOGY
Concept + Lab + Capstone

OUTPUT
AI-Native SDLC Pack

CAPSTONE
Multi-agent delivery pipeline

Eight components. One connected engineering system.

What each piece is, the engineering problem it solves, where it sits in the SDLC, and how it hands off to the rest of the stack.



The five most-confused concepts in the stack — what each is for, and when to reach for it.

CONCEPT	WHAT IT IS	TRIGGERED BY	USE IT WHEN
Skill	A reusable, versioned instruction set (SKILL.md).	A slash command or agent invocation.	You repeat the same engineering procedure often and want it standardized.

Subagent	A specialized agent with its own context and responsibility.	Delegation from Code, Cowork, or another agent.	Work needs a distinct role (test, review, security) kept out of the main context.
Hook	A deterministic script fired at a lifecycle event.	Pre/post-commit, PR open, release.	A rule must hold every time — no exceptions, no agent judgment.
MCP	A protocol connecting Claude to external tools/data.	An agent calling a registered tool or resource.	Claude needs live access to a real system (repo, ticket, DB).
Plugin	An installable package bundling Skills/MCP/UI into one capability.	Installed from the marketplace, then invoked like a Skill.	You want an off-the-shelf capability rather than building one from scratch.

Ten modules across two days. 16 live hours.

Every module states what is taught, how it's practiced, and what it produces.

DAY 1 — CLAUDE-NATIVE ENGINEERING

01 The 2026 AI-Native SDLC

1.5 h

Objective — Establish the shift from manual coding and isolated copilots to AI-native engineering orchestration.

TOPICS & SUBTOPICS

- Evolution of AI-assisted development: autocomplete → copilot → agentic
- The AI-native SDLC operating model
- Human vs. agent responsibilities
- Repository-aware engineering
- Universal / computer-use concepts & environment automation
- Engineering risks & governance boundaries

CONCEPTS & FRAMEWORKS

Manual SDLC → AI-Assisted → Agentic Engineering → AI-Native Orchestration maturity model; "Ask AI" vs. "Build with AI".

HANDS-ON

Map your team's current SDLC against the maturity model; identify where isolated prompting is masking systemic risk.

ARTIFACT

AI-Native SDLC Architecture (draft).

CAPABILITY DEVELOPED

Diagnosing where an engineering org sits on the AI-native maturity curve.

02 Claude Code & Autonomous Engineering Agents

2 h

Objective — Operate Claude Code as a repository-aware engineering engine, not a chat window.

TOPICS & SUBTOPICS

- Repository understanding & project/context configuration
- Codebase navigation & multi-file reasoning
- Planning before implementation
- Implementation, refactoring & debugging
- Testing workflows
- Permission / execution modes: supervised vs. autonomous
- Safe engineering workflows

CONCEPTS & FRAMEWORKS

Project context (CLAUDE.md) design; plan-then-execute loop; the permission-mode spectrum.

HANDS-ON

Configure Claude Code against a real multi-file repository and run a full plan → implement → test → refactor cycle.

ARTIFACT

Claude Code Repository Configuration + Repository-Aware Engineering Workflow.

CAPABILITY DEVELOPED

Directing an autonomous coding agent safely across a real codebase.

03 Claude Cowork & Collaborative Agentic Operations

1.5 h

Objective — Use Claude as a structured software-engineering coworker for persistent, delegated work — not only synchronous chat.

TOPICS & SUBTOPICS

- Cowork operating model
- Task delegation & persistent / long-running work
- Parallel work
- Dispatch & remote task assignment
- Human ↔ agent coordination
- Appropriate vs. inappropriate autonomy

CONCEPTS & FRAMEWORKS

Delegation boundaries; dispatch-from-any-device model; asynchronous work queues.

HANDS-ON

Delegate a multi-step engineering task to Cowork, dispatch it, and run a second task in parallel while reviewing the first.

ARTIFACT

Delegated Task Log & Autonomy Boundary Notes.

CAPABILITY DEVELOPED

Assigning and supervising long-running agentic work without babysitting it.

04 Subagents & Specialized Agent Teams

1.5 h

Objective — Decompose engineering work across specialized agents with clear responsibilities and context boundaries.

TOPICS & SUBTOPICS

- Architecture, implementation & testing agents
- Security / review & documentation agents
- DevOps / release agents
- Responsibility & context-boundary design
- Delegation, coordination & verification of agent output

CONCEPTS & FRAMEWORKS

Specialized-agent decomposition; context isolation; verification-before-trust.

HANDS-ON

Define three specialized subagents (implementation, test, review) for a sample feature and run them in sequence against one repository.

ARTIFACT

Specialized Subagent Definitions.

CAPABILITY DEVELOPED

Decomposing engineering work into verifiable, specialized agent roles.

05 The SKILL.md Ecosystem & Reusable Engineering Skills

1.5 h

Objective — Package repeatable engineering knowledge into reusable, team-level capabilities.

TOPICS & SUBTOPICS

- Purpose of Skills & SKILL.md structure
- Reusable engineering instructions & organizational standards
- Role-specific skills
- Slash-command workflows: /frontend, /test, /audit, /review
- Creating team-level reusable capabilities

CONCEPTS & FRAMEWORKS

Skill packaging pattern; organizational standards encoded as invocable capability.

HANDS-ON

Author a SKILL.md for a real recurring task (a code-review or test-generation standard) and invoke it via slash command.

ARTIFACT

Reusable SKILL.md.

CAPABILITY DEVELOPED

Turning tribal engineering knowledge into a reusable, invocable capability.

06 Model Context Protocol (MCP)

2 h

Objective — Connect Claude to enterprise engineering systems through a governed integration layer.

TOPICS & SUBTOPICS

- MCP architecture & mental model
- Servers, tools & resources
- Connecting GitHub, Jira, databases, APIs & internal systems
- Authentication & permissions
- Security boundaries
- Deciding what should — and shouldn't — be exposed to agents

CONCEPTS & FRAMEWORKS

MCP client-server model; least-privilege tool exposure.

HANDS-ON

Connect Claude to one real engineering tool or data source via MCP (GitHub or a database) and run a task through the connection.

ARTIFACT

MCP Integration.

CAPABILITY DEVELOPED

Wiring Claude into real engineering toolchains without over-exposing systems.

07 Claude Plugins & Extensible Engineering Ecosystem

1 h

Objective — Extend Claude's capabilities through Plugins, and know when to reach for one instead of a Skill or MCP server.

TOPICS & SUBTOPICS

- Plugin architecture & concept
- Official & third-party plugins
- Marketplace / ecosystem
- Evaluating plugins for trust & security
- Plugin vs. Skill vs. MCP

CONCEPTS & FRAMEWORKS

Extension vs. integration vs. packaging distinction.

HANDS-ON

Evaluate and install a plugin from the marketplace against a trust checklist; map it against Skill/MCP alternatives.

ARTIFACT

Plugin / Extension Configuration.

CAPABILITY DEVELOPED

Selecting and vetting the right extension mechanism for a given engineering need.

08 SDLC Lifecycle Hooks & Engineering Automation

1.5 h

Objective — Put deterministic controls around probabilistic agents at defined lifecycle events.

TOPICS & SUBTOPICS

- Lifecycle hooks: pre/post execution
- Pre-commit automation & PR review gates
- Testing hooks & security scanning
- Quality enforcement & policy checks
- Automated validation

CONCEPTS & FRAMEWORKS

Deterministic-gate-around-probabilistic-agent pattern.

HANDS-ON

Build an automated engineering gate — a pre-commit or PR hook running tests and a security scan — and wire it into the agent workflow.

ARTIFACT

SDLC Lifecycle Hook / Automated Gate.

CAPABILITY DEVELOPED

Enforcing non-negotiable quality & security controls around agent-driven changes.

09 Designing the AI-Native SDLC

1 h

Objective — Assemble Claude App, Code, Cowork, Subagents, Skills, Hooks, MCP and Plugins into one connected engineering system.

TOPICS & SUBTOPICS

- Requirement → Planning → Architecture → Implementation
- Testing → Review → Security → Documentation
- PR → Release, mapped to the Claude stack
- Human checkpoints & agent responsibilities
- Reusable skills, external integrations & automated gates
- Governance boundaries

CONCEPTS & FRAMEWORKS

End-to-end AI-native SDLC operating model.

HANDS-ON

Map your own team's SDLC stage-by-stage to the Claude stack, marking every human checkpoint and automated gate.

ARTIFACT

AI-Native SDLC Architecture (final).

CAPABILITY DEVELOPED

Designing a governed, connected AI-native SDLC operating model for a real engineering team.

10 Capstone — Multi-Agent Software Delivery Pipeline

2.5 h

Objective — Build a governed, end-to-end AI-native engineering pipeline using specialized agents against a real repository.

PIPELINE

Requirement/Architecture Agent →
Implementation Agent → Test Agent →
Review/Security Agent →
Release/Documentation Agent.

INCORPORATES

- Claude Code, Subagents, Skills, Hooks, MCP & Plugins
- Repository context & automated tests
- Review/security gates & human approval checkpoints

GOAL

Not five agents talking to each other — a governed AI-native SDLC where specialized agents perform meaningful engineering work across a real repository.

ARTIFACT

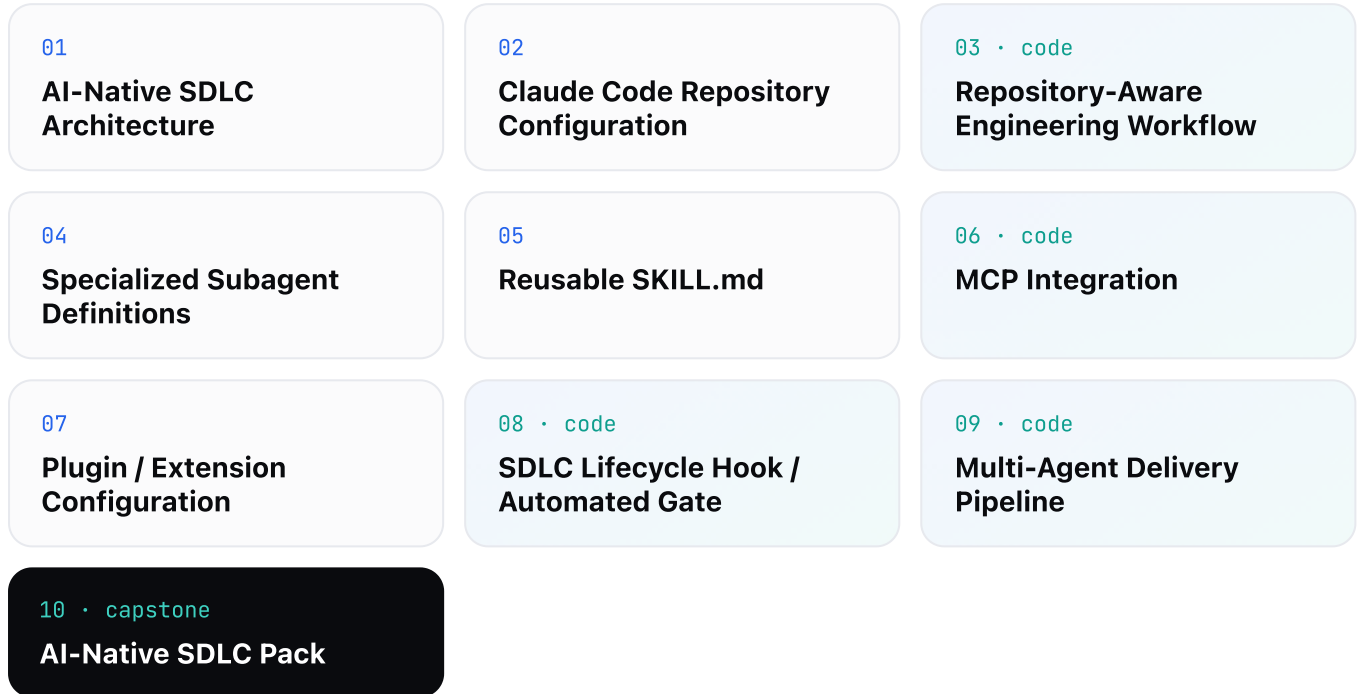
Multi-Agent Delivery Pipeline + AI-Native SDLC Pack.

CAPABILITY DEVELOPED

Demonstrating a governed AI-native SDLC in which specialized agents ship real engineering work end to end.

Ten cumulative outputs. One AI-Native SDLC Pack.

Each module's output builds on the last — by the end of Day 2, these ten artifacts together form the AI-Native SDLC Pack a real engineering team would take back and apply.



An advanced engineering program — not a beginner Claude or prompting workshop.

ROLES

- Software Developers
- Tech Leads
- Solution Architects
- QA / Test Engineers
- DevOps / Platform Engineers
- Database Engineers / DBAs

ALSO FOR

Engineering teams moving as a group toward AI-native delivery.

RECOMMENDED BACKGROUND

Working programming experience, Git/command-line familiarity, and standard SDLC

NOT THIS

Not a fit

Basic prompt-engineering or "intro to Claude" content.

This program

Advanced engineering orchestration, against real repositories.

exposure.

08 — PROGRAM OUTCOMES

By the end of this program, participants will be able to:

- ✓ Operate Claude Code effectively against real, multi-file repositories.
- ✓ Decompose engineering work across specialized subagents.
- ✓ Integrate engineering tools and data sources through MCP.
- ✓ Automate SDLC events with deterministic Hooks.
- ✓ Decide appropriate levels of agent autonomy per task.
- ✓ Design multi-agent software delivery pipelines.
- ✓ Design repository-aware AI engineering workflows.
- ✓ Build reusable Skills that encode organizational standards.
- ✓ Use and extend Claude through vetted Plugins.
- ✓ Introduce automated test, review & security gates.
- ✓ Maintain human oversight and engineering accountability.
- ✓ Create an AI-native SDLC operating model for an engineering team.

AIXL ACADEMY

From "Ask AI" to an AI-native SDLC you own.

16 live hours

2-day intensive

AIXL Academy Certificate