Subagents and Agent Teams in Claude Code

Course Number: AIC-3101

Duration: 1 days

Course Description

Scale Claude Code beyond single-session use with subagents and coordinated agent teams. Subagent creation and configuration cover what subagents are and when to use them, the built-in subagents that ship with Claude Code, authoring custom subagents as Markdown files with scoped frontmatter, and applying tool allowlists and denylists to bound each subagent's blast radius. Advanced patterns cover permission modes, persistent memory across sessions, preloaded skills, PreToolUse and PostToolUse hooks for conditional validation, foreground versus background execution, and the chaining and parallel patterns that isolate noisy operations. Agent teams cover when teams beat a single subagent, lead-and-teammates topologies, messaging and broadcast patterns, and quality gates that prevent silent failures. Hands-on labs produce a read-only code-reviewer subagent, a database-query subagent that validates through PreToolUse hooks, and a parallel code review run by competing-hypothesis teammates. The course is designed for software developers, AI engineers, and DevOps engineers who already use Claude Code for daily work.

Skills Gained

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

- Configure custom subagents with scoped tools, models, and permissions
- Wire persistent memory, hooks, and preloaded skills into advanced subagent workflows
- Choose between subagents and agent teams for different kinds of work

- Coordinate agent teams with shared task lists and TeammateIdle/TaskCompleted quality gates

Who Can Benefit

This course is designed for:

- Software Developers
- AI Engineers & Architects
- DevOps Engineers

Prerequisites

Participants should enter this course with:

- Practical Python experience
- Experience with AI Coding Agents (e.g., AIC-1101 or equivalent)

Organizational Objectives

This course assists organizations to:

- Reduce review cycles by delegating routine code review to scoped read-only subagents
- Lower agent error rates through PreToolUse validation and tool-allowlist scoping
- Accelerate parallel investigation work through agent-team orchestration
- Build a working knowledge of Claude Code's advanced agent surface in one day

Software

All attendees must have a modern web browser and an Internet connection.

Course Details

Module 1 - Creating and Configuring Subagents

By the end of this module, you will be able to define a custom Claude Code subagent as a Markdown file, scope it to project or user level, configure its tools, model, and turn budget through frontmatter, and apply tool allowlists and denylists to keep its blast radius bounded.

- What subagents are and why you'd use them
- Built-in subagents (Explore, Plan, general-purpose)
- Creating subagents with /agents and as Markdown files
- Scoping (project, user, CLI, plugin)
- Frontmatter fields (tools, disallowedTools, model, maxTurns) and tool allowlists/denylists

- Picking the right model for the job
- Hands-on Lab: Author a read-only code-reviewer subagent with a scoped tool allowlist and run it against an open PR.

Module 2 - Advanced Subagent Patterns

By the end of this module, you will be able to layer permission modes, persistent memory, and preloaded skills onto a subagent; enforce conditional validation through PreToolUse and PostToolUse hooks; and choose between foreground, background, chained, and parallel execution patterns.

- Permission modes (default, acceptEdits, dontAsk, bypassPermissions, plan)
- Persistent memory across sessions and preloaded skills
- Lifecycle hooks (PreToolUse, PostToolUse) and conditional validation
- Foreground vs. background execution
- Chaining, parallel research, and isolating noisy operations
- When to use a subagent vs. the main conversation
- Hands-on Lab: Wire persistent memory, preloaded skills, and a PreToolUse validation hook into a database-query subagent.

Module 3 - Orchestrating Agent Teams

By the end of this module, you will be able to recognize when an agent team beats a single subagent, set up a lead-and-teammates team with a shared task list, coordinate work through direct messaging and broadcasts, and apply quality gates with TeammateIdle and TaskCompleted hooks.

- Subagents vs. agent teams — when teams make sense (and when they don’t)
- Setting up a team (lead, teammates, shared task list) and display modes
- Task dependencies and self-claiming
- Direct messaging, broadcasting, and plan approval
- Quality gates with TeammateIdle and TaskCompleted hooks
- Practical limits (team size, task granularity, file conflicts)
- Hands-on Lab: Run a parallel code review with two competing-hypothesis teammates on the same PR and reconcile their findings through quality gates.