Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

AI-Assisted Development with C

Class Duration

35 hours of live training delivered over 5 days.

Student Prerequisites

- Professional C# and .NET development experience
- Working knowledge of Git and GitHub (issues, pull requests, branches, reviews)
- Comfort with the command line and the .NET CLI
- No prior AI tool experience required

Target Audience

Professional C# developers who want a structured, end-to-end path from AI-curious to AI-proficient. Ideal for engineers and teams adopting AI coding tools across the full development lifecycle — writing, refactoring, testing, reviewing, and shipping .NET code — who also need to understand the security, licensing, and governance implications of doing so responsibly. This is a tool-agnostic course: students work hands-on with Claude Code, GitHub Copilot, Codex, Cursor, Windsurf, and Gemini CLI rather than betting on a single vendor, and the skills transfer to whichever assistant your organization adopts. Labs run in both Visual Studio and VS Code.

Description

The .NET ecosystem gives AI coding tools an unusually strong foundation to work against: a rich type system, Roslyn analyzers, and a mature test ecosystem provide fast, reliable feedback that agentic workflows thrive on. This five-day intensive teaches professional C# developers how to take full advantage. Days one and two cover the foundations: how large language models generate code, the strengths and tradeoffs of the frontier models, and prompting and context engineering techniques that produce idiomatic, modern C# — nullable-aware, async-correct, and aligned with current .NET 10 APIs rather than patterns from a decade of training data. Days two and three move into the tools: inline assistance and chat-driven editing in Visual Studio and VS Code, then full agentic workflows across Claude Code, GitHub Copilot, Codex, Cursor, Windsurf, and Gemini CLI — plan-then-execute

Copyright 2025 Cloud Contraptions LLC – All Rights Reserved

sessions where the agent builds, reads compiler and analyzer output, and runs tests; grounding agents with project context files; and issue-to-PR automation. Day four applies AI to quality: generating and maintaining xUnit suites, AI-assisted debugging, code review, and modernizing legacy .NET Framework code. Day five addresses team-scale adoption: NuGet supply-chain risks, secret handling, licensing and IP questions, policy design, and honest productivity measurement. Students finish with a capstone that exercises the full workflow on a realistic ASP.NET Core codebase.

Learning Outcomes

- Explain how LLMs generate code and choose the right frontier model for a given C# task.
- Apply prompting and context engineering techniques that produce idiomatic, nullable-aware, modern C#.
- Use IDE assistants (Copilot in Visual Studio and VS Code, Cursor, Windsurf) effectively for completion, chat, and multi-file edits.
- Drive agentic coding sessions with Claude Code, Copilot, Codex, Cursor, Windsurf, and Gemini CLI, with `dotnet build`, analyzers, and tests in the loop.
- Ground agents in project context with CLAUDE.md / AGENTS.md, rules files, and MCP servers.
- Generate, evaluate, and maintain xUnit test suites with AI assistance.
- Use AI for debugging, code review, and modernizing legacy .NET Framework code to .NET 10.
- Recognize C#-specific AI failure modes: stale APIs, async anti-patterns, and nullable violations.
- Apply security guardrails: NuGet supply-chain vetting, secret handling, and prompt-injection awareness.
- Contribute to team standards for licensing, governance, and measuring AI’s real productivity impact.

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Software Requirements

.NET 10 SDK, Visual Studio 2026 or VS Code with the C# Dev Kit, a GitHub account with Copilot access, Claude Code CLI with an Anthropic API key or

Copyright 2025 Cloud Contraptions LLC – All Rights Reserved

subscription, Codex, Cursor, Windsurf, and the Gemini CLI (trials and free tiers acceptable), Git, and the GitHub CLI (`gh`).

Training Topics

Foundations: How AI Writes Code

- How LLMs generate code: tokens, context windows, and training data
- The frontier models for coding: Claude, GPT, and Gemini compared
- Capability tradeoffs: speed, cost, context size, and reasoning
- Where AI excels in .NET work and where it reliably fails

Prompting and Context Engineering for C

- Writing prompts that produce idiomatic, modern C# (.NET 10, C# 14)
- Steering toward nullable reference types, records, and pattern matching
- Async correctness: avoiding generated `async void` and sync-over-async
- Context strategies: solutions, projects, and what to include from large codebases
- Avoiding stale .NET Framework idioms in generated code

IDE Assistants in Daily C# Work

- GitHub Copilot in Visual Studio: completions, chat, and edits
- Copilot in VS Code with the C# Dev Kit; Cursor with rules files
- Windsurf: Cascade, planning, and memories
- The wider landscape: Aider, Cline, JetBrains Junie, and Amazon Kiro
- Multi-file edits and chat-driven refactoring across projects
- XML doc comments and documentation generation
- Customizing assistants with instructions files

Agentic Coding: Claude Code, Copilot, Codex, Cursor, Windsurf, and Gemini CLI

- The agentic loop: plan, act, observe, iterate
- The agent landscape compared: Claude Code, Copilot agent mode, Codex, Cursor CLI, Windsurf Cascade, and Gemini CLI
- Plan mode and the supervision spectrum: permission prompts to auto mode
- CLAUDE.md and AGENTS.md: solution layout, SDK, and convention context

Copyright 2025 Cloud Contraptions LLC – All Rights Reserved

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

- The verify loop: agents reading `dotnet build`, analyzer, and test output
- Issue-to-PR workflows: assigning work to cloud agents
- MCP servers: connecting agents to internal tools and data

AI-Driven Testing and Quality

- Generating xUnit suites: fixtures, theories, and coverage gaps
- Integration testing ASP.NET Core with WebApplicationFactory and AI assistance
- AI-assisted debugging: exceptions, async stack traces, and logging
- Agentic code review and pull request automation
- Modernizing legacy .NET Framework code: characterization tests first, then migrate

C#-Specific Pitfalls and Verification

- Hallucinated NuGet packages and API versions
- Stale framework knowledge: verifying against current .NET docs
- Nullable violations and analyzer warnings as quality gates
- EditorConfig, Roslyn analyzers, and `dotnet format` for generated code
- Performance review: allocations, LINQ overuse, and async overhead

Security, Licensing, and Governance

- Prompt injection and untrusted content in agent workflows
- Secret leakage: user secrets, appsettings, and context hygiene
- NuGet supply-chain vetting and lockfile discipline
- Licensing and IP status of AI-generated code
- Team policy design: review requirements and audit trails
- Measuring real productivity impact honestly

Capstone Workshop

- Plan and execute a multi-step feature on a realistic ASP.NET Core codebase with the coding agent of your choice
- Generate and harden an xUnit suite for an untested module
- Issue-to-PR exercise: agent-ready issue through reviewed merge
- Security review of an AI-generated changeset including dependency audit
- Q&A session

Copyright 2025 Cloud Contraptions LLC – All Rights Reserved