Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

# AI-Driven Development Life Cycle (AI-DLC): Methodology and Practice

## Class Duration

14 hours of live training delivered over 2-3 days to accommodate your scheduling needs.

## Student Prerequisites

- Experience on a software delivery team as a developer, tech lead, architect, product owner, QA engineer, or engineering manager
- Familiarity with Scrum or Kanban
- Basic knowledge of Git and pull requests; developers drive the coding agents, and other roles need little coding
- Some experience with an AI coding assistant is helpful

## Target Audience

Delivery teams and the people around them: developers, tech leads, architects, product owners, QA engineers, and engineering managers. The course is for teams that already use AI coding agents and need a shared method with clear roles and human approval gates. Teams can pair it with *Spec-Driven Agentic Coding with Kiro* or *Spec-Driven Development with AI Coding Agents*.

## Description

AI-DLC is a delivery method from AWS that puts AI at the center of the work. AI drafts plans, questions, requirements, designs, code, and tests, and the team approves each step. Short bolts replace sprints, units of work replace epics, and the team reviews AI proposals together in mob sessions. This course teaches the method and runs it with AI-DLC Workflows, the open-source implementation in `awslabs/aidlc-workflows`. Students work through each phase in Kiro, Claude Code, GitHub Copilot, Cursor, or Codex CLI, and plan a rollout for their own organization.

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved 

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

## Learning Outcomes

- Explain how AI-DLC differs from AI-assisted and AI-autonomous development.
- Describe the AI-DLC phases and the artifacts each produces.
- Organize work as intents, units of work, and bolts.
- Facilitate Mob Elaboration and Mob Construction sessions.
- Run approval gates and request changes precisely.
- Choose a workflow profile, depth, and test strategy for a change.
- Install and run AI-DLC Workflows with the `aidlc` CLI.
- Run AI-DLC in Kiro, Claude Code, Copilot, Cursor, or Codex CLI.
- Adopt AI-DLC on existing codebases with reverse engineering.
- Turn gate corrections into durable rules with the learning loop.
- Map AI-DLC onto Scrum or SAFe practices.
- Define adoption metrics and plan an organization-wide rollout.

## Training Materials

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

## Software Requirements

Git, the `aidlc` CLI from AI-DLC Workflows, at least one supported coding agent with an active plan or model access (Kiro, Claude Code, GitHub Copilot, Cursor, or Codex CLI), and a GitHub account with a repository for the hands-on portions.

## Training Topics

### Why the Lifecycle Changes When AI Drafts the Work

- AI bolted onto existing rituals
- AI-assisted, AI-autonomous, and AI-driven development
- AI plans, asks, then implements
- Bolts instead of sprints
- Units of work instead of epics
- Persistent context in the repository

### AI-DLC Principles and Core Concepts

- AI proposes, humans approve

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved 

- Approval gates at every stage
- Intents, spaces, units, and bolts
- Determinism in tools, judgment in humans
- From rules-based workflows to the current engine
- Native harness integrations
- The Classic profile

### Ideation and Inception: Mob Elaboration

- Intent capture and framing
- Feasibility, constraints, and scope
- Team formation and rough mockups
- Requirements analysis and user stories
- Domain and contract design
- Units generation and delivery planning
- Starting from a PRD or vision
- The unit dependency graph
- The bolt plan
- Running Mob Elaboration

### Construction and Operation: Mob Construction

- Functional design per unit
- NFR requirements and design
- Infrastructure design
- Code generation after plan approval
- Unit checkpoints
- The walking skeleton
- Build, test, and CI pipeline stages
- Running Mob Construction
- The Operation phase
- Deployment stages

### Artifacts, State, and the Audit Trail

- The intent record under `aidlc/spaces/`
- What each stage writes
- Requirements, stories, and bolt plans
- `aidlc-state.md` and resumable sessions
- Parking and resuming work

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved 

- Reviewer agents and findings
- The `audit/` trail
- Sensors for traceability and coverage
- Lint and type-check sensors

### Adaptive Depth and Workflow Profiles

- Workflow profiles
- `feature`, `mvp`, and `poc` profiles
- `bugfix`, `refactor`, and `security-patch` profiles
- Minimal, Standard, and Comprehensive depth
- Independent test strategy levels
- The adaptive composer
- Changing scope at a gate
- Matching ceremony to risk

### Roles and Responsibilities

- Decisions each role owns
- Domain agents and their human counterparts
- Answer modes
- Approve or Request Changes
- When to take over
- Integration leads and unit teams

### Installing and Running AI-DLC Workflows

- Installing the `aidlc` CLI
- `aidlc config --harness`
- `aidlc doctor` and `aidlc update`
- Starting work with `/aidlc`
- Checking progress with `/aidlc --status`
- What the team commits
- `aidlc.settings.json`
- Optional MCP servers
- Amazon Bedrock as a model provider
- Weak models and skipped reviews

### AI-DLC Across Coding Agents

- Kiro IDE and Kiro CLI
- Claude Code skills, agents, and hooks

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved 

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- GitHub Copilot CLI and VS Code
- Cursor IDE and CLI
- Codex CLI and `$aidlc`
- opencode support
- How strictly each harness enforces gates

**Greenfield and Brownfield Adoption**

- Greenfield MVP and feature runs
- The technical environment document
- Reverse engineering existing code
- Practices discovery
- Monorepos and multi-repo intents
- Spaces for several teams
- Starting small with Express runs

**Team Knowledge, Rules, and the Learning Loop**

- Team knowledge folders
- DocumentKB for existing documents
- Org, team, and project rules
- Phase rules
- The learning loop
- Plugins and custom stages
- Custom sensors

**AI-DLC in the Organization**

- AI-DLC with Scrum and SAFe
- Bolts, backlogs, and ceremonies
- AI-DLC and spec-driven development
- Gate wait time and rework
- Request Changes rate and lead time
- Audit trail and commit provenance
- Guard policy and unattended runs
- A phased rollout roadmap
- Pilot teams and champions

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved 