Cloud Contraptions
TRAINING & CONSULTING

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

Advanced Windsurf

Class Duration

7 hours of live training delivered over 1-2 days to accommodate your scheduling needs

Student Prerequisites

- Basic understanding of programming concepts
- Experience with a programming language such as Java, Python, JavaScript/TypeScript, or C#
- Completed Windsurf Essentials course or have similar experience

Target Audience

Designed for senior software engineers, staff/lead developers, tech leads, and engineering managers who want to operationalize AI-assisted development for real codebases. It's ideal for teams modernizing workflows, reducing cycle time, and improving review quality in regulated or high-reliability environments. L&D partners and org leaders seeking scalable, repeatable practices will find clear rules, checklists, and playbooks that translate directly into PR-ready changes. The course runs live (online or in-person) and assumes fluency with modern version control, testing, and code review practices.

Description

This live, hands-on course equips professional developers to ship higher-quality software faster using high-leverage AI workflows. You'll master a practical mental model for when to ask, when to code, and when to cascade tasks; retrieve the exact code context quickly and keep AI grounded; and coordinate multi-step changes through reviewable diffs and checkpoints. We'll keep refactors small and safe, converge debugging with evidence-driven loops, and speed up verification with targeted, coverage-aware tests. You'll also learn safe terminal automation and produce consistent, PR-ready artifacts that scale across teams. The result is measurable gains in developer productivity, software delivery velocity, and risk reduction—without sacrificing code correctness or maintainability.

Cloud Contraptions
TRAINING & CONSULTING

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

Learning Outcomes

1. Apply the Tab vs Command vs Cascade mental model to choose the fastest tool and sequence work with an ask-first approach.
2. Plan and execute disciplined patch loops (plan → patch → diff-review → run → iterate) while selecting the right model for each task.
3. Retrieve and ground the right context fast by pinning evidence, scoping precisely, and using effective prompts to find entry points and data flows.
4. Orchestrate multi-step changes with clear control patterns, checkpointing, and clean task splits between refactors and behavior changes.
5. Perform small, safe refactors using repeatable playbooks, API-stability guardrails, bounded cross-cutting edits, and sharp diff-review heuristics.
6. Drive debugging to convergence from symptom to minimal repro using log-first tactics, stack-trace navigation, and rapid hypothesis testing.
7. Accelerate verification with targeted unit tests, sturdier test design, intentional snapshot updates, and coverage-guided additions.
8. Automate terminal workflows safely with reviewable commands, environment guardrails, and repeatable git/test/format/codegen patterns, producing PR-ready outputs at team scale.

Training Materials

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

Software Requirements

Student may choose to simply watch the instructor or follow along with the instructor. If students choose to code along and/or complete lab exercises, they will need the Windsurf editor. Much of the class can be completed with the free version of Windsurf, but a Pro subscription is recommened.

Training Topics

High-Leverage Windsurf Mental Model

- Tab vs Command vs Cascade (Ask vs Code): map common dev tasks to the fastest tool

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

- "Ask-first" sequencing: clarify intent, risks, and constraints before generating patches
- Patch discipline at speed: plan → patch → diff-review → run → iterate (repeat in tight loops)
- Choosing the right model for the moment: lightweight edits vs deeper multi-step reasoning (switch intentionally per task)

Context Retrieval That Finds the Right Code Fast

- What gets used as context: open files, indexed repo, selected text, and recent editor/terminal activity
- "Pin the evidence": select/paste key logs, stack traces, and code paths so Ask mode stays grounded
- Retrieval prompts that work: "find entry points", "trace data flow", "list callers", "identify invariants"
- Avoiding context poisoning: keep scope small, confirm assumptions, and re-anchor with exact snippets
- Fast context patterns: rapid "where is this used?" loops before any edits (optimize for correctness)

Cascade Orchestration for Multi-Step Changes

- Ask mode for architecture and approach: tradeoffs, edge cases, rollout plan, and a short checklist
- Code mode for execution: multi-file patches as reviewable diffs (treat as proposed commits)
- Control patterns: "one file at a time", "stop after each step", "wait for confirmation before proceeding"
- Checkpointing: "summarize what changed + what remains" after each patch
- When to split the task: isolate mechanical refactors from behavior changes for cleaner reviews

Refactors That Stay Small and Safe While Moving Fast

- Refactor playbooks: rename → extract → move → verify (repeatable, commit-sized steps)
- API stability guardrails: "don't change public surface", "keep behavior identical", "no new dependencies"
- Cross-cutting edits with bounded blast radius: target packages/modules, avoid "whole repo" changes

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

- Migration tactics: incremental rollout, feature flags, and compatibility shims
- Diff review heuristics: spotting accidental logic changes, shadowed variables, and unintended formatting churn

Debugging and Triage Workflows That Converge

- From symptom to repro: tighten reproduction steps and isolate the minimal failing case
- Log-first debugging: ask for instrumentation suggestions, then implement the smallest logging patch
- Stack-trace navigation: identify the owning layer, the suspicious boundary, and the first “unexpected” value
- Hypothesis loop: propose 2–3 causes → test quickly → keep only what’s supported by evidence
- “Stop and go manual” triggers: flaky repro, hidden side effects, or any patch that grows too large

Test and Verification Acceleration

- Targeted tests: generate a narrow unit test from the bug, then strengthen assertions and add edge cases
- Test refactors: reduce brittleness, remove incidental coupling, and improve failure messages
- Snapshot discipline: update intentionally, explain diffs, and add non-snapshot assertions where possible
- Coverage-guided work: identify untested seams and add the smallest tests that lock in behavior
- Verification loop: run the fastest checks first, then expand (lint → unit → integration)

Terminal Automation Without Regrets

- Safe terminal automation knobs: allow list / deny list and when to enable Turbo mode
- “Reviewable commands” habit: generate, edit, then run—especially for destructive operations
- Guardrails for environments: staging vs prod awareness, read-only defaults, and careful credential handling
- Debug sessions: share selected terminal output to keep Ask mode grounded

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

- Common safe patterns: git/status/diff loops, test runs, formatting, and codegen commands

Team-Ready Consistency at Scale

- Rules vs Memories: what should be enforced vs what should be remembered across conversations
- Thin, high-signal rules: style, architecture boundaries, error handling conventions
- Shareable workflow library: reusable “bugfix”, “refactor”, and “test-writing” playbooks for the team
- PR-ready output: change summaries, reviewer checklists, and “risk notes” that prevent rework