Cloud Contraptions
TRAINING & CONSULTING

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

Windsurf Essentials

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#

Target Audience

Designed for professional software engineers, senior/staff developers, tech leads, and engineering managers who want reliable, auditable AI-assisted development. Ideal for teams operating in regulated or large-scale codebases (including monorepos) that demand strict reviewability and change control. Participants should be comfortable with Git, command-line tooling, unit tests, and standard code review practices. Platform, DevEx, and QA leaders will also benefit from the emphasis on guardrails, indexing boundaries, and a fast validation loop that ties AI-generated patches to measurable gains in code quality, delivery speed, and business outcomes.

Description

This live, practitioner-led course teaches professional developers how to turn Windsurf into a dependable AI pair-programmer that accelerates delivery without sacrificing control. You'll internalize a diff-first mental model, master the core edit loop, and learn when to use Tab, Command, or Cascade to fit the job. We'll configure models and workspace conventions that actually matter, then layer on intent-driven editing, selection-scoped refactors, and multi-file patches that remain small, reviewable, and reversible. Throughout, you'll apply enterprise-grade safeguards—secrets hygiene, indexing boundaries, explicit "only touch these files" constraints—and a fast validation loop that runs tests, checks edge cases, and confirms telemetry expectations. You'll finish with repeatable micro-workflows for bug fixes from repro, constraint-safe refactors, documentation cleanup, and targeted unit tests—so your team ships higher-quality code faster, with lower risk and clearer ROI.

Cloud Contraptions
TRAINING & CONSULTING

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

Learning Outcomes

1. Choose the right Windsurf mode (Tab, Command, Cascade) based on task scope, risk, and required turnaround.
2. Configure a reliable setup: model selection, indexing health, permissions, repo size expectations, and workspace rules.
3. Execute the core edit loop end-to-end: select context → request change → review diff → validate → iterate.
4. Drive inline momentum with Tab and Supercomplete to add, remove, and reshape code without losing flow.
5. Perform precise, reviewable changes using Command + Inline Edits with tight scoping and diff control.
6. Orchestrate multi-step work in Cascade (Ask/Code) with checklists, pausing, and one-file-at-a-time control.
7. Apply quality and safety guardrails: secrets hygiene, indexing boundaries, explicit constraints, and quick validation.
8. Ship production-ready changes via practical micro-workflows: bug fixes, constrained refactors, docs cleanup, and focused tests.

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

Windsurf Workflow Mental Model

- Tab vs Command vs Cascade (Ask vs Code): choosing the right mode for the job
- “Diff-first” habits: treat every suggestion as a proposed patch, not truth
- The core edit loop: select context → request change → review diff → validate → iterate

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

- Human-in-the-loop automation: when to automate, when to stay manual

Fast Setup That Actually Matters

- Install/sign-in and first-run sanity checks (indexing health, permissions, repo size expectations)
- Model selection basics: when to use fast, file-scoped edits vs heavier multi-step help (switch intentionally per task)
- Configure Tab behaviors (Autocomplete vs Supercomplete) for your day-to-day workflow
- Workspace conventions: thin Rules + when to rely on (or avoid) Memories for persistence

Core Edit Loop Mastery

- Context-awareness basics: what Windsurf “sees” by default (open files + indexed repo + retrieved snippets)
- Pin/select context on purpose: “only use these files” + keep context small and relevant
- Prompt pattern for dev work: goal → constraints → definition of done (DoD)
- Guardrails that prevent surprises: “only touch these files”, “no behavior changes”, “keep public API stable”
- Right-sizing requests: one change at a time vs guided multi-step (plan → patch → verify)

Tab + Supercomplete for Inline Momentum

- Tab flow: accept/cancel, steering suggestions while staying in motion
- Supercomplete for intent-driven edits (add/remove/reshape blocks without rewriting everything)
- Tab power moves: tab-to-import + tab-to-jump to keep typing velocity high
- Micro-patterns: small, frequent accepts vs big completions (optimize for reviewability)

Command + Inline Edits for Scoped Changes

- Cmd/Ctrl+I for targeted refactors, bug fixes, and small feature slices
- Scoping techniques: selection-first and “small surface area” prompts to reduce unintended edits

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

- Review mechanics: accept/reject/follow-up on diffs (iterate without losing intent)
- Command in the terminal to generate safe, reviewable CLI commands from plain English

Cascade Essentials for Multi-Step Work

- Cascade Ask mode: discovery, design questions, tradeoffs, and “what should we do next?”
- Cascade Code mode: multi-file changes as reviewable patches (keep it diff-first)
- Planning for longer tasks: ask for a short plan + a concrete checklist before edits begin
- Keep it controllable: “pause after each step”, “wait for my confirmation”, “one file at a time”
- Terminal awareness: share selections/logs when debugging to reduce back-and-forth

Quality, Safety, and Verification

- Secrets hygiene and safe sharing: what never goes into prompts (keys, tokens, customer data)
- Indexing boundaries: what isn’t indexed/edited by default (and how ignore rules affect results)
- Quick validation loop: run tests, read diffs, check edge cases, confirm logging/telemetry expectations
- Risk flags: ambiguous requirements, cross-cutting changes, migrations, security-sensitive code
- When to stop and take manual control: uncertainty, hidden dependencies, “too much changed” signals

Practical Micro-Workflows That Ship

- Bug fix from repro: logs → hypothesis → minimal patch → verify → regression test
- Refactor with constraints: style/perf/readability improvements without regressions
- Documentation and cleanup: naming, comments, dead code, formatting consistency
- Test acceleration: generate focused unit tests, then tighten assertions and add edge cases