// CORPORATE AI TRAINING

Training that is delivered and graded like engineering work.

We train software teams to work with AI agents the way engineers work with any other production system: with written requirements, verification, and measurable results — in the same environment where they do their real job.

Vendor-neutral by design — we train your workflow, not just a tool, so you can switch tools whenever you like. Our preferred tool is Claude Code, but the program can run on Codex, Cursor, Copilot, OpenCode, or any other popular tool.

See how we prove it →

- programs assembled around your real needs
- every practical assignment based on real-world scenarios
- a catalog of dozens of proven topics
- groups from a few people to hundreds
- support chat, session recordings, detailed self-study materials

Workshops run on your stack and your artifacts; practical assignments ship as pull (merge) requests; results live in an auditable journal with certificates verifiable online.

// THE VERIFIED TRAINING METHOD

The Verified Training Method

Participants train on tasks as close to their real work as possible, in the environment they already use. What gets credited as a training result is real, completed work that meets the quality bar. Every program below runs on the same method.

01 **Intake survey before session one**

The curriculum is tuned to your team's tools and needs, varied experience levels and real problem areas; you receive a participant portrait before we start.

02 **Workshops on your stack**

2-hour sessions, online or on-site, split roughly 50/50 between theory and practice.

03 **Homework as merge requests**

Reviewable, diffable, timestamped. Different difficulty levels make it possible to train people with no coding experience at all, or to upskill senior-level engineers.

04 **A grading mechanism**

The rubric is attached to every assignment when it is handed out; an AI reviewer grades against it; the trainer also reviews the AI's assessments, edge cases and regrade appeals, and personally reviews every final project. Late work is penalized under a published rule, which keeps participants submitting on time.

05 **A live progress journal**

Every submission and grade is tracked throughout the course. Leadership sees cohort aggregates by default; full per-person detail is enabled at your discretion.

06 **Certificates**

Computed from practical performance, not just handed out; each one can be verified online by number and QR code. A management tool: per-person capability evidence, not just attendance.

07 **Per-session review reports**

After every workshop you get a written review: what was submitted, what passed, and what to fix before the next session.

08 **Detailed materials**

Slides, session recordings, step-by-step guides, and more.

> **Start small**
>
> The lowest-risk entry is a pilot: intake survey, one 2-hour workshop, and the group's results. You see the method and your team's baseline before deciding to train the whole staff.

// THE PROGRAMS

Five programs, one method

The programs are flexible: the topics below form a catalog, and every cohort is assembled from it at intake to match your needs and your specialists' maturity level.

Session counts are flexible

The plans below are examples of formats we have run; the same program can be a short intensive or an extended multi-month course — the plan is settled before the start, and the duration scales with it.

We can run very large groups

AI agents help support the group and grade the practical work, and that scales well: the same program runs for a team of ten just as well as for dozens or hundreds of participants at once, with every submission still individually assessed and every participant receiving personal feedback.

PROGRAM	AUDIENCE	EXAMPLE PLAN	TEAM TIME (EXAMPLE PLAN)
1. Agentic Engineering for Developers	developers, tech leads, architects	12 × 2 h assembled from a 29-topic catalog	~40-50 h per person over ~3 months
2. Agentic QA	QA engineers	9 × 2 h + Session 0	~30 h per person
3. Agentic Business Analysis	BAs, product analysts	9 × 2 h + Session 0	~30 h per person

4. Agentic QA + BA (Combined Program)	QA and BA as one cohort	12 × 2 h + joint capstone; each attends 10	`~32 h per person`
5. AI Engineering for Leaders	CTO, VPs, heads, architects	2–6 sessions × 2 h from a 7-topic list	`~4-12 h per person`

> **Pricing**
>
> Per cohort: a base rate per session plus a per-participant fee for verification (grading, journal, credentials). Exact rates are quoted after the intake survey.

`// PROGRAM 1 — THE DEEPEST TRACK`

Agentic Engineering for Developers

AI agents in everyday development work.

> AFTER THIS PROGRAM
>
> Your team ships agent-assisted changes through a verified workflow: written intent, curated context, review tiered by risk.

Training cadence

The recommended pace is one workshop per week, which keeps the balance between intensity and progress. The cadence can be changed, project competitions added, and so on.

Certificates: multi-level at the client's request (with distinction, standard, theory-only). All computed from the journal and verifiable by QR code.

Typical group configurations

`Core-12 (default)`

F1–F8, one of F9/F10, F11, plus a few advanced sessions chosen at intake (most often A1 and A6).

`Advanced-12`

A1–A12, for teams that pass a placement diagnostic (fundamental-level homework at L2 difficulty) or have completed Core-12.

`Brownfield intensive (6)`

F11 + A1 + A2 territory: legacy archaeology, safe changes, judging generated code, PR review and team rules, MCP workflows.

`Training that grows into consulting`

Training plus guided consulting on your real system — your developers drive, the trainer guides and helps with the tricky parts.

`Migration-driven training`

The curriculum taught through a real migration project as part of the team's AI transformation (e.g. no-code workflows rebuilt as code by agents).

Level 1 — Fundamental (12 topics)

Goal: every developer delegates routine work to an agent safely — with written intent, curated context, and verification of every result. Each session has an observable objective, and the homework criteria are derived from it. Materials are updated before every session to reflect new tools and approaches.

WS	TOPIC	WHAT'S COVERED
F1	**State of AI development**	Where agentic development actually stands: the evidence, why some AI projects take off and others fail, what changes in the developer's role.
F2	**Prompt engineering & security**	Reliable prompting; what may and may not enter a tool; secrets hygiene; prompt-injection defenses; working with synthetic stand-ins.
F3	**Context engineering**	Managing what the agent sees: repository context, selective attachment, controlling context degradation, reliable first-pass results.
F4	**Evaluating and selecting agent tooling**	The landscape of modern tools and how they compare across different criteria.
F5	**Agentic IDE rules & commands**	AGENTS.md as the single source of truth, rules files, custom slash commands: the agent follows your team's conventions by default.
F6	**Agent Skills**	Packaging repeatable workflows into reusable, shareable skills — a key element of team expertise.

F7	**MCP**	Connecting ready-made MCPs and building your own for access to external resources (task tracker, documentation, databases, internal APIs), plus permission management.
F8	**Spec-driven development: writing intent down**	Goals, constraints, definition of done: specs an agent can act on and a reviewer can grade against. One framework in depth (usually OpenSpec), others compared (Spec Kit, BMAD).
F9	**Frontend development with AI**	Everyday agentic workflows on the frontend: scaffolding components, refactoring, styling and design systems, design-to-code (from mockup to working component), UI tests that ride along with every change.
F10	**Backend development with AI**	Everyday agentic workflows on the backend: APIs, data layers, migrations, debugging, integration tests that ride along with every change.
F11	**Code review with AI**	Reading AI diffs like a senior reviewer; review tiered by blast radius; PR review and team rules, review automation.
F12	**Verified documentation**	README, ADRs and onboarding guides generated and maintained by agents alongside the code, kept honest by the same review pipeline as the code itself. Structuring and maintaining documentation optimized for AI agents.

Level 2 — Advanced

Goal: the team stops hand-prompting and starts engineering the system that produces code — orchestrated, verified, and improving itself.

WS	TOPIC	WHAT'S COVERED
A1	**Large projects & legacy**	Code archaeology, safe changes in systems nobody fully understands, compact context indexes.

A2	**Judging AI-generated code**	Comprehension debt in brownfield work: what to trust, what to re-derive, how to keep understanding when most code is generated.
A3	**Shared agent infrastructure — the harness**	Skills, hooks, MCP configuration, sub-agents, durable memory: the shared platform that has the whole team working in synergy.
A4	**Multi-agent orchestration**	Maker/checker separation: the checker runs on a different model with independent context, escalating to a human when needed; isolated worktrees for agent work; running agents in parallel.
A5	**Loops & autonomy engineering**	Long-running agents with explicit stop conditions, guarded overnight loops, evidence-gated merges.
A6	**Verification at scale**	The single home of the verification stack: heterogeneous AI reviewers, red/green TDD, mutation testing mechanics, deterministic CI gates that carry the load human reading no longer can.
A7	**Evals — regression-proofing your agents**	Eval suites that guard prompts the way tests guard code; every failure becomes a permanent rule, skill, or hook (the ratchet).
A8	**Architecture & DevOps with AI**	Architecture decisions, infrastructure-as-code, CI/CD design with agents in the loop.
A9	**Testing & QA with AI**	The developer's view: generating test suites whose worth is proven against seeded defects (verification mechanics live in A6), plus the economics of test generation.
A10	**AI economics & FinOps**	Token economy, model routing, budgets and quotas, first-pass success rate, TCO modeling and ROI a CFO recognizes.
A11	**Software Factories — State of the Art**	What a software factory is and who already runs one: factory types, experience from around the world, why factories fail, practices, security and governance, the economics of autonomous code production.

A12	**Software Factories — How to Build One**	Factory principles shown on a real working factory.
A13	**Greenfield by design — architecture for agent-friendliness**	Starting new systems so agents stay effective as they grow: repository layout, tech stack, conventions, documentation and test scaffolds as first-class design inputs.
A14	**Building production agents**	Agents as product features: agent SDKs, tool design, guardrails, observability, and evals for agents you ship to users.
A15	**RAG & knowledge-base engineering**	Retrieval pipelines over your documentation and code: chunking, embeddings, document hygiene, grounded answers with citations, and when RAG is the wrong tool.
A16	**Incident response and debugging with AI**	Log and telemetry archaeology with agents, hypothesis-driven debugging, and draft postmortems that cut time-to-diagnosis on real incidents.
A17	**AI usage policy, security & governance**	A working policy for what may enter which tool: data classification, tool approval, audit trails, secure defaults. Also available as a standalone module for security teams.

// PROGRAM 2

Agentic QA

AI agents in QA work.

AFTER THIS PROGRAM

How AI agents help your QA team.

WHO IT'S FOR

Manual and automation QA engineers.

FORMAT

Session 0 — technical onboarding (git basics, environment, access checklist).

WS	TOPIC	WHAT'S COVERED
Q1	**Principles of AI agents: delegation, context and cost**	A working mental model of agents, practical prompting.
Q2	**From acceptance criteria to test artifacts**	Acceptance criteria turned into checklists, test cases and traceability — generated, then verified.
Q3	**Ecosystem, rules and guardrails**	The connected, governed agent: tracker, test management and knowledge base over MCP with safe permissions; the team library of rules and skills.
Q4	**Specification as a contract (OpenSpec)**	Testing against written intent, not folklore: the spec as the artifact QA holds everyone to.
Q5	**Technical context from code: a map of systems and integrations**	System and integration maps built by the agent — understanding what talks to what without reading the code.
Q6	**Test design, exploratory testing and test data with AI**	AI-assisted test design and review before any automation; the agent as an exploratory-testing partner; generating test data.
Q7	**Automated tests in natural language and their validation**	Tests written from plain-language scenarios (Playwright as the reference stack; the method transfers to your framework), plus page-object generation for existing apps. Validation: the pipeline breaks the app, and your suite must notice.
Q8	**API tests, regression suites and CI/CD**	API-level testing, regression tests.
Q9	**Defect diagnosis and a draft fix**	A three-step diagnosis discipline that ends with a draft fix as a pull (merge) request: QA proposes, developers decide. Includes bug reports an agent can act on, with reproduction and context.

FINAL PROJECT

You receive a prepared application with seeded defects (logic / boundary / integration classes, injected by the grading pipeline, categories undisclosed). Deliver a test suite and diagnosis packages that catch at

least 60% of the seeded defects. Building your own small product with AI is offered as the L3 stretch option.

// PROGRAM 3

Agentic Business Analysis

AI agents in business-analysis work.

AFTER THIS PROGRAM

Your analysts produce requirements that agents, testers and developers can act on directly: grounded in the real system, with quality scored against a reviewer rubric.

WHO IT'S FOR

Business analysts, product analysts, product owners — no coding required at the entry difficulty level.

FORMAT

Session 0 — technical onboarding; graded homework; final project.

WS	TOPIC	WHAT'S COVERED
B1	**Principles of AI agents: delegation, context and cost**	A working mental model of agents, practical prompting and cost.
B2	**From stakeholder input to acceptance criteria**	From raw conversations, documents and meeting recordings to acceptance criteria that agents, testers and developers can all act on, including process documentation drafted straight from recordings.
B3	**Ecosystem, rules and guardrails**	The tracker and the knowledge base as the agent's working memory over MCP. The analyst team library: story templates, review rubrics, generation guardrails, and stakeholder-communication drafts grounded in the tracker rather than written from memory.
B4	**Specification as a contract (OpenSpec)**	The spec as the artifact everything downstream hangs on, written so an agent can act on it.

B5	**Technical context from code: a map of systems and integrations**	Writing implementation-aware requirements.
B6	**Discovery and decomposition**	From problem statement to epics and stories with an agent: faster drafts, human judgment on scope. The session ends with an epic/story generator packaged as a reusable team skill.
B7	**Prototyping from stories**	Clickable prototypes generated from stories and validated with stakeholders before a line of production code is written, plus UAT scenario packs generated from the same stories.
B8	**Data analysis through script generation**	Data questions answered by scripts the agent writes and the analyst validates. A full session, not an afterthought.
B9	**AI as a reviewer: measurable requirements quality**	The agent reviews specs against a rubric before developers ever see them, and requirements quality becomes a number.

FINAL PROJECT

A complete package from problem statement to stories, prototype, UAT scenarios and tracker tasks. The test: the trainer injects two requirement changes near the end; all downstream artifacts must be updated with zero contradictions found by the AI reviewer.

// PROGRAM 4

Agentic QA + BA (Combined Program)

QA and BA trained as one cohort.

AFTER THIS PROGRAM

One shared language and one team library across QA and BA, and the requirements-to-tests handoff practiced on shared artifacts in the capstone.

WHO IT'S FOR

QA engineers and business analysts of one organization, trained as one cohort.

FORMAT

Joint BA + QA sessions.

WS	SESSION	WHO ATTENDS
C1	Principles of AI agents: delegation, context and cost	all
C2	Requirements analysis and refinement: from acceptance criteria to test artifacts	all
C3	A unified ecosystem: agent, tracker and knowledge base via MCP	all
C4	Specification as a contract: Spec-Driven Development with OpenSpec	all
C5	Rules, skills and guardrails. The team library	all
C6	Technical context from code: a map of systems and integrations	all
QA1	Automated tests in natural language and their validation	QA
QA2	API tests, regression suites and CI/CD	QA
QA3	Defect diagnosis and a draft fix	QA
BA1	Discovery and decomposition	BA
BA2	Prototyping and data analysis through script generation	BA
BA3	AI as a reviewer: measurable requirements quality	BA
C7	The handoff capstone: QA tests what BA specified, in a joint final review on shared artifacts	all

Why combined beats two separate trainings

- The handoff is real: in the capstone, QA finals run against the stories and UAT scenarios the BA track produced.
- One team library of rules, skills and prompts, built and owned together, that outlives the course.
- The foundation is identical, so the combined program is 13 sessions instead of 18.

FINAL PROJECTS

Both role finals (QA: ≥60% of seeded defects caught; BA: the change-injection consistency test), connected in the capstone, plus the shared team library as a cohort deliverable.

// PROGRAM 5

AI Engineering for Leaders

Strategy sessions for engineering leadership.

> AFTER THIS PROGRAM
>
> A defensible AI delivery strategy: what to automate, what to verify, what it costs, and what policy makes it safe. Grounded in evidence, not vendor decks.

WHO IT'S FOR

CTOs, VPs of Engineering, heads of delivery, principal architects. Often the first engagement: the briefing that scopes the cohorts.

FORMAT

2–6 sessions × 2 h, assembled from the topics below at intake; senior audience, intake survey before session one. No homework; an executive summary and readiness map instead.

WS	TOPIC	WHAT'S COVERED
L1	**Software Factories — State of the Art**	What autonomous software production looks like today: the evidence, who runs factories, why most attempts fail, security and governance.
L2	**How Real Factories Are Built**	Factory principles on a working reference factory, holdout verification shown live, separation of duties, and a live factory run with real compute spend.
L3	**The economics of AI delivery**	Token economy, model routing, budgets, first-pass success rate, TCO and ROI: the cost curve and how to invert it.

L4	**Estimation in the agentic era**	Why story points and velocity models break when agents compress implementation time; estimating by verification and review cost instead of typing time; token budgets next to person-hours; forecasting from first-pass success rates; and planning around the new bottleneck, review capacity.
L5	**The new SDLC: process and roles**	What the lifecycle looks like when agents write most of the code: spec-first flow with verification gates; how the developer, QA, BA and architect roles shift; team topology, hiring and seniority in the new pipeline.
L6	**Effective teamwork with agentic artifacts**	Running a team on shared agent infrastructure: team libraries of rules, skills and prompts; ownership and review duty when agents produce most changes; onboarding and knowledge flow. Artifacts that compound in value instead of rotting in a wiki.
L7	**Policy, governance and the adoption plan**	AI usage policy, data classification, verification gates, team readiness, and how to sequence the role programs above.

// CORPORATE AI TRAINING

Pick the program — or let the intake survey pick it

Tell us the roles, the stack and the constraint. We survey the team before session one and shape the curriculum to what the data says.

Book an intro call →

See the proof



This catalog is printed from the live page — the current version is always at
dev-ai.dev/en/training



Specimen certificates and the live progress-journal demo:
dev-ai.dev/en/proof



Book an intro call:
dev-ai.dev/en#contact