

InfoQ Certified AI Engineering Program syllabus

5-week online certification cohort

Week-by-week syllabus

Week 1: Becoming an AI-Native Engineering Team

- **Focus:** We define what "AI-native" engineering means for your team today. You will examine how this shift affects product thinking, SDLC, architectural tradeoffs, engineering roles and daily habits. We explore where AI adds real value beyond the hype and how teams can manage the risk in their AI adoption journeys.
- **Sample discussion task:** Apply Patrick Debois's "4 Patterns of AI Native Development" on how AI is changing where engineers create value in your organization and which of the shifts you are already seeing. Use Ian Thomas's working definition of AI-native engineering to separate "AI-augmented" habits from genuinely AI-native ones, then define one specific AI engineering opportunity in your current organization. Use "AI First, Quality Always: Agentic SDLC Adoption" by Catherine Weeks to understand how teams can maintain risk management and resilience practices when adopting AI-native SDLC .
- **Weekly homework:** Map out your current Software Development Life Cycle. Propose a strategy to integrate AI tools as team members instead of basic utilities. Detail how this integration improves developer experience and productivity. Name the specific friction points you expect to encounter. Identify one part of the SDLC where you will deliberately not let AI in, and explain why that boundary matters. Finally, define one measurable signal you will watch to know the strategy is working.

Week 2: Designing & Building RAG & Context Pipelines

- **Focus:** RAG is the workhorse pattern of enterprise AI, and its quality lives or dies on data, retrieval design, and memory. You will design and build production-grade RAG systems and context engineering pipelines, examine how data processing, retrieval and reranking work together. You will deploy knowledge graphs, and manage both ephemeral and long-term memory so that enterprise AI stays grounded and fresh rather than confidently wrong.
- **Sample discussion task:** Review Adi Polak's framework to identify where traditional RAG breaks down. Use Lan Chu's QCon London case study, *Reliable Retrieval for Production AI Systems*, to examine what it takes to move RAG from a prototype to production, including context processing, retrieval quality, and evaluation. Compare vector RAG against Cassie Shum's Graph RAG across two contrasting use cases (for example, customer support versus code search) and defend when each approach wins. You will then design context pipelines that separate ephemeral context from durable knowledge.
- **Weekly homework:** Build a retrieval architecture for a complex query scenario at your organization. Walk through every stage of the pipeline (sources, chunking and embedding, retrieval, prompt assembly, and memory) and justify the design choice at each layer, including whether vector retrieval or a graph approach better fits the query shape. Apply Ricardo Ferreira's Vector Sync Patterns to explain what happens when the underlying data changes, and name the single biggest failure mode you expect. Name the main context failure you expect, whether that's context collision, stale state, or a latency bottleneck, and lay out how you'd handle governance and observability around it.

Week 3: Designing & Building AI Agents

- **Focus:** You will design agentic systems without turning them into fragile demos. You will leave with a clear breakdown of the spectrum from single-purpose tools to multi-agent systems. We cover workflow patterns, , orchestration, control, memory, and the tradeoffs between autonomy and certainty.
- **Sample discussion task:** Critique an agent architecture (your team's, or a published case study) for the three places agents most commonly fail: ambiguous instructions, tool errors, and runaway loops. Then debate the tradeoffs we weigh in agentic AI: fixed workflows vs. autonomy, a single multi-tool agent vs. many single-tool agents, and composability vs. simplicity. Defend where you would land and what evidence would change your mind. Then

debate Itamar Friedman's question: when does a single, well-tooled agent beat a multi-agent orchestration, and when does the inverse hold? Defend your decision criteria with reference to Jake Mannix's argument that agents should be composed like real software, not stitched together like demos.

- **Weekly homework:** Design (or prototype, if tooling allows) an AI agent for a real task at your organization. Your deliverable must include a tool inventory, the failure modes you anticipate,, and an evaluation plan that tells you whether the agent is actually doing its job. Justify the choice you made for workflows, single or multi-agent, and implement guard rails and safety features that prevent unsafe behavior. Using Adrian Cockcroft's rules and specify what fast iteration in low-stakes environments looks like for your test setup, while maintaining strict BDD hygiene. Be specific about what failure looks like.

Week 4: AI Platforms & Infrastructure

- **Focus:** We focus on the platform and infrastructure patterns that make AI systems production-ready. You will explore what platform capabilities do we need to build, deploy, integrate, govern, and operate AI at scale. You will review inference gateways and GPU cost controls. You will decide what should be centralized and what should remain federated. How are we hosting these systems: on premise or clouds?
- **Sample discussion task:** Meryem Arik points out that most companies overspend on inference. Review the provided system architecture to identify waste, then defend the hardware selections and open-source inference stack choices required to produce the lowest-cost tokens possible without sacrificing throughput. Try to scale it down: what is the difference between serving 25 users vs 25,000 users? Then, whiteboard the gateway policies your org actually needs (model routing, rate limits, cost caps, and PII handling) and identify which policy you would enforce first if you could only ship one.
- **Weekly homework:** Outline an internal AI platform strategy for a mid-sized organization. Using advice from Merrin Kurian and Joseph Stein, detail how the platform will route workloads, separating batch processing from latency-sensitive real-time tasks. Specify the first observability signals you would add to know whether the platform is healthy, and identify one capability you would deliberately keep federated rather than centralized, explaining why federation serves the organization better there. What would break if the number of users grows by a factor ten?

Week 5: AI Operational Excellence: Evals, Trust & Reliability

- **Focus:** The final week covers the operational disciplines that turn AI systems into dependable production assets. We examine evaluation metrics and rollout readiness. Most of this session is devoted to the capstone project presentations.
- **Capstone Project:** Your group will choose one of two tracks and apply the frameworks from Weeks 1–4 to a concrete artifact.
- **Track A (build):** Design a new AI-powered feature your organization would realistically ship. Pick the use case yourselves: a RAG-grounded assistant, an agentic workflow, a structured-extraction pipeline, or another shape that fits your org.
- **Track B (harden):** Take an existing AI feature at your organization that works in demo but is not production-ready, and design what it takes to close the gap. Address Tracy Bannon's "fragile demo" warning head-on.
- Whichever track you choose, your design must walk through the full stack:
 - **From Week 1:** name the parts of the SDLC where AI is load-bearing, the parts where you deliberately keep humans in the loop, and the measurable signal that tells you the integration is working.
 - **From Week 2:** if your feature retrieves context, document the pipeline (sources, chunking and embedding, retrieval, prompt assembly, memory) and explain what happens when the underlying data changes.
 - **From Week 3:** if your feature is agentic, justify single-vs-multi-agent, list the tool inventory, the failure modes, and the escape hatches that prevent runaway behavior.
 - **From Week 4:** state the platform stance: gateway policies, observability signals, cost controls, and which capabilities you keep federated rather than centralized.
- Following Mallika Rao's constraints, layer in an evaluation loop spanning the model, system, product, and user experience. Precision and recall are not enough.
- Drawing on Sriram Madapusi Vasudevan's framing, integrate security policies for AI-accelerated development and address the shifting boundaries of trust between human-written code and AI-generated code.
- The deliverable is a 20-minute group presentation followed by a 30-minute peer discussion. End your presentation with an unresolved architectural question your team is still working out.