AI Engineer Code Summit 2026 — End-to-End Curriculum

Source schedule: https://ai.engineer/code/2026/schedule — 80 sessions extracted; 79 with a verified learning source, 1 honest gap (see end). Ordering below is **not** the conference order. It is derived from dependency: each module uses only what earlier modules built.

The ordering argument, from first principles

1. **See the field before touching it** (M0). 2. **Execution before everything** — a model in a loop in a harness (M1); there is no agent without it. 3. **Knowledge before compute** — context determines whether token spend is signal or noise (M2 before M3). 4. **Compute before autonomy** — inference ownership and cost bound how much autonomy you can afford (M3). 5. **Specification before verification** — a rule must be machine-readable before conformance can be checked (M4 before M5). 6. **Verification before delegation** — trust is the licence to delegate (M5 before M6). 7. **Single before many** — multi-agent multiplies error as fast as output (M6 before M7). 8. **Many before factory** — industrialisation is routine multi-agent work plus process (M7 before M8). 9. **Factory before organisation** — swarms are an organisational claim resting on all prior layers (M8 before M9). 10. **Judgement last** — cost, craft and judgement price the whole stack and decide what should never be automated (M10).

Source tiers: **T1** = talk notes/slides/recap of the actual session · **T2** = speaker's own blog/paper/repo/docs on the topic · **T3** = canonical topic resource.

Module 0 — Orientation: What Changed in 2026

No prerequisites. You cannot order a field you cannot see whole. This module is survey, not skill: the year's model/tool landscape, the ambition question, and the 10x-speed thought experiment that reframes every later module as 'what becomes possible, what becomes the bottleneck'.

Module build: Map the landscape; name your own bottleneck before touching a tool.

1. 2026 in LLMs — Simon Willison ()

Simon Willison's annotated slides and notes give a chronological lightning tour of 2026 in LLMs, from the November 2025 coding-agent inflection through OpenClaw, software factories, and token economics.

Syllabus — core concepts:

- November 2025 inflection: Claude Opus 4.5 and GPT-5.1 made coding agents reliable
- OpenClaw and 'Claws' as a new personal-agent category
- Sandboxing and coding-agent security
- Tokenmaxxing and agent token economics
- StrongDM Software Factory / Dark Factory

Primary source [1-notes]: 2026 in LLMs (so far) **Secondary:** https://simonwillison.net/2025/Dec/31/the-year-in-llms/

2. Ambition in AI — Jason Liu (OpenAI)

Jason Liu argues that when AI makes execution cheap, ambition is limited by owned data/context, ability to describe what you want, and curiosity, with the key failure mode being generation outrunning human judgement.

Syllabus — core concepts:

- When execution is free, ambition is limited by what you own
- What you can describe
- What you are curious enough to pursue
- Generator-discriminator gap
- Iteration and taste: no great work is one shot

Primary source [1-notes]: AI Engineer Code Summit 2026 Day 3 Morning: Ambition, the Software Atelier, and Factory Engineering **Secondary:** https://ai.engineer/code/2026/schedule

3. What Will You Build When AI Is 10x Faster? — Natalie Serrino (Gimlet Labs)

Serrino argues Gimlet's heterogeneous inference — dynamically splitting inference phases across the best-suited chips — makes AI 10x faster, shifting bottlenecks elsewhere and unlocking latency-constrained products like seconds-fast CVE patching by agents.

Syllabus — core concepts:

- heterogeneous multi-silicon inference
- prefill vs decode phase splitting
- 10x speedup exposes next bottleneck (tool calls, network, human approval)
- inference optimization suits verifiable agent tasks
- latency-unlocked product design

Primary source [1-notes]: AI Engineer Code Summit 2026 Day 2: Faster Inference, Smaller Models, and Code You Can Trust **Secondary:** https://gimletlabs.ai/blog

4. A New Age of Software Engineering — Walden Yan (Cognition)

Walden Yan updates his context-engineering principles for an era of exploding Devin usage, where multi-agents work when writes stay single-threaded and other agents contribute review, planning, and intelligence.

Syllabus — core concepts:

- context engineering
- single-threaded writes with multi-agent intelligence
- clean-context code-review loop
- smart friend / capability routing
- higher-level delegation via manager Devin

Primary source [2-speaker-blog/paper/repo]: Multi-Agents: What's Actually Working | Cognition **Secondary:** https://cognition.com/blog/dont-build-multi-agents

Module 1 — The Loop and the Harness

Prerequisite: Module 0. First principle: an agent is a model inside a loop inside a harness — nothing else in this curriculum exists until that loop runs and its failures can be fixed in code rather than in prompts. Minimal harness before maximal harness: Schmid's deletion lesson only makes sense after you have built one.

Module build: Build a harness from an empty loop; then delete 90% of it and measure what actually got worse.

5. Build a Coding Agent Harness From an Empty Loop — Tejas Kumar (IBM)

Tejas Kumar builds a TypeScript coding agent harness layer by layer from an empty loop — adding guardrails, context management, deterministic verification, and retries — without ever changing the prompt, fixing failures in code instead.

Syllabus — core concepts:

- agent loop
- guardrails and context management
- deterministic verification
- tool registry
- retries and secrets kept from the model

Primary source [1-notes]: AI Engineer Code Summit 2026 Day 1: From the Agent Loop to Agent Swarms — Apper AI Solutions **Secondary:** https://tej.as/blog/how-to-build-a-coding-agent-harness

6. Build Your Own Agent Harness with eve and Vercel — Owen Kephart (Vercel)

Vercel's official introduction to eve explains how its built-in harness provides durable workflows, sandboxed execution, and model access so you define an agent as a directory instead of hand-rolling plumbing.

Syllabus — core concepts:

- eve filesystem-first agent directory (agent.ts, instructions, tools, skills)
- Default harness: agent loop, model calls and tool execution
- Durable execution and checkpointed sessions
- Sandboxed isolated code execution
- Context compaction and model access via AI Gateway

Primary source [3-canonical-topic]: Introducing eve - Vercel **Secondary:** https://eve.dev/docs/concepts/default-harness

7. I deleted 90% of my harness (And It Got Better) — Philipp Schmid (Google DeepMind)

Philipp Schmid builds the same PR-review agent three times, deleting code each pass, to show that as models improve the harness should shrink to markdown files in a managed sandbox rather than grow.

Syllabus — core concepts:

- Bitter lesson of agent engineering: delete harness code as models improve
- Agents as files: AGENTS.md and SKILL.md replace Python orchestration
- Managed remote sandboxes and Gemini Interactions API move loop, state and tool routing server-side
- Cursor example: 12,000 lines of TypeScript replaced by ~200 lines of Markdown
- What stays yours: instructions, skills and evaluations

Primary source [2-speaker-blog/paper/repo]: Agents Without Code: Skills, YAML, and Filesystems Replaced Python — Philipp Schmid, Google DeepMind **Secondary:** https://yt-insights-explorer.rmax.ai/videos/agents-without-code-skills-yaml-and-filesystems-replaced-python-philipp-schmid-google-deepmind-fjf8eknxkcu/index.html

8. Rethinking Claude Code Best Practices — Lydia Hallie (Anthropic)

Lydia Hallie argues most Claude Code setup accumulated over the past year now holds back Claude 5.5 models, and walks through updated practices around auto mode, cloud sessions, and hooks.

Syllabus — core concepts:

- Pruning legacy Claude Code setups that now hold back stronger models
- Best practices for Claude 5.5 models: what to keep vs drop
- Auto mode replacing manual command approvals
- Hooks, tests and CI as verification
- Cloud sessions and orchestrator/worker sessions

Primary source [1-notes]: Rethinking Claude Code Best Practices — Lydia Hallie, Anthropic
Secondary: https://www.apper.cloud/blog/ai-engineer-code-summit-2026-day-2-morning-keynotes

9. New Ways of Working with Codex — Dominik Kundel + Charlie Guo (OpenAI)

OpenAI's workshop covered three new Codex workflows: handing local work off to Codex Cloud v2, using voice as a control plane to manage threads and agents, and using personal agents called dots between sessions.

Syllabus — core concepts:

- Codex Cloud handoff and local-to-cloud merging
- Publish an environment once, run isolated tasks
- Voice as a control plane for threads and delegation
- Dots as personal agents: safety net and delegation sink

Primary source [1-notes]: AI Engineer Code Summit: OpenAI on New Ways of Working With Codex — Latent Space **Secondary:** https://aibreakingwire.com/news/openai-workshop-shows-codex-cloud-handoff-voice-control-and-dots

10. Exploring Claude's frontier — Lance Martin (Anthropic)

Most Claude applications sit inside the cost/quality frontier because of fixable misconfigurations — stale prompts, broken caching, unclimbable evals, overfit hill-climbing and harnesses not built for multi-day agents.

Syllabus — core concepts:

- Prompt debt and prompt-audit
- Prompt caching as strict prefix match
- Climbable evals with binary rubrics
- Hill-climbing with held-out test split
- Harnesses for multi-day agents: decoupled brain and hands

Primary source [1-notes]: Your Old Prompts Are Making Claude Worse (Lance Martin, Anthropic) | Sparse Notes **Secondary:** https://www.youtube.com/watch?v=ys6YQ2yfBpM

11. Coding Agents and How to Tame Them — Ivan Leo (Google DeepMind)

Ivan Leo's own benchmark post shows how Harbor-packaged tasks, sandboxes and verifiers reveal where coding-agent harnesses break — the same auditing, sandbox and verifier themes as his talk.

Syllabus — core concepts:

- auditing coding benchmarks
- agent harnesses and Harbor evaluations
- isolated sandboxes and verifiers
- execution-trace failure analysis
- tool design as bottleneck

Primary source [2-speaker-blog/paper/repo]: How Simple Grep Beats Naive SQL | Ivan Leo **Secondary:** https://ai.engineer/code/2026/schedule

Module 2 — Context: What the Agent Knows

Prerequisite: Module 1. A loop with no knowledge only hallucinates faster. Context — your data, retrieval, search — must be engineered before inference is optimised, because wrong context invalidates every downstream peso spent on tokens.

Module build: Build a personal agent over your own data with a measured search/token budget.

12. The new job of the data team is providing context to AI — George Fraser (Fivetran)

George Fraser argues data teams must shift from building dashboards to giving AI agents trusted, governed business context via semantic models on an open data platform.

Syllabus — core concepts:

- AI as primary consumer of enterprise data
- Semantic models and context quality over quantity
- Trusted, governed business context for agents
- Open Data Infrastructure (Fivetran + dbt)
- Data teams providing context instead of dashboards

Primary source [2-speaker-blog/paper/repo]: Connecting Enterprise Data To The Age Of Ai Agents **Secondary:** https://www.getdbt.com/blog/what-fivetran-dbt-labs-brought-to-databricks-

data-ai-summit-and-what-you-can-take-home

13. How to build agents over your own data — Aamir Shakir (Mixedbread)

Shakir argues agents fail from poor access to their own data, and shows Mixedbread's visual late-interaction search replacing the lossy OCR/chunk/vector stack to lift a hedge-fund agent from 36% to 98% accuracy.

Syllabus — core concepts:

- Frankenstein RAG stack: OCR, chunking, single-vector embeddings are lossy
- Late-interaction multi-vector retrieval
- Visual page embeddings preserving tables and figures
- Agentic search with multi-hop query decomposition
- Retrieval/access bottleneck, not model intelligence — 36% to 98% case

Primary source [1-notes]: Better Search Took This AI Agent From 36% to 98% Accuracy - Aamir Shakir, Mixedbread **Secondary:** https://www.startuphub.ai/ai-news/artificial-intelligence/2026/better-search-pushed-an-agent-from-36-to-98

14. Connect Agents to Reality: The Context Engine — Brandon Waselnuk (Unblocked)

Brandon Waselnuk argues coding agents fail for lack of organizational context, not intelligence, and that a context engine connecting code, docs, conversations and tickets — resolving conflicts and returning a permission-aware, token-optimized slice — roughly halves token spend and eliminates correction doom loops.

Syllabus — core concepts:

- Context engine: six requirements — unified context, conflict resolution, token optimization, targeted retrieval, personalized relevance, permissions enforcement
- Curated-context trap and MCP plateau / satisfaction of search
- Conflict scoring by recency, authority and proximity (social graph)
- Agentic retrieval across code, docs, Slack, tickets and incidents
- Measuring correction loops, time and tokens with vs without context

Primary source [2-speaker-blog/paper/repo]: Building the Context Engine AI Agents Need: QCon Boston 2026 | Unblocked **Secondary:** https://getunblocked.com/videos/your-agents-lack-context-how-to-fix-youre-absolutely-right/

15. What Your AI Doesn't Know: trends in agent searches — Jeffrey Pinner (Exa)

Exa's ATLAS blog shows, from clustering anonymized real agent search demand, that frontier models have memorized older benchmarks but miss about a third of fresh, comprehensive search results — exposing what AI doesn't know without search.

Syllabus — core concepts:

- agent search trends from real search demand
- memorization vs retrieval — what models don't know
- ATLAS agentic search benchmark
- wide and deep comprehensive search
- cost-performance of search backends

Primary source [2-speaker-blog/paper/repo]: ATLAS: Evaluating Agents on Search-Intensive Tasks | Exa Blog **Secondary:** https://render.com/blog/search-for-agents-splits-discovery-from-proof

16. Your agent pays twice for every search: cut token spend 90% — Louis Walsh (Exa)

Exa's official coding-agents page explains the talk's thesis: agents pay both a search fee and token costs to read full pages, and Exa Highlights return only query-relevant excerpts, cutting context tokens by 95% on average.

Syllabus — core concepts:

- Paying twice: search API fee plus LLM input tokens to read results
- Exa Highlights / Dynamic Highlights vs full-page text
- 95% average context-token reduction with highlights
- Content budget controls: excerpt size, maxCharacters, result count

Primary source [3-canonical-topic]: Current Docs and Code Search for Coding Agents | Exa **Secondary:** https://www.beri.net/article/tavily-vs-exa-vs-brave-vs-perplexity-search-api-agent-cost-2026

17. Build a personal agent with Exa Search — Jakub Hojsan (Exa)

Exa's official guide explains how search agents work and walks through building one on the Exa Search API, plus using the managed Exa Agent API for structured, cited output.

Syllabus — core concepts:

- Search agent loop: intent detection, sub-queries, retrieve-read-re-search
- Exa Search API with token-efficient highlights
- Exa Agent API with structured output schema and grounding

- Cited synthesis with sources dictionary and stop conditions

Primary source [3-canonical-topic]: What is a search agent? How they work and how to build one **Secondary:** https://milvus.io/blog/i-built-a-stock-monitoring-agent-with-openclaw-exa-and-milvus-for-20month.md

Module 3 — Inference: Where It Runs and What Thinking Costs

Prerequisites: Modules 1–2. Only once the loop and its context exist does it matter where inference runs, who owns it, and how it is priced — KV cache, local vs frontier, post-training, and structured/fast decisions (Jev, System One) as the alternative to generation for everything.

Module build: Run the same harness against local, open-weights, and frontier inference; log cost/latency/quality per task.

18. Inference Engineering for Coding Agents — Philip Kiely (Baseten)

Philip Kiely's own book Inference Engineering maps the runtime, infrastructure and tooling techniques — KV cache re-use, speculative decoding, quantization and disaggregation — for adapting inference to the work coding agents actually do.

Syllabus — core concepts:

- KV cache management and re-use for coding agents
- Speculative decoding
- Quantization and model parallelism
- Latency vs throughput and prefill/decode disaggregation
- Adapting inference to agentic coding workloads

Primary source [2-speaker-blog/paper/repo]: Inference Engineering | Baseten Books **Secondary:** https://www.complexsystemspodcast.com/episodes/inference-engineering-with-philip-kiely/

19. Keep the Harness, Own Your Inference — Francisco Javier Arceo (Baseten)

Arceo showed Claude Code and Codex running unchanged against open-weight models via the vLLM Agentic API, arguing owning inference means controlling the agentic server stack — state, retrieval and compaction — not just the model.

Syllabus — core concepts:

- vLLM Agentic API
- Claude Code and Codex on open-weight models

- owning inference on own hardware (DGX Spark)
- llm-d and NVIDIA Dynamo distributed serving
- server-managed state, retrieval, file search and compaction

Primary source [1-notes]: AI Engineer Code Summit 2026 Day 1: From the Agent Loop to Agent Swarms — Apper AI Solutions **Secondary:** https://franciscojavierarceo.dev/

20. What Runs Where: The State of Local AI — Ahmad Osman (Osmantic)

Ahmad Osman explains, in his own words, what local models can run on phones, laptops, DGX Spark/Strix Halo and GPU workstations versus the cloud, and why hybrid sovereign deployment is rising.

Syllabus — core concepts:

- Local vs frontier gap — open models 4-8 months behind and closing
- Hardware tiers: phone, MacBook (4-bit Qwen), RTX 5090, DGX Spark / AMD Strix Halo, multi-GPU enterprise
- A model is only part of the system — harness, search, tools and document ingestion
- Hybrid and sovereign AI — dedicated/controlled compute for privacy and control
- Smaller specialized models tuned from enterprise traces

Primary source [2-speaker-blog/paper/repo]: Ahmad Osman on why local AI is catching up **Secondary:** https://www.ahmadosman.com/tags/ai/

21. Post-Training Gemini 4 for Real Engineering Work — Melvin Johnson (Google DeepMind)

Melvin Johnson's own site confirms he leads Gemini post-training from Gemini 1.0 to Gemini 4 Argon, covering RL, reasoning, alignment and evaluation for real engineering use.

Syllabus — core concepts:

- Gemini post-training
- reinforcement learning and RLVR
- reasoning and alignment
- agentic training for engineering work
- evaluation and unified modeling recipes

Primary source [2-speaker-blog/paper/repo]: Melvin Johnson — Google DeepMind **Secondary:** https://www.gitechdesign.com/cs329a/guest-lectures/

22. I love you so jevving much / Jev — Diogo Almeida (TypeSafe AI)

Diogo Almeida's launch post introduces Jev as TypeSafe's first System One model — unstructured state in, typed probabilistic decisions out — trained with RLCD for fast, calibrated decisions inside software instead of chat.

Syllabus — core concepts:

- System One Models
- Reinforcement Learning for Calibrated Decisions (RLCD)
- Typed probabilistic decisions: Choice / Score / Noul
- Parallel sampler vs sequential token generation
- 70-500ms, $0.042/MTok automation workflows

Primary source [2-speaker-blog/paper/repo]: Introducing System One Models & Jev - TypeSafe AI Blog **Secondary:** https://ai.engineer/speakers/diogo-almeida

23. Beyond Generation: Fast, Structured AI Decisions with Jev at Glean — Chinmay Goyal (Glean)

Latent Space's AIE Code recap covers Chinmay Goyal's talk on how Glean uses Jev's zero-shot System One decisions alongside LLMs, hiding model complexity from users while powering many flows.

Syllabus — core concepts:

- System One models vs generative LLMs
- Jev structured decisions: choices, scores, probabilities
- Zero-shot flexibility without fine-tuning as taxonomies change
- Using Jev alongside LLMs and classifiers at Glean
- When to use Jev: rules and patterns for production flows

Primary source [1-notes]: The Jev Revolution: What Engineers Are Already Building with System One Models **Secondary:** https://www.edenai.co/post/jev-a-new-kind-of-ai-model-built-for-decisions-not-conversation

24. Autocomplete for Actions: System One — Scott Roberts (Workday)

Scott Roberts showed Workday prototypes using Jev System One models to pre-fill actions like a time-off card while the user is still typing, shifting from predicting what you want to say to predicting what you want to do.

Syllabus — core concepts:

- System One models
- Jev by TypeSafe AI

- Autocomplete for actions
- Predicting what you want to do vs say
- Fast structured inference inside application UI

Primary source [1-notes]: The Jev Revolution: What Engineers Are Already Building with System One Models **Secondary:** https://ai.engineer/code/2026/schedule

25. Jarred Sumner / Bun — Jarred Sumner (Anthropic/Bun)

Jarred Sumner's own Bun blog post detailing how his team rewrote 535,496 lines of Bun from Zig to Rust in 11 days using Claude Code dynamic workflows — the same Rust rewrite and Bun AOT preview anchoring his AIE Code 2026 keynote.

Syllabus — core concepts:

- Rewriting Bun in Rust from Zig
- Claude Code dynamic workflows
- Adversarial code review
- Bun AOT compiler preview
- Memory safety and Rust borrow checker

Primary source [2-speaker-blog/paper/repo]: Rewriting Bun in Rust **Secondary:** https://claude.com/code-with-claude/session/tyo-ext-rewriting-bun-in-rust

Module 4 — Specification and Guardrails

Prerequisites: Modules 1–3. Before an agent may act unattended, what it may do must be stated enforceably — ontologies/contracts, credential separation, human-in-the-loop gates, grounded decisions. Specification precedes verification: you cannot check conformance to a rule that was never made machine-readable.

Module build: Write SHACL-style guardrails and an approval matrix for your Module 2 agent; try to make it violate them.

26. From Shared Language to Enforceable Contracts: Ontologies as Guardrails for Agents — Frank Coyle (UC Berkeley)

A blog recap of Coyle's Code Summit talk showing how a formal ontology with SHACL shapes mechanically validates agent actions before side effects, replacing prompt advice with enforceable domain rules.

Syllabus — core concepts:

- Ontologies as enforceable contracts, not prompts

- SHACL shapes and pySHACL validation
- LLM proposes, ontology disposes
- Neurosymbolic AI: probabilistic reasoning inside, logic outside
- Pydantic at the door, ontology at the ledger

Primary source [1-notes]: Your Agent Needs a Schema, Not a Better Prompt - DEV Community **Secondary:** https://www.youtube.com/watch?v=Sir59K8ZDPU&pp=ugUHEgVlbi1VU9IHCQk1DAGHKiGM7w%3D%3D

27. Machine trust — CJ Avilla (Anthropic)

CJ Avilla's Anthropic guide to trusting unattended agents: keep real credentials in vaults outside the sandbox, separate the brain from the hands, and verify and confirm every action before recording it.

Syllabus — core concepts:

- Brain/hands separation: inference separate from execution sandbox
- Credential vaults outside sandbox with placeholder tokens swapped at egress for allowlisted hosts
- Layered trust: separate generator and judge agents, trusted maintainers over star topology
- Guardrails: read-only access and per-run spending cap
- Verify before posting and confirm delivery before advancing bookmarks/ledger

Primary source [2-speaker-blog/paper/repo]: Building effective agent automations **Secondary:** https://finance.biggo.com/news/9ecaa6d8d3a2adb8

28. Talk But Verify: human-in-the-loop for agents — Michael Liendo (Auth0)

Detailed notes on Liendo's three human-in-the-loop patterns for agents — Interrupt, Token Vault, and CIBA — scaling security with the stakes of the action.

Syllabus — core concepts:

- Interrupt pattern: mid-task pause for human answer
- Token Vault: short-lived, narrowly scoped tokens on retry
- CIBA (Client Initiated Backchannel Authentication)
- Match the human-in-the-loop pattern to the stakes

Primary source [1-notes]: Trust, But Verify: Human-in-the-Loop for Agents That Actually Matter · CascadiaJS 2026 Notes **Secondary:** https://www.commityourcode.com/2026/agenda

29. The first Grounded Decision Model — Devin Gupta (Parallel)

Devin Gupta's Parallel post open-sources PWS-Decisions, a 300-question benchmark showing the Parallel Decisions API — the first decision model grounded in the live web — outscoring ungrounded competitors by 1.5x on decisions that depend on current web facts.

Syllabus — core concepts:

- Web-grounded decision models
- Parallel Decisions API
- System One models (Jev)
- PWS-Decisions benchmark
- Typed, cited answers in under a second

Primary source [2-speaker-blog/paper/repo]: PWS-Decisions: a benchmark for web-grounded decisions | Parallel **Secondary:** https://parallel.ai/blog/decisions-api

Module 5 — Verification, Trust and Security

Prerequisite: Module 4. Deterministic verification overrules a model's claim of 'done'. This is the trust layer — evals that heal rather than grade, hallucinating judges, oracles, CI verification, layered security, CVE remediation — without which delegation is negligence.

Module build: Add a verification gate to your harness: tests + lint + an independent judge; record one false-done it catches.

30. One team's path to trusting AI generated code — Ankit Jain (Aviator)

Ankit Jain argues Aviator learned to trust AI-generated code by moving review upstream to intent, codifying recurring feedback into invariants, and letting machines verify while humans judge evidence.

Syllabus — core concepts:

- Intent-driven development — review intent and acceptance criteria, not line-by-line diffs
- AI Slop Register and invariants codified from recurring review comments
- Five-layer trust model for AI-generated code
- Verification by machines, judgment by humans (Aviator Verify)
- Moving code review upstream before the code is written

Primary source [2-speaker-blog/paper/repo]: Move code review before the code - Aviator Blog **Secondary:** https://yt-insights-explorer.rmax.ai/videos/how-to-kill-the-code-review-ankit-jain-aviator-ygev7iqzgdm/index.html

31. Closing the Agentic Loop: Getting Agents to Safely Build, Verify, and Ship — Kent C. Dodds ()

Kent C. Dodds' official talk page details his hands-on workshop on trusting agent-shipped code via autopilot foundations, separate verification agents, and blast-radius controls fed by user feedback and friction loops.

Syllabus — core concepts:

- kcd-skills autopilot: invariants, ADRs and primitives catalog
- AGENTS.md as a router, not a novel
- Separate reviewer agents and system-level PR templates
- Blast-radius limits: feature flags, health checks, tested restores
- Feedback loops: bug reports, friction reports and nightly cleanup

Primary source [1-notes]: Closing the Agentic Loop: Getting Agents to Safely Build, Verify, and Ship by Kent C. Dodds **Secondary:** https://dev.to/dailycontext/kent-c-dodds-hasnt-looked-at-his-code-all-year-gd5

32. Hands-on Gitar: An AI Code Verification Agent for Your CI — Taylor Luttrell-Williams (Sonar)

A hands-on workshop showing Sonar's Gitar verifying AI-generated PRs for business intent, diagnosing CI failures, and auto-fixing them until the build is green.

Syllabus — core concepts:

- Intent-based verification vs syntax-only static analysis
- Autonomous review-fix loop iterating until CI passes
- .gitar/rules and review directory for business-rule enforcement
- Hands-on Boxoffice ticketing API with three staged PRs
- Gitar complementing SonarQube in CI

Primary source [1-notes]: Sonar's Gitar fixes what code review misses | StartupHub.ai **Secondary:** https://yt-insights-explorer.rmax.ai/videos/catch-the-bugs-your-code-review-misses-killian-carlsen-phelan-taylor-luttrell-williams-sonar-wp3p89yb9fa/index.html

33. Heal Your Agents, Don't Just Grade Them — Khasimul Asif Mohammad + Sudhir Tonse (Splunk)

Splunk's practitioner guide shows how to heal production agents — not just grade them — by tracing every step, diagnosing failure patterns, turning patterns into evaluators, and promoting trusted evals to guardrails.

Syllabus — core concepts:

- agent observability beyond final-output grading

- traces, sessions and spans for root-cause analysis
- Signals: clustering production failures into new evaluators
- promoting trusted evals to runtime guardrails
- Luna small evaluator models for low-cost continuous evals

Primary source [3-canonical-topic]: What Is AI Agent Observability? A Practitioner Guide **Secondary:** https://www.splunk.com/en_us/blog/author/khasmoha.html

34. Your Judge Is Hallucinating Too — Brian Kennedy (ZoomInfo)

LLM judges can be highly reliable yet invalid — raw agreement overstates discrimination by 33-41pp and position bias persists — so judges must be validated against human gold sets before trusting them.

Syllabus — core concepts:

- LLM-as-a-Judge validation
- Position, verbosity and self-preference bias
- Cohen's kappa vs exact-match agreement
- Consistency-bias paradox
- Minimum Viable Validation Protocol

Primary source [3-canonical-topic]: Reliability without Validity: A Systematic, Large-Scale Evaluation of LLM-as-a-Judge Models Across Agreement, Consistency, and Bias **Secondary:** https://www.aievals.co/learn/llm-as-judge/introduction

35. Ground Truth or Dare: Using Oracles to Make Postgres Reactive — James Arthur (Databricks)

James Arthur's Neon team makes Postgres reactive by running live SQL queries inside Lakebase Postgres itself, so any deterministic query stays up to date without a separate sync service, polling, or refetch logic.

Syllabus — core concepts:

- Live SQL queries as subscriptions
- Postgres as ground-truth oracle
- Incremental reactivity without polling or refetching
- Electric sync engine rebuilt inside Neon/Lakebase
- Scale-to-zero and branching for reactive apps

Primary source [2-speaker-blog/paper/repo]: Neon now has realtime - Neon **Secondary:** https://neon.com/blog/electrifying-neon-building-the-realtime-backend

36. More Checks, Less Noise: Layered Security for AI-Assisted Development — Taylor Luttrell-Williams + Killian Carlsen-Phelan (Sonar)

Sonar frames layered security for AI-assisted development as deterministic algorithmic checks paired with agentic verification for intent and logic flaws, with quality gates and noise reduction so more checks do not mean more false positives.

Syllabus — core concepts:

- Zero-trust multilayered verification
- Algorithmic layer: SonarQube SAST/SCA, secrets and taint analysis
- Agentic layer: Gitar, Hunter Agent and Remediation Agent
- Logic-level vulnerabilities SAST misses
- Adversarial validation and noise gates to cut false positives

Primary source [2-speaker-blog/paper/repo]: AI Code Verification Tools | Verify AI-Generated Code | Sonar **Secondary:** https://finance.biggo.com/news/99946b594137a6eb

37. Building a Safe Software Factory — Andrew Finch (Sonar)

Sonar's framework for making agentic software factories safe by building intent, guardrails, independent verification, and auditable evidence into the production line rather than reviewing after generation.

Syllabus — core concepts:

- code-confidence formula: intent, guide rails, verification, evidence
- independent deterministic verification with SonarQube quality gates
- Sonar Vortex in-loop context and guardrails for agents
- verification bottleneck and cognitive surrender in review
- containment and auditable evidence before deployment

Primary source [3-canonical-topic]: The Machine that Changed the World, vAI — AI Engineer NYC 2026 **Secondary:** https://www.oreilly.com/radar/human-judgment-doesnt-leave-the-software-factory-it-relocates/

38. CVE remediation with Factory — Nikhil Harithas (Factory)

Factory's Droids automate CVE remediation from scanner finding through dependency upgrades and breaking-change fixes to a reviewable, merged PR.

Syllabus — core concepts:

- CVE triage and finding affected repos
- Automated dependency upgrades

- Fixing breaking changes from upgrades
- Droids opening PRs for review and merge
- Prioritising by exploit maturity and reachability

Primary source [2-speaker-blog/paper/repo]: Factory Events · Events Calendar **Secondary:** https://snyk.io/news/snyk-factory-announce-partnership/

Module 6 — Delegation and Agent Shapes

Prerequisites: Modules 1–5. Delegation is earned: a verified single agent, given a mission instead of a prompt, in the right shape for the horizon — proactive, long-horizon, or open-ended. Autonomy here is scoped and reversible by construction.

Module build: Convert one prompted task into a delegated mission with acceptance criteria and a workstream trail.

39. Stop Prompting, Start Delegating: Hand off work to a team of agents — Ross Cefalu (Factory)

A canonical profile of Factory's Droids and delegated engineering model, covering Missions that decompose complex work into parallel multi-agent tracks while humans plan, inspect, and review.

Syllabus — core concepts:

- delegated engineering vs prompting
- Factory Droids
- Missions: multi-agent parallel tracks
- parallel task launch with human review
- planning and verification

Primary source [3-canonical-topic]: Factory.ai — Latent Space **Secondary:** https://ai.engineer/code/2026/schedule

40. A Field Guide to Agent Shapes: From Long-Horizon to Open-Ended — Yacine Mahdid (Datadom)

Yacine Mahdid's own tutorial maps the main agent shapes — chaining, routing, parallelization, orchestrator-workers and evaluator-optimizer — and argues for simple agentic systems, matching the talk's field-guide framing from long-horizon to open-ended agents.

Syllabus — core concepts:

- Agent vs workflow distinction
- Prompt chaining, routing and parallelization

- Orchestrator-workers pattern
- Evaluator-optimizer pattern
- Simple agentic systems over fully autonomous agents

Primary source [2-speaker-blog/paper/repo]: Free Video: Introduction to AI Agents - Theory and Code from Yacine Mahdid | Class Central **Secondary:** https://www.classcentral.com/course/youtube-what-are-multi-agent-systems-theory-and-code-overview-with-metagpt-446195

41. Proactive Agent Engineering — Sydney Runkle (LangChain)

Sydney Runkle's LangChain loop-engineering framework explains how event-driven loops turn reactive agents into proactive, always-on assistants, with Harrison Chase's companion post — her linked pre-read for this talk — detailing the memory, async, and goal-tracking challenges.

Syllabus — core concepts:

- Proactive, always-on agents: agent gives you tasks, not vice versa
- Event-driven loop: cron, webhooks and heartbeats
- Memory as a step change for background agents
- Async subagents and goal tracking
- Proactive without being annoying: UX calibration

Primary source [2-speaker-blog/paper/repo]: The Art of Loop Engineering **Secondary:** https://www.langchain.com/blog/context-engineering-challenges-late-2026

42. Let the Agent Cook: From APIs to Answers — Arsalan Bashir (Sentry)

Sentry's official guide to giving coding agents production context via MCP and CLI so they can go from raw APIs/issues to diagnosed answers and draft PR fixes.

Syllabus — core concepts:

- Sentry MCP and CLI as agent API surface
- Production context: stack traces, traces, breadcrumbs
- Seer root-cause analysis
- Alert-to-draft-PR agent workflow
- Querying production data for answers

Primary source [3-canonical-topic]: Your agent can't fix what it can't see | Sentry Blog **Secondary:** https://sentry.io/resources/automated-debugging/

43. Earning Autonomy: From Missions to Workstreams — Luke Alvoeiro (Factory)

Factory's Missions let a Droid earn autonomy over multi-day workstreams via an orchestrator that plans and writes a validation contract, ephemeral workers that implement features serially, and adversarial validators that verify each milestone.

Syllabus — core concepts:

- Factory Missions
- orchestrator-worker-validator architecture
- validation contracts
- serial execution with targeted parallelism
- human attention bottleneck and earned autonomy

Primary source [2-speaker-blog/paper/repo]: Introducing Missions | Factory **Secondary:** https://github.com/samanvaya5/ai-engineer-talks/blob/HEAD/per-talk/aie-europe/day2-luke-alvoeiro-factory-agent-missions.md

Module 7 — Multi-Agent Systems

Prerequisite: Module 6. Parallelism multiplies both output and error; it is only safe on top of Modules 4–5. PR review/repair fleets, Devin, Kiro Crew, always-on and remote/AFK operation live here — many agents, one verification spine.

Module build: Stand up a two-agent review-and-repair workflow on a real repo; keep a human merge gate.

44. Build a Multi-Agent PR Review and Repair Workflow with Omnigent — Elise Gonzales + Bradford Chang (Databricks)

Elise Gonzales demos Omnigent as a meta-harness that composes multiple coding agents (including Polly for multi-agent work) with policies, sandboxing and collaboration — the foundation for the workshop's PR review-and-repair workflow.

Syllabus — core concepts:

- Omnigent meta-harness over Claude Code, Codex and Pi
- Multi-agent PR review and repair with Polly orchestrator
- Agent roles, permissions and contextual policies
- Cross-agent review and tested fixes before human approval
- Version-controlled agent configurations and GitHub CI workflow

Primary source [2-speaker-blog/paper/repo]: Databricks Omnigent Deep Dive with Matei Zaharia: Collaboration and Control Layer for AI Agents **Secondary:** https://www.databricks.com/blog/introducing-omnigent-meta-harness-combine-control-and-share-your-agents

45. Do it All with Devin — Giovanni D'Antonio + Charlie Meyer (Cognition)

Cognition's own guide to using Devin as a teammate across every interface — delegating PRs, reviews, bug triage, data analysis and recurring work via Playbooks, MCPs and the API.

Syllabus — core concepts:

- Devin across web, Slack, Linear, CLI and API
- Ask Devin and DeepWiki for codebase Q&A and scoping
- Devin Review, autofix and automated code review
- Playbooks, MCP Marketplace and automations
- Automated bug triage and end-to-end debugging

Primary source [2-speaker-blog/paper/repo]: How Cognition Uses Devin to Build Devin
Secondary: https://cognition.com/blog/series-e

46. Build your first Kiro Crew app — Arundeep Nagaraj + Dan Kiuna (AWS Kiro)

Official Kiro Crew docs walk through building, installing and running your first Crew app in five minutes with a manifest, one agent, one skill and a dashboard UI page.

Syllabus — core concepts:

- app.json manifest
- Crew agents and skills (SKILL.md)
- Dashboard UI with @kirocrew/app-sdk
- Cron-scheduled agents
- Install, enable and App Store publishing

Primary source [3-canonical-topic]: Build your first app - Apps - Crew - Docs - Kiro **Secondary:** https://dev.to/aws-builders/how-i-built-a-kiro-crew-app-in-5-minutes-full-tutorial-with-code-3el0

47. Ship more as a team with Kiro Crew — Doug Clauson (AWS)

Official Kiro launch post explaining how Kiro Crew lets one developer hand off tickets, incidents and migrations to persistent, parallel agents that keep working across sessions.

Syllabus — core concepts:

- async persistent agent workspace
- parallel sub-agents and cross-session coordination
- shared memory, lessons and reusable skills
- schedules, heartbeats and webhooks for unattended work

- sandboxing, approvals and signed audit log

Primary source [3-canonical-topic]: Introducing Kiro Crew - Kiro **Secondary:** https://kiro.dev/blog/introducing-workflows/

48. Agents That Don't Clock Out — Erik Hanchett (AWS Kiro)

Erik Hanchett's speaker slides on building persistent 'Claw' agents with Kiro Crew that run on a schedule, remember across sessions, and keep working unattended on your own hardware.

Syllabus — core concepts:

- persistent always-on agents
- Kiro Crew self-hosted workspace
- scheduled cron and heartbeat jobs
- memory and skills across sessions
- autopilot, sandboxing and governance

Primary source [2-speaker-blog/paper/repo]: Beyond the Claw: Building Your Personal Coding Agent **Secondary:** https://kiro.dev/docs/crew/

49. AFK Engineering: From Terminal to Voice-Controlled Remote Agents — Philipp Schmid + Thor Schaeff + Ivan Leo (Google DeepMind)

Philipp Schmid and Thor Schaeff's earlier DeepMind workshop shows the voice-controlled agent stack behind AFK Engineering — building conversational, tool-using coding agents with Gemini's Live API for realtime audio, tool calls, and remote control.

Syllabus — core concepts:

- Gemini Live API realtime voice interfaces
- Voice-controlled remote coding agents
- Tool-using agents and the Interactions API
- Streaming audio via WebSockets and session management
- AFK / away-from-terminal agent workflows

Primary source [2-speaker-blog/paper/repo]: Building Conversational Agents — Thor Schaeff and Philipp Schmid, Google DeepMind **Secondary:** https://ai.engineer/code/2026/schedule

Module 8 — Software Factories

Prerequisites: Modules 6–7. When delegation and verification are routine, the unit of engineering stops being the file and becomes the factory: 12-Factor patterns, brownfield factories, self-improving codebases, structural migrations, review without reading diffs, agent-native environments.

Module build: Run one ast-grep structural migration end-to-end through your factory pipeline.

50. 12-Factor-Factories: Design Patterns for Agentic Coding — Dexter Horthy (Factory)

Dexter Horthy's own 12-Factor Agents guide codifies the design patterns — owning prompts, context and control flow and using small focused agents — that his 12-Factor-Factories talk extends to agentic coding factories.

Syllabus — core concepts:

- Own your prompts, context window and control flow
- Tools are just structured outputs
- Small, focused agents as stateless reducers
- Software factories and lights-off coding failure modes
- Human-in-the-loop contact via tool calls

Primary source [2-speaker-blog/paper/repo]: GitHub - humanlayer/12-factor-agents: What are the principles we can use to build LLM-powered software that is actually good enough to put in the hands of production customers? **Secondary:** https://ai.engineer/talks/8kMaTybvDUw-12-factor-agents

51. Software Factories: Escape from Trash Mountain — Steve Yegge ()

Yegge's own Gas Town page explains his first software factory — dozens of autonomous coding agents coordinated via Beads — and how it evolved into Gas City, the backdrop for escaping agent-generated slop and debt.

Syllabus — core concepts:

- Software factories / dark factories
- Gas Town and Beads multi-agent orchestration
- Agent-generated slop and technical debt — Trash Mountain
- Mechanical constraints, governors and quality spend
- Gas City as Gas Town's successor SDK

Primary source [2-speaker-blog/paper/repo]: Gas Town — Steve Yegge **Secondary:** https://re-cinq.com/podcast/steve-yegge-software-factory

52. Software Engineering Is Becoming Factory Engineering — Zach Lloyd (Warp)

Zach Lloyd argues engineers should stop prompting agents and instead engineer code-defined, closed-loop cloud software factories that continuously measure and improve coding agents, with

humans only at spec, code, and product review checkpoints.

Syllabus — core concepts:

- code-defined closed-loop software factories
- factory engineering vs product engineering
- cloud factory loop: triage-spec-implement-verify-ship-monitor
- factories stack and measurement (evals, benchmarks, A/B tests)
- self-improvement loops and human judgement checkpoints

Primary source [1-notes]: AI Engineer Code Summit 2026 Day 3 Morning: Ambition, the Software Atelier, and Factory Engineering **Secondary:** https://www.youtube.com/watch?v=LL0YmIyBKSU

53. The Software Atelier — Armin Ronacher (Earendil)

Ronacher argues engineers should master working with agents in a small, craft-focused atelier with the Pi harness before building software factories, because human attention and judgement have not scaled with model capability.

Syllabus — core concepts:

- Atelier vs. software factory
- Pi stable harness — models improve, harness stays same
- Human attention and judgement: saying 'not like this'
- Harness evolution: durability, codemode, classifier models
- Resist building your own — open protocols over Wildwuchs

Primary source [1-notes]: AI Engineer Code Summit 2026 Day 3 Morning: Ambition, the Software Atelier, and Factory Engineering **Secondary:** https://lucumr.pocoo.org/2026/6/23/the-coming-loop/

54. Brownfield Factories — Justin McCarthy (Diffusion)

Justin McCarthy explains how Diffusion/StrongDM-style software factories — where agents implement and evaluate software from specs — can be run inside large existing (brownfield) enterprises rather than only on greenfield code.

Syllabus — core concepts:

- dark software factories in brownfield enterprise systems
- no human writes or reviews code
- scenario-based validation and digital twin universe
- engineering the token environment for inevitable agent success
- existing tickets, logs and runbooks as factory raw material

Primary source [2-speaker-blog/paper/repo]: AI:AM LIVE — September 17, 2026 — Aaronson Says the Takeoff Already Started and Esvelt Reopens the Bio Question, Diffusion's Justin McCarthy on Software Factories, Five Nines and Buying an Airline's Data, and Cameron Berg on Suleyman, a Pain Direction in Five Model Families, and the Relief Button **Secondary:** https://web.navan.dev/posts/2026-05-06-how-to-build-your-own-software-factory.html

55. Refactor or Rot: Building a self-improving codebase — Jacob Prall (Render)

Render's Jacob Prall describes an outer-loop refactoring workflow that uses heat, coupling, and health signals and a gated agent in isolated sandboxes to propose only refactorings whose payback justifies their cost.

Syllabus — core concepts:

- better code = cheaper to change
- heat/coupling/health signals for refactoring ROI
- deterministic gate separating decision from work
- 16 smell cards and 35-refactoring catalog
- deterministic replay to verify behavior unchanged

Primary source [1-notes]: Refactor or Rot: Designing a Self-Improving Codebase - DEV Community **Secondary:** https://jacobknowledgeaddict.substack.com/p/code-is-clay-behavior-is-stone

56. One Rule, One Throwaway Skill: Agent Migrations with ast-grep — Justin Poehnelt (Helen's Foundry)

Slides from Poehnelt's talk on using a single ast-grep rule in CI to block old patterns and a temporary skill to teach agents to migrate TypeScript code to Effect.

Syllabus — core concepts:

- ast-grep structural rules
- throwaway agent skill
- TypeScript to Effect migration
- CI blocking old patterns
- agent-driven code migration

Primary source [1-notes]: One Rule, One Throwaway Skill | Justin Poehnelt **Secondary:** https://ast-grep.github.io/catalog/python/migrate-openai-sdk.html

57. Code review without reviewing code — Daniel Tao (Atlassian)

Atlassian's Rovo Dev Code Reviewer automates the routine first pass of review across 1,900+ repositories with a three-stage LLM pipeline, cutting median PR cycle time by 30.8% while keeping humans as final decision-makers.

Syllabus — core concepts:

- AI first-pass code review with Rovo Dev
- LLM-as-judge filtering for factual correctness
- Actionability filtering with fine-tuned ModernBERT
- Human-in-the-loop review
- PR cycle time and review burden reduction

Primary source [3-canonical-topic]: Atlassian: AI-Driven Code Review Agent Reduces PR Cycle Time by 30.8% - ZenML LLMOps Database **Secondary:** https://www.atlassian.com/agile/software-development/code-reviews

58. Warning: This Pull Request Contains Graphic Content — Stephen Chin + Cassandra Chin (Neo4j)

AI-written PRs lie about tests passing because session logs compress; Tracetoon plus the Relate evidence graph assembles a verifiable provenance brief before writing, improving PR Bench scores while using about half the tokens.

Syllabus — core concepts:

- PR provenance and evidence graphs
- Tracetoon/Tracetune rebuilding PRs from agent traces
- Relate embedded graph database in Rust
- Verifying test-pass claims against commands and timing
- PR Bench quality vs token cost

Primary source [1-notes]: Your PR Says the Tests Passed. Prove It. - Stephen Chin & Cassandra Chin **Secondary:** https://www.startuphub.ai/ai-news/artificial-intelligence/2026/stop-trusting-prs-that-claim-tests-passed

59. The Software Factory That Needs Zero FDEs — Raz Berman Gold (Overcut)

Raz Berman Gold's Overcut talk argues an enterprise software factory needs zero FDEs when work is split into small specialized agents (triage, root-cause, implementation, review, verification) orchestrated from ticket to merged PR.

Syllabus — core concepts:

- Software factory without Forward Deployed Engineers
- Specialized small agents vs agent monoliths
- Ticket-to-merged-PR automation
- Orchestration with human approval gates
- Self-improving workflows via retrospectives and memory

Primary source [2-speaker-blog/paper/repo]: Evolve to AI **Secondary:** https://docs.overcut.ai/docs/welcome

60. Leaving Localhost: Building an Agent-Native Dev Environment — Sajid Mehmood (Niteshift)

Niteshift, quoted via CEO Sajid Mehmood, runs any frontier coding agent in a full-stack cloud VM environment with Docker, 32GB+ machines and memory-snapshot previews so agents can develop, test and ship without a local laptop.

Syllabus — core concepts:

- Cloud dev environments replacing localhost
- VM-based infrastructure for full Docker compatibility
- Memory snapshots for instant live previews
- Model-agnostic agents (Claude Code, Codex, OpenCode, Cursor)
- Verification with full stack, services and databases running

Primary source [2-speaker-blog/paper/repo]: How Niteshift Runs Full-Stack Coding Agents on E2B | E2B **Secondary:** https://github.com/prime-radiant-inc/alltheagents.org/blob/HEAD/agents/niteshift.md

Module 9 — Swarms, Workforce and the Organisation

Prerequisite: Module 8. Organisation is the last technical problem: agent workforce infrastructure, company-on-swarms, shared assistants beyond MCP, multiplayer factories, agents as culture and GTM. This fails without every earlier layer — which is why it is ninth, not first.

Module build: Draft the governance charter for a small agent workforce: roles, budgets, reversibility, escalation.

61. What It Takes to Run a Company on Agent Swarms — Karan Vaidya (Composio)

Karan Vaidya argues running many agents at company scale requires building the infrastructure coding already has — centralization, history, context, verification, governance and reversibility — so agents can be supervised and trusted beyond coding.

Syllabus — core concepts:

- human attention bottleneck and supervisor agents
- six primitives: centralization, history, context, verification, governance, reversibility
- governance walls outside the prompt/agent
- sandbox verification before irreversible actions
- Composio infrastructure for knowledge-work agents

Primary source [2-speaker-blog/paper/repo]: From coding to Knowledge work agents — Karan Vaidya, Composio **Secondary:** https://av.codes/blog/on-agent-swarms/

62. The Rise of the Agent Workforce: Rethinking Agent Infrastructure — Ankit Mathur (Databricks)

Ankit Mathur argues proliferating coding agents create sprawl, security and cost risks, requiring a gateway layer for observability, cost controls, privacy and unified MCP authentication.

Syllabus — core concepts:

- agent sprawl
- agent gateway layer
- observability and cost controls
- unified authentication for MCP tools
- governance of coding agents at enterprise scale

Primary source [2-speaker-blog/paper/repo]: Agentic + AI Coding Night SF · Luma **Secondary:** https://mfaits.github.io/blog/2026/03/05/conference-recap-coding-agents-ai-driven-dev-conference.html

63. Scaling Your Agent Workforce: Lessons from Inside Cognition — Theodor Marcu (Cognition)

Cognition's Devin Desktop, presented by Theodor Marcu, is positioned as a single command centre for coordinating, monitoring and directing local, cloud and third-party agents across an organisation.

Syllabus — core concepts:

- managing a fleet of agents
- agent command centre / Devin Desktop
- Spaces and shared context across agents
- local vs cloud agents
- Agent Client Protocol (ACP) and agent routing

Primary source [3-canonical-topic]: Cognition Launches Devin Desktop as Agent Command Centre for Engineering Teams - techcoffeehouse.com **Secondary:** https://completeaitraining.com/news/cognition-launches-devin-desktop-to-coordinate-ai-agents/

64. Our Agents Shipped 6,000 PRs to 600+ Companies — Alex Lebedev (PostHog)

PostHog's self-driving system uses scout agents that turn product/CI signals into pull requests — explaining how PR volume scaled past 6,500 a month and how those agents are measured and kept trustworthy.

Syllabus — core concepts:

- self-driving PRs from scheduled scout agents
- signals from deterministic detectors (flaky checks, broken main, duration regressions)
- Engineering Analytics: PRs as users, CI data queried via MCP
- scaling agent PR volume — 1,354 to 6,500+ PRs/month toward 10,000
- benchmarking/evaluating scouts and review agents (ReviewHog) at scale

Primary source [2-speaker-blog/paper/repo]: Engineers are shipping faster, so we made our CI self-driving to keep up - PostHog **Secondary:** https://posthog.com/blog/how-to-benchmark-ai-agents

65. Building a Shared AI Assistant: The Platform Challenges Beyond MCP — Utkarsh Shrivastava (Netflix)

Netflix governs one shared assistant fed by many teams' MCP servers by certifying connectors, benchmarking tool routing on real employee queries, and classifying which agent actions need human approval.

Syllabus — core concepts:

- shared AI assistant governance
- MCP connector certification
- tool routing benchmarking against real employee queries
- overlapping MCP servers causing wrong-tool selection
- human-approval classification for agent actions

Primary source [1-notes]: AI Engineer Code Summit 2026 Day 1: From the Agent Loop to Agent Swarms — Apper AI Solutions **Secondary:** https://ai.engineer/code/2026/schedule

66. Multiplayer software factories — Patrick Debois (Tessl)

Patrick Debois maps what breaks when a team's coding agents all ship to the same main branch at once, and the coordination patterns — ticket claims, file announcements, and speculative merge trains — that make a software factory multiplayer.

Syllabus — core concepts:

- multiplayer software factory
- concurrent coding agents shipping to one main branch
- ticket claims and file announcements
- speculative merge trains
- intent clashes vs line conflicts

Primary source [1-notes]: Your Team's Agents Are All Shipping to the Same Code at Once - Patrick Debois, Tessl **Secondary:** https://www.startuphub.ai/ai-news/artificial-intelligence/2026/tessl-s-concurrent-coding-pitch-needs-proof

67. A Company Built Around the Agent — Quinn Slack (Amp)

Quinn Slack and Thorsten Ball explain how Amp operates as a 21-person company built around its agent — no departments, one weekly meeting, and shipping to main without mandatory code review.

Syllabus — core concepts:

- 21-person agent-native company with no departments or quarterly goals
- Single Monday meeting and silent goal-writing
- Pushing to main without mandatory code review, agents reviewing changes
- Orbs, feature flags and WIP channels for coordination
- Revenue per headcount over scaling headcount

Primary source [2-speaker-blog/paper/repo]: The Company of the Future — Raising an Agent - Amp **Secondary:** https://finance.biggo.com/news/e8f3ded59e1b89c3

68. How to build an agent that builds the culture — Zach Bruggeman (Ramp)

A deep dive with Inspect founding engineer Zach Bruggeman on how Ramp's public, collaborative, self-verifying background agent was built in-house and adopted organically across the engineering culture.

Syllabus — core concepts:

- Ramp Inspect background coding agent
- Public multiplayer sessions with no opt-outs as culture

- Agents given same tools and context as engineers
- Self-verification with tests, telemetry and screenshots
- Organic adoption to 75% of merged PRs and 200+ internal agents

Primary source [3-canonical-topic]: Why Ramp built its own in-house coding agent, Inspect **Secondary:** https://inspect.ramp.engineering/

69. Building World-class Agentic GTM — Mihail Eric (Monaco)

Mihail Eric explains how Monaco built its own agentic harness for GTM — powering TAM/ICP scoring, pipeline and pre-meeting workflows — and pairs it with a world-class GTM team acting as forward-deployed account executives.

Syllabus — core concepts:

- Monaco's bespoke agentic harness for sales/GTM
- TAM/ICP scoring and pipeline evolution agents
- Long- and short-horizon GTM agents
- Forward-deployed account executives (FDAEs)
- Codifying world-class GTM team expertise in the platform

Primary source [2-speaker-blog/paper/repo]: How Stanford Teaches AI-Native Engineering — Mihail Eric, Monaco — Latent Space **Secondary:** https://www.mihaileric.com/

70. cloudflare is a neolab — Michelle Chen (Cloudflare)

Cloudflare argues training its own small System One models — Clef for decisions and Clairic for extraction, compute, estimation and forecasting — completes its agent cloud and makes it a neolab, cutting cost ~72% and running 1.9x faster than an all-LLM baseline.

Syllabus — core concepts:

- System One models
- Clef decision model
- Clairic extraction model
- Workers AI agent cloud
- Latency-driven small models vs LLMs

Primary source [1-notes]: AI Engineer Code Summit 2026 Day 2: Faster Inference, Smaller Models, and Code You Can Trust — Apper AI Solutions **Secondary:** https://blog.cloudflare.com/clairic-extraction-model/

Module 10 — The Human Ledger: Cost, Craft and Judgement

Prerequisites: all above (taken last, revisited always). First principle: the scarce resources at the end are human — money honestly counted, attention, craft, and judgement, the one job that does not delegate. The summit's most contrarian talks close the curriculum because they price everything before it.

Module build: Write your own cost/judgement audit: what you will automate, what you will never delegate, and why.

71. The True Cost of AI Coding — Scott Tolinski (Syntax)

Scott Tolinski presents survey data and expert commentary on AI coding's mental-health costs — burnout, hard-to-put-down prompting, and rising pressure despite time saved — and what separates energizing from unhealthy use.

Syllabus — core concepts:

- AI coding and mental health
- burnout and compulsive prompting
- time saved vs increased pressure
- enjoyment, skill and identity
- energizing vs unhealthy AI use

Primary source [2-speaker-blog/paper/repo]: The True Cost of AI Coding - YouTube **Secondary:** https://pawelgrzybek.com/top-picks-2026-august/

72. Forget AGI. You're Surrounded by Low-Hanging Fruit — Lucas Meijer (AgentsInTheCloud)

Meijer argues to ignore AGI hype and harvest simple repetitive-work automations with coding agents — custom Pi workflows, tight context management, and evaluation artifacts that let agents prove their work.

Syllabus — core concepts:

- practical agent workflows over AGI hype
- low-hanging fruit: automating boring repetitive work
- context management for coding agents
- evaluation packs and agents proving their work
- making codebases agent-ready

Primary source [2-speaker-blog/paper/repo]: A love letter to Pi | Lucas Meijer **Secondary:** https://www.andreslee.com/blog/making-your-codebase-ai-agent-ready

73. Your Automation Is Overthinking It — Chibi Vikram + Kevin Bernstein (UiPath)

Vikram and Bernstein show UiPath using fast System One Jev calls in a tournament pattern to rank document chunks in under a second with thousands of calls, rather than over-relying on slow reasoning LLMs for automation.

Syllabus — core concepts:

- System One models (Jev) vs reasoning LLMs
- Tournament ranking of document chunks
- Jev as an adjustable building block
- Composing patterns for intelligence instead of overthinking
- Fast structured decisions in under a second

Primary source [1-notes]: The Jev Revolution: What Engineers Are Already Building with System One Models **Secondary:** https://dev.to/uipath/everything-is-an-agent-now-and-they-are-arguing-with-me-5fpa

74. Manage Yourself Out of the Loop — Erik Thorelli (CodeRabbit)

CodeRabbit's official summit page confirms Erik Thorelli's November 11 expo talk on managing yourself out of the loop in agentic coding workflows.

Syllabus — core concepts:

- human-in-the-loop vs out-of-the-loop
- agentic coding workflows
- automating code review oversight
- developer experience at CodeRabbit

Primary source [1-notes]: Find CodeRabbit at AIE Code Summit | November 10 to 12, San Francisco **Secondary:** https://ai.engineer/code/2026/schedule

75. All Code Is Slop (Except Mine) — Lauren Tan (SpaceXSI)

Lauren Tan argues all agent code is slop until trust is built bottom-up from a clean codebase through static analysis and verification, letting her team land ~10,000 PRs a month without compromising quality.

Syllabus — core concepts:

- low-trust vs high-trust agent loops
- trust pyramid: codebase, static analysis, verification, prose, human review
- code is memory — codebase patterns are repeated by agents
- runtime, formal (TLA+/Lean) and property-based verification
- Grok Bot as outer loop around coding agents

Primary source [1-notes]: AI Engineer Code Summit 2026 Day 2: Faster Inference, Smaller Models, and Code You Can Trust — Apper AI Solutions **Secondary:** https://ai.engineer/code/2026/schedule

76. Judgement is the Last Job — Ishaan Dey (Google DeepMind)

Ishaan Dey's Shadow explores background coding agents with graduated autonomy, leaving humans to judge, steer and approve work that agents execute in parallel.

Syllabus — core concepts:

- human judgement and taste as bottleneck
- generator-discriminator gap
- background coding agents
- graduated autonomy and oversight
- human attention scarcity

Primary source [2-speaker-blog/paper/repo]: Shadow | Ishaan Dey **Secondary:** https://www.apper.cloud/blog/ai-engineer-code-summit-2026-day-3-morning-sessions

77. What it takes to set intelligence free — Fabian Hedin (Lovable)

Hedin argues setting intelligence free requires infrastructure around the model: an ACP runtime with an external append-only task history, instantly recreatable agent computers from golden checkpoints, and inner and outer verification loops.

Syllabus — core concepts:

- Agent control plane (ACP) orchestrator
- Append-only task history: forkable, replayable, outside the process
- Golden checkpoint: recreate agent computer in ~230ms
- Inner verification loop (in-turn self-correction)
- Outer verification loop (product/business metrics)

Primary source [1-notes]: AI Engineer Code Summit 2026 Day 2 Afternoon: Automating with Calibrated AI and Running Agents Reliably — Apper AI Solutions **Secondary:** https://www.latent.space/p/lovable-future-of-saas

78. Let's Open Source the World — Brandon Thomas (ArtCraft)

Brandon Thomas presented ArtCraft as a manifesto for using AI agents to rebuild proprietary creative software as 100% open source — apps, infrastructure and protocols — to take back the internet.

Syllabus — core concepts:

- AI-built open-source clones of Adobe apps in Rust (PhotoCraft, PdfCraft, etc.)
- Clean-room rebuilding and feature parity / muscle memory
- MCP and CLI-compatible agentic interfaces
- Open apps, infrastructure, protocols and models vs Web 2.0 gatekeeping
- AutoCraft autonomous software factory and human vs agent work

Primary source [1-notes]: AI Ambition: ArtCraft Cloned Photoshop and Now Wants to "Rebuild Everything" — Latent Space **Secondary:** https://www.ardanlabs.com/news/2026/ai-cloned-adobes-software-what-does-that-mean-for-the-future-of-software-development/

79. Thariq Shihipar talk — Thariq Shihipar ()

Detailed notes on Thariq Shihipar's AI Engineer talk 'A Field Guide to Fable', covering unhobbling Claude, finding your unknowns, grief, and being unreasonable with new Mythos-class models.

Syllabus — core concepts:

- Unhobbling Claude / capability overhang
- Finding your unknowns (unknowns matrix, six prompting moves)
- Context rightsizing: removing 80% of Claude Code's system prompt
- Dealing with the grief of losing hand-written code
- Being unreasonable: tradeoffs are not real

Primary source [1-notes]: Field Guide to Fable (Thariq Shihipar, Anthropic) **Secondary:** http://simonwillison.net/2026/Jul/21/cat-and-thariq/

The one gap

Something to Celebrate — swyx (AI Engineer), closing slot. No notes, transcript, slides, speaker post, or topic-specific resource could be verified for this 10-minute closing talk; the Apper and Latent Space roundups do not cover it. It is placed by inference only — a closing celebration belongs after M10 — and carries no syllabus until a recording surfaces.

Coverage: 79/80 sessions with verified sources. Machine-readable source table: `sources.json` alongside this file in the working folder.