tripleten

AI Systems Engineering

22-week accelerator

Online · Part-time

For tech professionals with production experience

Syllabus

2026

tripleten

How this program works

Run production, not toy apps.

You join a platform team and work on one production-shaped system across all five projects: diagnose it, evolve its architecture, take it through production operations and chaos testing, harden its security, and finish by building and defending an autonomous AI system on top.

5 projects → one production system

~25% theory / ~75% hands-on

~20 hours a week · on your schedule

~2 hours of live sessions weekly

22 weeks total

Project

The unit of the program. You get a realistic brief, a provided production-shaped codebase or dataset, milestones, and a deliverable you can defend.

Theory topics

Short lessons attached to each project — around 15 minutes each. Experienced students move faster, others take more time: project hours are fixed, theory time flexes.

Hours

Every project shows theory hours, project hours, and the total. Totals assume the recommended pace of 20 hours per week.

Who this program is for

A quick self-check before you commit — honest in both directions.

You're a good fit if

- You have hands-on production experience — as an engineer or in an adjacent technical role
- You can find your way around an unfamiliar codebase and pick up new tools as you go
- You want to grow into higher-leverage engineering work without quitting your job — the program is part-time, on your own schedule

This is probably not for you if

- You're starting from zero — no coding experience yet. TripleTen's core bootcamps are the right entry point; this program assumes a working base.
- You're after a certificate rather than a portfolio of defended work.

Program summary

At the recommended pace of 20 hours per week. Project hours are fixed; theory time flexes with your experience, so the total may vary by a week or two.

5.5	16.5	22
weeks of theory ~109.75 hours	weeks of project work ~330 hours	weeks total

THE 5 PROJECTS · HOURS = THEORY + PROJECT WORK, TOTAL PER PROJECT

1	System Diagnostics and Scaling Review	~67 h
2	Service Architecture and the Polyglot Data Tier	~88.5 h
3	Production Operations, Resilience, and the Chaos Lab	~96 h
4	Zero-Trust Security, Compliance, and a Grounded AI Feature	~82.75 h
5	Autonomous AI System, Built and Defended	~105.5 h

Each project is detailed on the following pages.

Project 1

System Diagnostics and Scaling Review

Theory	~15 h
Project work	~52 h
Total	**~67 h**

Students join a platform team and diagnose a production-shaped reference platform that must survive 10x growth. The system is already running locally with an API, worker, PostgreSQL, Redis, dashboards, traces, logs, and a synthetic load generator.

THEORY TOPICS

- System architecture basics, request paths, state locations, and C4 diagrams
- Monolith vs. microservices as a diagnostic baseline
- Scale reasoning, latency percentiles, throughput, queues, and bottlenecks
- Observability first principles, metrics, dashboards, tracing, and metric cardinality
- LLM fundamentals for reviewing AI-generated engineering analysis
- ADR writing and evidence-based technical recommendations

SKILLS YOU'LL ADD

System architecture · Observability · Prometheus & Grafana · OpenTelemetry & Jaeger · PostgreSQL · Redis · Docker Compose

LEARNING OBJECTIVES

- → Explain how the reference platform works at runtime.
- → Compare monolith and microservice trade-offs from observed behavior, not assumptions.
- → Interpret golden-signal dashboards and traces under load.
- → Repair unsafe or incomplete observability instrumentation.

Project 2

Service Architecture and the Polyglot Data Tier

Theory	~24.5 h
Project work	~64 h
Total	**~88.5 h**

Students evolve the reference platform into a clearer service architecture and production-style data tier. The starter repo is opinionated, but the design choices still matter: REST behavior, one gRPC service boundary, data access, idempotency, caching, read freshness, and migration safety.

THEORY TOPICS

- REST design, API versioning, and API evolution
- gRPC, protobufs, service boundaries, and DDD
- Idempotency keys and duplicate side-effect prevention
- AI-native development as a code-review and implementation accelerator
- Relational production patterns, migrations, repositories, N+1 detection, and transaction boundaries
- Data replication, read replicas, Redis caching, cache invalidation, and freshness
- GraphQL, NoSQL families, partitioning, sharding, use cases, and architecture proposals as decision-level topics

SKILLS YOU'LL ADD

REST API design · gRPC & Protobuf · Service boundaries (DDD) · Idempotency · Migrations & data tier · Redis caching

LEARNING OBJECTIVES

- → Design APIs that remain stable as product requirements change.
- → Use idempotency keys to protect distributed requests from duplicate side effects.
- → Define service boundaries that reduce coupling instead of spreading complexity.
- → Implement a data layer that supports production-style reads, writes, migrations, and performance checks.

Project 3

Production Operations, Resilience, and the Chaos Lab

Theory	~24 h
Project work	~72 h
Total	**~96 h**

Students take the platform into a production-like local environment. It must run in local Kubernetes, pass CI gates, expose useful telemetry, apply one OpenTofu/LocalStack resource, survive one failure lab, and show one concrete resilience pattern.

THEORY TOPICS

- Docker, Kubernetes core concepts, and managed Kubernetes positioning
- OpenTofu/Terraform fundamentals, LocalStack, AWS guardrails, and cloud topology
- CI/CD pipelines, deployment gates, rollback, and teardown
- GitOps, Argo CD reconciliation, drift detection, and pull-request-based operations as optional or positioning topics
- Alerting, SLOs, observability, and failure-lab practice
- CAP, consistency models, partial-failure patterns, and backpressure
- Messaging patterns, Redpanda/Kafka basics, transactional outbox, idempotent consumers, DLQs, and recovery

SKILLS YOU'LL ADD

Docker · Kubernetes · OpenTofu/Terraform · LocalStack · CI/CD · Chaos testing · Kafka/Redpanda

LEARNING OBJECTIVES

- → Deploy a multi-service platform into a local production-like environment.
- → Explain how infrastructure, deployment, scaling, and topology choices affect reliability.
- → Use OpenTofu and LocalStack without creating uncontrolled cloud spend.
- → Explain managed EKS and GitOps at positioning level without requiring full implementation.

Project 4

Zero-Trust Security, Compliance, and a Grounded AI Feature

Theory	~20.75 h
Project work	~62 h
Total	**~82.75 h**

Students harden one core platform workflow and add a narrow grounded AI feature for a security-sensitive client scenario. The platform needs threat modeling, access control, secrets handling, a CI security gate, redaction, audit logging, and simulated control evidence.

THEORY TOPICS

- Threat modeling, STRIDE, zero-trust architecture, and trust boundaries
- Identity, access control, OIDC, RBAC, scopes, and least privilege
- Secrets management, Kubernetes Secrets, OpenBao/Vault concepts, and safe runtime configuration
- Pipeline security, dependency scanning, secret scanning, and deployment gates
- Compliance by design, simulated audit evidence, redaction, BAA boundaries, and safeguards
- EU AI Act classification, transparency, human oversight, and control mapping
- AI governance, grounded AI risks, security gaps, and technical communication

SKILLS YOU'LL ADD

Threat modeling (STRIDE) · Zero-trust · OIDC & RBAC · Secrets management · CI security gates · Compliance mapping

LEARNING OBJECTIVES

- Identify realistic threats across users, services, data stores, pipelines, and AI features.
- Implement identity, access, and secrets controls without production identity-cluster operations.
- Add one security check that fails unsafe changes early.
- Protect paid LLM API calls with redaction, logging, and cost/request limits.

Project 5

Autonomous AI System, Built and Defended

Theory	~25.5 h
Project work	~80 h
Total	**~105.5 h**

Students receive a production AI problem, not a demo brief: build an autonomous AI system that can be evaluated, bounded, observed, and defended. The required path uses paid LLM APIs through a provider adapter, PostgreSQL with pgvector, keyword search, one bounded LangGraph flow, and CI-safe eval fixtures.

THEORY TOPICS

- Production definition of done, build kickoff, and final architecture review
- Production LLM integration, provider adapters, API reliability, rate limits, and cost controls
- LLM model serving with vLLM, batching, GPU/API economics, routing, and cost/latency trade-offs
- RAG as production architecture, `pgvector`, keyword retrieval, reranking as optional depth, and grounding
- Prompt injection mitigation, agentic frameworks, state graphs, tool sandboxing, and human oversight
- Evaluation pipelines, LLM-as-judge calibration, 5-10 CI smoke gates, cached fixtures, and failure coverage
- AI observability, cost telemetry, latency telemetry, portfolio readiness, and hostile defense preparation

SKILLS YOU'LL ADD

LLM API integration · RAG with pgvector · LangGraph · Prompt-injection defense · LLM evaluation · Cost & latency controls

LEARNING OBJECTIVES

- Build an end-to-end AI system that satisfies production acceptance criteria.
- Compare hosted LLM APIs, Groq, Bedrock-style managed access, Ollama, and vLLM at decision level.
- Implement RAG that balances retrieval quality, latency, cost, and safety.
- Prevent prompt injection and unsafe tool execution with concrete controls.

Who will teach you

The program is designed and reviewed by working practitioners — with public GitHub and LinkedIn, so you can verify them yourself.

Program author

Gonen Eren

Computer-vision PhD with 20+ years engineering fault-tolerant distributed systems and high-throughput data pipelines on AWS/Azure. Designed this program and the review bar your projects are held to.

linkedin.com/in/goneneren

Instructor

Mohamed Cherif

Principal-level engineer with 20+ years building GPU-accelerated media pipelines, SDKs, and computer vision systems at AMD, Microsoft, Amazon, and Skype.

AMD | Microsoft | Amazon | Skype

github.com/mcherif · linkedin.com/in/mohamed-cherif-25b8b24

More experts are joining the team against the same bar: working practitioners with production experience in the program's domain.

How every project works

How each project gets reviewed — and what your week looks like along the way.

Reviews on every project

Readiness Review. When your project feels done, you book a short 15–30 min check with your instructor to confirm you're ready.

Final Review. You present and defend the project live on the nearest review day — they run every two weeks. The panel grows with the project, up to a Tier-1 engineer on the capstone.

Your weekly rhythm

A 30-minute 1:1 with an instructor every week — optional, bring anything

Your project group chat with tutors — always on

You move in project cohorts — grouped by the project you're on, not by start week

Review days run every two weeks — you book the nearest one when your project is ready

Group formats run evenings and weekends

Schedule example

At the guideline pace of 20 hours per week. The career track runs alongside your projects — its milestones are the highlighted rows.

WEEKS	STAGE	FINAL REVIEW
Before start	We get on a call and build your personal study plan together	
Weeks 1–3	Project 1 · System Diagnostics and Scaling Review	~week 3
~Week 3	**Career** · Your Career Coach joins — resume audit and target roles	
Weeks 4–8	Project 2 · Service Architecture and the Polyglot Data Tier	~week 8
Weeks 8–13	Project 3 · Production Operations, Resilience, and the Chaos Lab	~week 13
~Week 13	**Career** · Resume repositioned, reviewed projects become portfolio cases	
Weeks 13–17	Project 4 · Zero-Trust Security, Compliance, and a Grounded AI Feature	~week 17
Weeks 13–22	**Career** · Mock interviews — unlimited AI practice, live technical and HR	
Weeks 17–22	Project 5 · Autonomous AI System, Built and Defended	~week 22 · Tier-1 engineer
Graduation	**Career** · Final package and a structured search pipeline — support continues	

Theory time flexes with your experience, and pace is individual: faster and slower paths exist — treat week numbers as guides, not deadlines.

How we support you

You study on your own schedule, but you're never on your own: three layers of support run through the whole program.

Teaching

- An optional 30-minute 1:1 with an instructor every week — bring your questions, your code, or your architecture
- Fast answers in your project chat whenever you get stuck
- Every project reviewed live by a working engineer

Community

- A project group chat — everyone in it is building the same thing you are
- Senior-level engineers from different companies around you, not a beginners room
- Weekly live group sessions and events with instructors
- Networking and random coffee pairings to meet peers one-on-one
- A student directory — find peers by background and reach out directly
- On graduation you join TripleTen's alumni network — 7,500+ graduates worldwide

Onboarding

- A personal manager who has your back on anything organizational
- Keeps you on track with your deadlines and reviews
- Helps you build and adjust your individual study plan

Career support

Built for engineers with existing experience: we reposition the career capital you already have and get you fully ready for the interview room. Your Career Coach joins after your first Final Review and stays through the program.

1

Positioning & materials

- → Career capital audit: resume and LinkedIn reviewed against your real experience
- → Target-role hypothesis and a personal transition plan
- → Gap analysis against live vacancies for your target role

2

Portfolio from defended work

- → Every defended project becomes a portfolio case study
- → Resume and LinkedIn repositioned for the target role — not rebuilt from scratch

3

Mock interviews & debriefs

- → Unlimited AI interview practice
- → Live technical and HR mocks with clearly stated limits
- → Debriefs of real interview rounds as you complete them

✓

To a signed offer

- → Final package: resume, LinkedIn, portfolio, target company list
- → A structured search pipeline instead of mass applications
- → Support continues past graduation: debriefs and strategy iteration on your search data

Why this beats self-paced courses

Tutorials and course libraries teach pieces. This program makes you prove the whole thing — under review, on a system that behaves like production.

Every project is defended live to a working engineer. Free courses never make you prove it — here the review is the unit of progress.

Weekly 1:1s with an instructor and an always-on project chat. Real feedback when you are stuck, not a comment section under a video.

One production-shaped system built across the whole program. Architecture decisions and trade-offs, not isolated exercises and snippets.

A career track with real people runs alongside your projects. Coach, mock interviews, real-round debriefs — tutorials end where this starts.

Frequently asked questions

The questions engineers actually ask us before enrolling — answered straight.

Is this live classes or self-paced?

Neither, and that's the point. You study on your own schedule — no mandatory class times — but this is not a self-paced course library: every week there's an optional 30-minute 1:1 with an instructor and live group sessions, and every project ends with a live review in front of a working engineer, every two weeks.

Are real humans teaching, or is it videos plus AI?

It's not a prerecorded video library. Theory is short lessons attached to each project; the teaching layer is human — instructors in 1:1s and your project chat, and every project is defended live to a working engineer.

How much instructor time do I actually get?

Every week you can book a 30-minute 1:1 with an instructor — bring your questions, code, or architecture. Your project chat with tutors is always on, and each project ends with a Readiness Review (15–30 min) plus a live Final Review.

How do project reviews work — live or async?

Live. When your project feels done, you book a Readiness Review; then you present and defend the project on the nearest review day — they run every two weeks. The panel grows with the project, up to a Tier-1 engineer on the capstone.

Will I study alone, or is there a real cohort?

You move in project cohorts, grouped by the project you're on. Each has a group chat where everyone is building the same thing, weekly live group sessions and events with instructors, a student directory to find peers, and random coffee pairings — working engineers around you, not a beginners room.

I already know parts of this stack. What's the point?

The program isn't a content library — it's a review bar. You make architecture decisions on production-shaped briefs and defend them live to engineers who do this work in production. Familiar tools mean you move faster: project hours are fixed, theory time flexes.

What if I can't give it 20 hours a week?

20 hours is the guideline pace, not a rule. Fewer hours means a longer timeline — week numbers in this syllabus are guides, not deadlines.

Do you guarantee me a job?

No — and no honest program can. What we commit to is the career track that runs alongside your projects: a personal coach, a portfolio built from defended work, mock interviews and real-round debriefs, and a structured search pipeline that continues past graduation.