Faysal Aziz

Low-Code AI-Driven SOC Automation

with n8n, OpenAI, and Low-Code Agent Workflows - 2 days

FA-0293 · 2 days

Software Development · Cybersecurity · Automation, RPA & Power Platform

---

Introduction

Orchestrate SOC intelligence and automation with n8n, AI agents, and practical low-code security workflows.

Security operations teams do not need to become full software development shops to benefit from AI. What they do need is a practical way to connect systems, enrich data, structure decisions, and automate repetitive operational work without building everything from scratch. That is where a low-code platform becomes valuable.

This course is designed for SOC engineers and security operations teams that want to move faster using AI, but in a way that fits real-world operations: workflows, integrations, approvals, human review, alert processing, enrichment, summarization, documentation, and repeatable response patterns. The primary environment in this version of the training is n8n, which allows participants to build operational automations visually while still supporting API calls, AI agent behavior, memory, tool usage, and selective custom code when needed. The course uses OpenAI API for the hands-on agent labs and also covers on-prem model patterns using Qwen for organizations that need tighter control over data and execution.

The instructor brings over 30 years of industry experience and teaches from operational reality, with emphasis on industry-demanded implementation patterns rather than academic examples. n8n's current platform supports AI Agent workflows, tool calling, memory management, webhooks, HTTP requests, and code steps, which makes it well suited to low-code SOC automation labs.

Learning outcomes

By the end of this training, participants will be able to:

- Understand where low-code AI orchestration fits within SOC engineering and security operations
- Use n8n to build visual workflows for common SOC automation and investigation support tasks
- Connect n8n workflows to OpenAI API for summarization, classification, enrichment, and agent-style task execution
- Build AI-powered workflows using webhooks, HTTP requests, logic nodes, and human review checkpoints
- Understand how to use n8n's AI Agent capabilities for tool-based reasoning and controlled multi-step operations
- Create reusable low-code operational patterns for alert handling, log review, evidence transformation, and documentation
- Use low-code and selective custom code together, keeping Python or code usage minimal and purposeful
- Understand how memory, tools, and workflow state affect AI-powered automation design
- Design low-code agent workflows that can also support on-prem model usage with Qwen
- Apply security, governance, and operational controls to AI-enabled SOC workflows

Prerequisites

- Familiarity with SOC operations, alerts, incident handling, and investigation workflows
- Basic understanding of APIs, JSON, and web requests
- Some exposure to automation logic, even if not through formal programming
- Comfort with structured thinking and process design
- Basic awareness of security tools, SIEM platforms, or enrichment services
- No strong programming background required
- Prior exposure to Python is helpful but not mandatory

Training outline

1. Foundations of Low-Code AI for SOC Operations

1. Why low-code matters for security operations teams
2. The difference between traditional scripting-heavy automation and workflow-driven automation
3. The role of AI in SOC engineering without turning the team into software developers
4. Moving from manual triage and enrichment to orchestrated decision support
5. Where low-code fits in alert handling, investigation support, documentation, and escalation workflows
6. Strengths and limitations of visual workflow automation in security operations

7. Understanding the boundaries between low-code, no-code, and code-assisted automation
8. The operational value of combining workflow logic with AI reasoning
9. Human oversight as a permanent requirement in security workflows
10. Hosted AI versus self-hosted AI deployment models in enterprise environments

2. Introduction to n8n for Security Operations

1. What n8n is and why it is suitable for low-code SOC automation
2. Understanding workflows, nodes, triggers, and data movement
3. The n8n execution model
4. Visual workflow construction for operational teams
5. Working with input data, item-based processing, and node outputs

   Core node categories relevant to SOC workflows

   1. Trigger nodes
   2. Transformation nodes
   3. Logic and routing nodes
   4. HTTP integration nodes
   5. AI-related nodes
   6. Code nodes
7. How n8n supports AI workflows, agents, tools, and memory
8. Understanding where n8n reduces coding effort and where small code inserts still help
9. Designing SOC workflows in a maintainable visual form
10. Building automations that remain understandable to operations teams over time

3. n8n as the Primary Low-Code Workbench

1. Navigating the n8n interface
2. Creating and organizing workflows
3. Managing credentials and connections
4. Handling workflow variables and expressions
5. Using expressions to transform values without writing full code
6. Working with JSON inside n8n
7. Mapping fields between systems
8. Using test runs versus production runs
9. Debugging workflow execution paths
10. Inspecting node input and output for troubleshooting

11. Structuring projects for operational reuse
12. Separating experimental workflows from production-ready workflows
13. Building workflow patterns that can be reused across SOC use cases

4. Core Low-Code Building Blocks for SOC Automation

1. Trigger-based automation patterns
2. Webhook-driven workflows for inbound SOC events
3. Scheduled workflows for recurring analysis and housekeeping
4. HTTP Request node usage for external integrations
5. Decision logic using conditions, switches, and branching
6. Data transformation without heavy scripting
7. Formatting payloads for downstream systems
8. Building simple orchestration flows from multiple operational steps
9. Managing retries, delays, and wait states
10. Handling errors and fallback paths
11. Creating responses from workflow outputs
12. Building workflow endpoints that behave like lightweight internal services
13. Designing reliable low-code flows around inconsistent input data
14. Avoiding overly fragile visual workflows

5. AI Integration with OpenAI API in n8n

1. Connecting n8n workflows to OpenAI-powered capabilities
2. Structuring AI calls inside workflows
3. Sending alert, log, and incident data to AI models
4. Receiving structured outputs for downstream automation
5. Designing prompts for operational tasks
6. Separating instructions from raw operational input
7. Using AI for summarization, classification, normalization, and explanation
8. Creating predictable output formats for downstream nodes
9. Handling model responses safely and consistently
10. Managing tokens, response size, latency, and cost
11. Designing reusable AI steps inside broader SOC workflows
12. Embedding AI into low-code flows without making them opaque
13. Balancing workflow simplicity with model usefulness

6. Prompt Design for Low-Code SOC Workflows

1. Writing prompts that fit low-code operational pipelines
2. Structuring instructions for repeatable output quality
3. Prompting for alert explanation
4. Prompting for log interpretation
5. Prompting for enrichment summaries
6. Prompting for analyst handoff notes
7. Prompting for documentation generation
8. Prompting for evidence extraction
9. Prompting for remediation draft support
10. Controlling verbosity and output style
11. Requesting structured JSON responses for downstream automation
12. Reducing hallucination through tight task framing
13. Grounding AI on provided data only
14. Building reusable prompt blocks for common SOC tasks
15. Maintaining prompt libraries for operational reuse

7. AI Agents in n8n for SOC Engineering

1. What an AI agent means in a workflow-based environment
2. Understanding n8n's AI Agent model
3. The current AI Agent approach in n8n as a tools-based pattern
4. Designing agents around access to approved tools and operational context

   Agent roles in SOC use cases

   1. Alert triage assistant
   2. Log analysis assistant
   3. Documentation assistant
   4. Enrichment assistant
   5. Escalation preparation assistant
6. Building narrow-purpose agents instead of broad uncontrolled assistants
7. Supplying the agent with tools, memory, and task instructions
8. Creating structured input and output boundaries
9. Keeping agent steps transparent and inspectable
10. Deciding when to use an agent versus a fixed workflow sequence
11. Recognizing when a standard workflow is better than agentic behavior

12. Managing uncertainty and escalation in agent-driven flows
13. Human-in-the-loop checkpoints for agent tools and sensitive steps

8. Tool-Using Agents and Controlled Orchestration

1. Understanding tools in the n8n AI Agent model
2. Connecting tools to agents for task execution
3. Using HTTP Request as a tool
4. Using internal workflow actions as tools
5. Using custom tool nodes where needed
6. Using agent tools for enrichment and lookup
7. Using agent tools for routing and decision support
8. Understanding schema-driven tool calling
9. Making tool outputs useful for downstream steps
10. Human approval for gated actions
11. Limiting tool access based on sensitivity
12. Designing safe operational toolsets for SOC teams
13. Avoiding unrestricted or overly powerful tool patterns
14. Building trust through constrained agent action design
15. Using sub-agents and agent-as-tool patterns for layered workflows

9. Memory and Context Management in Workflow-Based Agents

1. Why memory matters in AI-assisted security workflows
2. Short-lived versus persistent context
3. Tracking conversation or investigation state
4. Passing state between steps in an n8n workflow
5. Managing what should be remembered and what should be discarded
6. Memory for analyst-assistant style workflows
7. Memory for multi-step investigation support
8. Context injection for repeated operational tasks
9. Avoiding memory bloat and irrelevant carryover
10. Keeping memory aligned with workflow purpose
11. Using memory only where it improves operational outcomes
12. Designing explicit state rather than relying on opaque behavior

13. Operational considerations for storing contextual traces

10. Low-Code SOC Use Cases in n8n

1. Alert intake and pre-processing workflow
2. Alert summarization and normalization workflow
3. Log parsing and interpretation workflow
4. Indicator enrichment workflow
5. Evidence transformation workflow
6. Incident note drafting workflow
7. Analyst handoff workflow
8. Escalation summary generation workflow
9. Query assistance workflow
10. Parser troubleshooting workflow
11. Detection review support workflow
12. Internal knowledge entry generation workflow
13. Runbook support workflow
14. Triage support workflow
15. Repetitive operational task acceleration workflow
16. Building reusable workflow blueprints for recurring SOC needs

11. Slightly Low-Code, Not No-Code: Where Light Coding Still Matters

1. Why some operational workflows still need selective code
2. Using n8n’s Code node sparingly and intentionally
3. Choosing between expressions and code
4. Light transformations that are easier in code than visually
5. Small formatting or parsing logic for security data
6. Handling edge cases that low-code nodes do not solve cleanly
7. Deciding when a short code step improves maintainability
8. Avoiding unnecessary custom code
9. Keeping code inserts small, readable, and replaceable
10. Using light code for data cleanup and response shaping
11. Using minimal code to preserve low-code ownership by operations teams

12. Webhooks, APIs, and Event-Driven Security Automation

1. Using webhooks to ingest security events
2. Building event-driven workflows from external tools
3. Triggering workflows from SIEM, SOAR, case management, or custom alert sources
4. Returning processed results through webhook responses
5. Designing n8n flows as lightweight internal service endpoints
6. Using HTTP Request to call enrichment APIs and internal services
7. Structuring request and response payloads
8. Handling authentication and credentials securely
9. Designing reliable retry behavior for external dependencies
10. Managing timeouts, failures, and partial data conditions
11. Chaining event-driven steps without overcomplicating the workflow

13. Agent Skills Reframed for a Low-Code Audience

1. Understanding “skills” as reusable operational capabilities
2. Translating the idea of an agent skill into workflow modules and reusable AI tasks
3. Defining a skill around one job, one purpose, and one expected output style

   Skill-like building blocks in n8n

   1. Reusable sub-workflows
   2. Reusable prompt blocks
   3. Reusable tool configurations
   4. Reusable transformation patterns
5. Packaging repeatable SOC tasks into reusable workflow units

   Designing low-code skill equivalents for

   1. Log summarization
   2. Evidence normalization
   3. Alert classification support
   4. Investigation note drafting
   5. Escalation summary preparation
7. Keeping skills narrow and composable
8. Avoiding overloading one workflow component with too many responsibilities
9. Testing and validating reusable low-code skills
10. Maintaining a library of trusted operational workflow components

14. Human Review, Approvals, and Controlled Execution

1. Why human-in-the-loop is essential in SOC AI workflows
2. Using approval checkpoints before actions
3. Pausing workflows for review
4. Reviewing agent tool use before execution
5. Approving escalations, outbound notifications, or sensitive automation steps
6. Separating analysis support from action-taking steps
7. Designing review interfaces and approval flows
8. Reducing the risk of incorrect automated behavior
9. Preserving accountability in low-code AI operations
10. Building trust gradually through supervised execution

15. On-Prem and Hybrid AI with Qwen

1. Why some SOC teams require on-prem model options
2. Privacy, data handling, and enterprise control considerations
3. Designing low-code workflows that can call internal model endpoints
4. Using Qwen as part of a self-hosted or internal AI strategy
5. Hosted model versus on-prem model trade-offs
6. Latency, cost, control, and deployment considerations
7. Using n8n as the orchestration layer regardless of where the model runs
8. Designing hybrid workflows that route some tasks to hosted models and others to internal models
9. Choosing appropriate security tasks for local model execution
10. Operational limitations and expected tuning considerations for on-prem model usage
11. Workflow portability across hosted and internal AI backends

16. Governance, Security, and Operational Standards

1. Data handling rules for AI-enabled workflows
2. Protecting secrets, credentials, and internal context
3. Logging and auditability in low-code automations
4. Workflow ownership and change management
5. Versioning workflow logic and AI prompts
6. Approval standards for production deployment
7. Defining what tasks AI may support and what tasks remain human-owned
8. Reviewing workflow changes for operational risk

9. Documenting workflow behavior for support and continuity
10. Retention considerations for prompts, inputs, outputs, and traces
11. Governance differences between hosted and on-prem AI usage
12. Building standards for reusable workflow modules and AI components

17. Testing, Validation, and Operational Reliability

1. Testing low-code workflows before production use
2. Validating prompt outputs against expected operational needs
3. Checking structured data consistency across nodes
4. Building safe fallback paths
5. Verifying agent tool selection and action boundaries
6. Handling malformed input data
7. Evaluating workflow usefulness and false-confidence risk
8. Monitoring workflow performance over time
9. Measuring operational benefit
10. Identifying brittle logic before it becomes production risk
11. Building iterative improvement loops for workflow quality
12. Moving from prototype to production in a controlled way

18. AI Workflow Builder and Accelerated Development

1. Using natural language to help generate workflows
2. Understanding where AI-assisted workflow generation can speed up prototyping
3. Reviewing generated workflows rather than trusting them blindly
4. Refining AI-generated workflow structure into operationally sound designs
5. Using AI assistance to reduce setup effort without losing control
6. Converting rough ideas into workflow drafts

Disclaimer

The outlines shared are indicative only and are provided for guidance purposes. They do not constitute a fixed or exhaustive agenda. The trainer expressly reserves the right, at his sole discretion and without prior notice, to modify, adapt, reorganize, substitute, add, or remove any module, topic, sub-topic, tool, exercise, demonstration, or delivery method based on participant feedback, participant capability, class progress, technical environment, operational requirements, and any other factors that may affect the effectiveness or relevance of the training. Such adjustments are considered part of the normal training process and are made to ensure the best possible learning outcome for the audience.

© 2026 Faysal Aziz · Kuala Lumpur