

HANDS-ON WORKSHOP

Enterprise Agentic AI

Architecting Autonomous Java Systems for Production

Two Parts • 12 Exercises



Part 1 — Foundations and Enterprise Control

90 minutes • Exercises 1–8



Part 2 — From Patterns to Production

80 minutes • Exercises 9–12



QUARKUS



LangChain4j
by QUARKUS



GOVERNED

Policies, guardrails & human approval



ADAPTIVE

Planning, memory & consensus



OBSERVABLE

OpenTelemetry traces & metrics



EVENT-DRIVEN

Kafka processing & remote agents



The Problem: Apex Systems NOC

Manual triage is failing



P1 outage misclassified as P3 → **4+ hours**,
\$150k lost



Payment incident routed to wrong team →
20 engineer-hours wasted



No audit trail →
compliance gap



Too many tickets. Too little context. Too much at stake.
It's time to move from manual to intelligent.



Why **Agentic AI**, Not a Chatbot

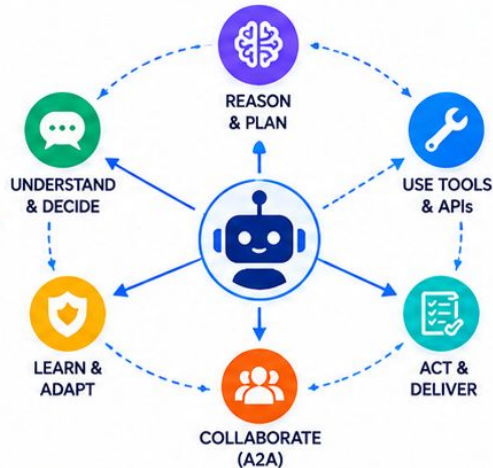
Chatbot



- ❓ Answers questions
- ⚡ One LLM call
- 🚫 No side effects
- 🗄️ Stateless
- 🧠 Single model

VS

AI Agent



- ✓ Takes actions
- ✓ Multi-step reasoning + tool calls
- ✓ Mutates state
- ✓ Shares context via AgenticScope
- ✓ Composed specialists

Agentic capabilities you will build



REASON

Understand context, plan next steps, and solve complex problems.



ACT

Use tools and APIs, execute actions, and update systems.



COLLABORATE

Work with other agents and humans across systems (A2A).



PAUSE FOR HUMANS

Seek approval, confirm decisions, and stay in control.



SCALE

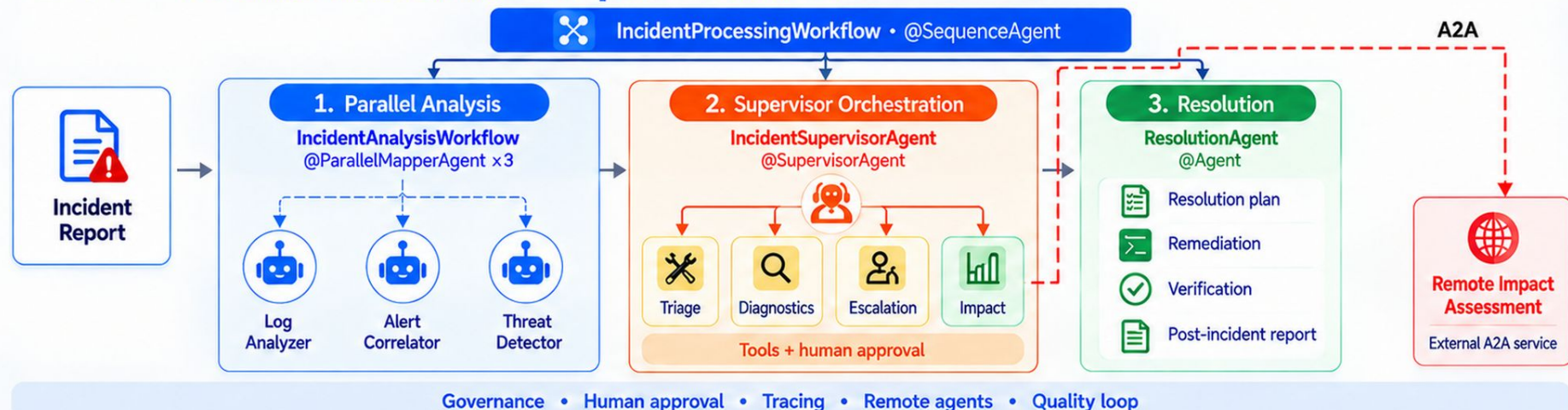
Scale out agents and workloads with Kubernetes and event-driven architectures.

What You Will Build

An enterprise incident management platform — from foundations to production

Part 1 — Foundations and Enterprise Control

Exercises 1–8



Part 2 — From Patterns to Production

Exercises 9–12



Outcome: Build, run, and govern agentic systems you can trust in production.

Part 1 — Foundations and Enterprise Control

Your Learning Path — 8 Exercises



Goal: Build, run, and govern agentic systems you can trust in production.

1



Agent + Tool



Sam,
NOC analyst

Build your first agent with a tool. Analyze an incident using real data.

2



Policy as Prompt



Chris,
Ops lead

Encode routing policies and guardrails as prompts and test their impact.

3



Parallel Agents



Chris,
Ops lead

Run multiple specialized agents in parallel to speed up analysis.

4



Supervisor Orchestration



Priya,
IT service mgr

Use a supervisor agent to orchestrate specialists and make decisions.

5



AI Governance



Jordan,
Platform engineer

Apply governance controls, model allow-lists, and content filters.

6



Human Gate + Tracing



Alex,
Compliance officer

Add human approval checkpoints and trace every step for auditability.

7



Remote Agents A2A



Riley,
SRE lead

Integrate external agents via A2A protocol for impact assessment.

8



Quality Loop (bonus)



Jordan,
Platform engineer

Add feedback loops to continuously improve agent performance.



Exercises 1–4: code-along

Build core agents and workflows together.



Exercise 5: AI governance

Learn how to govern agent behavior safely.



Exercises 6–7: run & explore

Add controls, integrate remote agents, and observe.



Exercise 8: bonus

Improve quality with continuous feedback loops.

Part 2 — From Patterns to Production

Your Learning Path — 6 Advanced Exercises | 75 minutes hands-on • 90-minute session



Goal: Build evidence-based, stateful, event-driven agent systems.

9

Structured Planning

15 min



Validate specialist plans, bound review cycles, and make recommendation readiness explicit.

Plan → Execute → Review

10

Required MCP Evidence

10 min



Require a tool lookup before planning. Fail closed when evidence is unavailable.

Evidence before planning

11

Memory Boundaries

10 min



Isolate conversations and observe eviction from a PostgreSQL-backed memory window.

Conversation isolation + eviction

12

Voting and Disagreement

15 min



Run specialist ballots in parallel. Apply a deterministic majority-or-review policy.

Parallel ballots → Majority or review

13

Event-Driven Processing

10 min



Trigger a sequential agent workflow with Kafka and correlate processing outcomes.

Kafka → Workflow → Outcome

14

Persistent Outcomes and Replay

15 min



Reuse completed work after hot reload. Inspect durable, duplicate-safe outcomes.

Persist → Replay → Reuse

One shared project • One Quarkus dev-mode terminal • Exercises 9–14

Hot reload between exercises • Switch dashboard modes

Part 3 — Reliable Response and Evaluation

Your Learning Path — 6 Advanced Exercises | 75 minutes hands-on • 90-minute session



Goal: Guide, approve, recover, and evaluate incident responses.

1

Guide & Validate



15

Runtime Skills

10 min



Load reusable incident-response guidance at runtime.

16

Runtime Output Guardrails

10 min



Enforce the response contract with bounded repairs and exact evidence checks.

Guidance → Validated recommendation

2

Approve & Recover



17

Asynchronous Approval

15 min



Persist the proposal; record the human decision in a later request.

18

Restart Recovery

15 min



Preserve pending approvals across a JVM restart. Retry interrupted planning explicitly.

Saved proposal → Durable decision

3

Test & Evaluate



19

Deterministic Regression Tests

10 min



Verify skills, repairs, evidence, and state with a local scripted model.

20

Quality Evaluation

15 min



Compare baseline and candidate on fixed cases using checks, judge reasons, and latency.

Repeatable tests → Measured quality

One shared project • One incident-response journey • Exercises 15–20

Hot reload for edits • One deliberate JVM restart in Exercise 18

The Technology Stack

A modern, open, and enterprise-ready foundation
for building agentic AI systems in Java.



How They Connect

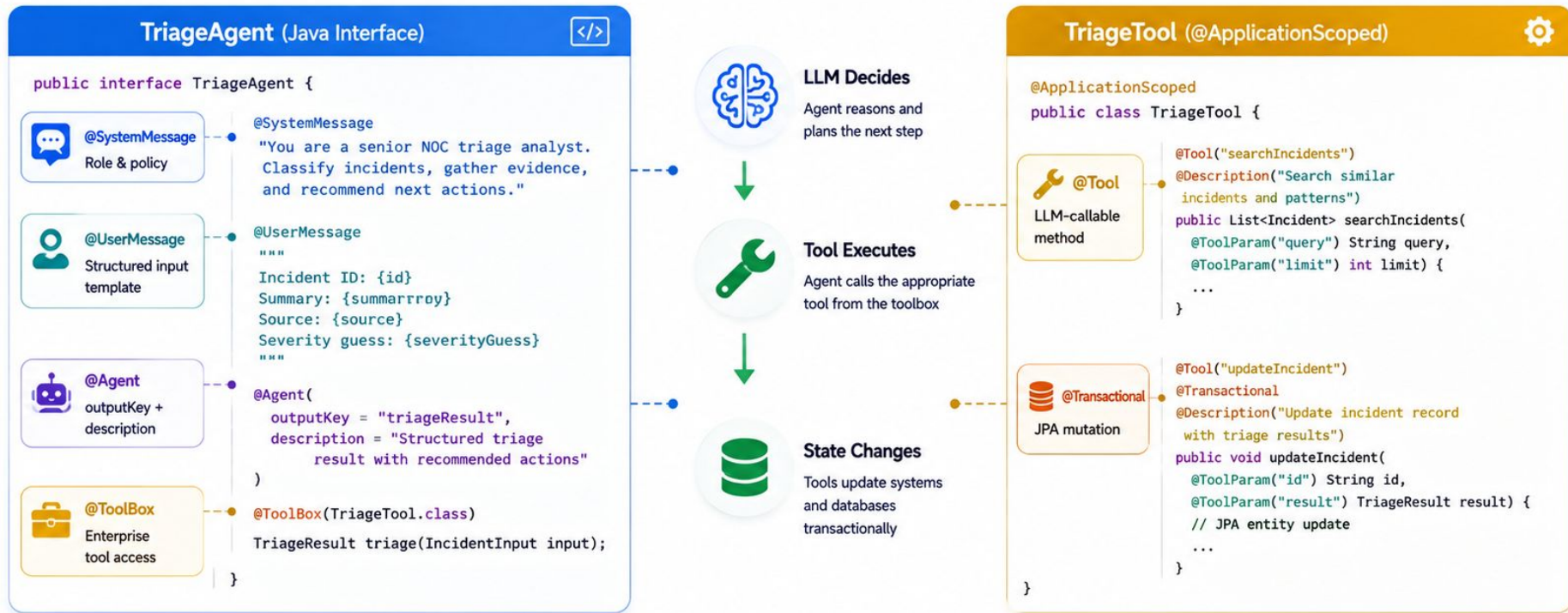


Built for Production

- Performance**
Fast startup, low memory footprint, and high throughput.
- Security**
Secure by design with policy enforcement and human oversight.
- Observability**
End-to-end tracing, metrics, and cost visibility.
- Autonomy**
Composed agents that reason, act, and adapt at scale.

The Declarative @Agent Model

You declare WHAT the agent does — Quarkus generates HOW it works.



You declare WHAT — Quarkus generates HOW.

No classes. No boilerplate. Just interfaces, annotations, and tools.



Fast to Build



Type Safe



Maintainable



Production Ready



Agent Interface



Tool Class



Execution Flow

--- Invocation Flow

Enterprise Controls in Action

Part 1 — Foundations and Enterprise Control



AI Governance

Exercise 5

When agents need boundaries

Project guidance + runtime controls



AGENTS.md

Project instructions guide the coding assistant.



Use approved APIs



Verify before changing code



Explain decisions



Runtime controls

Model allow-lists and content filters.



Separate project guidance from controls applied at runtime.

Guidance + explicit controls



Human Approval

Exercise 6

When an escalation needs approval

Illustrative approval checkpoint



Human Approval Required

P1 escalation

EscalationAgent requests approval to notify the Payments Team.

Reason: customer impact requires review.

✗ Reject

✓ Approve



Request



Human gate



Decision



Pause at a named checkpoint for an approval decision.

Approve or reject before execution



Observability

Exercise 6

When you need to explain a run

Illustrative trace view

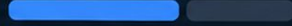
Incident execution trace



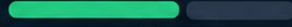
Agent call



Tool invocation



Human approval



Token usage • Execution status



Inspect agent calls, tool execution, and token usage.

OpenTelemetry traces



Controls around agent execution



Policies guide behavior



Approval gates control execution



Traces explain what happened

Production Patterns in Action

Part 2 — From Patterns to Production



Plan & Execute

Exercise 9

When analysis misses evidence

Illustrative example



Initial plan: inspect logs



Review: dependency evidence missing



Revised plan: add diagnostics



Review identifies a gap; the agent revises its plan.

Plan → Execute → Review



Memory Tiering

Exercise 10

When an investigation resumes

Illustrative example



Incident A: prior context restored



Incident B: separate conversation



Resume with stored context while keeping incidents isolated.

Bounded window + PostgreSQL



Consensus & Voting

Exercise 11

When specialists disagree

Illustrative example



Reliability: recommends rollback



Customer impact: recommends rollback



Cost / risk: recommends monitor

Illustrative ballots



Compare structured ballots and apply a deterministic tally.

Parallel voters + explicit decision rule



Event-Driven Agents

Exercise 12 • Capstone

When alerts keep arriving

Illustrative example



Kafka accepts alerts



Agents process asynchronously



Results recorded



Accept new alerts while agent processing continues.

Kafka + asynchronous processing



Production behavior beyond the first agent call



Revise the plan



Retain incident context



Compare specialist judgments



Process alerts asynchronously

Reliable Response in Action

Part 3 — Reliable Response and Evaluation



Skills & Guardrails

Exercises 15–16

When guidance must be enforced

Illustrative example



Runtime skill supplies response guidance



Invalid action triggers bounded repair



Exact report excerpts checked



Reusable guidance plus executable response checks.

Guide → Validate → Repair



Async Approval

Exercise 17

When a human decides later

Illustrative example



HTTP 202 returns immediately



Proposal saved:
WAITING_APPROVAL



Later decision is persisted



Record approval without a new model call.

Queue → Propose → Decide



Restart Recovery

Exercise 18

When the application restarts

Illustrative example



Pending proposal stored in PostgreSQL



JVM stops and starts again



Same proposal hash and attempt count



Pending approval survives; interrupted planning needs explicit retry.

Persist → Restart → Recover



Tests & Evaluation

Exercises 19–20

When a change needs evidence

Illustrative example



Scripted model checks regressions



Fixed cases: baseline vs. candidate



Inspect checks, judge reasons, latency



Test behavior repeatedly and assess quality from evidence.

Test → Compare → Decide



Reliable behavior beyond a plausible answer



Enforce response checks



Save human decisions



Recover pending approvals



Measure response quality

Let's Build! bit.ly/agents-labs

Spin up the over-engineered helpdesk server and explore the demo.



>_ Quick Start

```
# 1. Clone the repository
$ git clone https://github.com/danieloh30/agentc-ai-java-workshop.git
$ cd agentc-ai-java-workshop

# 2. Start the helpdesk server
$ ./mvnw clean quarkus:dev
[INFO] Quarkus started in 2.341s. Listening on: http://localhost:8080

# 3. Open the dashboard
$ open http://localhost:8080
```



This app is intentionally over-engineered with MCP.
Let's see why simplicity wins.



Incident Dashboard

Monitor and manage helpdesk incidents

8

Total Incidents
Across all priorities

2

P1 - Critical
Needs immediate
attention



3

P2 - High
High priority
incidents



2

P3 - Medium
Requires
attention



1

P4 - Low
Low priority
incidents



ID	TITLE	PRIORITY	STATUS	ASSIGNEE	UPDATED
INC-54821	Payment service degraded	P1 Critical	OPEN	Priya S.	2m ago
INC-54813	Checkout timeouts	P1 Critical	OPEN	Alex R.	7m ago
INC-54805	Search API slow responses	P2 High	IN PROGRESS	Jordan K.	15m ago
INC-54799	Email notifications failing	P2 High	OPEN	Riley M.	28m ago
INC-54788	User profile not updating	P2 High	IN PROGRESS	Chris L.	45m ago
INC-54770	Report generation slow	P3 Medium	OPEN	Sam D.	1h ago
INC-54766	Password reset delay	P3 Medium	IN PROGRESS	Taylor W.	2h ago
INC-54740	UI alignment on mobile	P4 Low	OPEN	Jamie P.	3h ago



Quarkus
Lightning fast
Java runtime



LangChain4j
Agentic workflows
made simple



MCP SDK
Model Context
Protocol



OpenAI
Powering intelligent
decisions



Open **localhost:8080** — Exercise 1 starts now.

List all tickets and tell me which one sounds most urgent.

Get the full source code

github.com/danieloh30/agentc-ai-java-workshop.git

