Faysal Aziz

AI-Powered Modern Development

Developer workflows, polyglot services and security review

FA-0575 · 2 days

Software Development · Agentic & Generative AI · DevOps, Cloud & Infrastructure · Cybersecurity

Explore AI-assisted coding and a small service prototype, with model integration, messaging, deployment choices and security checks.

Introduction

This two-day workshop is for developers with application and API experience in Python, Go or Java. It combines AI-assisted coding and prompt/context design with a conceptual review of model lifecycle, polyglot services, messaging and secure architecture.

Hands-on work uses one familiar language and a prepared small service/queue example. Other languages, ML infrastructure and scaling/orchestration frameworks are comparisons or clinics. The outcome is a reviewed prototype and design understanding, not mastery of three languages or a production-secure microservice platform.

Learning outcomes

The workshop teaches participants to:

- Test prompts/context for selected coding, review, refactoring, documentation and test tasks.
- Compare remote APIs, local inference and hybrid model integration.
- Explain model lifecycle, serving, features and feedback/evaluation responsibilities.
- Describe polyglot services, message queues, retries and idempotence.
- Review deployment, orchestration and observability trade-offs for AI components.
- Identify AI-specific and service-security risks and appropriate controls.
- Build and critique a small illustrative AI-assisted prototype.

Prerequisites

- Be comfortable programming in at least one of: Python, Go, Java.
- Have experience building and deploying non-trivial applications (i.e. some architecture exposure, microservices exposure helpful but not strictly required).
- Basic familiarity with ML/AI concepts (what a model, dataset, training vs inference are).
- Some knowledge of networked systems / APIs, message queues (not deep but basic).
- Willingness to experiment and work in teams / breakout sessions.

A laptop, prepared development environment and authorised model/tool access. Confirm lab dependencies and usage costs; bring competence in one of the listed languages rather than expect language instruction.

Training outline

1. Day 1 – Prompting, coding assistance and model integration

1. Prompt and context design

- What is a prompt, what are its components (system / user / role / examples)

  Selected model-appropriate prompting patterns and their limitations

  - Zero-shot, One-shot, Few-shot prompting
  - Task decomposition and structured rationale where useful; compare reasoning approaches without demanding hidden chain-of-thought or assuming one universal recipe.
  - Compare multiple outputs when useful; agreement is not proof and adds cost.
  - Meta-prompting (prompts that generate or refine other prompts)
  - Context window design: priming, retrieval-augmented generation (RAG), prompt chaining
  - Adversarial prompting, prompt injection risks and mitigations

  Tools, platforms, frameworks for prompt engineering

  - Selected current LangChain/LlamaIndex interfaces for the prepared example.
  - Local/cloud/hybrid models and embeddings; distinguish prompting from model adaptation and its additional data/resource requirements.
- Workshop in a familiar language: draft/refactor/document/test a small illustrative task, then review and run checks.

2. AI-assisted development workflows

- Use cases: code generation, code review, debugging assistance, pair programming with AI, auto completion, refactoring, documentation, translating code between languages
- Integrating LLMs via APIs vs hosting local models (ONNX, small LMs)
- Architectures for embedding AI in your dev loop (CI/CD, pre-commit, linting, auto-PRs)
- Evaluate output and review corrections; feedback does not automatically train or update a model.

3. ML lifecycle and application components

- ML model lifecycle: Data collection, preprocessing, training, validation, deployment, monitoring
- Feature stores, model serving, versioning, drift detection
- Light ML vs heavy ML: embedding-based features, classification/regression, anomaly detection, recommendation, etc.
- Compare selected compatible model-serving and pipeline tools for the chosen stack, not implementation of every framework in two days.

2. Day 2 — Services, messaging, safeguards and prototype

4. Polyglot services and queues

- Polyglot services: language interoperability and shared interfaces; distinguish implementation languages from multilingual model capabilities.
- Message queues: Kafka, RabbitMQ, etc. Use cases for ML / AI pipelines (event sourcing, asynchronous processing, streaming inference)
- Patterns: producer/consumer, event buses, stream processing, microservice interactions over queues
- Ensuring consistency, dealing with failures, scalability, idempotence

5. AI microservice architecture

- Review of microservice architecture style: API gateways, service meshes, domain boundaries, bounded contexts
- Patterns for AI microservices: model serving services, feature store services, data pipelines, inference vs training separation
- Compare container, Kubernetes, serverless, batch and streaming deployment choices at an architecture level.
- Workflow/pipeline orchestration options and their operational requirements; selected demonstration.
- Observability: logging, metrics, tracing especially in AI usage (latency, mispredictions, drift)

6. Security and responsible deployment

- Secure by design principles (least privilege, defense in depth, zero trust)
- Specific security risks with AI/LLMs: prompt injection; data leakage; privacy; model poisoning; malicious input; misuse
- Microservices security best practices: authentication & authorization per service; secure communication (TLS, mTLS, service mesh); API gateways; rate limiting; secret management; network segmentation
- Secure ML deployment: validation of inputs, adversarial robustness, securing model artifacts, versioning, monitoring for drift or malicious behavior
- Data-protection and applicable regulatory questions, including GDPR where relevant, for appropriate context-specific review.

7. Prototype and review

- Using AI for ideation: brainstorming, prototyping, creativity sprints
- Team prototype: a small prepared AI/service/queue workflow with selected security checks; not a production-secure deployment claim.
- Illustrative architecture cases and trade-offs, not unverified company success claims or stale 2024–2025 rankings.
- Encouraging experimentation: patterns for small side projects, internal tools, etc.

© 2026 Faysal Aziz · Kuala Lumpur