Faysal Aziz

Building Robust AI Backends

Model-serving APIs, data workflows and engineering controls

FA-0593 · 2 days

Software Development · DevOps, Cloud & Infrastructure · Data & Analytics

Prototype an AI inference endpoint and review data, performance, security and deployment decisions for a larger service.

Introduction

This two-day advanced workshop helps backend developers design services that integrate trained or hosted AI models. It connects API contracts and data preparation with model serving, performance, security, observability and operational review.

A selected inference endpoint and prepared pipeline or deployment example provide the practical focus. Kubernetes, alternative serving frameworks, large-scale data architectures and ethical assessment are comparative or review topics rather than separate complete implementations. The prototype remains subject to further security, model-quality and operational validation before production use.

Learning outcomes

The course teaches participants to:

- Design an inference API with appropriate request patterns, payload limits, errors and versioning.
- Integrate a selected model interface and package a small service in a compatible environment.
- Explain data ingestion, preprocessing, storage and traceability requirements.
- Measure selected performance characteristics and compare caching, batching and concurrency choices.
- Review authentication, authorisation, validation, scaling and failure-handling controls.
- Identify privacy, fairness, documentation and human-oversight questions for operational review.

Prerequisites

- Strong experience in backend development (any language/framework: e.g. Java, Python, Go, Node.js)
- Familiarity with REST / HTTP API design, asynchronous programming, concurrency basics
- Some exposure to AI/ML models (pre-trained models, basic understanding)
- Basic familiarity with containerization (Docker), cloud infrastructure, and ideally some experience with Kubernetes or microservices
- A prepared development/container environment and access to selected approved models or learning endpoints.
- Exercises use synthetic or appropriately approved data; deployment resources and costs are agreed separately.

Training outline

1. Day 1 — API, Model and Data Foundations

Module 1 — API Design

- Understanding use cases: inference APIs vs training / fine-tuning APIs
- Compare synchronous calls, queued background jobs, callbacks/webhooks, polling and streaming; implement a selected pattern.
- Handling large data payloads: chunking, streaming uploads/downloads, file storage vs inline data, memory limits
- Rate limits, quotas and backpressure to reduce abuse and overload risk.
- API versioning strategies: URL versioning, header versioning, backward compatibility, deprecation policies
- API documentation & contracts: OpenAPI / Swagger for AI endpoints, specifying error modes, result format, latency expectations, fallback behavior

Module 2 — Model Integration and Packaging

- Pre-trained models vs hosted vs custom models: trade-offs (cost, latency, control)

  Compare serving frameworks appropriate to the chosen model format and available support.

  - TensorFlow Serving as an example for compatible TensorFlow models.
  - TorchServe handler/archive concepts as legacy comparative coverage: the official project is no longer actively maintained and does not plan fixes or security patches; do not present it as the default for a new production service.

    - Compare an alternative such as Triton Inference Server, checking current model/backend and hardware requirements.
- Containerization with Docker: building reproducible images, handling dependencies, packaging model + server
- Kubernetes deployment, resources, replicas and rolling-update concepts through a prepared demonstration.
- Review CI/CD tests, versioned model artefacts and rollback requirements.

Module 3 — Data Pipelines

- Data ingestion: sources of data (logs, user inputs, sensors, etc.), batch vs streaming ingestion
- Data cleaning & preprocessing: missing values, outliers, normalization, feature extraction, tokenization for text, image preprocessing
- Handling structured vs unstructured data: relational / NoSQL databases, object storage, document stores, vector databases for embeddings
- Data storage optimized for AI workloads: formats (e.g. Parquet, Avro), storage systems, partitioning, indexing, retrieval efficiency
- Ensuring reproducibility and traceability in pipelines: data versioning, schema management, data lineage
- Build or inspect a small inference endpoint connected to a prepared model/data example.

2. Day 2 — Performance, Controls and Architecture Review

Module 4 — Performance and Observability

- Cache only where model/version, data freshness and user/tenant boundaries make reuse valid.
- Efficient data serialization: JSON vs binary formats (e.g. Protobuf, MessagePack), streaming serialization when large data
- Parallel and asynchronous processing: worker queues, batching requests, handling concurrency to avoid blocking
- Explore warm starts, preloading and supported accelerator use; measure effects rather than assume faster inference.
- Measure latency, throughput, errors and resources; discuss model/data drift separately from infrastructure metrics.

Module 5 — Security, Scaling and Failure Handling

- Authentication, authorisation and structured input/output validation; sanitisation alone does not eliminate injection or adversarial-input risk.

- Review rate limits and abuse controls without duplicating the API exercise.
- Scaling strategies: autoscaling (horizontal scaling), load balancing, scaling model servers separately from frontend APIs, using queues / message brokers to smooth spikes
- Microservices architecture vs serverless for AI inference: trade-offs, when to choose each, operational overheads
- API gateways: using gateways for routing, monitoring, authorization, throttling, versioning, and handling cross-cutting concerns
- Bounded retries, circuit breakers and graceful degradation appropriate to operation semantics; avoid duplicated side effects.

Module 6 — Responsible AI Engineering

- Data minimisation, encryption and relevant privacy/regulatory review questions; engineering controls do not themselves guarantee GDPR/CCPA compliance.
- Review dataset and outcome bias, selected fairness metrics and mitigation trade-offs; no universal fairness guarantee.
- Explainability appropriate to the model/task, documented strengths/limits and human oversight; avoid logging sensitive data unnecessarily.
- Auditing & accountability: keeping version history of models & data, tracking feature changes, test cases for ethical edge cases
- Fallback behaviour, alerts and user communication when model outputs or services fail.

Module 7 — Architecture and Release Review

- Microservices vs monolithic architectures for AI: deployment, ops complexity, isolation of AI components; separation of concerns
- Serverless inference/preprocessing concepts with cold-start, memory, duration and model-size constraints.
- API gateways and service meshes: traffic management, security, observability across multiple AI services
- Review versioned infrastructure/configuration and environment parity using a selected example.
- Testing/staging and shadow/canary release concepts, with approval and rollback decisions.
- Evaluate the selected prototype, record constraints and identify additional checks before deployment.
- Compare alternative patterns against actual workload and support requirements rather than assume a single architecture is always best.

© 2026 Faysal Aziz · Kuala Lumpur