Faysal Aziz

Azure AI Driven Backend

with Copilot & MCP in 2 days

FA-0114 · 2 days

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

---

Introduction

From prompt to protocol — building production-grade LLM systems with Python, Azure, and Copilot

---

Modern backend systems are evolving beyond deterministic APIs into intelligent infrastructures where language models act as reasoning layers. But raw model calls are not enough — context, retrieval, and tool orchestration are required to build trustworthy systems. This course prepares Python developers to leverage Copilot as more than a coding assistant, to design retrieval pipelines, and to implement MCP (Model Context Protocol) servers and clients that integrate seamlessly into enterprise environments.

The instructor brings over 30 years of industry experience, ensuring all topics are grounded in practical relevance — not academic abstraction. Attendees will leave with the skills to architect, implement, and deploy real-world intelligent backends that stand up to enterprise demands for reliability, scalability, and security.

Learning outcomes

Participants will be able to:

- Use Copilot productively for code generation, explanation, refactoring, and planning.
- Apply meta-prompting and structured task decomposition for more reliable outcomes.
- Understand retrieval-augmented generation (RAG) architectures and implement context-aware pipelines.
- Explain MCP concepts, architecture, and design principles.
- Build Python-based MCP servers and clients that integrate with databases, files, APIs, and streams.
- Combine RAG with MCP for efficient context handling and tool discovery.
- Deploy and secure MCP servers on Azure with authentication, monitoring, and observability.
- Apply enterprise practices for scaling, resilience, and governance in intelligent backends.

Prerequisites

- Expert level knowledge of Python (modules, async I/O, HTTP, data parsing).
- Familiarity with SQL and relational databases.
- Understanding of REST APIs, JSON, and HTTP basics.
- Experience with VS Code or equivalent IDE.
- Optional but useful: prior exposure to LLM APIs.

Training outline

1. 1. Intelligent Development with Copilot

- Positioning Copilot in the development lifecycle
- Modes of interaction: completion, inline code actions, chat
- Strategic usage patterns: scaffolding, refactoring, testing, documentation
- Limitations and error patterns of AI-assisted coding
- Advanced prompting: decomposition, meta-prompting, verification loops
- Applying Copilot for backend development tasks in Python and SQL

---

2. 2. Retrieval-Augmented Generation (RAG)

- Why context matters: limitations of prompt-only approaches
- Embeddings and vector search fundamentals
- Indexing strategies: chunking, hierarchical, adaptive
- Contextual retrieval and reranking techniques
- Memory systems: session memory, persistent memory, hybrid strategies
- Managing prompt size and mitigating prompt bloat
- Safety and governance in RAG pipelines
- Integrating RAG outputs into model workflows for improved reliability
- Tradeoffs: latency, scalability, cost, and data freshness

---

3. 3. Introducing the Model Context Protocol (MCP)

- Motivation: extending Copilot and agents with external tools
- MCP architecture and key concepts: tools, resources, prompts

- JSON-RPC communication model, sessions, and error reporting
- Discoverability and dynamic registration of tools
- Copilot ecosystem integration and interoperability with Azure services
- Comparison with ad-hoc API integrations

---

4. 4. Building MCP Servers and Clients in Python

- Designing tool surfaces: input/output schemas, versioning, error handling
- Implementation patterns: choosing frameworks, handling sessions, dispatch logic

  Connecting MCP servers to external systems:
  - Databases
  - File systems
  - REST APIs
  - Streaming endpoints
- Building MCP clients: session management, tool invocation, dynamic discovery
- Error handling, retries, logging, and resilience strategies
- Integrating RAG with MCP: retrieval-driven tool selection and context injection
- Tradeoffs in MCP system design: performance, cost, extensibility

---

5. 5. Enterprise Deployment on Azure

- Deployment targets: Container Apps, Functions, App Service, Kubernetes
- Authentication and authorization: Azure AD integration and RBAC
- Monitoring and observability: logging, telemetry, traces, alerts
- Reliability engineering: fault tolerance, retries, rate limiting
- Governance and compliance: secure tool exposure, permissions, sandboxing
- Versioning and lifecycle management of MCP servers
- Ensuring safety in production: prompt injection defenses, restricted tool execution

---

6. 6. Advanced Patterns and Architectural Considerations

- Multi-MCP orchestration and federated tool ecosystems
- Hybrid architectures: RAG + MCP interplay
- Tool caching, prefetching, and batching for performance
- Streaming and long-running tool operations

- Security challenges: adversarial misuse, chained exploits, data exfiltration
- Evaluation metrics: latency, accuracy, hallucination rate, tool effectiveness
- Migration of existing services to MCP-based infrastructure
- Future outlook: agent composability, memory-driven MCP integrations, evolving standards

---

© 2026 Faysal Aziz · Kuala Lumpur