AI-200 Course Summary

Course Overview

The **AI-200T00-A: Develop AI Cloud Solutions on Azure** course provides an in-depth, hands-on exploration of building, deploying, and managing robust AI-driven applications within the Azure ecosystem. Students develop expertise in container orchestration using **Azure Container Apps, App Service, and Azure Kubernetes Service (AKS)**, while mastering advanced data strategies like **vector similarity search and Retrieval-Augmented Generation (RAG)**. The curriculum emphasizes comprehensive security and observability, covering the protection of sensitive credentials through **Azure Key Vault** and the implementation of standardized monitoring with **OpenTelemetry**. Notably, this course is offered in two versions one utilizing **Bash** and the other **PowerShell** which contain identical technical content and learning objectives.

Detailed Lab Summaries

Lab 1: Build and Run a Container Image with ACR Tasks

- **Objective:** To learn how to use Azure Container Registry (ACR) Tasks to build and manage container images entirely in the cloud without a local Docker installation.
- **Key Topics Covered:** Azure Container Registry, ACR Tasks, image versioning, and production image locking.
- **Hands-On Activity:** Deploying an ACR instance, using `az acr build` to create images from source code, verifying image tags and manifests, and locking a specific image version to prevent accidental deletion.
- **Real-World Application:** Developers use ACR Tasks to automate cloud-based container builds, ensuring a consistent build environment and securing production-ready images.

Lab 2: Deploy a Container to Azure App Service

- **Objective:** To deploy a Linux container image from ACR to Azure App Service using secure managed identities.
- **Key Topics Covered:** Azure App Service for Containers, system-assigned managed identity, and RBAC (AcrPull role).
- **Hands-On Activity:** Configuring a Web App to use a managed identity, granting the `AcrPull` role for registry access, enabling persistent storage for logs, and testing a document processing API.
- **Real-World Application:** IT professionals host containerized web APIs in App Service to benefit from simplified scaling and secure, passwordless authentication to private registries.

Lab 3: Deploy a Containerized Backend API to Container Apps

- **Objective:** To deploy a containerized API to Azure Container Apps using managed identities for secure image pulls and secrets management.

- **Key Topics Covered:** Azure Container Apps, ingress configuration, secret management, and environment variable referencing.
- **Hands-On Activity:** Creating a Container Apps environment, deploying an API with system-assigned identity, configuring secrets as environment variables, and verifying deployment through health endpoints.
- **Real-World Application:** Engineers leverage Container Apps for serverless microservices that require secure access to AI model keys and automatic certificate management.

Lab 4: Diagnose and Fix a Failing Deployment

- **Objective:** To troubleshoot Azure Container Apps by diagnosing common issues like missing environment variables and ingress misconfigurations.
- **Key Topics Covered:** Troubleshooting workflows, revision status, target-port mapping, and Log Analytics.
- **Hands-On Activity:** Identifying a failing API revision, correcting port mismatches in YAML configurations, and querying historical console logs using Kusto Query Language (KQL).
- **Real-World Application:** Operations teams use these skills to minimize downtime by quickly isolating and fixing deployment errors in containerized AI solutions.

Lab 5: Configure Autoscaling for an API using KEDA

- **Objective:** To implement KEDA-based autoscaling in Azure Container Apps based on HTTP concurrency triggers.
- **Key Topics Covered:** KEDA (Kubernetes Event-driven Autoscaling), horizontal scaling, scaling to zero, and YAML-based scale rules.
- **Hands-On Activity:** Deploying a mock agent API, configuring an HTTP concurrency scale rule, generating load via a Flask dashboard, and observing real-time replica count changes.
- **Real-World Application:** Cloud architects use KEDA to ensure AI workloads scale out during peak inference demand and scale down to zero when idle to optimize costs.

Lab 6: Deploy an AI Inference API to Azure Kubernetes Service

- **Objective:** To deploy an AI inference API container to Azure Kubernetes Service (AKS) using standard Kubernetes manifests.
- **Key Topics Covered:** AKS cluster management, Kubernetes YAML (Deployments/Services), liveness probes, and resource limits.
- **Hands-On Activity:** Creating an AKS cluster, completing `deployment.yaml` and `service.yaml` manifests, deploying the API, and testing connectivity to a GPT-4 model.
- **Real-World Application:** Systems administrators use AKS to manage high-availability AI services that require precise control over container orchestration and hardware resources.

Lab 7: Configure Apps on Azure Kubernetes Service

- **Objective:** To configure Kubernetes deployments with persistent storage and secure management of sensitive and non-sensitive settings.
- **Key Topics Covered:** ConfigMaps, Kubernetes Secrets, PersistentVolumeClaims (PVC), and Azure Disk storage.

- **Hands-On Activity:** Creating ConfigMaps for app metadata, generating Secrets for sensitive keys, configuring PVCs for persistent logging, and mounting these resources into AKS pods.
- **Real-World Application:** Developers utilize these resources to ensure application state is preserved across restarts and sensitive credentials remain protected within the cluster.

Lab 8: Troubleshoot Apps on Azure Kubernetes Service

- **Objective:** To diagnose and resolve frequent Kubernetes errors including label mismatches, CrashLoopBackOff status, and readiness probe failures.
- **Key Topics Covered:** `kubectl` diagnostics, pod events, namespace management, and rolling update strategies.
- **Hands-On Activity:** Resolving Service selector mismatches, adding missing environment variables to fix container crashes, and correcting invalid health check paths in deployment manifests.
- **Real-World Application:** Support engineers use these diagnostic commands to maintain the health of production clusters and ensure successful application updates.

Lab 9: Build a RAG Document Store on Azure Cosmos DB for NoSQL

- **Objective:** To build a document storage backend optimized for Retrieval-Augmented Generation (RAG) using Azure Cosmos DB.
- **Key Topics Covered:** Cosmos DB for NoSQL, RAG patterns, document chunking, and metadata filtering.
- **Hands-On Activity:** Deploying a Cosmos DB account, building Python functions to store and query document chunks by metadata, and verifying retrieval results via a Flask application.
- **Real-World Application:** AI developers use Cosmos DB as a grounded data source to provide context to large language models, ensuring more accurate AI responses.

Lab 10: Build a Semantic Search Application with Azure Cosmos DB for NoSQL

- **Objective:** To implement vector similarity search in Azure Cosmos DB to enable semantic matching over support data.
- **Key Topics Covered:** Vector search, high-dimensional embeddings, the `VectorDistance` function, and vector indexing policies.
- **Hands-On Activity:** Configuring a container with vector embedding policies, loading support tickets with pre-computed vectors, and performing semantic searches using cosine distance.
- **Real-World Application:** Companies implement semantic search to find relevant information in knowledge bases even when user queries do not contain exact keyword matches.

Lab 11: Optimize Query Performance with Vector Indexes on Azure Cosmos DB for NoSQL

- **Objective:** To compare and tune different vector indexing strategies in Azure Cosmos DB to optimize performance and reduce costs.
- **Key Topics Covered:** Vector index types (Flat, QuantizedFlat, DiskANN), Request Unit (RU) consumption, and latency tuning.

- **Hands-On Activity:** Creating multiple containers with different index types, loading identical datasets, and benchmarking RU costs and execution times for various queries.
- **Real-World Application:** Database administrators optimize vector indexes to balance the trade-offs between search accuracy, speed, and operational expenses in large-scale AI apps.

Lab 12: Build an Agent Tool Backend on Azure Database for PostgreSQL

- **Objective:** To create a PostgreSQL database that serves as a persistent memory and tool backend for AI agents.
- **Key Topics Covered:** PostgreSQL Flexible Server, agent memory schema, Entra ID authentication, and JSONB for metadata.
- **Hands-On Activity:** Deploying a PostgreSQL server, designing tables for conversations and task checkpoints, and building Python tool functions to manage agent session state.
- **Real-World Application:** Developers use persistent relational stores to allow AI agents to maintain long-term memory across multiple user sessions and resume complex tasks.

Lab 13: Implement Vector Search on Azure Database for PostgreSQL

- **Objective:** To build a product similarity search application using Azure Database for PostgreSQL and the `pgvector` extension.
- **Key Topics Covered:** `pgvector` extension, HNSW indexes, cosine distance operators (`<=>`), and semantic search.
- **Hands-On Activity:** Enabling `pgvector`, creating a products table with a vector column, implementing an HNSW index, and testing similarity searches via a web interface.
- **Real-World Application:** E-commerce platforms use vector search in PostgreSQL to recommend semantically similar products based on descriptions rather than just categories.

Lab 14: Optimize Vector Search Performance in Azure Database for PostgreSQL

- **Objective:** To optimize PostgreSQL for vector workloads by comparing and tuning IVFFlat and HNSW index strategies.
- **Key Topics Covered:** IVFFlat vs. HNSW, index parameters (probes, `ef_search`), and hybrid metadata filtering.
- **Hands-On Activity:** Generating 100,000 test vectors, measuring sequential scan performance, building IVFFlat and HNSW indexes, and tuning search parameters to balance speed and recall.
- **Real-World Application:** Performance engineers tune vector search parameters to achieve sub-second latency on large-scale datasets while maintaining high accuracy.

Lab 15: Perform Data Operations in Azure Managed Redis

- **Objective:** To build a Python application that performs high-speed data operations using Azure Managed Redis.
- **Key Topics Covered:** Redis hash data structures, key expiration (TTL), and SSL-secured client connections.

- **Hands-On Activity:** Deploying an Azure Managed Redis resource, storing and retrieving field-value pairs in hashes, and managing key lifetimes with Time-To-Live settings.
- **Real-World Application:** Applications use Redis as a high-performance cache to store frequently accessed user session data or transient AI results, reducing primary database load.

Lab 16: Publish and Subscribe to Events in Azure Managed Redis

- **Objective:** To implement real-time, asynchronous messaging using Redis Pub/Sub patterns.
- **Key Topics Covered:** Redis Pub/Sub, channels, wildcard pattern matching, and message broadcasting.
- **Hands-On Activity:** Developing publisher and subscriber console apps, sending order events to specific channels, and using pattern matching to subscribe to multiple event types.
- **Real-World Application:** Developers use Pub/Sub to decouple microservices, allowing components like notification engines to react to system events without direct coupling.

Lab 17: Implement Semantic Search in Azure Managed Redis

- **Objective:** To implement low-latency vector storage and semantic similarity search using the RediSearch module in Redis.
- **Key Topics Covered:** RediSearch, HNSW algorithm configuration, K-Nearest Neighbor (KNN) queries, and binary vector storage.
- **Hands-On Activity:** Creating a vector index with HNSW, storing product embeddings as binary data with metadata, and executing KNN searches to find similar items.
- **Real-World Application:** High-performance applications like recommendation engines use Redis for real-time semantic search where millisecond latency is critical.

Lab 18: Process Messages with Azure Service Bus

- **Objective:** To implement reliable messaging patterns for AI workflows using Azure Service Bus queues and topics.
- **Key Topics Covered:** Queues, topics, peek-lock delivery, dead-lettering, and SQL filters.
- **Hands-On Activity:** Sending inference requests to a queue, processing valid messages while dead-lettering malformed ones, and using topic subscriptions to fan out results based on priority.
- **Real-World Application:** Messaging layers ensure that AI requests are processed reliably even during traffic spikes or downstream component failures.

Lab 19: Publish and Receive Events with Azure Event Grid

- **Objective:** To route AI content moderation events using Azure Event Grid and pull delivery operations.
- **Key Topics Covered:** Event Grid Namespace, namespace topics, pull delivery mode, and CloudEvents v1.0.
- **Hands-On Activity:** Publishing moderation events, creating filtered subscriptions for specific event types (e.g., flagged vs. approved), and using receive/acknowledge/reject operations.

- **Real-World Application:** High-scale event routers like Event Grid manage millions of AI-generated events, directing them to appropriate handlers without manual polling.

Lab 20: Create an MCP Server with Azure Functions

- **Objective:** To expose Azure Functions as Model Context Protocol (MCP) servers to allow AI agents to discover and invoke external tools.
- **Key Topics Covered:** Model Context Protocol (MCP), Azure Functions MCP extension, and tool triggers.
- **Hands-On Activity:** Building an Azure Functions project with the MCP extension, defining tool triggers for document classification and summarization, and testing discovery via GitHub Copilot Agent mode.
- **Real-World Application:** Standardizing tool access via MCP allows enterprise APIs to be seamlessly used by various LLMs as functional, discoverable tools.

Lab 21: Manage Secrets with Azure Key Vault

- **Objective:** To securely store and manage sensitive AI application credentials using Azure Key Vault.
- **Key Topics Covered:** Azure Key Vault, RBAC-based authorization, secret versioning, and cached retrieval.
- **Hands-On Activity:** Storing API keys and connection strings, simulating credential rotation by creating new versions, and implementing a time-based cache to reduce API calls.
- **Real-World Application:** Security professionals use Key Vault to ensure that sensitive AI model keys are never exposed in source code or configuration files.

Lab 22: Retrieve Settings and Secrets from Azure App Configuration

- **Objective:** To centralize management of both non-sensitive application settings and Key Vault-stored secrets.
- **Key Topics Covered:** Azure App Configuration, label-based stacking, Key Vault references, and sentinel-based dynamic refresh.
- **Hands-On Activity:** Loading settings with environment-specific overrides, automatically resolving Key Vault references, and triggering dynamic configuration updates without application restarts.
- **Real-World Application:** DevSecOps teams use App Configuration to manage feature flags and model parameters across development, test, and production environments from a single location.

Lab 23: Instrument an App with OpenTelemetry

- **Objective:** To implement standardized observability in an AI document pipeline using OpenTelemetry and Application Insights.
- **Key Topics Covered:** OpenTelemetry SDK, custom spans, span attributes, and latency diagnostics.
- **Hands-On Activity:** Configuring the Azure Monitor OpenTelemetry Distro, creating a hierarchical trace tree with parent and child spans for pipeline stages, and diagnosing simulated latency in the portal.
- **Real-World Application:** Site Reliability Engineers (SREs) use instrumentation to gain deep visibility into complex AI processing steps and identify specific performance bottlenecks.

Lab 24: Query Logs with KQL

- **Objective:** To analyze application health and performance by writing advanced Kusto Query Language (KQL) queries in Application Insights.
- **Key Topics Covered:** KQL (Kusto Query Language), percentile calculations, log joins, and automated alert rules.
- **Hands-On Activity:** Generating sample telemetry, writing KQL queries to identify failed requests and correlate exceptions, analyzing dependency latency, and creating automated search-based alerts.
- **Real-World Application:** Monitoring teams use KQL to proactively detect error spikes and set up automated alerts to notify developers before users are impacted.