Pass Guaranteed Quiz 2026 AI-200: Developing AI Cloud Solutions on Azure Updated Reliable Test Pass4sure

It is known to us that our AI-200 study materials have been keeping a high pass rate all the time. There is no doubt that it must be due to the high quality of our study materials. It is a matter of common sense that pass rate is the most important standard to testify the AI-200 Study Materials. The high pass rate of our study materials means that our products are very effective and useful for all people to pass their exam and get the related certification.

Microsoft AI-200 Exam Syllabus Topics:

Section	Weight	Objectives
Topic 1: Integrate backend services and build event-driven architectures	25%	- Implement messaging and event systems • 1. Azure Service Bus for reliable messaging • 2. Azure Event Grid for event-driven processing • 3. Connect services and expose APIs securely - Build serverless APIs and workflows • 1. Orchestrate AI pipelines and workflows • 2. Azure Functions for AI integration and processing
Topic 2: Develop containerized AI solutions on Azure	25%	- Implement container hosting environments • 1. Deploy to Azure Container Apps and Azure Kubernetes Service (AKS) • 2. Configure scaling, networking, and security for containers • 3. Azure Container Registry: store, version, manage images - Monitor and troubleshoot containerized workloads • 1. Log analysis, health checks, and performance monitoring • 2. Manage configurations and secrets for containers

Topic 3: Develop AI solutions using Azure data services	30%	- Implement vector-enabled databases • 1. Azure Database for PostgreSQL with pgvector extension • 2. Azure Managed Redis for caching, streaming, and vector storage • 3. Azure Cosmos DB for NoSQL with vector search - Design and optimize data access and retrieval • 1. Implement hybrid search and retrieval patterns • 2. Indexing strategies, query optimization, and consistency models
Topic 4: Secure, monitor, and optimize AI solutions	20%	- Implement observability and reliability • 1. Logging, metrics, and distributed tracing • 2. Optimize performance, cost, and scalability • 3. OpenTelemetry and Azure Monitor integration - Manage security and configuration • 1. App Configuration for dynamic settings • 2. Managed identities and access control • 3. Azure Key Vault for secrets, keys, and certificates

AI-200 Related Content, Valid AI-200 Study Materials

As this version is called software version or PC version, maybe many candidates may think our AI-200 PC test engine may just be used on personal computers. At first, it can be only used on PC. But with our IT staff's improvement, now our Microsoft AI-200 PC test engine can be installed on all electronic products. You can copy to your mobile, Ipad or others. No matter anywhere or any time you want to learn AI-200 PC test engine, it is convenient for you. For busy workers, you can make the best of your time on railway or bus, mastering one question and answers every time will be great.

Microsoft Developing AI Cloud Solutions on Azure Sample Questions (Q18-Q23):

NEW QUESTION # 18
You are implementing an application by using Azure Event Grid to push near-real-time information to customers.
You have the following requirements:
* You must send events to thousands of customers that include hundreds of various event types.
* The events must be filtered by event type before processing.
* Authentication and authorization must be handled by using Microsoft Entra ID.
* The events must be published to a single endpoint
You need to implement Azure Event Grid
Solution: Publish events to a partner topic. Create an event subscription for each customer.
Does the solution meet the goal?

- A. No
- B. Yes

Answer: A

Explanation:
Detailed Explanation: The proposed partner-topic design does not match the stated need for one publisher endpoint that can fan events into thousands of logically distinct customer topics. Event Grid domains are specifically designed to expose a single publishing endpoint while containing large numbers of topics, with subscriptions and filtering applied at the topic level. Partner topics are

intended for partner-published SaaS events and do not substitute for the domain architecture required by this high-scale custom publishing scenario.
Study Guide Alignment: Azure service integration: Service Bus, Event Grid, Azure Functions triggers /bindings, and event-driven processing.
Official Microsoft Learn References: AI-200 Study Guide | Azure Event Grid event domains | Authenticate Event Grid publishing with Microsoft Entra ID

NEW QUESTION # 19
You need to deploy Azure function resources and apps by using an automated, version-controlled CI/CD pipeline that supports declarative infrastructure deployment. What should you use?

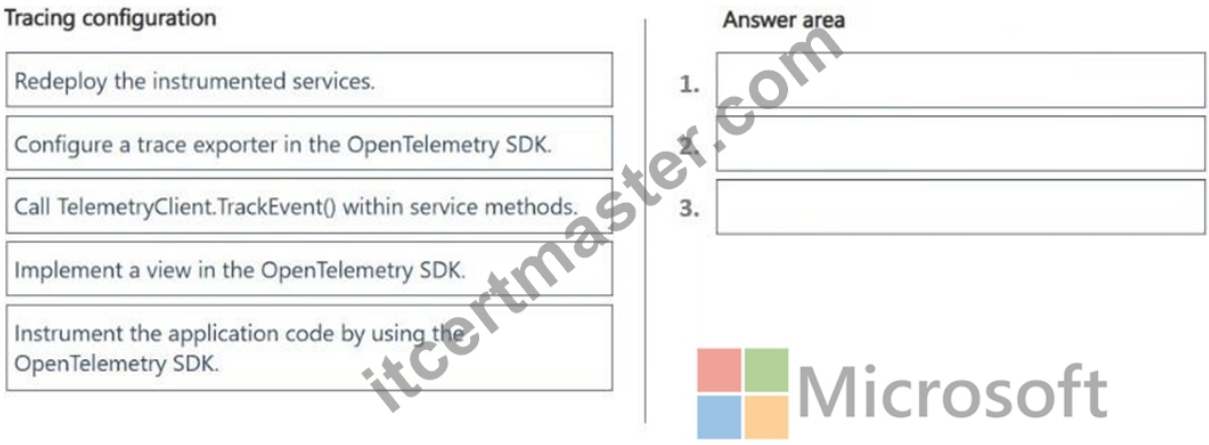
- A. Azure Functions Core Tools
- B. Local Git deployment
- C. GitHub Actions
- D. Azure CLI

Answer: C

Explanation:
Detailed Explanation: GitHub Actions is the correct orchestration mechanism among the options because the case requires an automated, version-controlled pipeline and Bicep-based declarative deployment. Microsoft documents GitHub Actions as a supported way to deploy Bicep and automate Azure resource deployment.
Local Git, Azure Functions Core Tools, and Azure CLI can participate in development or scripted deployment, but they do not by themselves satisfy the stated requirement for a repository-driven, repeatable CI /CD pipeline.
Study Guide Alignment: Security and operations: Key Vault, App Configuration, managed identity, OpenTelemetry, Azure Monitor, and KQL-based troubleshooting.
Official Microsoft Learn References: AI-200 Study Guide | Deploy Bicep with GitHub Actions | Automate Function app resource deployment

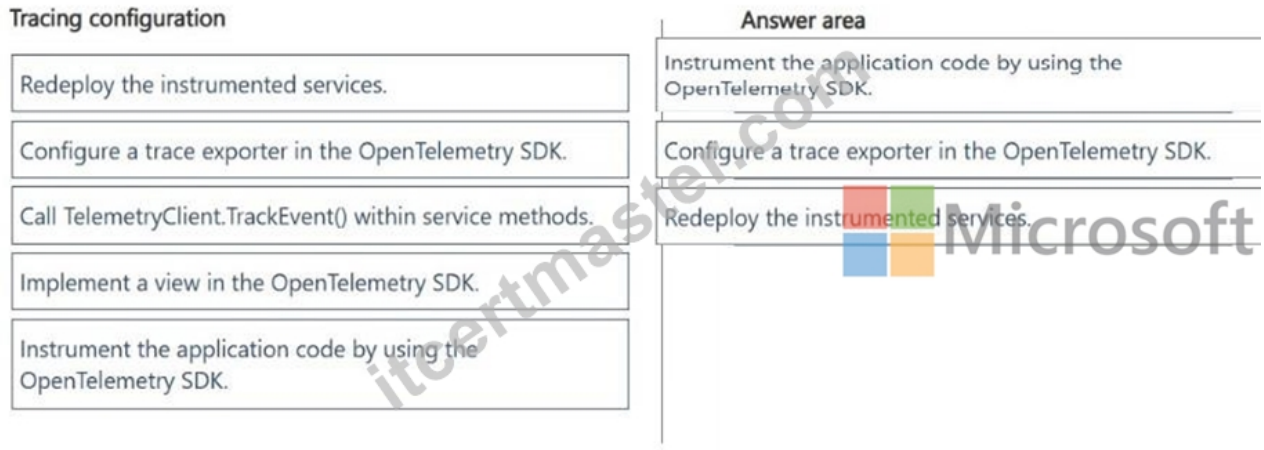
NEW QUESTION # 20
Case Study 2 - Proseware Inc.
Background
Proseware Inc. develops AI-powered knowledge management solutions for enterprise customers.
The company is modernizing its platform to support semantic search, intelligent document retrieval, and real-time partner integrations.
The engineering team uses Python and Azure SDKs. The architecture is being redesigned to support containerized microservices, vector search workloads, and serverless backend processing.
Planned Application Architecture
Microservices are containerized by using Docker.
Code for containerized microservices and Azure Function apps is developed locally but stored in a GitHub repository.
Custom images for containerized microservices are stored in Azure Container Registry (ACR).
Base images are stored in Docker Hub. Custom images must be rebuilt automatically whenever their base images are updated.
Azure Cosmos DB for NoSQL stores documents, metadata, and vector embeddings.
Azure Functions generate vector embeddings of Azure Cosmos DB for NoSQL-hosted documents and send messages to Service Bus to trigger search index updates.
Azure Container Apps (ACA) apps host backend API services that provide semantic search across Azure Cosmos DB for NoSQL documents. API services process Service Bus messages and update search indexes.
Azure Kubernetes Service (AKS) processes batch vector embedding regeneration for existing Azure Cosmos DB for NoSQL documents (whenever the embedding model is changed).
An extranet-facing containerized webhook allows business partners to submit documents to be processed by internal AI workflows for semantic search and retrieval.
Monitoring
Telemetry generated by Azure resources is sent to Azure Monitor.
A Log Analytics workspace is used to collect ACA apps logs, AKS container logs, and Azure Functions apps logs.
Monitoring of Azure Functions is currently implemented by using Azure Application Insights SDK instrumentation.
Business Requirements
Embeddings for new or updated Azure Cosmos DB for NoSQL-hosted documents must be automatically generated.
Backend API services must scale automatically during business hours.

Cold start delay of backend APIs must be minimized.
Secrets must be stored outside of container images.
Developers must be able to correlate telemetry across Azure Functions hosts and apps.
All tracing must be implemented by using OpenTelemetry SDK instrumentation.
Development efforts must be minimized.
Technical Requirements
Container images must be built automatically and validated before code updates are merged into the main branch.
Image build automation must run inside the Azure Container Registry, eliminating dependency on local developer machines and external build services.
Dependency of image builds on local developer machines must be eliminated.
Event-driven scaling in ACA must occur based on the number of pending messages in the Azure Service Bus queue.
Azure Cosmos DB for NoSQL RU consumption must be minimized.
Vector similarity search must use embeddings stored in Azure Cosmos DB for NoSQL.
The partner-facing containerized webhook service must run on Azure App Service.
Secrets must NOT be stored in container images, source control, or application configuration directly. They must be accessed securely at runtime.
All secrets must be stored centrally in Azure Key Vault and accessed at runtime through a managed identity.
Azure App Service must supply secrets at runtime without relying on external services.
Resources and workloads must be deployed by using Bicep templates through an automated, version-controlled pipeline. Local and command-line deployments must be eliminated to ensure repeatable, auditable deployments.
Known Issues
RU consumption spikes during vector similarity queries.
Drag and Drop Question
You need to implement trace correlation according to the business requirements.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.



Answer:

Explanation:



Explanation:

Scenario, Business Requirements
All tracing must be implemented by using OpenTelemetry SDK instrumentation.
Development efforts must be minimized.
Step 1: Instrument the application code by using OpenTelemetry SDK
This adds the foundational tracking APIs to your code so it can create and capture trace spans.
Step 2: Configure a trace exporter in the OpenTelemetry SDK
This instructs the initialized SDK where to transmit the captured trace data (e.g., to a local console or a cloud backend).
Step 3: Redeploy the instrumented services
This pushes the modified codebase and configuration changes into your active runtime environment.
Reference:
https://opentelemetry.io/docs/languages/python/instrumentation/

NEW QUESTION # 21
You are building a multi-agent solution in Azure AI Foundry where one agent handles scheduling and another handles billing questions, and a request may need both. What should you implement?

- A. A single monolithic prompt covering both domains
- B. Two separate applications with no coordination
- C. A single fine-tuned model trained on both domains only
- D. An orchestrator agent that routes sub-tasks to specialized agents and combines results

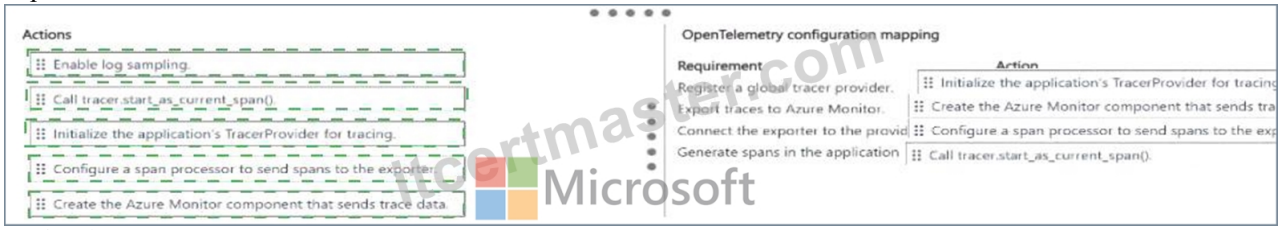
Answer: D

Explanation:
Multi-agent orchestration patterns use a coordinating (orchestrator) agent to decompose a request, delegate sub-tasks to specialized agents, and aggregate their outputs -- this is more maintainable and accurate than one broad prompt or model trying to cover disparate domains.

NEW QUESTION # 22
A Python API running in ACA must send distributed traces to Azure Monitor.
The API creates spans. However, no traces appear in Azure Monitor.
You need to configure the OpenTelemetry SDK pipeline to export traces to Azure Monitor.
What should you do? To answer, move the appropriate actions to the correct requirements. You may use each action once, more than once, or not at all. You may need to move the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

Answer:

Explanation:



Explanation:
Verified answer: Register a global trace provider # initialize the application TraceProvider. Export traces to Azure Monitor # create/configure AzureMonitorTraceExporter. Connect exporter to provider # add a span processor. Generate spans # use tracer.start_as_current_span(...).
Detailed Explanation: OpenTelemetry tracing is a pipeline: the application obtains a tracer from a configured provider, creates spans, passes ended spans through a span processor, and exports them using the Azure Monitor exporter. Creating spans alone is

insufficient; without the provider/processor/exporter chain, no telemetry reaches Azure Monitor. Log sampling and legacy Application Insights TrackEvent calls are unrelated to the required OpenTelemetry trace-export path.
Study Guide Alignment: Security and operations: Key Vault, App Configuration, managed identity, OpenTelemetry, Azure Monitor, and KQL-based troubleshooting.
Official Microsoft Learn References: AI-200 Study Guide | Enable Azure Monitor OpenTelemetry

NEW QUESTION # 23
.....

Using an updated Developing AI Cloud Solutions on Azure (AI-200) exam dumps is necessary to get success on the first attempt. So, it is very important to choose a Microsoft AI-200 exam prep material that helps you to practice actual Microsoft AI-200 Questions. Itcertmaster provides you with that product which not only helps you to memorize real Microsoft AI-200 questions but also allows you to practice your learning.