

Microsoft

Exam Questions AI-200

Developing AI Cloud Solutions on Azure



NEW QUESTION 1

You plan to deploy a web application to AKS.

The solution must:

- Scale out the application by adding more pods during peak CPU usage
- Expose the application internally within the cluster only

You need to configure a Kubernetes resource for each requirement

Which resources should you configure? To answer, move the appropriate resources to the correct requirements You may use each resource once, more than once, or not at all. You may need to move split bar between .. or scroll to view content.

Ingress

Ingress

Deployment

ClusterIP service

VerticalPodAutoscaler

HorizontalPodAutoscaler

Map Kubernetes resources to scaling and internal networking requirements

Requirement	Resource
Scale out the application.	
Expose the application internally only.	

- A. Mastered
B. Not Mastered

Answer: A

Explanation:

Verified Answer:Scale out the application: HorizontalPodAutoscaler. Expose internally only: ClusterIP service.

Detailed Explanation:The Kubernetes HorizontalPodAutoscaler increases or decreases pod replicas in response to CPU, memory, or other metrics, so it is the resource that implements CPU-driven pod scale-out. A ClusterIP service creates an internal service IP that is reachable within the Kubernetes cluster and is the normal choice for an internal-only service. A Deployment defines the workload but does not itself implement metric-driven autoscaling, while Ingress is intended to route inbound network traffic.

Study Guide Alignment:Containerized Azure workloads: registry builds, App Service containers, Container Apps revision/scaling behavior, and AKS deployment choices.

Official Microsoft Learn References:AI-200 Study Guide | AKS scaling overview | Kubernetes services in AKS

NEW QUESTION 2

You are developing a microservices-based application that uses Azure Container Apps. The application consists of several containerized services that handle tasks, such as processing orders, managing inventory, and generating reports. You deploy a new revision of the processing orders app.

Processing orders must be triggered by a web request and must always be available based on incoming web requests.

You need to validate that the replica is ready to handle incoming requests.

What should you implement?

- A. HTTP liveness probe
B. HTTP startup probe
C. TCP readiness probe
D. HTTP readiness probe
E. TCP liveness probe

Answer: D

Explanation:

Detailed Explanation:A readiness probe answers whether a running replica is currently ready to receive traffic. For a web-triggered order-processing service, an HTTP readiness probe can test the application endpoint and keep the replica out of the routing pool until it is able to handle requests. Liveness probes detect whether the process should be restarted, and startup probes protect slow-starting containers during initialization. A TCP readiness probe can test socket acceptance, but the HTTP readiness probe is the more application-aware choice for the HTTP service described.

Study Guide Alignment:Containerized Azure workloads: registry builds, App Service containers, Container Apps revision/scaling behavior, and AKS deployment choices.

Official Microsoft Learn References:AI-200 Study Guide | Health probes in Azure Container Apps

NEW QUESTION 3

You are developing a microservices-based application that uses Azure Container Apps The application consists of several containerized services that handle tasks, such as processing orders, managing inventory, and generating reports.

You must secure the container apps. All apps must reside in the same virtual network, share the same Dapr configuration, and share the same logging location.

Apps must support the configuration of the amount of memory and compute resources available to containers.

You need to configure the Azure Container App

How should you complete the CLI command? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

```
-n MyContainerApp -g MyResourceGroup --location eastus2
```

▼

env

create

add-on

▼

env

create

add-on

```
-n MyContainerApp -g MyResourceGroup --location eastus2
```

▼

--enable-mtls

--internal-only

--enable-workload-profiles

- A. Mastered
 B. Not Mastered

Answer: A

Explanation:

Verified Answer: `az containerapp env create ... --enable-workload-profiles`.

Detailed Explanation: The shared boundary for Container Apps is the managed environment. Apps in one environment can share the environment's virtual network integration, observability destination, and Dapr-related environment configuration. Enabling workload profiles provides selectable compute profiles and resource sizing options for workloads in that environment. The CLI must therefore create a Container Apps environment and enable workload profiles. Creating an individual container app alone would not establish the common environment-level network, logging, and resource-profile boundary required by the scenario. Study Guide Alignment: Containerized Azure workloads: registry builds, App Service containers, Container Apps revision/scaling behavior, and AKS deployment choices.

Official Microsoft Learn References: AI-200 Study Guide | Azure Container Apps environments

NEW QUESTION 4

You develop an ASP.NET Core app that uses Azure App Configuration. You also create an App Configuration containing 100 settings. The app must meet the following requirements:

- Ensure the consistency of all configuration data when changes to individual settings occur.
- Handle configuration data changes dynamically without causing the application to restart.
- Reduce the overall number of requests made to App Configuration APIs

You must implement dynamic configuration updates in the app.

What are two ways to achieve this goal? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

- A. Create and configure Azure Key Vault
 B. Implement the Azure Key Vault configuration provider
 C. Register all keys in the App Configuration store Set the refreshAll parameter of the Register method to false.
 D. Create and register a sentinel key in the App Configuration stor
 E. Set the refreshAll parameter of the Register method to true
 F. Decrease the App Configuration cache expiration from the default value.
 G. Increase the App Configuration cache expiration from the default value.
 H. Create and implement environment variables for each App Configuration store setting

Answer: CE

Explanation:

Detailed Explanation: A sentinel key allows the application to check one value and refresh all related configuration only when that sentinel changes, preserving a consistent set of settings instead of refreshing 100 keys independently. Increasing the refresh/cache interval reduces how frequently the application contacts App Configuration APIs. Together these measures support dynamic refresh without application restart while controlling request volume. Registering every key separately or decreasing the cache interval would increase API traffic, and Key Vault/environment variables do not solve coordinated App Configuration refresh. 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 | Dynamic configuration in ASP.NET Core

NEW QUESTION 5

You need to address the known issue resulting from vector similarity queries.

Which two actions should you perform? Each correct answer presents part of the solution. Choose two. NOTE: Each correct selection is worth one point.

- A. Add a composite index on the vector fields and metadata properties of the container.
 B. Reduce the numeric precision of the vector embeddings where supported.
 C. Set the account consistency level to Strong.
 D. Change the vector index type from flat to quantizedFlat or diskANN.

Answer: BD

Explanation:

The objective is to reduce RU consumption from vector similarity queries. Microsoft guidance identifies vector numeric precision and vector-index selection as major cost and performance levers. Using a lower supported vector precision can reduce storage and processing cost, while quantizedFlat and DiskANN are designed to reduce latency and RU consumption compared with flat search for appropriate data sizes. Strong consistency would increase resource cost rather than solve vector-search efficiency. A regular composite index is not a substitute for the vector index. Option B should be read as reducing vector numeric precision, not changing a generic indexing precision setting.

Study Guide Alignment: AI data-management workloads: Cosmos DB, PostgreSQL, caching, vector storage, vector retrieval, consistency, and connection

optimization.
Official Microsoft Learn References: AI-200 Study Guide | Vector search in Azure Cosmos DB | Optimize Cosmos DB vector search performance

NEW QUESTION 6

You are developing an AI application that retrieves database credentials from Key Vault by using the Azure SDK for Python. The application must use managed identity for authentication.

You review the following code segment that retrieves a secret from Key Vault:

```
01 from azure.identity import DefaultAzureCredential
02 from azure.keyvault.secrets import SecretClient
03
04 credential = DefaultAzureCredential()
05 client = SecretClient(
06     vault_url="https://vault.vault.azure.net/",
07     credential=credential
08 )
09
10 secret = client.get_secret("dbPassword", version="123")
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No. NOTE: Each correct selection is worth one point.

Secret retrieval behavior

Statement	Yes	No
The code retrieves a specific version of the dbPassword secret.	☐	☐
The code authenticates to Key Vault without storing client secrets in the application when deployed to an Azure hosted environment that has a managed identity enabled.	☐	☐
If dbPassword is rotated and a new version is created, subsequent calls to this code will retrieve the new version.	☐	☐

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

The get_secret call supplies an explicit version value, so it retrieves that particular version of dbPassword. DefaultAzureCredential can authenticate to Key Vault by using the Azure-hosted application's managed identity without embedding a client secret in the application. Because the code requests a fixed version, later secret rotation creates a newer version but subsequent executions of this exact code continue to request the specified old version. To follow the latest version automatically, the version parameter must be omitted.

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 | Managed identities for Azure resources | Use Key Vault references for App Service and Functions

NEW QUESTION 7

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.

Actions

Enable log sampling.

Call tracer.start_as_current_span()

Initialize the application's TracerProvider for tracing.

Configure a span processor to send spans to the exporter.

Create the Azure Monitor component that sends trace data.

OpenTelemetry configuration mapping

Requirement

Register a global tracer provider.

Export traces to Azure Monitor.

Connect the exporter to the provider.

Generate spans in the application code.

Action

- A. Mastered
- B. Not Mastered

Answer: A

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 8

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 system topic. Create an event subscription for each customer.

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

Explanation

System topics represent events emitted by Azure services and are not the correct resource for an application publishing its own hundreds of event types to thousands of customers through one custom endpoint. Event Grid domains are intended for exactly this type of application-level multi-tenant topic organization and expose one publishing endpoint for many topics. Therefore a system topic with a subscription per customer does not satisfy the stated publishing model, even though Event Grid supports filtering and secure delivery in other contexts.

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

Verification completed using official Microsoft Learn sources only. Original exhibits have been retained unchanged.

NEW QUESTION 9

You need to redesign the configuration approach so that:

configuration supports dynamic refresh;

secrets are stored outside App Configuration;

the application can securely access both App Configuration and Key Vault.

You need to enable secure dynamic configuration management for the platform.

Which three actions should you perform? Each correct answer presents part of the solution. Choose three.

NOTE: Each correct selection is worth one point.

- A. Store configuration values in environment variables.
- B. Use a service principal secret for both App Configuration and Key Vault access.
- C. Store API keys in Key Vault.
- D. Allow configuration updates with a polling interval.
- E. Use managed identity for both App Configuration and Key Vault access.

Answer: CDE

Explanation:

Explanation

The original answer omitted the secret-storage requirement. API keys and other secrets should be held in Key Vault, while App Configuration should retain non-secret configuration and Key Vault references. Dynamic refresh is implemented through the App Configuration refresh/polling mechanism rather than requiring an application restart. Managed identity provides credential-free access to both services in Azure. Environment variables do not provide centralized dynamic refresh, and a stored service-principal secret creates exactly the credential-management problem the design is intended to eliminate.

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 | Dynamic configuration in ASP.NET Core | Use Key Vault references in App Configuration | Managed identities for Azure resources

NEW QUESTION 10

You are reviewing the message processing code in a backend worker service.

You review the following Python code that initializes and starts a Service Bus processor

```
from azure.servicebus import ServiceBusClient, ServiceBusReceiveMode

@1 import json
@2 from azure.servicebus import ServiceBusClient, ServiceBusReceiveMode
@3 from azure.identity import DefaultAzureCredential
@4 credential = DefaultAzureCredential()
@5 with client.get_queue_receiver(
@6     queue_name="inference-requests",
```

Service Bus processor behavior and SDK defaults

Statement	Yes	No
Messages are removed from the queue as soon as they are received by the processor.	☐	☐
If message processing fails due to a transient service error, the message can be retried by another consumer.	☐	☐
Messages with invalid JSON payloads are retried until the maximum delivery count is reached.	☐	☐

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

The default Service Bus receive behavior is PeekLock rather than receive-and-delete, so merely receiving a message does not remove it from the queue; it must be settled successfully. If processing fails and the message remains unsettled or the lock is lost, it can become available for another delivery. Invalid JSON, however, is an application-level content problem: Service Bus does not inspect the payload as JSON and automatically retry it to MaxDeliveryCount solely because deserialization fails. The application's settlement/error path determines what happens.
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 | Service Bus message transfers, locks, and settlement | Service Bus messages, routing, and correlation

NEW QUESTION 10

You are designing an Azure Function app that exposes a public API. The solution must:
Validate incoming request data and return results immediately to the caller. Support Microsoft Entra ID authentication.
Scale automatically under variable load. You need to implement a trigger. Which trigger should you implement?

- A. HTTP
- B. Azure Queue storage
- C. Azure Event Grid
- D. Service Bus topic

Answer: A

Explanation:

An HTTP trigger is the only option that directly accepts a synchronous external request and can return a response immediately to the caller. Azure Functions uses HTTP triggers to implement serverless APIs and webhooks, while Microsoft Entra authentication can be applied to the Function App or API surface. Queue, Event Grid, and Service Bus triggers are asynchronous event-processing mechanisms and do not provide the request/response semantics required by this public API 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 Functions HTTP trigger

NEW QUESTION 13

You are developing several microservices to run on Azure Container Apps. External HTTP ingress traffic has been enabled for the Microservices. A deployed microservice must be updated to allow users to test new features. You have the following requirements:
* Enable and maintain a single URL for the updated microservice to provide to test users.
* Update the microservice that corresponds to the current microservice version. You need to configure Azure Container Apps.
Which features should you configure? To answer, select the appropriate options in the answer area. NOTE: Each correct selection is worth one point.

Answer Area

Requirement:

Single URL for test users

Feature

Revision Label

Revision mode

Container image

Container registry

Current microservice activation

Revision Label

Revision mode

Container image

Container registry

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Azure Container Apps revision labels provide a stable label-specific URL that can be reassigned to a different revision, which is appropriate for a consistent test endpoint. Revision mode controls whether one or multiple revisions can be active at the same time and therefore governs how the current and updated application versions are activated. The container image and registry identify what is deployed, but they do not provide the stable test URL or the application-level revision activation behavior requested. Study Guide Alignment: Containerized Azure workloads: registry builds, App Service containers, Container Apps revision/scaling behavior, and AKS deployment choices. Official Microsoft Learn References: AI-200 Study Guide | Azure Container Apps revisions | Azure Container Apps revision labels

NEW QUESTION 17

You have an Azure web app that uses Azure Cosmos DB as a data store. You create a Cosmos DB container by running the following PowerShell script:

```
$resourceGroupName = "testResourceGroup"
$accountName = "testCosmosAccount"
$databaseName = "testDatabase"
$containerName = "testContainer"
$partitionKeyPath = "/EmployeeId"
$autoscaleMaxThroughput = 5000

New-AzCosmosDBSqlContainer `
    -ResourceGroupName $resourceGroupName `
    -AccountName $accountName `
    -DatabaseName $databaseName `
    -Name $containerName `
    -PartitionKeyKind Hash `
    -PartitionKeyPath $partitionKeyPath `
    -AutoscaleMaxThroughput $autoscaleMaxThroughput
```

You create the following queries that target the container:

```
SELECT * FROM c WHERE c.EmployeeId > '12345'
SELECT * FROM c WHERE c.UserId = '12345'
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No. NOTE: Each correct selection is worth one point.

Answer Area

Statements	Yes	No
The minimum throughput for the container is 400 RU/s	☐	☐
The first query statement is an in-partition query.	☐	☐
The second query statement is a cross-partition query.	☐	☐

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

The question maps directly to the AI-200 objective ??Develop AI solutions by using Azure Cosmos DB for NoSQL,?? which includes running queries and optimizing query performance and Request Unit (RU) consumption.

Statement 1: No — The minimum throughput is not 400 RU/s.

The PowerShell command provisions the container with:

```
-AutoscaleMaxThroughput 5000
```

Azure Cosmos DB autoscale operates between approximately 10% and 100% of the configured maximum throughput . Microsoft documentation specifically gives the example of an autoscale container provisioned with 5,000 RU/s scaling between 500 RU/s and 5,000 RU/s . Therefore, for this container, the minimum operating autoscale throughput is 500 RU/s , not 400 RU/s.

Therefore:

??The minimum throughput for the container is 400 RU/s.?? # No

Statement 2: No — The first query is not an in-partition query. The container uses: /EmployeeId as its partition key. The first query is:

```
SELECT * FROM c WHERE c.EmployeeId > ' 12345 '
```

Although the query references the partition key, it uses a range predicate (>) , not an equality predicate. Microsoft explicitly states that a range filter on a partition key is not scoped to a single physical partition . To qualify as an in-partition query, the filter must identify the applicable partition, typically through an equality predicate such as:

```
WHERE c.EmployeeId = ' 12345 '
```

Microsoft ' s documentation provides essentially the same example: a query using DeviceId > ... against a container partitioned by DeviceId is not an in-partition query .

Therefore:

??The first query statement is an in-partition query.?? # No

Statement 3: Yes — The second query is a cross-partition query. The second query is:

```
SELECT * FROM c WHERE c.UserId = ' 12345 '
```

The container ' s partition key is /EmployeeId, not /UserId. Because the query contains no filter on the partition key , Azure Cosmos DB cannot route it to one logical partition. It must fan out the query across the applicable physical partitions and combine the results.

Microsoft describes this behavior directly: when a query does not contain a filter on the partition key, it must execute across the partitions.

Therefore:

??The second query statement is a cross-partition query.?? # Yes

NEW QUESTION 20

A semantic search application queries Azure Database for PostgreSQL and stores document embeddings and metadata in a table with the following columns:

- embedding (pgvector)
- department
- created_at

The application must return the top five most similar documents for a given query embedding only from the finance department. You need to implement semantic retrieval with metadata filtering.

Which query components should you select? To answer, move the appropriate query components to the correct requirements. You may use each query component once, more than once, or not at all. You may need to move the split bar between panes or scroll to view content.

Query components

WHERE department = 'Finance'

WHERE department LIKE 'dep%'

ORDER BY created_at DESC LIMIT 5

ORDER BY embedding <=> query_embedding LIMIT 5

Vector similarity search with metadata filtering

Requirement	Query component
Filter rows to finance.	
Rank by similarity and return the top five.	

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

The metadata predicate must restrict the candidate rows to the Finance department, so the `WHERE department = ' finance '` component supplies the required filter. The pgvector cosine-distance operator ` < = > ` orders rows by vector distance to the supplied query embedding, and `LIMIT 5` keeps only the five nearest matches. Ordering by creation date would rank recency rather than semantic similarity, and a wildcard department filter would not satisfy the Finance-only requirement.

Study Guide Alignment: AI data-management workloads: Cosmos DB, PostgreSQL, caching, vector storage, vector retrieval, consistency, and connection optimization.

Official Microsoft Learn References: AI-200 Study Guide | Vector similarity search with Azure PostgreSQL

NEW QUESTION 23

You deploy a private container image from Azure Container Registry (ACR) to App Service.

App Service must authenticate to ACR to pull the image.

The solution must NOT store static registry credentials.

You need to configure the secure image pull authentication.

Which configurations should you use? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Configure secure container image pull

Requirement

Authenticate App Service to ACR

Avoid storing static credentials

Configuration

Assign the Container Registry Repository Reader role to a managed identity.

Configure a webhook on ACR

Enable the admin user.

Embed credentials in Dockerfile.

Store the registry password in application settings.

Use managed identity with role assignment.

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

App Service can pull a private image from Azure Container Registry by using a managed identity rather than registry username/password credentials. The identity must be granted an

appropriate pull-only registry role at the registry scope, preserving least privilege. A webhook does not

authenticate an image pull, and enabling the ACR admin account or storing a registry password would

introduce static credentials. The managed-identity approach satisfies both secure authentication and credentialemelimination requirements.

Study Guide Alignment: Containerized Azure workloads: registry builds, App Service containers, Container

Apps revision/scaling behavior, and AKS deployment choices.

Official Microsoft Learn References: AI-200 Study Guide | Configure a custom container for App Service |

Managed identities for Azure resources

NEW QUESTION 25

A container in an AKS cluster repeatedly restarts.

Pod events show probe failures, although node-level CPU and memory metrics are normal.

You need to diagnose the cause of the repeating restarts.

What should you do first?

- A. Scale the deployment to more replicas.
- B. Decrease the initialDelaySeconds for the container liveness probe.
- C. Drain and reboot the node hosting the pod.
- D. Inspect the pod events and container logs.

Answer: D

NEW QUESTION 26

You configure ACR Tasks to automate image builds. Container images must rebuild when: Application updates occur. Base image updates occur, such as when the underlying OS image is updated. Regular scheduled rebuilds are required. You need to configure ACR Tasks to support automated image rebuilds. Which three triggers should you configure? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

A. A.Timer triggerB.Source code commit triggerC.Registry event triggerD.Base image update triggerE.Webhook notification trigger

Answer: ABD

Explanation:

To support automated container image rebuilds based on your requirements, you need to configure source triggers, base image triggers, and timer triggers in

Azure Container Registry (ACR) Tasks. Timer Triggers: Automates regular scheduled rebuilds. This uses a cron schedule expression to run image builds at

specific intervals (e.g., weekly or monthly) regardless of code changes. Source Triggers: Automates rebuilds when application updates occur. This monitors

changes in your source code repository (like GitHub or Azure Repos) and fires a build when code is committed. Base Image Triggers: Automates rebuilds when

the underlying OS or framework image updates. ACR tracks dependencies and automatically kicks off a new build when your defined FROM image changes in the

public registry or your private registry. Reference: <https://oneuptime.com/blog/post/2026-02-16-how-to-set-up-acr-tasks-for-automated-containerimage-builds-on-git-commit/view>

NEW QUESTION 30

DRAG DROP - You are developing several microservices to run on Azure Container Apps. The microservices must allow HTTPS access by using a custom

domain. You need to configure the custom domain in Azure Container Apps. In which order should you perform the actions? To answer, move all actions from the

list of actions to the answer area and arrange them in the correct order.

Actions

- Add the custom domain name.
- Bind the certificate.
- Enable ingress.
- Validate the custom domain name.
- Add DNS records to the domain provider.

Answer area

1.
2.
3.
4.
5.

- A. Mastered
B. Not Mastered

Answer: A

Explanation:

Answer area

1. Enable ingress.
2. Add DNS records to the domain provider.
3. Validate the custom domain name.
4. Add the custom domain name.
5. Bind the certificate.

NEW QUESTION 34

You have an Event Grid subscription that triggers an Azure Function.
You need to prevent loss of events in case the endpoint returns an HTTP 400 response.
Which action should you perform?

- A. Implement optimistic batchin
B. Implement asynchronous handshake validatio
C. Configure a dead-letter destinatio
D. Configure a retry policy.

Answer: C

Explanation:

Configuring a dead-letter destination is exactly the right step to take.
By default, when an endpoint returns an HTTP 400 (Bad Request) or HTTP 413 (Payload Too Large) response, Azure Event Grid immediately stops delivery attempts and drops the message. It treats these specific errors as non-transient, client-side issues, meaning it skips its standard 24-hour retry policy. Configuring a dead-letter destination ensures that these dropped events are safely preserved for future troubleshooting and reprocessing.
Reference:
<https://turbo360.com/blog/azure-event-grid-dead-letter-monitoring>

NEW QUESTION 37

You plan to deploy a container to an Azure App Service API app named api1. You host the source code for api1 in a GitHub repository. The container uses the API key at runtime to connect to a backend service.

The container must be able to retrieve the API key at runtime without exposing it in the source repository or Git commit history. You need to ensure that the API key remains outside of Git commit history and is available to the container at runtime. Solution: Store the API key as a GitHub repository secret. Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

Correct:

* Store the API key in Azure Key Vault and reference it from an App Service application setting.

Storing the API key in Azure Key Vault and referencing it via App Service application settings is the recommended, secure approach. This strategy completely removes sensitive credentials from your GitHub repository and Git commit history while injecting them safely into your container environment at runtime.

Incorrect:

* Embed the API key as a hardcoded environment variable in the Dockerfile.

* Store the API key as a GitHub repository secret.

Reference:

<https://www.qservicesit.com/full-stack-applications-on-azure>

NEW QUESTION 41

You are developing an AI search API that caches semantic search results in Redis.

Search results must remain cached for 10 minutes. If the underlying data changes, cached entries must NOT be returned.

You need to implement a cache-aside strategy to ensure data consistency.

Which two actions should you perform? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

- A. Configure a cache notification for key space event
- B. Delete related cache keys when the source data change
- C. Implement sliding expiration based on key acces
- D. Configure a 10-minute Time to Live on each key.

Answer: BD

Explanation:

Delete keys on mutation. Implement an event-driven or database-hook system to purge related Redis keys the moment the source data updates.

Use absolute expiration. Set a strict, un-extendable Time-To-Live (TTL) of 10 minutes when writing to the cache, ensuring the data naturally expires even if a deletion event is missed.

Incorrect:

[Not C]

Implementing a sliding expiration is not a good step for this specific architecture because it violates the requirement that entries must not be returned if the underlying data changes.

Reference:

<https://medium.com/real-world-net/net-caching-strategies-compared-redis-vs-memory-vs-hybrid-b8b8be1e708d>

NEW QUESTION 42

You deploy an AI application across multiple Azure regions.

The application must be able to view writes across all regions within a predictable time window.

You need to determine the appropriate consistency level.

What are two consistency levels you can use to achieve the goal? Each correct answer presents a complete solution.

NOTE: Each correct selection is worth one point.

- A. Session
- B. Strong
- C. Bounded staleness
- D. Eventual

Answer: BC

Explanation:

The two appropriate consistency levels for a globally distributed AI application needing to view writes within a predictable time window are Bounded Staleness and Strong.

Bounded Staleness: This provides a guarantee that reads will lag behind writes by a user-specified "staleness window" (either in time or number of operations). It delivers high availability and predictable read consistency across all regions, balancing performance with strict read-order guarantees.

Strong: This provides linearizable and absolute consistency, ensuring that writes are seen synchronously across all regions. It features an exact zero-second predictable time window, though it comes at the expense of higher latency for multi-region operations due to the time required to replicate globally.

Reference:

<https://learn.microsoft.com/en-us/azure/cosmos-db/consistency-levels>

NEW QUESTION 43

.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

AI-200 Practice Exam Features:

- * AI-200 Questions and Answers Updated Frequently
- * AI-200 Practice Questions Verified by Expert Senior Certified Staff
- * AI-200 Most Realistic Questions that Guarantee you a Pass on Your First Try
- * AI-200 Practice Test Questions in Multiple Choice Formats and Updates for 1 Year

100% Actual & Verified — Instant Download, Please Click
[Order The AI-200 Practice Test Here](#)