

Microsoft

Exam Questions AI-200

Developing AI Cloud Solutions on Azure



NEW QUESTION 1

You are investigating high latency in an AI search application that processes millions of requests daily. Telemetry is stored in Azure Monitor Logs. You must create a KQL query that correlates information from the AppRequests table and the AppDependencies table. The query must meet the following requirements:

- Include only data from the last 24 hours.
- Filter for failed requests only.
- Calculate the average duration of dependencies, grouped by operation.

The query must follow best practice to optimize the performance by minimizing the initial data scan. You need to create the query. In which order should 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

⋮ Summarize average dependency duration by operation.

⋮ Join the dependencies table.

⋮ Select the requests table.

⋮ Filter the failed requests.

⋮ Apply a time filter.

Answer Area

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Verified Answer:1) Select the requests table; 2) apply the 24-hour time filter; 3) filter failed requests; 4) join the dependencies table; 5) summarize average dependency duration by operation.

Detailed Explanation:KQL performs best when high-selectivity filters are applied as early as possible, especially the time predicate that limits the amount of data scanned. Starting from AppRequests, restricting to the last 24 hours, and then filtering failures reduces the left-side dataset before the join. The dependency table is then correlated with those requests, and aggregation is performed last to calculate average dependency duration by operation. Joining or summarizing before the time and failure filters would process more data than necessary.

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 | Optimize log queries in Azure Monitor

NEW QUESTION 2

You are configuring sampling for a distributed application that sends traces to Azure Monitor. The solution must:

- Preserve upstream sampling decisions across distributed traces
- Capture all spans during local testing.
- Sample 10 percent of traces in production.

You need to apply the appropriate sampling configuration for each requirement.

What should you do? To answer, move the appropriate configurations to the correct requirements. You may use each configuration 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.

Configurations

⋮ AlwaysOnSampler

⋮ BatchSpanProcessor

⋮ ParentBasedSampler

⋮ AtureMonitorTraceExporter

⋮ TraceIdRatioBasedSampler(0.1)

Selecting OpenTelemetry sampling strategies

Requirement

Maintain parent sampling decisions.

Record all spans during testing.

Sample 10 percent of traces in production.

Configuration

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Verified Answer:Preserve parent sampling decisions: ParentBasedSampler. Capture all spans locally: AlwaysOnSampler. Sample 10% in production: TraceIdRatioBasedSampler(0.1).

Detailed Explanation:Parent-based sampling propagates the sampling decision from the upstream parent, preventing broken sampling decisions across a distributed trace. AlwaysOnSampler records every span, which is suitable for local test environments where complete trace visibility is more important than telemetry volume. TraceIdRatioBasedSampler with 0.1 selects approximately ten percent of traces based on trace identifiers, providing deterministic fixed-ratio sampling for production. A batch span processor and an exporter control processing/export, not the sampling decision itself.

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 | OpenTelemetry sampling in Azure Monitor | Enable Azure Monitor OpenTelemetry

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 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 4

You are developing a .NET application that uses Azure Cosmos DB for NoSQL to store application data The application uses the Azure Cosmos DB for NoSQL SDK to interact with the database account. The application must perform the following tasks

- Initialize the connection by using the account endpoint and key.
- Define shared throughput.
- Perform create, read, update, and delete (CRUD) operations on items stored in a container

Components

Database

CosmosClient

Container

Indexing policy

Answer Area

Azure Cosmos DB for NoSQL SDK object hierarchy

Requirement

Initialize the connection by using the account endpoint and key.

Define shared throughput.

Perform item CRUD operations.

Component

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Verified Answer:Initialize connection: CosmosClient. Define shared throughput: Database. Perform item CRUD: Container.
Detailed Explanation:The Cosmos DB SDK follows the service resource hierarchy. CosmosClient represents the account connection and is initialized with the account endpoint and credential. Shared throughput is configured at the database level when multiple containers share database throughput. Item create, read, update, and delete operations are performed through a container object because containers hold items and define partitioning/indexing behavior. An indexing policy is configuration attached to a container, not the SDK object used for CRUD.
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 | Cosmos DB databases, containers, and items

NEW QUESTION 5

A Python API retrieves a document from Azure Database for PostgreSQL by using a SQL statement. The API accepts the document ID from user input. The current implementation inserts the document ID directly into the SQL statement. You need to secure the SQL statement execution by using a parameterized query. You must minimize the possibility of SQL injection. How should you update the implementation to execute the SQL statement safely? To answer, move the appropriate configurations to the correct requirements. You may use each configuration 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.

Configurations

Use a parameterized query.

Pass the ID as an argument.

Escape quotation marks in the ID.

Supply the parameter tuple to the SDK method.

Secure SDK-based query implementation using parameterized queries

Requirement

Modify the SQL statement structure.

Bind user input.

Execute the statement.

Configuration

- A. Mastered
- B. Not Mastered

Passing Certification Exams Made Easy

visit - <https://www.surepassexam.com>

Answer: A

Explanation:

Secure SDK-based query implementation using parameterized queries	
Requirement	Configuration
Modify the SQL statement structure.	Use a parameterized query.
Bind user input.	Pass the ID as an argument.
Execute the statement.	Supply the parameter tuple to the SDK

Verified Answer:Modify SQL: use a parameterized query. Bind input: pass the ID as an argument. Execute: supply the parameter tuple to the SDK/driver method.
Detailed Explanation:The security boundary is created by separating SQL syntax from user-supplied values. The statement should contain a parameter placeholder rather than concatenated input, and the document ID should be supplied separately through the database driver's parameter mechanism. The driver then performs the correct protocol-level binding and escaping. Manually escaping quotation marks is error-prone and does not provide the same injection resistance as true parameterization.
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 6

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.

Actions

⋮ Redeploy the instrumented services.

⋮ Configure a trace exporter in the OpenTelemetry SDK.

⋮ Instrument the application code by using the OpenTelemetry SDK.

⋮ Call TelemetryClient.TrackEvent() within service methods.

⋮ Implement a view in the OpenTelemetry SDK.

Answer Area

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Trace correlation requires the application to create OpenTelemetry spans and an exporter to send those spans to Azure Monitor. Instrumentation must therefore be present before the updated service is deployed. The previous Application Insights SDK instrumentation does not satisfy the case-study requirement that all tracing use OpenTelemetry. Configuring an OpenTelemetry view is unrelated to basic distributed-trace export, and calling TrackEvent is an Application Insights SDK pattern rather than the required OpenTelemetry approach.
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 7

You deploy an API to Azure Container Apps.
The solution must provide the following functionality:
• Support the concurrent activation of multiple application versions.
• Allocate a specific percentage of incoming requests to a secondary version
You need to configure revision behavior.
Which configurations should you use? To answer, move the appropriate configurations to the correct requirements. You may use each configuration 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.

Configurations

⋮ Traffic splitting

⋮ Single revision mode

⋮ Traffic isolation

⋮ Multiple revision mode

Revision management and traffic routing in Azure Container Apps

Requirement

Test a new revision without affecting production traffic.
Route 20 percent of traffic to a new revision.

Configuration

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Azure Container Apps supports side-by-side application versions through multiple revision mode. Once multiple revisions can be active, ingress traffic can be distributed by percentage across those revisions. This is the required combination for canary or controlled rollout scenarios. Single revision mode deactivates the previous revision as a new one becomes active, while traffic splitting is the specific feature that assigns a percentage of incoming requests to each active revision. 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

NEW QUESTION 8

A company is implementing a publish-subscribe (Pub/Sub) messaging component by using Azure Service Bus. You are developing the first subscription application.

In the Azure portal you see that messages are being sent to the subscription for each topic. You create and initialize a subscription client object by supplying the correct details, but the subscription application is still not consuming the messages.

You need to ensure that the subscription client processes all messages. Which code segment should you use?

- A. `subscriptionClient = new SubscriptionClient(ServiceBusConnectionString, TopicName, SubscriptionName);`
- B. `subscriptionClient.RegisterMessageHandler(ProcessMessagesAsync, messageHandlerOptions);`
- C. `await subscriptionClient.AddRuleAsync(new RuleDescription(RuleDescription.DefaultRuleName, new TrueFilter()));`
- D. `await subscriptionClient.CloseAsync();`

Answer: B

Explanation:

Creating a subscription client establishes the client object but does not start application-level message processing. The message handler must be registered so received subscription messages are delivered to the application callback. Adding a default TrueFilter is only necessary when subscription rules require adjustment and does not start the receiver; the default subscription rule already accepts messages in a normal subscription. Closing the client would stop processing rather than enable it.

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

NEW QUESTION 9

You optimize an AI inference API that uses Redis caching.

You must reduce the risk of serving outdated data while minimizing cache management overhead. You need to implement the caching strategy that satisfies the requirements. What should you do?

- A. Reset expiration on each read.
- B. Disable expiration for cache keys.
- C. Set eviction policy to allkeys-lru.
- D. Trigger invalidation when source data changes

Answer: D

Explanation:

If the priority is to minimize stale results, source-driven invalidation is the direct control: when the underlying record changes, delete the corresponding cache entry so the next request repopulates it from current data. Sliding expiration can keep a frequently accessed stale item alive, disabling expiration is worse, and an allkeys-LRU policy evicts according to memory pressure rather than data freshness. Invalidation therefore best addresses correctness without requiring constant cache-wide maintenance.

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 | Azure Managed Redis documentation

NEW QUESTION 10

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 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

01 import json
02 from azure.servicebus import ServiceBusClient, ServiceBusReceiveMode
03 from azure.identity import DefaultAzureCredential
04 credential = DefaultAzureCredential()
05 with client.get_queue_receiver(
06     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 14

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. Choose two
NOTE: Each correct selection is worth one point.

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

Answer: AD

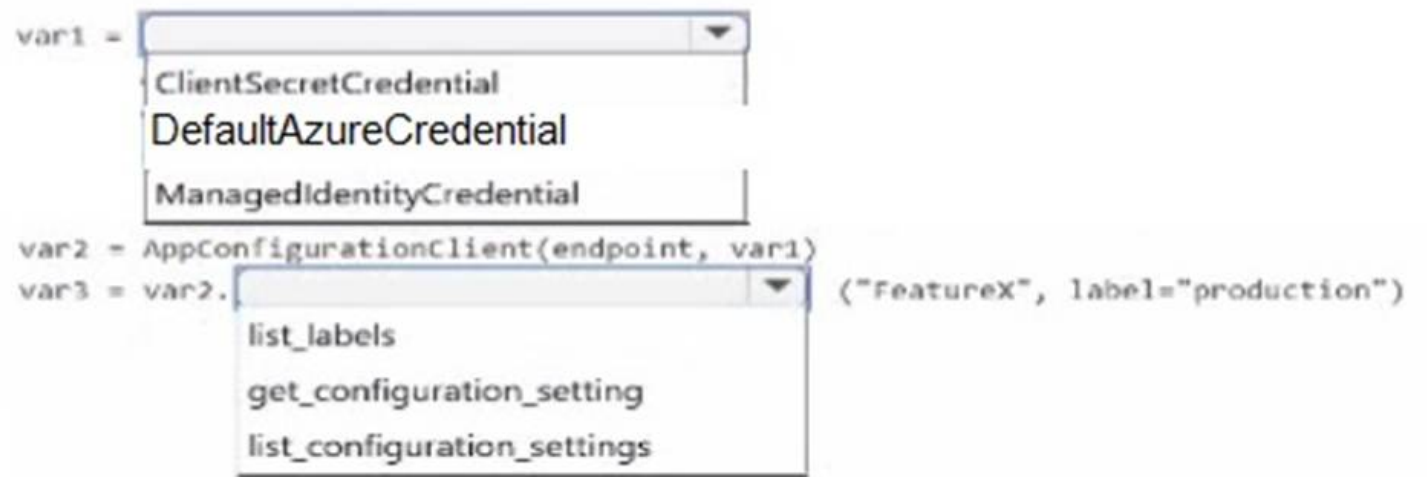
Explanation:

The cache-aside design needs both bounded lifetime and explicit invalidation. A ten-minute TTL ensures cached search results do not persist indefinitely, while deleting related keys when source data changes prevents known-stale entries from being returned during that ten-minute window. Sliding expiration would extend lifetime based on access and can preserve stale data. Keyspace notifications report Redis key events; they do not replace application logic that invalidates entries when the authoritative source changes.
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 | Azure Managed Redis documentation

NEW QUESTION 16

You are using the Python SDK to develop a containerized AI application that retrieves the value of a runtime setting stored in an Azure App Configuration resource. The application will run in Azure in the security context of a managed identity.
The application must work in a local development environment without any code changes.
You need to complete the code that implements the retrieval.
How should you complete the code? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

SDK-based configuration retrieval



- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

DefaultAzureCredential is designed to support the same application code across local development and Azure-hosted environments. Locally it can use developer credentials, while in Azure it can use the workload's managed identity. The App Configuration client's get_configuration_setting method retrieves a specific key/label pair, which matches the code shown. Using ManagedIdentityCredential directly would not provide the same local-development credential chain without code or environment-specific handling.

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 | Azure App Configuration Python client | Managed identities for Azure resources

NEW QUESTION 19

You deploy a production Azure Function app that connects to an Azure SQL Database. The solution must provide the following functionality:

- Prevent secrets from being exposed in source control.
- Support secret rotation without redeploying the function app.
- Avoid downtime during credential updates.

You need to configure secure and maintainable secret management. What should you configure?

- A. Application settings with Key Vault references
- B. Environment variables in local.settings.json
- C. Parameter file stored in source control
- D. Hard coded connection string in the startup class

Answer: A

Explanation:

Key Vault references in Function/App Service application settings keep the connection secret outside source control while allowing the platform to resolve it at runtime by identity. When a referenced secret version is not fixed, rotation can be picked up without rebuilding or redeploying the function code.

`local.settings.json` is intended for local development and must not be used as a production secret store. A source-controlled parameter file or hard-coded connection string directly violates the security requirement.

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 | Use Key Vault references for App Service and Functions

NEW QUESTION 24

You deploy a new revision of an app in Azure Container Apps.

You need to gradually shift production traffic to the revision while monitoring performance. In addition, you need to be able to quickly roll back. 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. Use traffic splitting
- B. Increase replica count.
- C. Restart the revision.
- D. Use single revision mode
- E. Enable multiple revision mode.

Answer: AE

Explanation:

Gradual production rollout requires more than one revision to remain active and a mechanism to divide traffic between them. Multiple revision mode permits concurrent active revisions, and traffic splitting lets a specified percentage of requests flow to the new revision while the remainder stays on the proven revision. This also enables rapid rollback by returning traffic to the prior revision. Increasing replicas or restarting a revision changes capacity or process state, not release traffic distribution.

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

NEW QUESTION 26

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

Filter rows to finance.

Rank by similarity and return the top five.

Query component

- 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 28

You are reviewing an Azure Function app that processes incoming order requests for a company. The function must:

- Accept order submissions from an external client application.
- Require controlled access for security
- Return a response containing the processed request payload.

You review the following code segment:

```
01 import azure.functions as func
02 app = func.FunctionApp(http_auth_level=func.AuthLevel.FUNCTION)
03 @app.route(route="processorder", methods=["POST"])
04 def processorder(req: func.HttpRequest) -> func.HttpResponse:
05     result = req.get_body().decode("utf 8")
06     return func.HttpResponse(result)
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No.

NOTE: Each correct selection is worth one point.

Code-level analysis of HTTP trigger configuration

Statement	Yes	No
The function requires a function key for invocation.	☐	☐
The function supports HTTP GET requests.	☐	☐
The response returns the request body.	☐	☐
The function validates the request body before returning a response.	☐	☐

- A. Mastered
B. Not Mastered

Answer: A

Explanation:

auth_level=FUNCTION` means invocation requires a function key unless a stronger platform authentication layer is configured, so the first statement is true. The route explicitly permits POST, not GET. The function reads `req.get_body()` and returns that value in the HttpResponse, so the response contains the request body. No schema, type, required-field, or other payload validation is performed before the response is constructed, making the final statement false.

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 31

You develop a message-processing service deployed to Azure Container Apps. The service reads messages from an Azure Service Bus queue. The solution must minimize costs by ensuring NO compute resources are consumed when the queue is empty. You need to configure scaling for the service. Which two actions should you perform? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

A. A.Increase the scaling rule to allow for the maximum running replica count.B.Configure the scaling rule to allow for the termination of all active replicas.C.Configure a Kubernetes Event-driven Autoscaler rule that monitors queue length.D.Enable HTTP ingress concurrency scaling.

Answer: BC

NEW QUESTION 33

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 37

Drag and Drop

You are developing a new page for a website that uses Azure Cosmos DB for data storage. The feature uses documents that have the following format:

```
{
  "name": "John",
  "city": "Seattle"
}
```

You must display data for the new page in a specific order. You create the following query for the page:

```
SELECT *
FROM People p
ORDER BY p.name, p.city DESC
```

You need to configure an Azure Cosmos DB policy to support the query.
How should you configure the policy? To answer, drag the appropriate JSON segments to the correct locations. Each JSON segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

JSON segments	Answer Area
orderBy	<pre>{ "automatic": true, "ngMode": "Consistent", "includedPaths": [{ "path": "/" }], "excludedPaths": [], "": [{ "path": "/name", "order": "descending" }, { "path": "/city", "order": " }] }</pre>
sortOrder	
ascending	
descending	
compositeIndexes	

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Box 1: compositeIndexes
Azure Cosmos DB requires a defined composite index for any query utilizing an ORDER BY clause with multiple properties.
Box 2: descending
In an ORDER BY clause, properties without an explicit direction default to ascending order. Your query ORDER BY p.name, p.city DESC translates to an ascending sort on /name and a descending sort on /city.
Reference:
<https://docs.azure.cn/en-us/cosmos-db/index-policy>

NEW QUESTION 42

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 43

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 46

You deploy multiple instances of a change feed processor to handle a high ingestion rate within Azure Cosmos DB for NoSQL.

Each processor instance must process a different subset of partitions.

You need to ensure the workload is load-balanced across all processor instances.

What should you configure?

- A. indexing precision
- B. strong consistency
- C. lease container
- D. autoscale throughput

Answer: C

Explanation:

To handle a high ingestion rate in Azure Cosmos DB for NoSQL, you must deploy multiple instances of the change feed processor using the same workflow name and same lease container.

Reference:

<https://github.com/Azure/azure-cosmos-dotnet-v3/wiki/Comprehensive-Test-Suite-for-Azure-Cosmos-DB-Change-Feed-Processor:-All-Versions-and-Deletes>

NEW QUESTION 50

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 52

.....

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](#)