

Microsoft.AI-200.v2026-10-09.q70

Exam Code:	AI-200
Exam Name:	Developing AI Cloud Solutions on Azure
Certification Provider:	Microsoft
Free Question Number:	70
Version:	v2026-10-09
# of views:	102
# of Questions views:	700
https://www.freecram.net/torrent/Microsoft.AI-200.v2026-10-09.q70.html	

NEW QUESTION: 1

You are designing an Azure Function app that processes large image uploads submitted by users through an HTTP endpoint.

The solution must:

- * Prevent client timeouts by decoupling image processing from the initial upload request.
- * Support automatic retry behavior for failed processing attempts.
- * Scale the background processing independently of the rate of incoming HTTP uploads.

You need to design a scalable and reliable asynchronous processing solution.

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

NOTE: Each correct selection is worth one point.

- A. Implement a queue-triggered function for image processing.
- B. Configure retry policies on a storage queue.
- C. Process the image inside the HTTP-triggered function.
- D. Persist upload metadata to Azure Storage.

Answer: (SHOW ANSWER)

The correct architecture uses an Azure Storage queue as an asynchronous buffer between the HTTP ingestion path and the computationally expensive image-processing workload. The HTTP-triggered function should accept the upload, enqueue a work message, and return promptly rather than waiting for image processing to finish. Microsoft specifically recommends asynchronous patterns for long-running HTTP operations because an HTTP-triggered Azure Function can encounter the Azure Load Balancer response timeout even though the underlying function continues executing.

A queue-triggered function then performs image processing independently. Azure Functions applies target- based scaling to Azure Queue Storage triggers based on queued workload, allowing worker capacity to expand as the backlog increases rather than being directly tied to HTTP arrival processing.

For reliability, Azure Queue-triggered Functions provide retry behavior. The visibilityTimeout setting controls the delay before a failed message becomes available for another attempt, while maxDequeueCount controls how many processing attempts occur before the message is transferred to the < queue-name > - poison queue. The default maximum is five attempts.

C defeats the asynchronous requirement. D can be useful for tracking upload state, but it does not itself provide workload decoupling, retry handling, or queue-driven scaling.

Study Guide references: Azure Functions # Queue Storage triggers and bindings; event-driven scaling; retry /poison queue behavior; asynchronous HTTP processing.

NEW QUESTION: 2

You are provisioning and configuring a Service Bus processor for AI batch jobs.

The processor must connect to an existing queue, register handlers for message and error processing, and then begin receiving messages. You need to provision and configure the Service Bus processor for message and error handling. Which four 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.

Actions

Dead-letter failed messages.

Create the Service Bus client.

Create the Service Bus processor for the queue.

Register message and error handlers.

Start the message processor.

Answer Area



Answer:

Actions

Dead-letter failed messages.

Create the Service Bus client.

Create the Service Bus processor for the queue.

Register message and error handlers.

Start the message processor.

Answer Area

Create the Service Bus client.

Create the Service Bus processor for the queue.

Register message and error handlers.

Start the message processor.

Explanation:

Answer Area



1 Create the Service Bus client.

2 Create the Service Bus processor for the queue.

3 Register message and error handlers.

4 Start the message processor.

Verified Answer: 1) Create the Service Bus client; 2) create the queue processor/receiver; 3) register message and error handlers; 4) start the message processor.

Detailed Explanation: The processor depends on a namespace client/connection, so the client is created first.

Next, the queue-specific processor or receiver is created from that client. Message and error callbacks must be registered before processing starts so incoming messages and failures have defined handlers.

Starting the processor is therefore the final step. Dead-lettering failed messages is an optional application settlement decision and is not a prerequisite for constructing and starting the processor.

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: 3

A company uses Azure Monitor Application Insights to monitor application behavior, including incoming requests and dependencies. You must identify failed requests from the last hour. You must also calculate the average duration of failed request dependency calls, grouped by operation name. You need to analyze telemetry in Application insights. Which operators should you mm? To answer, select the appropriate options in the answer area NOTE: Each cot reel selection is worth one point.

Completing a KQL query

requests

timestamp > ago(1h) and success == false

extend

project

where

dependencies

distinct

render

where

project operation_id, depDuration = duration

) on operation_id

avgDepDuration = avg(depDuration) by operation_Name

extend

project

summarize



Answer:

Completing a KQL query

requests

timestamp > ago(1h) and success == false

extend

project

where

dependencies

distinct

render

where

project operation_id, depDuration = duration

) on operation_id

avgDepDuration = avg(depDuration) by operation_Name

extend

project

summarize



Explanation:

Completing a KQL query

requests

where timestamp > ago(1h) and success == false

join kind=inner (

dependencies

where timestamp > ago(1h)

project operation_id, depDuration = duration

) on operation_id

avgDepDuration = avg(depDuration) by operation_Name

summarize



NEW QUESTION: 4

You are evaluating a fine-tuned Azure OpenAI model against the base model before promoting it to production.

Which Azure AI Foundry capability should you use?

- A.** Prompt flow bulk testing / evaluation with metrics (groundedness, coherence, relevance)
- B.** Azure Monitor Application Insights only
- C.** Content Safety category configuration
- D.** Azure AI Search indexer scheduling

Answer: ([SHOW ANSWER](#))

Use Azure AI Foundry evaluation with a representative evaluation dataset and quality metrics to compare the fine-tuned model against the base model before production promotion. Among the available options, this corresponds to Prompt flow bulk testing / evaluation . Foundry evaluation supports systematic testing of model or application outputs across many test records rather than relying on manual inspection of isolated prompts. Microsoft provides built-in evaluators including Groundedness, Coherence, Relevance, Fluency, and Response Completeness , allowing teams to quantify quality consistently across candidate implementations.

For a valid comparison, run the same representative dataset against both the base and fine-tuned deployments and compare the resulting evaluation metrics. This creates objective evidence about whether fine-tuning improves the targeted behavior without materially degrading relevance, coherence, grounding, or other required quality dimensions. Prompt flow evaluation workflows also support batch execution, per-record scores, and aggregated metrics for comparing variants.

Application Insights is essential for production telemetry such as latency, failures, and usage, but it is not by itself a model-quality comparison framework. Content Safety configuration evaluates or controls harmful content rather than comparative model quality. Azure AI Search indexer scheduling manages search-index ingestion and is unrelated to model evaluation.

Therefore, A is the correct capability.

Study Guide references: Microsoft Foundry # evaluations; evaluation datasets; bulk/batch testing; Groundedness; Coherence; Relevance; comparative model assessment.

NEW QUESTION: 5

You are designing an Azure Database for PostgreSQL table for semantic search. Queries frequently filter on the created_at column. The schema must support vector similarity search and reliable date filtering

You need to ensure that the table meets the requirements.

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.** Store embeddings in a pgvector column.
- B.** Store embeddings in a varchar column.
- C.** Use a typed timestamp column for created_at.
- D.** Store created_at as a free-form string.

Answer: ([SHOW ANSWER](#))

Detailed Explanation: A semantic-search schema should use the pgvector vector type for embeddings so PostgreSQL can apply vector distance operators and vector indexes. The creation timestamp should use an actual timestamp data type so comparisons, range predicates, ordering, and indexes use date semantics rather than string ordering. Storing vectors or dates as free-form character data would sacrifice type validation and make similarity or date filtering less efficient and less reliable.

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 provisioned an Azure Cosmos DB for NoSQL account named account1 with the default consistency level.

You plan to configure the consistency level on a per request basis. You plan to request Consistent Prefix consistency on a per-request basis.

You need to identify the resulting consistency level for read and write operations.

Which levels result from this configuration? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Answer Area

Operation type

Read operations

Write operations

Resulting consistency level

strong

session

consistent prefix

strong

session

consistent prefix

Microsoft

Answer:

Answer Area

Operation type

Read operations

Write operations

Resulting consistency level

strong

session

consistent prefix

strong

session

consistent prefix

Microsoft

Explanation:

Answer Area

Operation type

Read operations

Write operations

Resulting consistency level

consistent prefix

session

Microsoft

Verified Answer: Read operations: Consistent Prefix. Write operations: Session (the account default remains effective for writes).

Detailed Explanation: Azure Cosmos DB allows a client or request to override consistency for reads.

Microsoft explicitly notes that such an override applies only to reads; it does not change how writes are committed and replicated under the account's configured consistency. Because a new account uses Session consistency by default, requesting Consistent Prefix affects the reads while the account continues to use its Session consistency behavior for writes. This distinction is the key point tested by the hotspot.

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 | Manage Cosmos DB consistency levels | Cosmos DB consistency level choices

NEW QUESTION: 7

You are developing an Azure Functions app.

All functions in the app meet the following requirements:

- * Run until either a successful run or until 10 run attempts occur.
- * Ensure that there are at least 20 seconds between attempts for up to 15 minutes.

You need to configure the host.json file.

How should you complete the code segment? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Answer Area

credential =

client =

client =

DefaultAzureCredential

ChainedTokenCredential

ManagedIdentityCredential

AddAzureAppConfiguration

DefaultAzureCredential

ChainedTokenCredential

ManagedIdentityCredential

AddAzureAppConfiguration

()

{base_url="...", credential=credential}

{base_url="...", credential=credential}

Answer:

credential =

client =

client =

DefaultAzureCredential

ChainedTokenCredential

ManagedIdentityCredential

AddAzureAppConfiguration

DefaultAzureCredential

ChainedTokenCredential

ManagedIdentityCredential

AddAzureAppConfiguration

()

{base_url="...", credential=credential}

{base_url="...", credential=credential}

Explanation:

credential =

client =

AddAzureAppConfiguration

ManagedIdentityCredential

()

{base_url="...", credential=credential}

NEW QUESTION: 8

You have an existing AKS cluster and a container image stored in Azure Container Registry.

You must deploy a new version by using a Deployment manifest that supports rolling updates without interrupting traffic.

You need to perform a rolling update by using a manifest file.

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.

QUESTIONS

Microsoft

- Scale deployment to zero before updating.
- Delete the existing deployment and recreate it.
- Verify rollout status.
- Update the image tag in the deployment manifest.
- Apply the updated manifest.

ANSWER AREA

Answer:

Actions

- Scale deployment to zero before updating.
- Delete the existing deployment and recreate it.
- Verify rollout status.
- Update the image tag in the deployment manifest.
- Apply the updated manifest.

Microsoft

Answer Area

- Update the image tag in the deployment manifest.
- Apply the updated manifest.
- Verify rollout status.

Explanation:

Answer Area

Microsoft

- 1 Update the image tag in the deployment manifest.
- 2 Apply the updated manifest.
- 3 Verify rollout status.

NEW QUESTION: 9

You plan to deploy an Azure Container Apps app named App1. App1 will use an access key to connect to a backend API.

The solution must store the key outside of the App1 environment and minimize maintenance efforts.

You need to configure secure key storage for App1.

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

NOTE: Each correct selection is worth one point.

Secret configuration in Azure Container Apps



Requirement
Provide the key storage for App1.

Secure App1 access to the key storage.

Configuration

Azure Key Vault key
Azure Key Vault secret
Azure Container Apps secret

Service principal
Managed identity
Workload identity

Answer:

Secret configuration in Azure Container Apps

Requirement
Provide the key storage for App1.

Secure App1 access to the key storage.

Configuration

Azure Key Vault key
Azure Key Vault secret
Azure Container Apps secret

Service principal
Managed identity
Workload identity

Explanation:

* Provide the key storage for App1: Azure Key Vault secret

* Secure App1 access to the key storage: Managed identity

An API access key is a sensitive string value, so it should be stored as an Azure Key Vault secret , not as an Azure Key Vault cryptographic key. Azure Container Apps supports direct Key Vault secret references , allowing sensitive values to remain centrally stored outside the container app rather than being embedded in images or supplied directly as Container Apps secret values. Microsoft recommends Key Vault references for production secrets because they provide centralized lifecycle management and avoid placing secret values directly in application configuration.

App1 should authenticate to Key Vault by using a managed identity . Azure Container Apps supports both system-assigned and user-assigned managed identities, enabling the workload to obtain Microsoft Entra tokens without maintaining client secrets or credentials. The identity must be granted appropriate Key Vault permissions, such as the Key Vault Secrets User RBAC role, to retrieve the stored secret. This configuration also minimizes maintenance. When the Key Vault reference omits a specific secret version, Container Apps uses the latest version and automatically detects newer versions, retrieving the updated value within approximately 30 minutes.

Therefore, the required combination is Azure Key Vault secret + Managed identity .

Study Guide references: Azure Container Apps # secrets management; Key Vault references; managed identities; secret rotation and runtime retrieval.

NEW QUESTION: 10

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: ([SHOW ANSWER](#))

Detailed 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: 11

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. The API key must not be stored in application configuration directly. The API key must be accessed securely from Azure Key Vault.

You need to ensure that the API key is stored securely in Azure Key Vault and is available to the container at runtime without being exposed in source control or Git commit history.

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

Does the solution meet the goal?

- A. Yes
- B. No

Answer: ([SHOW ANSWER](#))

NEW QUESTION: 12

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.

You need to implement the SDK components required for the application to access and manage data in Azure Cosmos DB for NoSQL.

Which SDK components should you use? To answer, move the appropriate components to the correct requirements. You may use each component 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.

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

Answer:

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

CosmosClient

Database

Container

Explanation:

- * Initialize the connection by using the account endpoint and key: CosmosClient
- * Define shared throughput: Database
- * Perform item CRUD operations: Container

The Azure Cosmos DB for NoSQL .NET SDK uses a clear resource hierarchy. CosmosClient is the top-level client object and is responsible for connecting to the Azure Cosmos DB account. Microsoft identifies it as the primary SDK class for accessing account-level resources and databases. It can be initialized by using the account endpoint together with an account key or another supported credential.

A Database represents a database resource within the account. Shared provisioned throughput is configured at the database level , allowing multiple containers within that database to share the same RU/s allocation.

Microsoft documents that setting throughput on the database creates a shared-throughput database.

A Container is the SDK object used to interact with items stored in a Cosmos DB container. Microsoft specifically describes the Container class as the object used for item operations such as creating, reading, updating, deleting, and querying documents.

The Indexing policy option is not used for any of these requirements. It controls how Cosmos DB indexes document properties for query execution, not connectivity, shared throughput, or CRUD access.

Study Guide references: Azure Cosmos DB for NoSQL .NET SDK # CosmosClient; Database; shared throughput; Container; item CRUD operations.

You need to add a custom voice that matches your company ' s brand for a text-to-speech feature, distinct from any standard neural voice.

What should you use?

- A. Azure AI Speech prebuilt neural voices
- B. Azure AI Speech Custom Neural Voice
- C. Azure AI Language sentiment analysis
- D. Azure AI Translator

Answer: (SHOW ANSWER)

Use Azure AI Speech Custom Neural Voice . Microsoft describes custom voice as a text-to-speech capability for creating a unique synthetic voice for a product, brand, or character rather than selecting one of the standard voices provided by the Speech service. The organization supplies authorized voice recordings and corresponding scripts, and the custom voice workflow creates a voice model that can subsequently be deployed and consumed through Speech text-to-speech functionality.

This directly satisfies the requirement that the voice be distinct from the standard neural voice catalog.

Prebuilt neural voices are high-quality, ready-to-use voices supplied by Microsoft, but they do not create a company-specific branded voice. Microsoft explicitly differentiates standard voices from custom voices and identifies custom voice as the mechanism for creating a voice unique to a brand.

Custom Neural Voice is a Limited Access capability. Organizations must meet Microsoft ' s eligibility and responsible-AI requirements, including obtaining appropriate consent from voice talent and receiving access approval for the intended use case.

Azure AI Language sentiment analysis evaluates text sentiment, while Azure AI Translator performs language translation. Neither creates a custom text-to-speech voice.

Study Guide references: Azure AI Speech # text-to-speech; Custom Neural Voice; voice model creation; branded synthetic voices; responsible AI and Limited Access.

NEW QUESTION: 14

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

Microsoft

Statements

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.

Yes

No

Answer:

Answer Area

Microsoft

Statements

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.

Yes

No

Explanation:

Answer Area

Microsoft

Statements

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.

Yes

No

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: 15

You are deploying a model in Azure AI Foundry and must ensure the endpoint can handle unpredictable bursts of traffic while keeping cost low during idle periods.

Which deployment type should you choose?

- A. Provisioned Throughput Units (PTU)
- B. Standard (pay-as-you-go) deployment
- C. Batch deployment
- D. Dedicated managed compute with fixed capacity

Answer: (SHOW ANSWER)

Choose a Standard pay-as-you-go deployment . Microsoft Foundry documentation identifies Standard deployments as appropriate for low-to-medium volume workloads with high burstiness . Standard uses pay-per-token billing , so charges are based on actual inference consumption rather than continuously reserved model capacity. This makes it well suited to workloads whose traffic varies significantly and may have long idle periods.

By contrast, Provisioned Throughput Units (PTUs) reserve model-processing capacity and are designed for sustained, predictable high-throughput workloads where consistent latency is important. Microsoft states that PTUs are billed based on deployed capacity even when that capacity is not actively serving requests.

Consequently, PTU deployments can be economically inefficient when the endpoint is frequently idle or traffic is highly variable.

A Batch deployment is optimized for asynchronous, non-latency-sensitive processing and is not appropriate for an interactive endpoint receiving unpredictable live traffic. Dedicated managed compute with fixed capacity introduces continuously allocated infrastructure and therefore does not align with the requirement to minimize idle-period cost.

Microsoft also recommends Standard or Global Standard for variable, bursty traffic , while provisioned types are recommended for consistent high-volume workloads.

Study Guide references: Microsoft Foundry Models # Standard deployments; pay-per-token billing; bursty workloads; provisioned throughput; batch processing.

NEW QUESTION: 16

-

You are preparing a container image for deployment to production.

The container image build and deployment process must ensure the following:

- *The image is uniquely versioned.
- *Secure authentication is used when pushing the image to Azure Container Registry (ACR).
- *The image is stored in ACR for deployment.

You need to ensure that the container image build-and-push process meets the requirements.

Which action should you perform for each requirement? 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	ACR image management
Build the image locally.	Requirement
Push the image to ACR.	Ensure that the image is versioned.
Use a unique tag for the image.	Allow secure push to ACR.
Authenticate to ACR by using Microsoft Entra ID.	Store the image in the registry for deployment.

Answer:

Actions	ACR image management
Build the image locally.	Requirement
Push the image to ACR.	Ensure that the image is versioned.
Use a unique tag for the image.	Allow secure push to ACR.
Authenticate to ACR by using Microsoft Entra ID.	Store the image in the registry for deployment.

Explanation:

- * Ensure that the image is versioned: Use a unique tag for the image.
- * Allow secure push to ACR: Authenticate to ACR by using Microsoft Entra ID.
- * Store the image in the registry for deployment: Push the image to ACR.

For production deployments, each image should receive a unique tag so that a specific build can be identified and deployed consistently. Microsoft recommends unique tags for deployments because each pushed image receives a distinct version reference instead of reusing mutable tags such as latest. Typical unique-tagging schemes include build IDs, timestamps, or source-control identifiers. Before pushing the image, authenticate to Azure Container Registry by using Microsoft Entra ID . Microsoft recommends Azure identity-based authentication, such as through az acr login, rather than relying on long- lived registry administrator credentials. Appropriate Azure RBAC/data-plane permissions must also permit the identity to push artifacts. Finally, use a push operation such as docker push < registry > .azurecr.io/ < repository > : < tag > to upload the tagged image and its layers to ACR. Once pushed successfully, the image is stored in the registry and can be referenced by downstream deployment services. Build the image locally is not mapped to any stated requirement. Building may be necessary in some workflows, but it does not itself provide unique versioning, authentication, or storage in ACR. Study Guide references: Azure Container Registry # image tagging and versioning; Microsoft Entra authentication; pushing container images; repositories and artifacts.

Valid AI-200 Dumps shared by EduDump.com for Helping Passing AI-200 Exam! EduDump.com now offer the **newest AI-200 exam dumps**, the EduDump.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** EduDump.com AI-200 dumps with Test Engine here: <https://www.edudump.com/exams/Microsoft/AI-200/premium/> (144 Q&As Dumps, **35%OFF** Special Discount Code: **freecram**)

NEW QUESTION: 17

You have a newly provisioned Azure subscription. You are designing a custom Event Grid workflow for AI inference events.

You need to implement the Event Grid components to support routing of high-confidence events to a downstream processor.
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.

ACTIONS

Create an event subscription.

Create a partner topic.

Create a custom topic.

Create a domain.

Register the provider.

Answer Area

Answer:

1

Create an event subscription.

2

Create a partner topic.

3

Create a custom topic.

4

Create a domain.

5

Register the provider.

1

Register the provider.

2

Create a custom topic.

3

Create an event subscription.

Explanation:

Answer Area

1

Register the provider.

2

Create a custom topic.

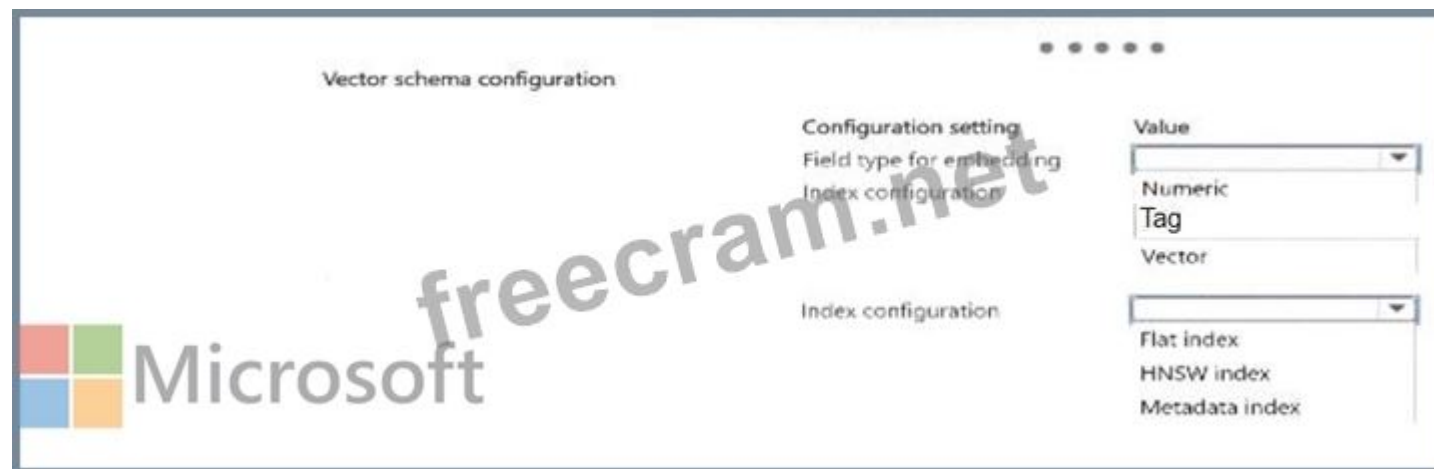
3

Create an event subscription.

Verified Answer: 1) Register the Event Grid resource provider; 2) create a custom topic; 3) create an event subscription.
Detailed Explanation: A newly provisioned subscription must have the Event Grid resource provider available before Event Grid resources can be created. The publisher needs a custom topic as the event-ingress resource, and routing to a downstream processor is then defined by an event subscription on that topic. The event subscription can include filters such as event type or data fields so that only high-confidence events reach the processor. Creating a partner topic or domain is unnecessary for the stated custom workflow.
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 | Create an Event Grid custom topic or domain

NEW QUESTION: 18

You are implementing semantic retrieval in Redis.
The solution must support low-latency, approximate nearest neighbor (ANN) vector similarity search for large-scale AI retrieval workloads.
You need to select the appropriate vector schema settings.
Which configuration values should you select? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:



NEW QUESTION: 19

You are building a semantic search feature for a chatbot. You store document embeddings in Redis. You review the following Python code that connects to Redis and stores an embedding value:

```
01 import numpy as np
02 from redis import Redis
03 embedding = numpy.array([0.11, 0.98, 0.44], dtype=np.float32).tobytes()
04 r = Redis(
05     host="myredis.redis.cache.windows.net"
06     port=6380,
07     password="accesskey",
08     ssl=True
09 )
10 r.hset("doc:1", mapping={"embedding": embedding})
11 r.expire("doc:1", 600)
```

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

NOTE: Each correct selection is worth one point.

Vector search fundamentals

Microsoft

Statement	Yes	No
The embedding is stored as a hash field.	☐	☐
The code enables similarity search on the embedding field.	☐	☐
The key will expire after 10 minutes.	☐	☐

Answer:

Vector search fundamentals

Microsoft

Statement	Yes	No
The embedding is stored as a hash field.	☒	☐
The code enables similarity search on the embedding field.	☐	☒
The key will expire after 10 minutes.	☒	☐

Explanation:

Verified Answer: Yes; No; Yes.

Detailed Explanation: `HSET` stores the binary embedding as a field in the Redis hash keyed by `doc:1`, so the first statement is true. The code does not create a RediSearch/Redis Query Engine vector index or define a vector field schema; merely storing bytes does not enable similarity search, so the second statement is false.

`EXPIRE doc:1 600` sets a 600-second lifetime, which is ten minutes, making the third statement true.

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 Managed Redis

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

Microsoft

Vector similarity search with metadata filtering

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

Query components:

- WHERE department = 'Finance'
- WHERE department ILIKE 'dept%'
- ORDER BY created_at DESC LIMIT 5
- ORDER BY embedding <=> :query_embedding LIMIT 5

Answer:

Query components

Microsoft

Vector similarity search with metadata filtering

Requirement	Query component
Filter rows to finance.	WHERE department = 'Finance'
Rank by similarity and return the top five.	ORDER BY embedding <=> :query_embedding LIMIT 5

Query components:

- WHERE department = 'Finance'
- WHERE department ILIKE 'dept%'
- ORDER BY created_at DESC LIMIT 5
- ORDER BY embedding <=> :query_embedding LIMIT 5

Explanation:



Verified Answer: Filter: `WHERE department = 'finance '` . Rank/top five: `ORDER BY embedding < = > query\_embedding LIMIT 5` .

Detailed 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: 21

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: (SHOW ANSWER)

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: 22

You are developing an Azure Function that calls external APIs by providing an access token for the API. The access token is stored in a secret named token in an Azure Key Vault named mykeyvault.

You need to ensure the Azure Function can access the token. Which value should you store in the Azure Functions app configuration?

- A. AZUREKVCONNSTR_https://mykeyvault.vault.azure.net/secrets/token/
- B. KeyVault:mykeyvault;Secret:token
- C. @Microsoft.KeyVault(SecretUri= https://mykeyvault.vault.azure.net/secrets/token/)
- D. App:Settings:Secret:mykeyvault:token

Answer: (SHOW ANSWER)

NEW QUESTION: 23

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.

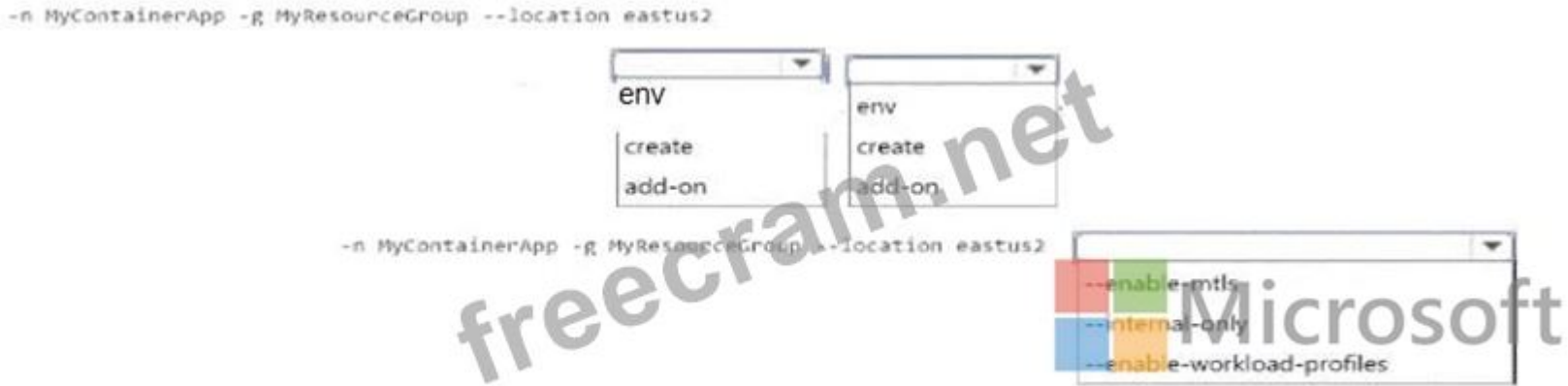
You must secure the container apps. All apps must reside in the same virtual network, share the same Dapr configuration, and share the same logging location.

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

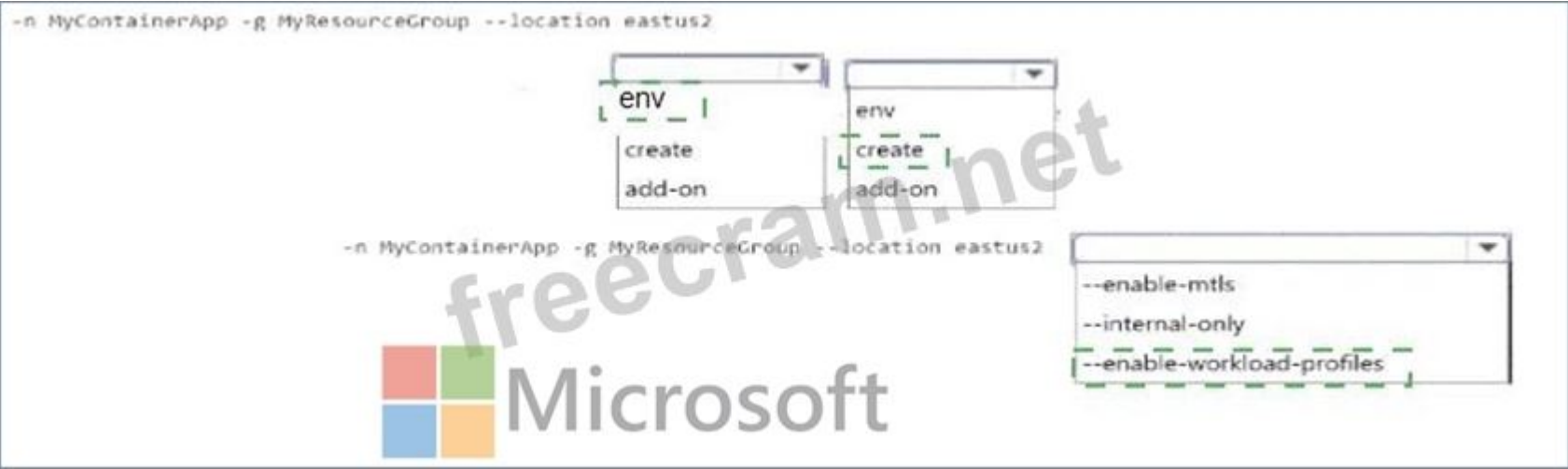
You need to configure the Azure Container App.

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.



Answer:



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.

NEW QUESTION: 24

You are creating a hazard notification system that has a single signaling server which triggers audio and visual alarms to start and stop.

You implement Azure Service Bus to publish alarms, each alarm controller uses Azure Service Bus to receive alarm signals as part of a transaction Alarm events must be recorded for audit purposes Each transaction record must include information about the alarm type that was activated.

You need to implement a reply-trail auditing solution

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

NOTE: Each correct selection is worth one point.

- A. Assign the value of the hazard message SessionId property to the ReplyToSessionId property.
- B. Assign the value of the hazard message MessageId property to the SequenceNumber property
- C. Assign the value of the hazard message SequenceNumber property to the DeliveryCount property.
- D. Assign the value of the hazard message SessionId property to the SequenceNumber property.
- E. Assign the value of the hazard message MessageId p ' operty to the DeliveryCount property.
- F. Assign the value of the hazard message MessageId property to the CorrelationId property

Answer: ([SHOW ANSWER](#))

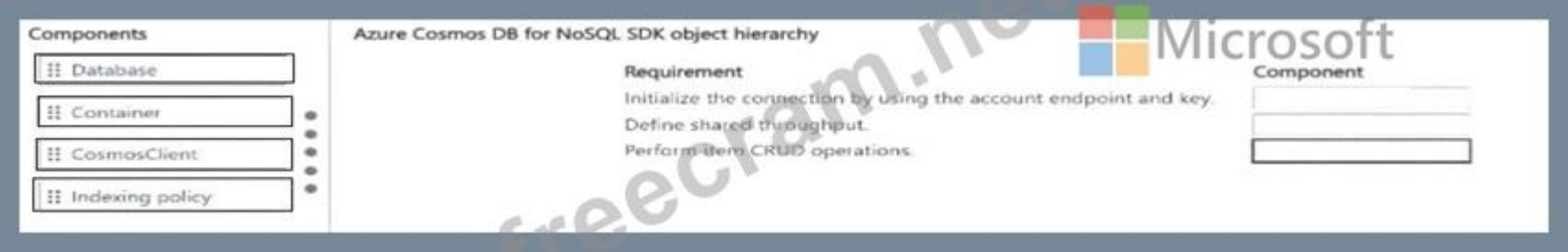
Detailed Explanation: For a multiplexed request/reply pattern, ReplyToSessionId on the request tells the responder which SessionId to place on the reply, so carrying the request SessionId into ReplyToSessionId provides the reply-session routing context. For correlation, the standard request/reply pattern copies the original request MessageId into the reply CorrelationId. SequenceNumber and DeliveryCount are broker- managed delivery metadata and are not writable application correlation fields, so they cannot be used as the proposed audit-link mechanism.

Study Guide Alignment: Azure service integration: Service Bus, Event Grid, Azure Functions triggers /bindings, and event-driven processing.

NEW QUESTION: 25

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



Answer:



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: 26

You must ensure an Azure OpenAI-powered application never exceeds a defined token-per-minute budget across all users to avoid runaway costs.

What should you configure?

- A. Content filtering severity levels
- B. Deployment-level rate limits (TPM/RPM) in Azure AI Foundry
- C. A vector index size limit in Azure AI Search
- D. Model temperature

Answer: (SHOW ANSWER)

Configure deployment-level Tokens-per-Minute (TPM) and Requests-per-Minute (RPM) rate limits for the Azure OpenAI model deployment. Azure OpenAI quota is allocated to individual model deployments in TPM units, and Microsoft documents that the TPM assigned to a deployment directly maps to the rate limit enforced for inference requests against that deployment. An associated RPM limit is also enforced according to the model ' s TPM-to-RPM ratio.

Because all application users invoke the same deployment, this enforcement occurs at the deployment boundary rather than independently for each end user. Once request traffic would exceed the configured capacity, Azure applies throttling rather than allowing unrestricted token consumption. Microsoft Foundry additionally supports explicit token-management controls through AI Gateway, where a TPM limit can be set for a model deployment; requests exceeding the limit receive HTTP 429 Too Many Requests responses.

Content filtering governs safety categories, not consumption. An Azure AI Search vector-index limit controls search storage rather than generated tokens. Temperature changes sampling randomness and has no enforcement role in rate or cost control.

Study Guide references: Azure OpenAI quota management; TPM/RPM rate limits; deployment capacity; throttling; Foundry AI Gateway token management.

NEW QUESTION: 27

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. Choose two.

NOTE: Each correct selection is worth one point.

- A. Eventual
- B. Session

- C. Strong
- D. Bounded staleness

Answer: (SHOW ANSWER)

NEW QUESTION: 28

You are developing a Retrieval-Augmented Generation (RAG) solution for a company. AI responses and embedding vectors are cached in Redis. The solution must meet the following requirements.

- * AI responses must expire exactly 24 hours after they are cached
- * Cached embeddings must always reflect the current source data.

You need to configure Redis to meet the requirements.

Cache expiration and invalidation strategy

Requirement

AI responses must expire exactly 24 hours after they are cached.

Cached embeddings must always reflect the current source data.

Action

Configure allkeys-lru eviction.

Reset expiration on every read.

Set a time to Live (TTL) on each key.

Configure volatile-lru eviction.

Delete related keys when the document changes

Increase maximum memory allocation.

Answer:

Cache expiration and invalidation strategy

Requirement

AI responses must expire exactly 24 hours after they are cached.

Cached embeddings must always reflect the current source data.

Action

Configure allkeys-lru eviction.

Reset expiration on every read.

Set a time to Live (TTL) on each key.

Configure volatile-lru eviction.

Delete related keys when the document changes

Increase maximum memory allocation.

Explanation:

Cache expiration and invalidation strategy

Requirement

AI responses must expire exactly 24 hours after they are cached.

Cached embeddings must always reflect the current source data.

Action

Set a Time to Live (TTL) on each key.

Delete related keys when the document changes.

I response expiry: set a 24-hour TTL on each key. Embedding freshness: delete related cache keys when the source document changes.

Detailed Explanation: A fixed TTL encodes the requirement that an AI response must expire after a defined period rather than survive indefinitely or have its lifetime extended on every read. Embedding freshness is a separate consistency requirement: when the authoritative document changes, any cached representation derived from the old document must be invalidated immediately. Memory eviction policies are capacity- management mechanisms and cannot guarantee a precise 24-hour lifetime or source-data consistency.

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

NEW QUESTION: 29

You maintain multiple versions of a container image in Azure Container Registry.

The production deployment must always run the exact same image build even if tags are changed later.

You need to ensure predictable and immutable image selection during deployment.

What should you do?

A. Tag the image as production and deploy it by using the production tag.

B. Schedule nightly rebuilds of the image.

C. Configure deployment to use the latest tag.

D. Identify the image by using its SHA digest.

Answer: (SHOW ANSWER)

Use the image 's SHA-256 manifest digest when defining the production deployment. Azure Container Registry assigns every pushed image manifest a unique digest, and Microsoft explicitly states that pulling an image by digest guarantees the image version being retrieved , even if an identically named tag is later pushed to a different image. A digest reference has the form myregistry.azurecr.io/repository@sha256:< digest > .

Tags such as production or latest are mutable references . By default, a user or pipeline with sufficient permissions can push a different image under the same tag. Microsoft therefore warns against relying on reusable stable tags for production deployments when exact image reproducibility is required.

A scheduled rebuild also creates a new image artifact and therefore cannot guarantee that production executes the original build. In contrast, the manifest digest is content-addressed and resolves to the precise image manifest selected at deployment time.

Thus, for deterministic and immutable production image selection, reference the container image by its SHA digest rather than by a mutable tag .

Study Guide references: Azure Container Registry # image manifests and digests; image addressing; tagging

/versioning recommendations; immutable deployment references.

NEW QUESTION: 30

You need to translate real-time spoken customer conversations from English to Spanish text during a support call, with minimal latency.

Which Azure AI service should you use?

A. Azure AI Translator (text translation REST API)

B. Azure AI Speech (Speech Translation)

C. Azure AI Language (Text Analytics)

D. Azure AI Document Intelligence

Answer: (SHOW ANSWER)

Use Azure AI Speech with Speech Translation . Microsoft documents Speech Translation as a real-time capability that accepts a live audio stream in a source language and returns translated text, synthesized speech, or both in one or more target languages. This matches the scenario precisely: the input is spoken English during a live support call, the required output is Spanish text, and latency must be minimal.

Speech Translation can be implemented through the Speech SDK , which returns intermediate recognition and translation results as speech is detected. That streaming behavior is materially better suited to live conversations than a workflow that first transcribes audio and then separately calls a text-only translation API.

Microsoft specifically positions Speech Translation for low-latency, real-time speech-to-text and speech-to- speech translation scenarios.

Azure AI Translator translates text and therefore would require a separate speech-to-text stage before translation. Azure AI Language provides NLP capabilities such as classification, sentiment, and entity extraction rather than live speech translation. Azure AI Document Intelligence extracts structured information from documents and forms and is unrelated to streaming conversational audio.

Therefore, Azure AI Speech (Speech Translation) is the correct choice.

Study Guide references: Azure AI Speech # Speech Translation; real-time transcription and translation; Speech SDK; streaming audio translation; target-language text output.

NEW QUESTION: 31

You are training a Language Understanding model for a user support system.
You create the first intent named GetContactDetails and add 200 examples.
You need to decrease the likelihood of a false positive.
What should you do?

- A. Enable active learning.
- B. Add a machine learned entity.
- C. Add additional examples to the GetContactDetails intent.
- D. Add examples to the None intent.

Answer: (SHOW ANSWER)

To reduce false-positive intent predictions , add representative utterances to the None intent . The None intent exists specifically for user utterances that should not map to any defined business intent. Microsoft recommends adding examples that resemble potential false positives so that the model learns a stronger decision boundary between valid intent utterances and unrelated or ambiguous input. For example, if the GetContactDetails intent contains requests such as "give me the customer phone number," the None intent should contain similar-looking but semantically unrelated phrases that could otherwise be incorrectly classified as GetContactDetails. During prediction, an utterance can be classified as None when it resembles None-training examples or when the highest intent score falls below the configured None threshold. Microsoft specifically advises adding false-positive examples to the None intent to improve intent discrimination. Adding more examples to GetContactDetails alone generally strengthens recognition of that intent but does not provide the model with enough negative examples. A machine-learned entity addresses entity extraction rather than intent classification. Active learning helps identify uncertain utterances for review, but it is not the direct corrective action requested. Study Guide references: Azure AI Language # Conversational Language Understanding; intents; None intent; intent classification; reducing false positives.

Valid AI-200 Dumps shared by EduDump.com for Helping Passing AI-200 Exam! EduDump.com now offer the **newest AI-200 exam dumps**, the EduDump.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** EduDump.com AI-200 dumps with Test Engine here: <https://www.edudump.com/exams/Microsoft/AI-200/premium/> (144 Q&As Dumps, **35%OFF** Special Discount Code: **freecram**)

NEW QUESTION: 32

You are reviewing secret access patterns used by an AI application that retrieves credentials from Key Vault.
You need to evaluate the security impact of each implementation approach.
What is the outcome of each approach? To answer, move the appropriate outcomes to the correct implementations. You may use each outcome 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.

Outcomes

Prevents credential storage

Supports automatic rotation

Introduces credential exposure

Requires manual update after rotation

Secret access patterns and security outcomes

Implementation

Retrieve secret without specifying version.

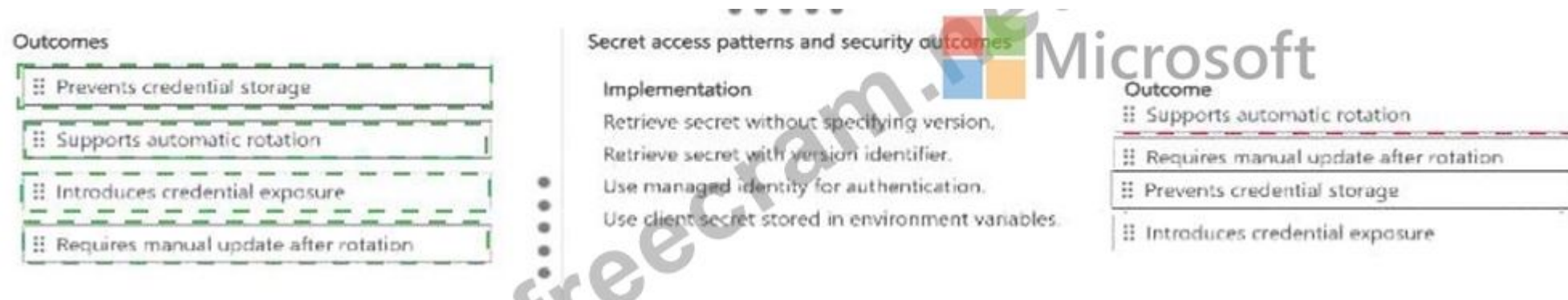
Retrieve secret with version identifier.

Use managed identity for authentication.

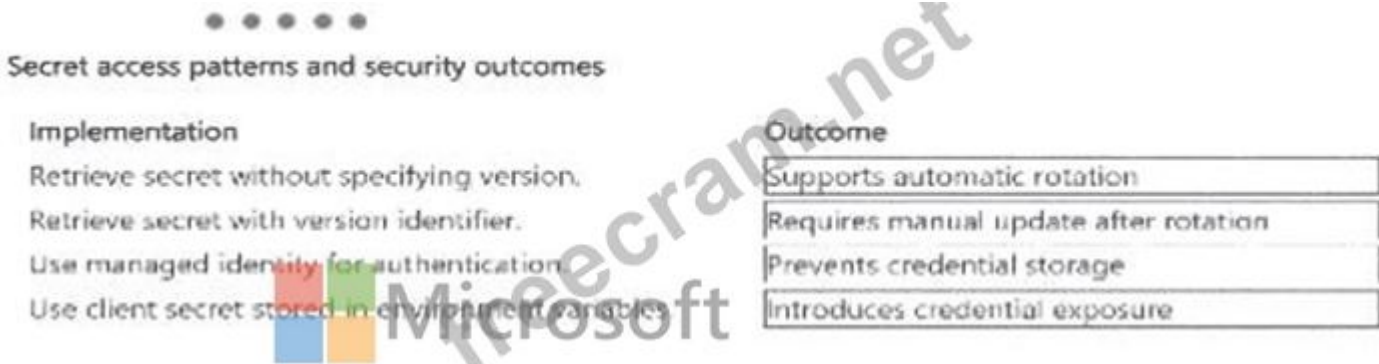
Use client secret stored in environment variables.

Outcome

Answer:



Explanation:



NEW QUESTION: 33

You are developing a solution that will use a multi-partitioned Azure Cosmos DB database. You plan to use the latest Azure Cosmos DB SDK for development. The solution must meet the following requirements:

- * Send insert and update operations to an Azure Blob storage account
- * Process changes to all partitions immediately

Allow parallelization of change process.

You need to process the Azure Cosmos DB operations.

What are two possible ways to achieve this goal? Each correct answer presents a complete solution.

NOTE: Each correct selection is worth one point.

- A.** Create an Azure App Service API and implement the change feed estimator of the SDK. Scale the API by using multiple Azure App Service instances.
- B.** Create a background job in an Azure Kubernetes Service and implement the change feed feature of the SDK.
- C.** Create an Azure Function to use a trigger for Azure Cosmos DB. Configure the trigger to connect to the container.
- D.** Create an Azure Function that uses a FeedIterator object that processes the change feed by using the pull model on the container. Use a FeedRange object to parallelize the processing of the change feed across multiple functions.

Answer: ([SHOW ANSWER](#))

NEW QUESTION: 34

You need to deploy Azure Function resources and apps to meet the business and technical requirements.

What should you use?

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

Answer: (SHOW ANSWER)

Use GitHub Actions because Proseware requires an automated, version-controlled deployment pipeline and explicitly prohibits local and command-line deployments. Microsoft documents GitHub Actions as a supported CI/CD mechanism for Azure Functions: a workflow stored under .github/workflows can build and deploy Function app code automatically in response to repository events such as pushes or pull requests. The Azure Functions deployment action can deploy the application package directly to the target Function app.

GitHub Actions also satisfies the infrastructure requirement. Microsoft provides first-class support for deploying Bicep templates through GitHub Actions , enabling Azure resources to be provisioned declaratively from the same source-controlled pipeline. A pull-request workflow can perform validation or Bicep what-if analysis before changes are merged, while a main-branch workflow can perform the actual deployment after approval. This provides repeatability, auditing, and separation from individual developer machines.

Azure Functions Core Tools and Azure CLI are commonly used for interactive/local deployments, which directly conflicts with the requirement to eliminate command-line deployment. Local Git deployment also fails the requirement for a centrally automated Bicep-based deployment process.

Study Guide references: Azure Functions CI/CD; GitHub Actions; infrastructure as code with Bicep; automated Azure deployments; version-controlled deployment workflows.

Category Breakdown

Category Number of Questions

- Connect to and consume Azure services 41
- Develop AI solutions by using Azure data management services 39
- Secure, monitor, troubleshoot Azure solutions 34
- Develop containerized solutions on Azure 28
- TOTAL 142

Exam Topic Breakdown

Exam Topic Number of Questions

NEW QUESTION: 35

You are implementing an Azure solution that uses Azure Cosmos DB and the latest Azure Cosmos DB SDK.

You add a change feed processor to a new container instance.

You attempt to read a batch of 100 documents. The process fails when reading one of the documents. The solution must monitor the progress of the change feed processor instance on the new container as the change feed is read. You must prevent the change feed processor from retrying the entire batch when one document cannot be read.

You need to implement the change feed processor to read the documents.

Which features should you use? To answer, move the appropriate features to the correct requirements. Each feature may be used 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.

Features

Lease container

Deployment unit

Dead-letter queue

Change feed estim

Answer Area

Requirement

Monitor the progress of the change feed processor.

Prevent the change feed processor from retrying the entire batch when one document cannot be read.

Feature

Feature

Feature



Answer:

Features

- Lease container
- Deployment unit
- Dead-letter queue
- Change feed estim

Answer Area

Requirement

Monitor the progress of the change feed processor.

Prevent the change feed processor from retrying the entire batch when one document cannot be read.

Feature

- Change feed estim
- Dead-letter queue

Explanation:

Features

- Lease container
- Deployment unit
- Dead-letter queue
- Change feed estim

Answer Area

Requirement

Monitor the progress of the change feed processor.

Prevent the change feed processor from retrying the entire batch when one document cannot be read.

Feature

- Change feed estimator
- Dead-letter queue

NEW QUESTION: 36

You are developing a solution that uses several Azure Service Bus queues. You create an Azure Event Grid subscription for the Azure Service Bus namespace. You use Azure Functions as subscribers to process the messages.

You need to ensure that events can be sent to Azure Event Grid from the queues. The solution must use the principle of least privilege and minimize costs.

Which Azure Service Bus role and tier level should you use? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Answer Area

Configuration	Value
Tier level	<div>Basic</div> <div>Standard</div> <div>Premium</div>
Role	<div>Azure Service Bus Data Receiver</div> <div>Azure Service Bus Data Sender</div> <div>Azure Service Bus Data Owner</div>

Answer:

Answer Area



Configuration	Value
Tier level	<div>Basic</div> <div>Standard</div> <div>Premium</div>
Role	<div>Azure Service Bus Data Receiver</div> <div>Azure Service Bus Data Sender</div> <div>Azure Service Bus Data Owner</div>

Explanation:

Answer Area

Configuration	Value
Tier level	Premium
Role	Azure Service Bus Data Receiver



Verified Answer: Tier: Premium. Role: Azure Service Bus Data Receiver.

Detailed Explanation: Microsoft documents Event Grid integration for Service Bus queues and topics as requiring a Service Bus Premium namespace. The Azure Function subscriber must then receive messages from the queue; the least-privilege built-in data-plane role for that purpose is Azure Service Bus Data Receiver. Data Sender does not permit receiving, and Data Owner is broader than necessary. Premium therefore satisfies the feature prerequisite, while Data Receiver satisfies the stated least-privilege requirement.

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 to Event Grid integration | Authenticate Service Bus with Microsoft Entra ID

NEW QUESTION: 37

Your solution must answer questions about numeric data in large Excel-based reports, for example:

" What was Q3 revenue for the West region? "

Which approach is most appropriate?

- A. Store the Excel data as unstructured text and embed it for vector search
- B. Convert data to a structured table/SQL source and use a function-calling agent to query it
- C. Rely on the model ' s parametric knowledge
- D. Use Azure AI Vision to read the spreadsheet as an image

Answer: (SHOW ANSWER)

For precise questions over tabular numeric data, convert the workbook data into a structured relational/table representation and expose controlled query functionality to the AI application through function calling. This preserves columns such as quarter, region, revenue, product, and fiscal period as typed fields that can be filtered, aggregated, and compared deterministically. Microsoft demonstrates natural-language-to-SQL patterns in which a user ' s question is translated into a query against structured PostgreSQL data, while Foundry function calling allows a model or agent to invoke application-defined query tools with structured parameters.

This is materially more reliable than embedding spreadsheet rows as unstructured text. Vector search is optimized for semantic similarity , not exact arithmetic, grouping, filtering, or aggregation. A query such as " Q3 revenue for the West region " is fundamentally a structured lookup or aggregation problem. SQL can express those constraints explicitly and return the authoritative numeric result.

Microsoft Foundry also supports agents working with structured data and provides examples where agent tools analyze numeric files and return computed values.

Relying on parametric model knowledge cannot provide current organization-specific financial figures. Azure AI Vision could OCR an image of a spreadsheet, but converting an already structured workbook into an image would destroy useful schema and increase extraction error risk.

Study Guide references: Microsoft Foundry agents; function calling; structured data access; natural- language-to-SQL; grounding AI responses in enterprise data.

NEW QUESTION: 38

An AI application retrieves configuration values from App Configuration. The application uses static configuration.

You need to implement a solution that supports dynamic configuration updates and minimizes latency for application requests.

What should you do?

- A. Inject configuration values in environment variables.
- B. Cache configuration values with a refresh interval.
- C. Retrieve configuration values from App Configuration on each request.
- D. Store configuration values in Key Vault.

Answer: ([SHOW ANSWER](#))

Use the Azure App Configuration provider 's caching and dynamic refresh mechanism with a configured refresh interval . The application loads configuration into a local cache and periodically checks App Configuration for changes. When a registered key changes, the provider refreshes the cached configuration without requiring an application restart. Microsoft documents `SetRefreshInterval` and equivalent provider settings as the mechanism for controlling the minimum time between configuration-refresh checks.

This approach satisfies both requirements. The configuration is dynamic because updated values can be detected and reloaded at runtime, while request latency remains low because normal application requests read configuration from the local cached copy instead of making a remote App Configuration request every time.

For request-driven refresh, Microsoft notes that refresh occurs asynchronously and does not block the incoming request. Before the configured refresh interval expires, additional refresh calls effectively perform no remote operation.

Microsoft also recommends caching App Configuration values instead of loading them whenever they are used, both to reduce latency and to avoid excessive service requests, throttling, and unnecessary charges.

Environment variables remain static unless the application is restarted or redeployed. Retrieving configuration on every request increases latency and service load. Key Vault is primarily intended for secrets, not general dynamic application configuration.

Study Guide references: Azure App Configuration; dynamic configuration refresh; provider caching; refresh intervals; request-driven refresh; configuration resiliency.

NEW QUESTION: 39

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: ([SHOW ANSWER](#))

The lease container is the Azure Cosmos DB component that coordinates change feed processing across multiple processor instances. Microsoft defines the lease container as the state store used by the change feed processor to track progress and coordinate ownership of partition ranges among workers. Each lease corresponds to a portion of the change feed workload, and at any given time a lease is owned by one processor instance.

When several change feed processor instances use the same lease container and the same processor name while maintaining unique instance names, the processor distributes leases across the active instances by using an equal-distribution algorithm. If instances are added or removed, the processor dynamically redistributes leases, which provides automatic workload balancing and parallelism. This is exactly what is required when ingestion volume is high and each instance must process a distinct subset of partition ranges.

Indexing precision affects query indexing behavior, not change feed coordination. Strong consistency controls read consistency semantics and does not assign partitions to processors. Autoscale throughput can increase database RU/s capacity, but it does not coordinate which processor instance owns which change feed partition range.

Study Guide references: Azure Cosmos DB for NoSQL # Change Feed Processor; lease container; parallel processing; dynamic scaling and load balancing.

NEW QUESTION: 40

You need to configure a connection string for the partner-facing service according to the technical requirements.

What should you use?

- A. Azure Key Vault references in App Service settings
- B. GitHub secrets
- C. Azure Container Registry Helm chart package
- D. Dockerfile ENV instructions

Answer: (SHOW ANSWER)

Detailed Explanation: Azure Key Vault references in App Service settings satisfy the requirement to keep secrets out of container images, source control, and directly stored application configuration. App Service resolves the referenced secret at runtime by using the app identity, so the application can consume the value as a normal setting without embedding credentials in the image. GitHub secrets are build/deployment secrets rather than a runtime App Service secret-delivery mechanism. Dockerfile ENV instructions would place secret material in the image configuration and violate the case requirements.

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 | Managed identities for Azure resources

Topic 2, Fabrikam Inc

Background - Fabrikam Inc. is a global retail analytics company that provides AI-driven demand forecasting and product recommendation services to online retailers. The company is modernizing its solution to run entirely on Microsoft Azure. The platform ingests transaction data, generates embeddings for semantic retrieval, performs vector similarity search, and returns product recommendations through containerized microservices. Developers use Python and Azure SDKs. Operations teams manage container orchestration, scaling, monitoring, and security. The solution must meet strict performance, scalability, and security requirements. Current environment - Application architecture - The Recommendation engine is a customer-facing HTTP API running as a containerized Python application. The engine is deployed to Azure Container Apps (ACA). Embeddings are stored in Azure Database for PostgreSQL by using pgvector. Semantic retrieval uses metadata filtering combined with vector similarity search. Azure Managed Redis is used as a caching layer. Front-end and API workloads are deployed to Azure Container Apps (ACA). Batch model retraining workloads run in Azure Kubernetes Service (AKS). Container and CI/CD - Container images are stored in Azure Container Registry (ACR). CI/CD uses ACR Tasks to build images on commit. ACA environments support revision management. AKS workloads are deployed by using Kubernetes manifest files stored in Git.

Monitoring - Logs are collected in Azure Monitor. Teams inspect container logs and Kubernetes events when troubleshooting. Developers write KQL queries to analyze latency spikes. Business requirements - Customer experience: Maintain a seamless, low-latency recommendation experience for end-users, even during unpredictable seasonal traffic spikes. Operational cost efficiency: Minimize compute expenditures by deallocating resources during periods of inactivity and by preventing runaway scaling costs. Data integrity and freshness: Ensure that product recommendations always reflect the most current catalog metadata and pricing to prevent customer dissatisfaction. Security and compliance: Adhere to a Zero Trust security model by eliminating long-lived credentials and centralizing the management of all sensitive secrets. Global scalability: Support the rapid ingestion of millions of new product embeddings daily without degrading query performance for existing retailers. Technical requirements - Performance: Semantic search latency must remain under 200 milliseconds at peak load. Database optimization: Use pgvector for embeddings and implement metadata filtering to reduce compute overhead. Configure compute and memory appropriately for vector workloads to ensure high-dimensional index residency in RAM and efficient mathematical throughput.

Vector similarity calculations must be performed only against products that satisfy mandatory metadata constraints. Database performance: Database connections must support high concurrency with minimal latency through the implementation of connection optimization. Data load strategy: To ensure maximum ingestion throughput, secondary indexes must be applied only after bulk loading of embeddings is complete.

Caching: Redis cache entries must expire automatically after 10 minutes. Implement a reactive mechanism to invalidate cache entries upon metadata updates. Identity: Use managed identities for all service-to-service and service-to-database authentication. Plain-text credentials in configuration files are strictly prohibited. Secret management: All secrets must be stored centrally. Secrets must be rotated automatically by using a centralized lifecycle policy. Scaling: Use Kubernetes event-driven autoscaling (KEDA) for event-driven scaling. The Recommendation API must scale based on HTTP traffic, while batch jobs must scale based on queue length and support scale-to-zero. CI/CD: All images must be stored in Azure Container Registry. Use ACR Tasks to automate image builds triggered by source code commits. Monitoring: Use KQL to analyze performance telemetry and troubleshoot microservice connectivity failures. Inspect logs and events when troubleshooting AKS and ACA.

NEW QUESTION: 41

You need to configure vector embedding updates according to the business and technical requirements.

Which information should you use? To answer, select the appropriate option in the answer area.

NOTE: Each correct selection is worth one point.

Requirement

Identify document changes that should trigger vectorization.

Scale out the vectorization processing.

Configuration

Change feed processor

Periodic full container scan

Cross-partition SELECT query

Lease container

Strong consistency level

RU throughput on the container

Answer:

Requirement

Identify document changes that should trigger vectorization.

Scale out the vectorization processing.

Configuration

Change feed processor

Periodic full container scan

Cross-partition SELECT query

Lease container

Strong consistency level

RU throughput on the container

Explanation:

Update vector embeddings

Requirement

Identify document changes that should trigger vectorization.

Scale out the vectorization processing.

Configuration

Change feed processor

Lease container

Verified Answer: Identify document changes: Change feed processor. Scale out vectorization processing: Lease container.

Detailed Explanation: The change feed processor is designed to react to inserts and updates in a monitored Cosmos DB container, which directly matches the requirement to generate embeddings for new or changed documents. Its lease container stores processing state and coordinates work across multiple workers, allowing the processing workload to scale out without duplicating ownership of change-feed ranges. A periodic full scan would consume unnecessary RUs, and neither strong consistency nor container RU throughput is the coordination mechanism for distributed change-feed workers.

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 Cosmos DB change feed processor

NEW QUESTION: 42

You are developing a solution for a hospital to support the following use cases:

*The most recent patient status details must be retrieved even if multiple users in different locations have updated the patient record.

*Patient health monitoring data retrieved must be the current version or the prior version.

*After a patient is discharged and all charges have been assessed, the patient billing record contains the final charges.

You provision a Cosmos DB NoSQL database and set the default consistency level for the database account to Strong. You set the value for Indexing Mode to Consistent.

You need to minimize latency and any impact to the availability of the solution. You must override the default consistency level at the query level to meet the required consistency guarantees for the scenarios.

Which consistency levels should you implement? To answer, drag the appropriate consistency levels to the correct requirements. Each consistency level 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.

Consistency levels

StrongBounded StalenessConsistent PrefixEventual

Answer Area

Return the most recent patient status.

Return health monitoring data that is no less than one version behind.

After patient is discharged and all changes are assessed, retrieve the correct billing data with the final charges

Answer:

Consistency levels

StrongBounded StalenessConsistent PrefixEventual

Answer Area

Return the most recent patient status.

Strong

Return health monitoring data that is no less than one version behind.

Bounded Staleness

After patient is discharged and all changes are assessed, retrieve the correct billing data with the final charges

Eventual

Explanation:

Return the most recent patient status.

Strong

Return health monitoring data that is no less than one version behind.

Bounded Staleness

After patient is discharged and all changes are assessed, retrieve the correct billing data with the final charges

Eventual

Strong consistency is required for the patient-status scenario because it provides linearizability : reads return the latest committed value. This is the appropriate guarantee when updates can originate from different locations and the application must never return an older patient status. Microsoft defines Strong as the highest Azure Cosmos DB consistency level and notes that it provides the strongest freshness guarantee, at the cost of greater latency and reduced availability compared with weaker models.

For health-monitoring data, Bounded Staleness is the intended consistency level. It explicitly limits how far reads can lag behind writes using a maximum number of versions (K) or a time interval (T).

Conceptually, this matches a requirement that data may be current or only a limited number of versions behind.

For finalized billing data, Eventual consistency provides the lowest latency and highest availability. Once discharge processing is complete and no further billing writes occur, replicas eventually converge on the same final value. Microsoft specifically notes that, in the absence of further writes, weaker consistency levels ultimately yield the same converged data.

Azure Cosmos DB also permits requests to relax consistency below an account's Strong default, which is exactly the optimization pattern described here.

Study Guide references: Azure Cosmos DB for NoSQL # Strong consistency; Bounded Staleness; Eventual consistency; per-request consistency override; latency/availability trade-offs.

NEW QUESTION: 43

An AI application uses a database. The database credential rotates every 30 days.

The application currently requires a manual update each time the credential rotates.

You need to ensure that the application always uses the latest secret version without manual updates.

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.** Cache the secret for 30 days.
- B.** Retrieve secrets without specifying a version identifier.
- C.** Configure a Key Vault rotation policy.
- D.** Retrieve secrets by specifying a version identifier.

Answer: ([SHOW ANSWER](#))

NEW QUESTION: 44

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

Answer: ([SHOW ANSWER](#))

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: 45

You have container source code stored in a Git repository.
The container registry must automatically build and store a new container image whenever a developer commits code to the Git repository.
You must minimize the use of an external build infrastructure.
You need to configure Azure Container Registry (ACR) to manage the build process natively and automatically.
Which two ACR components should you use? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

- A. Webhook
 - B. Quick task
 - C. Artifact Cache rule
 - D. Source-triggered task
- Answer:** (SHOW ANSWER)

Azure Container Registry Tasks provides Azure-hosted container build capabilities, eliminating the need to maintain external Docker build servers or dedicated CI build infrastructure. A Quick task uses commands such as az acr build to send the build context to ACR-managed compute, where Azure builds the container image and, by default, pushes the successfully built image directly into the registry. Microsoft describes quick tasks as cloud-based, on-demand build-and-push operations that do not require a local Docker Engine.

For automatic execution after source-code changes, use a Source-triggered task . ACR Tasks can associate a task with a GitHub or Azure Repos repository and automatically execute the image build whenever code is committed to the configured branch. Microsoft documents that when a commit occurs, the source-control trigger causes ACR Tasks to build and push the resulting image.

A Webhook is not the required standalone ACR build component here. ACR Tasks creates and manages the necessary source-repository webhook as part of configuring the source trigger. An Artifact Cache rule caches upstream container artifacts and does not build application images from Git commits.

Study Guide references: Azure Container Registry Tasks # Quick tasks; Automatically triggered tasks; Source-code commit triggers; cloud-based image builds.

NEW QUESTION: 46

You need to deploy a batch retraining workload.
How should you complete the scaling configuration? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Scale with **Kubernetes Event-driven Autoscaler** in Azure Kubernetes Service

Requirement	Configuration
Trigger scaling based on queue depth.	<div>CPU utilization scaler</div> <div>Azure Service Bus scaler</div> <div>HTTP concurrency scaler</div>
Deallocate instances when the message queue is empty.	<div>Minimum replicas = 0</div> <div>Minimum replicas = 1</div>
Prevent uncontrolled burst scaling.	<div>Maximum replicas = 10</div> <div>Allow the system to determine the maximum replicas</div>

Answer:

Scale with Kubernetes Event-driven Autoscaler in Azure Kubernetes Service

Requirement

Trigger scaling based on queue depth.

Configuration

The screenshot shows the KEDA configuration interface with three requirements and their settings:

- Trigger scaling based on queue depth:** The configuration dropdown is set to "Azure Service Bus scaler".
- Deallocate instances when the message queue is empty:** The configuration dropdown is set to "Minimum replicas = 0".
- Prevent uncontrolled burst scaling:** The configuration dropdown is set to "Maximum replicas = 10".

Explanation:

- * Trigger scaling based on queue depth: Azure Service Bus scaler
- * Deallocate instances when the message queue is empty: Minimum replicas = 0
- * Prevent uncontrolled burst scaling: Maximum replicas = 10

The batch retraining workload must scale according to queue length, so the correct KEDA trigger is the Azure Service Bus scaler. KEDA integrates with Azure Service Bus and monitors queue or subscription metrics, allowing AKS workloads to scale according to the number of pending messages rather than CPU utilization or HTTP request concurrency. Microsoft documents KEDA as the event-driven autoscaling mechanism for AKS and specifically supports Azure-based scalers for queue-driven workloads.

To meet the requirement to deallocate workload instances when no messages remain, configure minimum replicas = 0. KEDA supports scale-to-zero for event-driven workloads, which is particularly appropriate for intermittent batch or machine-learning processing because no pods need to remain running while the queue is empty. Microsoft specifically identifies scale-to-zero as a KEDA capability and recommends it for sporadic ML workloads where cost optimization is important.

Finally, configure maximum replicas = 10. A finite maximum replica count establishes an explicit upper scaling boundary, preventing large queue bursts from creating an uncontrolled number of worker replicas and protecting both infrastructure cost and downstream services.

Study Guide references: AKS # KEDA; Azure Service Bus event-driven scaling; minReplicaCount; maxReplicaCount; scale-to-zero; autoscaling limits.

Valid AI-200 Dumps shared by EduDump.com for Helping Passing AI-200 Exam! EduDump.com now offer the **newest AI-200 exam dumps**, the EduDump.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** EduDump.com AI-200 dumps with Test Engine here: <https://www.edudump.com/exams/Microsoft/AI-200/premium/> (144 Q&As Dumps, **35%OFF** Special Discount Code: **freecram**)

NEW QUESTION: 47

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: (SHOW ANSWER)

Detailed 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: 48

An ACA app processes messages from an Azure Storage queue.

The app must scale automatically based on messages in a specific Azure Storage queue by using a Kubernetes Event-driven Autoscaler (KEDA) custom scale rule. You need to configure the required scale rule values.

Which two values should you configure? Each correct answer presents part of the solution. Choose two.

NOTE: Each correct selection is worth one point.

A. Polling interval

B. Trigger type

C. Maximum replicas

D. Queue name

Answer: (SHOW ANSWER)

Detailed Explanation: A KEDA custom scale rule must identify what scaler to use and which resource within that scaler should be observed. For Azure Storage queues, the trigger type selects the Storage Queue scaler and the queue name identifies the specific queue whose message count drives scaling. Polling interval and maximum replicas are general scaling controls, but they do not define the required queue-specific trigger.

Therefore trigger type and queue name are the essential rule values tested here.

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 | Set scaling rules in Azure Container Apps

NEW QUESTION: 49

You need to ensure that responses from your Azure OpenAI application include citations back to the specific source documents used, to support user trust and verification.

What should you implement?

A. Increase top_p to broaden token sampling

B. Configure the RAG pipeline to return retrieved document metadata (source, page) alongside generated answers and instruct the model to cite them

C. Use only the base model with no retrieval

D. Enable higher content filter severity

Answer: (SHOW ANSWER)

A retrieval-augmented generation (RAG) architecture is the correct mechanism when generated answers must be traceable to specific source documents. RAG first retrieves relevant document chunks from a search index or other knowledge source, supplies those chunks to the language model as grounding context, and then generates an answer based on that retrieved information. Microsoft specifically documents that RAG enables responses containing citations back to source content .

For reliable citation generation, each retrieved chunk should carry identifying metadata such as a source identifier, document title, section, page, or URL . Microsoft's Azure RAG guidance recommends labeling retrieved chunks and including source metadata with them, then explicitly instructing the model to cite the corresponding sources in its response. This allows users to verify individual claims against the underlying documents.

Modern Azure AI Search agentic retrieval can also return structured references and grounding metadata , specifically designed for citation linking.

Increasing top_p only alters token sampling. Content filtering addresses safety, not provenance. Using a base model without retrieval provides no authoritative document linkage.

Study Guide references: RAG architecture; grounding data; retrieval metadata; Azure AI Search references; prompt engineering for citations.

NEW QUESTION: 50

You are developing an application that must extract structured field data (invoice number, total, vendor name) from scanned invoices in multiple layouts.

You need a solution that requires no custom model training.

What should you use?

A. Azure AI Vision Read API

B. Azure AI Document Intelligence prebuilt invoice model

C. Azure AI Language custom named entity recognition

D. Azure AI Document Intelligence custom neural model

Answer: (SHOW ANSWER)

Use the Azure AI Document Intelligence prebuilt invoice model . The prebuilt invoice model is specifically designed to analyze invoices and extract structured fields and line items without requiring an organization to train a custom model. Microsoft documents support for fields including InvoiceId , VendorName , InvoiceTotal , InvoiceDate, CustomerName, AmountDue, tax information, addresses, and invoice line items.

Because the model is pretrained to recognize common invoice structures, it can process invoices that use different layouts and formats while returning normalized structured output. This directly satisfies the requirement to extract invoice number, total, and vendor information from scanned documents while minimizing development and training effort.

The Azure AI Vision Read API performs OCR and can extract printed or handwritten text, but it does not provide invoice-specific semantic field extraction. Custom named entity recognition would require training and is designed for text entity extraction rather than document-layout understanding. A Document Intelligence custom neural model can handle organization-specific forms and layouts, but it requires custom model training, directly violating the requirement.

Therefore, the prebuilt invoice model provides the required combination of OCR, document understanding, structured extraction, and zero custom training.

Study Guide references: Azure AI Document Intelligence # prebuilt invoice model; structured field extraction; OCR; invoice fields and line items.

NEW QUESTION: 51

You plan to create a Docker image that runs an ASP.NET Core application named ContosoApp. You have a setup script named setupScript.ps1 and a series of application files including ContosoApp.dll.

You need to create a Dockerfile document that meets the following requirements:

* Call SQiupScript.ps1 when the container is built.

* Run CantosoApp.dill when the container starts.

The Dockerfile document must be created in the same folder where ContosoApp.dll and setupScnpt.ps! are stored.

Commands

```

::: RUN powershell ./setupScript.ps1
::: WORKDIR /apps/ContosoApp
::: COPY ./ .
::: FROM microsoft/aspnetcore:latest
::: CMD ["dotnet", "ContosoApp.dll"]

```

Answer Area

Answer:

Commands

```

::: RUN powershell ./setupScript.ps1
::: WORKDIR /apps/ContosoApp
::: COPY ./ .
::: FROM microsoft/aspnetcore:latest
::: CMD ["dotnet", "ContosoApp.dll"]

```

Answer Area

```

::: FROM microsoft/aspnetcore:latest
::: WORKDIR /apps/ContosoApp
::: COPY ./ .
::: RUN powershell ./setupScript.ps1
::: CMD ["dotnet", "ContosoApp.dll"]

```

Explanation:

Answer Area

- FROM microsoft/aspnetcore:latest
- WORKDIR /apps/ContosoApp
- COPY ./ .
- RUN powershell ./setupScript.ps1
- CMD ["dotnet", "ContosoApp.dll"]

NEW QUESTION: 52

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

Traffic isolation

Single revision mode

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

Answer:

Configurations

Traffic splitting

Traffic isolation

Single revision mode

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

Multiple revision mode

Traffic splitting

Explanation:

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

Multiple revision mode

Traffic splitting

oncurrent active versions: Multiple revision mode. Route a percentage to the secondary version: Traffic splitting.

Detailed 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: 53

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

Answer:

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	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 method.



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: 54

You are creating an Azure Functions app project in your local development environment by using Azure Functions Core Tools.

You must create the project in either Python or C# without using a template.

You need to specify the command and its parameter required to create the Azure Functions app project.

Which command and parameter should you specify? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Core Tools syntax elements

Syntax element	Value
Command	<div><div></div><div>func new func init func azure</div></div>
Parameter	<div><div></div><div>--language --worker-runtime --target-framework</div></div>

Answer:

Core Tools syntax elements	
Syntax element	Value
Command	<div><div></div><div>func new func init func azure</div></div>
Parameter	<div><div></div><div>--language --worker-runtime --target-framework</div></div>

Explanation:

* Command: func init

* Parameter: --worker-runtime

Use func init to initialize a new Azure Functions project. Microsoft documents func init as the Azure Functions Core Tools command that creates the project structure in either the current directory or a specified project folder. The command is specifically intended for creating a new Functions application project, whereas func new creates an individual function from a template inside an already initialized project. The --worker-runtime parameter specifies the language/runtime model for the Functions project. For example, Microsoft documents func init --worker-runtime python for Python projects and func init --worker-runtime dotnet-isolated for modern C#.NET isolated-worker projects.

--language is not the primary parameter used to establish the Functions worker runtime in this scenario. It can be relevant to certain runtime combinations, such as Node.js, but the Core Tools project initialization examples for Python and C# rely on --worker-runtime. --target-framework is used to select a specific .NET target framework after the worker/runtime model is determined; it does not identify whether the project should be Python or C#.

Therefore, the correct combination is func init + --worker-runtime .

Study Guide references: Azure Functions Core Tools # func init; local Functions project creation; --worker- runtime; Python and .NET isolated worker projects.

NEW QUESTION: 55

A large retail company operates online and physical stores. The company tracks inventory levels in real time to manage stock efficiently across all locations. You develop an Azure Event Grid solution to handle events generated by the inventory management system deployed to Azure.

You need to implement a subscription filter that dynamically adjusts to seasonal changes in product demand.

Which event filter should you use?

- A. A prefix filter on the event type field that matches the current season ' s name
- B. A static subject filter that targets events with a subject ending in " /seasonal/inventory "
- C. A subscription filter that uses a label filter to include events tagged with seasonal promotional codes
- D. An advanced filter using a Boolean condition that evaluates multiple data fields, including a season field within the event data

Answer: D ([LEAVE A REPLY](#))

NEW QUESTION: 56

You are designing Azure Functions for three different backend workloads.

Each workload requires one of the following dispatching models:

- Return an immediate response to a client.
- Process background work from a queue.
- Run code on a fixed schedule.

You need to select the trigger for each requirement.

Which triggers should you select? To answer, move the appropriate triggers to the correct requirements. You may use each trigger 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.

Triggers

HTTP trigger

Timer trigger

Event Grid trigger

Blob Storage trigger

Queue Storage trigger

Trigger-to-workload mapping

Requirement

Return an immediate response to a client.

Process background work from a queue.

Run code on a fixed schedule.

Trigger

Answer:

Triggers

HTTP trigger

Timer trigger

Event Grid trigger

Blob Storage trigger

Queue Storage trigger

Trigger-to-workload mapping

Requirement

Return an immediate response to a client.

Process background work from a queue.

Run code on a fixed schedule.

Trigger

HTTP trigger

Queue Storage trigger

Timer trigger

Explanation:

* Return an immediate response to a client: HTTP trigger

- * Process background work from a queue: Queue Storage trigger
- * Run code on a fixed schedule: Timer trigger

Use an HTTP trigger when a function must be invoked by an HTTP request and return a response directly to the calling client. Microsoft documents HTTP triggers as the standard mechanism for building serverless APIs and handling web requests or webhooks. This makes the HTTP trigger the correct fit for synchronous request- response workloads.

For asynchronous background processing, use an Azure Queue Storage trigger . A queue-triggered function runs when messages are added to an Azure Storage queue. This decouples the producer from the consumer and is appropriate for background processing where work can be buffered and processed independently of the original request. Microsoft also documents target-based scaling for Queue Storage triggers on supported Azure Functions hosting plans.

For scheduled execution, use a Timer trigger . Microsoft states that a timer trigger runs a function according to a configured schedule, typically expressed with an NCRONTAB/CRON expression. This makes it the correct choice for periodic maintenance, scheduled synchronization, reporting, or other fixed-time workloads.

Event Grid and Blob Storage triggers are event-driven mechanisms for event notifications and blob changes respectively, but neither matches these three stated dispatching requirements.

Study Guide references: Azure Functions # HTTP triggers; Queue Storage triggers; Timer triggers; event- driven and scheduled execution.

NEW QUESTION: 57

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	Configuration
Authenticate App Service to ACR.	<div>Assign the Container Registry Repository Reader role to a managed identity.</div> <div>Configure a webhook on ACR.</div> <div>Enable the admin user.</div>
Avoid storing static credentials.	<div>Embed credentials in Dockerfile.</div> <div>Store the registry password in application settings.</div> <div>Use managed identity with role assignment.</div>

Answer:

Configure secure container image pull

Requirement	Configuration
Authenticate App Service to ACR.	<div>Assign the Container Registry Repository Reader role to a managed identity.</div> <div>Configure a webhook on ACR.</div> <div>Enable the admin user.</div>
Avoid storing static credentials.	<div>Embed credentials in Dockerfile.</div> <div>Store the registry password in application settings.</div> <div>Use managed identity with role assignment.</div>

Explanation:

Configure secure container image pull



Authenticate App Service to ACR: assign the registry pull role to a managed identity. Avoid static credentials:

Configure App Service to use that managed identity for the image pull.

Detailed 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 credential-elimination 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: 58

An Azure Service Bus queue processes AI enrichment jobs.

Some messages fail repeatedly because the payload is malformed.

You need to ensure that repeatedly failing messages do NOT block valid messages and can be inspected separately.

Which message action should you perform?

- A. complete
- B. defer
- C. dead-letter
- D. abandon

Answer: ([SHOW ANSWER](#))

Use the dead-letter action. Azure Service Bus provides a dedicated dead-letter queue (DLQ) for each queue and subscription. Its purpose is to isolate messages that cannot be successfully processed, including malformed or poison messages, so that they do not continue cycling through the active queue and interfering with valid workload processing. Microsoft explicitly identifies malformed or unexpected payloads as a typical poison-message scenario suitable for the DLQ.

When an application explicitly dead-letters a message, the message is moved to the queue's dead-letter subqueue where it can be inspected, diagnosed, corrected, or resubmitted later. The dead-letter operation can also include a reason and error description, which improves operational troubleshooting.

Complete removes a successfully processed message permanently, so it is inappropriate for a failed message that must be retained. Abandon releases the message lock and makes the message available for redelivery, which would perpetuate repeated failures. Defer postpones processing but leaves the message associated with the main entity and requires its sequence number for later retrieval; it is not the standard mechanism for isolating poison messages.

Therefore, the correct action is dead-letter.

Study Guide references: Azure Service Bus # message settlement; dead-letter queues; poison-message handling; DeadLetter operation.

NEW QUESTION: 59

You are designing a solution that will use two Azure Functions apps: App1 and App2. App1 is Windows based and will be deployed as code. App2 is Linux based and will be deployed as a container image.

Estimates show that the duration of the request processing for both apps will range from 1 to 10 minutes.

You plan to implement App1 and App2 by using the hosting plan to satisfy the following requirements:

- * Request processing can complete within the estimated time range.
- * The autoscaling behavior is event driven.

* The upper scaling limit is maximized.
You need to create the hosting plan for the implementation.
Which hosting plan should you create? To answer, move the appropriate hosting plans to the correct apps.
You may use each hosting plan once, more than once, or not at all. You may..



Answer:



Explanation:
pp1: Consumption. App2: Premium.
Detailed Explanation: For App1, the Windows code-based Consumption plan satisfies event-driven autoscaling, allows execution up to the stated ten-minute range, and has the highest scaling ceiling among the listed event-driven choices for this workload. App2 is deployed as a Linux container image; among Premium, Dedicated, and Consumption, Premium is the event-driven plan that supports Linux containerized Functions and permits long-running execution. Dedicated hosting is not the requested event-driven autoscale model. The original Premium/Premium answer therefore did not maximize App1's upper scaling limit.
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 scale and hosting

NEW QUESTION: 60

You need to configure the database resources for the Azure Database for PostgreSQL instance.
How should you complete the configuration to meet the business and technical requirements? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

PostgreSQL compute, memory, and storage configuration for vector workloads



Requirement

Meet the 200ms semantic search latency requirement.

Optimize the environment for high-dimensional pgvector index residency.

Support the continuous ingestion of transaction-based embeddings.

Configuration

Increase compute vCores.
Increase max_connections.
Increase memory allocation.

Enable read replica.
Increase max_connections.
Increase memory allocation.

Enable storage autoscale.
Increase compute vCores.
Increase backup retention.

Answer:

PostgreSQL compute, memory, and storage configuration for vector workloads

Requirement	Configuration
Meet the 200ms semantic search latency requirement.	<div> <div></div> <div> Increase compute vCores. Increase max_connections. Increase memory allocation. </div> </div>
Optimize the environment for high-dimensional pgvector index residency.	<div> <div></div> <div> Enable read replica. Increase max_connections. Increase memory allocation. </div> </div>
Support the continuous ingestion of transaction-based embeddings.	<div> <div></div> <div> Enable storage autoscale. Increase compute vCores. Increase backup retention. </div> </div>



Microsoft

Explanation:

- * Meet the 200-ms semantic search latency requirement: Increase compute vCores.
- * Optimize the environment for high-dimensional pgvector index residency: Increase memory allocation.
- * Support the continuous ingestion of transaction-based embeddings: Enable storage autoscale.

For the strict sub-200-ms vector-search latency target, increasing compute vCores is the appropriate choice.

Vector similarity operations are computationally intensive, and additional CPU capacity improves mathematical throughput and parallel query execution. Microsoft's pgvector guidance emphasizes query-plan optimization, ANN indexes such as HNSW, and sufficient compute resources when optimizing vector workloads.

For high-dimensional pgvector index residency , increase memory allocation . HNSW provides strong query-performance characteristics but consumes more memory than IVFFlat. Keeping frequently accessed vector index structures in memory minimizes disk access and materially improves latency. Microsoft explicitly notes that HNSW requires more memory while providing a better speed/recall tradeoff.

For the continuous ingestion of millions of embeddings, enable storage autoscale . Azure Database for PostgreSQL Flexible Server can automatically increase allocated storage as capacity approaches configured thresholds, avoiding an out-of-storage condition as data volumes grow. Microsoft recommends storage autogrow for workloads whose storage demand can increase dynamically.

Increasing max_connections does not directly improve vector computation or index residency, read replicas primarily scale reads, and backup retention does not address ingestion capacity.

Study Guide references: Azure Database for PostgreSQL Flexible Server # pgvector performance optimization; compute and memory sizing; HNSW indexing; storage autogrow.

NEW QUESTION: 61

You deploy a Linux container image to App Service.

The container requires the following environment variables at runtime:

* A non-sensitive configuration value named MODEL_VERSION

* A database password that must remain secure

You need to configure App Service to provide these environment variables at runtime.

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

NOTE: Each correct selection is worth one point.

Requirement	Configuration
MODEL VERSION	<div>App setting with plain text value</div> <div>Connection string</div> <div>Dockerfile ENV instruction</div>
Database password	<div>App setting with plain text value</div> <div>Connection string with embedded password</div> <div>Key vault reference syntax</div>

Answer:

Requirement	Configuration
MODEL VERSION	<div>App setting with plain text value</div> <div>Connection string</div> <div>Dockerfile ENV instruction</div>
Database password	<div>App setting with plain text value</div> <div>Connection string with embedded password</div> <div>Key vault reference syntax</div>

Explanation:

Container configuration settings

Requirement	Configuration
MODEL VERSION	<div>App setting with plain text value</div>
Database password	<div>Key Vault reference syntax</div>

Valid AI-200 Dumps shared by EduDump.com for Helping Passing AI-200 Exam! EduDump.com now offer the **newest AI-200 exam dumps**, the EduDump.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** EduDump.com AI-200 dumps with Test Engine here: <https://www.edudump.com/exams/Microsoft/AI-200/premium/> (144 Q&As Dumps, **35%OFF** Special Discount Code: **freecram**)

NEW QUESTION: 62

A python web API uses OpenTelemetry for tracing. The call downstream service function makes an outbound HTTP request by using the requests library. The following code is the only OpenTelemetry configuration in the application:

```
01 from opentelemetry import trace
02 tracer = trace.get_tracer(__name__)

03 def process_request():
04     with tracer.start_as_current_span("process_request"):
05         call_downstream_service()
```

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

Identifying missing context propagation for distributed trace correlation

Statement	Yes	No
The code creates a span for each request.	☐	☐
The code automatically propagates trace context to downstream HTTP services.	☐	☐
The solution requires additional instrumentation for end-to-end trace correlation.	☐	☐

Answer:

Identifying missing context propagation for distributed trace correlation

Statement	Yes	No
The code creates a span for each request.	☒	☐
The code automatically propagates trace context to downstream HTTP services.	☐	☒
The solution requires additional instrumentation for end-to-end trace correlation.	☒	☐

Explanation:

Identifying missing context propagation for distributed trace correlation

Statement	Yes	No
The code creates a span for each request.	☒	☐
The code automatically propagates trace context to downstream HTTP services.	☐	☒
The solution requires additional instrumentation for end-to-end trace correlation.	☒	☐

NEW QUESTION: 63

You are implementing semantic retrieval for a chatbot.
Embeddings are already stored in Redis. However, vector similarity queries do not return matches.
You need to resolve the vector similarity search issue.
What should you do?
A. Set a 24-hour Time to Live (TTL) on embedding keys.

- B. Create an HNSW vector index on the embedding field.
- C. Reset expiration on each read.
- D. Enable geo-replication.

Answer: (SHOW ANSWER)

NEW QUESTION: 64

You are developing an application that uses a Python API to perform similarity queries against Azure Database for PostgreSQL. The application creates a new database connection for every request. During peak traffic, the application intermittently fails to open new database sessions, logs indicate that the maximum number of connections have been reached. You need to configure the connection pooling strategy to reduce connection setup overhead and maximize reuse for the high-concurrency workload. What should you configure? To answer, select the appropriate options in the answer area NOTE: Each correct selection is worth one point.

Connection pooling and session management for high-concurrency AI workloads

Requirement	Configuration
Connection pooling technology	<div>PgBouncer</div> <div>Compute tier scaling</div> <div>Automatic vacuum</div> <div>Connection timeout extension</div>
Pooling mode	<div>Session pooling mode</div> <div>Direct connection mode</div> <div>Statement pooling mode</div> <div>Transaction pooling mode</div>

Answer:

Connection pooling and session management for high-concurrency AI workloads

Requirement	Configuration
Connection pooling technology	<div>PgBouncer</div> <div>Compute tier scaling</div> <div>Automatic vacuum</div> <div>Connection timeout extension</div>
Pooling mode	<div>Session pooling mode</div> <div>Direct connection mode</div> <div>Statement pooling mode</div> <div>Transaction pooling mode</div>

Explanation:

Requirement	Configuration
Connection pooling technology	<div>PgBouncer</div>
Pooling mode	<div>Transaction pooling mode</div>

Connection pooling technology: PgBouncer. Pooling mode: Transaction pooling mode.

Detailed Explanation: Azure Database for PostgreSQL Flexible Server supports PgBouncer to reduce connection setup cost and protect the server from excessive client sessions. For a high-concurrency API whose requests do not require a server session to remain pinned between transactions, transaction pooling maximizes reuse by returning the server connection to the pool after each transaction. Session pooling holds a server connection for the lifetime of the client session and therefore provides less multiplexing under the connection-pressure scenario described.

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 | PgBouncer in Azure Database for PostgreSQL

NEW QUESTION: 65

A recommendation system stores 1 million embeddings in Azure Database for PostgreSQL During peak hours. P95 similarity-query latency increases, and the cache hit ratio drops significantly. You suspect that the vector index working set no longer fits in memory, resulting in more disk reads. You need to scale server resources and validate the outcome. Which four 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.

Actions

Scale to a tier with more memory per vCore.

Review memory pressure and cache hit ratio during peak hours and compare them to the baseline.

Disable autovacuum.

Before peak hours, run a fixed workload to capture baseline P95 latency and capture EXPLAIN ANALYZE for the vector query.

Re-run the same benchmark and compare P95 and P99 latency.

Increase embedding dimensionality.

.....

Answer Area

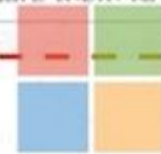
Answer:

Actions

- Scale to a tier with more memory per vCore.
- Review memory pressure and cache hit ratio during peak hours and compare them to the baseline.
- Disable autovacuum.
- Before peak hours, run a fixed workload to capture baseline P95 latency and capture EXPLAIN ANALYZE for the vector query.
- Re-run the same benchmark and compare P95 and P99 latency.
- Increase embedding dimensionality.

Answer Area

- Before peak hours, run a fixed workload to capture baseline P95 latency and capture EXPLAIN ANALYZE for the vector query.
- Scale to a tier with more memory per vCore.
- Re-run the same benchmark and compare P95 and P99 latency.
- Review memory pressure and cache hit ratio during peak hours and compare them to the baseline.



Microsoft

Explanation:

Answer Area

Microsoft

- Before peak hours, run a fixed workload to capture baseline P95 latency and capture EXPLAIN ANALYZE for the vector query.
- Scale to a tier with more memory per vCore.
- Re-run the same benchmark and compare P95 and P99 latency.
- Review memory pressure and cache hit ratio during peak hours and compare them to the baseline.

NEW QUESTION: 66

You need to configure image build automation based on the technical requirements.

Which settings should you configure? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Custom image build configuration

Configuration

Trigger for container images build

Implementation of image builds



Microsoft

Setting

	▼
Commit	
Scheduled	
Base image update	

	▼
ACR Task	
GitHub workflow	
Docker Compose configuration	

Answer:

Custom image build configuration

Configuration

Trigger for container images build

Implementation of image builds



Microsoft

Setting

	▼
Commit	
Scheduled	
Base image update	

	▼
ACR Task	
GitHub workflow	
Docker Compose configuration	

Explanation:

* Trigger for container images build: Base image update

* Implementation of image builds: ACR Task

The correct trigger is Base image update because Proseware requires every custom container image to be rebuilt automatically whenever its Docker Hub base image changes. Azure Container Registry Tasks can detect base-image dependencies from the Dockerfile FROM instruction and automatically trigger a rebuild when that base image is updated. Microsoft confirms that ACR Tasks can track base images in public repositories such as Docker Hub and Microsoft Container Registry, as well as images stored in Azure Container Registry.

The build implementation must be an ACR Task . This directly satisfies the technical requirement that image- build automation run inside Azure Container Registry , eliminating dependencies on local developer machines or separate external build services. ACR Tasks provides cloud-based container building and can automate builds based on source-code commits, base-image updates, or scheduled triggers.

A GitHub workflow could orchestrate CI/CD, but using it as the actual image-building implementation would conflict with the explicit requirement that builds execute inside ACR. Docker Compose defines multi-container applications; it is not the native ACR build automation mechanism. A Commit trigger addresses source changes rather than the stated base-image update requirement, while Scheduled builds would introduce unnecessary periodic execution.

Study Guide references: Azure Container Registry Tasks; base-image dependency tracking; automated rebuilds; Docker Hub base-image triggers; cloud-native container builds.

NEW QUESTION: 67

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. Yes

B. No

Answer: ([SHOW ANSWER](#))

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: 68

An AI platform uses App Configuration for feature flags and endpoint routing.

The platform stores secrets alongside configuration data and does NOT support dynamic configuration refresh.

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: C,D,E ([LEAVE A REPLY](#))

Detailed 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: 69

An application performs similarity search across 5 million embeddings stored in Azure Database for PostgreSQL with pgvector. Queries often filter by department before ranking by cosine distance. P95 latency for vector similarity queries exceeds the SLA target. Monitoring shows sustained high CPU use during query execution. You need to reduce P95 latency for filtered vector similarity queries.

What should you do?

- A. Increase statement timeout.
- B. Store embeddings as JSON.
- C. Create a pgvector index on the embedding column.
- D. Remove the similarity ORDER BY clause.

Answer: C ([LEAVE A REPLY](#))

NEW QUESTION: 70

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: (SHOW ANSWER)

Detailed 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

Valid AI-200 Dumps shared by EduDump.com for Helping Passing AI-200 Exam! EduDump.com now offer the **newest AI-200 exam dumps**, the EduDump.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** EduDump.com AI-200 dumps with Test Engine here: <https://www.edudump.com/exams/Microsoft/AI-200/premium/> (**144** Q&As Dumps, **35%OFF** Special Discount Code: **freecram**)