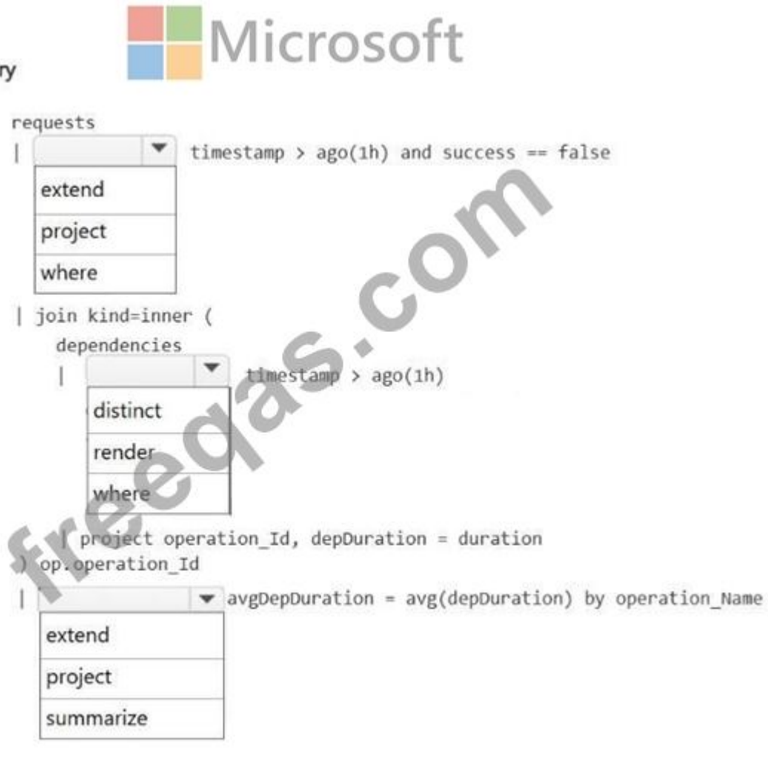
Microsoft.AI-200.v2026-11-23.q122

Exam Code:	AI-200
Exam Name:	Developing AI Cloud Solutions on Azure
Certification Provider:	Microsoft
Free Question Number:	122
Version:	v2026-11-23
# of views:	139
# of Questions views:	1220
https://www.freeqas.com/qa/Microsoft/AI-200/Microsoft.AI-200.v2026-11-23.q122.html	

NEW QUESTION: 1
Hotspot Question
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 use? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

Completing a KQL query

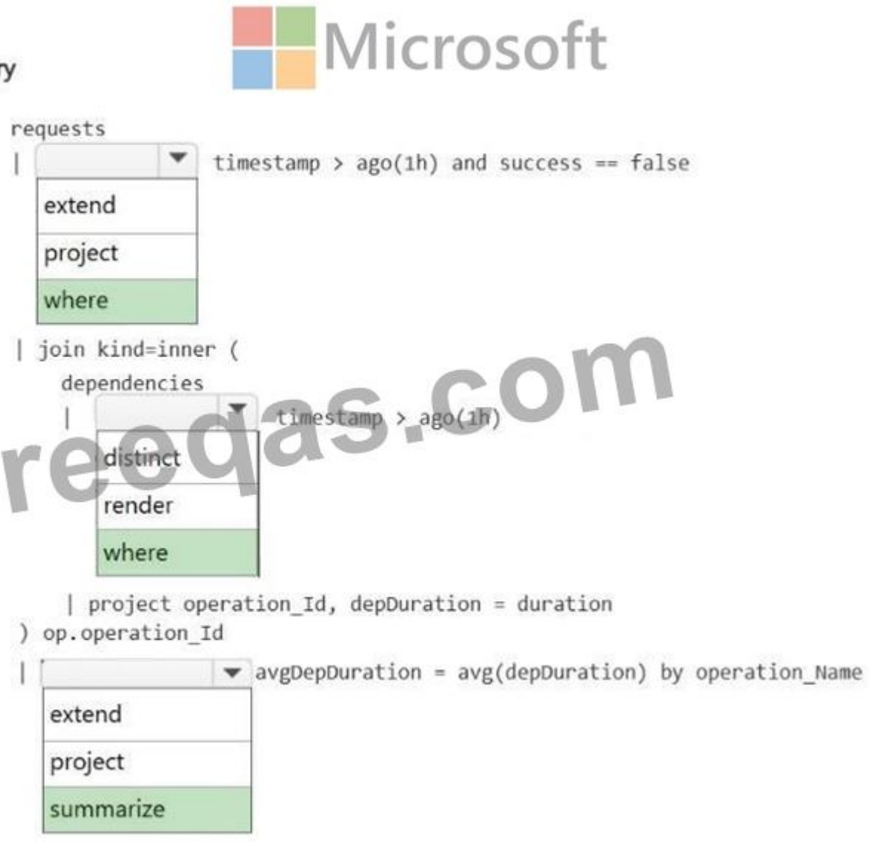


freeqas.com

Answer:

Answer Area

Completing a KQL query



freeqas.com

Explanation:

Box 1: where

Filter telemetry rows

The where operator filters a table to rows that satisfy a specific predicate. Both box 1 and box 2 require filtering telemetry data to the last hour (timestamp > ago(1h)).

Box 2: where

In the Kusto Query Language (KQL), the where operator filters a table to rows that satisfy a specific predicate or condition. Inside the inner subquery, you need to filter the dependencies telemetry table to only include records from the last hour (timestamp > ago(1)) Box 3: summarize Aggregate calculated data The summarize operator produces a table that aggregates the content of the input table. It is used here with the avg() aggregation function to calculate the average duration grouped by the operation_Name.

Reference:

https://tryhackme.com/room/kqlkustobasicqueries

NEW QUESTION: 2

Drag and Drop Question

You are developing several microservices to run on Azure Container Apps.

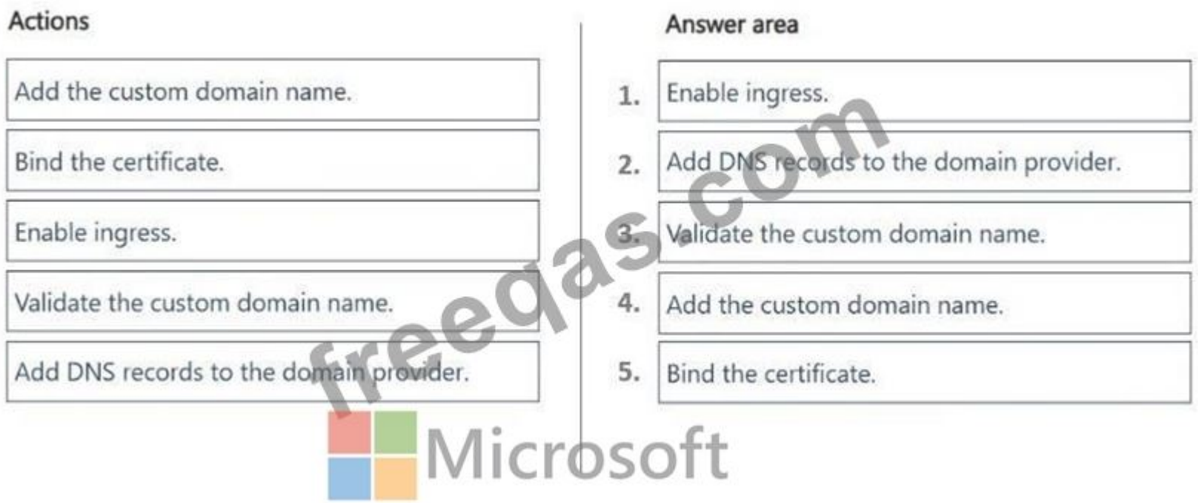
The microservices must allow HTTPS access by using a custom domain.

You need to configure the custom domain in Azure Container Apps.

In which order should you perform the actions? To answer, move all actions from the list of actions to the answer area and arrange them in the correct order.



Answer:



Explanation:

Step 1: Enable ingress

You must first expose your container app to external traffic to generate the default fully qualified domain name (FQDN) needed for DNS mapping.

Step 2: Add DNS records to the domain provider

Log into your domain registrar to create the required TXT (for verification) and CNAME/A records pointing to your container app.

Step 3: Validate the custom domain name

Azure checks your DNS records to confirm that you actually own the domain before allowing it to be linked.

Step 4: Add the custom domain name

Once validation passes, you officially add and register the custom domain name within the Azure Container App configuration.

Step 5: Bind the certificate

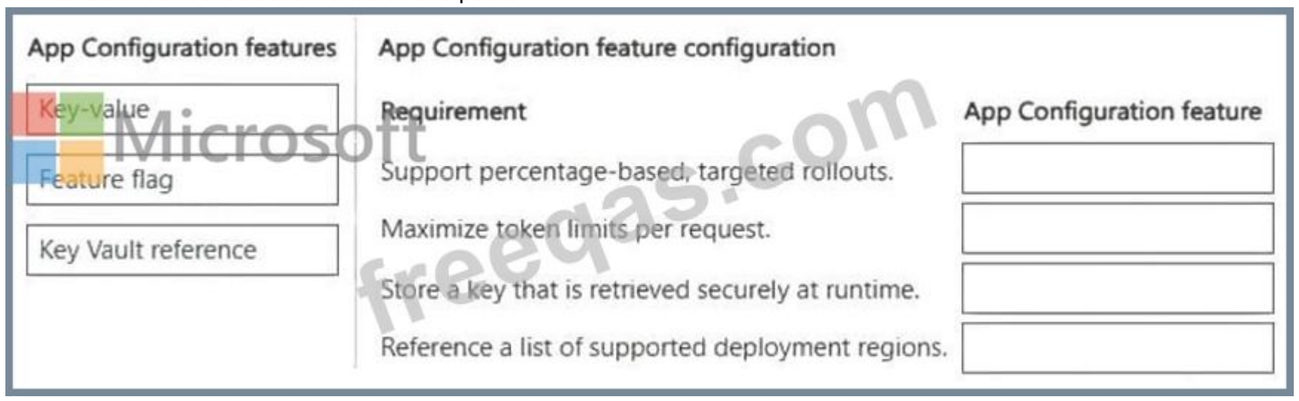
Finally, bind an SSL/TLS certificate to the custom domain to secure the connection and enable HTTPS access.

Reference:

https://learn.microsoft.com/en-us/azure/container-apps/ingress-overview

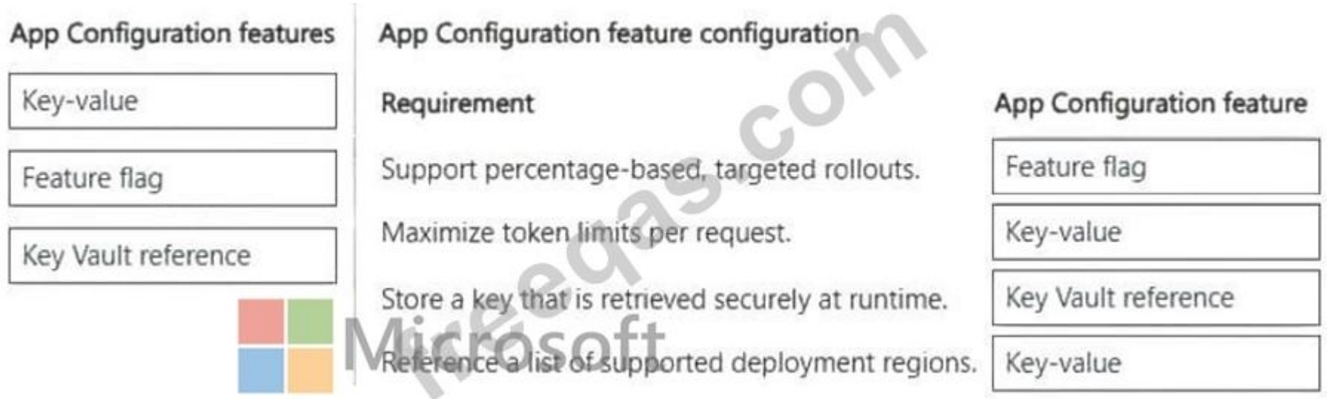
NEW QUESTION: 3
You are implementing semantic retrieval for a chatbot.
Embeddings are already stored in Redis. However, vector similarity queries do not return matches.
You need to resolve the vector similarity search issue.
What should you do?
A. Configure a metadata-only secondary index.
B. Enable geo-replication.
C. Create a FLAT vector index on the embedding field.
D. Set a 24-hour Time to Live (TTL) on embedding keys.
Answer: C (LEAVE A REPLY)
Creating a FLAT vector index on your embedding field will resolve the issue and allow your Redis vector similarity queries to return matches.
In Redis (using the Redis Search and Query features), vector fields cannot be queried using Vector Similarity Search (VSS) syntax until a dedicated vector index is explicitly built over them.
No Automatic Indexing: Redis does not automatically index JSON or Hash fields containing raw binary or string embeddings.
Query Failure: Without an index, VSS queries (using the KNN operator) will fail with syntax errors or return zero results because the query engine cannot parse the unindexed field.
Reference:
https://www.louisbouchard.ai/indexing-methods/

NEW QUESTION: 4
Drag and Drop Question
You are configuring an Azure-hosted AI application that uses Azure App Configuration to manage its runtime settings.
The application must provide the following functionality:
- Support percentage-based, targeted rollouts.
- Maximize token limits per request.
- Store a key that is retrieved securely at runtime.
- Reference a list of supported deployment regions.
You need to configure the App Configuration features for the app.
Which features should you configure? To answer, move the appropriate App Configuration features to the correct configuration items. You may use each App Configuration features 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.



Microsoft
freeqas.com

Answer:



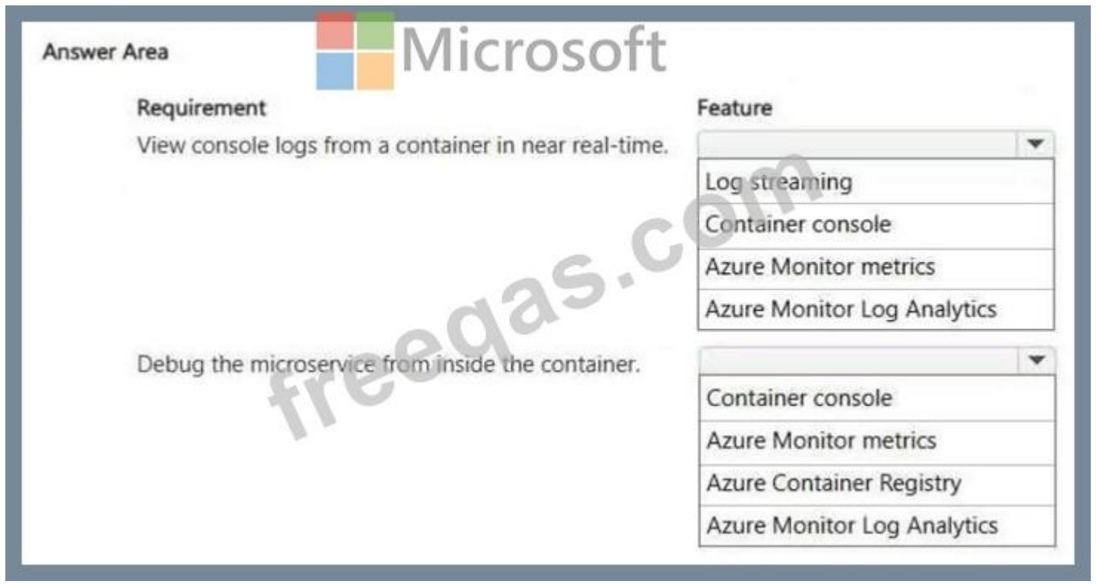
Explanation:
Box 1: Feature flag
Use a Feature flag.
Azure App Configuration provides built-in feature management specifically designed to dynamically control application behavior without redeploying code.
Percentage-Based Rollouts: Feature flags support targeting filters (such as the standard Microsoft.Targeting filter), which natively handle percentage-based rollouts and specific audience targeting rules.
Dynamic Changes: You can safely toggle features or change the exposure percentage on the fly directly through the Azure portal or via APIs.
Box 2: Key-value
Key-value is the correct feature to use.
A standard key-value pair is the optimal choice for managing runtime settings like numeric thresholds or token limits.
Direct Data Injection: It allows you to store the specific maximum token limit (e.g., MaxTokensPerRequest = 4096) as an integer or string that your application can parse directly at runtime.
No Extra Overhead: Unlike the other options, a standard key-value pair maps directly to application configuration variables without requiring secondary lookups or conditional logic evaluation.
Box 3: Key Vault reference
Use an Azure Key Vault reference.
Azure App Configuration is optimized for non-sensitive runtime configurations like feature flags, environment names, and general application settings. Sensitive values (such as API keys for OpenAI/LLM models, database passwords, and connection strings) should always be stored natively in Azure Key Vault.
By creating a Key Vault Reference inside Azure App Configuration, you store only the URI pointing to the secret rather than the secret value itself. When your AI application requests configuration data at runtime, the App Configuration provider automatically contacts Key Vault, resolves the reference, and securely injects the actual secret directly into your application's memory pipeline.
Box 4: Key-value
Use a standard configuration key-value pair (or a JSON array within a key).
How to store it: Save the regions as a comma-separated string or a JSON array under a single key.
Key Example: Deployment:SupportedRegionsValue Example: ["eastus", "westeurope",
"southeastasia"]
Content Type: Set the content type to application/json so your application can automatically parse it into a list or array at runtime.
Reference:
https://learn.microsoft.com/en-us/azure/azure-app-configuration/manage-feature-flags
https://turbo360.com/blog/get-azure-key-value-pairs-from-app-configuration
https://learn.microsoft.com/en-us/azure/app-service/app-service-key-vault-references

NEW QUESTION: 5

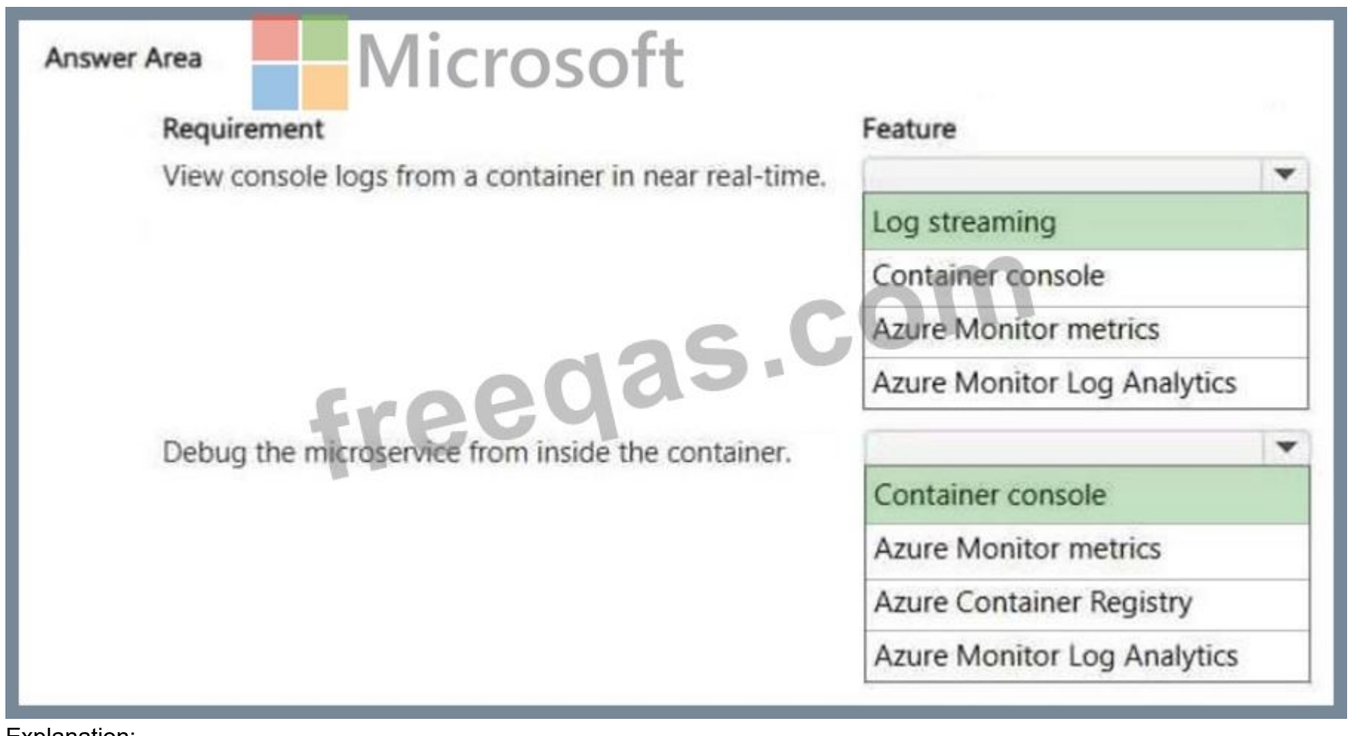
Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals.
You are preparing a production deployment for an Azure Function app. The app will run across multiple environments.
The solution must support environment-specific configuration and prevent secrets from being stored in source control.
You need to develop the solution.
Solution: Store connection strings in the Function app application settings configured in the Azure Portal.
Does the solution meet the goal?
A. Yes
B. No
Answer: (SHOW ANSWER)
Azure Functions application settings are the correct mechanism for keeping environment-specific configuration outside the application source code. Microsoft documents that application settings, including connection strings and other configuration values, are defined separately for each function app . This allows development, test, staging, and production deployments to use different values without modifying the application code.
At runtime, Function app settings are exposed to the application as environment variables , so code can retrieve the required connection string or configuration value without embedding it in a configuration file committed to source control. This directly satisfies the requirement to prevent secrets from being stored in the Git repository.
For production-grade secret management, Azure Key Vault references are often preferable because they provide centralized secret lifecycle management and can be combined with managed identities. However, the question asks only whether the proposed solution meets the stated goals. Storing connection strings in Azure Function application settings does satisfy both: values can vary by deployed environment and remain outside source-controlled application files.
Study Guide references: Azure Functions configuration; application settings; environment variables; secure application configuration.

NEW QUESTION: 6
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: A (LEAVE A REPLY)
Azure AI Foundry's evaluation tooling (often via prompt flow or the Evaluation SDK) runs bulk test sets against both models and scores outputs on metrics like groundedness, coherence, and relevance, enabling an objective comparison before promotion.

NEW QUESTION: 7
Hotspot Question
You are developing several microservices to run on Azure Container Apps.
You need to monitor and diagnose the microservices.
Which features should you use? To answer, select the appropriate feature in the answer area.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

Box 1: Log streaming

To view console logs from an Azure Container App in near real-time, you should use the Log stream feature.

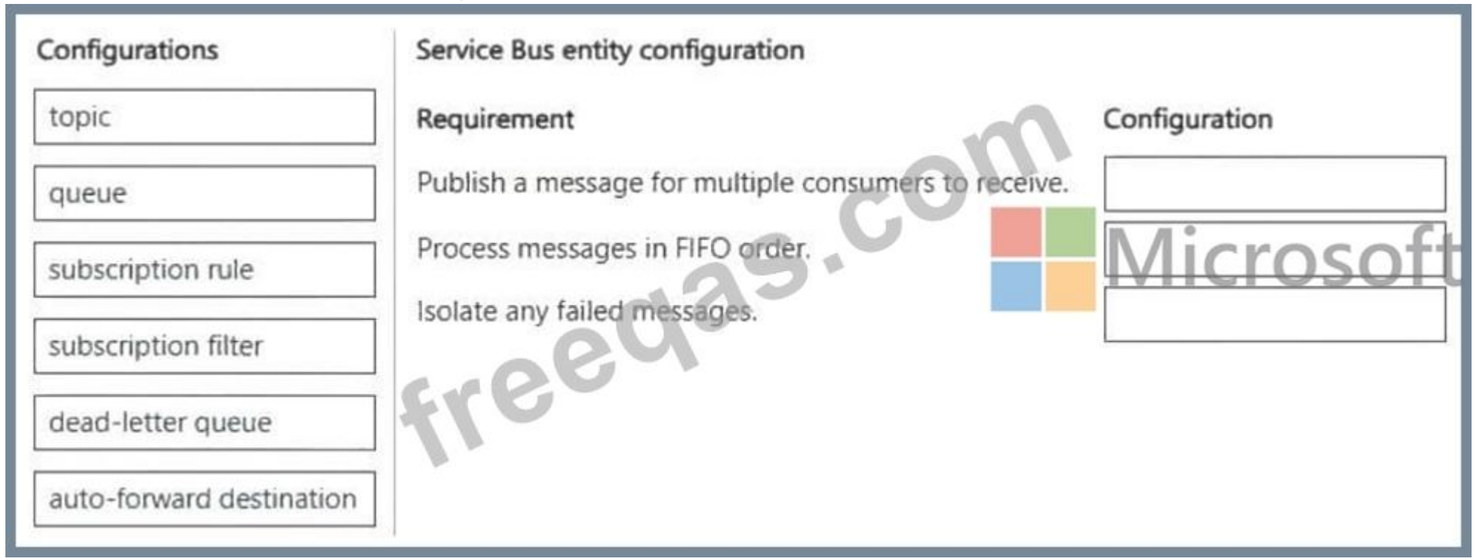
How to Access Log Stream

Navigate to your container app in the Azure portal.
Under the Monitoring section of the sidebar menu, select Log stream.
Set the log type to Console to instantly stream your application's stdout and stderr streams.
Box 2: Container console
To debug your microservice from inside the container, you should use the Container console feature.
Key Features for Debugging and Monitoring
Container console: This feature allows you to connect directly to the Linux console/shell inside your running container instance. You can use it to execute commands, inspect files, and troubleshoot the container's environment in real-time.
Log stream: This tool is used alongside the console to view streaming system and console logs from your container in near real-time.
Azure Monitor Logs: Used for querying and analyzing historical application and system logs over longer periods.
Reference:
https://learn.microsoft.com/en-us/azure/container-apps/observability

NEW QUESTION: 8
You plan to deploy an Azure Container app.
You need to configure the container app to support session affinity.
Which ingress type and revision mode should you assign to the container app?
A. TCP ingress type and single revision mode
B. TCP ingress type and multiple revision mode
C. HTTP ingress type and multiple revision mode
D. HTTP ingress type and single revision mode
Answer: D (LEAVE A REPLY)
Azure Container Apps session affinity, also called sticky sessions , routes requests from the same client to the same container-app replica. Microsoft documents that this functionality is implemented using HTTP cookies , so it requires HTTP ingress rather than raw TCP ingress.
The second requirement is single revision mode . Microsoft explicitly states that session affinity is supported only when the container app is in single revision mode and HTTP ingress is enabled . In single revision mode, all traffic is directed to the latest active revision, allowing the ingress layer to consistently associate a client with a particular replica within that revision.
Multiple revision mode is designed for scenarios such as traffic splitting, blue-green deployments, and A/B testing, where inbound traffic can be distributed among multiple active revisions. That model is incompatible with the documented session-affinity restriction. TCP ingress is also incorrect because sticky-session behavior relies on HTTP cookie affinity rather than generic TCP connection routing.
Therefore, the required configuration is HTTP ingress with single revision mode .
Study Guide references: Azure Container Apps # ingress; sticky sessions/session affinity; revision modes; HTTP routing.

NEW QUESTION: 9
Drag and Drop Question
You are building a back-end pipeline that receives AI inference requests.
The pipeline must perform the following actions:
- Publish a message so that multiple independent consumers receive it.
- Process messages in first-in, first-out (FIFO) order.
- Isolate any failed messages.
You need to configure the appropriate Service Bus entities.
How should you configure the Service Bus entities? 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.



Answer:



Explanation:

Box 1: topic

You should use an Azure Service Bus Topic to publish the message.

In a pub/sub architecture, the topic acts as the single destination where the sender publishes the message.

Box 2: queue

You should use a dedicated FIFO queue for this purpose.

While a standard queue allows parallel processing that can disrupt the arrival order, a strict FIFO (First-In, First-Out) queue guarantees that your AI inference requests are processed in the exact order they are received.

Order Preservation: Standard message queues only guarantee "best-effort" ordering. FIFO queues ensure strict sequential processing.

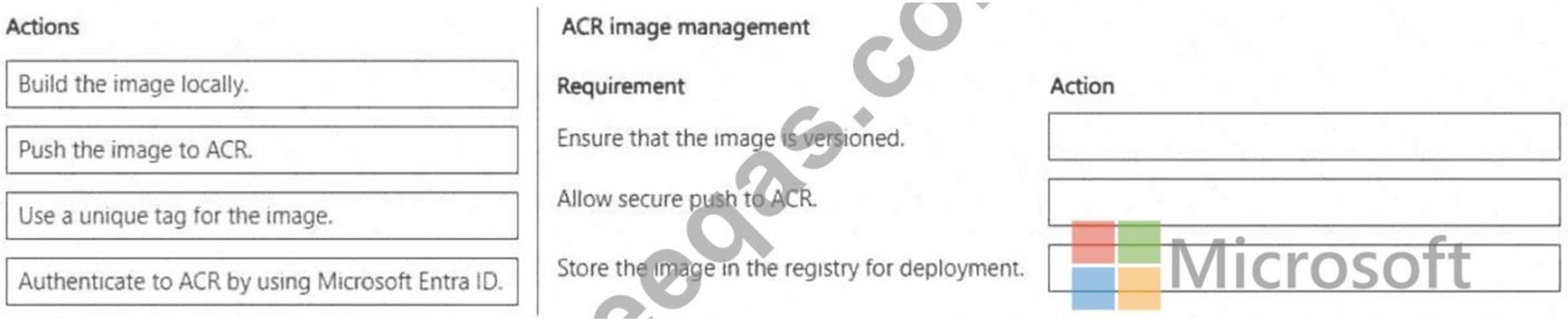
Exactly-Once Processing: FIFO queues inherently prevent duplicate message delivery. This prevents running the same costly AI inference twice.

Concurrency Control: You can limit or throttle consumers. This prevents overwhelming your AI models or GPUs.

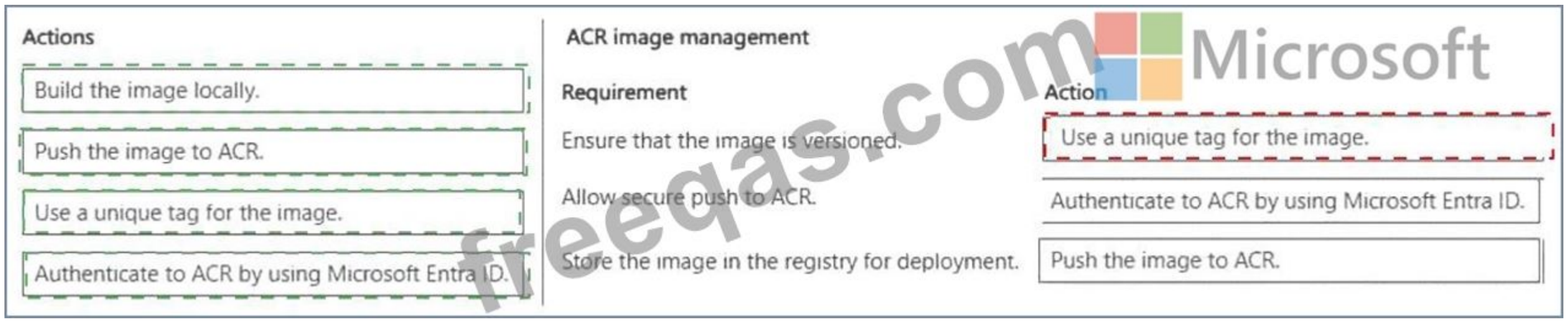
Box 3: dead-letter queue
A dead-letter queue (DLQ) is the exact industry-standard pattern and correct architectural choice to isolate failed messages in an AI inference pipeline. It captures messages that fail processing repeatedly due to errors like model timeouts, malformed input payloads, or downstream GPU memory crashes.
Prevents Blockages: Stops bad requests from clogging the main queue and halting pipeline throughput.
Enables Debugging: Preserves the exact failed payload and error metadata for post-mortem analysis.
Protects Resources: Keeps infrastructure from getting stuck in endless retry loops on broken data.
Reference:
https://wearenotch.com/blog/azure-service-bus-tips-to-optimize-functions/

NEW QUESTION: 10
You have an Event Grid subscription that triggers an Azure Function.
You need to prevent loss of events in case the endpoint returns an HTTP 400 response.
Which action should you perform?
A. Implement optimistic batching.
B. Implement asynchronous handshake validation.
C. Configure a dead-letter destination.
D. Configure a retry policy.
Answer: C (LEAVE A REPLY)
Configuring a dead-letter destination is exactly the right step to take.
By default, when an endpoint returns an HTTP 400 (Bad Request) or HTTP 413 (Payload Too Large) response, Azure Event Grid immediately stops delivery attempts and drops the message.
It treats these specific errors as non-transient, client-side issues, meaning it skips its standard 24- hour retry policy.
Configuring a dead-letter destination ensures that these dropped events are safely preserved for future troubleshooting and reprocessing.
Reference:
https://turbo360.com/blog/azure-event-grid-dead-letter-monitoring

NEW QUESTION: 11
-
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.



Answer:



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.

NEW QUESTION: 12

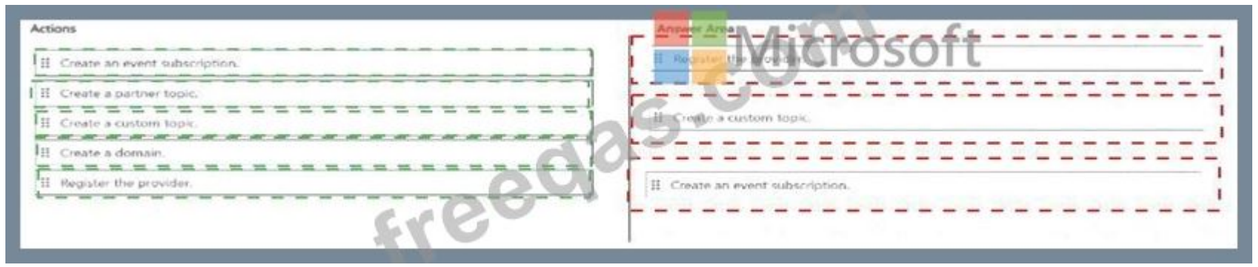
You have a newly provisioned Azure subscription. You are designing a custom Event Grid workflow for AI inference events.

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.



Answer:



Explanation:



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: 13

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)

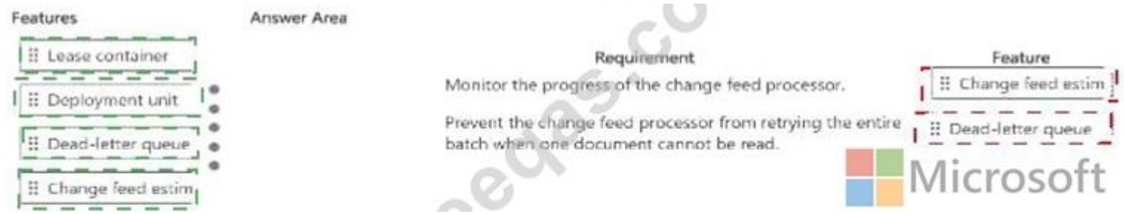
For precise questions over tabular numeric data, convert the workbook data into a structured relational/table representation and expose controlled query functionality to the AI application through function calling. This preserves columns such as quarter, region, revenue, product, and fiscal period as 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: 14
You store embeddings in Redis by using keys formatted as doc:(id). Some embeddings are accessed frequently. Others are rarely used.
You need to implement a caching strategy that keeps only frequently accessed embeddings in memory.
What should you use?
A. volatile-ttl
B. allkeys-lru
C. EXPIRE command
D. time-window expiration
Answer: B (LEAVE A REPLY)
To implement this strategy, you should use allkeys-lru combined with the EXPIRE command as a secondary fallback.
Primary Mechanism: Configure your Redis maxmemory-policy to allkeys-lru.
Secondary Mechanism: Apply the EXPIRE command to your keys as a safety net.
Why allkeys-lru is the Best Choice
Memory Management: It automatically evicts the Least Recently Used (LRU) keys across your entire dataset when Redis hits its memory limit.
Frequent Access: It guarantees that frequently accessed embeddings stay in memory, regardless of when they were created.Prefix Independent: It scans all keys, making it perfect for your doc:(id) format Reference: https://rahulchowdhury.in/blog/redis-caching-patterns-every-mern-dev

NEW QUESTION: 15
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.



Answer:



Explanation:



NEW QUESTION: 16

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear on the review screen.

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

The container must be able to retrieve the API key at runtime without exposing it in the source repository or Git commit history.

You need to ensure that the API key remains outside of Git commit history and is available to the container at runtime.

Solution: Store the API key as a GitHub repository secret.

Does the solution meet the goal?

A. Yes

B. No

Answer: (SHOW ANSWER)

Correct:

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

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

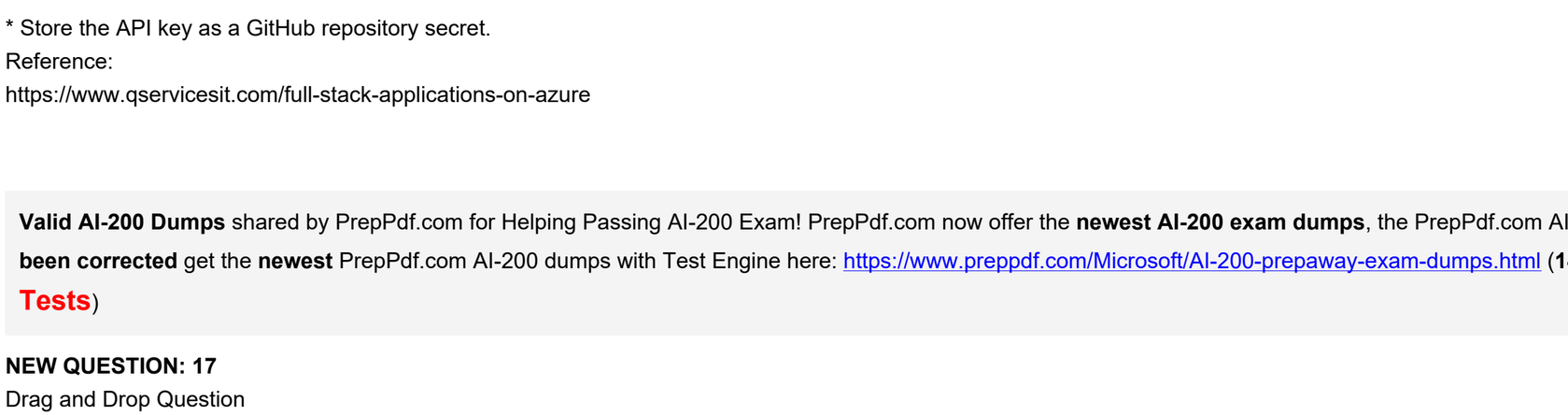
Incorrect:

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

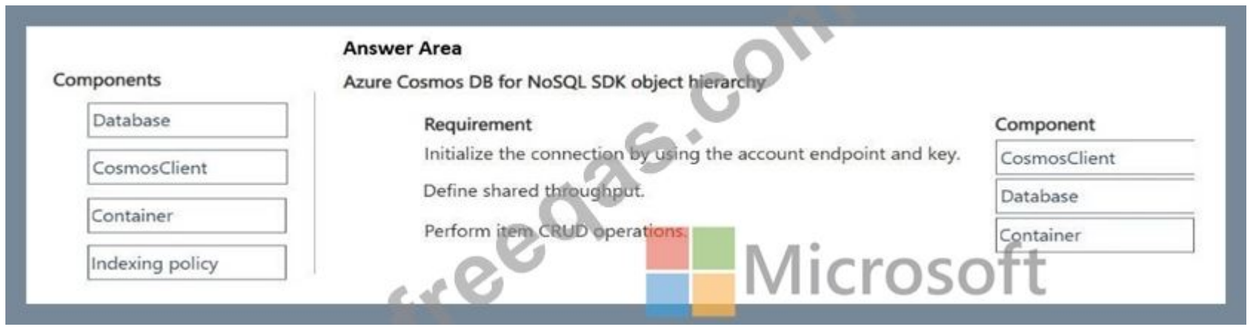
* Store the API key as a GitHub repository secret.
Reference:
https://www.qservicesit.com/full-stack-applications-on-azure

Valid AI-200 Dumps shared by PrepPdf.com for Helping Passing AI-200 Exam! PrepPdf.com now offer the **newest AI-200 exam dumps**, the PrepPdf.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** PrepPdf.com AI-200 dumps with Test Engine here: https://www.preppdf.com/Microsoft/AI-200-prepaway-exam-dumps.html (**144** Q&As Dumps, **40%OFF** **Special Discount: Exam-Tests**)

NEW QUESTION: 17
Drag and Drop Question
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.



Answer:



Explanation:

Box 1: CosmosClient

To initialize the connection to an Azure Cosmos DB for NoSQL account using the account endpoint and key, you must use the CosmosClient class.

Box 2: Database

The database SDK component should be used to define shared throughput.

In Azure Cosmos DB, shared throughput (provisioned Request Units per second or RU/s) is configured at the database level. When throughput is provisioned on a database, that capacity is shared among all the containers created within that specific database.

Box 3: Container

To perform item create, read, update, and delete (CRUD) operations, you should use the Container SDK component In the Azure Cosmos DB for NoSQL SDK for .NET, individual JSON documents (items) live inside a container. The Container class exposes the specific methods required to execute CRUD operations on these items.

Reference:

https://learn.microsoft.com/en-us/azure/cosmos-db/how-to-dotnet-get-started

NEW QUESTION: 18

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: A,C (LEAVE A REPLY)

To build a scalable and reliable asynchronous solution that meets the requirements, you should use an HTTP-triggered function to save images to Azure Blob Storage and enqueue a message, then use a Queue-triggered function to process the image.

1. Decouple via Queue Messaging

HTTP Function: Receives the initial user upload, saves the large image file directly to Azure Blob Storage, and writes a small metadata message (e.g., the blob URI) into an Azure Storage Queue or Azure Service Bus Queue.

Instant Response: The HTTP function immediately returns a 202 Accepted status to the client along with a status-tracking URL. This completely eliminates client timeout issues by shifting heavy work to the background.
2. Leverage Built-In Queue Triggers
Queue-Triggered Function: A second, separate Azure Function is configured to trigger whenever a new message lands in the queue. It downloads the image from Blob Storage and performs the processing.
Automatic Retries: Azure Queue storage and Service Bus triggers feature native retry behavior. If the processing function fails or crashes, the message is automatically returned to the queue to be retried. Persistent failures are automatically moved to a Poison Queue (Dead Letter Queue) after a configured number of attempts (default is 5).
Independent Scaling: Under the Azure Consumption or Premium plan, the Queue-triggered function scales its instances up or down based on the queue length (the backlog of images to process). This scales independently of the HTTP-triggered function, which scales purely based on incoming request volume.
Reference:
https://oneuptime.com/blog/post/2026-02-16-azure-functions-python-http-triggers-blob-storage/view

NEW QUESTION: 19
Hotspot Question
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.

Answer Area

Container configuration settings

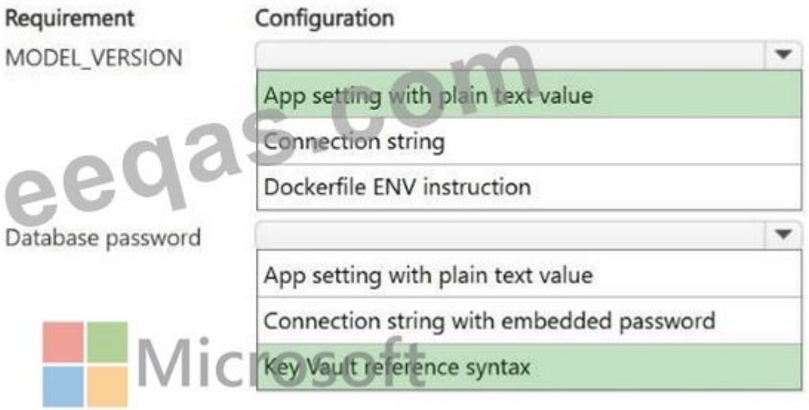
Requirement	Configuration
MODEL_VERSION	App setting with plain text value Connection string Dockerfile ENV instruction
Database password	App setting with plain text value Connection string with embedded password Key Vault reference syntax

Microsoft
freeqas.com

Answer:

Answer Area

Container configuration settings



freeqas.com
Microsoft

Explanation:

Box 1: App setting with plain text value

To provide a non-sensitive environment variable named MODEL_VERSION to a Linux custom container deployed to Azure App Service, you should configure it as an Application Setting (App Setting). In Azure App Service, any custom configuration defined within the Application Settings is automatically injected into Linux containers as an environment variable at runtime.

Box 2: Key Vault reference syntax

You should configure the database password as an Azure Key Vault reference inside App Service Application Settings. This prevents the secret from being stored in plaintext in the app configuration and injects it securely into the container as an environment variable at runtime.

Reference:

https://learn.microsoft.com/en-us/azure/app-service/tutorial-custom-container?tabs=azure-cli&pivots=container-linux

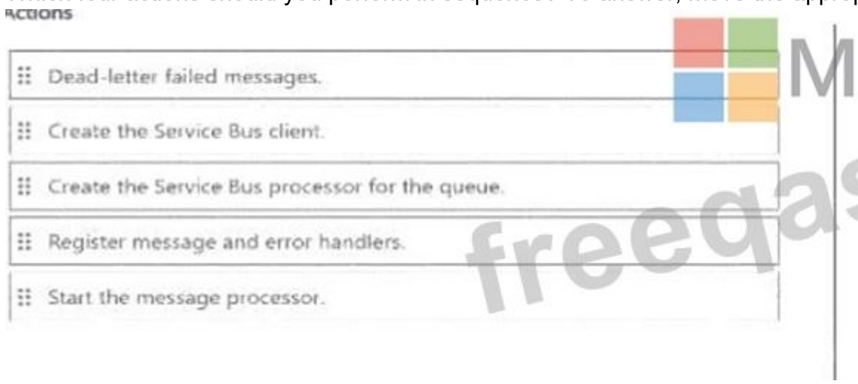
NEW QUESTION: 20

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.

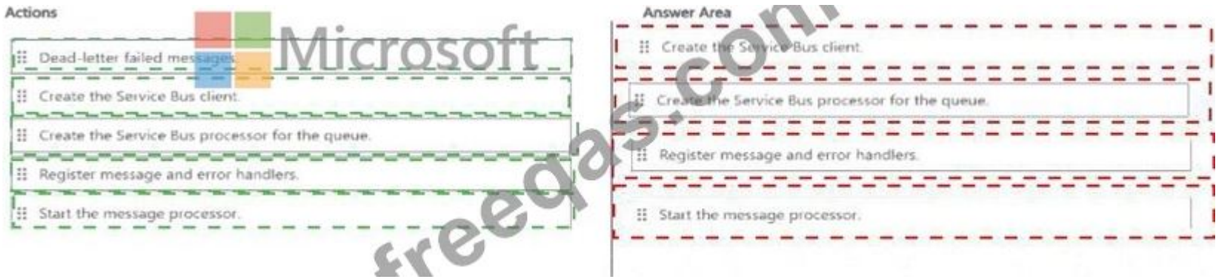
You need to provision and configure the Service Bus processor for message and error handling.

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.



Microsoft
freeqas.com

Answer:



Explanation:



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: 21

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: B (LEAVE A REPLY)

You should cache configuration values with a refresh interval.

Enables Dynamic Configuration: Azure App Configuration providers natively support a cache- polling mechanism. Once the refresh interval expires, the application checks for updates and reloads configuration values automatically without requiring an application restart.

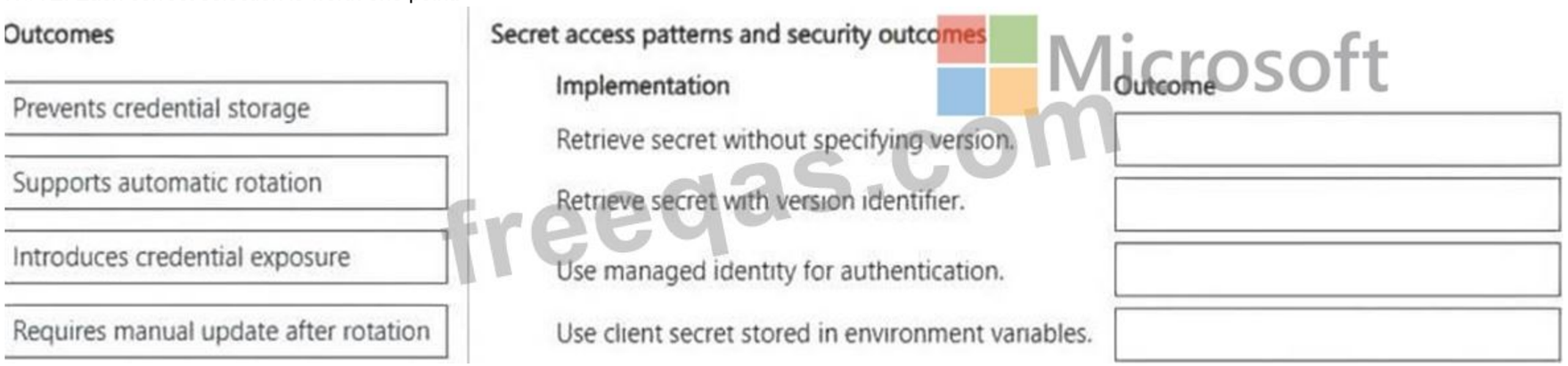
Minimizes Latency: By pulling configuration values from a local, in-memory cache for all intermediary requests, you completely bypass the network latency of making a remote API call to Azure on every single application request.

Prevents Throttling: Relying on cached values prevents your application from exhausting your App Configuration resource's request quota.

Reference:

https://learn.microsoft.com/en-us/azure/azure-app-configuration/howto-best-practices

NEW QUESTION: 22
Drag and Drop Question
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.



Answer:



Explanation:
Box 1: Supports automatic rotation.
When an AI application retrieves a secret from Azure Key Vault without specifying a version, it always fetches the latest active version. This choice impacts your security and operations in the following ways:
Latest Version Fetch: The vault automatically serves the current active secret value.
No Code Changes: Your application reads the new credential immediately after rotation without a code redeployment.
Zero Downtime: Automated rotation pipelines can update the secret in Key Vault while the app seamlessly switches to the new token.
Box 2: Requires manual update after rotation

Using a specific version identifier when retrieving a secret from Key Vault means the application always requests one exact version of that secret. This choice requires manual update after rotation because the application code or configuration pins a specific version ID and will not fetch the new value when the secret rotates.
Box 3: Prevents credential storage
Prevents credential storage is the correct outcome of using managed identity for authentication.
No credentials to manage: Managed identities eliminate the need for developers to manage credentials entirely.
Automatic token handling: Azure automatically creates an identity for the application in Microsoft Entra ID and handles the service principal's lifecycle behind the scenes, ensuring that no secrets or connection strings are ever hardcoded or stored in configuration files.
Box 4: Introduces credential exposure
Introduces credential exposure and Requires manual update after rotation are the primary outcomes of storing a client secret in environment variables.
Storing clear-text secrets in environment variables poses a significant security risk. Any process, compromised dependency, or user with access to the execution environment (such as container details, application logs, or process dumps) can easily read them.
Reference:
https://medium.com/simform-engineering/building-automated-secret-rotation-using-azure-key-vault-and-net-01e0701f0a2e
https://www.varonis.com/blog/azure-managed-identities

NEW QUESTION: 23
You are developing an AI API deployed to ACA. The API requires database credentials that are stored in Key Vault. Key Vault is configured to use Azure RBAC for access control.
The database credentials are rotated periodically by the security team. The application must always use the latest version of each credential without being redeployed and without exposing secrets in code or configurations.
You need to implement a secure secret access strategy that prevents credential exposure and fetches the latest version of each secret at runtime without redeploying the container.
Which three actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.
A. Configure a Key Vault RBAC role assignment.
B. Configure a Key Vault access policy.
C. Export the secret during deployment.
D. Assign a system-assigned managed identity.
E. Retrieve the secret at runtime by using the SDK.
Answer: (SHOW ANSWER)
Step-by-Step Implementation Guide
1. Assign a System-Assigned Managed Identity
2. Configure Key Vault RBAC Role Assignment
Role Selection: Assign the Key Vault Secrets User role to the container app's managed identity.
Scope Limitation: Limit the scope of this assignment to the specific Key Vault or individual secrets rather than the entire resource group.
3. Retrieve the Secret at Runtime Using the SDK
Reference:
https://oneuptime.com/blog/post/2026-02-16-how-to-use-managed-identity-with-azure-container-apps-to-access-azure-services/view

NEW QUESTION: 24
You are troubleshooting a production Azure OpenAI service that is experiencing intermittent connectivity issues. All telemetry is sent to a Log Analytics workspace.
You must write a KQL query that analyzes the AppRequests table. The query must filter for requests that occurred within the last 30 minutes and have a failed status.
The final output must display the total number of failures for each unique operation.
You need to select the KQL operators required to filter the time range and aggregate the failure counts.

Which two operators should you use? Each correct answer presents part of the solution. Choose two.
NOTE: Each correct selection is worth one point.
A. distinct
B. project
C. where
D. summarize
Answer: (SHOW ANSWER)

NEW QUESTION: 25
You are designing an Azure Function that will process orders from a new Azure Service Bus queue.
You need to prevent the system from processing messages more than once and preserve failed messages for investigation.
Which two actions should you implement? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.
A. Include a dead-letter handling process.
B. Process orders by using an HTTP trigger.
C. Enable duplicate detection on the queue.
D. Enable sessions on Service Bus.
Answer: A,C (LEAVE A REPLY)
Enable duplicate detection on the Azure Service Bus queue to suppress duplicate sends. Service Bus tracks application-defined MessageId values during the configured duplicate-detection window. If another message with the same MessageId arrives during that interval, Service Bus accepts the send operation but discards the duplicate message. Microsoft specifically recommends using a deterministic business identifier, such as an order ID, as the basis for MessageId when duplicate suppression is required. Duplicate detection must be enabled when the queue is created and is supported in the Standard and Premium tiers.
For failed or poison messages, implement dead-letter handling . A Service Bus queue has a dead-letter subqueue that retains messages that cannot be successfully processed, including messages that exceed the configured maximum delivery count. Those messages remain available for inspection, troubleshooting, correction, or controlled reprocessing rather than being silently discarded.
Sessions provide ordered/FIFO processing and session affinity; they do not primarily prevent duplicates. An HTTP trigger would bypass the required Service Bus processing architecture.
For strict production-grade exactly-once business effects, the consumer should additionally be idempotent , because a message can still be redelivered after a lock or settlement failure.
Study Guide references: Azure Service Bus # duplicate detection; MessageId; dead-letter queues; poison- message handling; reliable asynchronous processing.

NEW QUESTION: 26
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. Create B-tree indexes on frequently filtered metadata columns.
B. Store embeddings as JSON.
C. Increase embedding dimensionality.
D. Increase statement timeout.
Answer: A (LEAVE A REPLY)
Creating a B-tree index on the department column will help, but only if you use the correct indexing strategy, as a standard B-tree index alone is often ignored during vector searches.
When you run a query that filters by metadata before performing a vector similarity search, PostgreSQL must choose between filtering the rows first or searching the vector index first. In many cases, a standard B-tree index on the metadata column combined with an HNSW/IVFFlat index on the vector column results in pre-filtering that defaults to a sequential scan, or post- filtering that returns fewer results than requested.
Reference:

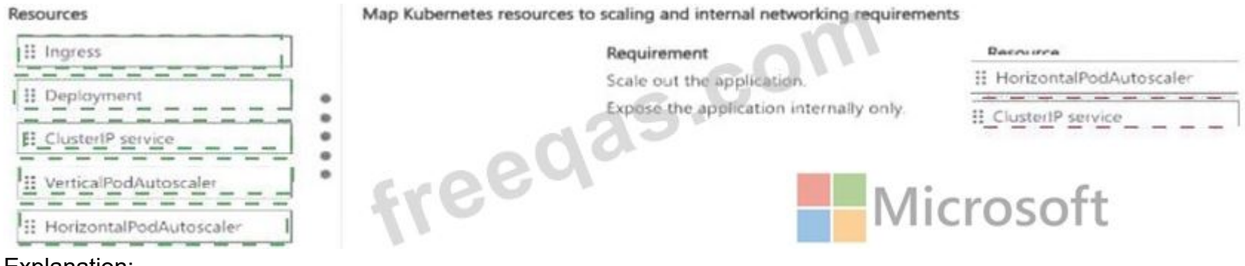
https://learn.microsoft.com/en-us/azure/horizondb/ai/vector-search-pgvector

NEW QUESTION: 27
You are developing a Java application to be deployed in Azure. The application stores sensitive data in Azure Cosmos DB.
You need to configure Always Encrypted to encrypt the sensitive data inside the application.
What should you do first?
A. Create a customer-managed key (CMK) and store the key in a new Azure Key Vault instance.
B. Create a data encryption key (DEK) by using the Azure Cosmos DB SDK and store the key in Azure Cosmos DB.
C. Create a Microsoft Entra ID managed identity and assign the identity to a new Azure Key Vault instance.
D. Create a new container to include an encryption policy with the JSON properties to be encrypted.
Answer: A (LEAVE A REPLY)
To configure Always Encrypted (client-side encryption) for your Java application using Azure Cosmos DB, you must first create a customer-managed key (CMK) and store it in a new Azure Key Vault instance Reference:
https://lennilobel.wordpress.com/2022/08/20/configuring-client-side-encryption-aka-always-encrypted-in-azure-cosmos-db/

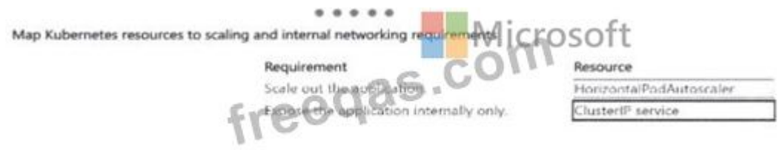
NEW QUESTION: 28
You plan to deploy a web application to AKS.
The solution must:
* Scale out the application by adding more pods during peak CPU usage
* Expose the application internally within the cluster only
You need to configure a Kubernetes resource for each requirement
Which resources should you configure? To answer, move the appropriate resources to the correct requirements You may use each resource once, more than once, or not at all. You may need to move split bar between .. or scroll to view content.



Answer:



Explanation:

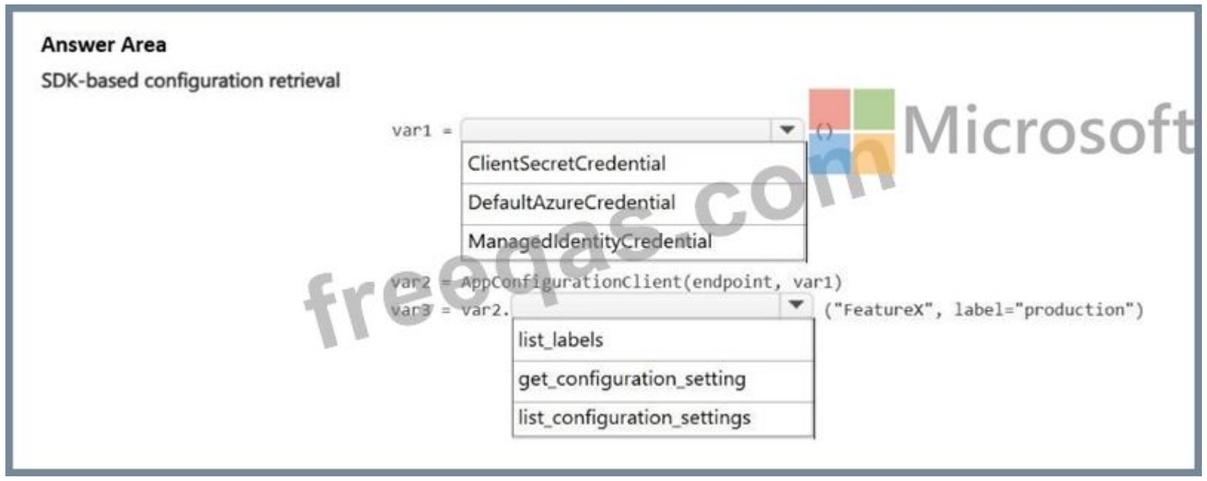


Microsoft
freeqas.com

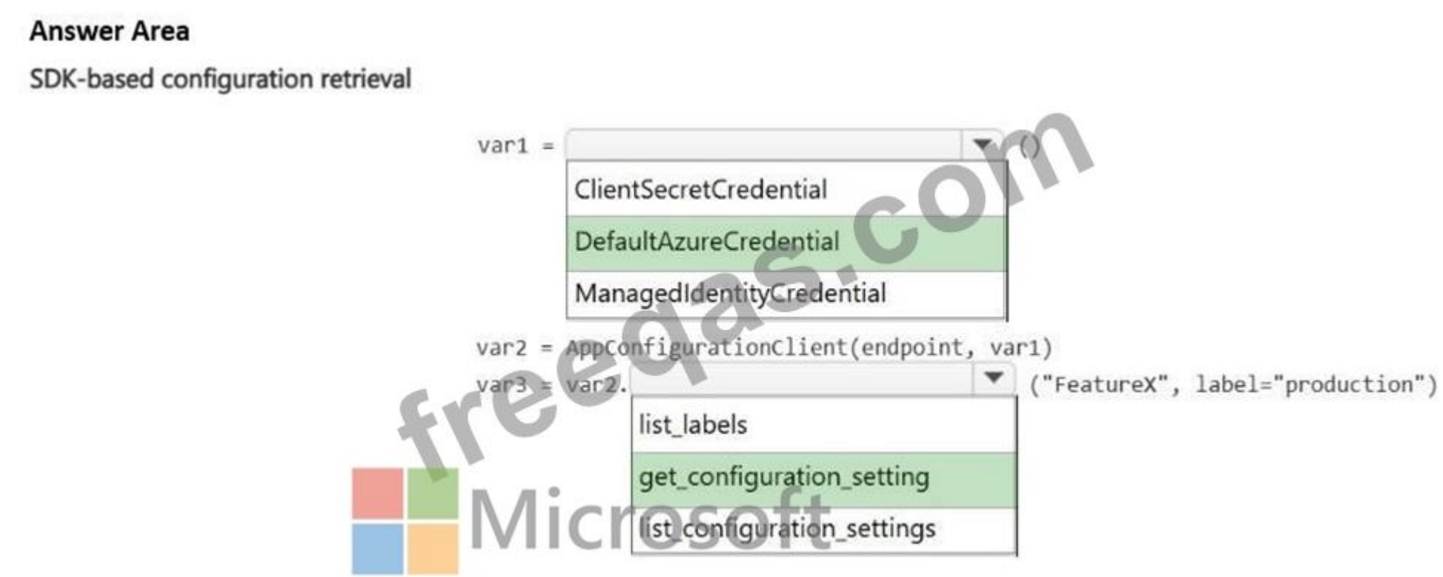
Verified Answer: Scale out the application: HorizontalPodAutoscaler. Expose internally only: ClusterIP service.
Detailed Explanation: The Kubernetes HorizontalPodAutoscaler increases or decreases pod replicas in response to CPU, memory, or other metrics, so it is the resource that implements CPU-driven pod scale-out. A ClusterIP service creates an internal service IP that is reachable within the Kubernetes cluster and is the normal choice for an internal-only service. A Deployment defines the workload but does not itself implement metric-driven autoscaling, while Ingress is intended to route inbound network traffic.
Study Guide Alignment: Containerized Azure workloads: registry builds, App Service containers, Container Apps revision/scaling behavior, and AKS deployment choices.
Official Microsoft Learn References: AI-200 Study Guide | AKS scaling overview | Kubernetes services in AKS

NEW QUESTION: 29
You are developing an AI search API that caches semantic search results in Redis.
Search results must remain cached for 10 minutes. If the underlying data changes, cached entries must NOT be returned.
You need to implement a cache aside strategy to ensure data consistency.
Which two actions should you perform? Each correct answer presents part of the solution. Choose two NOTE: Each correct selection is worth one point.
A. Configure a 10-minute Time to Live on each key.
B. Implement sliding expiration based on key access.
C. Configure a cache notification for key space events.
D. Delete related cache keys when the source data changes
Answer: A,D (LEAVE A REPLY)
Detailed Explanation: The cache-aside design needs both bounded lifetime and explicit invalidation. A ten- minute TTL ensures cached search results do not persist indefinitely, while deleting related keys when source data changes prevents known-stale entries from being returned during that ten-minute window. Sliding expiration would extend lifetime based on access and can preserve stale data. Keyspace notifications report Redis key events; they do not replace application logic that invalidates entries when the authoritative source changes.
Study Guide Alignment: AI data-management workloads: Cosmos DB, PostgreSQL, caching, vector storage, vector retrieval, consistency, and connection optimization.
Official Microsoft Learn References: AI-200 Study Guide | Azure Managed Redis documentation

NEW QUESTION: 30
Hotspot Question
You are using Python SDK to develop a containerized AI application that retrieves the value of a runtime setting stored in an Azure App Configuration resource. The application will run in Azure in the security context of a managed identity.
The application must work in a local development environment without any code changes.
You need to complete the code that implements the retrieval.
How should you complete the code? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

Box 1: DefaultAzureCredential

This class from the azure-identity library provides an automated, sequenced token authentication flow. When running locally, it automatically falls back to your local development environment credentials (such as Azure CLI, environment variables, or VS Code login). Once deployed to Azure, it seamlessly switches to the environment's managed identity without requiring any code changes.

Box 2: get_configuration_setting

This method belongs to the AppConfigurationClient in the azure-appconfiguration SDK. It is used to fetch a specific configuration key-value pair using its key name ("FeatureX") and an optional label.

Reference:

https://learn.microsoft.com/en-us/azure/developer/python/tutorial-containerize-deploy-python-web-app-azure-04

NEW QUESTION: 31

Your solution must answer questions about numeric data in large Excel-based reports (e.g., "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: B (LEAVE A REPLY)

Precise numeric/tabular lookups are handled poorly by vector similarity search or free-text embedding of numbers. Structuring the data (e.g., in a database) and letting a function-calling agent issue precise queries yields accurate, verifiable answers.

Valid AI-200 Dumps shared by PrepPdf.com for Helping Passing AI-200 Exam! PrepPdf.com now offer the **newest AI-200 exam dumps**, the PrepPdf.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** PrepPdf.com AI-200 dumps with Test Engine here: https://www.preppdf.com/Microsoft/AI-200-prepaway-exam-dumps.html (**144** Q&As Dumps, **40%OFF** **Special Discount: Exam-Tests**)

NEW QUESTION: 32

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: 33

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: Store the API key as an App Service application setting configured through the Azure Portal.

Does the solution meet the goal?

A. No

B. Yes

Answer: A (LEAVE A REPLY)

NEW QUESTION: 34

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

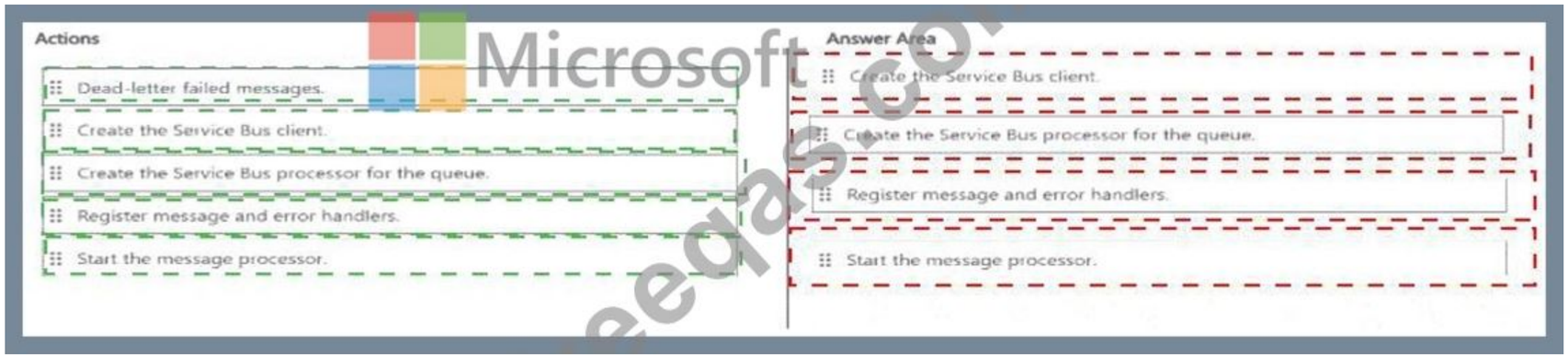
The processor must connect to an existing queue, register handlers for message and error processing, and then begin receiving messages.

You need to provision and configure the Service Bus processor for message and error handling.

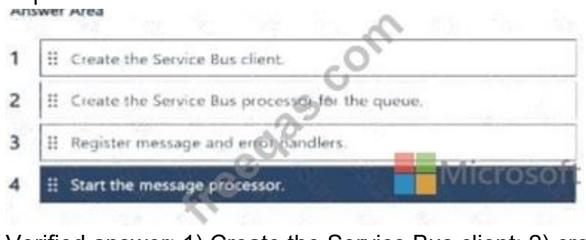
Which four actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.



Answer:



Explanation:



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: 35
Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals.
You are using Azure Monitor Application Insights to investigate a production API. You open the Logs blade and set the time range to Last 24 hours.
An engineer recommends the following query to count requests by result code and sort the results from most frequent to least frequent:

```
requests
| summarize request_count = count() by resultCode
| order by request_count desc
```

You need to determine whether the query returns the number of requests grouped by result code and sorted from most to least frequent.
Solution: The query lists every individual request along with its result code.
Does the solution meet the goal?
A. Yes
B. No
Answer: B (LEAVE A REPLY)

The solution statement is incorrect because the KQL query does not return every individual request . The summarize operator aggregates input records according to the grouping expression. In this query, count() calculates the number of request records in each group, while by resultCode creates one group for every distinct HTTP result code represented in the selected Application Insights time range. Microsoft documents that summarize produces one output row for each distinct combination of its by expressions.
The resulting dataset therefore contains columns equivalent to resultCode and request_count, not individual request rows. The subsequent order by request_count desc sorts those aggregate groups in descending order, placing the most frequent result code first. Microsoft provides essentially the same Application Insights pattern for displaying the distribution of requests by result code: summarize request counts by result code and order the resulting counts descending.
If the intent were to list individual requests, the query would omit summarize and typically use project to select fields such as timestamp, operation name, and result code.
Therefore, because the proposed solution incorrectly describes the query as listing each individual request, the answer is No .
Study Guide references: Azure Monitor Logs; Application Insights requests; KQL summarize; count(); grouping; descending sorting.

NEW QUESTION: 36
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: A (LEAVE A REPLY)
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

NEW QUESTION: 37
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.
NOTE: Each correct selection is worth one point.
A. Retrieve secrets without specifying a version identifier.
B. Retrieve secrets by specifying a version identifier.
C. Cache the secret for 30 days.
D. Configure a Key Vault rotation policy.
Answer: (SHOW ANSWER)
Configure a Key Vault rotation policy to automate periodic credential generation and updates.
Retrieve secrets without a version identifier (versionless reference) so the app automatically resolves the newest active version.
Incorrect:
[Not B]
To stop manual updates, you should not retrieve secrets by specifying a version identifier, because locking in a version stops the application from seeing newer updates.
Reference:

https://learn.microsoft.com/en-us/azure/key-vault/secrets/secure-secrets

NEW QUESTION: 38
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.



Answer:



Explanation:



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: 39
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: B (LEAVE A REPLY)
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.

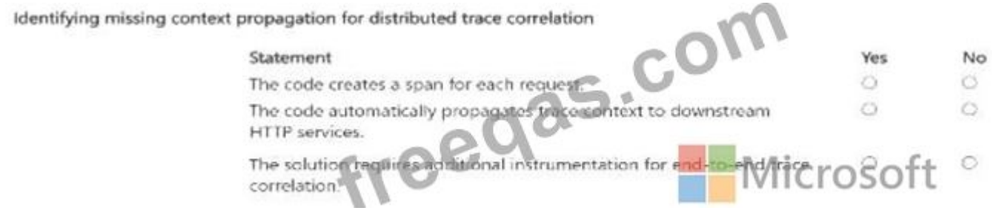
NEW QUESTION: 40
A RAG application's retrieval step is returning documents that are topically related but not precisely relevant to the user's question. You need to improve retrieval precision without retraining any model. What should you do?
A. Increase the embedding model's dimensionality only
B. Enable semantic ranking (re-ranking) in Azure AI Search
C. Disable vector search and use only keyword search
D. Reduce the number of indexed documents
Answer: B (LEAVE A REPLY)
Azure AI Search's semantic ranker re-scores the initial retrieval results using a secondary language-understanding model, improving precision by pushing more contextually relevant results to the top without any custom model training.

NEW QUESTION: 41
You plan to deploy an Azure Container app.
You need to configure the container app to support session affinity.
Which ingress type and revision mode should you assign to the container app?
A. TCP ingress type and single revision mode
B. TCP ingress type and multiple revision mode
C. HTTP ingress type and multiple revision mode
D. HTTP ingress type and single revision mode
Answer: (SHOW ANSWER)
HTTP ingress type and single revision mode must be assigned. Session affinity (sticky sessions) requires an HTTP-based proxy layer to track and route client requests to the same replica, which is incompatible with raw TCP ingress. Single revision mode ensures standard traffic flow without multi-revision splitting complexity.
Reference:
https://learn.microsoft.com/en-us/azure/container-apps/ingress-environment-configuration

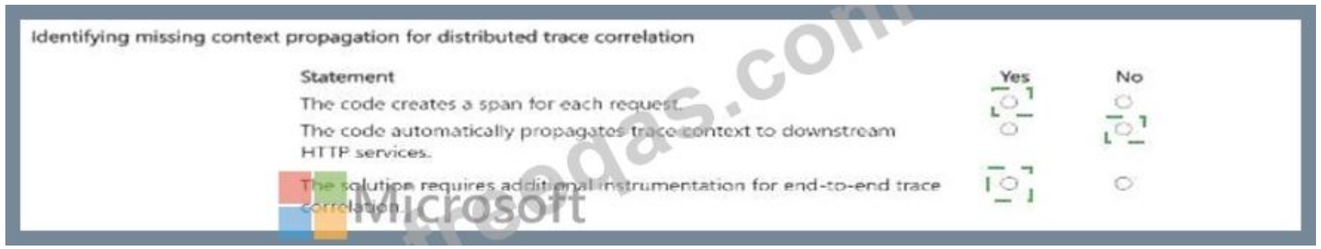
NEW QUESTION: 42
A python web API uses OpenTelemetry for tracing. The call downstream service function makes an outbound HTTP request by using the requests library. The following code is the only OpenTelemetry configuration in the application:

```
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()
```

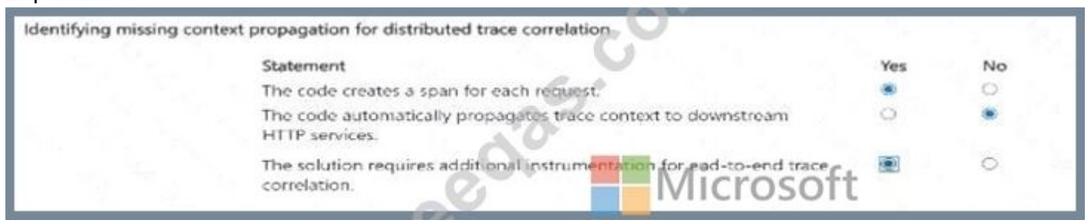
For each of the following statements, select Yes if the statement is true. Otherwise, select No.
NOTE: Each correct selectin is worth one point.



Answer:



Explanation:



NEW QUESTION: 43
Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear on the review screen.
You are using Azure Monitor Application Insights to investigate a production API. You open the Logs blade and set the time range to Last 24 hours.
An engineer recommends the following query to count requests by result code and sort the results from most frequent to least frequent:
requests
| summarize request_count = count() by resultCode
| order by request_count desc
You need to determine whether the query returns the number of requests grouped by result code and sorted from most to least frequent.
Solution: The query lists every individual request along with its result code.
Does the solution meet the goal?
A. Yes

B. No
Answer: (SHOW ANSWER)
Correct:
* The query returns one row per unique resultCode value with the number of requests in each group.
The Kusto Query Language (KQL) query uses the summarize operator, which acts as a grouping and aggregation mechanism.
summarize request_count = count() by resultCode
This groups all the individual rows in the requests table by their unique resultCode. It then counts the total number of logs within each group and places that value into a new column called request_count.
order by request_count desc: This sorts those aggregated rows so that the resultCode with the highest number of requests appears at the top.
Incorrect:
* The result codes are sorted alphabetically.
* The query lists every individual request along with its result code.
Reference:
https://learn.microsoft.com/en-us/azure/azure-monitor/reference/tables/apprequests

NEW QUESTION: 44
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: 45
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: B (LEAVE A REPLY)
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: 46
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 custom topic. Create an event subscription for each customer.
Does the solution meet the goal?
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Detailed Explanation: A single custom topic is a user-defined publishing endpoint, but creating thousands of customer subscriptions on one custom topic is not the architecture Microsoft provides for managing thousands of isolated topics behind one publisher endpoint. An Event Grid domain is the service construct designed for that scale and partitioning model. The requirement for Microsoft Entra authentication can also be satisfied when publishing to Event Grid resources, but the custom-topic design still misses the domain-level multi- topic 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 | Azure Event Grid event domains | Authenticate Event Grid publishing with Microsoft Entra ID

Valid AI-200 Dumps shared by PrepPdf.com for Helping Passing AI-200 Exam! PrepPdf.com now offer the **newest AI-200 exam dumps**, the PrepPdf.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** PrepPdf.com AI-200 dumps with Test Engine here: https://www.preppdf.com/Microsoft/AI-200-prepaway-exam-dumps.html (**144** Q&As Dumps, **40%OFF** **Special Discount: Exam-Tests**)

NEW QUESTION: 47
You need to deploy a batch embedding workload according to the planned application architecture.
What should you use?
A. YAML-formatted files

B. kubectl run and create commands
C. XML-formatted files
D. az aks commands
Answer: A (LEAVE A REPLY)
The batch embedding workload is planned to run in Azure Kubernetes Service (AKS) , and Kubernetes workloads should be defined declaratively by using YAML manifest files . Microsoft states that a Kubernetes manifest defines the desired state of cluster resources, including Deployments, Jobs, Services, ConfigMaps, and other Kubernetes objects. These manifests can then be deployed by using kubectl apply -f < manifest > .
yaml.
For this case, YAML is particularly appropriate because the technical requirements emphasize repeatable, version-controlled, automated deployments . A batch vector embedding regeneration process would typically be represented as a Kubernetes workload such as a Job or another batch-oriented resource, with its configuration committed to Git alongside the rest of the deployment definitions.
kubectl run and kubectl create are imperative commands and are less appropriate for a controlled Git-based deployment model because they do not inherently preserve the complete desired-state definition as source-controlled configuration. XML is not the standard Kubernetes manifest format. az aks commands are primarily used to provision and manage the AKS cluster itself rather than declaratively define application workloads running inside the cluster.
Therefore, the correct choice is YAML-formatted Kubernetes manifest files .
Study Guide references: Azure Kubernetes Service # Kubernetes manifests; declarative deployments; kubectl apply; Git-based workload configuration.

NEW QUESTION: 48
Drag and Drop Question
A recommendation system stores 3 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.

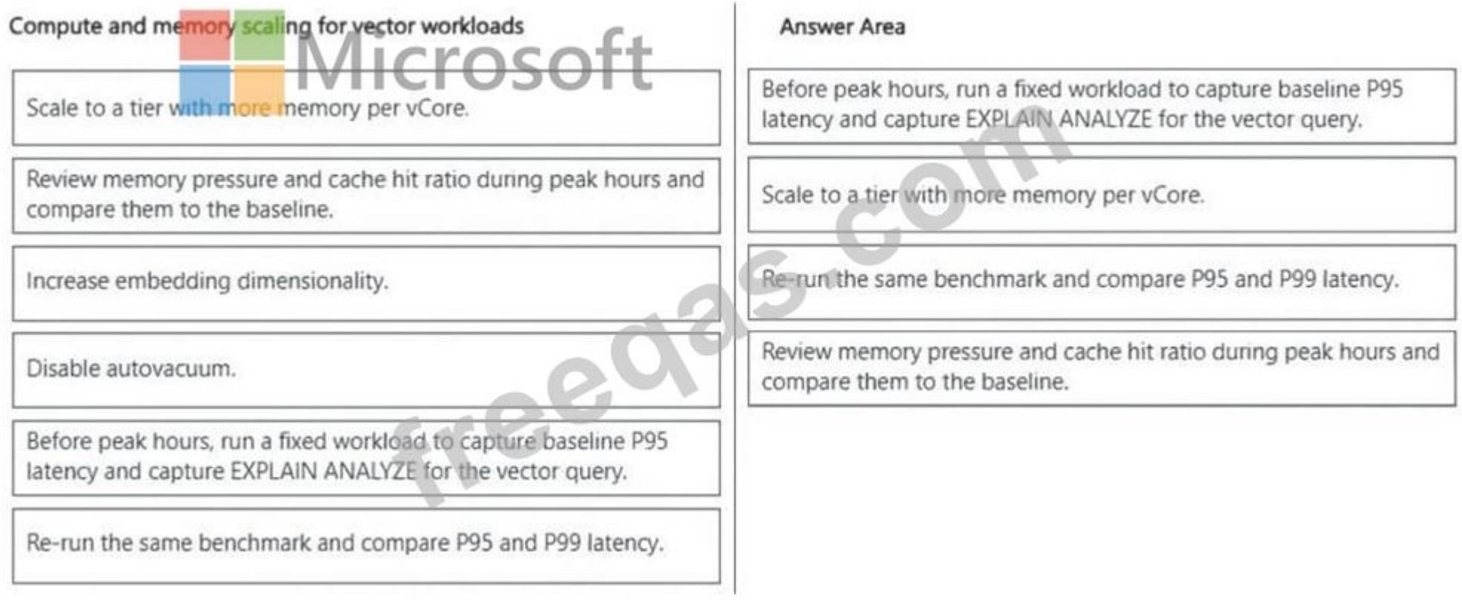
Compute and memory scaling for vector workloads

- 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.
- Increase embedding dimensionality.
- 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.

Answer Area

Microsoft
freeqas.com

Answer:



Microsoft
freeqas.com

Explanation:

Step 1: Before peak hours, run a fixed workload to capture baseline P95 latency and capture EXPLAIN ANALYZE for the vector query Establishing a baseline is crucial for performance tuning. Running EXPLAIN ANALYZE provides deep insight into whether the query is actually using the vector index (like HNSW or IVFFlat) or falling back to a slow sequential scan.

Step 2: Scale to a tier with more memory per vCore

Since the problem states that the vector index working set no longer fits in memory, scaling to a tier with higher memory (such as Memory-Optimized tiers) directly solves the root cause by allowing the RAM to fully cache the vector index, eliminating slow disk reads.

Step 3: Re-run the same benchmark and compare P95 and P99 latency

To validate the outcome of your scaling operations, you must run the identical benchmark workload and compare high-percentile latencies (P95, P99) against your baseline to prove the bottleneck was resolved.

Step 4: Review memory pressure and cache hit ratio during peak hours and compare them to the baseline This step confirms the success of the resource scaling. A successful remediation will show a significantly increased cache hit ratio and reduced memory pressure/disk I/O compared to the baseline metrics.

Reference:

https://www.pgedge.com/blog/ai-features-in-pgadmin-ai-insights-for-explain-plans

NEW QUESTION: 49

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: 50
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.



Answer:

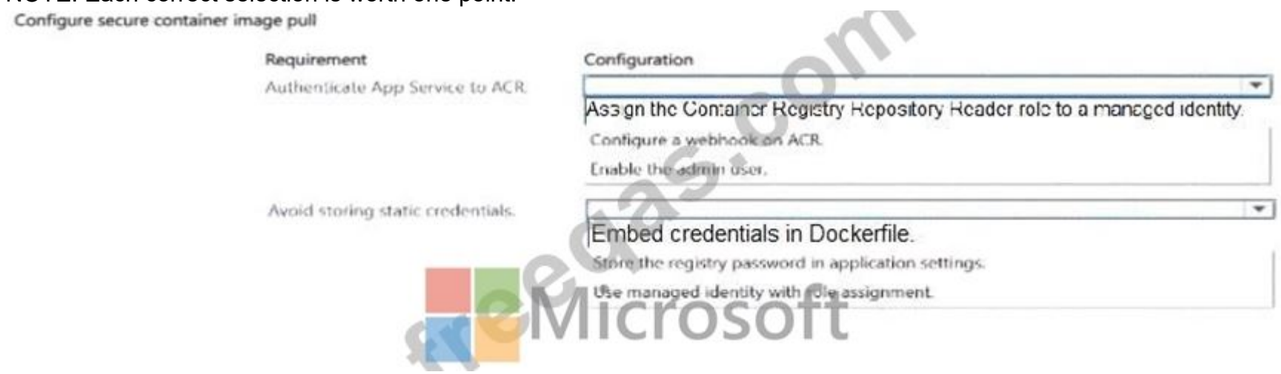


Explanation:

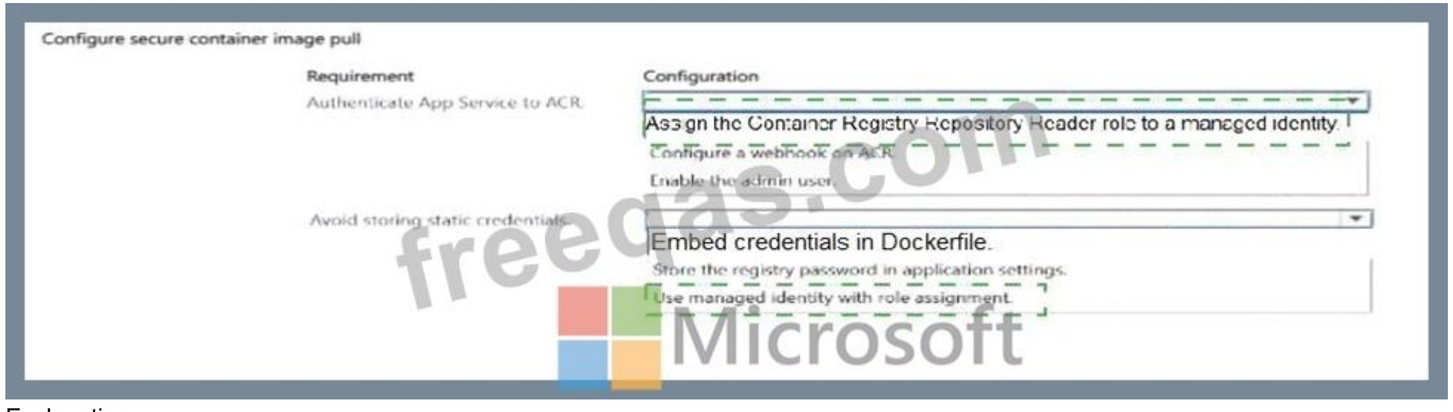
oncurrent active versions: Multiple revision mode. Route a percentage to the secondary version: Traffic splitting.
Detailed Explanation: Azure Container Apps supports side-by-side application versions through multiple revision mode. Once multiple revisions can be active, ingress traffic can be distributed 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: 51
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.



Answer:



Explanation:



uthenticate 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: 52
Drag and Drop Question
You are developing a new page for a website that uses Azure Cosmos DB for data storage. The feature uses documents that have the following format:

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

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

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

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


JSON segments
orderBy
sortOrder
ascending
descending
compositeIndexes
Answer Area
Microsoft
{
"automatic": true,
"ngMode": "Consistent",
"includedPaths": [
{
"path": "/*"
}
], "excludedPaths": [ ],
" JSON segment ": [
[
{
"path": "/name", "order": "descending"
},
{
"path": "/city", "order": " JSON segment "
}
]
]
}
freeqas.com

Answer:



Explanation:

Box 1: compositeIndexes

Azure Cosmos DB requires a defined composite index for any query utilizing an ORDER BY clause with multiple properties.

Box 2: descending

In an ORDER BY clause, properties without an explicit direction default to ascending order. Your query ORDER BY p.name, p.city DESC translates to an ascending sort on /name and a descending sort on /city.

Reference:

https://docs.azure.cn/en-us/cosmos-db/index-policy

NEW QUESTION: 53

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear on the review screen.

You are using Azure Monitor Application Insights to investigate a production API. You open the Logs blade and set the time range to Last 24 hours.

An engineer recommends the following query to count requests by result code and sort the results from most frequent to least frequent:

requests

| summarize request_count = count() by resultCode

| order by request_count desc

You need to determine whether the query returns the number of requests grouped by result code and sorted from most to least frequent.

Solution: The query returns one row per unique resultCode value with the number of requests in each group.

Does the solution meet the goal?

A. Yes

B. No

Answer: (SHOW ANSWER)

Correct:

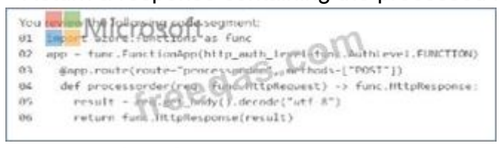
* The query returns one row per unique resultCode value with the number of requests in each group.
The Kusto Query Language (KQL) query uses the summarize operator, which acts as a grouping and aggregation mechanism.
summarize request_count = count() by resultCode
This groups all the individual rows in the requests table by their unique resultCode. It then counts the total number of logs within each group and places that value into a new column called request_count.
order by request_count desc: This sorts those aggregated rows so that the resultCode with the highest number of requests appears at the top.
Incorrect:
* The result codes are sorted alphabetically.
* The query lists every individual request along with its result code.
Reference:
https://learn.microsoft.com/en-us/azure/azure-monitor/reference/tables/apprequests

NEW QUESTION: 54
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: 55
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. Remove the similarity ORDER BY clause.
B. Create a pgvector index on the embedding column.
C. Increase statement timeout.
D. Store embeddings as JSON.
Answer: B (LEAVE A REPLY)

NEW QUESTION: 56
You are reviewing an Azure Function app that processes incoming order requests for a company. The function must:
* Accept order submissions from an external client application.

* Require controlled access for security
* Return a response containing the processed request payload.



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:



Explanation:

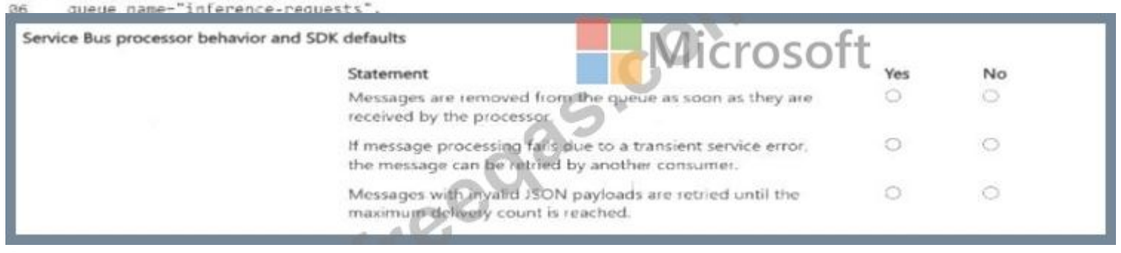
Verified Answer: Yes; No; Yes; No.
Detailed Explanation: `auth_level=FUNCTION` means invocation requires a function key unless a stronger platform authentication layer is configured, so the first statement is true. The route explicitly permits POST, not GET. The function reads `req.get_body()` and returns that value in the HttpResponse, so the response contains the request body. No schema, type, required-field, or other payload validation is performed before the response is constructed, making the final statement false.
Study Guide Alignment: Azure service integration: Service Bus, Event Grid, Azure Functions triggers
/bindings, and event-driven processing.
Official Microsoft Learn References: AI-200 Study Guide | Azure Functions HTTP trigger

NEW QUESTION: 57
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.
NOTE: Each correct selection is worth one point.
A. Store embeddings in a pg vector 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)

NEW QUESTION: 58
You are reviewing the message processing code in a backend worker service.
You review the following Python code that initializes and starts a Service Bus processor

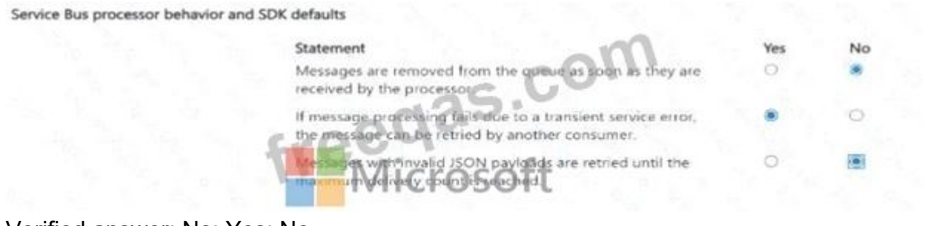
```
from azure.servicebus import ServiceBusClient, ServiceBusReceiveMode

01 import json
02 from azure.servicebus import ServiceBusClient, ServiceBusReceiveMode
03 from azure.identity import DefaultAzureCredential
04 credential = DefaultAzureCredential()
05 with client.get_queue_receiver(
06     queue_name="inference-requests",
```



Answer:



Explanation:



Verified answer: No; Yes; No.
Detailed Explanation: The default Service Bus receive behavior is PeekLock rather than receive-and-delete, so merely receiving a message does not remove it from the queue; it must be settled successfully. If processing fails and the message remains unsettled or the lock is lost, it can become available for another delivery. Invalid JSON, however, is an application-level content problem: Service Bus does not inspect the payload as JSON and automatically retry it to MaxDeliveryCount solely because deserialization fails. The application's settlement/error path determines what happens.
Study Guide Alignment: Azure service integration: Service Bus, Event Grid, Azure Functions triggers

/bindings, and event-driven processing.
Official Microsoft Learn References: AI-200 Study Guide | Service Bus message transfers, locks, and settlement | Service Bus messages, routing, and correlation

NEW QUESTION: 59
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: C (LEAVE A REPLY)
Dead-lettering is the message action you must use to isolate repeatedly failing messages from valid ones in Azure Service Bus.
Isolation: It moves corrupted or malformed messages into a separate sub-queue.
Non-blocking: Valid messages continue processing without getting stuck behind bad data.
Inspection: It allows developers to analyze failed payloads and system errors later.
Reference:
https://azureintegrations.com/2020/03/01/azure-servce-bus-best-practises/

NEW QUESTION: 60
An HTTP-triggered Azure Function must read a JSON configuration file stored in Azure Blob Storage.
You need to retrieve the configuration file contents.
What should you use?
A. Output binding
B. Input binding
C. Timer trigger
D. HTTP trigger
Answer: B (LEAVE A REPLY)
To read a configuration file stored in Azure Blob Storage from an HTTP-triggered function, you must use an Azure Blob storage input binding.
Input bindings allow you to connect external data to your function natively without writing code to explicitly connect, authenticate, and download data using SDK clients.
Strategic Overview
The Trigger: You will use an HTTP Trigger to execute the function via a web request.
The Input: You will use a Blob Input Binding pointing to the specific JSON path in your storage account.
The Result: The Azure Functions runtime automatically retrieves the file contents and provides them directly to your function code as a string, byte array, or parsed object parameter.
Reference:
https://learn.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings

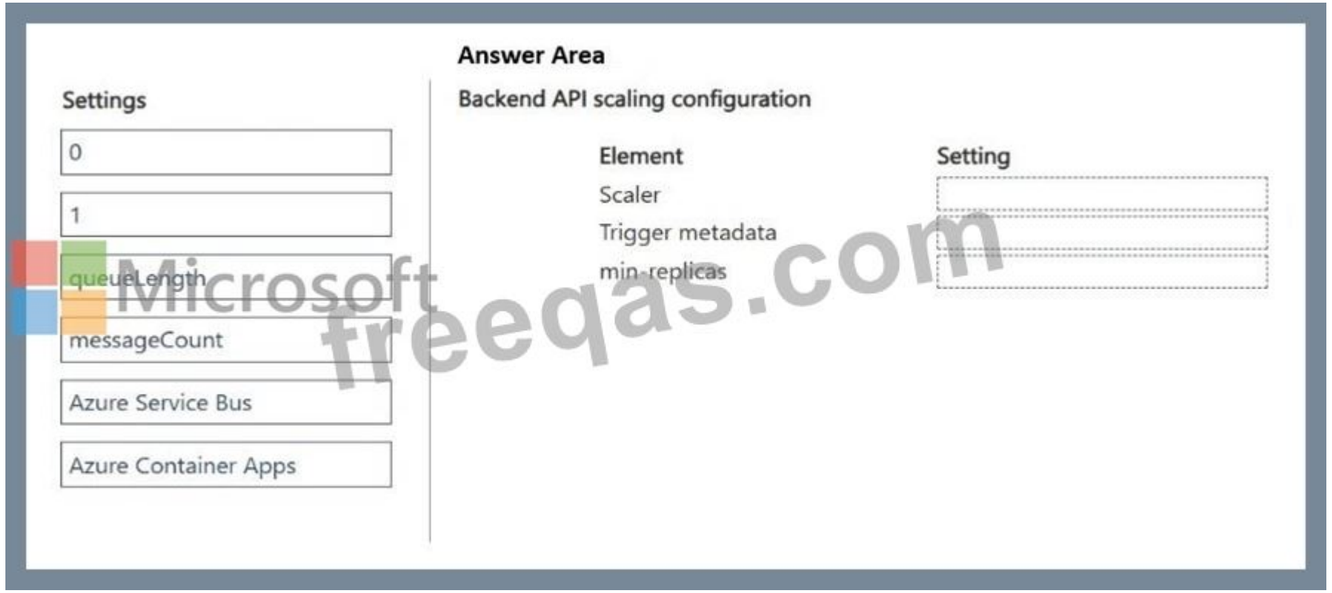
NEW QUESTION: 61
Your application must classify uploaded product images into one of 40 custom categories specific to your business (e.g., proprietary part numbers). What should you use?
A. Azure AI Vision prebuilt image analysis

B. Azure AI Custom Vision (image classification)
C. Azure AI Document Intelligence
D. Azure AI Face API
Answer: (SHOW ANSWER)
Prebuilt Vision models recognize general objects/scenes but not business-specific categories.
Custom Vision allows you to train a classifier on your own labeled images for domain-specific categories such as proprietary part numbers.

Valid AI-200 Dumps shared by PrepPdf.com for Helping Passing AI-200 Exam! PrepPdf.com now offer the **newest AI-200 exam dumps**, the PrepPdf.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** PrepPdf.com AI-200 dumps with Test Engine here: https://www.preppdf.com/Microsoft/AI-200-prepaway-exam-dumps.html (**144** Q&As Dumps, **40%OFF** **Special Discount: Exam-Tests**)

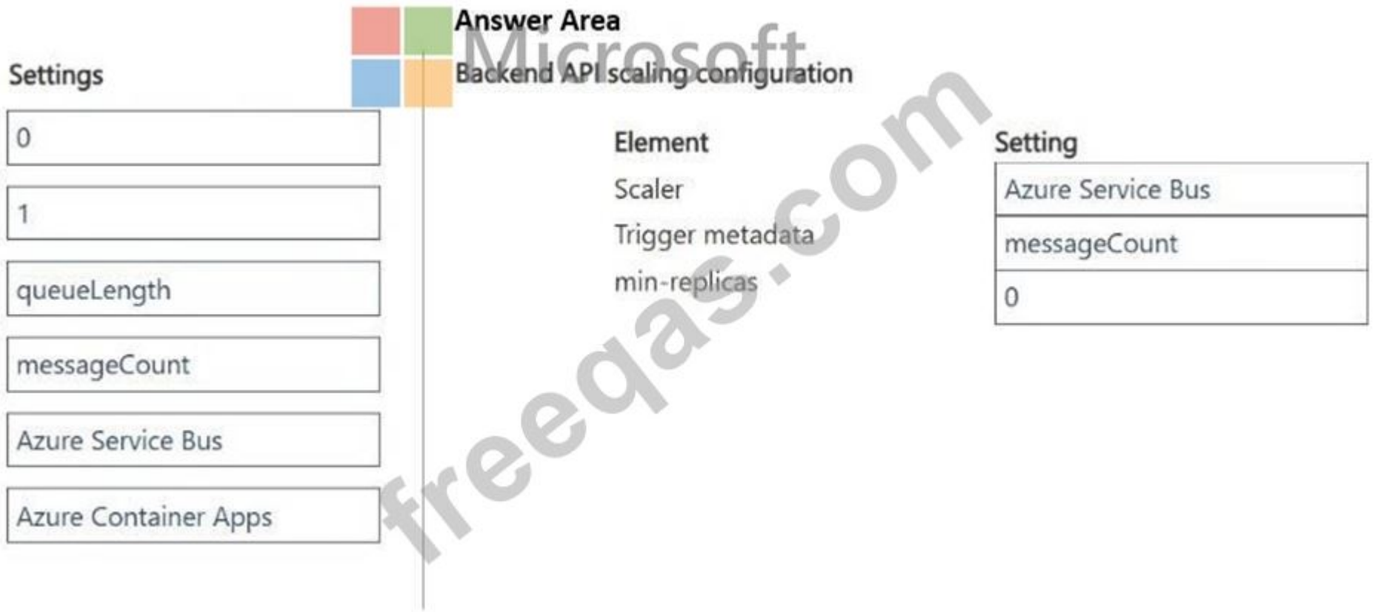
NEW QUESTION: 62
Case Study 2 - Proseware Inc.
Background
Proseware Inc. develops AI-powered knowledge management solutions for enterprise customers.
The company is modernizing its platform to support semantic search, intelligent document retrieval, and real-time partner integrations.
The engineering team uses Python and Azure SDKs. The architecture is being redesigned to support containerized microservices, vector search workloads, and serverless backend processing.
Planned Application Architecture
Microservices are containerized by using Docker.
Code for containerized microservices and Azure Function apps is developed locally but stored in a GitHub repository.
Custom images for containerized microservices are stored in Azure Container Registry (ACR).
Base images are stored in Docker Hub. Custom images must be rebuilt automatically whenever their base images are updated.
Azure Cosmos DB for NoSQL stores documents, metadata, and vector embeddings.
Azure Functions generate vector embeddings of Azure Cosmos DB for NoSQL-hosted documents and send messages to Service Bus to trigger search index updates.
Azure Container Apps (ACA) apps host backend API services that provide semantic search across Azure Cosmos DB for NoSQL documents. API services process Service Bus messages and update search indexes.
Azure Kubernetes Service (AKS) processes batch vector embedding regeneration for existing Azure Cosmos DB for NoSQL documents (whenever the embedding model is changed).
An extranet-facing containerized webhook allows business partners to submit documents to be processed by internal AI workflows for semantic search and retrieval.
Monitoring
Telemetry generated by Azure resources is sent to Azure Monitor.
A Log Analytics workspace is used to collect ACA apps logs, AKS container logs, and Azure Functions apps logs.
Monitoring of Azure Functions is currently implemented by using Azure Application Insights SDK instrumentation.
Business Requirements
Embeddings for new or updated Azure Cosmos DB for NoSQL-hosted documents must be automatically generated.
Backend API services must scale automatically during business hours.
Cold start delay of backend APIs must be minimized.
Secrets must be stored outside of container images.
Developers must be able to correlate telemetry across Azure Functions hosts and apps.
All tracing must be implemented by using OpenTelemetry SDK instrumentation.
Development efforts must be minimized.

Technical Requirements
Container images must be built automatically and validated before code updates are merged into the main branch.
Image build automation must run inside the Azure Container Registry, eliminating dependency on local developer machines and external build services.
Dependency of image builds on local developer machines must be eliminated.
Event-driven scaling in ACA must occur based on the number of pending messages in the Azure Service Bus queue.
Azure Cosmos DB for NoSQL RU consumption must be minimized.
Vector similarity search must use embeddings stored in Azure Cosmos DB for NoSQL.
The partner-facing containerized webhook service must run on Azure App Service.
Secrets must NOT be stored in container images, source control, or application configuration directly. They must be accessed securely at runtime.
All secrets must be stored centrally in Azure Key Vault and accessed at runtime through a managed identity.
Azure App Service must supply secrets at runtime without relying on external services.
Resources and workloads must be deployed by using Bicep templates through an automated, version-controlled pipeline. Local and command-line deployments must be eliminated to ensure repeatable, auditable deployments.
Known Issues
RU consumption spikes during vector similarity queries.
Drag and Drop Question
You need to configure event-driven scaling for the backend API services to meet the technical requirements.
Which settings should you use for each element? To answer, move the appropriate settings to the correct elements. You may use each setting 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.



Microsoft freeqas.com

Answer:



Microsoft
freeqas.com

Explanation:

Scenario, technical requirements.

Event-driven scaling in ACA must occur based on the number of pending messages in the Azure Service Bus queue.

Box 1: Azure Service Bus

Scaler TypeType

Set the scale rule type to custom.

KEDA Scaler Name: Set the type inside the custom specification block to azure-service-bus.

Box 2: messageCount

Trigger Metadata Values

The metadata block dictates how KEDA calculates the required replica count.

Configure the following core metadata parameters:

queueName: The explicit string name of your targeted Azure Service Bus

*-> queue.messageCount: The target integer threshold of concurrent pending messages assigned per replica (e.g., 5 or 10). KEDA uses this value to scale out systematically.

activationMessageCount: Set to 1 or 0. This defines the exact metric floor required to transition the container app out of a dormant state.

namespace: The canonical name of your Service Bus namespace

Box 3: 0

Min-Replicas ConfigurationValue

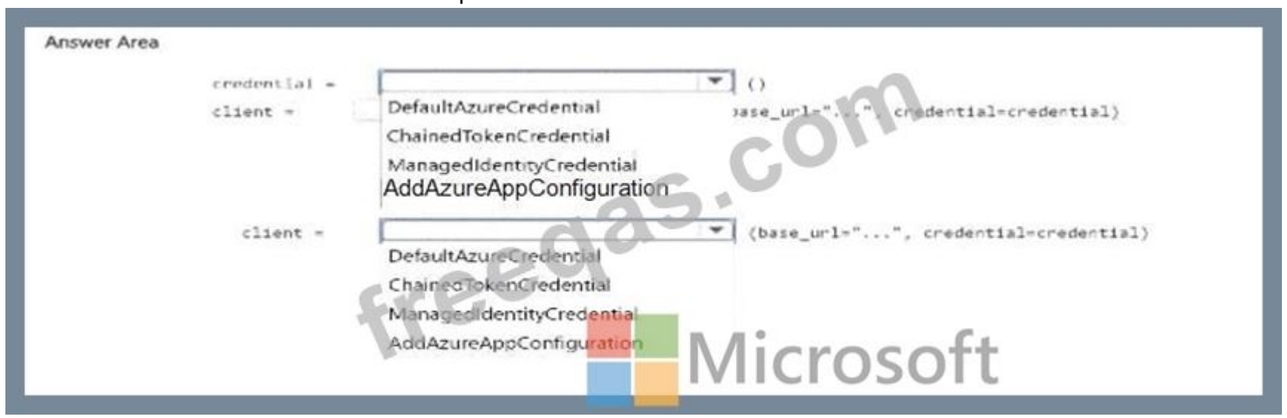
Set --min-replicas to 0.

Setting this value to zero enables full serverless cost efficiency. KEDA actively handles the background polling, allowing your backend API containers to scale down completely and run zero active replicas when the Service Bus queue contains zero pending messages.

Reference:

https://oneuptime.com/blog/post/2026-02-16-how-to-set-up-keda-based-auto-scaling-with-queue-triggers-in-azure-container-apps/view

NEW QUESTION: 63
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:



Explanation:



NEW QUESTION: 64
A company is implementing a publish-subscribe (Pub/Sub) messaging component by using Azure Service Bus. You are developing the first subscription application.

In the Azure portal you see that messages are being sent to the subscription for each topic. You create and initialize a subscription client object by supplying the correct details, but the subscription application is still not consuming the messages.
You need to ensure that the subscription client processes all messages.
Which code segment should you use?
A. subscriptionClient = new SubscriptionClient(ServiceBusConnectionString, TopicName, SubscriptionName);
B. subscriptionClient.RegisterMessageHandler(ProcessMessagesAsync, messageHandlerOptions);
C. await subscriptionClient.AddRuleAsync(new RuleDescription(RuleDescription.DefaultRuleName, new TrueFilter()));
D. await subscriptionClient.CloseAsync();
Answer: B (LEAVE A REPLY)
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: 65
You configure ACR Tasks to automate image builds.
Container images must rebuild when:
- Application updates occur.
- Base image updates occur, such as when the underlying OS image is
updated.
- Regular scheduled rebuilds are required.
You need to configure ACR Tasks to support automated image rebuilds.
Which three triggers should you configure? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.
A. Timer trigger
B. Source code commit trigger
C. Registry event trigger
D. Base image update trigger
E. Webhook notification trigger
Answer: (SHOW ANSWER)
To support automated container image rebuilds based on your requirements, you need to configure source triggers, base image triggers, and timer triggers in Azure Container Registry (ACR) Tasks.
Timer Triggers: Automates regular scheduled rebuilds. This uses a cron schedule expression to run image builds at specific intervals (e.g., weekly or monthly) regardless of code changes.
Source Triggers: Automates rebuilds when application updates occur. This monitors changes in your source code repository (like GitHub or Azure Repos) and fires a build when code is committed.
Base Image Triggers: Automates rebuilds when the underlying OS or framework image updates.
ACR tracks dependencies and automatically kicks off a new build when your defined FROM image changes in the public registry or your private registry.
Reference:
https://oneuptime.com/blog/post/2026-02-16-how-to-set-up-acr-tasks-for-automated-container-image-builds-on-git-commit/view

NEW QUESTION: 66

Drag and Drop Question

You are implementing a Retrieval-Augmented Generation (RAG) system by using the native vector search capabilities of Azure Cosmos DB for NoSQL API.

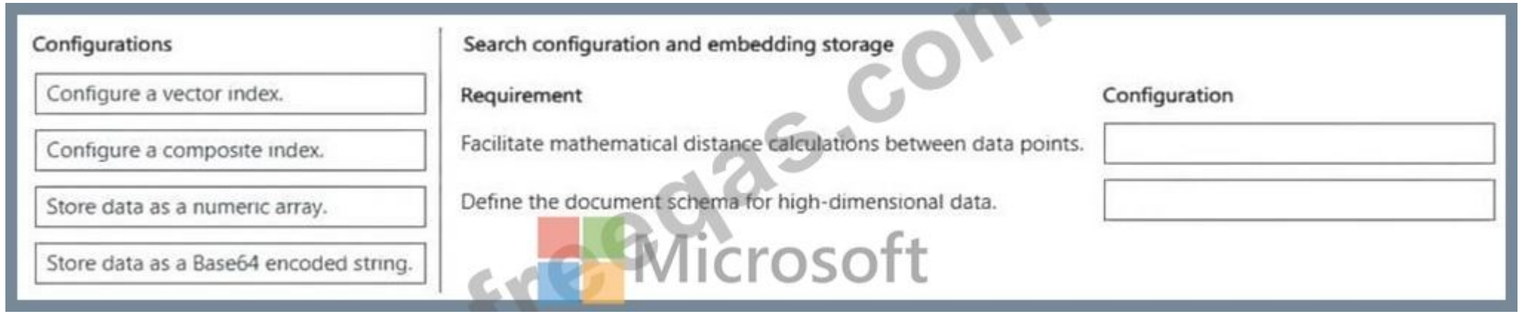
You have a container named Documents that stores technical articles. Each article includes a property named embedding.

You must ensure that the system can perform efficient similarity searches between user queries and the stored articles.

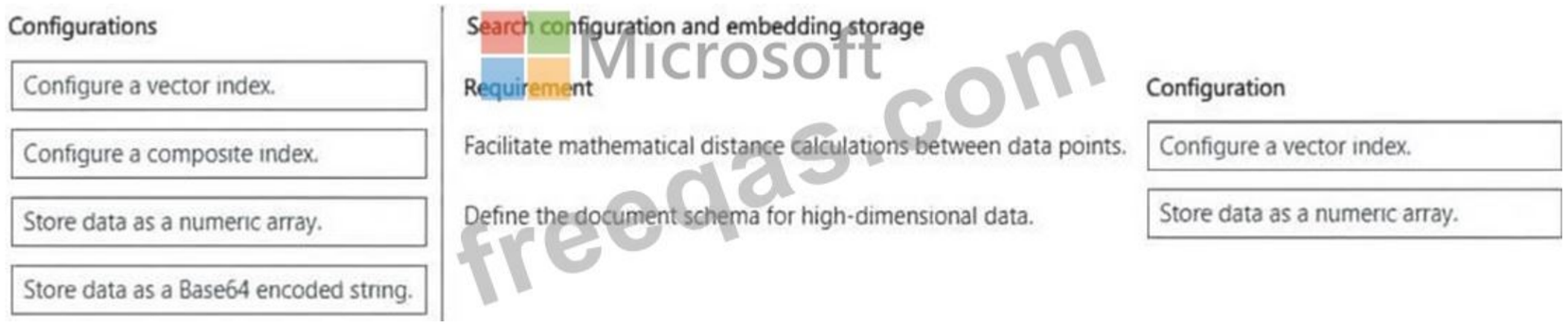
You need to configure the database resources to support semantic retrieval.

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.



Answer:



Explanation:

Box 1: Configure a vector index.

To support semantic retrieval and perform mathematical distance calculations using the native vector search capabilities of Azure Cosmos DB for NoSQL, you must configure a vector index.

Azure Cosmos DB for NoSQL relies on a specialized indexing structure (such as DiskANN or QuantizedFlat) to organize vector data. Defining a vector index policy within the container configurations is a strict requirement for enabling high-speed similarity searches (using functions like VectorDistance) without requiring full, unindexed scans of documents.

Box 2: Store data as a numeric array

Azure Cosmos DB for NoSQL, you must store the embedding property as a numeric array (an array of floating-point numbers like float32) inside your JSON documents.

Reference:

https://learn.microsoft.com/en-us/azure/cosmos-db/gen-ai/agentic-retrieval

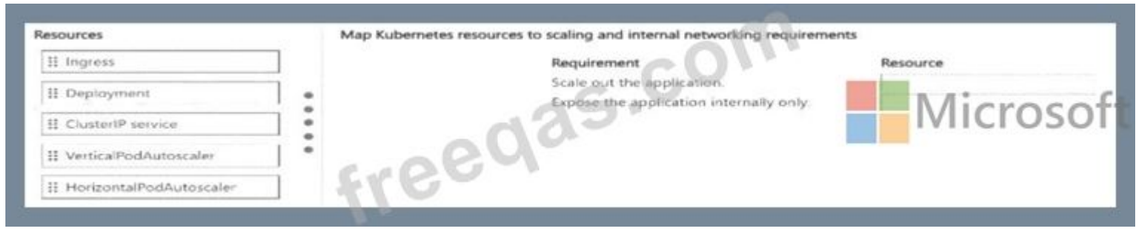
NEW QUESTION: 67

You plan to deploy a web application to AKS.

The solution must:

* Scale out the application by adding more pods during peak CPU usage

* Expose the application internally within the cluster only
You need to configure a Kubernetes resource for each requirement
Which resources should you configure? To answer, move the appropriate resources to the correct requirements You may use each resource once, more than once, or not at all. You may need to move split bar between .. or scroll to view content.



Answer:



Explanation:

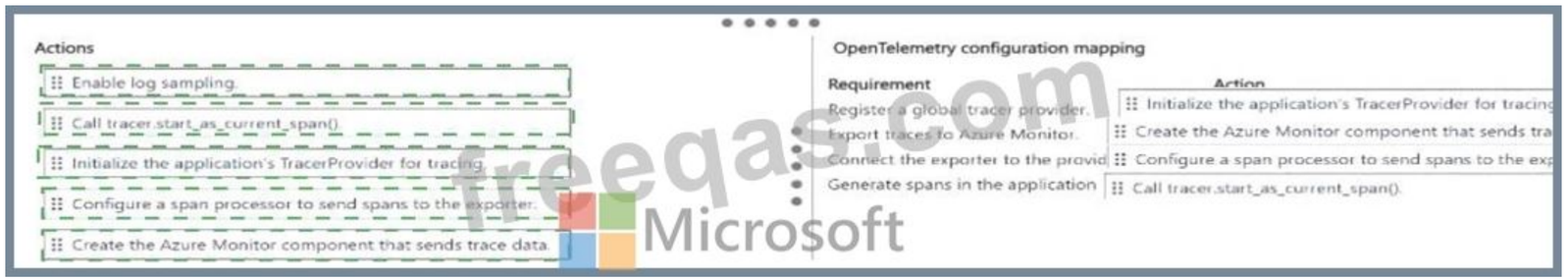


Verified answer: Scale out the application: HorizontalPodAutoscaler. Expose internally only: ClusterIP service.
Detailed Explanation: The Kubernetes HorizontalPodAutoscaler increases or decreases pod replicas in response to CPU, memory, or other metrics, so it is the resource that implements CPU-driven pod scale-out. A ClusterIP service creates an internal service IP that is reachable within the Kubernetes cluster and is the normal choice for an internal-only service. A Deployment defines the workload but does not itself implement metric-driven autoscaling, while Ingress is intended to route inbound network traffic.
Study Guide Alignment: Containerized Azure workloads: registry builds, App Service containers, Container Apps revision/scaling behavior, and AKS deployment choices.
Official Microsoft Learn References: AI-200 Study Guide | AKS scaling overview | Kubernetes services in AKS

NEW QUESTION: 68
A Python API running in ACA must send distributed traces to Azure Monitor.
The API creates spans. However, no traces appear in Azure Monitor.
You need to configure the OpenTelemetry SDK pipeline to export traces to Azure Monitor.
What should you do? To answer, move the appropriate actions to the correct requirements. You may use each action once, more than once, or not at all. You may need to move the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

Verified answer: Register a global trace provider # initialize the application TraceProvider. Export traces to Azure Monitor # create/configure AzureMonitorTraceExporter. Connect exporter to provider # add a span processor. Generate spans # use tracer.start_as_current_span(...).

Detailed Explanation: OpenTelemetry tracing is a pipeline: the application obtains a tracer from a configured provider, creates spans, passes ended spans through a span processor, and exports them using the Azure Monitor exporter. Creating spans alone is insufficient; without the provider/processor/exporter chain, no telemetry reaches Azure Monitor. Log sampling and legacy Application Insights TrackEvent calls are unrelated to the required OpenTelemetry trace-export path.

Study Guide Alignment: Security and operations: Key Vault, App Configuration, managed identity, OpenTelemetry, Azure Monitor, and KQL-based troubleshooting.

Official Microsoft Learn References: AI-200 Study Guide | Enable Azure Monitor OpenTelemetry

NEW QUESTION: 69

Hotspot Question

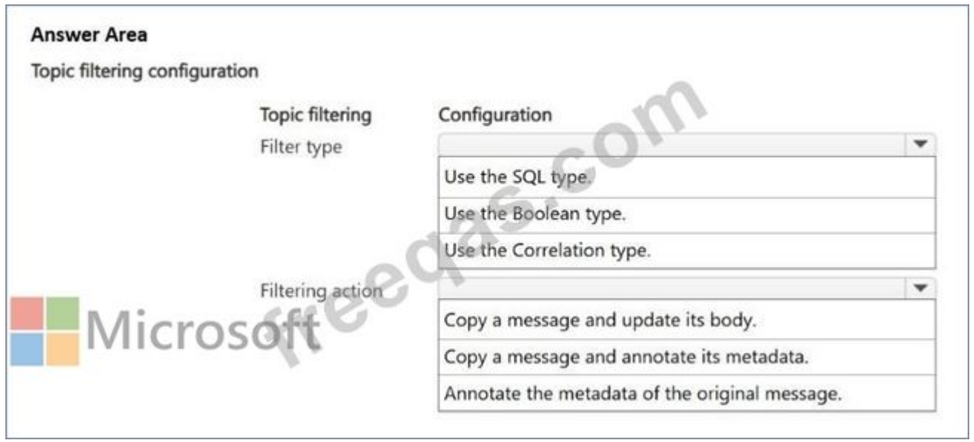
You have an Azure Service Bus namespace that contains a topic named Topic1.

You plan to create a subscription named Sub1 to Topic1. In Sub1, you plan to filter messages from Topic1 based on their system properties and apply an action that will annotate each filtered message.

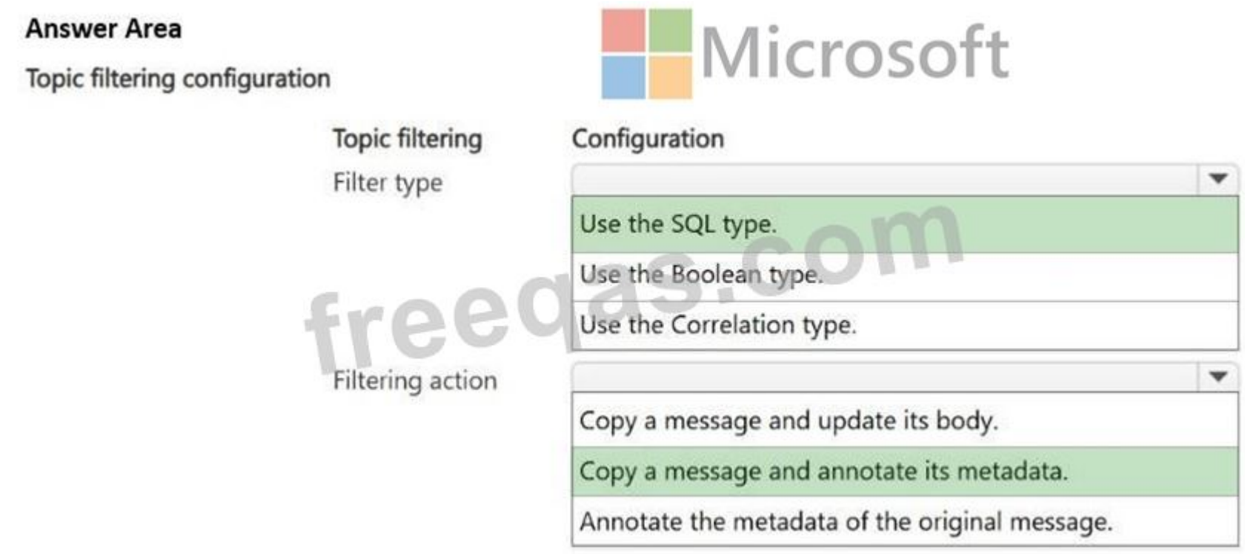
You need to configure the filtering.

How should you configure the filtering? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.



Answer:



Explanation:
Box 1: Use the SQL type.
To achieve this goal, you must use a SQL filter (also known as a SqlFilter) type Box 2: Copy a message and annotate its metadata.
The correct filtering action to use is to copy a message and annotate its metadata.
When an action is applied (such as setting a new property or modifying an existing one via SQL filter actions), Azure Service Bus creates a copy of the inbound message for the subscription and annotates the copy's metadata.
Reference:
https://turbo360.com/blog/azure-service-bus-topic-filter

NEW QUESTION: 70
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: B (LEAVE A REPLY)
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: 71
You deploy multiple instances of a change feed processor to handle a high ingestion rate within Azure Cosmos DB for NoSQL.
Each processor instance must process a different subset of partitions.
You need to ensure the workload is load-balanced across all processor instances.
What should you configure?
A. indexing precision
B. strong consistency
C. lease container
D. autoscale throughput
Answer: C (LEAVE A REPLY)
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: 72
You need to improve throughput for concurrent application requests to PostgreSQL.
What should you implement?
A. Implement connection pooling.
B. Increase shared_buffers.
C. Enable read replicas.
D. Increase max_connections.
Answer: A (LEAVE A REPLY)

The correct solution is to implement connection pooling , ideally by using the built-in PgBouncer capability in Azure Database for PostgreSQL Flexible Server. Connection pooling allows many concurrent application requests to share and reuse a smaller set of established PostgreSQL server connections instead of repeatedly opening and closing database sessions. Microsoft notes that creating a new PostgreSQL connection for each operation consumes server resources because each connection involves its own backend process. Reusing pooled connections therefore reduces connection overhead, lowers CPU and memory consumption, and improves throughput for high-concurrency workloads.
This directly matches Fabrikam ' s stated requirement that database connections support high concurrency with minimal latency through connection optimization . PgBouncer can efficiently multiplex large numbers of client connections onto fewer PostgreSQL connections, particularly in transaction pooling mode.
Increasing max_connections is not the preferred solution because each additional PostgreSQL connection consumes memory and other resources; Microsoft explicitly recommends PgBouncer instead when more concurrent connections are required. Increasing shared_buffers addresses caching and memory allocation rather than connection-management overhead. Read replicas can scale read workloads, but they do not optimize the application ' s connection lifecycle.
Study Guide references: Azure Database for PostgreSQL Flexible Server # connection pooling; PgBouncer; connection management and high-concurrency performance.

NEW QUESTION: 73
Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear on the review screen.
You are using Azure Monitor Application Insights to investigate a production API. You open the Logs blade and set the time range to Last 24 hours.
An engineer recommends the following query to count requests by result code and sort the results from most frequent to least frequent:
requests
| summarize request_count = count() by resultCode
| order by request_count desc
You need to determine whether the query returns the number of requests grouped by result code and sorted from most to least frequent.
Solution: The result codes are sorted alphabetically.
Does the solution meet the goal?
A. Yes
B. No
Answer: B (LEAVE A REPLY)
Correct:
* The query returns one row per unique resultCode value with the number of requests in each group.
The Kusto Query Language (KQL) query uses the summarize operator, which acts as a grouping and aggregation mechanism.
summarize request_count = count() by resultCode
This groups all the individual rows in the requests table by their unique resultCode. It then counts the total number of logs within each group and places that value into a new column called request_count.
order by request_count desc: This sorts those aggregated rows so that the resultCode with the highest number of requests appears at the top.
Incorrect:
* The result codes are sorted alphabetically.
* The query lists every individual request along with its result code.
Reference:
https://learn.microsoft.com/en-us/azure/azure-monitor/reference/tables/apprequests

NEW QUESTION: 74
Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear on the review screen.
You plan to deploy a container to an Azure App Service API app named api1. You host the source code for api1 in a GitHub repository. The container uses the API key at runtime to connect to a backend service.
The container must be able to retrieve the API key at runtime without exposing it in the source repository or Git commit history.
You need to ensure that the API key remains outside of Git commit history and is available to the container at runtime.
Solution: Store the API key in Azure Key Vault and reference it from an App Service application setting.
Does the solution meet the goal?
A. Yes
B. No
Answer: A (LEAVE A REPLY)
Correct:
* Store the API key in Azure Key Vault and reference it from an App Service application setting.
Storing the API key in Azure Key Vault and referencing it via App Service application settings is the recommended, secure approach. This strategy completely removes sensitive credentials from your GitHub repository and Git commit history while injecting them safely into your container environment at runtime.
Incorrect:
* Embed the API key as a hardcoded environment variable in the Dockerfile.
* Store the API key as a GitHub repository secret.
Reference:
https://www.qservicesit.com/full-stack-applications-on-azure

NEW QUESTION: 75
Hotspot Question
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:

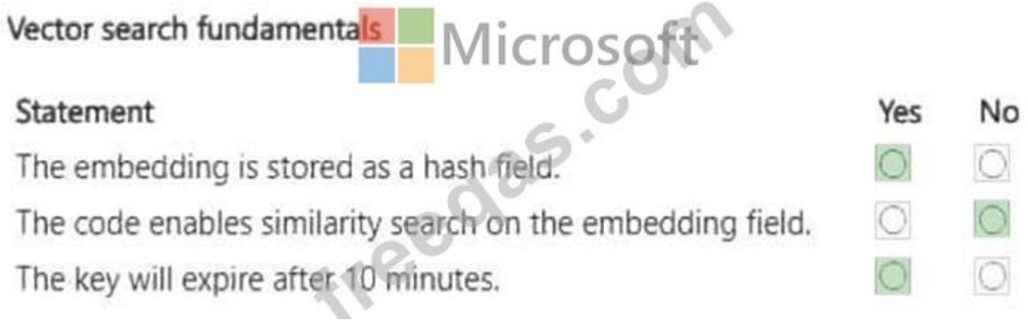
```
01  import numpy as np
02  from redis import Redis
03  embedding = np.array([0.12, 0.98, 0.44], dtype=np.float32).tobytes()
04  r = Redis(
05      host="myredis.redis.cache.windows.net",
06      port=6380,
07      password="accesskey",
08      ssl=True
09  )
10  r.hset("doc:1", mapping={"embedding": embedding})
11  r.expire("doc:1", 600)
```

freeqas.com
Microsoft

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:



Explanation:
Box 1: Yes
The line r.hset("doc:1", mapping={"embedding": embedding}) will store the embedding as a field within a Redis Hash.
Box 2: No
Box 3: Yes
Reference:
https://redis.io/docs/latest/commands/expire/

NEW QUESTION: 76
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: Store the API key as a GitHub repository secret.
Does the solution meet the goal?
A. Yes
B. No
Answer: B (LEAVE A REPLY)

Valid AI-200 Dumps shared by PrepPdf.com for Helping Passing AI-200 Exam! PrepPdf.com now offer the **newest AI-200 exam dumps**, the PrepPdf.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** PrepPdf.com AI-200 dumps with Test Engine here: https://www.preppdf.com/Microsoft/AI-200-prepaway-exam-dumps.html (**144** Q&As Dumps, **40%OFF** **Special Discount: Exam-Tests**)

NEW QUESTION: 77
Case Study 2 - Proseware Inc.
Background
Proseware Inc. develops AI-powered knowledge management solutions for enterprise customers.
The company is modernizing its platform to support semantic search, intelligent document retrieval, and real-time partner integrations.
The engineering team uses Python and Azure SDKs. The architecture is being redesigned to support containerized microservices, vector search workloads, and serverless backend processing.
Planned Application Architecture
Microservices are containerized by using Docker.
Code for containerized microservices and Azure Function apps is developed locally but stored in a GitHub repository.
Custom images for containerized microservices are stored in Azure Container Registry (ACR).
Base images are stored in Docker Hub. Custom images must be rebuilt automatically whenever their base images are updated.
Azure Cosmos DB for NoSQL stores documents, metadata, and vector embeddings.
Azure Functions generate vector embeddings of Azure Cosmos DB for NoSQL-hosted documents and send messages to Service Bus to trigger search index updates.
Azure Container Apps (ACA) apps host backend API services that provide semantic search across Azure Cosmos DB for NoSQL documents. API services process Service Bus messages and update search indexes.
Azure Kubernetes Service (AKS) processes batch vector embedding regeneration for existing Azure Cosmos DB for NoSQL documents (whenever the embedding model is changed).
An extranet-facing containerized webhook allows business partners to submit documents to be processed by internal AI workflows for semantic search and retrieval.
Monitoring
Telemetry generated by Azure resources is sent to Azure Monitor.
A Log Analytics workspace is used to collect ACA apps logs, AKS container logs, and Azure Functions apps logs.
Monitoring of Azure Functions is currently implemented by using Azure Application Insights SDK instrumentation.
Business Requirements
Embeddings for new or updated Azure Cosmos DB for NoSQL-hosted documents must be automatically generated.
Backend API services must scale automatically during business hours.
Cold start delay of backend APIs must be minimized.
Secrets must be stored outside of container images.
Developers must be able to correlate telemetry across Azure Functions hosts and apps.
All tracing must be implemented by using OpenTelemetry SDK instrumentation.
Development efforts must be minimized.
Technical Requirements
Container images must be built automatically and validated before code updates are merged into the main branch.
Image build automation must run inside the Azure Container Registry, eliminating dependency on local developer machines and external build services.
Dependency of image builds on local developer machines must be eliminated.
Event-driven scaling in ACA must occur based on the number of pending messages in the Azure Service Bus queue.
Azure Cosmos DB for NoSQL RU consumption must be minimized.
Vector similarity search must use embeddings stored in Azure Cosmos DB for NoSQL.
The partner-facing containerized webhook service must run on Azure App Service.

Secrets must NOT be stored in container images, source control, or application configuration directly. They must be accessed securely at runtime.
All secrets must be stored centrally in Azure Key Vault and accessed at runtime through a managed identity.
Azure App Service must supply secrets at runtime without relying on external services.
Resources and workloads must be deployed by using Bicep templates through an automated, version-controlled pipeline. Local and command-line deployments must be eliminated to ensure repeatable, auditable deployments.
Known Issues
RU consumption spikes during vector similarity queries.
You need to configure Azure Functions to generate vector embeddings according to planned environment application architecture.
Which trigger type should you use?
A. Event Grid
B. Azure Cosmos DB
C. HTTP
D. Service Bus
Answer: B (LEAVE A REPLY)
The most appropriate trigger type for this architecture is the Azure Cosmos DB Trigger.
This trigger monitors your NoSQL container's underlying change feed. When a document is created or modified, the function fires automatically, processes the data, generates the embedding, and places a message in Azure Service Bus.
Scenario, Planned Application Architecture
Azure Functions generate vector embeddings of Azure Cosmos DB for NoSQL-hosted documents and send messages to Service Bus to trigger search index updates.
Reference:
https://learn.microsoft.com/en-us/azure/cosmos-db/integrated-embeddings

NEW QUESTION: 78
An Azure Storage Blob upload must trigger backend processing.
The solution must:
* Avoid polling.
* Minimize latency.
* Trigger automatically when the upload occurs.
You need to ensure that processing starts as soon as a file is uploaded.
Which trigger should you implement?
A. Timer
B. Event Grid
C. Queue storage
D. HTTP
Answer: (SHOW ANSWER)
Use an Azure Event Grid trigger for event-driven processing of new blob uploads. Azure Blob Storage can publish events such as Microsoft.Storage.BlobCreated to Event Grid, which then pushes the event to the subscribed Azure Function or other event handler. This eliminates the need for the application to periodically scan the storage container for changes.
Microsoft distinguishes traditional polling-based Blob triggers from event-driven Blob/Event Grid processing . Event Grid-based processing provides low-latency, near-real-time delivery and supports high- scale workloads.
Microsoft specifically notes that the Event Grid source eliminates polling delays and can process newly created blobs within seconds of upload.
A Timer trigger would explicitly introduce polling on a schedule. A Queue Storage trigger requires another component to place a message in a queue when the blob arrives, so it is not the direct event source described.
An HTTP trigger requires an external caller to invoke the function and therefore does not automatically react to a Blob Storage creation event.

Therefore, Event Grid is the correct trigger for automatic, low-latency, non-polling blob processing.
Study Guide references: Azure Functions # Event Grid triggers; Azure Blob Storage events; event-driven Blob processing; BlobCreated events; near-real-time processing.

NEW QUESTION: 79
An application deployed to AKS depends on an internal API hosted in the same cluster. All pods are healthy and report Ready status, but requests between the services time out.
You need to determine whether the issue is related to Kubernetes service configuration or application code.
What should you do first?
A. Restart the failing deployment.
B. Inspect Service and Endpoints objects.
C. Test DNS resolution from your workstation.
D. Increase replica count for both services.
Answer: B (LEAVE A REPLY)

NEW QUESTION: 80
Hotspot Question
You are reviewing the message-processing code in a backend worker service.
You review the following Python code that initializes and starts a Service Bus processor.

```
from azure.servicebus import ServiceBusClient, ServiceBusReceiveMode
```

```
01 import json
02 from azure.servicebus import ServiceBusClient, ServiceBusReceiveMode
03 from azure.identity import DefaultAzureCredential
04 credential = DefaultAzureCredential()
05 with client.get_queue_receiver(
06    queue_name="inference-requests",
07    receive_mode=ServiceBusReceiveMode.PEEK_LOCK,
08    max_wait_time=30
09 ) as receiver:
10     for msg in receiver:
11         try:
12             payload = json.loads(str(msg))
13             result = run_inference(payload)
14             save_result(result, payload.get("document_id"))
15             receiver.complete_message(msg)
16         except json.JSONDecodeError:
17             receiver.dead_letter_message(
18                 msg,
19                 reason="MalformedPayload",
20                 error_description="Message body is not valid JSON"
21             )
22         except ModelNotFoundError:
23             receiver.dead_letter_message(
24                 msg,
25                 reason="UnsupportedModel",
26                 error_description=f"Model {payload.get('model')} is not available"
27             )
28         except TransientServiceError:
29             receiver.abandon_message(msg)
```

Microsoft
freegas.com

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

Service Bus processor behavior and SDK defaults

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

Answer:

Answer Area

Service Bus processor behavior and SDK defaults

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

Explanation:

Box 1: No

No, messages are not removed from the queue as soon as they are received when using PEEK_LOCK mode.

How PEEK_LOCK Works

The Lock: The Service Bus processor locks the message exclusively for your worker for a specific duration.

The Invisible State: Other workers cannot see or process the message while the lock is active.

The Removal: The message is only permanently removed from the queue after it is successfully processed and explicitly completed.

Box 2: Yes

Yes, the message can be retried by another consumer if it fails due to a TransientServiceError.

Explicit abandonment: The code calls receiver.abandon_message(msg) when catching a TransientServiceError.

Lock release: Abandoning a message immediately releases the PEEK_LOCK held by the current worker.

Immediate availability: The message returns to the active queue and becomes instantly visible to other competing consumers.

Box 3: No
No, messages with invalid JSON payloads will not be retried until the maximum delivery count is reached. They will be sent to the dead-letter queue (DLQ) immediately on the very first attempt.
Immediate Dead-Lettering: The code catches the json.JSONDecodeError and explicitly calls receiver.dead_letter_message(msg).
Instant Termination: This API call tells Azure Service Bus to move the message to the DLQ right away, bypassing the normal delivery count logic.
No Re-delivery: Because the message is moved to the DLQ, the broker removes it from the main queue, stopping any further processing attempts.
Reference:
https://learn.microsoft.com/en-us/python/api/azure-servicebus/azure.servicebus.aio.servicebusreceiver
https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-python-how-to-use-queues

NEW QUESTION: 81
You deploy a production Azure Function app that connects to an Azure SQL Database.
The solution must provide the following functionality:
* Prevent secrets from being exposed in source control.
* Support secret rotation without redeploying the function app.
* Avoid downtime during credential updates.
You need to configure secure and maintainable secret management. What should you configure?
A. Hard coded connection string in the startup class
B. Application settings with Key Vault references
C. Parameter file stored in source control
D. Environment variables in local.settings.json
Answer: (SHOW ANSWER)

NEW QUESTION: 82
You process Azure Service Bus messages that require a dependent external API call.
If the API is temporarily unavailable, you must delay processing of the message without incrementing the delivery count. You need to find a way to process the message when the API is available while keeping the message accessible. Which message action should you perform?
A. Defer
B. Complete
C. Dead-letter
D. Abandon
Answer: (SHOW ANSWER)
Detailed Explanation: Deferral is intended for a message that cannot be processed now because a particular dependency is temporarily unavailable, while the application wants to keep the message in the main queue for later retrieval by sequence number. Complete would permanently remove it, dead-letter would move it aside as a failed message, and abandon immediately releases the lock for redelivery. Deferral therefore best matches the requirement to postpone the individual message while keeping it available for deliberate later processing.
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 deferral | Service Bus message transfers, locks, and settlement

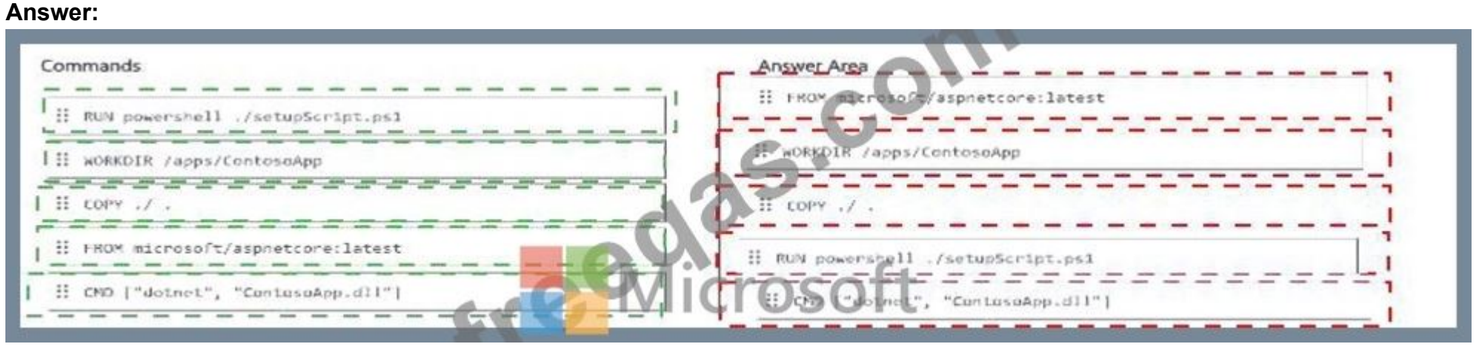
NEW QUESTION: 83
You plan to create a Docker image that runs an ASP.NET Core application named ContosoApp. You have a setup script named setupScript.psl and a series of application files including ContoscApp.dll.
You need to create a Dockerfile document that meets the following requirements:

* Call SQiupScript.psI when the container is built.
* Run CantosoApp.dill when the container starts.

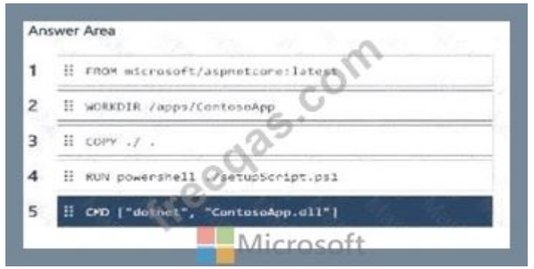
The Dockerfile document must be created in the same folder where ContosoApp.dll and setupScnpt.ps! are stored.



Answer:



Explanation:



NEW QUESTION: 84

Case Study 2 - Proseware Inc.

Background

Proseware Inc. develops AI-powered knowledge management solutions for enterprise customers.

The company is modernizing its platform to support semantic search, intelligent document retrieval, and real-time partner integrations.

The engineering team uses Python and Azure SDKs. The architecture is being redesigned to support containerized microservices, vector search workloads, and serverless backend processing.

Planned Application Architecture

Microservices are containerized by using Docker.

Code for containerized microservices and Azure Function apps is developed locally but stored in a GitHub repository.

Custom images for containerized microservices are stored in Azure Container Registry (ACR).

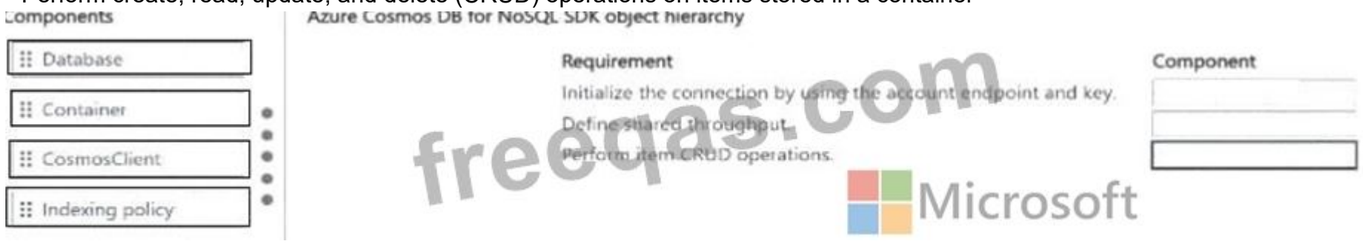
Base images are stored in Docker Hub. Custom images must be rebuilt automatically whenever their base images are updated.

Azure Cosmos DB for NoSQL stores documents, metadata, and vector embeddings.

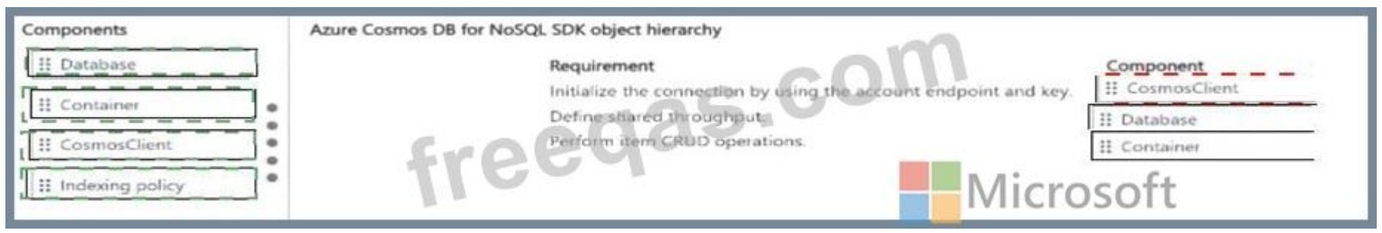
Azure Functions generate vector embeddings of Azure Cosmos DB for NoSQL-hosted documents and send messages to Service Bus to trigger search index updates.
Azure Container Apps (ACA) apps host backend API services that provide semantic search across Azure Cosmos DB for NoSQL documents. API services process Service Bus messages and update search indexes.
Azure Kubernetes Service (AKS) processes batch vector embedding regeneration for existing Azure Cosmos DB for NoSQL documents (whenever the embedding model is changed).
An extranet-facing containerized webhook allows business partners to submit documents to be processed by internal AI workflows for semantic search and retrieval.
Monitoring
Telemetry generated by Azure resources is sent to Azure Monitor.
A Log Analytics workspace is used to collect ACA apps logs, AKS container logs, and Azure Functions apps logs.
Monitoring of Azure Functions is currently implemented by using Azure Application Insights SDK instrumentation.
Business Requirements
Embeddings for new or updated Azure Cosmos DB for NoSQL-hosted documents must be automatically generated.
Backend API services must scale automatically during business hours.
Cold start delay of backend APIs must be minimized.
Secrets must be stored outside of container images.
Developers must be able to correlate telemetry across Azure Functions hosts and apps.
All tracing must be implemented by using OpenTelemetry SDK instrumentation.
Development efforts must be minimized.
Technical Requirements
Container images must be built automatically and validated before code updates are merged into the main branch.
Image build automation must run inside the Azure Container Registry, eliminating dependency on local developer machines and external build services.
Dependency of image builds on local developer machines must be eliminated.
Event-driven scaling in ACA must occur based on the number of pending messages in the Azure Service Bus queue.
Azure Cosmos DB for NoSQL RU consumption must be minimized.
Vector similarity search must use embeddings stored in Azure Cosmos DB for NoSQL.
The partner-facing containerized webhook service must run on Azure App Service.
Secrets must NOT be stored in container images, source control, or application configuration directly. They must be accessed securely at runtime.
All secrets must be stored centrally in Azure Key Vault and accessed at runtime through a managed identity.
Azure App Service must supply secrets at runtime without relying on external services.
Resources and workloads must be deployed by using Bicep templates through an automated, version-controlled pipeline. Local and command-line deployments must be eliminated to ensure repeatable, auditable deployments.
Known Issues
RU consumption spikes during vector similarity queries.
You need to address the known issue resulting from vector similarity queries.
Which two actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.
A. Add a composite index on the vector fields and metadata properties of the container.
B. Modify the indexing precision of the vector fields.
C. Set the account consistency level to Strong.
D. Change the vector index type from Oat to quantizedFlat or diskANN.
Answer: B,D (LEAVE A REPLY)
Known Issues: RU consumption spikes during vector similarity queries.
To address the RU (Request Unit) consumption spikes during vector similarity queries in Azure Cosmos DB, you should perform the following two steps:

Modify the indexing precision of the vector fields
Change the vector index type from flat to quantizedFlat or diskANN
Indexing Precision Tuning: Modifying parameters like vector quantization (e.g., configuring quantizedByteSize) alters the size and precision of the stored vector elements. Lowering the precision decreases memory usage and index size, allowing faster searches that consume far fewer RUs at the expense of marginal recall accuracy.
Index Type Modification: A standard flat index conducts a brute-force k-nearest neighbors (kNN) exact search across every document. This requires vast computational overhead and causes RU spikes as your dataset grows. Transitioning to quantizedFlat or diskANN leverages compression techniques and advanced graph-traversal algorithms to perform approximate nearest neighbor (ANN) searches, dropping query latency and compute costs significantly.
Reference:
https://learn.microsoft.com/en-us/azure/cosmos-db/gen-ai/vector-search-performance-tips

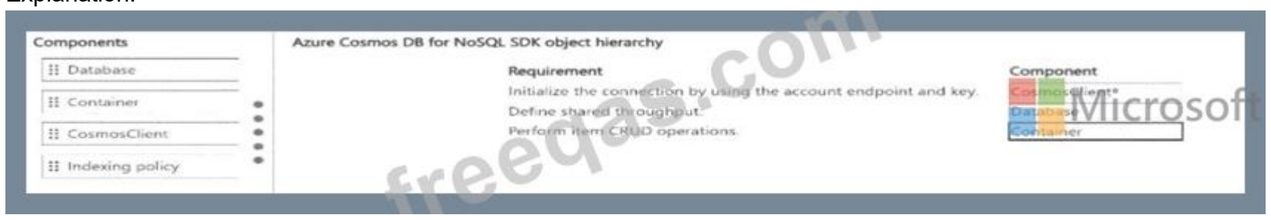
NEW QUESTION: 85
You are developing a .NET application that uses Azure Cosmos DB for NoSQL to store application data The application uses the Azure Cosmos DB for NoSQL SDK to interact with the database account. The application must perform the following tasks
* Initialize the connection by using the 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: 86
Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals.
You are using Azure Monitor Application Insights to investigate a production API. You open the Logs blade and set the time range to Last 24 hours.
An engineer recommends the following query to count requests by result code and sort the results from most frequent to least frequent:
requests
| summarize request_count = count() by resultCode
| order by request_count desc
You need to determine whether the query returns the number of requests grouped by result code and sorted from most to least frequent.
Solution: The result codes are sorted alphabetically.
Does the solution meet the goal?
A. Yes
B. No
Answer: (SHOW ANSWER)
The solution does not meet the goal because the query does not sort by resultCode. The summarize operator groups request records by each distinct resultCode value and calculates the number of requests in each group by using count().
The resulting table contains a resultCode column and a calculated request_count column. The next statement:
| order by request_count desc
sorts the output by request_count in descending order , which places the most frequently occurring result code first and the least frequent last.
Therefore, the output is sorted by frequency , not alphabetically or numerically by the resultCode field.
If alphabetical sorting by result code were required, the query would instead use a statement such as:
| order by resultCode asc
That is not what the provided query does.
Because the Logs blade is already scoped to the Last 24 hours , the query operates over that selected time range unless additional time filtering is explicitly added.
Study Guide references: Azure Monitor Logs; Application Insights requests table; KQL summarize; count() aggregation; order by; descending sort semantics.

NEW QUESTION: 87
Case Study 1 - 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.
Hotspot Question
You need to optimize secure database connectivity from the containerized Recommendation API.
How should you configure the application? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

Requirement	Configuration
Comply with the authentication policy for database access.	Use connection pooling library. Use managed identity authentication. Store credentials in environment variables.
Support high-concurrency requests with minimal latency.	Use unlimited pool size. Use a connection pooling library. Increase the database max_connections parameter.
Protect database stability during traffic spikes.	Configure maximum pool size. Enable asynchronous query execution. Implement a retry policy with exponential backoff.

Microsoft
freeqas.com

Answer:

Answer Area

Requirement	Configuration
Comply with the authentication policy for database access.	Use connection pooling library. **Use managed identity authentication.** Store credentials in environment variables.
Support high-concurrency requests with minimal latency.	Use unlimited pool size. **Use a connection pooling library.** Increase the database max_connections parameter.
Protect database stability during traffic spikes.	**Configure maximum pool size.** Enable asynchronous query execution. Implement a retry policy with exponential backoff.

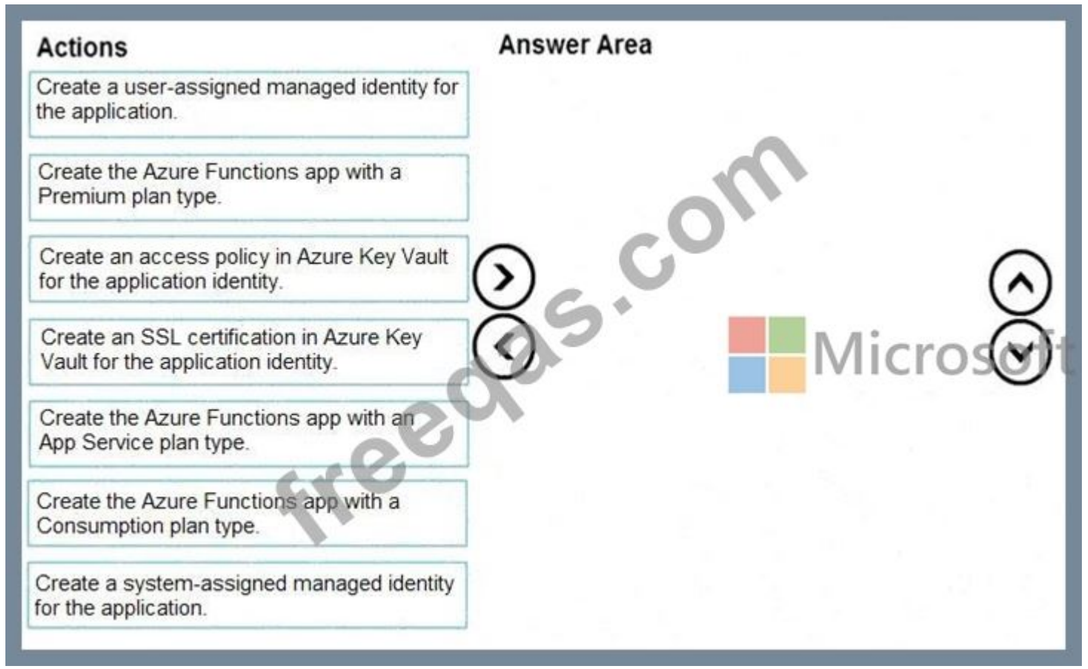
freeqas.com Microsoft

Explanation:

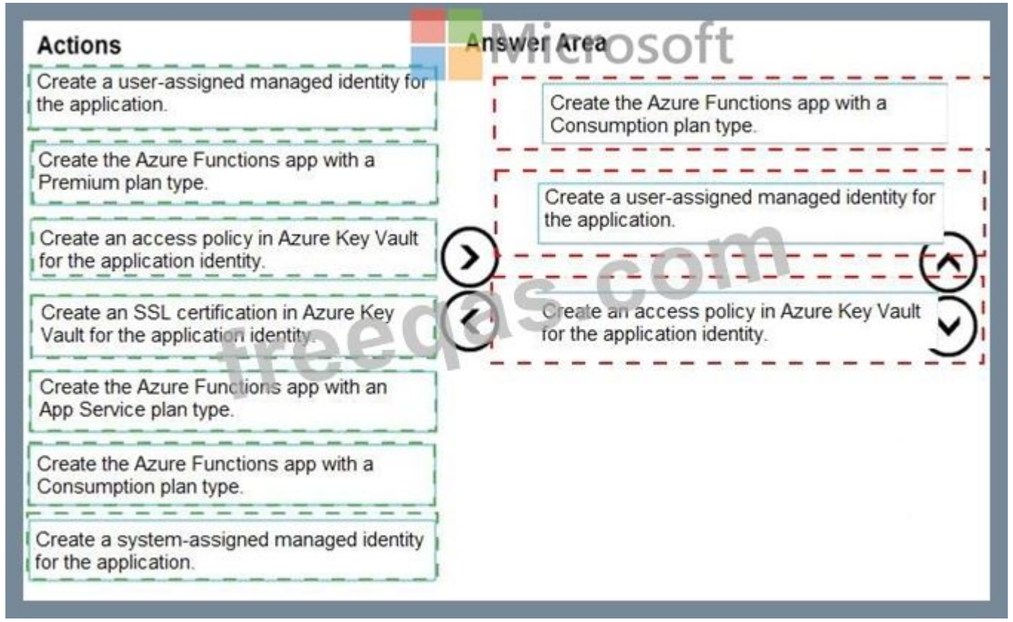
Box 1: Use managed identity authentication.

Scenario: Identity: Use managed identities for all service-to-service and service-to-database authentication. Plain-text credentials in configuration files are strictly prohibited.' Box 2: Use a connection pooling library Scenario: Environment: Embeddings are stored in Azure Database for PostgreSQL by using pgvector.
To support high-concurrency requests with minimal latency on Azure Database for PostgreSQL, the best action is to use a connection pooling library (or leverage Azure's built-in PgBouncer feature).
Eliminates Connection Overhead:
PostgreSQL utilizes a process-per-connection model. Forking a new backend process for every incoming request introduces substantial CPU and memory overhead, severely degrading latency under high concurrency. A connection pool keeps a warm set of reusable database sessions active.
Optimized for Azure: Microsoft provides a built-in managed PgBouncer integration for Azure Database for PostgreSQL. Enabling it in transaction mode allows the database to accept thousands of concurrent client connections while keeping actual backend processes lean and stable Box 3: Configure a maximum pool size Configure a maximum pool size is the best action to directly protect database stability during sudden traffic spikes.
Prevents Resource Exhaustion: Traffic spikes naturally lead to a surge in connection requests.
Unchecked connections consume substantial RAM and CPU overhead, which can crash the database or trigger severe latency. Limiting the pool size stops the "thundering herd" problem Acts as a Shock Absorber: When the pool hits its maximum limit, extra client requests are safely queued at the application or connection pooling layer (like Azure's built-in PgBouncer proxy) rather than overwhelming the database backend Reference:
https://docs.azure.cn/en-us/postgresql/connectivity/concepts-pgbouncer
https://learn.microsoft.com/en-au/answers/questions/5884412/best-practise-azure-postgresql-flexible-server-max

NEW QUESTION: 88
You are developing a serverless Java application on Azure. You create a new Azure Key Vault to work with secrets from a new Azure Functions application.
The application must meet the following requirements:
Reference the Azure Key Vault without requiring any changes to the Java code.
Dynamically add and remove instances of the Azure Functions host based on the number of incoming application events.
Ensure that instances are perpetually warm to avoid any cold starts.
Connect to a VNet.
Authentication to the Azure Key Vault instance must be removed if the Azure Function application is deleted.
You need to grant the Azure Functions application access to the Azure Key Vault.
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.



Answer:



Explanation:



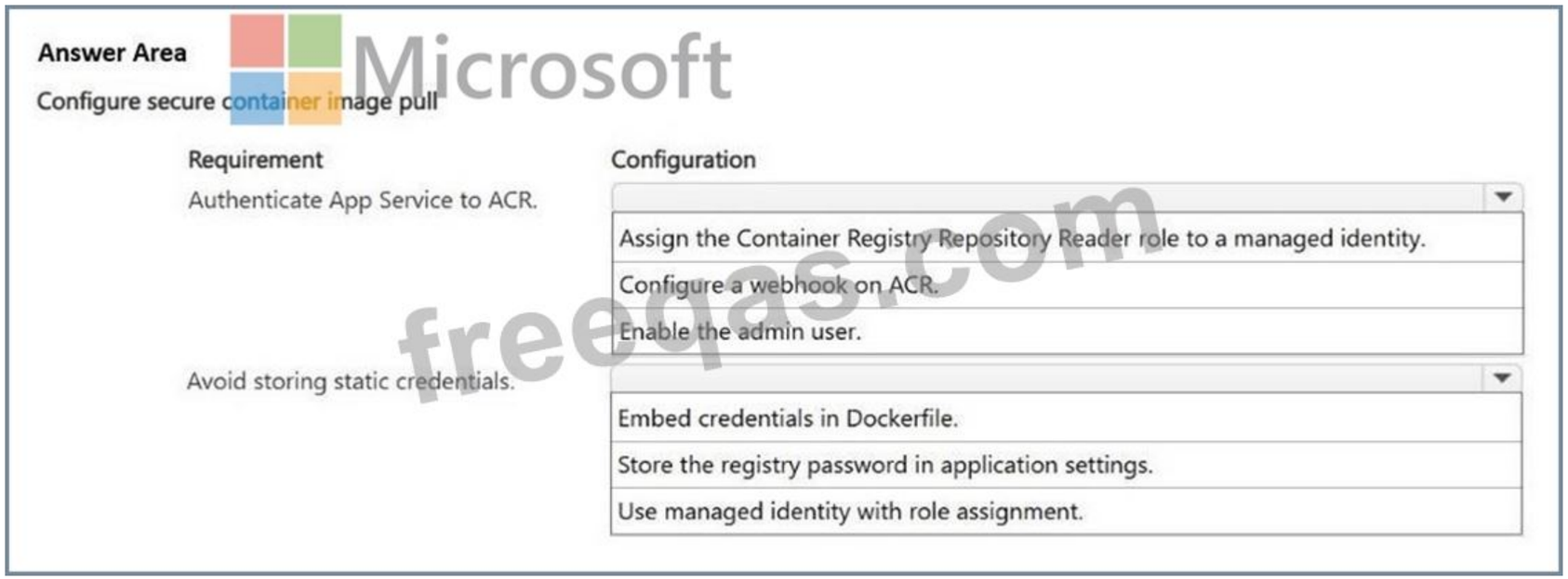
Step 1: Create the Azure Functions app with a Consumption plan type.
Use the Consumption plan for serverless.
Step 2: Create a system-assigned managed identity for the application.
Create a system-assigned managed identity for your application.
Key Vault references currently only support system-assigned managed identities. User-assigned identities cannot be used.
Step 3: Create an access policy in Key Vault for the application identity.
Create an access policy in Key Vault for the application identity you created earlier. Enable the " Get " secret permission on this policy. Do not configure the " authorized application " or applicationId settings, as this is not compatible with a managed identity.
Reference:
https://docs.microsoft.com/en-us/azure/app-service/app-service-key-vault-references

NEW QUESTION: 89
An application deployed to AKS depends on an internal API hosted in the same cluster. All pods are healthy and report Ready status, but requests between the services time out.
You need to determine whether the issue is related to Kubernetes service configuration or application code.
What should you do first?
A. Restart the failing deployment.
B. Inspect Service and Endpoints objects.
C. Increase replica count for both services.
D. Test DNS resolution from your workstation.
Answer: B (LEAVE A REPLY)
Inspect Service and Endpoints objects is the best step to start with.
The symptoms describe pods that are healthy and passing their readiness probes (meaning they are Ready), but traffic is failing to reach them and timing out. In Kubernetes, a Service routes traffic to pods by matching labels to create an Endpoints object.
Checking these objects immediately verifies if the Kubernetes network routing layer is intact.If the Endpoints object is empty or missing, the Service selector does not match the Pod labels. Traffic has nowhere to go, causing a timeout.
If the Endpoints list correctly displays the Pod IP addresses, the Kubernetes routing layer is working, shifting the blame toward the application code (e.g., misconfigured listening ports) or network policies.
Reference:
https://www.xtivia.com/blog/kubernetes-container-health-checks/

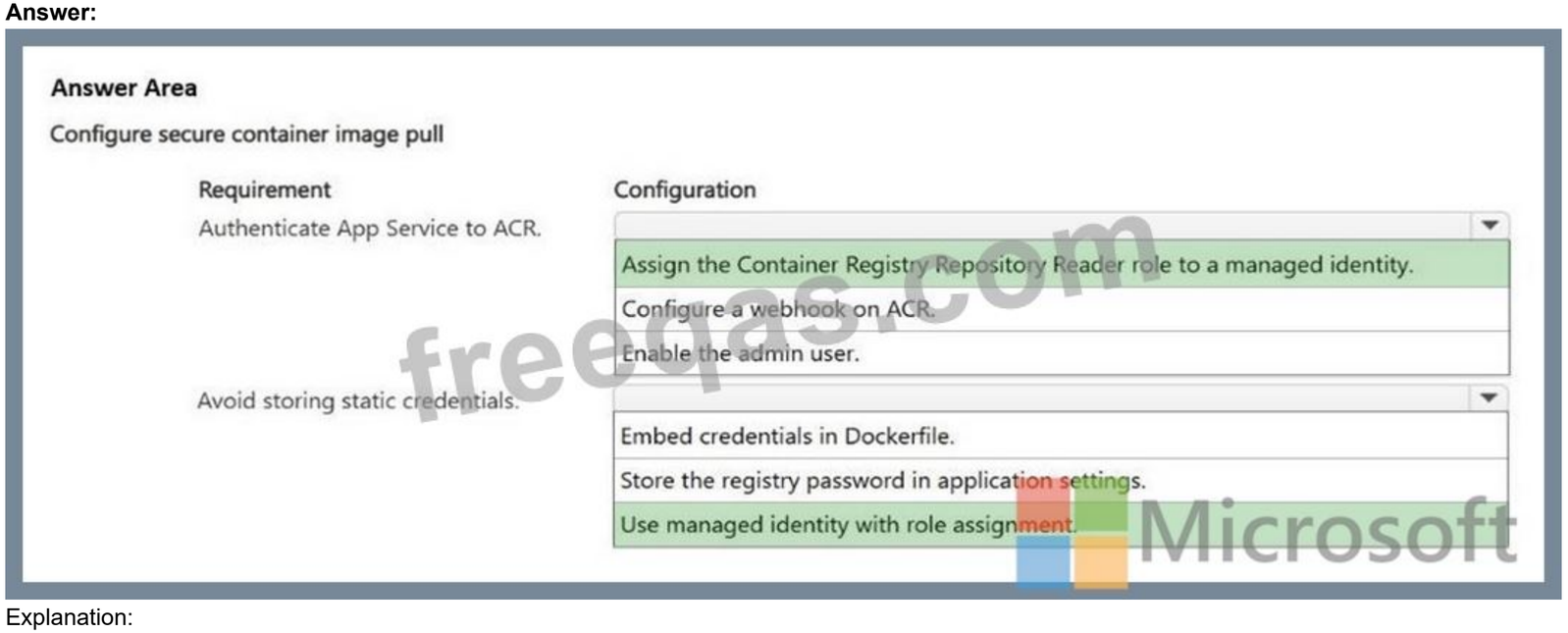
NEW QUESTION: 90
You deploy a production Azure Function app that connects to an Azure SQL Database.
The solution must provide the following functionality:
- Prevent secrets from being exposed in source control.
- Support secret rotation without redeploying the function app.

- Avoid downtime during credential updates.
You need to configure secure and maintainable secret management.
What should you configure?
A. Application settings with Key Vault references
B. Environment variables in local.settings.json
C. Hard-coded connection string in the startup class
D. Parameter file stored in source control
Answer: A (LEAVE A REPLY)
To meet all requirements, you should configure Application settings with Key Vault references.
Source Control Protection: The Function App source code and configuration files only store a reference URI (e.g., @Microsoft.KeyVault(SecretUri=...)) rather than the actual connection string, keeping secrets entirely out of source control.
Seamless Secret Rotation: Azure Key Vault handles secret rotation natively. When a database password changes, you simply update the secret in Key Vault.
Zero Downtime: By using versionless Key Vault references (omitting the specific version GUID from the URI), the Azure Function App will automatically fetch the latest secret version within 24 hours without requiring a code redeployment or app restart.
References:
https://oneuptime.com/blog/post/2026-02-16-how-to-configure-managed-identity-for-azure-app-service-to-access-key-vault-secrets-without-credentials/view

NEW QUESTION: 91
Hotspot Question
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.



Answer:



Explanation:

Box 1: Assign the Container Registry Reader role to a managed identity.

To securely deploy a private container image from Azure Container Registry (ACR) to an Azure App Service without managing passwords, you should configure a Managed Identity on the App Service and grant it the AcrPull role on the ACR.

Box 2: Use managed identity with role assignment.

you should use a Managed Identity combined with an Azure Role-Based Access Control (RBAC) role assignment to meet the requirement.

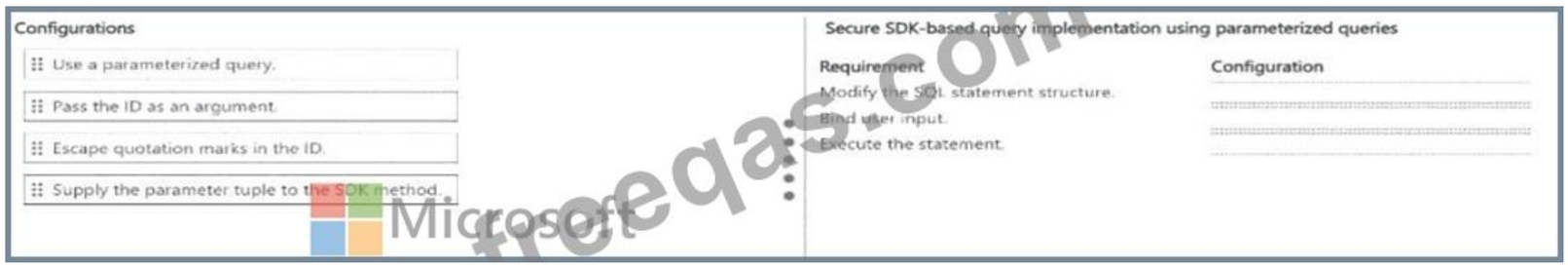
Reference:
https://medium.com/@miketobicarter/deploying-a-net-app-to-azure-using-docker-acr-app-service-and-github-actions-step-by-step-3ed6ae8a4d0b

Valid AI-200 Dumps shared by PrepPdf.com for Helping Passing AI-200 Exam! PrepPdf.com now offer the **newest AI-200 exam dumps**, the PrepPdf.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** PrepPdf.com AI-200 dumps with Test Engine here: https://www.preppdf.com/Microsoft/AI-200-prepaway-exam-dumps.html (**144** Q&As Dumps, **40%OFF** **Special Discount: Exam-Tests**)

NEW QUESTION: 92
You have an Azure Service Bus namespace with a partitioned queue named queue1.
You plan to send a large number of messages through queue1 over the next few weeks. The order of messages will be random. You must minimize the possibility of message transmission interruption by transient failures of individual partitions.
You need to use the optimal configuration of the partition key in the messages.
Which configuration should you use?
A. Leave the partition key value as null.
B. Set the partition key of messages to the message ID value.
C. Enable sessions. Set the partition key of messages to the session ID value.
D. Enable sessions. Ensure that the partition key is different from the session ID value.
Answer: A (LEAVE A REPLY)

NEW QUESTION: 93
A development team wants to compare multiple prompt variations against the same test dataset and visualize which prompt performs best. What should they use in Azure AI Foundry?
A. Model deployment quota management
B. Prompt flow with variants
C. Azure AI Search indexers
D. Speech Studio custom voice
Answer: B (LEAVE A REPLY)
Prompt flow variants allow you to define multiple versions of a prompt/node, run them against the same evaluation dataset, and compare results side-by-side to identify the best-performing variant.

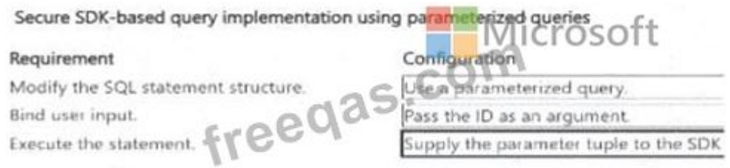
NEW QUESTION: 94
A Python API retrieves a document from Azure Database for PostgreSQL by using a SQL statement. The API accepts the document ID from user input. The current implementation inserts the document ID directly into the SQL statement.
You need to secure the SQL statement execution by using 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.



Answer:



Explanation:



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: 95
You are configuring an Azure-hosted AI application that uses Azure App Configuration to manage its runtime settings.
The application must provide the following functionality:
- Support percentage-based, targeted rollouts.
- Maximize token limits per request.
- Store a key that is retrieved securely at runtime.
- Reference a list of supported deployment regions.
You need to configure the App Configuration features for the app.
Which features should you configure? To answer, move the appropriate App Configuration features to the correct configuration items. You may use each App Configuration features 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.

App Configuration features

Key-value

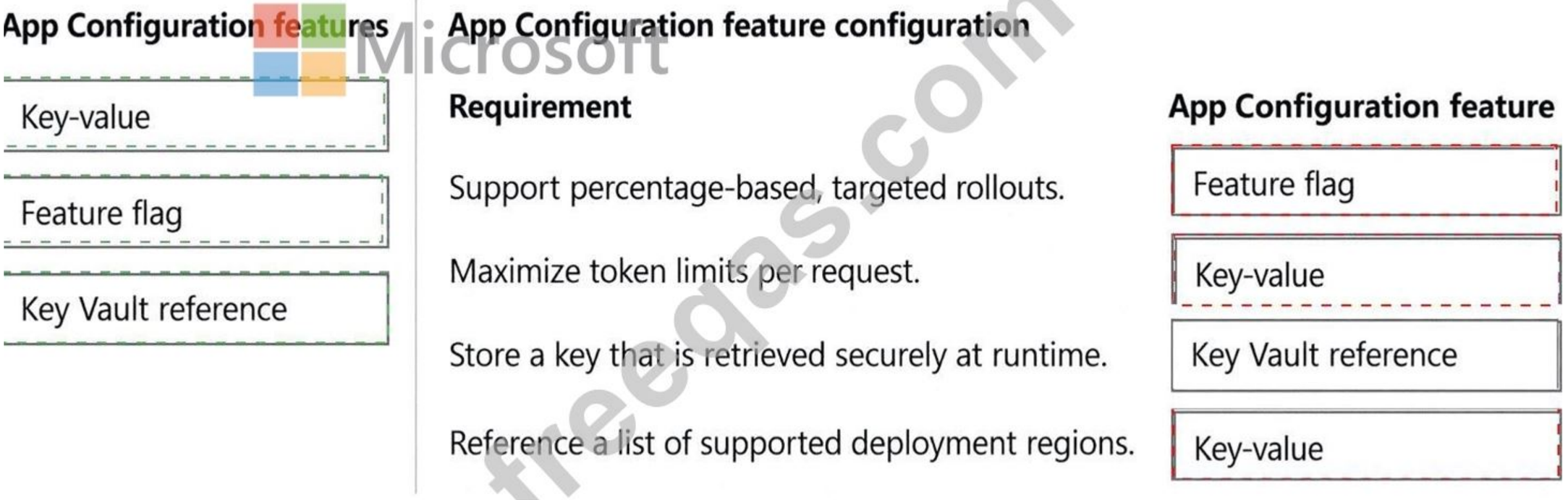
Feature flag

Key Vault reference

App Configuration feature configuration

Requirement	App Configuration feature
Support percentage-based, targeted rollouts.	
Maximize token limits per request.	
Store a key that is retrieved securely at runtime.	
Reference a list of supported deployment regions.	

Answer:



Explanation:

* Support percentage-based, targeted rollouts: Feature flag
* Maximize token limits per request: Key-value
* Store a key that is retrieved securely at runtime: Key Vault reference
* Reference a list of supported deployment regions: Key-value

Azure App Configuration provides three distinct mechanisms for these scenarios. A Feature flag is the correct choice for percentage-based and targeted rollouts. Feature Management supports conditional activation based on filters such as percentage allocation, users, groups, and other targeting criteria, making it appropriate for gradual deployment and controlled experimentation.

A Key-value is the correct mechanism for ordinary runtime configuration data such as a model token limit or a list of supported deployment regions. These values are application settings rather than secrets or feature- state controls. The application can retrieve and refresh them dynamically without redeployment. A token-limit setting such as MaxTokensPerRequest and a region list such as SupportedRegions fit naturally into App Configuration key-value entries.

For sensitive data, use a Key Vault reference . App Configuration stores only the reference to the Azure Key Vault secret rather than the secret value itself. At runtime, the application resolves the secret securely, typically by using a managed identity. This avoids embedding credentials or sensitive keys in configuration values.

Therefore, the correct mapping is Feature flag # Key-value # Key Vault reference # Key-value .

Study Guide references: Azure App Configuration # Feature Management; key-values; Key Vault references; dynamic application configuration.

NEW QUESTION: 96

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

Microsoft
freeqas.com

Answer: B (LEAVE A REPLY)
Each Azure OpenAI deployment can have Tokens-Per-Minute (TPM) and Requests-Per-Minute (RPM) limits configured, which cap consumption at the deployment level and directly control cost exposure.

NEW QUESTION: 97
You are implementing a Retrieval-Augmented Generation (RAG) system by using the native vector search capabilities of Azure Cosmos DB for NoSQL API.
You have a container named Documents that stores technical articles. Each article includes a property named embedding.
You must ensure that the system can perform efficient similarity searches between user queries and the stored articles.
You need to configure the database resources to support semantic retrieval.
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.

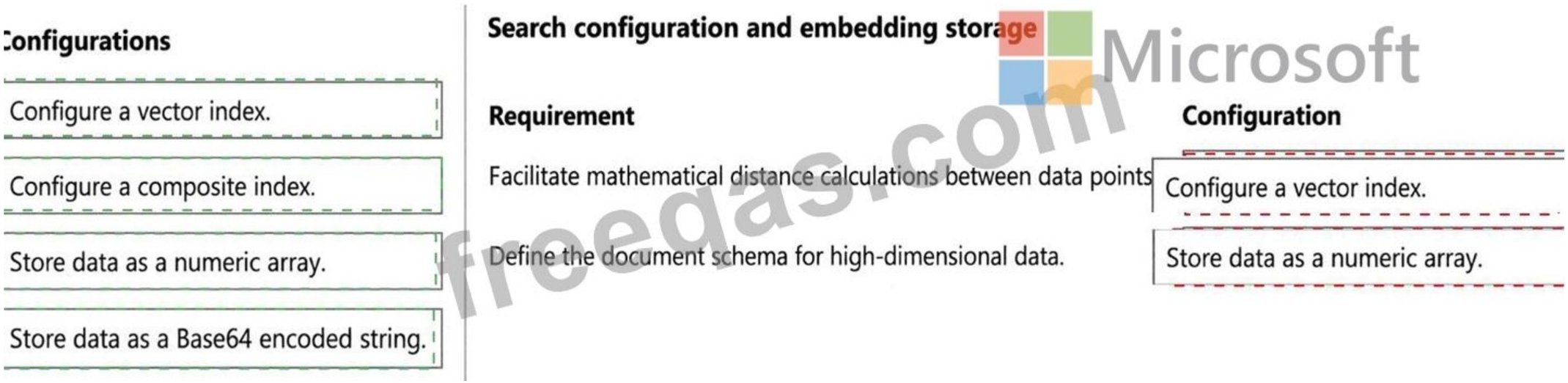
Configurations

- Configure a vector index.
- Configure a composite index.
- Store data as a numeric array.
- Store data as a Base64 encoded string.

Search configuration and embedding storage

Requirement	Configuration
Facilitate mathematical distance calculations between data points.	
Define the document schema for high-dimensional data.	

Answer:



Microsoft
freedas.com

Explanation:

* Facilitate mathematical distance calculations between data points: Configure a vector index.
* Define the document schema for high-dimensional data: Store data as a numeric array.

Azure Cosmos DB for NoSQL supports native vector search by storing embeddings directly in JSON documents and defining a vector embedding policy plus a vector index on the embedding path. Microsoft documents that vector search compares a query vector with stored vectors by calculating similarity or distance through the VectorDistance() system function. A vector index materially improves this process by reducing search latency, increasing throughput, and lowering RU consumption compared with an unindexed vector scan.

The embedding property itself must be stored as an array of numeric values . Microsoft examples show embedding properties such as " contentVector " : [2, -1, 4, ...] and define the associated vector policy with attributes including path, data type, dimensions, and distance function. This representation is required because embeddings are high-dimensional numerical vectors generated by an embedding model.

A composite index optimizes queries involving multiple scalar properties but does not provide vector- distance indexing. A Base64-encoded string cannot be used directly for native vector similarity calculations because Cosmos DB expects the vector field to contain numeric values matching the configured dimensionality.

Study Guide references: Azure Cosmos DB for NoSQL # vector embedding policies; vector indexes; VectorDistance(); numeric embedding arrays; native vector search.

NEW QUESTION: 98

Drag and Drop Question

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

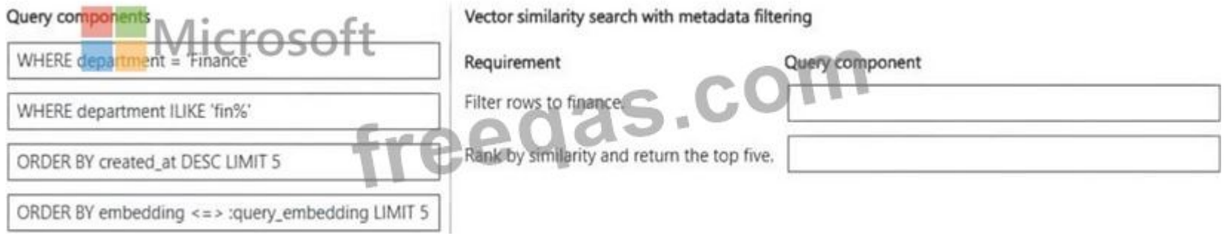
- embedding (pgvector)
- department
- created_at

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.

NOTE: Each correct selection is worth one point.



Answer:



Explanation:

Box 1: WHERE department = 'Finance'

Box 2: ORDER BY embedding <=> :query_vector LIMIT 5

To rank and return the top five most similar documents by similarity in Microsoft Learn, use the ORDER BY embedding <=> [query_vector] expression combined with LIMIT 5, where <=> computes the cosine distance.

Reference:

https://learn.microsoft.com/en-us/azure/postgresql/extensions/how-to-use-pgvector

NEW QUESTION: 99

You need to detect whether an uploaded image contains adult or violent content before allowing it to be posted to a public forum. What should you use?

A. Azure AI Custom Vision object detection

B. Azure AI Content Safety image moderation

C. Azure AI Face API

D. Azure AI Document Intelligence

Answer: (SHOW ANSWER)

Azure AI Content Safety's image moderation capability is purpose-built to detect categories such as sexual, violent, self-harm, and hate content in images, returning severity scores you can act on.

NEW QUESTION: 100

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.

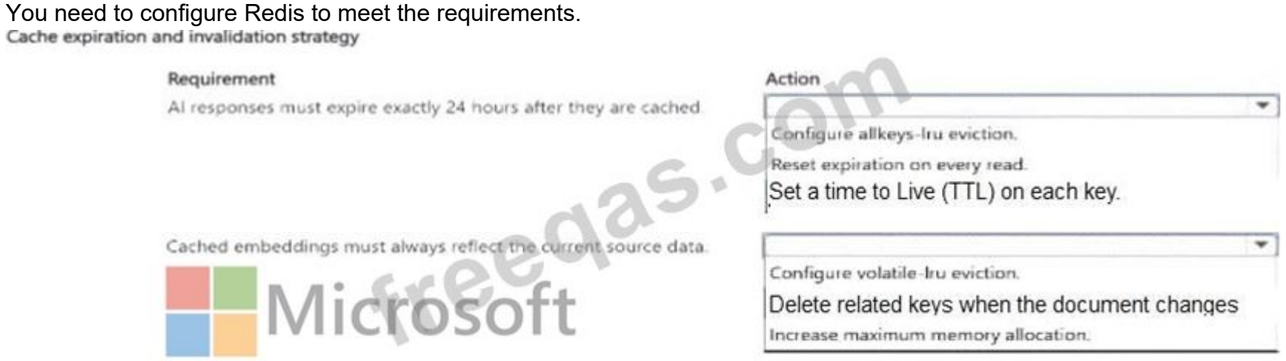
C. Retrieve configuration values from App Configuration on each request.

D. Store configuration values in Key Vault.

Answer: B (LEAVE A REPLY)

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: 101
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.



Answer:



Explanation:

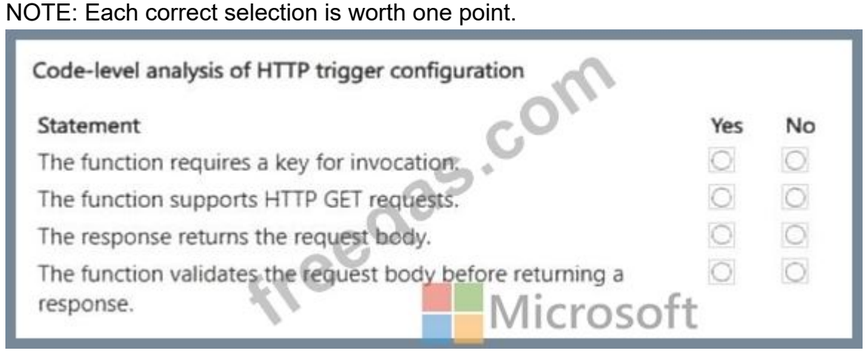


l response expiry: set a 24-hour TTL on each key. Embedding freshness: delete related cache keys when the source document changes.
Detailed Explanation: A fixed TTL encodes the requirement that an AI response must expire after a defined period rather than survive indefinitely or have its lifetime extended 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.
Official Microsoft Learn References: AI-200 Study Guide | Azure Managed Redis documentation

NEW QUESTION: 102
Hotspot Question
You are reviewing an Azure Function app that processes incoming order requests for a company.
The function must
- Accept order submissions from an external client application.
- Require controlled access for security.
- Return a response containing the processed request payload.
You review the following code segment:

```
01  [FunctionName("ProcessOrder")]
02  public static async Task<IActionResult> Run(
03    [HttpTrigger(AuthorizationLevel.Function, "post")] HttpRequest req,
04    ILogger log)
05  {
06    string result = await new StreamReader(req.Body).ReadToEndAsync();
07    return new OkObjectResult(result);
08  }
```

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:

Code-level analysis of HTTP trigger configuration

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

freeqas.com

Microsoft

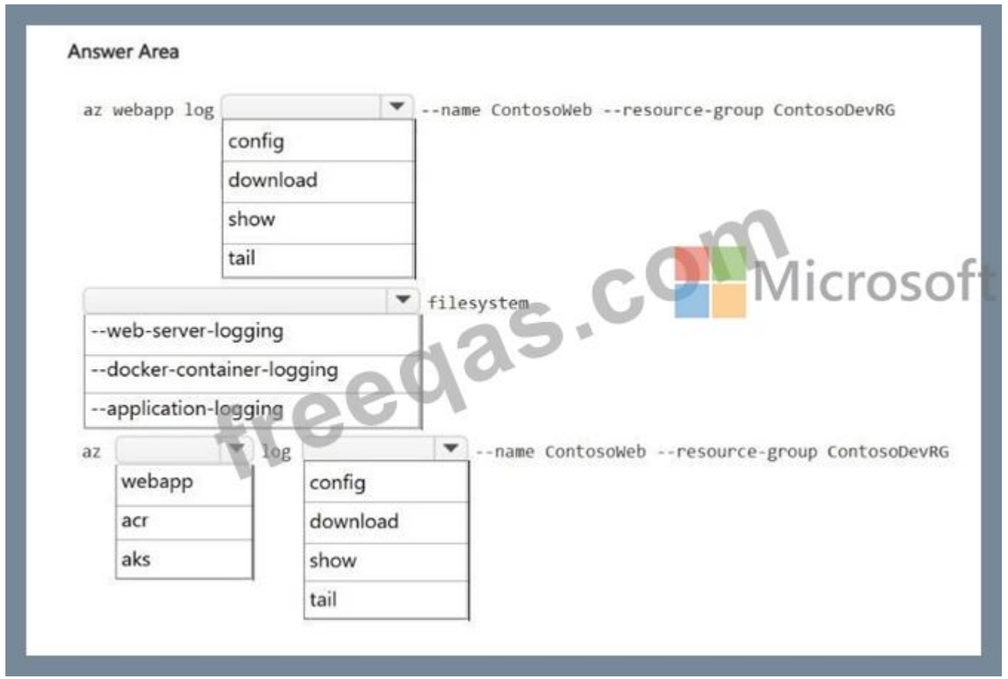
Explanation:
Box 1: Yes
Yes, this function requires an API key for invocation.
Authorization Level: The AuthorizationLevel.Function parameter in the trigger dictates the security requirement.
Function Key: You must provide a specific function-level API key to successfully execute the request.
HTTP Status: Accessing this endpoint without a valid key will return a 401 Unauthorized HTTP status code.
How to Pass the Key
You can authenticate your request by passing the key in one of two ways:
Query String: Append ?code=YOUR_FUNCTION_KEY to the end of your request URL.
HTTP Header: Include an x-functions-key header with your key as the value.
Box 2: No
No, this Azure Function does not support HTTP GET requests.
Explicit Method Restriction: The HttpTrigger attribute explicitly defines "post" as the only accepted HTTP method.
Default Behavior Overridden: When you specify methods in the trigger array, all other unlisted methods are blocked.
Box 3: Yes
Yes, this specific Azure Function code segment returns the exact incoming request body back to the caller.
StreamReader(req.Body).ReadToEndAsync(): This line reads the entire raw payload sent in the HTTP POST request and stores it as a string in the result variable.
OkObjectResult(result): This creates an HTTP 200 OK response and places that exact result string directly into the response body.
Box 4: No
No, the provided Azure Function code does not validate the request body before returning a response.
Direct Echo: The function reads the raw request stream into a string variable named result.
Immediate Return: It instantly passes that unverified string directly into an OkObjectResult.
Missing Logic: There are no schema checks, null checks, type validations, or business logic evaluations performed on the data.
Reference:
https://learn.microsoft.com/en-us/azure/durable-task/durable-functions/durable-functions-http-api

NEW QUESTION: 103
You are building a multi-agent solution in Azure AI Foundry where one agent handles scheduling and another handles billing questions, and a request may need both. What should you implement?
A. A single monolithic prompt covering both domains
B. An orchestrator agent that routes sub-tasks to specialized agents and combines results
C. Two separate applications with no coordination
D. A single fine-tuned model trained on both domains only

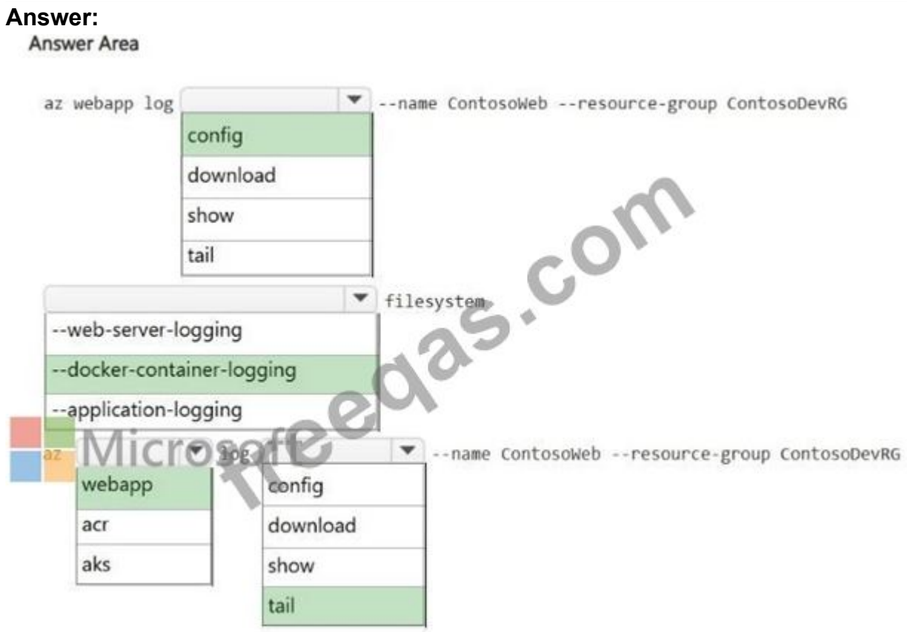
Answer: B (LEAVE A REPLY)
Multi-agent orchestration patterns use a coordinating (orchestrator) agent to decompose a request, delegate sub-tasks to specialized agents, and aggregate their outputs -- this is more maintainable and accurate than one broad prompt or model trying to cover disparate domains.

NEW QUESTION: 104
You are building a RAG (retrieval-augmented generation) solution using Azure AI Foundry. The knowledge base consists of 50,000 PDF documents stored in Azure Blob Storage.
You need to make the document content searchable by the language model with minimal custom code.
What should you use?
A. Azure AI Search integrated vectorization with a skillset
B. Azure Cognitive Service for Language custom text classification
C. Azure Functions to manually chunk and embed each document
D. Azure Table Storage with full-text indexing
Answer: A (LEAVE A REPLY)
Use Azure AI Search integrated vectorization with a skillset . Integrated vectorization automates the main ingestion stages required for RAG: an Azure AI Search indexer can read documents directly from a supported data source such as Azure Blob Storage, extract and split document content into chunks, invoke an embedding model, and store both searchable text and vector representations in the search index. Microsoft specifically states that integrated vectorization reduces development and maintenance effort because developers do not need to implement the chunking and embedding pipeline manually.
A skillset can include a Text Split skill for chunking and an Azure OpenAI Embedding skill or another supported embedding skill for vector generation. At query time, a matching vectorizer can convert user questions into vectors so Azure AI Search can perform similarity or hybrid retrieval for grounding the language model.
Custom text classification solves a different NLP problem. Building Azure Functions to process all 50,000 PDFs manually would introduce unnecessary custom orchestration and maintenance. Azure Table Storage does not provide the native vector-search and RAG indexing capabilities required here.
Study Guide references: Azure AI Search # integrated vectorization; indexers; skillsets; Text Split skill; embedding skills; vector indexes; RAG retrieval.

NEW QUESTION: 105
Hotspot Question
You plan to deploy a web app to App Service on Linux. You create an App Service plan. You create and push a custom Docker image that contains the web app to Azure Container Registry.
You need to access the console logs generated from inside the container in real-time.
How should you complete the Azure CLI command? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

Box 1: config

Modifies the logging configuration settings for the specified Web App.

Box 2: --docker-container-logging

--docker-container-logging: The explicit parameter that defines where or how Docker container console logs are saved (e.g., filesystem).
Configure Logging:
az webapp log config --name ContosoWeb --resource-group ContosoDevRg --web-server-logging filesystem Box 3: webapp webapp specifies the group identifier within the az CLI execution context.
Box 4: tail
tail: Initiates a live, real-time diagnostic log stream directly from the active Linux container.
Reference:
https://learn.microsoft.com/en-gb/answers/questions/2225551/mis-show-azure-container-app-logs

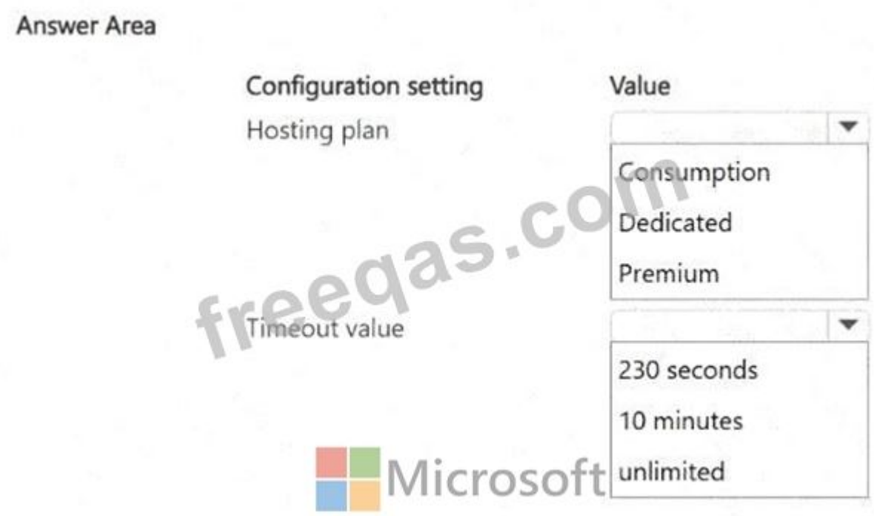
NEW QUESTION: 106
Your Azure AI Search index must support both exact keyword matches on product SKUs and semantic similarity search on product descriptions in a single query. What should you configure?
A. A keyword-only index
B. A vector-only index
C. A hybrid search index combining vector fields and searchable text fields
D. Two completely separate indexes queried independently
Answer: C (LEAVE A REPLY)
Hybrid search in Azure AI Search combines traditional keyword (BM25) search over text fields with vector similarity search over embedding fields in a single query, giving both exact-match precision (SKUs) and semantic recall (descriptions).

Valid AI-200 Dumps shared by PrepPdf.com for Helping Passing AI-200 Exam! PrepPdf.com now offer the **newest AI-200 exam dumps**, the PrepPdf.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** PrepPdf.com AI-200 dumps with Test Engine here: https://www.preppdf.com/Microsoft/AI-200-prepaway-exam-dumps.html (**144** Q&As Dumps, **40%OFF** **Special Discount: Exam-Tests**)

NEW QUESTION: 107
You configure ACR Tasks to automate image builds.
Container images must rebuild when:
- Application updates occur.
- Base image updates occur, such as when the underlying OS image is
updated.
- Regular scheduled rebuilds are required.
You need to configure ACR Tasks to support automated image rebuilds.
Which three triggers should you configure? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.
A. Webhook notification trigger
B. Timer trigger
C. Registry event trigger
D. Base image update trigger
E. Source code commit trigger
Answer: B,D,E (LEAVE A REPLY)

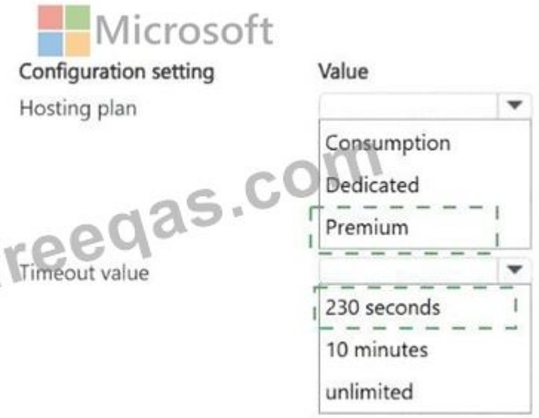
NEW QUESTION: 108
You need to add a custom voice that matches your company's brand for a text-to-speech feature, distinct from any standard neural voice. What should you use?
A. Azure AI Speech prebuilt neural voices
B. Azure AI Speech Custom Neural Voice
C. Azure AI Language sentiment analysis
D. Azure AI Translator
Answer: (SHOW ANSWER)
Custom Neural Voice lets you train a unique synthetic voice from recorded voice samples, producing a brand-specific voice rather than using one of the standard prebuilt neural voices.

NEW QUESTION: 109
You plan to develop an Azure Functions app with an HTTP trigger.
The app must support the following functionality:
Event-driven scaling -
Ability to use custom Linux images for function execution
You need to identify the app ' s hosting plan and the maximum amount of time that the app function can take to respond to incoming requests.
Which configuration setting values should you use? To answer, select the appropriate values in the answer area.
NOTE: Each correct selection is worth one point.



freeqas.com
Microsoft
Answer:

Answer Area



freeqas.com

Explanation:

* Hosting plan: Premium

* Timeout value: 230 seconds

The correct hosting option is the Azure Functions Premium plan . Microsoft documents that the Premium plan supports event-driven scaling , where Azure Functions dynamically adds or removes host instances based on incoming trigger activity. It also supports Linux container deployments , which is required when the function app must execute from a custom Linux container image. A Dedicated App Service plan can host Linux containers, but it does not provide native Functions event-driven scaling. The legacy Consumption plan supports event-driven scaling but does not satisfy the custom Linux container requirement in this question.

For an HTTP-triggered function , the maximum time available to return an HTTP response is 230 seconds .

This limit applies even though a Premium-plan function can have a much longer or effectively unbounded execution timeout. Microsoft explains that the 230-second response limit results from the Azure Load Balancer ' s default idle timeout. If processing continues beyond this point, the function can continue executing, but the client will receive a timeout and the function cannot return its eventual response over that original HTTP connection.

Therefore, Premium + 230 seconds is the required combination.

Study Guide references: Azure Functions hosting options; Premium plan; Linux container support; event- driven scaling; HTTP-trigger timeout behavior.

NEW QUESTION: 110

Hotspot Question

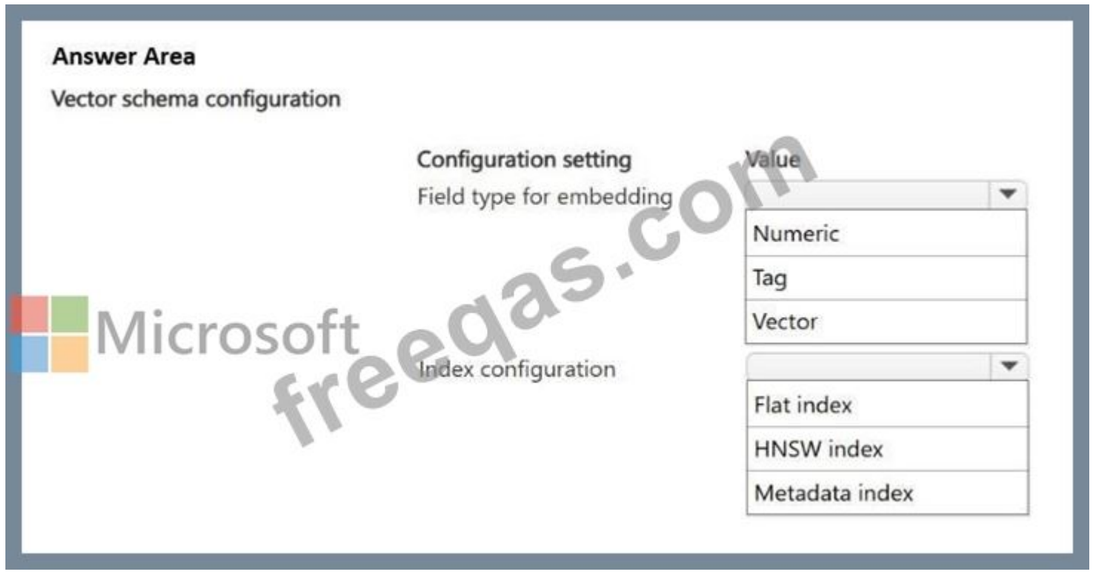
You are implementing semantic retrieval in Redis.

The solution must support low-latency, approximate nearest neighbor (ANN) vector similarity search for large-scale AI retrieval workloads.

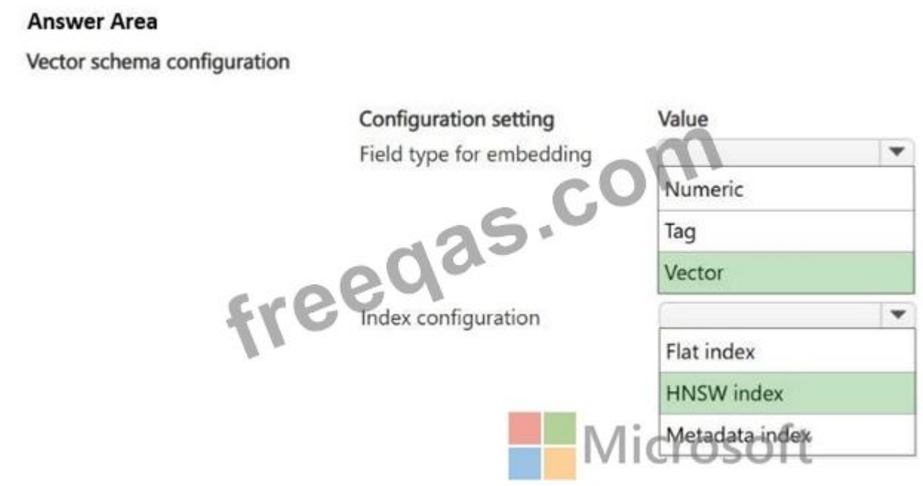
You need to select the appropriate vector schema settings.

Which configuration values should you select? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.



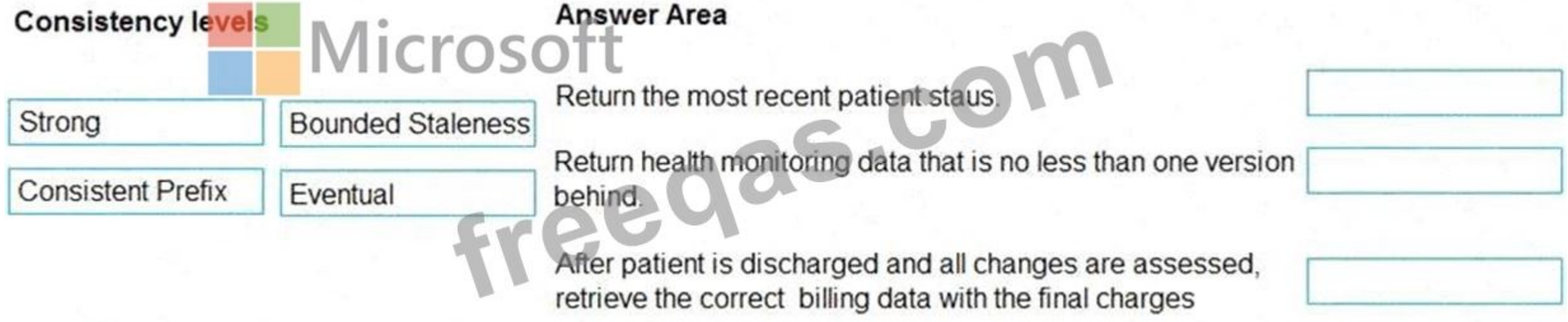
Answer:



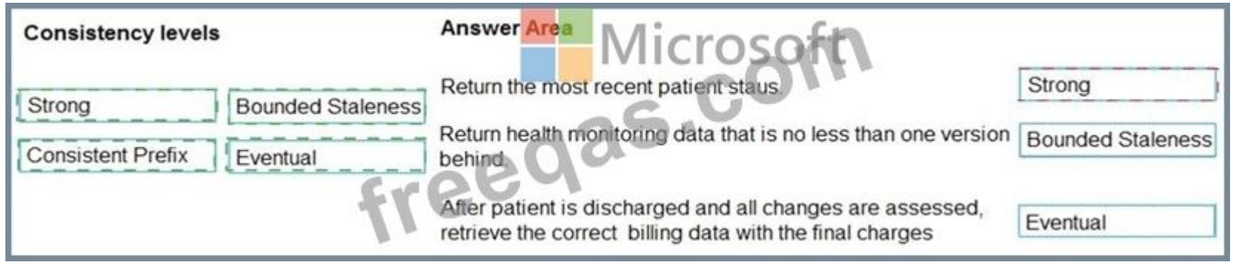
Explanation:
Box 1: Vector
The correct field type for storing and querying embeddings in Redis is VECTOR.
Box 2: HNSW index
To support low-latency, large-scale approximate nearest neighbor (ANN) vector search in Redis, you should use the HNSW (Hierarchical Navigable Small World) index type with Float32 data type.
Reference:
https://redis.io/docs/latest/develop/ai/search-and-query/vectors/

NEW QUESTION: 111
You are developing a solution for a hospital to support the following use cases:

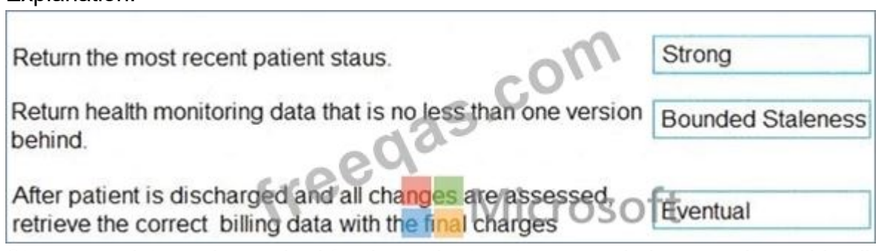
*The most recent patient status details must be retrieved even if multiple users in different locations have updated the patient record.
*Patient health monitoring data retrieved must be the current version or the prior version.
*After a patient is discharged and all charges have been assessed, the patient 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.



Answer:



Explanation:



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.

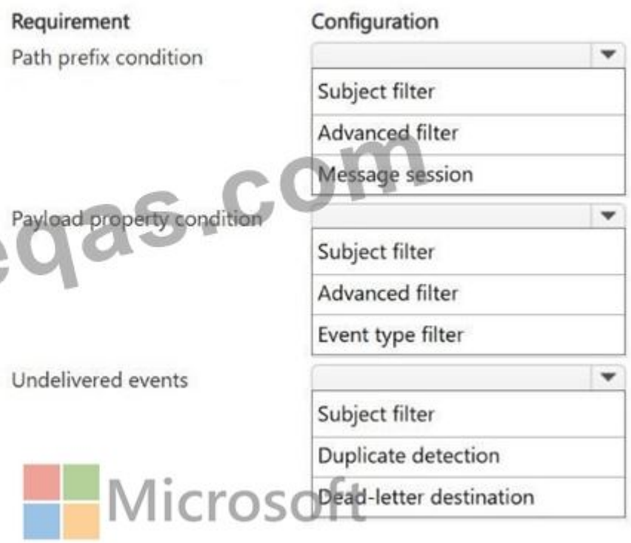
For finalized billing data, Eventual consistency provides the lowest latency and highest availability. Once discharge processing is complete and no further billing writes occur , replicas eventually converge on the same final value. Microsoft specifically notes that, in the absence of further writes, 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: 112
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: C,E (LEAVE A REPLY)
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: 113
Hotspot Question
You configure an Event Grid event subscription that routes AI file-upload events to an Azure Function endpoint.
Events must be delivered only in the following conditions:
- The event path starts with /uploads/ai/.
- The payload property data.fileType is "pdf."
You must store undelivered events for later investigation and reprocessing if the subscriber endpoint cannot accept an event after multiple retries.
You need to configure the filtering and reliability settings.
Which configurations should you use? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

Event Grid filtering and reliability

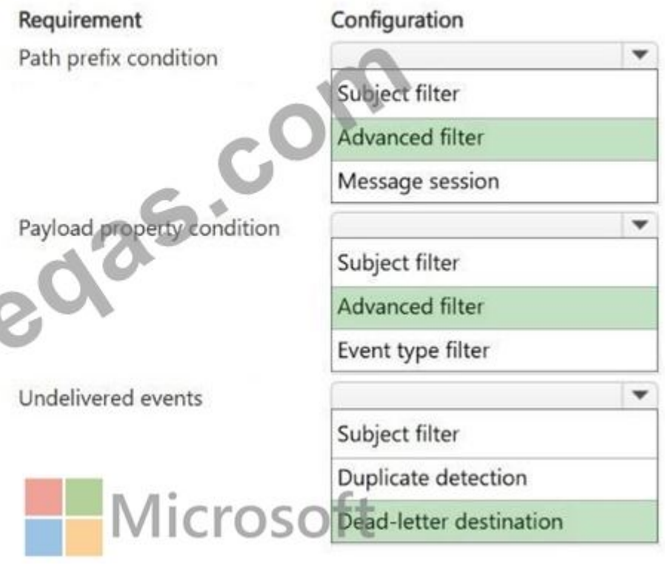


freeqas.com
Microsoft

Answer:

Answer Area

Event Grid filtering and reliability



freeqas.com
Microsoft

Explanation:

Box 1: Advanced filter

You should use an advanced filter to check that the path starts with /uploads/ai/ and that the payload property matches pdf.

Box 2: Advanced filter
Payload Property Condition: Use an Advanced Filter. While Subject filters handle prefix/suffix matching and Event Type filters handle the event schema type, Advanced filters are required to inspect deeply nested properties inside the event's data payload.
Box 3: Dead-letter destination
You should configure a Dead-letter destination to store undelivered events for later investigation and reprocessing.
Reference:
https://oneuptime.com/blog/post/2026-02-16-how-to-filter-azure-event-grid-subscription-events-with-advanced-filters/view
https://oneuptime.com/blog/post/2026-02-16-how-to-use-azure-event-grid-with-power-automate-to-react-to-blob-storage-events/view

NEW QUESTION: 114
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. Trigger invalidation when source data changes.
B. Set eviction policy to allkeys-lru.
C. Increase memory allocation.
D. Reset expiration on each read.
Answer: A (LEAVE A REPLY)
Trigger invalidation when source data changes is the best action.
Data Freshness: This is a proactive cache-aside or write-through approach. By explicitly deleting or updating the specific Redis key when the underlying database or model data changes, you completely eliminate the risk of serving stale data.
Low Overhead: It keeps cache management overhead minimal because invalidation logic is execution-dependent (event-driven). Instead of constantly updating expirations or relying on heavy computing logic, you only trigger an action when a write/change event occurs.
Incorrect:
[not B]
Set eviction policy to allkeys-lru:
This policy is a memory management feature, not a data-freshness strategy. It only deletes the Least Recently Used (LRU) keys when your Redis instance runs out of RAM. If memory usage is stable, outdated data will sit in the cache forever, posing a major risk of serving stale inferences.
[not D]
Reset expiration on each read (Sliding TTL):
This actively increases the risk of serving outdated data. If a specific AI response is highly popular and read continuously, its expiration window will extend indefinitely. The data remains stuck in the cache even if the source data undergoes critical updates.
Reference:
https://oneuptime.com/blog/post/2026-01-30-llm-caching-strategies/view

NEW QUESTION: 115
You are developing several microservices to run on Azure Container Apps. External HTTP ingress traffic has been enabled for the Microservices.
A deployed microservice must be updated to allow users to test new features. You have the following requirements:
* Enable and maintain a single URL for the updated microservice to provide to test users.
* Update the microservice that corresponds to the current microservice version.

You need to configure Azure Container Apps.
Which features should you configure? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area



Answer:

Answer Area



Explanation:

Answer Area



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

NEW QUESTION: 116
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: B (LEAVE A REPLY)
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: 117
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 = np.array([0.12, 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

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:

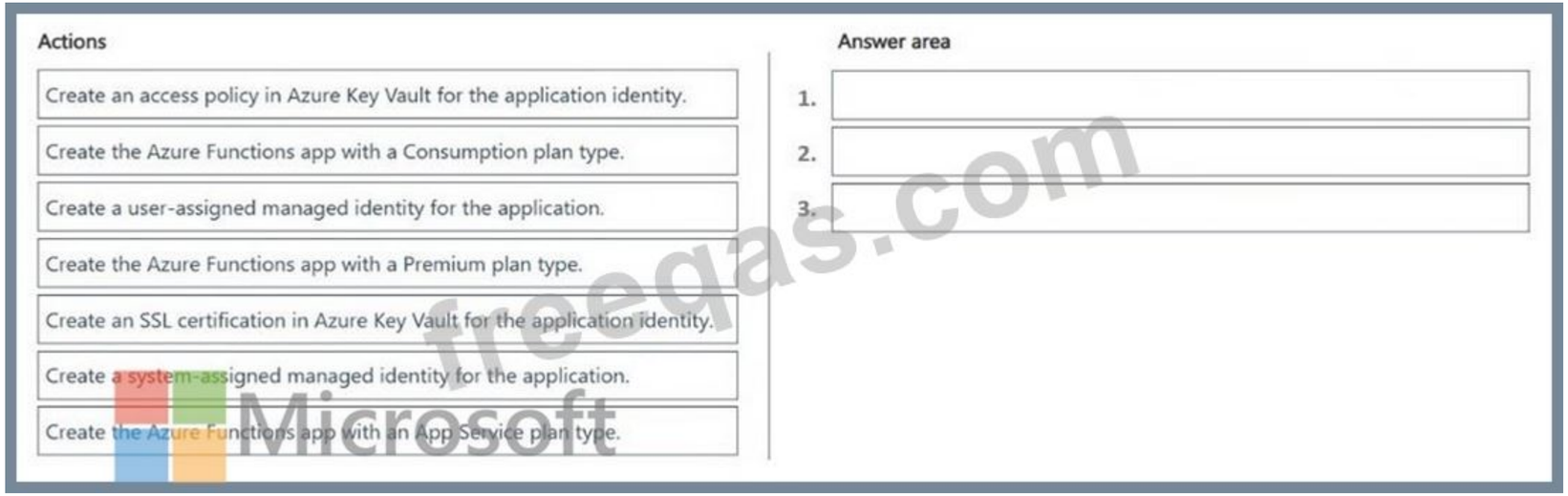
Vector search fundamentals

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:

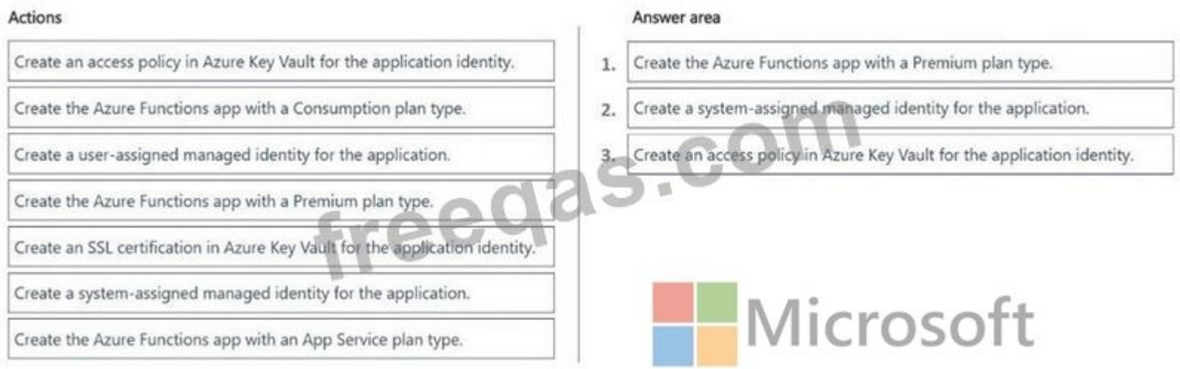
Verified answer: Yes; No; Yes.
Detailed Explanation: `HSET` stores the binary embedding as a field in the Redis hash keyed by `doc:1`, so the first statement is true. The code does not create a RediSearch/Redis Query Engine vector index or define a vector field schema; merely storing bytes does not enable 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: 118
Drag and Drop Question
You are developing a serverless Java application on Azure. You create a new Azure Key Vault to work with secrets from a new Azure Functions application.
The application must meet the following requirements:
- Reference the Azure Key Vault without requiring any changes to the
Java code.
- Dynamically add and remove instances of the Azure Functions host
based on the number of incoming application events.
- Ensure that instances are perpetually warm to avoid any cold starts.
- Connect to a VNet.
- Authentication to the Azure Key Vault instance must be removed if the Azure Functions application is deleted.
You need to grant the Azure Functions application access to the Azure Key Vault.
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.



freeqas.com
Microsoft

Answer:



freeqas.com

Explanation:
Step 1: Create the Azure Functions app with a Premium plan type
Perpetually Warm & VNet: The Premium plan type is the only serverless-scale hosting option that satisfies requirement 3 (keeping instances warm to eliminate cold starts) and requirement 4 (native regional VNet integration). The Consumption plan type does not keep instances warm and does not support VNet injection.
Event-Driven Scaling: It fulfills requirement 2 by dynamically adding and removing host instances using the event-driven scale controller.
Step 2: Create a system-assigned managed identity for the application
Lifecycle Coupling: It directly satisfies requirement 5. A system-assigned managed identity is tied strictly to the lifecycle of the Azure Functions resource. If the Functions application is deleted, Azure automatically removes the identity and its authentication token, ensuring that access to the Key Vault is immediately revoked. A user-assigned identity exists as an independent resource and would persist after the application's deletion.
Step 3: Create an access policy in Azure Key Vault for the application identity Granting Permissions: This step completes the access requirements. By adding an access policy (or an Azure RBAC role assignment) in Azure Key Vault specifically for the created system- assigned managed identity, you grant the runtime environment the necessary permissions to retrieve secrets.
No Code Changes: Combining the managed identity with an App Setting configured as an Azure Key Vault Reference allows the Functions platform to fetch secrets automatically at runtime, keeping your Java source code unchanged Reference:
https://learn.microsoft.com/en-us/azure/azure-functions/functions-best-practices

NEW QUESTION: 119
A company is developing a solution that allows smart refrigerators to send temperature information to a central location.
The solution must receive and store messages until they can be processed. You create an Azure Service Bus instance by providing a name, pricing tier, subscription, resource group, and location.
You need to complete the configuration.
Which Azure CLI or PowerShell command should you run?
A. Plaintext
az servicebus namespace create
--resource-group fridge-rg
--name fridge-ns
--location fridge-loc
B. Plaintext
az servicebus queue create
--resource-group fridge-rg
--namespace-name fridge-ns
--name fridge-q
C. Plaintext

Get-AzureRmServiceBusKey
-ResourceGroupName fridge-rg
-Namespace fridge-ns
-Name RootManageSharedAccessKey
D. Plaintext
New-AzureRmServiceBusNamespace
-ResourceGroupName fridge-rg
-NamespaceName fridge-ns
-Location fridge-loc
Answer: C (LEAVE A REPLY)

NEW QUESTION: 120
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: B (LEAVE A REPLY)
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: 121
You are using Azure Monitor Application Insights to collect dependency data.
You must be able to:
- Correlate failed requests with dependency calls during the last hour.
- Calculate the average dependency duration per operation.
You need to construct the KOL query by using the minimum number of statements.
Which three operators should you use? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.
A. extend
B. summarize
C. join
D. where

E. distinct
Answer: (SHOW ANSWER)
To satisfy both requirements with the minimum number of statements using the Kusto Query Language (KQL), you should use the following KQL operators:
where: Filters data by time range (ago(1h)) and request status (success == false).
join: Correlates the requests and dependencies tables using a unique identifier (usually operation_Id).
summarize: Computes the average (avg()) dependency duration grouped by the specific operation (operation_Name).
Reference:
https://learn.microsoft.com/en-us/azure/azure-monitor/app/application-insights-faq

Valid AI-200 Dumps shared by PrepPdf.com for Helping Passing AI-200 Exam! PrepPdf.com now offer the **newest AI-200 exam dumps**, the PrepPdf.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** PrepPdf.com AI-200 dumps with Test Engine here: https://www.preppdf.com/Microsoft/AI-200-prepaway-exam-dumps.html (**144** Q&As Dumps, **40%OFF** **Special Discount: Exam-Tests**)

NEW QUESTION: 122
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: B (LEAVE A REPLY)
The prebuilt invoice model in Azure AI Document Intelligence is trained to recognize common invoice fields (vendor, total, line items, invoice number) out of the box across varied layouts, requiring no training data or custom model.

Valid AI-200 Dumps shared by PrepPdf.com for Helping Passing AI-200 Exam! PrepPdf.com now offer the **newest AI-200 exam dumps**, the PrepPdf.com AI-200 exam **questions have been updated** and **answers have been corrected** get the **newest** PrepPdf.com AI-200 dumps with Test Engine here: https://www.preppdf.com/Microsoft/AI-200-prepaway-exam-dumps.html (**144** Q&As Dumps, **40%OFF** **Special Discount: Exam-Tests**)