DP-420$^{Q\&As}$

Designing and Implementing Cloud-Native Applications Using Microsoft Azure Cosmos DB

Pass Microsoft DP-420 Exam with 100% Guarantee

QUESTION 1

You have an Azure Cosmos DB for NoSQL account named account1 that supports an application named App1. App1 uses the consistent prefix consistency level.

You configure account1 to use a dedicated gateway and integrated cache.

You need to ensure that App1 can use the integrated cache.

Which two actions should you perform for APP1? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Change the connection mode to direct

B. Change the account endpoint to https://account1.sqlx.cosmos.azure.com.

C. Change the consistency level of requests to strong.

D. Change the consistency level of requests to session.

E. Change the account endpoint to https://account1.documents.azure.com

Correct Answer: BD

the Azure Cosmos DB integrated cache is an in-memory cache that is built- in to the Azure Cosmos DB dedicated gateway. The dedicated gateway is a front-end compute that stores cached data and routes requests to the backend database.

You can choose from a variety of dedicated gateway sizes based on the number of cores and memory needed for your workload1. The integrated cache can reduce the RU consumption and latency of read operations by serving them from

the cache instead of the backend containers2.

For your scenario, to ensure that App1 can use the integrated cache, you should perform these two actions:

Change the account endpoint to https://account1.sqlx.cosmos.azure.com. This is the dedicated gateway endpoint that you need to use to connect to your Azure Cosmos DB account and leverage the integrated cache. The standard gateway

endpoint (https://account1.documents.azure.com) will not use the integrated cache2.

Change the consistency level of requests to session. This is the highest consistency level that is supported by the integrated cache. If you use a higher consistency level (such as strong or bounded staleness), your requests will bypass the

integrated cache and go directly to the backend containers

QUESTION 2

You need to configure an Apache Kafka instance to ingest data from an Azure Cosmos DB Core (SQL) API account. The data from a container named telemetry must be added to a Kafka topic named iot. The solution must store the data in a compact binary format.

Which three configuration items should you include in the solution? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. "connector.class": "com.azure.cosmos.kafka.connect.source.CosmosDBSourceConnector"

B. "key.converter": "org.apache.kafka.connect.json.JsonConverter"

C. "key.converter": "io.confluent.connect.avro.AvroConverter"

D. "connect.cosmos.containers.topicmap": "iot#telemetry"

E. "connect.cosmos.containers.topicmap": "iot"

F. "connector.class": "com.azure.cosmos.kafka.connect.source.CosmosDBSinkConnector"

Correct Answer: CDF

C: Avro is binary format, while JSON is text.

F: Kafka Connect for Azure Cosmos DB is a connector to read from and write data to Azure Cosmos DB. The Azure Cosmos DB sink connector allows you to export data from Apache Kafka topics to an Azure Cosmos DB database. The

connector polls data from Kafka to write to containers in the database based on the topics subscription.

D: Create the Azure Cosmos DB sink connector in Kafka Connect. The following JSON body defines config for the sink connector.

Extract:

"connector.class": "com.azure.cosmos.kafka.connect.sink.CosmosDBSinkConnector",

"key.converter": "org.apache.kafka.connect.json.AvroConverter"

"connect.cosmos.containers.topicmap": "hotels#kafka"

Incorrect Answers:

B: JSON is plain text.

Note, full example:

```
{ "name": "cosmosdb-sink-connector", "config": {

 "connector.class": "com.azure.cosmos.kafka.connect.sink.CosmosDBSinkConnector",

 "tasks.max": "1",

 "topics": [

 "hotels"

 ],

 "value.converter": "org.apache.kafka.connect.json.AvroConverter",
"value.converter.schemas.enable": "false",

"key.converter": "org.apache.kafka.connect.json.AvroConverter",

"key.converter.schemas.enable": "false",

"connect.cosmos.connection.endpoint": "https://.documents.azure.com:443/",

"connect.cosmos.master.key": "",

"connect.cosmos.databasename": "kafkaconnect",

"connect.cosmos.containers.topicmap": "hotels#kafka"

} }
```

Reference:

https://docs.microsoft.com/en-us/azure/cosmos-db/sql/kafka-connector-sink

https://www.confluent.io/blog/kafka-connect-deep-dive-converters-serialization-explained/

QUESTION 3

You configure multi-region writes for account1.

You need to ensure that App1 supports the new configuration for account1. The solution must meet the business requirements and the product catalog requirements.

What should you do?

A. Set the default consistency level of account1 to bounded staleness.

B. Create a private endpoint connection.

C. Modify the connection policy of App1.

D. Increase the number of request units per second (RU/s) allocated to the con-product and con-productVendor containers.

Correct Answer: D

App1 queries the con-product and con-productVendor containers.

Note: Request unit is a performance currency abstracting the system resources such as CPU, IOPS, and memory that are required to perform the database operations supported by Azure Cosmos DB.

Scenario:

Develop an app named App1 that will run from all locations and query the data in account1.

Once multi-region writes are configured, maximize the performance of App1 queries against the data in account1.

Whenever there are multiple solutions for a requirement, select the solution that provides the best performance, as long as there are no additional costs associated.

Incorrect Answers:

A:

Bounded staleness relates to writes. App1 only do reads.

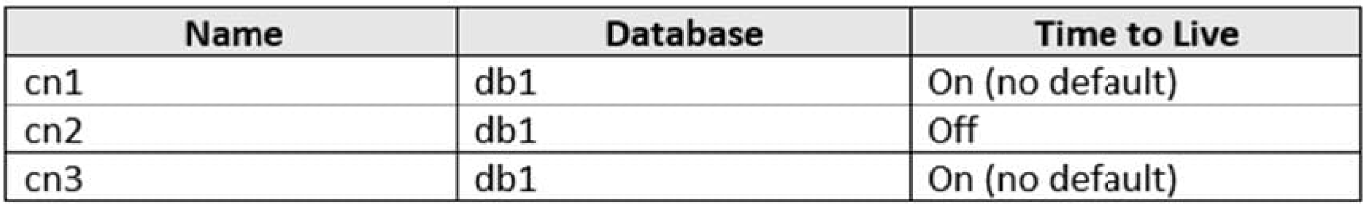
Note: Bounded staleness is frequently chosen by globally distributed applications that expect low write latencies but require total global order guarantee. Bounded staleness is great for applications featuring group collaboration and sharing, stock ticker, publish-subscribe/queueing etc.

Reference: https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels

QUESTION 4

HOTSPOT

You have three containers in an Azure Cosmos DB Core (SQL) API account as shown in the following table.



Name	Database	Time to Live
cn1	db1	On (no default)
cn2	db1	Off
cn3	db1	On (no default)

You have the following Azure functions:

1.

A function named Fn1 that reads the change feed of cn1

2.

A function named Fn2 that reads the change feed of cn2

3.

A function named Fn3 that reads the change feed of cn3 You perform the following actions:

1.

Delete an item named item1 from cn1.

2.

Update an item named item2 in cn2.

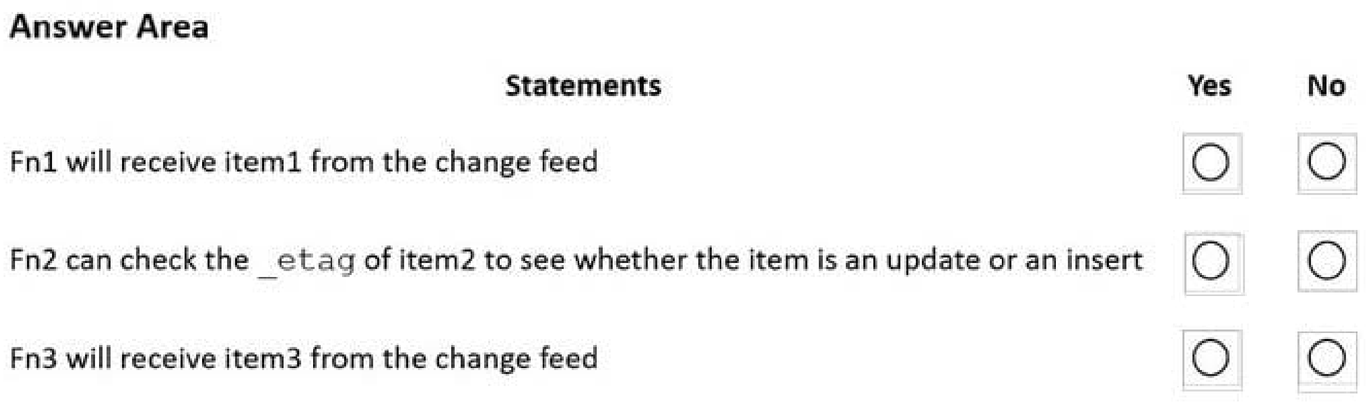
3.

For an item named item3 in cn3, update the item time to live to 3,600 seconds.

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

NOTE: Each correct selection is worth one point.

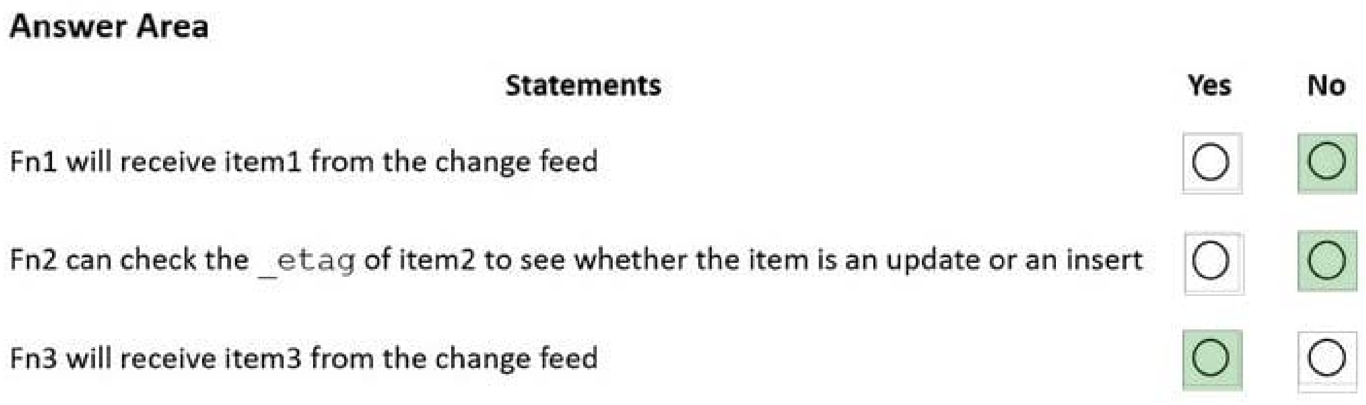
Hot Area:



Answer Area

Statements	Yes	No
Fn1 will receive item1 from the change feed	○	○
Fn2 can check the `_etag` of item2 to see whether the item is an update or an insert	○	○
Fn3 will receive item3 from the change feed	○	○

Correct Answer:



Answer Area

Statements	Yes	No
Fn1 will receive item1 from the change feed	○	[✓ ○]
Fn2 can check the `_etag` of item2 to see whether the item is an update or an insert	○	[✓ ○]
Fn3 will receive item3 from the change feed	[✓ ○]	○

Box 1: No

Azure Cosmos DB\\'s change feed is a great choice as a central data store in event sourcing architectures where all data ingestion is modeled as writes (no updates or deletes).

Note: The change feed does not capture deletes. If you delete an item from your container, it is also removed from the change feed. The most common method of handling this is adding a soft marker on the items that are being deleted. You

can add a property called "deleted" and set it to "true" at the time of deletion. This document update will show up in the change feed. You can set a TTL on this item so that it can be automatically deleted later.

Box 2: No

The _etag format is internal and you should not take dependency on it, because it can change anytime.

Box 3: Yes

Change feed support in Azure Cosmos DB works by listening to an Azure Cosmos container for any changes.

Reference:

https://docs.microsoft.com/en-us/azure/cosmos-db/sql/change-feed-design-patterns

https://docs.microsoft.com/en-us/azure/cosmos-db/change-feed

QUESTION 5

You have a global ecommerce application that stores data in an Azure Cosmos OB for NoSQL account. The account is contoured for multi-region writes.

You need to create a stored procedure for a custom conflict resolution policy for a new container.

In the event of a conflict caused by a deletion the deletion must always take priority.

Which parameter should you check m the stored procedure function?

A. conf1ictingItems

B. is Tombstone

C. existingitem

D. incoming1tem

Correct Answer: B