az 204 finalExam5

Done by Ahmed Fouad
Ahmed Fouad | LinkedIn

Nokhba Academy FB page

https://shorturl.at/UUDf2

1. Your team has customized a development environment VM with the necessary tools and configurations for a new Azure-based project.

Question 1: You need to generalize this VM to remove personal information and prepare it for replication as a template. Which tool or service should you use to generalize the VM?

A) Azure PowerShell

B) Visual Studio command prompt

C) Azure Migrate

D) Azure Backup

Answer: A

Feedback(if correct):- Azure PowerShell is the correct tool for generalizing a VM in Azure. It allows you to run the Sysprep command inside the VM, which generalizes the VM by removing all personal information and specific system data, making it suitable for capturing as an image. This process is essential for creating a template that can be used to deploy new VMs with the same configuration.

Feedback(if wrong):-

B) The Visual Studio command prompt is primarily used for development tasks and cannot generalize a VM for Azure.

C) Azure Migrate is a service designed to assist in the migration of applications, workloads, and data from on-premises environments or other cloud providers to Azure. It is not used for generalizing VMs.

D) Azure Backup is a service that provides backup solutions for Azure services. It does not offer functionalities for generalizing or preparing VMs as reusable templates.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solution

Competencies: Azure Networking and Web Services

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

2. Your team has customized a development environment VM with the necessary tools and configurations for a new Azure-based project.

Question 2: After generalizing your development environment VM, you aim to store its image for future provisioning of identical environments for new team members. Which Azure storage service should you choose to store the generalized VM image?

A) Azure Data Lake Storage

B) Azure File Storage

C) Azure Table Storage

D) Azure Blob Storage

Answer: D

Feedback(if correct):- Azure Blob Storage is the ideal service for storing VM images in Azure. It provides a scalable and secure destination for storing large amounts of unstructured data, such as VM images. Blob Storage supports the creation and management of disk images that can be used to provision new VMs, making it suitable for your requirement to store a generalized VM image for future use.

Feedback(if wrong):-

A) Azure Data Lake Storage is optimized for big data analytics and is not specifically designed for storing VM images.

B) Azure File Storage offers shared storage for applications using the standard SMB protocol. While it could technically store VM images, it's not as efficient or cost-effective as Blob Storage for this purpose.

C) Azure Table Storage is a NoSQL data store for semi-structured data. It's not suitable for storing large binary files like VM images, making it an incorrect choice for this requirement.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solution

Competencies: Azure Networking and Web Services

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

3. Your team has customized a development environment VM with the necessary tools and configurations for a new Azure-based project. To facilitate the quick setup for new project members, you need to replicate this customized environment efficiently. You decide to capture an image of the VM after generalizing it.

What is the correct sequence of actions to prepare and store this image for future use, and where should the image be stored?

A) Generalize the VM using Azure Migrate and store the image in Azure File Storage.

B) Generalize the VM using Azure PowerShell and store the image in Azure Blob Storage.

C) Generalize the VM using the Visual Studio command prompt and store the image in Azure Data Lake Storage.

D) Generalize the VM using Azure Backup and store the image in Azure Table Storage.

Answer: B

Feedback(if correct):- The correct sequence involves using Azure PowerShell to generalize the VM, which prepares it by removing system-specific data and personal information. Afterward, the image should be stored in Azure Blob Storage, which is optimized for storing large amounts of unstructured data, like VM images. This approach ensures that the development environment can be quickly replicated for new project members with minimal modifications.

Feedback(if wrong):-

A) Azure Migrate is primarily used for migrating workloads to Azure, not for generalizing VMs, and Azure File Storage is not the ideal service for storing VM images.

C) Visual Studio command prompt does not have built-in support for generalizing VMs in Azure, and Azure Data Lake Storage , while powerful for big data analytics, is not specifically designed for storing VM images.

D) Azure Backup is intended for data protection and backup solutions, not for generalizing VMs, and Azure Table Storage is a NoSQL data store that is not suitable for storing large binary files like VM images.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solution

Competencies: Azure Networking and Web Services

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

4. In the process of developing a large-scale processing solution utilizing Azure Batch, you've established a pool of compute nodes designated for executing numerous tasks. The next step involves the submission of a batch job to orchestrate these tasks efficiently. Identify the correct method to submit a batch job to the Azure Batch service from the options below:

A) `JobOperations.CreateJob()`

B) `CloudJob.Enable(IEnumerable<BatchClientBehavior>)`

C) `CloudJob.CommitAsync(IEnumerable<BatchClientBehavior>, CancellationToken)`

D) `JobOperations.EnableJob(String, IEnumerable<BatchClientBehavior>)`

Answer: C

Feedback(if correct):-

Selecting `CloudJob.CommitAsync(IEnumerable<BatchClientBehavior>, CancellationToken)` demonstrates an understanding of the asynchronous operations within Azure Batch services. This method is critical for submitting a configured batch job to the Azure Batch service, marking the transition from job creation to execution readiness. The use of `CommitAsync` signifies the candidate's grasp of effective job management practices in Azure Batch, emphasizing asynchronous programming patterns for efficient cloud resource utilization. This knowledge is essential for developers aiming to implement scalable, distributed computing solutions on the Azure platform, aligning with the skills tested in the AZ-204 exam.

Feedback(if wrong):-

Option A) `JobOperations.CreateJob()`: This option only creates a job object locally and does not submit it to the Azure Batch service. It is the initial step in defining a job but must be followed by `Commit` or `CommitAsync` to actually submit the job for execution.

Option B) `CloudJob.Enable(IEnumerable<BatchClientBehavior>)`: This method is intended to enable a previously disabled job, not to submit a new job. It assumes the job has already been created and submitted.

Option D) `JobOperations.EnableJob(String, IEnumerable<BatchClientBehavior>)`: Similar to option B, this is used for enabling a job rather than submitting a new one. It operates on the assumption that the job exists in the Batch service.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solution

Competencies: Azure Networking and Web Services

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

5. You are in charge of deploying a secure application onto an Azure Kubernetes Service (AKS) cluster. The application needs to be tightly secured and only accessible from within the Virtual Network (VNet) that encompasses the AKS cluster. You have decided to implement a Kubernetes Service with a LoadBalancer to achieve this, but with specific configurations to ensure the application is only internally accessible. Your task involves correctly configuring the deployment YAML file to adhere to these requirements.

Question 1: In the deployment YAML file, you need to define the service type to properly configure internal access.

What should the `kind` be set to in your YAML file to start configuring the service for internal access?

A) Ingress

B) Service

C) Deployment

D) Pod

Answer: B

Feedback(if correct):-

Selecting "Service" as the `kind` in the deployment YAML is correct because it signifies the intention to define a Kubernetes Service, which is essential for exposing applications running on a set of Pods as a network service. Since the goal is to create a LoadBalancer that restricts access within the VNet, beginning with a Service definition is the foundational step. This choice demonstrates an understanding of Kubernetes objects and their roles in managing and exposing applications within the cluster, aligning with best practices for deploying secure and scalable applications on Azure Kubernetes Service (AKS).

To configure the service for internal access, you should set the kind to Service and the type to ClusterIP. This will create a service that is only accessible within the cluster.

Here is an example of how to configure the service for internal access:

Edit

Full Screen

Copy code

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
  type: ClusterIP
```

This will create a service named my-service that selects Pods with the label app: MyApp and exposes them on port 80. The service will be accessible only within the cluster.

To make the service accessible from within the VNet, you can use a Kubernetes Service with a LoadBalancer and configure the load balancer to be internal. This will create a load balancer that is only accessible from within the VNet.

Here is an example of how to configure the service for internal access using a LoadBalancer:

Edit

Full Screen

Copy code

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
```

```
    port: 80
    targetPort: 9376
  type: LoadBalancer
  loadBalancerSourceRanges:
  - 10.0.0.0/8
```

This will create a service named my-service that selects Pods with the label app: MyApp and exposes them on port 80. The service will be accessible from within the VNet using the IP address of the load balancer.

Feedback(if wrong):-

Option A (Ingress): While Ingress can be used to manage external access to services in a Kubernetes cluster, choosing it as the starting point for configuring internal access is incorrect. Ingress is more suited for routing external HTTP(S) traffic to services, which does not align with the scenario's requirement for internal access only.

Option C (Deployment): A Deployment is used to define desired states for Pods, focusing on the application deployment aspect rather than the networking or service exposure. It's not the correct choice when the task is specifically about configuring network access via a Service.

Option D (Pod): Directly defining a Pod in this context is not suitable for the scenario's requirements. The question focuses on exposing an application through a network service, which is achieved through a Service object, not directly through individual Pods.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solution

Competencies: Azure Networking and Web Services

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

6. You are in charge of deploying a secure application onto an Azure Kubernetes Service (AKS) cluster. The application needs to be tightly secured and only accessible from within the Virtual Network (VNet) that encompasses the AKS cluster. You have decided to implement a Kubernetes Service with a LoadBalancer to achieve this, but with specific configurations to ensure the application is only internally accessible. Your task involves correctly configuring the deployment YAML file to adhere to these requirements.

Question 2: After setting the `kind` to `Service`, the next step involves specifying the correct type of service and annotation to ensure the load balancer is internal.

Which `type` and annotation should be included in your service configuration?

A) `type: ClusterIP` and `annotations: service.beta.kubernetes.io/azure-load-balancer-internal: "false"`

B) `type: LoadBalancer` and `annotations: service.beta.kubernetes.io/azure-load-balancer-internal: "true"`

C) `type: NodePort` and `annotations: service.beta.kubernetes.io/azure-load-balancer-internal: "false"`

D) `type: LoadBalancer` without any annotations

Answer: B

Feedback(if correct):-

Selecting `type: LoadBalancer` with the annotation `service.beta.kubernetes.io/azure-load-balancer-internal: "true"` is the correct configuration for ensuring that the service is only accessible internally within the Virtual Network (VNet). This setup instructs Azure to provision an internal load balancer for the AKS service, aligning with the requirement to restrict access within the VNet. It demonstrates a solid understanding of Kubernetes service types and Azure-specific annotations for managing network access and service exposure in a cloud-native environment.

To configure the LoadBalancer for internal access, you should set the type to LoadBalancer and include the annotation service.beta.kubernetes.io/azure-load-balancer-internal: "true". This will create a service that is accessible from the VNet and the load balancer will be internal.

Here is an example of how to configure the service for internal access:

Edit

Full Screen

Copy code

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  annotations:
    service.beta.kubernetes.io/azure-load-balancer-internal: "true"
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
```

```
    port: 80
    targetPort: 9376
  type: LoadBalancer
```

This will create a service named my-service that selects Pods with the label app: MyApp and exposes them on port 80. The service will be accessible from the VNet using the IP address of the load balancer.

Feedback(if wrong):-

Option A (`type: ClusterIP` and `annotations: service.beta.kubernetes.io/azure-load-balancer-internal: "false"`): While `ClusterIP` does make a service only accessible within the cluster, it does not utilize Azure's internal load balancer, and the annotation value `"false"` contradicts the requirement for internal-only access. This option does not meet the scenario's needs for VNet-restricted access.

Option C (`type: NodePort` and `annotations: service.beta.kubernetes.io/azure-load-balancer-internal: "false"`): `NodePort` exposes the service on each Node's IP at a static port, which is not suitable for the scenario aiming for internal access only. Furthermore, the annotation incorrectly indicates an external load balancer.

Option D (`type: LoadBalancer` without any annotations): Without the specific annotation to make the load balancer internal, Azure defaults to creating an external load balancer, which would expose the service outside of the VNet, contrary to the requirements.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solution

Competencies: Azure Networking and Web Services

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

7. You are in charge of deploying a secure application onto an Azure Kubernetes Service (AKS) cluster. The application needs to be tightly secured and only accessible from within the Virtual Network (VNet) that encompasses the AKS cluster. You have decided to implement a Kubernetes Service with a LoadBalancer to achieve this, but with specific configurations to ensure the application is only internally accessible. Your task involves correctly configuring the deployment YAML file to adhere to these requirements.

Question 3: After correctly setting the `type` of the service to `LoadBalancer` and specifying the annotation to use an internal load balancer by setting `service.beta.kubernetes.io/azure-load-balancer-internal` to `"true"`, the next step involves addressing the connectivity within the Virtual Network (VNet).

Should you explicitly configure the load balancer's frontend IP address to ensure the AKS application is accessible only within the VNet?

A) Yes, specify a frontend IP configuration in the service manifest to bind the internal load balancer to a specific subnet within the VNet.

B) No, the Azure Kubernetes Service (AKS) automatically assigns an appropriate frontend IP from the VNet subnet, making manual configuration unnecessary.

C) Yes, configuring a frontend IP is mandatory for all types of services in AKS, regardless of whether they are internal or external.

D) No, specifying a frontend IP address is only required when configuring external load balancers, not for internal load balancers.

Answer: A

Feedback(if correct):-

Choosing to explicitly configure the load balancer's frontend IP address is correct, as it allows for precise control over the service's accessibility within the Virtual Network (VNet). This step is particularly important in scenarios where the AKS application must be accessible only within a specific subnet or adhere to strict network security policies. By specifying a frontend IP configuration in the service manifest, you ensure that the internal load balancer is correctly bound to the desired subnet, enhancing the application's security and integration with the existing network infrastructure. This decision reflects an advanced understanding of AKS and Azure networking, demonstrating the ability to implement nuanced configurations for optimized application deployment and access control.

To ensure the AKS application is accessible only within the VNet, you should specify a frontend IP configuration in the service manifest to bind the internal load balancer to a specific subnet within the VNet.

Here is an example of how to configure the service for internal access:

Edit

Full Screen

Copy code

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  annotations:
    service.beta.kubernetes.io/azure-load-balancer-internal: "true"
```

```
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
  type: LoadBalancer
  loadBalancerIP: <frontend-ip-address>
```

This will create a service named my-service that selects Pods with the label app: MyApp and exposes them on port 80. The service will be accessible from the VNet using the specified frontend IP address.

Feedback(if wrong):-

Option B (No, automatic assignment): Relying solely on AKS to automatically assign an IP address from the VNet subnet may not always align with specific network configurations or security requirements. While AKS's automatic IP management is efficient, it might not offer the level of control needed for all scenarios, especially in tightly secured environments.

Option C (Mandatory configuration): While configuring a frontend IP can be crucial for certain scenarios, it's not mandatory for all AKS services. The necessity depends on the specific network and security requirements of the deployment.

Option D (External load balancers only): This choice misunderstands the role and configuration scope of internal vs. external load balancers. Explicitly configuring a frontend IP address can be just as important for internal load balancers, especially when precise network control and security are paramount.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solution

Competencies: Azure Networking and Web Services

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

8. As you architect an Azure Cosmos DB solution using the Azure Cosmos DB SQL API, your dataset comprises millions of documents, each featuring numerous properties without distinct values for partitioning. Your

objective is to evenly distribute the workload across partitions over time to ensure optimal application performance. In the design of your Azure Cosmos DB solution, focusing on even workload distribution and optimal performance, which consistency level would you choose to guarantee immediate and accurate data retrieval?

A. Bounded Staleness

B. Session Consistency

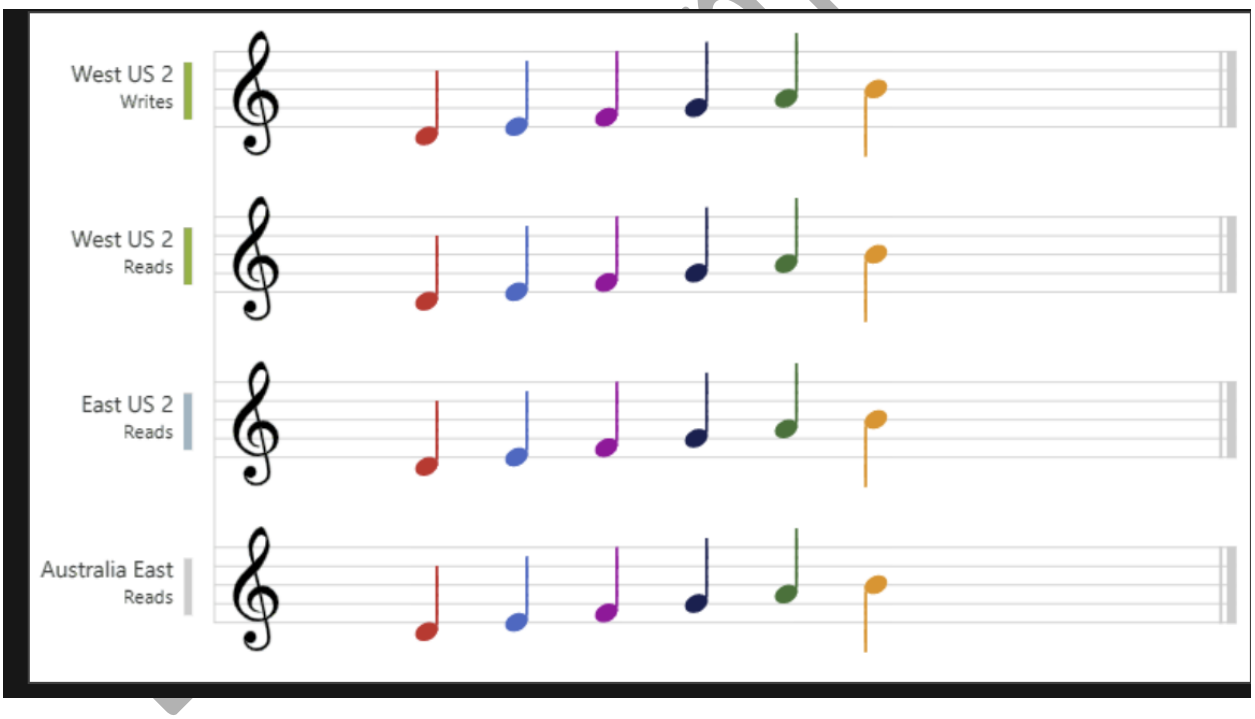
C. Consistent Prefix

D. Strong Consistency

Answer: D

Feedback (if correct):

Strong Consistency (Option D) is the correct choice for scenarios prioritizing immediate and accurate data retrieval, aligning to ensure data accuracy over latency.



Feedback (if wrong):

A. Bounded Staleness, B. Session Consistency, and C. Consistent Prefix are incorrect choices for scenarios prioritizing immediate and accurate data retrieval. They do not align to ensure data accuracy over latency.

Skill mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage.

Competency: Develop Azure Cosmos DB solutions, Choose appropriate consistency levels for Azure Cosmos DB

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

9. As you architect an Azure Cosmos DB solution using the Azure Cosmos DB SQL API, your dataset comprises millions of documents, each featuring numerous properties without distinct values for partitioning. Considering the Azure Cosmos DB solution's design with a focus on even workload distribution and optimal performance, which consistency level provides a compromise between strong consistency and availability, ensuring linearizability within a partition?

A. Bounded Staleness

B. Session Consistency

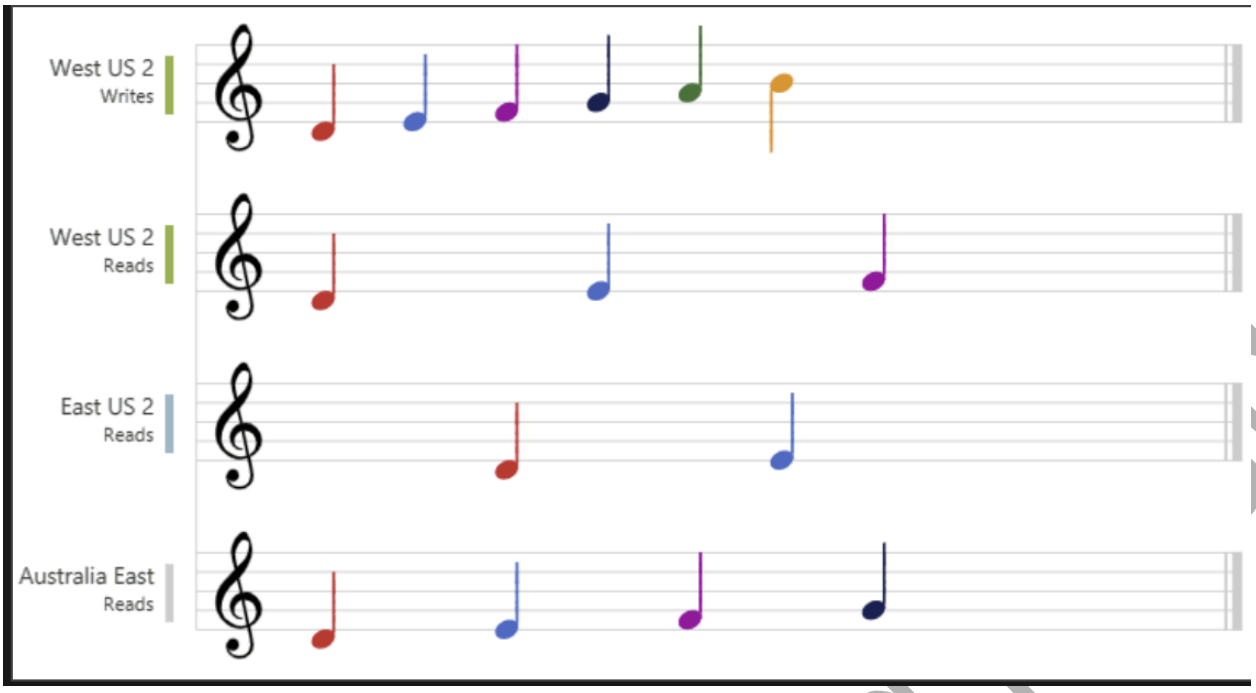
C. Consistent Prefix

D. Eventual Consistency

Answer: C

Feedback (if correct):

Consistent Prefix (Option C) is the correct choice for scenarios seeking a compromise between strong consistency and availability. It ensures linearizability within a partition while allowing for higher availability.



Feedback (if wrong):

A Bounded Staleness, B. Session Consistency, and D. Eventual Consistency are incorrect choices for scenarios seeking a compromise between strong consistency and availability. Bounded Staleness provides a specified lag between replicas but may not achieve the desired balance between consistency and availability. Session Consistency ensures consistency within a session but does not specifically address the requirement for linearizability within a partition. Eventual Consistency allows for eventual convergence of data but does not guarantee linearizability within a partition. Therefore, Consistent Prefix (Option C) is the appropriate choice as it provides a compromise between strong consistency and availability by ensuring linearizability within a partition while allowing for higher availability.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage.

Competency: Develop Azure Cosmos DB solutions, Choose appropriate consistency levels for Azure Cosmos DB

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

10. As you architect an Azure Cosmos DB solution using the Azure Cosmos DB SQL API, your dataset comprises millions of documents, each featuring numerous properties without distinct values for partitioning. which consistency level would be suitable for scenarios requiring a specified lag between replicas to balance consistency and performance?

A. Strong Consistency

B. Session Consistency

C. Bounded Staleness

D. Eventual Consistency

Answer: C

Feedback (if correct):

Bounded Staleness (Option C) is the right choice for scenarios requiring a specified lag between replicas to balance consistency and performance. This provides flexibility in achieving consistency goals.

Feedback (if wrong):

Strong Consistency, B. Session Consistency, and D. Eventual Consistency are incorrect choices for scenarios requiring a specified lag between replicas to balance consistency and performance. Strong Consistency guarantees immediate consistency across all replicas, which may not be suitable for scenarios where a specified lag is needed. Session Consistency ensures consistency within a session but does not address the requirement for a lag between replicas. Eventual Consistency allows for eventual convergence of data across replicas but does not provide the specified lag between them. Therefore, Bounded Staleness is the appropriate choice as it allows defining a lag between replicas to balance consistency and performance.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage.

Competency: Develop Azure Cosmos DB solutions, Choose appropriate consistency levels for Azure Cosmos DB

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

11. As you architect an Azure Cosmos DB solution using the Azure Cosmos DB SQL API, your dataset comprises millions of documents, each featuring numerous properties without distinct values for partitioning, which consistency level would be suitable for scenarios where monotonic reads and writes within a session are essential, allowing temporary inconsistency between sessions?

A. Strong Consistency

B. Session Consistency

C. Bounded Staleness

D. Eventual Consistency

Answer: B

Feedback (if correct):

Session Consistency (Option B) is the correct choice for scenarios requiring monotonic reads and writes within a session while allowing temporary inconsistency between sessions. It ensures data consistency within a session.

Feedback (if wrong):

A Strong Consistency, C. Bounded Staleness, and D. Eventual Consistency are incorrect choices for scenarios requiring monotonic reads and writes within a session. They do not provide the necessary balance between session consistency and flexibility. Strong Consistency guarantees immediate consistency across all replicas, which may lead to higher latency and reduced availability. Bounded Staleness offers consistency guarantees within a specified lag time or version, but it may not be suitable for scenarios requiring monotonic reads and writes within a session. Eventual Consistency allows for eventual convergence of data across replicas but does not ensure consistency within a session.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage.

Competency: Develop Azure Cosmos DB solutions, Choose appropriate consistency levels for Azure Cosmos DB

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

12. As you design your Azure Cosmos DB solution to achieve even workload distribution and optimal performance, which consistency level is suitable for scenarios where low-latency reads are prioritized over the most recent writes?

A. Strong Consistency

B. Session Consistency

C. Consistent Prefix

D. Eventual Consistency

Answer: D

Feedback (if correct):

Eventual Consistency (Option D) is the correct choice for scenarios prioritizing low-latency reads over the most recent writes. It allows for the quick availability of data with the understanding that eventual consistency will be achieved.

Feedback (if wrong):

A. Strong Consistency, B. Session Consistency, and C. Consistent Prefix are incorrect choices for scenarios prioritizing low-latency reads over recent writes. They do not align to achieve eventual consistency.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage.

Competency: Develop Azure Cosmos DB solutions, Choose appropriate consistency levels for Azure Cosmos DB

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

13. To automate moving specific audio files between two Azure Blob Storage containers meeting particular criteria, what approach should be implemented without including duplicates while maximizing efficiency, safety, and expandability?

A) Event Grid Trigger Function + AzCopy

B) Logic App Workflow + Serverless Function

C) Custom Script + Azure CLI + AzCopy

D) Azure Data Factory Pipelines + Tumbling Window Trigger

Answer: B

Feedback(if correct):- Selecting a monitoring and filtering solution integrated with serverless computing strikes the perfect balance between functional simplicity, low maintenance, and dynamic response.

Feedback(if wrong):- Relying solely on event-driven functions increases development effort, reduces maintainability, and lacks inherent filtering capabilities. Writing personalized scripts offers adaptability at the expense of greater complexity, reduced maintainability, and potentially slower responses. Utilizing ETL tools brings power but adds complexity, making it less suited for straightforward file movements requiring minimal transformations.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage.

Competency: Develop Azure Cosmos DB solutions, Choose appropriate consistency levels for Azure Cosmos DB

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

14. Manufacturing stores critical production information in Azure Blob Storage. Every month, admins shift blobs from the hot storage tier to the archive tier. Now, you've been assigned to devise a technique for automatically moving unaccessed blocks to the archive tier once they reach 180 days of idleness, placing non-archived items' paths onto an existing queue. This procedure should occur monthly. With the given settings, how would you properly arrange the Logic App configuration?

A) If blob aged > 180d, Move to Archive & Append to Queue

B) Schedule check, Delay, Get props, Change tier, Add msg to Q

C) Monthly scan for blobs untouched past 180d, update Q

D) When blob changed, delay, verify age, act, send msg to Q

Answer: A

Feedback (correct): Setting a condition for aging blobs combined with moving them to a lower tier and appending related info to a queue ensures proper management of infrequently accessed data and supports further analysis.

Feedback (wrong):

Scheduling frequent checks with delays may lead to decreased efficiency and elevated costs, whereas nested actions increase complexity without significantly improving results. Periodic scans might overlook recent changes near the cutoff date, resulting in inconsistencies and missed opportunities for early tier promotion. An overcomplicated design involving loops and variables may result in poor performance and heightened susceptibility to errors, diminishing reliability and robustness.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage.

Competency: Develop Azure Cosmos DB solutions, Choose appropriate consistency levels for Azure Cosmos DB

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

15. As a developer, suppose your employer heavily relies on Azure Blob Storage for massive quantities of data, aiming to minimize expenses and expedite data access. In light of this, which Blob Storage characteristic or accompanying service would prove most beneficial in tackling this assignment?

A) Harness Blob Storage lifecycle management policies

B) Invoke Azure Archive Storage

C) Engage Blob Storage rehydration

D) Take advantage of Azure Blob Storage tiers

Answer: D

Feedback (if correct):- Leveraging Azure Blob Storage tiers lets you strike an optimal balance between data access speed and cost reduction by dynamically shifting between access tiers.

Feedback (if wrong):

Blob Storage lifecycle management policies allow automating data management tasks, e.g., expiration, but won't necessarily optimize cost and speed simultaneously. Azure Archive Storage suits storing rarely accessed data for long-term retention, but restoration takes time and defeats the purpose of fast access. Blob Storage rehydration applies when reverting data from Archive to a warmer tier, also causing delayed access.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage

Competency: Optimize Azure storage solutions

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

16. You are tasked with managing application data stored in an Azure blob named "data" for an application. The application frequently creates snapshots of the blob to capture its state at different points in time. The storage account is configured with soft delete enabled.

The system performs the following operations chronologically:

1. The blob is updated.

2. Snapshot 1 is created.

3. Snapshot 2 is created.

4. Snapshot 1 is deleted.

Due to a system error, the data blob and all snapshots are accidentally deleted. You need to assess which application states can be restored.

What is the restorability of the application data?

A) All application states can be restored.

B) Only Snapshot 2 can be restored.

C) No application states can be restored.

D) Both Snapshot 1 and Snapshot 2 can be restored.

Answer: D

Feedback (if correct): Both Snapshot 1 and Snapshot 2 can be restored because soft delete allows for the recovery of deleted blobs and snapshots within the retention period, ensuring that any deleted snapshots can be restored to their previous state.

Feedback (if wrong): Option A is incorrect because not all application states can be restored; only the snapshots created before deletion can be recovered. Option B is incorrect because only Snapshot 2 was created before deletion. Option C is incorrect because soft delete enables the recovery of deleted blobs and snapshots within the retention period. Option D is incorrect because Snapshot 1 was deleted before the system error occurred, so only Snapshot 2 can be restored.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage

Competencies: Understanding blob snapshots, soft delete feature, and restoration of deleted data

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Analysis

17. You are developing an application that utilizes Azure Blob Storage to store critical data. The application regularly creates snapshots of the blobs to enable rollback to previous states. Soft delete is enabled on the storage account. After a series of operations, including updates and snapshot creation, an unexpected system error

occurs, resulting in the deletion of the data blob and all snapshots. Which of the following accurately describes the restorability of the application data?

A) The data blob and all snapshots can be restored entirely from the soft delete feature.

B) Only the snapshots can be restored; the data blob cannot be recovered.

C) Both the data blob and all snapshots cannot be restored due to the system error.

D) The data blob can be restored, but the snapshots are irretrievable.

Answer: A

Feedback (if correct):

Option A is correct. Soft delete enables the recovery of deleted blobs and their snapshots, ensuring that both the data blob and all snapshots can be restored from the storage account.

Feedback (if wrong):

B) Only the snapshots can be restored; the data blob cannot be recovered.

- This option is incorrect because soft delete enables the recovery of both blobs and their snapshots. Since soft delete was enabled on the storage account, both the data blob and the snapshots can be restored.

C) Both the data blob and all snapshots cannot be restored due to the system error.

- This option is incorrect because soft delete provides a mechanism to recover deleted blobs and snapshots, regardless of the cause of deletion. As long as soft delete was enabled before the deletion occurred, both the data blob and its snapshots can still be restored.

D) The data blob can be restored, but the snapshots are irretrievable.

- This option is incorrect because soft delete applies to both blobs and snapshots. If soft delete is enabled, both the data blob and its snapshots can be restored from the storage account. Therefore, neither the data blob nor the snapshots are irretrievable if soft delete is properly configured.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage

Competencies: Understanding blob snapshots, soft delete feature, and restoration of deleted data

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Analysis

18. You are designing a healthcare application with specific consistency requirements. You have provisioned an Azure Cosmos DB NoSQL database with a default consistency level set to Strong and Indexing Mode set to Consistent. To meet the application requirements while minimizing latency and ensuring availability, you need to override the default consistency level at the query level. Match the appropriate consistency levels to the corresponding requirements:

Provides the strongest consistency level and ensures that the application will always get the most recent data, even if that data is still in the process of being updated.

A) Strong

B) Bounded Staleness:

C) Eventual

D) Consistent Prefix

Answer: A

Feedback(if correct):

Option A (Strong) is correct. Strong consistency provides the strongest consistency level, ensuring that the application always receives the most recent data, even if it is still being updated.

Feedback(if wrong):

B) Bounded Staleness: This consistency level guarantees that data is at least a certain number of versions behind the most recent one, but it doesn't ensure getting the most recent data.

C) Eventual: Eventual consistency does not provide strong guarantees about the recency of the data and may result in receiving outdated information.

D) Consistent Prefix: While Consistent Prefix guarantees that the reads honor the order of writes, it doesn't ensure receiving the most recent data.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage.

Competency: Develop Azure Cosmos DB solutions, Choose appropriate consistency levels for Azure Cosmos DB

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

19. You are tasked with enhancing the messaging system of a distributed inventory management application for a retail chain. The application is designed to update store inventories in real-time as sales happen across different locations. To streamline communication between the stores and the central inventory system, you decide to utilize Azure Service Bus.

Requirement:

The system must support the following functionalities:

1. Send inventory updates from individual stores to the central system.

2. Ensure that updates are processed in a scalable manner as new stores are added.

3. Allow the central system to send specific directives to stores based on inventory needs.

To set up the Azure Service Bus for the inventory management system, which three actions should you perform in sequence? Select the correct order of actions from the options below.

A) Create a single Service Bus Namespace, Create a single Service Bus topic, and Create a Service Bus subscription for each store.

B) Create a Service Bus Namespace for each store, Create a Service Bus topic for each inventory update, and Create a single Service Bus subscription.

C) Create a single Service Bus Namespace, Create a Service Bus topic for each store, and Create a single Service Bus subscription.

D) Create a single Service Bus Namespace, Create a Service Bus subscription for each inventory update, and Create a single Service Bus topic.

Answer: A

Feedback(if correct):-

1. Create a single Service Bus Namespace : A namespace provides a scoping container for addressing Service Bus resources within your application, allowing for a centralized management point for all stores.

2. Create a single Service Bus topic : A topic categorizes messages and supports multiple subscriptions. Using a single topic for inventory updates simplifies the architecture and enables scalable communication from all stores.

3. Create a Service Bus subscription for each store : Subscriptions to the topic allow for messages to be filtered so that only relevant updates and directives are received by each store, ensuring targeted communication.

This setup leverages the Azure Service Bus's topics and subscriptions to efficiently manage and scale communications across multiple stores in the inventory management system.

Feedback(if wrong):-

Create a Service Bus topic for each event type.

- This approach would not be scalable or efficient for an event management system that handles a large number of events. Creating a separate topic for each event type would lead to unnecessary complexity and administrative overhead, especially as the number of events grows. It could also lead to resource constraints within the Service Bus namespace due to the potential for a large number of topics.

Create a single Service Bus subscription.

- Using a single subscription for all types of notifications would not allow for targeted communications. Attendees would receive all notifications, regardless of relevance, leading to a poor user experience. It doesn't meet the requirement for attendees to receive updates and notifications relevant only to their registered events.

Create a Service Bus Namespace for each event type.

- Similar to creating a topic for each event type, creating a namespace for each event would be highly inefficient and costly. A namespace is a scoping container intended to group related messaging components. Using multiple namespaces for different event types would complicate management and access control and increase costs without providing tangible benefits.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop for Azure storage

Competencies: Azure Services Integration, Azure Service Bus

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application, Analysis

1. You are optimizing the database queries for a healthcare application using Azure Cosmos DB. The queries need to sort patient records first by the last visit date and then by the patient's name. To achieve optimal query performance, you're tasked with configuring the composite indexes in the Cosmos DB indexing policy.

Given the partial indexing policy configuration:

```json
{
  "indexingPolicy": {
    "compositeIndexes": [
      [
        {
```

```
        "path": "slot1",
        "order": "slot2"
      },
      {
        "path": "slot3",
        "order": "slot4"
      }
    ]
  ]
 }
}
```

Question 1: Selecting the Path for the Last Visit Date (`slot1`)

What should replace `slot1` to correctly configure the composite index for sorting by the last visit date?

A) `"/lastVisitDate"`

B) `"/name"`

C) `"ascending"`

D) `"descending"`

Answer: A

Feedback(if correct)

Selecting `"/lastVisitDate"` as the correct answer for `slot1` ensures that the composite index is correctly configured to sort patient records by the last visit date. This choice aligns with the requirement to prioritize the sorting of patient records based on the last visit date, facilitating efficient query performance that matches the application's needs for organizing and accessing patient information. It demonstrates a clear understanding of how to structure composite indexes in Azure Cosmos DB to support specific query patterns.

Feedback(if wrong):

Option B) `"/name"`: This option incorrectly prioritizes the patient's name over the last visit date for the initial sorting criterion, which does not align with the specified requirement to sort by the last visit date first.

Option C) `"ascending"`: This choice represents a sort order rather than a path for the data to be sorted by. It is misplaced in this context, where a field path is expected.

Option D) `"descending"`: Similar to option C, this option specifies a sort order rather than the required field path. It does not fit the requirement for `slot1`, which needs to identify the data field for sorting.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage

Competency: Optimize Azure storage solutions

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

20. You are optimizing the database queries for a healthcare application using Azure Cosmos DB. The queries need to sort patient records first by the last visit date and then by the patient's name. To achieve optimal query performance, you're tasked with configuring the composite indexes in the Cosmos DB indexing policy.

Given the partial indexing policy configuration:

```json
{
  "indexingPolicy": {
    "compositeIndexes": [
      [
        {
          "path": "slot1",
          "order": "slot2"
```

```
      },
      {
        "path": "slot3",
        "order": "slot4"
      }
    ]
  ]
 }
}
```

Question 2: Selecting the Order for the Last Visit Date (`slot2`)

What should replace `slot2` to specify the order for the last visit date in the composite index?

A) `"ascending"`

B) `"descending"`

C) `"/lastVisitDate"`

D) `"/name"`

Answer: A

Feedback(if correct):

Choosing `"ascending"` for `slot2` correctly specifies the order for sorting by the last visit date within the composite index. This decision ensures that patient records are sorted from the earliest to the most recent visit date, adhering to the query's requirement to efficiently organize patient data in a way that is intuitive and useful for healthcare providers. It highlights an accurate understanding of applying sort orders in composite indexes to meet specific data retrieval needs in Azure Cosmos DB.

Feedback(if wrong):

Option B) `"descending"`: Selecting this option would sort patient records starting from the most recent to the earliest visit date, which contradicts the scenario's requirement to sort by the last visit date in ascending order.

Option C) `"/lastVisitDate"`: This option mistakenly identifies a path instead of a sort order. The question specifically asks for the order in which the data should be sorted, not the field to be sorted.

Option D) `"/name"`: Like option C, this response incorrectly selects a field path when the question requires a sort order. It demonstrates a misunderstanding of the distinction between specifying the field to sort on (which is addressed in another question) and the direction in which the sorting should occur.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage

Competency: Optimize Azure storage solutions

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

21. You are optimizing the database queries for a healthcare application using Azure Cosmos DB. The queries need to sort patient records first by the last visit date and then by the patient's name. To achieve optimal query performance, you're tasked with configuring the composite indexes in the Cosmos DB indexing policy.

Given the partial indexing policy configuration:

```json
{
  "indexingPolicy": {
    "compositeIndexes": [
      [
        {
          "path": "slot1",
          "order": "slot2"
        },
        {
          "path": "slot3",
          "order": "slot4"
```

```
      }
    ]
  ]
 }
}
```

Question 3: Selecting the Path for the Patient's Name (`slot3`)

What should replace `slot3` to correctly configure the composite index for sorting by the patient's name?

A) `"/name"`

B) `"/lastVisitDate"`

C) `"ascending"`

D) `"descending"`

Answer: A

Feedback(if correct):

Selecting `"/name"` as the correct answer for `slot3` is the right choice because it configures the composite index to sort patient records alphabetically by the patient's name after initially sorting by the last visit date. This configuration aligns with the requirement to further organize patient records that have the same last visit date, making it easier to navigate and retrieve specific data. It showcases an understanding of how to effectively use composite indexes in Azure Cosmos DB to meet complex query requirements.

Feedback(if wrong):

Option B) `"/lastVisitDate"`: Choosing this option would incorrectly suggest sorting by the last visit date again, which is redundant and does not meet the requirement to sort by the patient's name as a secondary criterion.

Option C) `"ascending"`: This choice represents a sort order rather than the path of the data to be sorted, which is not the information needed for `slot3`. The question seeks the field path for the secondary sorting criterion, not the order.

Option D) `"descending"`: Similar to option C, this option specifies a sort order and not a field path. It fails to address the requirement for `slot3`, which is to identify the secondary field (the patient's name) by which the data should be sorted.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage

Competency: Optimize Azure storage solutions

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

22. You are optimizing the database queries for a healthcare application using Azure Cosmos DB. The queries need to sort patient records first by the last visit date and then by the patient's name. To achieve optimal query performance, you're tasked with configuring the composite indexes in the Cosmos DB indexing policy.

Given the partial indexing policy configuration:

```json
{
  "indexingPolicy": {
    "compositeIndexes": [
      [
        {
          "path": "slot1",
          "order": "slot2"
        },
        {
          "path": "slot3",
          "order": "slot4"
        }
      ]
    ]
```

```
  }
}
```

Question 4: Selecting the Order for the Patient's Name (`slot4`)

What should replace `slot4` to specify the order for the patient's name in the composite index?

A) `"ascending"`

B) `"descending"`

C) `"/lastVisitDate"`

D) `"/name"`

Answer: A

Feedback(if correct):

Choosing `"ascending"` for `slot4` accurately specifies the order for alphabetically sorting patient records by name for those with the same last visit date. This selection ensures that the composite index supports the application's need for an intuitive and practical way to access patient information, particularly when multiple records share the same last visit date. It demonstrates a correct application of sort orders within composite indexes in Azure Cosmos DB, tailoring data retrieval to specific user requirements.

Feedback(if wrong):

Option B) `"descending"`: This choice would result in patient records being sorted in reverse alphabetical order by name, which contradicts the specified requirement for ascending alphabetical order. It would not support the intended user experience for navigating patient records.

Option C) `"/lastVisitDate"`: Selecting this option indicates a misunderstanding, as it suggests a path for sorting rather than the required sort order. The question aims to identify the correct order for the secondary sorting criterion, not another field to sort by.

Option D) `"/name"`: Like option C, this response incorrectly selects a field path when the question requires a sort order. It demonstrates a confusion between specifying what to sort by (already addressed in `slot3`) and how that sorting should be performed (ascending or descending order).

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage

Competency: Optimize Azure storage solutions

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

23. As part of the development team for a healthcare management system, you are tasked with enhancing the system's data retrieval process. The system stores patient records in Azure Cosmos DB. To improve the performance of queries that retrieve patient data in a specific order based on the patient's last visit date and then alphabetically by their name, you need to adjust the Cosmos DB indexing policy.

Given the partial Cosmos DB indexing policy configuration below, select the correct JSON segments from the list provided to complete the policy and support efficient query performance as required.

Partial Cosmos DB Indexing Policy Configuration:

```json
{
  "indexingPolicy": {
    "automatic": true,
    "indexingMode": "Consistent",
    "includedPaths": [
      {
        "path": "/ ",
        "indexes": [
          {
            "kind": "Range",
            "dataType": "String",
            "precision": -1
          },
          {
            "kind": "Range",
            "dataType": "Number",
```

```
          "precision": -1
        }
       ]
     }
   ],
   "excludedPaths": [
    {
      "path": "/\"_etag\"/?"
    }
   ],
   "compositeIndexes": [
    [
     {
       "path": "slot1",
       "order": "slot2"
     },
     {
       "path": "slot3",
       "order": "slot4"
     }
    ]
   ]
  }
}
```

Select the correct JSON segments for `slot1`, `slot2`, `slot3`, and `slot4` from the following list:

- orderBy
- sortOrder
- ascending
- descending

- compositeIndexes

Which selection from the following fits into the code segment to meet the query requirements?

A) `slot1`: `/lastVisitDate`, `slot2`: `ascending`, `slot3`: `/name`, `slot4`: `ascending`

B) `slot1`: `/name`, `slot2`: `descending`, `slot3`: `/lastVisitDate`, `slot4`: `descending`

C) `slot1`: `/lastVisitDate`, `slot2`: `descending`, `slot3`: `/name`, `slot4`: `ascending`

D) `slot1`: `/name`, `slot2`: `ascending`, `slot3`: `/lastVisitDate`, `slot4`: `descending`

Answer: A

Feedback(if correct):

The correct selection, Option A , specifies `slot1` as `/lastVisitDate` with `slot2` as `ascending`, and `slot3` as `/name` with `slot4` as `ascending`. This configuration aligns precisely with the requirement to retrieve patient data sorted first by the last visit date in ascending order and then by name, also in ascending order. It supports efficient query performance by ensuring that the Azure Cosmos DB indexing policy is optimized for the desired query pattern, enabling quick retrieval of patient records in the specified order.

Feedback(if wrong):

Option B suggests sorting both `/name` and `/lastVisitDate` in descending order. This configuration contradicts the requirement to sort the data based on the last visit date in ascending order, making it inefficient for the specified query needs.

Option C positions `/lastVisitDate` in descending order, which is incorrect for the requirement to sort patient records starting from the earliest to the latest visit. Although it correctly suggests sorting `/name` in ascending order, the initial sorting by `/lastVisitDate` in descending order does not meet the query's requirements.

Option D inaccurately suggests sorting `/name` in ascending order and `/lastVisitDate` in descending order. This arrangement fails to prioritize the last visit date as the primary sorting criterion in ascending order, essential for achieving the desired query performance and accuracy in data retrieval.

The concise explanations for the incorrect options highlight the importance of closely aligning the indexing policy configuration with the specific requirements of the query pattern. Understanding how to correctly apply sorting orders in composite indexes is crucial for optimizing Azure Cosmos DB's performance to meet application needs.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Develop for Azure storage

Competency: Optimize Azure storage solutions

Difficulty Level: Intermediate

Blooms Taxonomy Level: Application

24. You are managing a web application hosted on Azure App Service. This application requires access to sensitive information stored in Azure Key Vault. To ensure secure access, you have implemented Azure Active Directory authentication and configured an Azure Key Vault access policy. You are reviewing the PowerShell commands used to manage the Key Vault access policy. The commands include an XML segment placed within the <inbound> section of the policy definition. Additionally, the policy definition includes a size restriction for incoming requests. Furthermore, the policy seems to prioritize specific API versions based on the request URL.

Based on your understanding of Azure Key Vault access policies and Azure App Service security best practices, analyze the following statements and select "Yes" if the statement is true, otherwise select "No":

Statements:

Placing the XML segment in the <inbound> section of the policy ensures it handles incoming requests effectively.

If the size of the request body exceeds 256 KB, an error will be triggered.

The policy will prioritize the higher version of the API if the incoming request is directed to http://contoso.com/api/9.2/.

A) Yes, No, Yes

B) No, Yes, No

C) Yes, Yes, No

D) No, No, Yes

Answer:D

Feedback(if correct):-

Correct Answer (D): No, No, Yes

1. Placing the XML segment in the <inbound> section of the policy doesn't guarantee effective handling of incoming requests, as other factors such as policy logic and configuration also play a role.

2. There's no mention of a size restriction for incoming requests, so it's unclear whether exceeding 256 KB would trigger an error.

3. While the scenario doesn't explicitly state how API versions are prioritized, it does suggest that the policy may prioritize the higher version of the API if the incoming request is directed to a specific URL, indicating a possible "Yes" for this statement.

Feedback(if wrong):-

A) Yes, No, Yes

- Incorrect because the first statement lacks sufficient information to determine its accuracy, and the second statement's condition about request size isn't mentioned.

B) No, Yes, No

- Incorrect because the second statement lacks context from the scenario, and there's no information about API version prioritization.

C) Yes, Yes, No

- Incorrect because the second statement lacks context, and the third statement's condition about API version prioritization isn't confirmed in the scenario.

Skill Mapping:

Skills: Azure Developer Certification (AZ 204)

Subskill: Implement Azure security

Competency: Azure Security, Authentication, and authorization.

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

25. You are tasked with configuring Azure Key Vault to securely store sensitive information for an application. You need to write PowerShell commands to retrieve a secret from Azure Key Vault and set it as a secure string for use in your application. Which sequence of PowerShell commands should you use to accomplish this task?

A) Set-AzKeyVaultSecret -VaultName $vaultName -Name $secretName -SecretValue $secretvalue, ConvertTo-SecureString <secretValue> -AsPlainText -Force, Get-AzSubscription, Get-AzKeyVaultSecret -VaultName $vaultName

B) Get-AzKeyVaultSecret -VaultName $vaultName, ConvertTo-SecureString <secretValue> -AsPlainText -Force, Set-AzKeyVaultSecret -VaultName $vaultName -Name $secretName -SecretValue $secretvalue

C) Get-AzSubscription, Set-AzContext -SubscriptionId $subscriptionID, Get-AzStorageAccountKey -ResourceGroupName $resGroup -Name $storAcct, ConvertTo-SecureString <secretValue> -AsPlainText -Force, Set-AzKeyVaultSecret -VaultName $vaultName -Name $secretName -SecretValue $secretvalue, Get-AzKeyVaultSecret -VaultName $vaultName

D) ConvertTo-SecureString <secretValue> -AsPlainText -Force, Set-AzKeyVaultSecret -VaultName $vaultName -Name $secretName -SecretValue $secretvalue, Get-AzKeyVaultSecret -VaultName $vaultName

Answer: B

Feedback(if correct):-

B) Get-AzKeyVaultSecret -VaultName $vaultName, ConvertTo-SecureString <secretValue> -AsPlainText -Force, Set-AzKeyVaultSecret -VaultName $vaultName -Name $secretName -SecretValue $secretvalue

1. Get-AzKeyVaultSecret -VaultName $vaultName: Retrieves the secret from Azure Key Vault.

2. ConvertTo-SecureString <secretValue> -AsPlainText -Force: Converts the retrieved secret value to a secure string.

3. Set-AzKeyVaultSecret -VaultName $vaultName -Name $secretName -SecretValue $secretvalue: Sets the retrieved secret as a key vault secret.

Feedback(if wrong):-

A) Set-AzKeyVaultSecret -VaultName $vaultName -Name $secretName -SecretValue $secretvalue, ConvertTo-SecureString <secretValue> -AsPlainText -Force, Get-AzSubscription, Get-AzKeyVaultSecret -VaultName $vaultName

- Incorrect because it attempts to set the key vault secret before retrieving it from Azure Key Vault.

C) Get-AzSubscription, Set-AzContext -SubscriptionId $subscriptionID, Get-AzStorageAccountKey -ResourceGroupName $resGroup -Name $storAcct, ConvertTo-SecureString <secretValue> -AsPlainText -Force, Set-AzKeyVaultSecret -VaultName $vaultName -Name $secretName -SecretValue $secretvalue, Get-AzKeyVaultSecret -VaultName $vaultName

- Incorrect because it involves unnecessary commands related to Azure subscriptions and storage account keys, which are not relevant to the task of retrieving a secret from Azure Key Vault.

D) ConvertTo-SecureString <secretValue> -AsPlainText -Force, Set-AzKeyVaultSecret -VaultName $vaultName -Name $secretName -SecretValue $secretvalue, Get-AzKeyVaultSecret -VaultName $vaultName

- Incorrect because it attempts to set the key vault secret without retrieving it first, which is necessary for the subsequent commands to work properly.

Skill Mapping:

Skills: Azure Developer Certification (AZ 204)

Subskill: Implement Azure security

Competency: Azure Security, Authentication, and authorization.

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

26. You are responsible for enhancing the security of an internal website that employees use to access sensitive data. The website currently authenticates users using Azure Active Directory (AAD). To further strengthen

security, you need to implement multi-factor authentication (MFA) for the website. Which action should you take to achieve this security enhancement?

A) Configure the website to use Azure AD B2C.

B) Enable application proxy in Azure AD.

C) Create a new conditional access policy in Azure AD.

D) Upgrade to Azure AD Premium.

Answer: C

Feedback (if correct):

Option C is the correct answer. Creating a new conditional access policy in Azure AD allows administrators to enforce multi-factor authentication (MFA) under certain conditions, such as when accessing sensitive data or logging in from an untrusted location. Conditional Access policies are a powerful tool in Azure AD that help protect against potential threats by requiring additional verification steps, thereby enhancing the security of internal websites that authenticate through Azure Active Directory.

Feedback (if wrong):

Option A: Configuring the website to use Azure AD B2C is primarily aimed at external customer identity and access management scenarios, rather than internal applications authenticated through Azure AD. It's not the direct path to implementing MFA for an internal website.

Option B: Enabling application proxy in Azure AD is useful for providing secure remote access to internal applications but does not directly implement MFA. It's a feature more aligned with connectivity than with authentication security enhancements like MFA.

Option D: While upgrading to Azure AD Premium provides additional security features and customization options, including conditional access policies and advanced MFA configurations, it's not the only step required to implement MFA. The action that directly addresses the need for MFA implementation is creating a new conditional access policy.

Skill Mapping:

- Skills: Azure Developer Certification (AZ-204)

- Subskills: Implement Azure security

- Competencies: Azure Active Directory (Azure AD), Multifactor Authentication (MFA), Conditional Access Policies

- Difficulty Level: Intermediate

- Bloom's Taxonomy Level: Application

27. Your organization employs Azure Front Door Service to enhance web performance through Brotli compression for inbound content. However, you've encountered a problem where XML files, each around 9 MB, are not undergoing compression as anticipated. Your objective is to pinpoint the issue's cause, ensuring alignment with the Azure Developer (AZ-204) exam requirements. Assess the accuracy of the following statements to determine why Azure Front Door Service does not compress the inbound XML files.

Statements to Evaluate:

1. Azure Front Door Service recognizes and supports the MIME type of the XML files for compression.

2. To initiate compression for these XML files, it is necessary to purge all cached content from the edge nodes.

3. Azure Front Door Service is configured to support Brotli as a compression format for incoming files.

Select the Correct Option Based on the Statements:

A) True, False, False

B) False, False, True

C) False, False, False

D) True, True, True

Answer: B

Feedback(if correct):-

Option B (Correct): The inability to compress XML files likely stems from an unsupported MIME type by Azure Front Door Service (Statement 1 is False). Clearing cached content at edge nodes (Statement 2) is unrelated to activating compression for incoming files, thus also False. Azure Front Door does indeed support Brotli compression (Statement 3 is True), suggesting the service's configuration or the specific attributes of the XML files (such as MIME type) are at the heart of the issue.

Feedback(if wrong):-

B) Creating a nam Option A (Incorrect): Suggests that the MIME type is supported, which contradicts the observed behavior of non-compression for XML files, indicating a potential misunderstanding of how Azure Front Door handles MIME types for compression.

Option C (Incorrect): Indicates all statements are false, including the incorrect assertion that Brotli compression isn't supported by Azure Front Door, which contradicts its documented capabilities.

Option D (Incorrect): Implies all conditions are ideal for compression (including MIME type support and the necessity of clearing cache), which does not align with the observed issue of XML files not being compressed. This overlooks the specifics of compression support and the role of MIME types in the process.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Monitor, optimize, and troubleshoot Azure solutions

Competencies: Azure Networking and Web Services, Knowledge of Azure Front Door Service, focusing on its content delivery and optimization features, including dynamic content compression. Troubleshooting and Performance Optimization: Skills in diagnosing issues related to content delivery networks (CDN) and understanding of content compression mechanisms.

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

28. You're facing troubles with a line-of-business application hosted on Azure App Services. Occasionally, the application returns error codes suggesting an inconsistent state, possibly caused by race conditions during concurrent user interactions. Improvement suggestions include introducing Retry logic backed by a distributed cache.

Requirement:

The system must support the following:

1. Handling intermittent errors gracefully

2. Coordinating retry attempts across users

3. Keeping session data synchronized

Which Azure services should you combine to implement the suggested improvements?

A) Azure App Services + Azure Redis Cache

B) Azure Functions + Azure Cosmos DB

C) Azure SignalR Service + Azure Cache for Redis

D) Azure API Management + Azure Key Vault

Answer: A

Feedback(if correct):-

A) Azure App Services + Azure Redis Cache

1. Azure App Services is already hosting the application, which provides a robust environment for building, deploying, and scaling web apps. Continuing to use Azure App Services allows you to leverage its full suite of features, including integrated health checks, automatic scaling, and deployment slots for staging and production environments.

2. Azure Redis Cache is a high-performance, distributed cache service that can significantly improve the application's performance and scalability by storing frequently accessed data in memory. It effectively handles intermittent errors and ensures data consistency across user sessions by:

Handling intermittent errors gracefully: Azure Redis Cache can quickly retrieve cached data, reducing the load on the database and mitigating issues caused by database connection errors or timeouts. Implementing retry logic with Azure Redis Cache helps manage transient errors smoothly.

Coordinating retry attempts across users: By using a distributed cache, you can store and share status information about retry attempts or transient errors across user sessions. This coordination helps in avoiding redundant retries and managing the application's state more effectively.

Keeping session data synchronized: Azure Redis Cache supports session state caching, which ensures that user session data is kept consistent across multiple instances of the application. This is crucial for maintaining a consistent application state during concurrent interactions.

Feedback(if wrong):-

B) Azure Functions + Azure Cosmos DB: While Azure Functions provide a serverless compute service and Azure Cosmos DB offers a globally distributed database service with multi-model support, this combination focuses more on compute and database scalability rather than session management and error handling via caching and retry logic.

C) Azure SignalR Service + Azure Cache for Redis: Azure SignalR Service is primarily used for real-time web functionality. While Azure Cache for Redis could support the caching requirements, SignalR Service doesn't directly address the application's need for error handling and retry logic.

D) Azure API Management + Azure Key Vault: Azure API Management helps manage APIs, and Azure Key Vault secures application secrets. Neither service directly contributes to handling race conditions, coordinating retries, or caching session data as required by the scenario.

Choosing Azure App Services combined with Azure Redis Cache aligns best with the goals of enhancing application reliability and user experience by effectively managing intermittent errors and maintaining session consistency.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Monitor, optimize, and troubleshoot Azure solutions

Competencies: Azure Networking and Web Services, Knowledge of Azure Front Door Service, focusing on its content delivery and optimization features, including dynamic content compression. Troubleshooting and Performance Optimization: Skills in diagnosing issues related to content delivery networks (CDN) and understanding of content compression mechanisms.

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

29. You're experiencing issues with a distributed inventory management application for a retail chain. Sales happening at various locations should update store inventories in real-time. To optimize communication between the stores and the central system, you've decided to employ Azure Service Bus.

The system must support the following:

Store inventory updates are sent to the central system

Scalable processing of updates as new stores are added

Targeted directives from the central system to stores

In sequence, which three actions should you take to set up the Azure Service Bus for the inventory management system?

A) Create a single namespace, create a single topic, create a subscription for each store

B) Create a namespace for each store, create a topic for each inventory update, create a single subscription

C) Create a single namespace, create a topic for each store, create a single subscription

D) Create a single namespace, create a subscription for each inventory update, and create a single topic

Answer: A

Feedback(if correct):- A) Create a single namespace, create a single topic, create a subscription for each store

1. Create a single namespace: A namespace serves as a container for all Azure Service Bus resources within your application, simplifying management and scaling. By using a single namespace, you centralize the control over your messaging components, which is cost-effective and reduces complexity.

2. Create a single topic: A topic supports a publish-subscribe pattern, making it suitable for scenarios where messages sent by the central system (such as inventory updates) need to be broadcast to multiple subscribers. A single topic can handle messages for all stores, making the system scalable as new stores are added without requiring additional topics.

3. Create a subscription for each store: Subscriptions to a topic allow for messages to be consumed by specific subscribers. By creating a separate subscription for each store, the central system can send targeted directives to individual stores, ensuring that each store only receives messages relevant to its inventory. This setup supports the requirement for targeted communication and scalability.

Feedback(if wrong):-

B) Create a namespace for each store, create a topic for each inventory update, and create a single subscription: This approach introduces unnecessary complexity and management overhead. It fragments the messaging infrastructure unnecessarily, making it more difficult to scale and manage as new stores are added.

C) Create a single namespace, create a topic for each store, and create a single subscription: Creating a separate topic for each store complicates the architecture unnecessarily and doesn't take full advantage of the publish-subscribe model's scalability. This setup could also limit the system's ability to efficiently broadcast messages that are relevant to all stores.

D) Create a single namespace, create a subscription for each inventory update, and create a single topic: This option misunderstands the role of subscriptions and topics. Subscriptions are intended for consumers to receive messages from a topic, not for each inventory update, which would lead to a cluttered and unmanageable system.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Monitor, optimize, and troubleshoot Azure solutions

Competencies: Azure Networking and Web Services, Knowledge of Azure Front Door Service, focusing on its content delivery and optimization features, including dynamic content compression. Troubleshooting and Performance Optimization: Skills in diagnosing issues related to content delivery networks (CDN) and understanding of content compression mechanisms.

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

30. You're tasked with enhancing the security of a web application using Azure Web Application Firewall (WAF). The application is accessible through an Azure Application Gateway, which also serves other web applications. The specific web app you're working on is hosted at the address contoso.azurewebsites.net, and all its incoming traffic must be safeguarded with SSL encryption. You are responsible for configuring the Azure Application Gateway to meet this requirement for the app. Which two steps are essential to achieve this goal? Select two options that together form a complete solution.

A. Activate the "Use for App service" option within the HTTP settings of the Azure Application Gateway.

B. Transition the web app to operate within an Azure App Service Environment (ASE).

C. Integrate an SSL certificate for contoso.azurewebsites.net into the Azure Application Gateway.

D. Modify the "Override backend path" setting in the Azure Application Gateway's HTTP settings to point to contoso22.azurewebsites.net.

Answer: A, C

Feedback(if correct):-

Option A: Turning on the "Use for App service" feature in the HTTP settings is crucial when the gateway is directing traffic to Azure App Services. It ensures the gateway is finely tuned to manage traffic for App Services, aiding in the secure and efficient handling of SSL-encrypted communications.

Option C: Implementing an SSL certificate specifically for contoso.azurewebsites.net on the Azure Application Gateway is critical for establishing a secure SSL connection. This step is fundamental for encrypting traffic to the web app, even though the original explanations suggest a slightly different approach.

Feedback(if wrong):-

B. Transition the web app to operate within an Azure App Service Environment (ASE).

While moving a web app to an Azure App Service Environment offers enhanced isolation and can contribute to overall security, it doesn't directly influence the configuration of SSL encryption via Azure Application Gateway. SSL configuration involves specific settings on the gateway itself, such as enabling SSL termination or end-to-end SSL encryption, rather than the hosting environment of the web application.

D. Modify the "Override backend path" setting in the Azure Application Gateway's HTTP settings to point to contoso22.azurewebsites.net.

Adjusting the "Override backend path" is a routing configuration that alters the request path as it's forwarded to the backend pool. While useful in certain routing scenarios, it does not contribute to the SSL configuration necessary to secure the web application. SSL security is achieved through the correct setup of SSL termination at the gateway or by passing encrypted traffic to the backend, neither of which is addressed by simply overriding the backend path.

The essential steps for ensuring SSL encryption for the web app involve proper configuration within the Azure Application Gateway's settings to handle SSL traffic correctly and the inclusion of the appropriate SSL certificate. These steps ensure that all incoming traffic is securely encrypted, fulfilling the security requirement without the need for environmental changes or routing adjustments unrelated to SSL.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Monitor, optimize, and troubleshoot Azure solutions

Competencies: Azure Networking and Web Services, Knowledge of Azure Front Door Service, focusing on its content delivery and optimization features, including dynamic content compression. Troubleshooting and Performance Optimization: Skills in diagnosing issues related to content delivery networks (CDN) and understanding of content compression mechanisms.

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

31. You are developing a web application for a client. The application is expected to handle a surge in traffic once it is deployed. To ensure high availability and responsiveness while keeping costs low, you decide to host the application on Azure. Which deployment strategy should you choose?

A) Deploy the website to an Azure App Service using the Basic service tier and manually adjust the instance count based on traffic fluctuations.

B) Utilize Azure Kubernetes Service (AKS) to deploy the website in containerized form, allowing for dynamic scaling based on resource usage.

C) Deploy the website to an Azure App Service using the Standard service tier and configure auto-scaling based on CPU load.

D) Deploy the website to an Azure virtual machine and manually adjust the instance size based on traffic fluctuations.

Answer: C

Feedback(if correct):-

The best deployment strategy to ensure high availability and responsiveness while keeping costs low is:

C) Deploy the website to an Azure App Service using the Standard service tier and configure auto-scaling based on CPU load.

Explanation:

Azure App Service in the Standard service tier offers auto-scaling capabilities, which allows the service to automatically increase or decrease the number of instances running based on predefined rules or schedules. Configuring auto-scaling based on CPU load ensures that the application can handle surges in traffic by automatically adding more resources when needed and scaling down during periods of low usage to keep costs low.

Feedback(if wrong):-

Option A suggests using the Basic service tier and manually adjusting the instance count. While this may offer some level of scalability, it requires manual intervention and may not respond quickly enough to sudden traffic surges, potentially leading to downtime or degraded performance.

Option B involves using Azure Kubernetes Service (AKS) for deploying containerized applications. AKS supports dynamic scaling and high availability but might introduce additional complexity and cost considerations for container orchestration, especially if the team is not already familiar with Kubernetes.

Option D suggests deploying the website to an Azure virtual machine and manually adjusting the instance size. This approach offers the least automation in terms of scaling and can be the most time-consuming and least cost-effective, as it requires manual monitoring and intervention to adjust resources in response to traffic changes.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Monitor, optimize, and troubleshoot Azure solutions

Competencies: Azure Networking and Web Services, Knowledge of Azure Front Door Service, focusing on its content delivery and optimization features, including dynamic content compression. Troubleshooting and Performance Optimization: Skills in diagnosing issues related to content delivery networks (CDN) and understanding of content compression mechanisms.

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

32. Your company is launching a new e-commerce website and has chosen Azure as the hosting platform. The website is anticipated to attract a large number of visitors, particularly during promotional events. How should you deploy the website to ensure it remains responsive under varying traffic loads while keeping costs optimized?

A) Deploy the website to an Azure App Service using the Free service tier and manually scale up the resources during peak traffic.

B) Use Azure Functions to host the website's backend logic and leverage the serverless architecture for automatic scaling based on demand.

C) Deploy the website to an Azure App Service using the Standard service tier and configure auto-scaling based on HTTP queue length.

D) Utilize Azure Container Instances (ACI) to host the website and configure scaling policies based on memory usage.

Answer: C

Feedback(if correct):-

C) Deploy the website to an Azure App Service using the Standard service tier and configure auto-scaling based on HTTP queue length.

Azure App Service in the Standard service tier offers auto-scaling capabilities, which is crucial for handling variable traffic loads efficiently. Configuring auto-scaling based on HTTP queue length is particularly effective for web applications, as it directly relates to incoming web traffic. This approach allows the service to automatically add more instances as the number of incoming requests increases, ensuring that the website remains responsive during peak traffic times. Once the traffic subsides, the service can scale down to reduce costs.

Deploying the website on Azure App Service in the Standard tier and setting up auto-scaling based on HTTP queue length (Option C) strikes the best balance between responsiveness, scalability, and cost efficiency for a high-traffic e-commerce website.

Feedback(if wrong):-

Option A involves using the Free service tier, which is not suitable for production environments, especially for an e-commerce website expecting high traffic. The Free tier has significant limitations in terms of resources and features, such as CPU time, which could lead to poor performance and unavailability during critical times.

Option B suggests using Azure Functions for the website's backend logic, benefiting from serverless architecture's dynamic scaling. While Azure Functions is an excellent choice for running event-driven applications, hosting an entire e-commerce website's backend might not be the most efficient or cost-effective approach due to potential cold starts and the complexity of managing a fully serverless architecture for such use cases.

Option D mentions utilizing Azure Container Instances (ACI) with scaling policies based on memory usage. ACI provides a great way to run containers on Azure without managing the underlying infrastructure. However, it might not be the most cost-effective solution for long-running, high-traffic applications compared to Azure App Service, which offers more comprehensive features for web hosting and easier management of scaling policies.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Monitor, optimize, and troubleshoot Azure solutions

Competencies: Azure Networking and Web Services, Knowledge of Azure Front Door Service, focusing on its content delivery and optimization features, including dynamic content compression. Troubleshooting and Performance Optimization: Skills in diagnosing issues related to content delivery networks (CDN) and understanding of content compression mechanisms.

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

33. Scenario: Your company, Global Media Solutions, specializes in digital content and has a global customer base. The company's website features high-resolution images, including a frequently accessed company logo. To enhance the user experience by reducing load times and ensuring the high availability of content worldwide, Global Media Solutions decides to utilize Azure Content Delivery Network (CDN). The goal is to understand how Azure CDN serves content to users and how it optimizes content delivery.

Question 1: When a user first requests a high-resolution image from the Global Media Solutions website via the Azure CDN URL, what sequence of events correctly describes the process?

A) The CDN directly serves the image from the origin server without utilizing any POP locations.

B) The user's request is immediately served by the nearest edge server, regardless of whether the image is cached.

C) The DNS routes the user's request to the best-performing POP location, where the image is served from the cache if available.

D) The DNS routes the user's request to the best-performing POP location. If the image isn't in the cache, the request is forwarded to the origin server.

Answer: D

Feedback(if correct):-

D) The DNS routes the user's request to the best-performing POP location. If the image isn't in the cache, the request is forwarded to the origin server.

- This is the correct option because it accurately describes the entire process of handling a request within Azure CDN. It includes routing the request based on performance to the closest or most suitable POP, checking for cached content, and the subsequent action if the content is not found in the cache (fetching from the origin server and caching it for future requests). This complete process ensures efficient content delivery and optimization, embodying Azure CDN's operational model.

Feedback(if wrong):-

A) The CDN directly serves the image from the origin server without utilizing any POP locations.

- This option is incorrect because one of the primary functions of Azure CDN is to serve content from the closest Point of Presence (POP) to the user, not directly from the origin server. This approach reduces latency by serving cached content from a geographically closer edge server.

B) The user's request is immediately served by the nearest edge server, regardless of whether the image is cached.

- This statement is misleading. While the nearest edge server in the POP location does handle the user's request, it only serves the content directly if it is already cached. If the content is not in the cache, the edge server must retrieve it from the origin server before serving it to the user, which is not implied in this option.

C) The DNS routes the user's request to the best-performing POP location, where the image is served from the cache if available.

- While this option might seem correct as it involves routing to the best-performing POP and serving cached content, it lacks the complete process that happens when the cache does not have the content. It doesn't address what occurs if the image isn't available in the cache, leaving out a critical part of how Azure CDN operates with cache misses.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Monitor, optimize, and troubleshoot Azure solutions

Competencies: Azure Networking and Web Services, Knowledge of Azure Front Door Service, focusing on its content delivery and optimization features, including dynamic content compression. Troubleshooting and Performance Optimization: Skills in diagnosing issues related to content delivery networks (CDN) and understanding of content compression mechanisms.

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

34. Scenario: Your company, Global Media Solutions, specializes in digital content and has a global customer base. The company's website features high-resolution images, including a frequently accessed company logo. To enhance the user experience by reducing load times and ensuring the high availability of content worldwide, Global Media Solutions decides to utilize Azure Content Delivery Network (CDN). The goal is to understand how Azure CDN serves content to users and how it optimizes content delivery.

Question 2: In the scenario where an edge server in the chosen POP doesn't have the requested image in its cache, what is the correct order of actions taken by Azure CDN to serve the content to the user?

A) The POP directly serves a default image to minimize latency.

B) The request bypasses CDN and is served directly from the origin server.

C) The POP requests the image from the origin server, caches it, and then serves it to the user.

D) Azure CDN redirects the user to a different POP location that might have the image cached.

Answer: C

Feedback(if correct):-

For Question 2 on Azure CDN cache miss and content retrieval, the correct process when an edge server in the chosen POP doesn't have the requested image in its cache is:

C) The POP requests the image from the origin server, caches it, and then serves it to the user.

- This is the correct process for handling a cache miss within Azure CDN. When the requested content is not available in the cache of the nearest or chosen POP's edge server, the CDN retrieves the content from the origin server. This ensures that the content is delivered with minimal delay for the initial request and improves response times for subsequent requests by caching the content at the edge server. This approach optimizes content delivery, reduces load on the origin server, and enhances the user experience by leveraging the CDN's global network of POPs.

Feedback(if wrong):-

A) The POP directly serves a default image to minimize latency.

- This option is incorrect because Azure CDN's typical behavior on a cache miss is to retrieve the content from the origin server, not to serve a default image. Serving a default image instead of the requested content would not meet user expectations or the application's content delivery needs.

B) The request bypasses CDN and is served directly from the origin server.

- While the result of serving content from the origin server is part of the cache miss process, this option is misleading because it suggests that Azure CDN is entirely bypassed, which is not the case. The CDN still plays a crucial role in the process by fetching the content from the origin server, caching it for future requests, and then serving it to the user.

D) Azure CDN redirects the user to a different POP location that might have the image cached.

- This option is incorrect because Azure CDN does not redirect user requests to different POP locations based on cache availability. The CDN's design is to serve the user from the geographically closest or best-performing POP, and if a cache miss occurs, the content is fetched from the origin server, cached at the edge server in the initial POP, and served to the user to ensure future requests can be served more quickly.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Monitor, optimize, and troubleshoot Azure solutions

Competencies: Azure Networking and Web Services, Knowledge of Azure Front Door Service, focusing on its content delivery and optimization features, including dynamic content compression. Troubleshooting and Performance Optimization: Skills in diagnosing issues related to content delivery networks (CDN) and understanding of content compression mechanisms.

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

35. You are architecting a solution for a traffic monitoring system that analyzes traffic across six highways. The system leverages Azure Event Hub to ingest real-time data from traffic sensors for time-series analysis. Each of the six highways has a dedicated stream of data. To ensure optimal data processing and minimize latency, how should the Azure Event Hub be configured to handle the data streams from the six highways?

A) Set the number of partitions to 3, focusing on reducing the overall cost.

B) Set the number of partitions to 6, aligning each highway with a dedicated partition for parallel processing.

C) Set the number of partitions to 12, doubling the highways' count to enhance data throughput.

D) Set the number of partitions to 4, balancing between throughput and system complexity.

Answer: B

Feedback(if correct):-

Selecting to set the number of partitions to 6 is correct because it matches the number of highways being monitored. This configuration allows each highway's data to be processed independently and in parallel, which is crucial for minimizing latency and maximizing throughput in the traffic monitoring system. It demonstrates an understanding of how partitions in Azure Event Hub facilitate data organization and parallel processing, directly impacting the efficiency and performance of real-time analytics solutions.

Feedback(if wrong):-

Option A (3 partitions): This choice would limit the system's ability to process data from each highway independently, potentially causing bottlenecks and increased latency as multiple highways' data would need to be processed through fewer partitions.

Option C (12 partitions): While more partitions can increase throughput, setting the partition count to double the number of highways does not directly contribute to efficiency and could complicate data processing without a clear benefit.

Option D (4 partitions): Similar to option A, having fewer partitions than highways would restrict parallel processing capabilities, negatively affecting data throughput and system responsiveness.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Monitor, optimize, and troubleshoot Azure solutions

Competencies: Azure Networking and Web Services

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

36. You are architecting a solution for a traffic monitoring system that analyzes traffic across six highways. The system leverages Azure Event Hub to ingest real-time data from traffic sensors for time-series analysis. Each of the six highways has a dedicated stream of data. , where each highway's data is ingested into Azure Event Hub for analysis, which partition key strategy should be implemented to ensure efficient data routing and analysis?

A) Use the "Department" as the partition key to segregate data by consuming department.

B) Use the "Highway" as the partition key to ensure data from each highway is grouped together.

C) Use the "Timestamp" as the partition key to organize data chronologically across all highways.

D) Use the "VM name" as the partition key to distribute data based on the virtual machine processing it.

Answer: B

Feedback(if correct):-

Using "Highway" as the partition key is the optimal strategy for ensuring efficient data routing and analysis in the traffic monitoring system. This approach ensures that all data for a specific highway is kept together, allowing for streamlined processing and analysis. It leverages Azure Event Hub's partitioning capabilities to maintain the order of events per highway, which is essential for accurate time-series analysis and real-time monitoring. This choice reflects a solid understanding of partition key usage to enhance data organization and processing efficiency in distributed systems.

Feedback(if wrong):-

Option A ("Department"): This would not effectively segregate data in a manner that benefits the time-series analysis of each highway, as the focus of the monitoring system is on highways, not departments.

Option C ("Timestamp"): While organizing data chronologically might seem logical, using the timestamp as a partition key does not support the requirement to analyze data per highway, as it would scatter events from the same highway across different partitions based on time.

Option D ("VM name"): This choice misaligns the intention to efficiently route and analyze traffic data, as VM names have no logical relation to the data's content or the system's analytical requirements, leading to ineffective data partitioning and processing.

skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Monitor, optimize, and troubleshoot Azure solutions

Competencies: Azure Networking and Web Services

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension, Application

37. You are responsible for managing several existing Logic Apps and regularly making changes to their definitions, adding new logic, and optimizing them. Which tools should you use for each functionality? Choose the best option from the answer choices provided.

A) Logic Apps Designer - Edit B2B workflows

B) Code View Editor - Edit definitions in JSON

C) Enterprise Integration Pack - Visually and functionality editing

D) Power Automate - Copy multiple selections from Choice columns

Answer: A, B

Feedback(if correct):-

To manage and optimize existing Logic Apps, you can use the following tools for specific functionalities:

A) Logic Apps Designer: This tool allows you to visually design and edit Logic Apps, making it suitable for editing B2B workflows.

B) Code View Editor: This tool enables you to view and edit the definitions of Logic Apps in JSON format, making it suitable for editing definitions in JSON.

Feedback(if wrong):-

C) Enterprise Integration Pack: This tool is not directly related to editing and optimizing Logic Apps. It is used for creating business-to-business (B2B) workflows with an integration account that has partners and agreements.

D) Power Automate: While Power Automate is a powerful tool for automating workflows, it is not specifically mentioned in the context of managing and optimizing Logic Apps.

Skill Mapping :

Skill: Azure Developer Certification (AZ-204)

Subskill: Connect to and consume Azure services and third-party services

Competency: Azure Logic Apps, Technical Editing and Definition Optimization

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

38. You are developing an application for a travel and bookings management service that will integrate restaurant bookings. The application needs to allow users to search for restaurants by various criteria such as name, description, location, cuisine, and more. Which Azure service should you integrate into your solution to meet these requirements?

A) Azure Functions

B) Azure Cognitive Search

C) Azure App Service

D) Azure Cosmos DB

Answer: B

Feedback(if correct):

Azure Cognitive Search is the appropriate Azure service for enabling search functionalities in the application. It allows users to search for restaurants based on various criteria such as name, description, location, cuisine, and more. Azure Cognitive Search provides powerful indexing and querying capabilities, making it ideal for implementing search functionality in applications.

Feedback(if wrong):

A) Azure Functions: Azure Functions is a serverless compute service used for running event-driven code in response to various triggers. While it can be used to perform specific tasks or execute business logic, it is not designed to enable search functionality.

C) Azure App Service: Azure App Service is a fully managed platform for building, deploying, and scaling web applications and APIs. While it provides capabilities for hosting and managing web applications, it does not offer built-in search functionalities.

D) Azure Cosmos DB: Azure Cosmos DB is a globally distributed, multi-model database service for building highly responsive and scalable applications. While it is suitable for storing and querying data, including data related to restaurant bookings, it does not provide native search capabilities like Azure Cognitive Search.

Skill Mapping :

Skill: Azure Developer Certification (AZ-204)

Subskill: Connect to and consume Azure services and third-party services

Competency: Azure Logic Apps, Technical Editing and Definition Optimization

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

39. You're tasked with integrating an Azure API Management service instance with a public-facing news API, which is implemented as a RESTful service using an OpenAPI specification. What Azure PowerShell command should you execute to ensure seamless access to the news API through the API Management service?

A. Create-AzureRmApiManagement -ResourceGroupName $ResourceGroup -Name $Name -Location $Location -Organization $Org -AdminEmail $AdminEmail

B. Add-AzureRmApiManagementApi -Context $ApiMgmtContext -SpecificationFormat "Swagger" -SpecificationPath $SwaggerPath -Path $Path

C. Register-AzureRmResourceProvider -ProviderNamespace 'Microsoft.ApiManagement' -Force

D. New-AzureRmApiManagementBackendProxy -Url $ApiUrl

Answer: D

Feedback (if correct):

Option D (New-AzureRmApiManagementBackendProxy) is the correct answer. This PowerShell command creates a new Backend Proxy Object, which is essential for accessing the news API through the Azure API Management service instance.

Feedback (if wrong):

Option A (Import-AzureRmApiManagementApi) is incorrect because it is used for importing APIs into Azure API Management but does not specifically address setting up a backend proxy for accessing a news API.

Option B (New-AzureRmApiManagementBackend) is incorrect because it is used for creating a new backend in Azure API Management but does not specifically relate to setting up a backend proxy for accessing a news API.

Option C (New-AzureRmApiManagement) is incorrect because it is used for creating a new instance of Azure API Management but does not address setting up a backend proxy for accessing a news API.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Connect to and consume Azure services and third-party services

Competency: Develop an App Service Logic App, Create a Logic App, Create a custom connector for Logic Apps

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

40. You are tasked with connecting to a globally distributed No-SQL database using the .NET API. To configure and execute requests in the database, you need to select the appropriate code segment. Which option should you choose?

A) Initialize a new Container object with the EndpointUri and PrimaryKey.

B) Initialize a new Database object with the Endpoint and PrimaryKey.

C) Initialize a new CosmosClient object with the EndpointUri and PrimaryKey.

D) Initialize a new TableClient object with the Endpoint and PrimaryKey.

Answer: C

Feedback(if correct): Option C is the correct answer. The CosmosClient class in the .NET API is specifically designed for connecting to and interacting with Azure Cosmos DB, which is a globally distributed No-SQL database service provided by Azure.

Feedback(if wrong):

Option A is incorrect because there is no Container class in the .NET API for connecting to a No-SQL globally distributed database like Cosmos DB.

Option B is incorrect because there is no Database class in the .NET API for connecting to a No-SQL globally distributed database like Cosmos DB.

Option D is incorrect because there is no TableClient class in the .NET API for connecting to a No-SQL globally distributed database like Cosmos DB.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Connect to and consume Azure services and third-party services

Competency: Develop an App Service Logic App, Create a Logic App, Create a custom connector for Logic Apps

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

41. You are developing a solution for a hospital that requires specific consistency levels for accessing patient data and billing information in a Cosmos DB NoSQL database. You need to determine the appropriate consistency levels to ensure data accuracy and availability for each scenario. Select the correct consistency level for each scenario:

Scenario 1: Retrieve the most recent patient status details, ensuring that all users see the latest updates regardless of their location.

A) Strong

B) Bounded Staleness

C) Consistent Prefix

D) Eventual

Answer: A

Feedback(if correct):

The correct answer is A) Strong. Strong consistency guarantees that all reads will return the most recent committed version of an item. This ensures that users across different locations will see the latest updates to the patient status details.

Feedback(if wrong):

- B) Bounded Staleness: Bounded staleness ensures that reads will be within a certain version or time interval behind the latest update, but it does not guarantee the most recent data for all users.

- C) Consistent Prefix: Consistent prefix consistency level ensures that reads honor a consistent ordering of operations, but it may not guarantee the most recent data for all users.

- D) Eventual: Eventual consistency does not provide any ordering guarantee for reads and may result in users seeing stale data.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Connect to and consume Azure services and third-party services

Competency: Azure Cosmos DB

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

42. You are developing a solution for a hospital that requires specific consistency levels for accessing patient data and billing information in a Cosmos DB NoSQL database. You need to determine the appropriate consistency levels to ensure data accuracy and availability for each scenario. Select the correct consistency level for each scenario:

Scenario 2: Retrieve health monitoring data that is no less than one version behind the latest update.

A) Strong

B) Bounded Staleness

C) Consistent Prefix

D) Eventual

Answer: B

Feedback(if correct):

The correct answer is B) Bounded Staleness. The bounded staleness consistency level ensures that reads will be no more than a certain number of versions behind the latest update. This guarantees that the health monitoring data retrieved will be at most one version behind the latest update.

Feedback(if wrong):

- A) Strong: Strong consistency guarantees the most recent committed version of an item, which may not satisfy the requirement of being one version behind.

- C) Consistent Prefix: Consistent prefix consistency level ensures that reads honor a consistent ordering of operations, but it may not guarantee the health monitoring data to be one version behind.

- D) Eventual: Eventual consistency does not provide any ordering guarantee for reads and may result in users seeing stale data.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Connect to and consume Azure services and third-party services

Competency: Azure Cosmos DB

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

43. You are developing a solution for a hospital that requires specific consistency levels for accessing patient data and billing information in a Cosmos DB NoSQL database. You need to determine the appropriate consistency levels to ensure data accuracy and availability for each scenario. Select the correct consistency level for each scenario:

Scenario 3: Retrieve patient billing data after discharge, ensuring accurate final charges.

A) Strong

B) Bounded Staleness

C) Consistent Prefix

D) Eventual

Answer: C

Feedback(if correct):

The correct answer is C) Consistent Prefix. Consistent prefix consistency level ensures that reads honor a consistent ordering of operations, which is crucial for retrieving accurate billing data after discharge. This guarantees that the final charges are calculated based on the correct sequence of operations.

Feedback(if wrong):

- A) Strong: Strong consistency guarantees the most recent committed version of an item, but it may not be necessary for retrieving accurate billing data after discharge.

- B) Bounded Staleness: Bounded staleness ensures reads within a certain version or time interval behind the latest update, but it may not be suitable for ensuring accurate final charges.

- D) Eventual: Eventual consistency does not provide any ordering guarantee for reads and may result in users seeing stale data, which is not ideal for retrieving accurate billing data after discharge.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Connect to and consume Azure services and third-party services

Competency: Azure Cosmos DB

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

44. You are tasked with implementing Application Insights instrumentation capabilities in your mobile app using the Azure Mobile Apps SDK to analyze user interactions effectively. Which three data values should you capture to enable the Usage Analytics feature of Application Insights?

A) Page Views

B) User Interactions

C) Device Type

D) Custom Events

Answer: A, B, D

Feedback (if correct):

By tracking Page Views, User Interactions, and Custom Events, you can comprehensively analyze user interactions in your mobile app using the Usage Analytics feature of Application Insights.

Page Views represent the screens viewed by users, helping you understand engagement levels and popular pages. User Interactions refer to various user actions, such as button clicks, swipes, or gestures, which shed light on user interaction patterns. Lastly, Custom Events enable you to record specific events or milestones in the app uniquely tailored to your business goals or user journey.

To implement Usage Analytics for a mobile app, you need to capture the following three data values:

User Id: This allows you to track individual users and their behavior within the app.

Session Id: This allows you to track individual sessions and their duration.

Events: These are custom events that you can define to track specific actions or interactions within the app.

Here's an example of how you can capture these data values using the Azure Mobile Apps SDK:

python

Edit

Download

Copy code

```
from azure.applicationinsights import TelemetryClient

from azure.applicationinsights.channel import TelemetryChannel

from azure.applicationinsights.models import EventTelemetry, User, Session


# Initialize the TelemetryClient

telemetry_client = TelemetryClient(instrumentation_key="<your_instrumentation_key>")


# Create a new User object

user = User(id="<user_id>")


# Create a new Session object

session = Session(id="<session_id>")


# Create a new EventTelemetry object

event = EventTelemetry(name="<event_name>")


# Set the User and Session properties for the EventTelemetry object

event.properties["user"] = user

event.properties["session"] = session


# Send the EventTelemetry object to the Application Insights service

telemetry_client.track_event(event)
```

In this example, the User and Session objects are created with the required data values. The EventTelemetry object is created with the custom event name. The User and Session objects are set as properties for the EventTelemetry object. Finally, the EventTelemetry object is sent to the Application Insights service.

By capturing these data values, you can effectively implement Usage Analytics for your mobile app and gain valuable insights into user behavior and interactions.

Feedback (if wrong):

Incorrect: Option C (Session Id) is not directly related to the data required for Usage Analytics in Application Insights.

Based on the AZ-204 certification for Azure Developer, here is a skill mapping for the provided question:

Skill mapping:

Skills: Azure Developer Certification (AZ-204)

Subskill: Connect to and consume Azure services and third-party services

Competency: Azure Services Integration, Application Integration

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

45. You've been assigned the task of fixing a testing error in your Azure App Service Web App by updating the APIs. Which Azure CLI command should you use to enable Cross-Origin Resource Sharing (CORS) specifically for this web app?

A) az webapp config cors add --allowed-origins http://test.swap1websiteapp.com

B) az webapp cors add --allowed-origins http://test.swap1websiteapp.com

C) az webapp add cors --allowed-origins http://test.swap1websiteapp.com

D) az webapp config allowed-origins add --cors http://test.swap1websiteapp.com

Answer: A

Feedback(if correct):-

This command correctly adds allowed origins for CORS configuration to the Azure App Service Web App.

Feedback(if wrong):-

B) Incorrect. This command is missing the "config" keyword, which is necessary for configuring CORS settings.

C) Incorrect. The "add" keyword should come before "cors" to specify that a new CORS rule is being added.

D) Incorrect. The "config" keyword is used to specify configuration settings, but it should be followed by "cors add" to add a new CORS rule.

Skill mapping:

Skills: Azure Developer Certification (AZ 204)

Subskill: Connect to and consume Azure services and third-party services

Competencies: Azure Services Integration, Azure Logic Apps, Azure Service Bus, Azure Event Grid, Integration with third-party services

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

46. Your team is building a platform for an online art gallery where artists can submit their work. Once artwork is uploaded via a web portal, it is stored in Azure Blob Storage, using a General-purpose V2 storage account. Each submitted piece must undergo an automated review process to verify adherence to gallery standards, which should commence no more than 45 seconds after the artwork is uploaded.

Solution: Upgrade the storage account to use the Premium performance tier to ensure the processing starts as required.

Does the solution meet the goal?

A) Yes

B) No

Answer: B

Feedback(if correct):-

Switching to a Premium performance tier storage account focuses on enhancing access speed to stored data rather than initiating a processing workflow for new uploads. To meet the requirement of starting the automated review process within 45 seconds, leveraging an Azure Function that triggers the event of a new blob creation is more appropriate. This approach directly responds to the event of an artwork upload, initiating the review process promptly. General-purpose V2 accounts already support event-driven integration, making this solution feasible without needing a performance tier upgrade.

Feedback(if wrong):-

Selecting "Yes" suggests a misconception that upgrading the storage account to a Premium performance tier directly influences the initiation of automated processes, such as the review of uploaded artwork. The Premium performance tier in Azure Blob Storage is designed to provide faster access to data, lower latency, and higher transaction rates, which benefits scenarios requiring intensive read/write operations or where storage performance is critical.

However, the initiation of an automated review process following an upload is not inherently tied to the storage performance tier. Instead, it depends on the ability to trigger processes based on events, such as the arrival of new blobs (artwork files) in the storage account. This functionality is more accurately achieved through the integration of Azure services like Azure Functions, which can listen for Blob storage events and trigger the necessary review workflows automatically and within the specified time frame.

Therefore, while upgrading to a Premium performance tier may offer benefits in terms of data access and performance, it does not address the primary requirement of ensuring that the review process starts within 45 seconds of the artwork upload. A solution focusing on event-driven architecture, utilizing Azure Functions or similar services for real-time processing, would be necessary to meet the goal effectively.

Skill mapping:

Skills: Azure Developer Certification (AZ 204)

Subskill: Connect to and consume Azure services and third-party services

Competencies: Azure Services Integration, Azure Logic Apps, Azure Service Bus, Azure Event Grid, Integration with third-party services

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

47. You are developing a cloud-based document management system for a legal firm. The system allows users to upload documents through a web interface, which are then stored in Azure Blob Storage. The storage account used is of type General-purpose V2. Whenever a document is uploaded, it needs to be converted into a secure PDF format to ensure consistency across all viewing platforms. This conversion process must be initiated within 30 seconds of upload. Implement an Azure Function that triggers blob creation events within the Blob Storage to start the document conversion process. Does the solution meet the goal?

A) Yes

B) No

Answer: A

Feedback(if correct):-

Azure Functions can be triggered by Blob storage events, such as the uploading of a new document. This makes it an ideal solution for starting the conversion process almost immediately after the document upload, without the need to convert the storage account type. The General-purpose V2 storage account type supports integration with Azure Functions through Azure Event Grid, facilitating real-time processing of blob storage events. This setup adheres to the requirement of initiating the document conversion within 30 seconds.

Feedback(if wrong):-

The proposed solutions correctly leverage Azure's capabilities for event-driven processing, negating the need for changes like account type conversion or performance tier upgrades. In both scenarios, the essential requirement is to initiate a specific process (image conversion for mobile-friendly versions in the first scenario, and automated review in the second) swiftly after an upload to Azure Blob Storage. This need is best addressed by utilizing Azure Functions triggered by blob creation events, a feature fully supported by General-purpose V2 storage accounts. Upgrading to a different storage account type or performance tier does not inherently improve the system's ability to start processing uploads within the specified time frames. Instead, the integration of Azure Functions with Event Grid for real-time event handling provides a direct and efficient solution to meet the stated goals. The key is to focus on the event-driven architecture facilitated by Azure's ecosystem, ensuring processes begin promptly post-upload, aligning with the requirements of timely initiation of image conversion and automated review processes.

skill mapping:

Skills: Azure Developer Certification (AZ 204)

Subskills: - Connect to and consume Azure services and third-party services

Competencies: Azure Compute Solutions, Azure Functions, Azure Storage, Blob storage, Azure Services Integration, Azure Event Grid

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

48. Culinary Compass, an innovative service that connects diners with culinary experiences worldwide, is integrating dining venue listings into their search platform using Azure Cognitive Search. You are tasked with employing the Azure Search .NET SDK to import venue data into the search service efficiently.

You outline a plan for incorporating the dining venue data into Azure Cognitive Search as follows:

1. Create a `SearchServiceClient` to facilitate interaction with the Azure Cognitive Search index.

2. Assemble the venue data into a `DataSource` object, ensuring it encapsulates all necessary information.

3. Configure a `DataImport` object, setting its source to the previously defined `DataSource`.

4. Perform a call to `SearchServiceClient.SubmitDataImport`, providing the `DataImport` object to initiate the import process.

Does this strategy accurately execute the importation of dining venue data into the Azure Cognitive Search service?

A) Yes

B) No

Answer: B

Feedback(if correct): The correct answer is B) No. The outlined plan is incorrect because it suggests using a `DataImport` object for importing data into Azure Cognitive Search, whereas the correct approach involves using an `Indexer` object. The `Indexer` object is configured to extract data from a specific data source and push it into the search index.

This approach contains several misconceptions about importing data into Azure Cognitive Search using the .NET SDK:

- The `SearchServiceClient` does not exist within the Azure Search .NET SDK; the correct client for interacting with the index is `SearchIndexClient` or `SearchClient` in newer versions.

- The process described involving `DataSource`, `DataImport`, and a `SubmitDataImport` method does not accurately reflect the SDK's capabilities or methods for data importation. Instead, data should be prepared in documents and uploaded using `IndexDocumentsBatch` through the `SearchClient.IndexDocuments` method (or `SearchIndexClient.Documents.Index` in older SDK versions).

- The steps outlined do not follow the standard procedure for indexing documents directly within Azure Cognitive Search, showing a fundamental misunderstanding of the SDK's functionality for data import.

Feedback(if wrong):

- Option A) Yes: This is incorrect because the outlined plan does not accurately execute the importation process. It suggests using a `DataImport` object instead of an `Indexer` object.

skill mapping:

Skills: Azure Developer Certification (AZ 204)

Subskills: - Connect to and consume Azure services and third-party services

Competencies: Azure Compute Solutions, Azure Functions, Azure Storage, Blob storage, Azure Services Integration, Azure Event Grid

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

49. Global Getaways, a leading travel agency specializing in package tours, is diversifying into offering online restaurant reservations. To facilitate this new feature, they intend to integrate restaurant listings into their digital platform using Azure Cognitive Search. Your task involves using the Azure Search .NET SDK to efficiently upload the restaurant data into the service.

Task:

To integrate the restaurant listings, you devise the following strategy using the Azure Search .NET SDK:

1. Establish a connection to the search index by initializing a `SearchClient` object.

2. Compile the restaurant listings into a collection of documents ready for indexing.

3. Utilize the `IndexDocumentsBatch` to group the documents prepared for upload.

4. Invoke the `SearchClient.IndexDocuments` method, passing in the `IndexDocumentsBatch` to import the restaurant data.

Given this approach, will the restaurant data be successfully imported into the Azure Cognitive Search service?

A) Yes

B) No

Answer: A

Feedback(if correct):

The correct answer is A) Yes.

The outlined strategy accurately follows the correct steps for uploading restaurant data into Azure Cognitive Search using the Azure Search .NET SDK. It establishes a connection to the search index, compiles the restaurant listings into documents, groups them into an `IndexDocumentsBatch`, and then invokes the `SearchClient.IndexDocuments` method to import the data. This approach aligns with best practices for data ingestion into Azure Cognitive Search.

Feedback(if wrong):

Option B) No: This is incorrect. The plan provided outlines the correct steps for uploading restaurant data into Azure Cognitive Search using the Azure Search .NET SDK. Therefore, the restaurant data will be successfully imported into the service.

skill mapping:

Skills: Azure Developer Certification (AZ 204)

Subskills: - Connect to and consume Azure services and third-party services

Competencies: Azure Compute Solutions, Azure Functions, Azure Storage, Blob storage, Azure Services Integration, Azure Event Grid

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

50. You are in charge of deploying a web application for Tech Innovations Inc., which is hosted on Azure. The source code is stored on GitHub at `https://github.com/TechInnovations/webapp-source`. The Azure Web App, named `TechWebApp2024`, should automatically update whenever changes are pushed to the GitHub repository. You aim to use Azure CLI to establish this CI/CD pipeline.

Task:

To automate the deployment from GitHub to the Azure Web App, what sequence of commands should you execute?

A) Use `az webapp up` to create or update the web app and configure source control.

B) Initialize an App Service plan using `az appservice plan create`, then set up the web app with `az webapp create`, and finally link GitHub with `az webapp deployment source config`.

C) Directly clone the GitHub repository to the Azure cloud shell and manually copy the files to the Web App's FTP site.

D) Configure a virtual machine with IIS and manually pull code from GitHub for deployments.

Answer: B

Feedback(if correct):-

Correct Sequence: B) Initialize an App Service plan using `az appservice plan create`, then set up the web app with `az webapp create`, and finally link GitHub with `az webapp deployment source config`.

Detailed Steps:

1. `az appservice plan create`: Begin by creating an Azure App Service plan, which provides the environment necessary to host the web app. Use the command with appropriate parameters, such as the desired SKU for pricing tier and a unique name for the plan, ensuring it's within the same resource group intended for the web app.

2. `az webapp create`: Once the App Service plan is in place, proceed to create the web app itself with this command. Specify the previously created App Service plan as its hosting plan and give the web app a unique name, `TechWebApp2024`, making it ready to serve your application.

3. `az webapp deployment source config`: The final step involves linking the web app to the GitHub repository. This command configures continuous deployment by setting the repository URL and the branch from which deployments should occur. It ensures that updates pushed to the specified branch in the GitHub repository are automatically deployed to the Azure Web App.

Explanation:

Following these steps sets up a continuous integration and deployment (CI/CD) pipeline between a GitHub repository and an Azure Web App. It automates the deployment process, ensuring that the latest version of the application is always available without manual intervention. This approach leverages Azure's native integration capabilities with GitHub to streamline development workflows and deployment processes.

Feedback(if wrong):-

- Option A (`az webapp up`) is incorrect because it doesn't involve creating an App Service plan and configuring source control from GitHub.

- Option C involves manual steps and does not automate the deployment process.

- Option D suggests using a virtual machine with IIS, which is not necessary for deploying a web app to Azure Web App using Azure CLI.

skill mapping:

Skills: Azure Developer Certification (AZ 204)

Subskills: Connect to and consume Azure services and third-party services

Competencies: Azure Compute Solutions

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

51. Your development team at Global Solutions LLC is working on a new project, a web application stored in a GitHub repository (`https://github.com/GlobalSolutions/project-repo`). The application must be hosted on Azure as `GlobalAppService2024`, with the requirement that the deployment process automatically reflects any changes made in the repository. What steps are necessary to configure the Azure environment and automate the deployment process from GitHub using Azure CLI?

A) Configure DNS settings to point to Azure and manually update the web app via FTP.

B) Provision a Docker container on Azure Kubernetes Service (AKS) and pull the latest image from GitHub.

C) Create the hosting environment with `az appservice plan create`, deploy the web app using `az webapp create`, and automate deployments via `az webapp deployment source config`.

D) Set up a Logic App to monitor the GitHub repository and trigger Azure Functions to deploy updates manually.

Answer: C

Feedback(if correct):-

To automate deployments from GitHub to Azure, first, an App Service plan is required to host the web application on Azure. Following this, the `az webapp create` command is used to create the web app `GlobalAppService2024` in the specified service plan. The final step involves linking the web app to the GitHub repository using `az webapp deployment source config`, ensuring that any changes in the repository trigger an automatic deployment to the Azure Web App. This process aligns with best practices for automating deployment pipelines in Azure environments.

Correct Steps:

1. `az appservice plan create` to create a hosting plan on Azure.

2. `az webapp create` to deploy the web application within the Azure App Service plan.

3. `az webapp deployment source config` to connect the Azure Web App with the GitHub repository, enabling automatic deployments.

Feedback(if wrong):-

- Option A involves manual updating via FTP, which does not align with the requirement for automatic deployment from GitHub.

- Option B suggests using Azure Kubernetes Service (AKS) with Docker containers, which is a valid approach but not necessary for a simple web application deployment and does not directly integrate with GitHub for automatic deployments.

- Option D involves setting up a Logic App and Azure Functions, which can be used for automation but is more complex and not directly related to deploying a web application from GitHub using Azure CLI.

skill mapping:

Skills: Azure Developer Certification (AZ 204)

Subskills: Connect to and consume Azure services and third-party services

Competencies: Azure Compute Solutions

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

52. A development team is tasked with enhancing a retail management application, which is crucial for sales representatives operating across diverse geographical locations with varying internet availability. The application utilizes Azure App Service Mobile Apps for its backend. To improve usability in low-connectivity areas, the application must allow sales representatives to access and modify data offline, with all changes synchronized to the cloud backend once an internet connection is re-established. You are responsible for implementing this offline functionality. Which of the following approaches should you take to enable offline data access and ensure data synchronization occurs correctly when the application reconnects to the internet?

A. Initialize a `MobileServiceClient` instance with the mobile app URL, create a `MobileServiceSQLiteStore` for local data storage, define the necessary tables in the local store, and use `GetSyncTable` to interact with data both offline and online. Implement `PushAsync` and `PullAsync` on the synchronization context to handle data synchronization.

```csharp
var client = new MobileServiceClient("MOBILE_APP_URL");
```

```
var store = new MobileServiceSQLiteStore("localstore.db");

store.DefineTable<SalesData>();

await client.SyncContext.InitializeAsync(store);

var salesDataTable = client.GetSyncTable<SalesData>();

await client.SyncContext.PushAsync();

await salesDataTable.PullAsync("allSalesData", salesDataTable.CreateQuery());
```

B. Utilize `GetTable` to directly interact with the Azure backend from the mobile app without local data storage, relying on periodic checks for internet connectivity to manually synchronize data.

```csharp
var client = new MobileServiceClient("MOBILE_APP_URL");

var salesDataTable = client.GetTable<SalesData>();

// Manual synchronization logic here
```

C. Deploy a separate Azure Function to periodically sync data between the mobile app and Azure backend, bypassing the need for local data storage on the mobile device.

```csharp
// Azure Function code for data synchronization
```

D. Implement a custom API within the Azure App Service Mobile Apps backend to manually manage data synchronization and offline storage mechanisms on the mobile device.

```csharp
// Custom API and mobile app logic for manual sync and offline storage
```

Answer: A.

Feedback(if correct):

The correct approach, as described in option A, involves initializing a `MobileServiceClient` with the mobile app URL, creating a `MobileServiceSQLiteStore` for local data storage, defining necessary tables in the local store, and using `GetSyncTable` to interact with data in both offline and online scenarios. This setup is essential for enabling offline functionality in mobile apps that rely on Azure App Service Mobile Apps as their backend. The use of `PushAsync` and `PullAsync` methods for data synchronization ensures that changes made offline are synced with the cloud backend once an internet connection is re-established, providing a seamless user experience and maintaining data integrity across different connectivity scenarios.

A. Initialize a `MobileServiceClient` instance with the mobile app URL, create a `MobileServiceSQLiteStore` for local data storage, define the necessary tables in the local store, and use `GetSyncTable` to interact with data both offline and online. Implement `PushAsync` and `PullAsync` in the synchronization context to handle data synchronization.

This approach allows for the local storage and manipulation of data when offline, using `MobileServiceSQLiteStore` and `GetSyncTable<T>`. Changes are queued and synchronized with the Azure backend through `PushAsync` and `PullAsync` methods once connectivity is restored, ensuring that sales representatives can seamlessly work with the application in environments with variable internet availability. This method leverages Azure's built-in features for offline data sync, making it the most efficient and straightforward solution to meet the requirements of the given scenario.

The key to enabling offline functionality and synchronization in an Azure App Service Mobile Apps backend is to utilize the offline sync feature, which allows the application to store and modify data locally and synchronize changes with the cloud backend later. The `MobileServiceSQLiteStore` is used for local data storage, and the `GetSyncTable` method enables operations on the data that automatically synchronize when connectivity is available, leveraging `PushAsync` and `PullAsync` methods for syncing changes. This approach ensures that sales representatives can access and update sales data even in areas with poor or no internet connectivity, improving the application's usability and reliability in various operational scenarios.

Feedback(if wrong):

B. This option suggests directly interacting with the Azure backend without local data storage, which does not fulfill the offline functionality requirement. Without local storage, the app cannot offer offline access to data or store changes made when offline, making it an incorrect choice for scenarios requiring offline capabilities.

C. Deploying a separate Azure Function for data synchronization introduces unnecessary complexity and does not directly address the requirement for local data storage and offline data access in the mobile app itself. While Azure Functions can be part of a broader architecture for background tasks or processing, they do not replace the need for local data synchronization logic within the app.

D. Implementing a custom API for managing data synchronization and offline storage mechanisms manually adds complexity and is not necessary when Azure Mobile Apps provides built-in support for these functionalities through the `MobileServiceSQLiteStore` and synchronization context. This approach also risks reinventing features that are already well-implemented and tested in the existing Azure Mobile Apps framework.

Skill Mapping:

Skills: Azure Developer Certification (AZ 204)

Subskills: Connect to and consume Azure services and third-party services

Competencies: Azure Compute Solutions: Azure Storage:

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

53. a healthcare organization is developing a patient care mobile application using Azure App Service Mobile Apps, there's a critical requirement for the app to remain operational in offline modes due to the variable internet access in remote areas. It must ensure patient data is locally stored, synchronized, and updated with the cloud backend upon re-establishing internet connectivity. How should the offline support and data synchronization be implemented?

A) Utilize `MobileServiceClient` without setting up a `MobileServiceSQLiteStore`, relying on cloud connectivity for data storage and retrieval.

B) Deploy `MobileServiceClient` with `GetTable` for direct access to cloud services, bypassing the need for local data storage and synchronization.

C) Initialize a `MobileServiceClient` connection, set up `MobileServiceSQLiteStore` for offline storage, define the necessary tables, and use `GetSyncTable` for synchronization.

D) Configure a direct connection to Azure Blob Storage for data management, avoiding the use of `MobileServiceClient` and local data storage solutions.

Answer: C

Feedback(if correct):

The correct approach for implementing offline support and data synchronization in this scenario is:

C) Initialize a `MobileServiceClient` connection, set up `MobileServiceSQLiteStore` for offline storage, define the necessary tables, and use `GetSyncTable` for synchronization.

This option ensures the application can operate in offline modes by storing patient data locally in `MobileServiceSQLiteStore` and synchronizing it with the Azure backend upon re-establishing internet connectivity. This approach meets the critical requirement of maintaining operational functionality in remote areas with variable internet access, providing seamless patient care.

Feedback(if wrong):

Option A : Utilizing `MobileServiceClient` without setting up a `MobileServiceSQLiteStore` relies solely on cloud connectivity for data storage and retrieval. This does not support offline functionality, failing to meet the requirement for the app to operate in areas with unreliable internet access.

Option B : Deploying `MobileServiceClient` with `GetTable` provides direct access to cloud services but bypasses the need for local data storage and synchronization. This approach does not cater to the offline operational requirements of the application, making it unsuitable for environments with inconsistent internet connectivity.

Option D : Configuring a direct connection to Azure Blob Storage for data management and avoiding the use of `MobileServiceClient` and local data storage solutions does not address the need for offline data access and synchronization. This method lacks the necessary infrastructure to support the application's offline functionality requirements.

Skill Mapping:

Skills: Azure Developer Certification (AZ 204)

Subskills: Connect to and consume Azure services and third-party services

Competencies: Azure Compute Solutions: Azure Storage:

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

54. You are tasked with architecting a real-time analytics solution for a retail company that processes large volumes of sales data. The solution must efficiently handle spikes in data and distribute processing tasks among several consumers without requiring them to constantly check for new messages. Which two Azure services should you integrate to ensure a scalable publish-subscribe mechanism that supports real-time data processing and load balancing?

A) Azure Event Hubs

B) Azure Event Grid

C) Azure Queue Storage

D) Azure Service Bus

Answer: A, D

Feedback(if correct):

Selecting Azure Event Hubs (A) and Azure Service Bus (D) as the answers correctly identify the two Azure services that support scalable real-time data processing and a publish-subscribe mechanism. Azure Event Hubs is ideal for handling massive volumes of event data in real-time, while Azure Service Bus offers advanced messaging capabilities like topics and subscriptions for efficient message distribution and load balancing. This choice demonstrates an understanding of Azure's messaging services and their appropriate application in scenarios requiring high throughput and dynamic scalability without constant polling.

Feedback(if wrong):

Azure Event Grid (B): While Azure Event Grid supports event-driven architectures and can handle large-scale event routing, it is not specifically optimized for the high-throughput data ingestion scenarios described in the question.

Azure Queue Storage (C): This service provides simple messaging for cloud-based applications. However, it does not inherently support a publish-subscribe model nor is it designed for the real-time processing of large volumes of data like Azure Event Hubs or the complex messaging patterns supported by Azure Service Bus.

Skill Mapping:

Skills: Azure Developer Certification (AZ 204)

Subskills: Connect to and consume Azure services and third-party services

Competencies: Azure Compute Solutions: Azure Storage:

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

55. You are designing a cloud solution for an IoT-based smart city project. The system requires an efficient way to decouple microservices and enable event-driven communication between them, facilitating smooth data flow without continuous polling for updates. Which two Azure services should you use to implement a publish-subscribe model that decouples components and ensures timely message delivery?

A) Azure Queue Storage

B) Azure Functions

C) Azure Event Grid

D) Azure Service Bus

Answer: C, D

Feedback(if correct):

Choosing Azure Event Grid (C) and Azure Service Bus (D) accurately reflects the requirements for implementing a decoupled, event-driven architecture that supports efficient communication between microservices in a smart city IoT solution. Azure Event Grid excels in managing events across Azure services and applications, facilitating an event-driven communication model that minimizes the need for continuous polling. Azure Service Bus enhances this architecture by providing robust messaging capabilities, including the publish-subscribe model for complex inter-service communication. This demonstrates a solid grasp of leveraging Azure messaging services to achieve application decoupling and efficient message delivery.

Feedback(if wrong):

Azure Queue Storage (A): This option offers basic message queuing services but lacks the advanced pub/sub messaging capabilities and the event-driven model provided by Azure Event Grid and Azure Service Bus, making it less suitable for the described scenario.

Azure Functions (B): While Azure Functions can respond to events from various services, including Event Grid and Service Bus, on its own, it does not provide messaging or event routing capabilities and thus does not meet the criteria for a messaging service that supports a publish-subscribe model for decoupling components.

Skill Mapping:

Skills: Azure Developer Certification (AZ 204)

Subskills: Connect to and consume Azure services and third-party services

Competencies: Azure Compute Solutions: Azure Storage:

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application