Azure Developer Certification (AZ-204)

Exam1

1. You are developing a cloud-based solution using Azure Functions to process messages from an Azure Service Bus queue. The solution requires handling different types of messages with varying processing times. It is crucial to ensure optimal resource utilization, minimize latency, and manage costs effectively. What configuration strategy should you adopt for Azure Functions to meet the specified requirements? Choose the best solution from the options below:

A. Implement a dedicated App Service Plan for each function to ensure dedicated resources and better isolation.

B. Use a Consumption Plan for all functions, allowing automatic scaling based on the number of incoming messages.

C. Combine all functions into a single Premium Plan to ensure enhanced performance and efficient resource utilization.

D. Deploy functions in a Standard Plan with multiple instances for redundancy, manually adjusting the scale based on the workload.

Answer: B

Feedback(if correct):

Using a Consumption Plan for all functions allows automatic scaling based on the number of incoming messages, ensuring optimal resource utilization and effective cost management.

Feedback(if wrong):

Option A (Dedicated App Service Plan): This might lead to underutilization of resources during periods of low activity.

Option C (Single Premium Plan): Might be overkill and more expensive than necessary for functions with varying processing times.

Option D (Standard Plan with Manual Adjustment): Manual adjustment can be cumbersome and less responsive to varying workloads.

Skill: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competency: Configuring Azure Functions

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

2. Your team is building a serverless application in Azure that analyzes audio files uploaded by customers. Upon receiving an audio file, the application must initiate transcription and perform sentiment analysis. Transcriptions and analyses should commence immediately after file upload completion. Given the large volume of anticipated daily submissions, efficiency and scalability are paramount concerns. To address these requirements, you contemplate designing a process centered around Azure Functions, activated via Azure Blob Storage events. You intend to utilize Cognitive Services Text Analytics API for sentiment analysis and Speech Services SDK for transcription tasks. Considering the described scenario, select the appropriate statement regarding this proposed solution:

A. This solution satisfies the project criteria since it capitalizes on Azure Functions' seamless connection with Blob Storage events, thereby triggering immediate transcription and sentiment analysis post-upload.

B. Regrettably, this proposal falls short as neither Cognitive Services nor Speech Services APIs integrate natively with Azure Functions, necessitating additional architectural layers for interaction mediation.

C. Despite harnessing Azure Functions' responsiveness to Blob Storage events, this design encounters difficulties in catering to the expected surge in submission rates, thus impacting overall system effectiveness negatively.

D. While utilizing Azure Functions for activating downstream processes sounds promising, picking General Purpose v1 Storage Account over V2 hinders event integration functionality, rendering this setup insufficient for our purposes.

Answer: A

Feedback(if correct):

This solution is aligned with the project criteria as Azure Functions seamlessly integrates with Blob Storage events, enabling immediate transcription and sentiment analysis after file uploads.

Feedback(if wrong):

Both Cognitive Services and Speech Services APIs can integrate with Azure Functions. There is no need for additional architectural layers for interaction mediation. The proposed solution is feasible. The design, leveraging Azure Functions and Blob Storage events, is intended to address efficiency and scalability concerns. The statement about difficulties in handling submission rates is not accurate in the context of the proposed solution. The type of Storage Account (General Purpose v1 or V2) doesn't hinder event integration functionality with Azure Functions. The proposed solution, utilizing Azure Functions for activation and downstream processes, remains viable regardless of the Storage Account type chosen.

Question Type: Single Selection

Skill: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competence: Triggering Azure Functions with Events

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Evaluation

3. You're developing a mobile app for a delivery service. Drivers choose restaurants, receive orders, and only see orders from selected restaurants. The first accepting driver removes an order. What Azure Service Bus solution should you implement?

A. Create a single Service Bus Namespace. Create a Service Bus Topic for each restaurant. Create a Service Bus subscription for each restaurant.

B. Create a single Service Bus Namespace. Create a single Service Bus Topic. Create a single Service Bus subscription.

C. Create a Service Bus Namespace for each restaurant. Create a Service Bus Topic for each restaurant. Create a Service Bus subscription for each restaurant.

D. Create 2 Service Bus Topics for each restaurant. Create a single Service Bus subscription.

Answer: A

Feedback(if correct):

This solution aligns with the best practices for using Azure Service Bus, where a single namespace is created to contain topics, and each restaurant has its topic with subscriptions for drivers.

Feedback(if wrong):

creating a single Service Bus subscription does not align with the requirement of having subscriptions for each restaurant, as specified in the scenario. creating a Service Bus Namespace for each restaurant would lead to unnecessary complexity and is not the recommended approach. creating 2 Service Bus Topics for each restaurant is unnecessary and does not align with the best practice of using a single topic for each entity.

Skills:

Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Designing Azure Service Bus Solutions

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

Bloom's Taxonomy Category: Application

4. You are developing a cloud-based solution for a food delivery service. The service operates with a network of independent drivers who deliver orders from various restaurants to customers. The application follows this workflow:

1. A customer places an order from a restaurant through the app.
2. The order is sent to all available drivers in the area.
3. Only drivers who have opted to deliver for that restaurant receive the order notification.
4. The first driver to accept the order removes it from the list of available orders.

You need to implement an Azure Service Bus solution to manage the distribution of orders to drivers.

Which sequence of actions should you perform to implement this solution? Choose the best sequence from the options below:

A. Create a single Service Bus Namespace -> Create a Service Bus Topic for each restaurant -> Create a Service Bus Subscription for each driver.

B. Create a Service Bus Namespace for each restaurant -> Create a single Service Bus Topic -> Create a Service Bus Subscription for each driver.

C. Create a single Service Bus Namespace -> Create a single Service Bus Topic -> Create a Service Bus Subscription for each restaurant.

D. Create a Service Bus Namespace for each driver -> Create a Service Bus Topic for each restaurant -> Create a single Service Bus Subscription.

Answer: A.

Feedback(if correct

With this option, you can create a centralized Service Bus Namespace that manages all the communication between the components of the application. Then, for each restaurant, you can create a distinct Topic that corresponds to that establishment, keeping the communications organized. Next, by creating a Subscription for each driver, you guarantee that the drivers who have opted to deliver orders for a particular restaurant will get the order notifications, and the first one to claim the order will be removed from the list of available orders.

This approach permits decoupling and independence between the publishers (restaurants) and the subscribers (drivers), letting them grow separately and ensuring loose coupling for scalability and fault tolerance. Besides, this solution aligns with the best practices for using Azure Service Bus, where a single namespace is utilized to encompass topics, and each restaurant owns its topic with subscriptions for drivers.

Feedback (if wrong):

B) Creating a Service Bus Namespace for each restaurant may add unnecessary complexity and duplicated efforts in managing multiple namespaces. Instead, a single namespace can host multiple topics for each restaurant.

C) Creating a single Service Bus Topic for all restaurants may not allow enough granularity to properly distribute orders based on the specific restaurant. Different restaurants might require different distributions or filterings, and having separate topics per restaurant addresses this situation adequately.

D) Creating a Service Bus Namespace for each driver is impractical and unscalable. Drivers should merely receive filtered messages based on their preferences and location, and creating subscriptions for each driver inside a single namespace is a better fit for this scenario.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Setting up and configuring APIs, Implementing rate limiting and throttling, Enabling caching, Configuring authentication

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

5. FoodieExpress, a popular food delivery service, is rolling out a new feature allowing customers to tip drivers through their web app. To facilitate this, they need to add a "tip" field to their existing Azure Cosmos DB orders collection. The problem is, that some partner restaurants haven't updated their integrations yet, so some orders might arrive without the tip field defined. You're tasked with creating an Azure Function trigger that ensures all orders have a properly formatted "tip" field:

If the tip field is present, it must be a number.

If the tip field is missing for orders from older services, the trigger should handle the situation gracefully without impacting order processing.

How can you write the best trigger code to meet these requirements while also prioritizing efficiency, maintainability, and security?

Key Requirements:

1. Enforce `Ordertip` Property: The trigger must ensure that all incoming orders contain an `Ordertip` property with a numeric value.

2. Handle Older Web Services: It must gracefully handle orders that might not have the `Ordertip` property yet, potentially due to older web services not being updated for the new feature.

3. Error Handling: The trigger should provide informative error messages or responses when the `Ordertip` property is missing or invalid.

4. Efficiency and Maintainability: The code should be written in a clear, concise, and efficient manner to facilitate maintenance and understanding.

5. Security: The validation approach should consider potential security risks like injection attacks and mitigate them appropriately.

Code Snippet:

```javascript
function validateOrderTrigger() {
  // Slot 1: Access incoming order document
  const order = getContext().getRequest().getBody();

  // Slot 2: Check for Ordertip property and its type
  if (/ Condition for missing or invalid Ordertip /) {
    // Slot 3: Handle missing or invalid Ordertip property
  } else {
    // Process the order
  }
}
```

Given the prompt's requirements, choose the correct option for Slot 1.

A. const order = getContext().getRequest().getBody();

B. if (!order || typeof order.Ordertip !== 'number') {

C. if (!order || !('Ordertip' in order) || isNaN(order.Ordertip)) {

D. if (!order || typeof order.Ordertip !== 'numeric') {

Answers: A

Feedback(if correct):-

getContext().getRequest().getBody() is the appropriate method to access the incoming order document in the context of an Azure Function trigger.

Slot 1: Access incoming order document

Slot 2: Check for Ordertip property and its type

In Slot 2, we're ensuring that the order exists and that the Ordertip property is indeed a number. If not, we jump into Slot 3 to handle any missing or invalid Ordertip properties.

Feedback(if wrong):-

if (!order || typeof order.Ordertip !== 'number') {: This condition checks for the existence of Ordertip and its type It doesn't access the incoming order document as required in Slot 1.

if (!order || !('Ordertip' in order) || isNaN(order.Ordertip)) {: This condition checks for the existence and type of Ordertip. However, it doesn't directly access the incoming order document, making it inappropriate for Slot 1.

if (!order || typeof order.Ordertip !== 'numeric') {: The typeof operator checks for the type 'number', not 'numeric', so this condition wouldn't correctly handle the case where Ordertip is a number. Additionally, it doesn't access the incoming order document as required in Slot 1.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Creating and managing APIs, Setting up API gateways, Configuring policies, Authenticating and authorizing access

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

6. FoodieExpress, a popular food delivery service, is rolling out a new feature allowing customers to tip drivers through their web app. To facilitate this, they need to add a "tip" field to their existing Azure Cosmos DB orders collection. The problem is, that some partner restaurants haven't updated their integrations yet, so some orders might arrive without the tip field defined.

You're tasked with creating an Azure Function trigger that ensures all orders have a properly formatted "tip" field:

If the tip field is present, it must be a number.

If the tip field is missing for orders from older services, the trigger should handle the situation gracefully without impacting order processing.

How can you write the best trigger code to meet these requirements while also prioritizing efficiency, maintainability, and security?

Key Requirements:

1. Enforce `Ordertip` Property: The trigger must ensure that all incoming orders contain an `Ordertip` property with a numeric value.

2. Handle Older Web Services: It must gracefully handle orders that might not have the `Ordertip` property yet, potentially due to older web services not being updated for the new feature.

3. Error Handling: The trigger should provide informative error messages or responses when the `Ordertip` property is missing or invalid.

4. Efficiency and Maintainability: The code should be written in a clear, concise, and efficient manner to facilitate maintenance and understanding.

5. Security: The validation approach should consider potential security risks like injection attacks and mitigate them appropriately.

Code Snippet:

```javascript
function validateOrderTrigger() {
  // Slot 1: Access incoming order document
  const order = getContext().getRequest().getBody();

  // Slot 2: Check for Ordertip property and its type
  if (/ Condition for missing or invalid Ordertip /) {
    // Slot 3: Handle missing or invalid Ordertip property
  } else {
    // Process the order
  }
}
```

Given the prompt's requirements, choose the correct option for Slot 2.

A. const order = getContext().getRequest().getBody();

B. if (!order || typeof order.Ordertip !== 'number') {

C. if (!order || !('Ordertip' in order) || isNaN(order.Ordertip)) {

D. if (!order || typeof order.Ordertip !== 'numeric') {

Answer: C

Feedback(if correct): `if (!order || !('Ordertip' in order) || isNaN(order.Ordertip)) {`

This condition correctly checks for the existence of `Ordertip` in the order document and ensures it is either a numeric value or not present at all, addressing the key requirements for Slot 2.

Feedback(if wrong):

`const order = getContext().getRequest().getBody();`: This line is used for accessing the incoming order document in Slot 1, not for checking the `Ordertip` property and its type in Slot 2.

`if (!order || typeof order.Ordertip !== 'number') {`: This condition checks the type of `Ordertip` but doesn't account for the situation where `Ordertip` might be missing, making it less suitable for Slot 2.

`if (!order || typeof order.Ordertip !== 'numeric') {`: The `typeof` operator checks for the type `'number'`, not `'numeric'`, so this condition wouldn't correctly handle the case where `Ordertip` is a number. Additionally, it doesn't consider the situation where `Ordertip` might be missing.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Creating and managing APIs, Setting up API gateways, Configuring policies, Authenticating and authorizing access

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

7. FoodieExpress, a popular food delivery service, is rolling out a new feature allowing customers to tip drivers through their web app. To facilitate this, they need to add a "tip" field to their existing Azure Cosmos DB orders collection. The problem is, that some partner restaurants haven't updated their integrations yet, so some orders might arrive without the tip field defined.

You're tasked with creating an Azure Function trigger that ensures all orders have a properly formatted "tip" field:

If the tip field is present, it must be a number.

If the tip field is missing for orders from older services, the trigger should handle the situation gracefully without impacting order processing.

How can you write the best trigger code to meet these requirements while also prioritizing efficiency, maintainability, and security?

Key Requirements:

1. Enforce `Ordertip` Property: The trigger must ensure that all incoming orders contain an `Ordertip` property with a numeric value.

2. Handle Older Web Services: It must gracefully handle orders that might not have the `Ordertip` property yet, potentially due to older web services not being updated for the new feature.

3. Error Handling: The trigger should provide informative error messages or responses when the `Ordertip` property is missing or invalid.

4. Efficiency and Maintainability: The code should be written in a clear, concise, and efficient manner to facilitate maintenance and understanding.

5. Security: The validation approach should consider potential security risks like injection attacks and mitigate them appropriately.

Code Snippet:

```javascript
function validateOrderTrigger() {
  // Slot 1: Access incoming order document
  const order = getContext().getRequest().getBody();

  // Slot 2: Check for Ordertip property and its type
  if (/ Condition for missing or invalid Ordertip /) {
    // Slot 3: Handle missing or invalid Ordertip property
  } else {
    // Process the order
  }
}
```

Given the prompt's requirements, choose the correct option for Slot 3.

A. const order = getContext().getRequest().getBody();

B. if (!order || typeof order.Ordertip !== 'number') {

C. if (!order || !('Ordertip' in order) || isNaN(order.Ordertip)) {

D. if (!order || typeof order.Ordertip !== 'numeric') {

Answer: C

Feedback(if correct):

`if (!order || !('Ordertip' in order) || isNaN(order.Ordertip)) {`

This condition effectively checks whether `Ordertip` is either missing or not a numeric value, satisfying the key requirements for Slot 3. It gracefully handles the absence of `Ordertip` and ensures that it's either a number or not present at all.

Feedback(if wrong):

`const order = getContext().getRequest().getBody();`: This line is used for accessing the incoming order document in Slot 1, not for handling missing or invalid `Ordertip` in Slot 3.

`if (!order || typeof order.Ordertip !== 'number') {`: This condition checks the type of `Ordertip` but doesn't account for the situation where `Ordertip` might be missing, making it less suitable for Slot 3.

`if (!order || typeof order.Ordertip !== 'numeric') {`: The `typeof` operator checks for the type `'number'`, not `'numeric'`, so this condition wouldn't correctly handle the case where `Ordertip` is a number. Additionally, it doesn't consider the situation where `Ordertip` might be missing.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Creating and managing APIs, Setting up API gateways, Configuring policies, Authenticating and authorizing access

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

8. You work as a developer for a company that has recently migrated its application to an Azure Kubernetes Service (AKS) cluster. The application consists of a web frontend and a database backend, both of which require load balancing to handle varying levels of user requests. You need to design and configure an ingress controller for the AKS cluster to efficiently distribute incoming HTTP(S) traffic between the web frontend instances while maintaining high availability and scalability. You also need to ensure that traffic is routed only to the appropriate services based on the URL paths. Which two actions should you perform first when designing and configuring an ingress controller for the AKS cluster to efficiently distribute incoming HTTP(S) traffic between the web frontend instances while maintaining high availability and scalability?

A) Deploy an ingress controller such as NGINX in the AKS cluster

B) Configure the AKS cluster to use Azure Container Instances (ACI)

C) Define Ingress Resources in Kubernetes to define rules for routing incoming HTTP(S) traffic to the appropriate services based on URL paths.

D) Update the default ingress rule to send traffic to the web frontend service

Answer: A, C

Feedback(if correct):

Deploying an ingress controller such as NGINX in the AKS cluster (Option A) allows you to manage and route incoming HTTP(S) traffic efficiently. Defining ingress rules that route traffic based on URL paths (Option C) ensures that traffic is directed to the appropriate services based on specific paths.

Feedback(if wrong):

Configuring the AKS cluster to use Azure Container Instances (ACI) is unrelated to setting up an ingress controller for routing traffic and load balancing. Updating the default ingress rule is not a specific action; instead, defining custom ingress rules (Option C) is more appropriate for routing traffic based on URL paths.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Creating and managing APIs, Setting up API gateways, Configuring policies, Authenticating and authorizing access

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

9. You work as a developer for a company that has recently migrated its application to an Azure Kubernetes Service (AKS) cluster. The application consists of a web frontend and a database backend, both of which require load balancing to handle varying levels of user requests. You have recently migrated a critical application to an Azure Kubernetes Service (AKS) cluster, which consists of a web frontend and a database backend. The ingress controller has been configured to efficiently distribute incoming HTTP(S) traffic between the web frontend instances while maintaining high availability and scalability. Additionally, you have secured the web frontend and database backend components against unauthorized access. Now, you need to further improve the networking security of the AKS cluster without causing disruptions to the existing architecture. Please pick the correct pair of actions that would bolster the AKS cluster's networking security without causing disruptions to the existing architecture.

A) Configure network policies to limit communication between pods and nodes in the cluster.

B) Enable Azure Security Center for Kubernetes to monitor and detect threats in the cluster.

C) Use Azure Application Gateway to terminate SSL connections and offload certificate management.

D) Disable all ingress traffic to the AKS cluster and only allow traffic from specific IP addresses.

Answer: A , B

Feedback(if correct):

Configuring network policies helps limit communication between pods and nodes in the cluster, enhancing security. Enabling Azure Security Center for Kubernetes adds an additional layer of monitoring and threat detection.

Feedback(if wrong):

Using Azure Application Gateway is related to SSL termination but is not directly focused on improving networking security within the AKS cluster. Disabling all ingress traffic and only allowing traffic from specific IP addresses (Option D) would likely disrupt existing functionality and is not recommended for general security enhancement within an AKS cluster.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskill: Develop Azure compute solutions

Competencies: Sub-topic: Implement Advanced Networking Features

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

10. Your company is deploying a new microservices-based application on Azure Kubernetes Service (AKS) cluster. To manage and deploy container images, your client has requested you to use Azure Container Registry (ACR). You need to publish a container image to the ACR and ensure it has the advantages of using ACR over other container image repositories. What steps should you take to achieve this?

A) Build a container image, push it to Docker Hub, and configure AKS to pull images from Docker Hub.

B) Build a container image, push it to ACR, and configure AKS to pull images from ACR.

C) Build a container image, push it to the Amazon Elastic Container Registry (ECR), and configure AKS to pull images from ECR.

D) Build a container image, push it to Google Container Registry (GCR), and configure AKS to pull images from GCR.

Answer: B

Feedback(if correct):

Option B is correct. To take advantage of Azure Container Registry (ACR) benefits, build the container image, push it to ACR, and configure AKS to pull images from ACR.

Feedback(if wrong):

A) Using Docker Hub does not provide the benefits specific to Azure and ACR. Amazon Elastic Container Registry (ECR) is not the preferred choice when deploying on Azure with AKS. Google Container Registry (GCR) is not the recommended solution when working with Azure services like AKS.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Sub-Topics: Publish container images to Azure Container Registry, Configure Azure Kubernetes Service to pull images from Azure Container Registry

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

11. You're a software developer tasked with deploying your team's latest Java web application to a staging environment in Azure for testing and validation. You'll be using GitHub as your code repository and leveraging the power of Azure App Service deployment slots. Your goal is to perform the initial deployment efficiently and securely. Among the given choices below, which TWO actions should you prioritize when initially setting up your Azure App Service for deployment?

A. Create a web app and deployment slot named "businesswebapp" and "staging", respectively.

B. Configure authentication for the web app using Azure Active Directory.

C. Choose a resource group location based on expected user distribution and cost considerations.

D. Connect your Azure subscription to your GitHub repository using a service principal or managed identity.

Answer: A , D

Feedback(if correct):

When initiating the setup of an Azure App Service for deploying your team's latest Java web application to a staging environment, you want to take care of two fundamental aspects immediately. Prioritize creating a web app and deployment slot named "businesswebapp" and "staging", respectively. This way, you allocate sufficient space in Azure for receiving the incoming web application codebase and smartly handle deployments, making it easier to roll back or forward among versions without disrupting the live site. Simultaneously, connect your Azure subscription to your GitHub repository using a service principal or managed identity, establishing a secured pathway for code movement. This connection avoids credential mishandling and keeps sensitive information protected by automating the transferal process between GitHub and Azure. Taking these proactive measures strengthens the cornerstone of a resilient and effective deployment pipeline.

Feedback(if wrong):

(configure authentication) is unnecessary at this stage because it doesn't contribute significantly to the initial deployment efficiency and security. That can be addressed later in the development cycle. (resource group location) involves tradeoffs primarily affected by latency tolerance and budget constraints. Although important, it shouldn't hinder starting off the deployment process.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Sub-Topics: Configure source control integration, Configure deployment slots

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

12. You're a software developer tasked with deploying your team's latest Java web application to a staging environment in Azure for testing and validation. You'll be using GitHub as your code repository and leveraging the power of Azure App Service deployment slots. Your goal is to perform the initial deployment efficiently and securely. After setting up the App Service, how should you configure the deployment process for optimal security and control?

A. Deploy the web app to the "staging" slot manually from the Azure portal.

B. Implement a CI/CD pipeline that automatically deploys code pushed to the "master" branch in GitHub, emphasizing security best practices.

C. Configure the web app to continuously monitor application health and trigger deployments upon successful builds.

D. Restrict access to the "staging" slot and surrounding resources to authorized users within your development team for optimal security and control.

Answer: B, D

Feedback (if correct): Implementing a CI/CD pipeline eliminates manual mistakes, eases tracking of deployments, and follows security best practices. Furthermore, restricting access to the "staging" slot and surrounding resources secures the environment, reducing risks linked to unwanted manipulations or breaches.

Feedback (if wrong): Manual deployment exposes the process to human error, making tracing deployments challenging, and lacking the consistency and ease of rollbacks. Moreover, manual processes may pose security risks due to potential unauthorized access. Continuous monitoring, while essential for observability and maintenance, might not directly contribute to optimal security and control during the deployment process.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Sub-Topics: Automate deployments, Manage access controls

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

13. You are tasked with deploying a Python-based web application to Azure App Service. After deployment, you encounter the following error when attempting to access the application:

```
Access to XMLHttpRequest at 'https://api.azurewebsites.net/data' from origin 'https://localhost:5000' has been blocked by CORS policy.
```

To address this issue and enable cross-origin resource sharing, what action should you take?

A. Configure a custom domain for the App Service.

B. Implement authentication for the Python application.

C. Enable Cross-Origin Resource Sharing (CORS) for the App Service.

D. Install an SSL certificate for the App Service.

Answer: C

Feedback(if correct):-

To resolve the CORS policy issue, it is essential to enable Cross-Origin Resource Sharing (CORS) for the Azure App Service. This configuration allows the Python application hosted on the App Service to accept requests from the specified origin, in this case, 'https://localhost:5000.'

Feedback(if wrong):- (SSL certificate): While securing communication is important, it doesn't directly address the CORS policy restriction on origins.

(Authentication): Implementing authentication may be necessary for security, but it doesn't solve the specific issue of allowing requests from the specified origin.

(Custom domain): Mapping a custom domain can enhance branding but doesn't impact the CORS configuration that restricts access from specific origins.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Sub-Topics: Configure Cross-Origin Resource Sharing (CORS)

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

14. BestFood an imaginary company specializing in food products, is transitioning its ASP.NET Core web app to Azure. Your responsibility is to provision an App Service Web App to host the Docker image, ensuring the custom domain www.bestfood.com is mapped to the App Service web app. A resource group named `BestFoodWebResourceGroup` has already been established in the WestUS region, containing an App Service Plan named `BestFoodLinuxPlan`. Arrange the Azure CLI commands in the correct order to achieve this. Choose the correct sequence from the options below:

A.

```bash
appName="BestFoodWebApp$random"

location="WestUS"

dockerHubContainerPath="BestFood/web:v1"

fqdn="http://www.bestfood.com"


az webapp create --name $appName --plan BestFoodLinuxPlan --resource-group BestFoodWebResourceGroup


az webapp config container set --docker-custom-image-name $dockerHubContainerPath --name $appName --resource-group BestFoodWebResourceGroup


az webapp config hostname add --webapp-name $appName --resource-group BestFoodWebResourceGroup --hostname $fqdn
```

B.

```bash
appName="BestFoodWebApp$random"

location="WestUS"

dockerHubContainerPath="BestFood/web:v1"

fqdn="http://www.bestfood.com"


az webapp create --name $appName --plan BestFoodLinuxPlan --resource-group BestFoodWebResourceGroup


az webapp config hostname add --webapp-name $appName --resource-group BestFoodWebResourceGroup --hostname $fqdn


az webapp config container set --docker-custom-image-name $dockerHubContainerPath --name $appName --resource-group BestFoodWebResourceGroup
```

C.

```bash
appName="BestFoodWebApp$random"

location="WestUS"

dockerHubContainerPath="BestFood/web:v1"

fqdn="http://www.bestfood.com"


az webapp config container set --docker-custom-image-name $dockerHubContainerPath --name $appName --resource-group BestFoodWebResourceGroup


az webapp config hostname add --webapp-name $appName --resource-group BestFoodWebResourceGroup --hostname $fqdn


az webapp create --name $appName --plan BestFoodLinuxPlan --resource-group BestFoodWebResourceGroup
```

D.

```bash
appName="BestFoodWebApp$random"
location="WestUS"
dockerHubContainerPath="BestFood/web:v1"
fqdn="http://www.bestfood.com"

az webapp config container set --docker-custom-image-name $dockerHubContainerPath --name $appName --resource-group BestFoodWebResourceGroup

az webapp create --name $appName --plan BestFoodLinuxPlan --resource-group BestFoodWebResourceGroup

az webapp config hostname add --webapp-name $appName --resource-group BestFoodWebResourceGroup --hostname $fqdn
```

Answer: A.

Feedback(if correct):

I apologize if there was any confusion. However, after reviewing the information, I still believe that Option A is the correct sequence of Azure CLI commands for your scenario:

```bash
appName="BestFoodWebApp$random"
location="WestUS"
dockerHubContainerPath="BestFood/web:v1"
fqdn="http://www.bestfood.com"

# Create a web app in the specified resource group and service plan
az webapp create --name $appName --plan BestFoodLinuxPlan --resource-group BestFoodWebResourceGroup

# Configure the web app to use the specified Docker image
az webapp config container set --docker-custom-image-name $dockerHubContainerPath --name $appName --resource-group BestFoodWebResourceGroup
```

```
# Add the custom domain to the web app

az webapp config hostname add --webapp-name $appName --resource-group BestFoodWebResourceGroup --hostname $fqdn
```

This sequence first creates the web app, then sets the Docker container for the web app, and finally adds the custom domain to the web app. The other options are incorrect because they either set the Docker container or add the custom domain before the web app is created. The web app must exist before these configurations can be applied. If you have any further questions or need clarification.

Feedback(if wrong):

B.

```bash
appName="BestFoodWebApp$random"

location="WestUS"

dockerHubContainerPath="BestFood/web:v1"

fqdn="http://www.bestfood.com"

az webapp create --name $appName --plan BestFoodLinuxPlan --resource-group BestFoodWebResourceGroup

az webapp config hostname add --webapp-name $appName --resource-group BestFoodWebResourceGroup --hostname $fqdn

az webapp config container set --docker-custom-image-name $dockerHubContainerPath --name $appName --resource-group BestFoodWebResourceGroup
```

The reason the selection is incorrect is that the custom domain (`www.bestfood.com`) is added to the App Service Web App before the Docker container configuration is applied. This results in an attempt to map the custom domain to a non-existent container.

C.

```bash
appName="BestFoodWebApp$random"

location="WestUS"

dockerHubContainerPath="BestFood/web:v1"

fqdn="http://www.bestfood.com"


az webapp config container set --docker-custom-image-name $dockerHubContainerPath --name $appName --resource-group BestFoodWebResourceGroup


az webapp config hostname add --webapp-name $appName --resource-group BestFoodWebResourceGroup --hostname $fqdn


az webapp create --name $appName --plan BestFoodLinuxPlan --resource-group BestFoodWebResourceGroup
```

In this case, the Docker container is configured before the App Service Web App is created. However, the `create` command should be executed first to generate a default hostname for the App Service Web App. Then, the Docker container can be configured, and finally, the custom domain can be added.

D.

```bash
appName="BestFoodWebApp$random"

location="WestUS"

dockerHubContainerPath="BestFood/web:v1"

fqdn="http://www.bestfood.com"


az webapp config container set --docker-custom-image-name $dockerHubContainerPath --name $appName --resource-group BestFoodWebResourceGroup


az webapp create --name $appName --plan BestFoodLinuxPlan --resource-group BestFoodWebResourceGroup


az webapp config hostname add --webapp-name $appName --resource-group BestFoodWebResourceGroup --hostname $fqdn
```

the Docker container is configured before the App Service Web App is created. But more importantly, the custom domain is added after the App Service Web App is created, which prevents the default hostname from functioning appropriately.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Sub-Topics: Provision an App Service Web App, Map a custom domain to an App Service Web App

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

15. You are tasked with automating the deployment of an innovative e-commerce application for a fictitious company named "TechTrends." The company's web app, called "TechStore," needs to be deployed to Azure using a CLI script. The script utilizes the following variables:

- Git Repository (TechTrendsRepo): <https://github.com/techtrends/webapp>
- Web App Name (TechStoreWebApp): TechStore2024

You must ensure that the application is automatically deployed from the provided GitHub repository to the newly created web app in Azure. Arrange the Azure CLI commands to accomplish this task. Select the appropriate options in the answer area.

A.

```bash
appName="TechStore2024"
location="EastUS"
gitRepo="https://github.com/techtrends/webapp"

az group create --location $location --name techstore-rg
az appservice plan create --name techstore-asp --location $location --resource-group techstore-rg --sku FREE
az webapp create --name $appName --plan techstore-asp --resource-group techstore-rg
```

B.

```bash
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --continuous-deployment
```

C.

```bash
appName="TechStore2024"
location="EastUS"
gitRepo="https://github.com/techtrends/webapp"

az group create --location $location --name techstore-rg
az appservice plan create --name techstore-asp --location $location --resource-group techstore-rg --sku FREE
az webapp create --name $appName --plan techstore-asp --resource-group techstore-rg
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --manual-integration
git clone $gitRepo
```

D.

```bash
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --triggers-include linux
```

Answer: A, B

Feedback (if correct): options ```bash

appName="TechStore2024"

location="EastUS"

gitRepo="https://github.com/techtrends/webapp"

az group create --location $location --name techstore-rg

az appservice plan create --name techstore-asp --location $location --resource-group techstore-rg --sku FREE

```
az webapp create --name $appName --plan techstore-asp --resource-group techstore-rg
```

First, creates the necessary Azure resources such as the resource group, app service plan, and the web app. By specifying the app name, location, and git repository URL, this option ensures that the Azure infrastructure is properly set up for the e-commerce application. Next, option

```bash
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --continuous-deployment
```

configures the continuous deployment from the GitHub repository to the web app. When both options are run in sequence, the e-commerce application is automatically deployed to the Azure web app once the necessary Azure resources are created. This automation reduces the need for manual intervention and enables faster deployment times, ultimately leading to a better developer experience.

which are the correct choices for creating an Azure web app and configuring its deployment from the provided GitHub repository. Both options set the necessary variables and include the correct sequence of commands to accomplish the task.

Feedback (if wrong):

```bash
appName="TechStore2024"
location="EastUS"
gitRepo="https://github.com/techtrends/webapp"

az group create --location $location --name techstore-rg
az appservice plan create --name techstore-asp --location $location --resource-group techstore-rg --sku FREE
az webapp create --name $appName --plan techstore-asp --resource-group techstore-rg
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --manual-integration
git clone $gitRepo
```

While this option sets up the Azure resources and deploys the code from the GitHub repository, it uses the `--manual-integration` flag, which is not necessary since the question states that the application should be automatically deployed. Additionally, the `git clone` command is not necessary in this case.

```bash
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --triggers-include linux
```

This option tries to include a platform trigger (Linux) that is not necessary for this scenario. It is not necessary to specify a platform trigger when configuring the deployment source for a web app in Azure.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Automatic Deployment Setup, Repository Configuration

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

16. You are tasked with automating the deployment of an innovative e-commerce application for a fictitious company named "TechTrends." The company's web app, called "TechStore," needs to be deployed to Azure using a CLI script. The script utilizes the following variables:

- Git Repository (TechTrendsRepo): <https://github.com/techtrends/webapp>
- Web App Name (TechStoreWebApp): TechStore2024

You must ensure that the application is automatically deployed from the provided GitHub repository to the newly created web app in Azure. Arrange the Azure CLI commands to accomplish this task. Select the appropriate options in the answer area.

A.

```bash
appName="TechStore2024"

location="EastUS"

gitRepo="https://github.com/techtrends/webapp"


az group create --location $location --name techstore-rg

az appservice plan create --name techstore-asp --location $location --resource-group techstore-rg --sku FREE

az webapp create --name $appName --plan techstore-asp --resource-group techstore-rg
```

B.

```bash
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --continuous-deployment
```

C.

```bash
appName="TechStore2024"
location="EastUS"
gitRepo="https://github.com/techtrends/webapp"

az group create --location $location --name techstore-rg
az appservice plan create --name techstore-asp --location $location --resource-group techstore-rg --sku FREE
az webapp create --name $appName --plan techstore-asp --resource-group techstore-rg
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --manual-integration
git clone $gitRepo
```

D.

```bash
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --triggers-include linux
```

Answer: A, B

Feedback (if correct): options ```bash

appName="TechStore2024"

location="EastUS"

gitRepo="https://github.com/techtrends/webapp"

az group create --location $location --name techstore-rg

az appservice plan create --name techstore-asp --location $location --resource-group techstore-rg --sku FREE

```bash
az webapp create --name $appName --plan techstore-asp --resource-group techstore-rg
```

First, create the necessary Azure resources such as the resource group, app service plan, and the web app. By specifying the app name, location, and git repository URL, this option ensures that the Azure infrastructure is properly set up for the e-commerce application. Next, option

```bash
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --continuous-deployment
```

configures the continuous deployment from the GitHub repository to the web app. When both options are run in sequence, the e-commerce application is automatically deployed to the Azure web app once the necessary Azure resources are created. This automation reduces the need for manual intervention and enables faster deployment times, ultimately leading to a better developer experience.

which are the correct choices for creating an Azure web app and configuring its deployment from the provided GitHub repository. Both options set the necessary variables and include the correct sequence of commands to accomplish the task.

Feedback (if wrong):

```bash
appName="TechStore2024"
location="EastUS"
gitRepo="https://github.com/techtrends/webapp"

az group create --location $location --name techstore-rg
az appservice plan create --name techstore-asp --location $location --resource-group techstore-rg --sku FREE
az webapp create --name $appName --plan techstore-asp --resource-group techstore-rg
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --manual-integration
git clone $gitRepo
```

While this option sets up the Azure resources and deploys the code from the GitHub repository, it uses the `--manual-integration` flag, which is not necessary since the question states that the application should be automatically deployed. Additionally, the `git clone` command is not necessary in this case.

```bash
az webapp deployment source config --name $appName --repo-url $gitRepo --branch master --triggers-include linux
```

This option tries to include a platform trigger (Linux) that is not necessary for this scenario. It is not necessary to specify a platform trigger when configuring the deployment source for a web app in Azure.

Skill Mapping:

Skills: Azure Developer Associate Exam (AZ-204)

Subskills: Develop Azure compute solutions

Competencies: Sub-Topics:. Create and manage Azure App Service Web Apps, Configure deployment sources for Azure App Service

Difficulty Level: Medium

Bloom's Taxonomy Level: Applying

17. As part of building a data storage solution for a popular new social media app, you're responsible for writing code to store user information efficiently in Azure Table Storage. Since the app already handles millions of requests per day, it's important to optimize database writes whenever possible. Given the existing code snippet, how should you modify it to insert multiple sets of user information at once?

A. Use `TableOperation` with `InsertOrReplace` method.

B. Implement retry logic for handling transient errors during bulk insertions.

C. Utilize `TableBatchOperation` class with `Insert` methods.

D. Store related entities together within a transaction scope.

Correct Answer: C

Feedback (if correct):

When working with Azure Table Storage, using the `TableBatchOperation` class allows us to group several `Insert`, `Update`, or `Delete` operations into a single request sent to the server. By doing so, we reduce the overall network traffic and increase throughput compared to sending separate HTTP requests for every single record update. Therefore, the best approach to handle multiple sets of user information insertion is to utilize the `TableBatchOperation` class with `Insert` methods.

Feedback (if wrong):

A. Using `TableOperation` with `InsertOrReplace` method replaces an existing row rather than inserting multiple rows simultaneously.

B. Retry logic is useful for handling transient errors during bulk insertions, but it doesn't help optimize database writes.

D. Storing related entities together within a transaction scope is valuable for atomicity and consistency, but it isn't aimed at increasing write performance.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop for Azure storage

Competencies: Bulk Insert Operations in Azure Table Storage

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

18. You are developing a data storage solution for a social networking app, and the architecture involves utilizing Azure Table Storage to store user information. The task is to develop code that can efficiently insert multiple sets of user information into the table. You need to complete the code for inserting multiple sets of user information into Azure Table Storage. Choose the appropriate options.

```csharp
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
   CloudConfigurationManager.GetSetting("StorageConnectionString"));
CloudTableClient tableClient = storageAccount.CreateCloudTableClient();
CloudTable table = tableClient.GetTableReference("clients");
Table.CreateIfNotExists();

// Choose the correct code to create a batch operation and insert multiple sets of user information.
A. TableBatchOperation batchOperation = new TableBatchOperation();
   batchOperation.InsertOrMerge(new TableEntity());
   table.ExecuteBatch(batchOperation);

B. TableOperation tableOperation = TableOperation.Insert(new TableEntity());
   table.Execute(tableOperation);

C. TableBatchOperation batchOperation = new TableBatchOperation();
   batchOperation.Insert(new TableEntity());
   table.ExecuteBatch(batchOperation);
```

D. TableQuery<TableEntity> query = new TableQuery<TableEntity>();

table.ExecuteQuery(query);

Options:

A. TableBatchOperation batchOperation = new TableBatchOperation();

batchOperation.InsertOrMerge(new TableEntity());

table.ExecuteBatch(batchOperation);

B. TableOperation tableOperation = TableOperation.Insert(new TableEntity());

table.Execute(tableOperation);

C. TableBatchOperation batchOperation = new TableBatchOperation();

batchOperation.Insert(new TableEntity());

table.ExecuteBatch(batchOperation);

D. TableQuery<TableEntity> query = new TableQuery<TableEntity>();

table.ExecuteQuery(query);

Answer: C

Feedback(if correct):-

```csharp
TableBatchOperation batchOperation = new TableBatchOperation();
batchOperation.Insert(new TableEntity());
table.ExecuteBatch(batchOperation);
```

Option C (TableBatchOperation with Insert): This is the correct choice for creating a batch operation to insert multiple sets of user information efficiently into Azure Table Storage.

Feedback(if wrong):- Option A (TableBatchOperation with InsertOrMerge): While `InsertOrMerge` can be useful for inserting or updating entities, it is not efficient for inserting multiple sets of user information in a batch operation.

Option B (TableOperation with Insert): This option inserts a single entity into the table. It is not suitable for inserting multiple sets of user information.

Option D (TableQuery): This option is for querying entities from the table, not for inserting data. Therefore, it is not suitable for the given task.

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop for Azure storage

Competencies: Azure Storage

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

19. Your task is to complete the code for storing user information in Azure Table Storage for a social networking app. Which steps are required to execute a batch of insert operations?

A. Create a `TableBatchOperation` object and call `Execute` method.

B. Create a `TableOperation` object and call `InsertOrMerge` method.

C. Create a `TableEntity` object and call `Insert` method.

D. Create a `TableQuery` object and call `Yield` method.

Answer: A

Feedback(if correct):- Create a TableBatchOperation object and call the Execute method.

Here's a breakdown of the steps involved in executing a batch of insert operations in Azure Table Storage:

Create a TableBatchOperation object:

This object represents a collection of operations to be performed on the table in a single batch.

Add Insert operations to the batch:

For each user entity you want to insert, create a TableOperation object with the Insert operation type.

Add each TableOperation object to the TableBatchOperation using its Add method.

Call the Execute method:

Execute the batch of operations by calling the Execute method on the TableBatchOperation object. This sends all insert operations to the Azure Table Storage service in a single request, optimizing performance.

Feedback(if wrong):- Create a `TableOperation` object and call `InsertOrMerge` method: This is used for individual insert or merge operations, not batches. Create a `TableEntity` object and call `Insert` method: This performs a single insert operation, not a batch. Create a `TableQuery` object and call `Yield` method: This is used for querying data, not inserting it.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop for Azure storage

Competencies: Table storage

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

20. An e-commerce website uses an Azure storage account with soft delete enabled. A data migration error has accidentally deleted a product image blob named "img.jpg" along with its two existing snapshots. The website administrators are familiar with various Azure tools and services. Which of the following methods can they use to recover the deleted blob and its snapshots?

A. Restore from the Azure portal
B. Use Azure PowerShell or Azure CLI
C. Use Azure Storage Explorer
D. All of the above

Answer: D

Feedback(if correct):- All of the above options are valid methods for recovering soft-deleted blobs and snapshots in Azure Blob Storage.

This flexibility allows administrators to choose the approach that best suits their preferences, skills, and specific recovery needs.

Feedback(if wrong):-

While each of these options is individually correct, they do not encompass the full range of available recovery methods.

Choosing "All of the above" demonstrates a comprehensive understanding of the various tools and services that can be used for data recovery in Azure Blob Storage.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop for Azure storage

Competencies: Blob storage

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Comprehension

21. An e-commerce website uses an Azure storage account with soft delete enabled. The e-commerce website administrators attempt to recover the deleted blob "img.jpg" and its two snapshots from the Azure Storage account using the Azure portal. However, the recovery process fails with an error message. Which of the following actions would be the MOST effective in troubleshooting the recovery failure and identifying the cause of the error?

A. Increase the soft delete retention period in the storage account.

B. Check the Storage Account metrics for any recent errors or throttling events.

C. Rebuild the data migration script to avoid future accidental deletions.

D. Contact Azure Support for immediate assistance resolving the error.

Answer: B

Feedback(if correct):-

Check the Storage Account metrics for any recent errors or throttling events that directly address the immediate problem of a failed recovery attempt. Checking the Storage Account metrics provides insight into potential technical issues causing the failure, such as:

- Storage Account errors: The metrics might reveal specific error codes related to blob restoration, exceeding quota limits, or service interruptions.
- Throttling events: If the retrieval request exceeds the allocated throughput for the storage account, throttling mechanisms may restrict access and cause failures.

Therefore, by analyzing the Storage Account metrics first, the administrators can identify the root cause of the failure and take targeted actions to resolve it or escalate with informed details, significantly improving the likelihood of a successful recovery.

Feedback(if wrong):-

Increasing retention while important for future protection, doesn't solve the current recovery issue. Script modification is useful for preventing future accidents, but not relevant to the immediate failure. Contacting support, While an option, understanding the issue through metrics offers valuable context before escalation.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop for Azure storage

Competencies: Understanding soft delete functionality, Troubleshooting storage account issues

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

22. You are developing a SaaS application for media management. Users upload videos to a web service, and these videos are stored in Azure Storage Blob storage using a General-purpose V2 storage account. The application requires the initiation of a video processing workflow within one minute of video upload to generate a mobile-friendly version. What should you do to meet this requirement?

A. Convert the Azure Storage account to a BlockBlobStorage storage account.

B. Enable soft delete for the Azure Storage account.

C. Increase the storage account's replication factor.

D. Implement Azure CDN to cache the uploaded videos.

Answer: D

Feedback(if correct):

Azure CDN (Content Delivery Network) can improve the performance and responsiveness of your application by caching the uploaded videos closer to the end-users. By caching the videos on edge servers located geographically closer to the users, the mobile-friendly versions can be delivered quickly, reducing the processing time.

Feedback(if wrong):

Converting the Azure Storage account to a BlockBlobStorage storage account does not directly address the requirement of initiating the video processing workflow within one minute. Enabling soft delete for the Azure Storage account is not directly related to the requirement of initiating the video processing workflow within one minute. Increasing the storage account's replication factor may improve data redundancy and availability but does not directly address the requirement of initiating the video processing workflow within one minute.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Develop for Azure storage

Competencies: Understanding different storage account types and their capabilities, Understanding Azure CDN and its benefits

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

23. In the process of developing a web application on Azure, you must register it with Azure Active Directory (Azure AD) for secure authentication. Arrange the following actions in the correct order:

Actions:

1. Provide a name, account type, and redirect URL for the new application.

2. Sign in to the Azure portal using your work or school account. Select the appropriate Azure AD instance linked to your project.

3. In "App Registrations," choose "New registration." Select the middle-tier service registration and add a cryptographic key.

4. Navigate to "Enterprise Applications" and click on "New application."

Select the correct sequence of actions to register the web application on Azure Active Directory for secure authentication.

A) 1, 2, 3, 4

B) 2, 1, 3, 4

C) 1, 3, 2, 4

D) 3, 2, 1, 4

Answer: B

Feedback(if correct): Registering a new application in Azure AD involves signing in, selecting the AD instance, creating a new application with details, and then navigating to the Enterprise Applications for further configuration.

Feedback(if wrong):

A) Incorrect. The sequence starts with signing in and selecting the AD instance.

C) Incorrect. The correct sequence places the creation of the new application before navigating to "Enterprise Applications."

D) Incorrect. The correct sequence involves creating a new application first.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Implement Azure security

Competencies: Azure Active Directory: Application registration for secure authentication

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Synthesis

24. You are developing a cloud-based application that uses Azure Active Directory (Azure AD) for authentication. The application is designed to provide users with personalized content based on their roles within the organization. The application has the following requirements:

- Azure AD users must be able to log in to the application.

- Personalization of the application must be based on the user's role in the organization.

- The application must be able to access the user's role information from Azure AD.

You need to configure the application's manifest to meet these requirements.

How should you configure the manifest? Choose the best solution from the options below:

A. Set the `groupMembershipClaims` to "Role" and `oauth2Permissions` to include the necessary scopes for user login.

B. Set the `groupMembershipClaims` to "All" and `oauth2Permissions` to include the necessary scopes for user login.

C. Set the `groupMembershipClaims` to "Role" and `oauth2AllowImplicitFlow` to true.

D. Set the `groupMembershipClaims` to "All" and `oauth2AllowImplicitFlow` to true.

Answer: A

Feedback(if correct):

Option B (Set `groupMembershipClaims` to "All"): This could expose unnecessary information and is not aligned with the principle of least privilege.

Option C (Set `groupMembershipClaims` to "Role" and `oauth2AllowImplicitFlow` to true): The use of implicit flow is discouraged for security reasons.

Option D (Set `groupMembershipClaims` to "All" and `oauth2AllowImplicitFlow` to true): Implicit flow is generally not recommended.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Implement Azure security

Competency: Authentication and authorization

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

25. You develop a cloud-based application that uses Azure Active Directory (Azure AD) for authentication. The application needs to ensure that user authentication is secure and follows best practices. Additionally, the application should allow users to reset their passwords securely.

A. Multi-Factor Authentication (MFA)

B. Self-Service Password Reset (SSPR)

C. Azure AD B2C

D. Conditional Access Policies

Answer: B

Feedback(if correct):

Self-Service Password Reset (SSPR) allows users to securely reset their passwords without the need for administrator intervention, thus enhancing security and user convenience. While Multi-Factor Authentication (MFA) and Conditional Access Policies contribute to authentication security, SSPR specifically addresses the requirement for users to reset their passwords securely within the application. Azure AD B2C, on the other hand, is designed for business-to-consumer identity management and may not directly fulfill the requirement for secure password resets within the described scenario.

Feedback(if wrong):

A. Multi-Factor Authentication (MFA): While MFA enhances authentication security, it doesn't directly address the requirement for users to securely reset their passwords within the application.

C. Azure AD B2C: Azure AD B2C is designed for business-to-consumer identity management and may not directly support secure password resets within a cloud-based application.

D. Conditional Access Policies: While Conditional Access Policies can enforce security requirements based on specific conditions, they may not directly facilitate secure password resets for users within the application.

Skills: Azure Developer Certification (AZ-204)

Subskills: Implement Azure security

Competencies: Azure Security: Authentication and authorization, Azure Active Directory

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

26. You are working on a solution involving multiple microservices deployed on an Azure Kubernetes Service (AKS) cluster, leveraging Azure Functions, Cosmos DB, and Event Hub. All resources reside within the same subscription and region. The client insists on controlling the access to the AKS cluster and underlying resources through Azure Policy. They request the ability to deny unwanted updates on specific Azure Function app settings and impose naming conventions on Cosmos DB databases. Address the client's demands by selecting the optimum pair of actions:

A) Assign 'Deny' effect to Azure Policy definition governing undesired Function app settings updates; enforce consistent naming convention on Cosmos DB databases via Azure Policy assignment

B) Employ Azure Blueprints to construct a predefined template comprising the requested policy restrictions; delegate permission to modify the AKS cluster scope to the client

C) Leverage Azure Policy Add-on for Kubernetes to link Azure Policy to the AKS cluster's GateKeeper admission controller; combine conditional tags for Cosmos DB databases and Function apps to reflect client preferences

D) Activate Azure Arc for Servers on the AKS cluster, followed by assigning custom roles defining limitations on Function app settings and Cosmos DB database manipulation

Answer: A

Feedback(if correct):

By assigning 'Deny' effects to Azure Policy definitions for Function app settings and enforcing naming conventions on Cosmos DB databases, you address the client's demands effectively.

Feedback(if wrong):

Azure Blueprints are not specifically designed for managing AKS cluster resources and policies. The Azure Policy Add-on for Kubernetes is suitable for AKS but does not directly address Cosmos DB naming conventions. Azure Arc for Servers is not the primary tool for managing AKS policies and Cosmos DB configurations in this context.

Skills: Azure Developer Certification (AZ-204)

Subskills: Implement Azure security

Competencies: Azure Security: Access control, Azure Policy

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

27. You are developing a web application that will be hosted on Azure Web App and will require users to authenticate using their Azure AD credentials. The application has three permission levels: Admin, Normal, and Reader. You aim to assign permission levels based on the user's Azure AD group membership. Goal: Configure authorization for the Azure Web App to allow only authenticated requests, requiring Azure AD login, and assign appropriate permission levels based on Azure AD group membership.

Statement:

To meet the goal, you propose the following solution:

1. Set the `authentication` mode to `On` in the Azure Web App configuration.
2. Register an Azure AD application representing the web application.
3. Define allowed Azure AD groups corresponding to each permission level in the Azure AD application's manifest.
4. Update the Azure AD application's manifest, setting the value of the `groupMembershipClaims` option to `"All"`.

Question: Will this solution meet the stated goal?

A) Yes

B) No

Answer: A

Feedback(if correct): Yes, this solution will meet the stated goal. By enabling authentication in the Azure Web App, requiring Azure AD login, and setting the `groupMembershipClaims` option to `"All"`, the application obtains the user's Azure AD group membership details. The Azure AD groups assigned to each permission level can then be evaluated, and appropriate permission levels granted accordingly.

Feedback(if wrong):- An answer of "No" suggests that the proposed solution does not meet the stated goals. Specifically, in this question, the goal is to configure authorization for the Azure Web App to allow only authenticated requests and require Azure AD log-on.

However, simply setting the `groupMembershipClaims` option to `"All"` in the Azure AD application's manifest does not satisfy the stated goal. This change only modifies the claims returned by Azure AD, indicating whether a user is a member of any group. It does not enforce authentication or authorization on the Azure Web App itself. Therefore, the proposed solution cannot be considered a valid solution to the stated goal.

It is important to note that configuring authentication and authorization for an Azure Web App requires multiple steps beyond just updating the Azure AD application's manifest. Other steps, such as enabling authentication in the Azure Web App, configuring the Azure AD tenant, and assigning appropriate role-based access control (RBAC) roles, are also necessary to meet the stated goal.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Implement Azure security

Competency: Configure authentication and authorization for Azure resources

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

28. To increase the security of an employee-only internal website displaying confidential data, you are tasked with implementing Multi-Factor Authentication (MFA) for the site. Currently, Azure Active Directory (Azure AD) handles the authentication process. What steps should you follow to enable MFA for the website?

A. Create a Conditional Access Policy in Azure AD.

B. Enable Azure AD Application Proxy.

C. Switch to using Azure AD Business-to-Consumer (B2C).

D. Upgrade your Azure AD license to Premium P1 or P2.

Answer A.D

Feedback (if correct): By creating a Conditional Access Policy in Azure AD and upgrading to Azure AD Premium, you've ensured that stronger authentication mechanisms are in place, protecting sensitive data displayed on the website. Keep in mind that careful planning and consideration are needed to balance security and productivity.

Feedback (if incorrect): To enable Multi-Factor Authentication (MFA) for the website, you need to create a Conditional Access Policy in Azure AD and upgrade your Azure AD license to Premium P1 or P2. These features fortify authentication and protect sensitive data shown on the website. Ensure that future designs strike a balance between security and productivity.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Implement Azure security

Competencies: Azure Security: Authentication, Multi-Factor Authentication (MFA)

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

29. Envision yourself responsible for augmenting an Azure Functions implementation catering to a confidential health sector organization, managing delicate patient records and mandating protected exchange with auxiliary entities. In pursuit of securing outgoing communication between Azure Functions and external APIs employing digital certificates, undertake the subsequent measures:

Available Options:

A. Orchestrate a self-signed certificate in Azure Key Vault.

B. Instantiate Azure Functions to operate with client certificates for egress traffic.

C. Deploy Azure API Management as an intermediary governing interaction with external systems.

D. Leverage Azure Managed Identities to substantiate Azure Functions vis-à-vis external APIs.

Answers: A,B

Feedback (if correct):

When configuring dynamic data masking for your Azure SQL Database, you need to identify the sensitive columns (A) containing sensitive user information and the schema (B) that houses the sensitive tables and columns. This ensures that the sensitive data is properly protected.

Feedback (if wrong):

For option C (Table), selecting tables is not sufficient to enable dynamic data masking, as masking applies to individual columns, not entire tables. Option D (Login Account) is irrelevant to dynamic data masking, as it deals with authentication, not securing sensitive data.

Skill Mapping:

- Skill: Azure Developer Certification (AZ-204)

- Subskill: Implement Azure security

- Competency: Azure SQL Database security features

- Difficulty Level: Intermediate

- Bloom's Taxonomy Level: Application

30. You're developing a microservices-based architecture for a SaaS company, where each web service must adhere to specific security requirements:

1. Leverage API Management for service access.

2. Implement OpenID Connect for authentication.

3. Prevent unauthorized and anonymous usage.

Following a security audit, it's found that certain web services lack proper authentication, allowing anonymous access. Which API Management policy should be enforced to rectify this?

A. rate-limit

B. enforce-ssl

C. jwt-bearer

D. validate-jwt

Answer: D

Feedback(if correct): Apply the validate-jwt policy to authenticate and validate the OAuth token for each incoming request, ensuring proper authentication is enforced.

Feedback(if wrong):

(rate-limit) is focused on controlling the number of requests and does not directly address authentication.

(enforce-ssl) is related to enforcing secure connections but doesn't handle authentication concerns.

(jwt-bearer) involves JSON Web Token (JWT) bearer authentication but may not address anonymous access.Description: Evaluate Azure API Management knowledge. Difficulty Level: Moderate. Completion Time: 1-2 minutes.

SaaS microservices need secure API access. Identify Azure API Management policy for OAuth token validation to prevent unauthorized and anonymous access.

Skill Mapping:

- Skill: Azure Developer Certification (AZ-204)

- Subskill: Implement Azure security

- Competency: Azure API Management

- Bloom's Taxonomy Level: Application

31. One of the web services offered by your Software as a Service (SaaS) organization allows unauthorized access, risking sensitive data exposure. The service must abide by stringent authentication and authorization protocols, mandating the use of OpenID Connect and preventing anonymous access. To rectify the current lapse, you must employ an API Management policy. Which policy should you apply to safeguard the web service?

A. Implement JSONP (JSON with padding) policy to pad responses with callback functions

B. Apply authentication-certificate policy to verify clients presenting digital certificates

C. Introduce check-header policy to inspect headers and block unverified requests

D. Integrate validate-jwt policy to confirm JWT tokens for every incoming request

Answer: D

Feedback (if correct): Congratulations, your response is spot on! By adding the validate-jwt policy, you can thoroughly scrutinize the OAuth token carried within each incoming request, ensuring proper authentication and authorization.

Feedback (if incorrect): Close, but not exact. The correct answer lies in incorporating the validate-jwt policy. Doing so verifies the JWT token accompanying each request, curbing unauthorized access and heightening security.

Skills: Azure Developer Certification (AZ-204)

Subskills: Implement Azure security

Competencies: Azure Security: API Management, Authentication, Authorization

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Application

32. Recently, a security assessment revealed that a few web services in your Software as a Service (SaaS) firm could be invoked without proper authentication. Complying with corporate regulations, the web services ought to use API Management for access, embrace OpenID Connect for authentication, and prohibit anonymous usage. To tackle the discovered vulnerabilities, you must incorporate an API Management directive. Which policy should you apply to shield the susceptible web services?

A) Implement the jsonp (JSON with padding) policy to embellish responses with callback functions

B) Instate the authentication-certificate policy to corroborate clients presenting digital certificates

C) Invoke the check-header policy to scrutinize headers and reject unconfirmed solicitations

D) Merge the validate-jwt policy to authenticate JWT tokens coupled with every incoming petition

Answer: D

Feedback (if correct): The correct answer is D) Merge the validate-jwt policy to authenticate JWT tokens coupled with every incoming petition.

D) validate-jwt policy: This policy is specifically designed to validate JSON Web Tokens (JWTs), which are a secure way to implement OpenID Connect authentication. This aligns with the requirements to use OpenID Connect and prohibit anonymous usage.

Feedback (if incorrect):

A) jsonp policy: This policy is used for cross-origin resource sharing (CORS) and doesn't address authentication directly.

B) authentication-certificate policy: This policy validates client certificates, which might be suitable for certain scenarios but would not prevent anonymous access.

C) check-header policy: This policy can be used to verify specific headers, but it's not sufficient for robust authentication in this case.

Skill Mapping:

Skills: Azure Developer Certification (AZ-204)

Subskills: Implement Azure security

Competencies: Azure Security: API Management, Authentication, Authorization

Difficulty Level: Intermediate

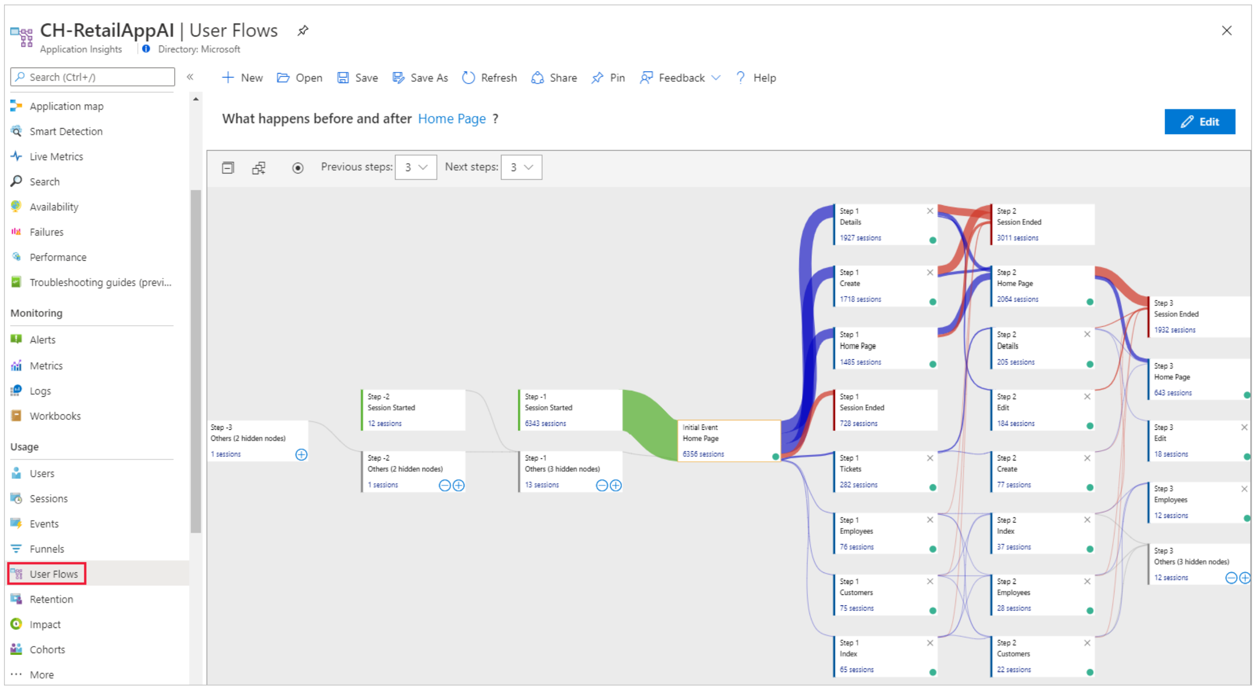
Bloom's Taxonomy Level: Application

33. You are developing an ASP.NET Core MVC application, and you have configured the application to track webpages and custom events. Your goal is to identify trends in application usage for various scenarios. Choose the most appropriate Azure Application Insights feature for each given requirement. Which Azure Application Insights feature should you use to identify pages visited by users most often that correlate to a product purchase?

A. Impact
B. Retention
C. User Flows
D. Users

Answer: C.

Feedback (if correct): User Flows display how users navigate between pages and features of your application. It helps identify frequent user journeys, especially those leading to product purchases.


CH-RetailAppAI | User Flows
Application Insights
Directory: Microsoft
Search (Ctrl+/)
New
Open
Save
Save As
Refresh
Share
Pin
Feedback
Help
Application map
Smart Detection
Live Metrics
Search
Availability
Failures
Performance
Troubleshooting guides (previ...
Monitoring
Alerts
Metrics
Logs
Workbooks
Usage
Users
Sessions
Events
Funnels
User Flows
Retention
Impact
Cohorts
More
What happens before and after Home Page ?
Edit
Previous steps: 3
Next steps: 3
Step -3 Others (2 hidden nodes) 1 sessions
Step -2 Session Started 12 sessions
Step -2 Others (2 hidden nodes) 1 sessions
Step -1 Session Started 6343 sessions
Step -1 Others (3 hidden nodes) 13 sessions
Initial Event Home Page 6356 sessions
Step 1 Details 1927 sessions
Step 1 Create 1718 sessions
Step 1 Home Page 1485 sessions
Step 1 Session Ended 728 sessions
Step 1 Tickets 282 sessions
Step 1 Employees 76 sessions
Step 1 Customers 75 sessions
Step 1 Index 65 sessions
Step 2 Session Ended 3011 sessions
Step 2 Home Page 2064 sessions
Step 2 Details 205 sessions
Step 2 Edit 184 sessions
Step 2 Create 77 sessions
Step 2 Index 37 sessions
Step 2 Employees 28 sessions
Step 2 Customers 22 sessions
Step 3 Session Ended 1932 sessions
Step 3 Home Page 643 sessions
Step 3 Edit 18 sessions
Step 3 Employees 12 sessions
Step 3 Others (3 hidden nodes) 12 sessions

Feedback (if wrong): Impact analyzes how slowdowns in specific application aspects affect user behavior, while Retention reveals returning user rates. Users present analytics on active sessions and demographics. None of these provide insights on navigational trends correlated to product purchases.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Monitor, optimize, and troubleshoot Azure solutions

Competency: Application Insights

Difficulty Level: Intermediate

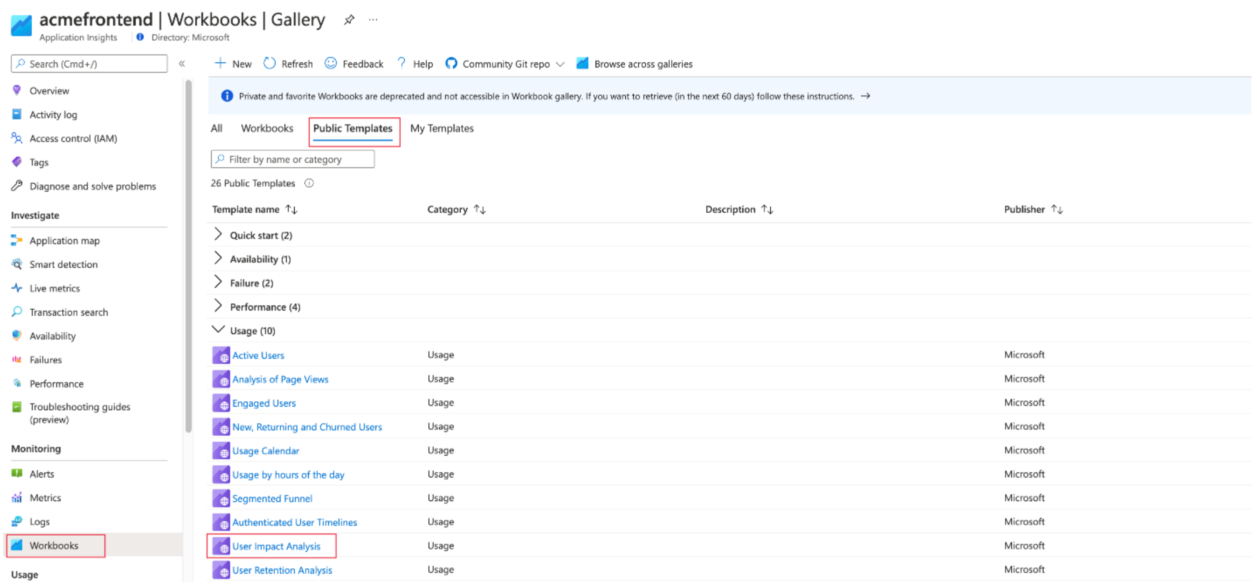
Bloom's Taxonomy Level: Analysis

34. You are developing an ASP.NET Core MVC application, and you have configured the application to track webpages and custom events. Your goal is to identify trends in application usage for various scenarios. Choose the most appropriate Azure Application Insights feature for each given requirement. To analyze how the load time of the product display page affects a user's decision to purchase a product, which Azure Application Insights feature is most suitable?

A. Impact
B. Retention
C. User Flows
D. Users

Answer: A.

Feedback (if correct): The "Impact" feature in Azure Application Insights is designed to provide insights into how specific aspects of the site, such as slowness in loading pages, affect user behavior and conversion. It helps in understanding the impact of performance on user engagement, making it a suitable choice for analyzing how the load time of the product display page influences a user's decision to make a purchase. The "Impact" feature provides valuable information to balance optimization and performance for maximizing user conversion.



acmefrontend | Workbooks | Gallery
Application Insights | Directory: Microsoft

Search (Cmd+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Investigate
Application map
Smart detection
Live metrics
Transaction search
Availability
Failures
Performance
Troubleshooting guides (preview)
Monitoring
Alerts
Metrics
Logs
Workbooks
Usage

New | Refresh | Feedback | Help | Community Git repo | Browse across galleries

Private and favorite Workbooks are deprecated and not accessible in Workbook gallery. If you want to retrieve (in the next 60 days) follow these instructions. →

All | Workbooks | Public Templates | My Templates

Filter by name or category

26 Public Templates

Template name	Category	Description	Publisher
Quick start (2)			
Availability (1)			
Failure (2)			
Performance (4)			
Usage (10)			
Active Users	Usage		Microsoft
Analysis of Page Views	Usage		Microsoft
Engaged Users	Usage		Microsoft
New, Returning and Churned Users	Usage		Microsoft
Usage Calendar	Usage		Microsoft
Usage by hours of the day	Usage		Microsoft
Segmented Funnel	Usage		Microsoft
Authenticated User Timelines	Usage		Microsoft
User Impact Analysis	Usage		Microsoft
User Retention Analysis	Usage		Microsoft

Feedback (if wrong): Retention monitors user returns, User Flows traces navigation flow, and Users lists active users and session statistics. These do not relate to performance impacts on user decisions.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Monitor, optimize, and troubleshoot Azure solutions

Competency: Application Insights

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Analysis

35. You are developing an ASP.NET Core MVC application, and you have configured the application to track webpages and custom events. Your goal is to identify trends in application usage for various scenarios. Choose the most appropriate Azure Application Insights feature for each given requirement. For understanding the

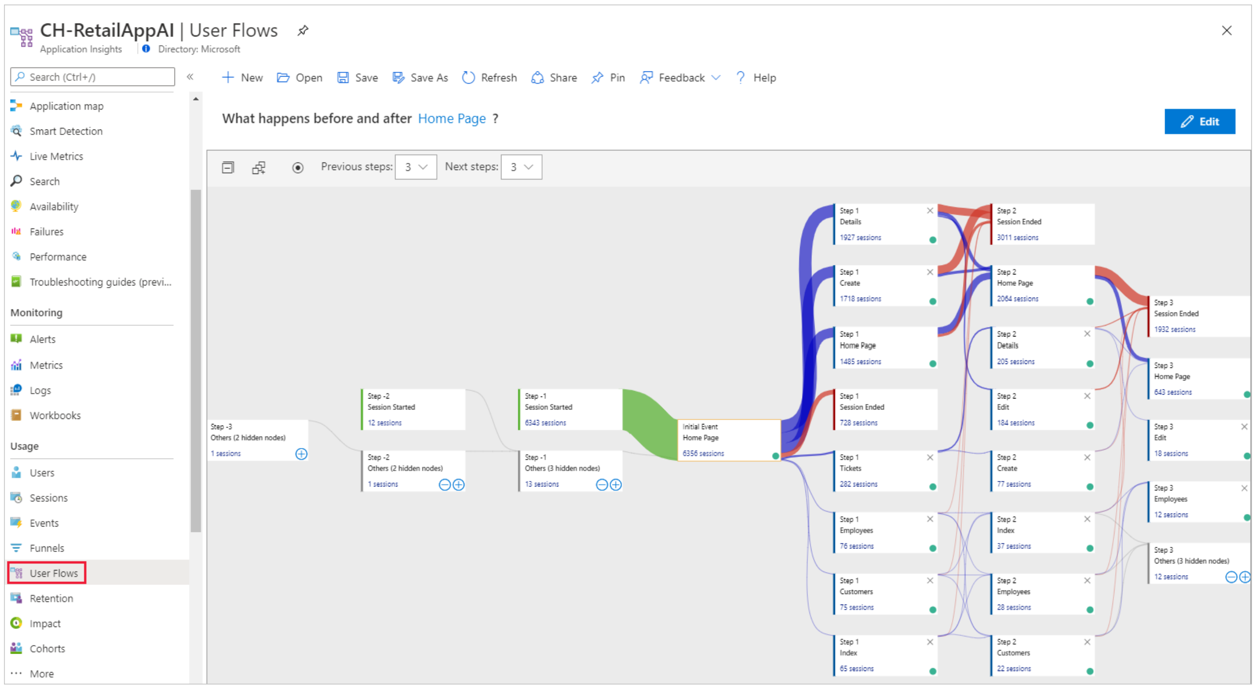
events that most influence a user’s decision to continue using the application, which Azure Application Insights feature should be utilized?

A. Impact
B. Retention
C. User Flows
D. Users

Answer: C

Feedback (if correct): For understanding the events that most influence a user’s decision to continue using the application in the given ASP.NET Core MVC scenario, the most appropriate Azure Application Insights feature is. User Flows

The User Flows feature in Azure Application Insights visualizes how users navigate between the pages and features of your site. It provides insights into how users interact with the application, helping you understand the paths they take and identify areas that might influence their decision to continue using the application. User Flows are valuable for analyzing user behavior and optimizing the user experience by identifying specific events or actions that contribute to user engagement.


CH-RetailAppAI | User Flows
Application Insights
Directory: Microsoft
Search (Ctrl+/)
New
Open
Save
Save As
Refresh
Share
Pin
Feedback
Help
Application map
Smart Detection
Live Metrics
Search
Availability
Failures
Performance
Troubleshooting guides (previ...
Monitoring
Alerts
Metrics
Logs
Workbooks
Usage
Users
Sessions
Events
Funnels
User Flows
Retention
Impact
Cohorts
More
What happens before and after Home Page ?
Edit
Previous steps: 3
Next steps: 3
Step -3 Others (2 hidden nodes) 1 sessions
Step -2 Session Started 12 sessions
Step -2 Others (2 hidden nodes) 1 sessions
Step -1 Session Started 6343 sessions
Step -1 Others (3 hidden nodes) 13 sessions
Initial Event Home Page 6356 sessions
Step 1 Details 1927 sessions
Step 1 Create 1718 sessions
Step 1 Home Page 1485 sessions
Step 1 Session Ended 728 sessions
Step 1 Tickets 282 sessions
Step 1 Employees 76 sessions
Step 1 Customers 75 sessions
Step 1 Index 65 sessions
Step 2 Session Ended 3011 sessions
Step 2 Home Page 2064 sessions
Step 2 Details 205 sessions
Step 2 Edit 184 sessions
Step 2 Create 77 sessions
Step 2 Index 37 sessions
Step 2 Employees 28 sessions
Step 2 Customers 22 sessions
Step 3 Session Ended 1932 sessions
Step 3 Home Page 643 sessions
Step 3 Edit 18 sessions
Step 3 Employees 12 sessions
Step 3 Others (3 hidden nodes) 12 sessions

Feedback (if wrong): Retention indicates user recurrence and loyalty, User Flows charts user navigation patterns, and Users enumerates active users and sessions. None of these provide insights into decisive events prompting users to stay engaged.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Monitor, optimize, and troubleshoot Azure solutions

Competency: Application Insights

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Analysis

36. You are developing an ASP.NET Core MVC application, and you have configured the application to track webpages and custom events. Your goal is to identify trends in application usage for various scenarios. Choose the most appropriate Azure Application Insights feature for each given requirement. To identify places in the application where users often perform repetitive actions, which Azure Application Insights feature provides the necessary visualization?

A. Impact
B. Retention
C. User Flows
D. Users

Answer: C

Feedback (if correct): User Flows in Azure Application Insights deliver a visual representation of user navigation patterns, revealing spots in the application where users tend to perform repetitive actions.

Feedback (if wrong): Impact inspects the causality between performance indicators and user behaviors, Retention gauges user recurrence, and Users count active users and sessions. None of these directly indicate repetitive actions.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Monitor, optimize, and troubleshoot Azure solutions

Competency: Application Insights

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Analysis

37. You are tasked with developing a cloud-native application on Azure that integrates with various third-party services. The application must maintain comprehensive telemetry and dependency tracking, including calls made to a third-party database. As part of your solution design, you decide to utilize Azure Application Insights

for telemetry and dependency tracking. You are developing a cloud-native application on Azure that integrates with third-party services, including a database outside the Microsoft SQL Server ecosystem. The application must maintain detailed telemetry and dependency tracking. Which approach should you take to ensure that dependency tracking works for calls to the third-party database?

A) Store Telemetry.Context.Operation.Id and Telemetry.Id in the database.

B) Save Telemetry.Context.Cloud.RoleInstance and Telemetry.Context.Session.Id in the database.

C) Record Telemetry.Id and Telemetry.Name in the database.

D) Store Telemetry.Name and Telemetry.Context.Operation.Id in the database.

Answer: D

Feedback (if correct):

Option D is the correct answer. Storing Telemetry.Name and Telemetry.Context.Operation.Id in the database ensures that the necessary information for dependency tracking is captured. Telemetry.Name identifies the type of dependency being tracked, while Telemetry.Context.Operation.Id uniquely identifies the operation associated with the dependency, enabling effective tracking and analysis.

Feedback (if wrong):

Option A: Incorrect. Storing Telemetry.Context.Operation.Id alone is insufficient for comprehensive dependency tracking, and Telemetry.Id is not typically used for dependency tracking purposes.

Option B: Incorrect. Telemetry.Context.Cloud.RoleInstance is not directly relevant to dependency tracking, and Telemetry.Context.Session.Id is not commonly used for this purpose.

Option C: Incorrect. While Telemetry.Id may be useful for identifying individual telemetry items, Telemetry.Name alone does not provide sufficient context for dependency tracking in this scenario.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Monitor, optimize, and troubleshoot Azure solutions

Competency: Application Insights

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Analysis

38. As a developer working on a project requiring extensive user interaction tracking within a mobile application deployed on Azure, how would you effectively utilize Azure Monitor's Application Insights through

Azure Mobile Apps SDK to gather relevant analytics data? Identify the THREE most crucial data points to track for generating insights based on user activities using Application Insights.

A. Dependency Call

B. Page View

C. Authentication

D. Custom Event

Answers: A. B. C.

Feedback(if correct):-

A. Dependency call - Capture dependency calls to recognize bottleneck areas stemming from external API reliance.

B. Page view - Measure page views to gauge interest levels and popularity among visitors concerning particular pages.

C. Custom event - Define custom events to highlight critical user interactions demanding explicit observation and further examination.

Feedback(if wrong):-

Option D, "Custom Event," is not necessarily wrong, but it may not be one of the three most crucial data points to track for generating insights based on user activities using Application Insights in the context of a mobile application deployed on Azure.

While tracking custom events can be valuable for capturing specific user interactions or actions within the application, the three options mentioned earlier (Dependency Call, Page View, and Authentication) are typically considered more crucial for generating insights based on user activities.

It's worth noting that the choice of data points to track ultimately depends on the specific needs and objectives of your project. It is recommended to analyze your application's requirements and consult the

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

Subskill: Monitor, optimize, and troubleshoot Azure solutions

Competency: Azure Monitor, Application Insights

Difficulty Level: Intermediate

Bloom's Taxonomy Level: Analysis

39. A business is in the process of deploying an order processing system, intending to utilize an Azure Service Bus topic for order publications. The message properties include:

- Location: Indicates the shipment's region.
- CorrelationId: Functions as the priority value for the order.
- Quantity: A user-defined property specifying the order quantity.
- Audited: A user-defined property denoting the order date.

You need to implement the right filters for each of the subscriptions given above:

- LaterOrders: This subscription is reserved for future use and should not currently accept any orders.
- HighPriorityOrders: Intended for receiving all high-priority orders.
- GlobalOrders: Designed for orders not originating from the USA.
- HighOrders: Meant for receiving orders with a quantity greater than 1000.
- AllOrders: Created for auditing purposes, capturing all orders.

To implement the requirement that the subscription "LaterOrders" should not currently accept any orders and is reserved for future use, which filter type would be the most suitable?

A. `SqlFilter`
B. `CorrelationFilter`
C. `No Filter`
D. `FalseFilter`

Answer: C

Feedback(if correct):

'No Filter' is the correct choice for the "LaterOrders" subscription. This means that there are no filtering conditions applied to the subscription, allowing it to accept all messages without specific criteria. This aligns with the requirement for "LaterOrders" to not currently accept any orders and be reserved for future use.

Feedback(if wrong):

`SqlFilter` is used for filtering messages based on specific conditions expressed in a SQL-like syntax. While it could be used to create a filter that always evaluates to false. `CorrelationFilter` matches messages based on their correlation ID, which is not relevant to the "LaterOrders" requirement. `No Filter` would allow all messages to pass through, contradicting the requirement to block messages for the "LaterOrders" subscription.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

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

Competency: Configure and manage Azure Service Bus, Implement message-based communication, Develop for Azure Service Bus

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

40. A business is in the process of deploying an order processing system, intending to utilize an Azure Service Bus topic for order publications. The message properties include:

- Location: Indicates the shipment's region.
- CorrelationId: Functions as the priority value for the order.
- Quantity: A user-defined property specifying the order quantity.
- Audited: A user-defined property denoting the order date.

You need to implement the right filters for each of the subscriptions given above:

- LaterOrders: This subscription is reserved for future use and should not currently accept any orders.
- HighPriorityOrders: Intended for receiving all high-priority orders.
- GlobalOrders: Designed for orders not originating from the USA.
- HighOrders: Meant for receiving orders with a quantity greater than 1000.
- AllOrders: Created for auditing purposes, capturing all orders.

To ensure that the subscription "HighPriorityOrders" receives all high-priority orders, which filter type should be utilized?

A. `SqlFilter`
B. `CorrelationFilter`
C. `No Filter`
D. `FalseFilter`

Answer: B

Feedback(if correct):

Utilizing a `CorrelationFilter` allows the "HighPriorityOrders" subscription to receive messages based on the value of the `CorrelationId` property. This is suitable for ensuring that all high-priority orders are captured.

Here's an example of how you can define a CorrelationFilter in C#:

```
var filter = new CorrelationFilter
{
   CorrelationId = "high" // Set the condition to match the high-priority orders
};

// Create the subscription with the CorrelationFilter
await adminClient.CreateSubscriptionAsync(
   new CreateSubscriptionOptions(topicName, "HighPriorityOrders"),
   new CreateRuleOptions("HighPriorityOrders", filter)
```

);

Feedback(if wrong):

`SqlFilter` is incorrect as it involves specifying conditions, which may limit the subscription. `No Filter` is incorrect as it won't filter messages based on priority. `FalseFilter This is not an actual filter offered by Azure Service Bus, It was probably introduced to check your attention or test invalid inputs.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

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

Competency: Configure and manage Azure Service Bus, Implement message-based communication, Develop for Azure Service Bus

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

41. A business is in the process of deploying an order processing system, intending to utilize an Azure Service Bus topic for order publications. The message properties include:

- Location: Indicates the shipment's region.
- CorrelationId: Functions as the priority value for the order.
- Quantity: A user-defined property specifying the order quantity.
- Audited: A user-defined property denoting the order date.

You need to implement the right filters for each of the subscriptions given above:

- LaterOrders: This subscription is reserved for future use and should not currently accept any orders.
- HighPriorityOrders: Intended for receiving all high-priority orders.
- GlobalOrders: Designed for orders not originating from the USA.
- HighOrders: Meant for receiving orders with a quantity greater than 1000.
- AllOrders: Created for auditing purposes, capturing all orders.

In configuring the subscription for "GlobalOrders" to receive orders not originating from the USA, what filter type should be implemented?

A. `SqlFilter`
B. `CorrelationFilter`
C. `No Filter`
D. `FalseFilter`

Answer: A

Feedback(if correct):

Implementing a `SqlFilter` with a condition like `Location <> 'USA'` allows the "GlobalOrders" subscription to receive messages for orders not originating from the USA. This enables filtering based on the specified SQL-like expression.

Feedback(if wrong):

`CorrelationFilter` is incorrect as it filters based on correlation values, not on the order location. `No Filter` is incorrect as it won't specifically filter out orders from the USA. `FalseFilter` is incorrect as it doesn't represent a valid filter type for this scenario.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

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

Competency: Configure and manage Azure Service Bus, Implement message-based communication, Develop for Azure Service Bus

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

42. A healthcare provider is implementing a real-time patient monitoring system that tracks vital signs from various devices across four separate wards. The system is designed to provide real-time analysis for each ward based on data gathered. The devices in each ward send their data to Azure Event Hubs. This data is then consumed by three different healthcare departments, each using an Azure Web App to visualize the data. You are responsible for configuring the Azure Event Hub instance for this system. Your implementation needs to maximize data throughput and minimize latency. How many partitions should you create in the Event Hub?

A. 2

B. 3

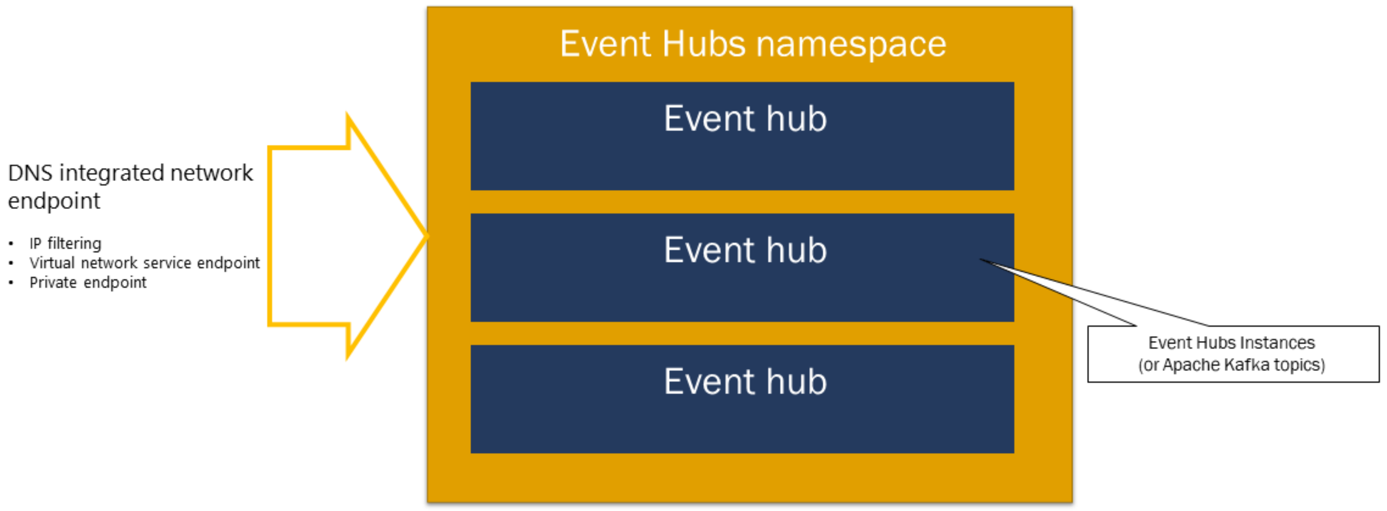
C. 4

D. 5

Answer: B

Feedback(if correct):

The correct answer is B. Azure Event Hubs uses partitions as a data organization mechanism that assists downstream parallelism in consuming applications. The number of partitions in an event hub is directly related to the number of concurrent readers you anticipate. In this case, since there are four wards and the data from each ward could potentially be read concurrently, having three partitions would be an optimal choice. This would provide a good balance between parallelism, data throughput, and latency reduction.


Event Hubs namespace
Event hub
Event hub
Event hub
DNS integrated network endpoint
• IP filtering
• Virtual network service endpoint
• Private endpoint
Event Hubs Instances (or Apache Kafka topics)

Feedback(if wrong):

Having two partitions could limit parallelism, and having four partitions would match the number of wards but may result in the underutilization of resources. Five partitions would exceed the number of wards, potentially leading to resource contention. Therefore, the optimal choice for this scenario would be three partitions (B), providing a good balance between parallelism, data throughput, and latency reduction.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

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

Competency: Configure and manage Azure Event Hubs

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

43. A publishing firm wants to leverage Azure Service Bus for distributing articles to several departments (e.g., editorial, marketing, etc.) via Publish-Subscribe (Pub/Sub) architecture. Currently, they face challenges delivering messages efficiently to all parties concerned. As a certified Azure developer, devise a solution for improving the delivery mechanism and mitigating latency during transmission. Given the scenario presented, suppose you notice that despite having properly configured the Azure Service Bus namespace, topic, and subscription, certain departmental clients struggle to obtain messages consistently. After investigating the issue, you suspect that slow polling intervals affect the timely reception of messages. How would you address this problem? Choose the best alternative from the given list:

A. Modify the OnMessageOptions parameter with an increased MaxConcurrentCalls value.

B. Apply a TimeSpan interval in the MessageReceiver constructor for ReceiveTimeout.

C. Call RenewLockAsync() periodically inside the registered message handler.

D. Increase the prefetchCount limit in the registration phase of the message handler.

Answer: A

Feedback (if Correct):

The correct answer is modifying the OnMessageOptions parameter with an increased MaxConcurrentCalls value. Boosting the maximum number of simultaneous requests processed by the message pump loop helps diminish latency caused by sluggish consumers. This decision demonstrates your aptitude for efficient Azure Solution development. Check out the accompanying diagram for a graphical representation of the implemented solution.

Feedback (if Wrong):

increasing the ReceiveTimeout duration isn't ideal for reducing latency, as it simply extends waiting periods for messages. Similarly, calling RenewLockAsync() repetitively introduces unnecessary overhead, adversely impacting performance. Lastly, raising the prefetchCount excessively strains system resources without tangible benefits in tackling latency woes. Revisit the question and consider the alternatives thoughtfully.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

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

Competency: Configure and manage Azure Service Bus

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

44. You are tasked with deploying a web backend for a new mobile game using Azure. Your goal is to ensure cost-efficiency, reliability, and scalability. How should you proceed with deploying the backend for maximum cost-effectiveness, reliability, and scalability?

A. Utilize the Free tier of Azure App Service to host the backend.

B. Deploy the backend on a virtual machine with automatic scaling enabled.

C. Opt for a dedicated App Service environment to host the backend.

D. Distribute the workload using Azure Container Instances.

Answer: B

Feedback (if correct):

Option B is correct because deploying the backend on a virtual machine with automatic scaling enabled allows for dynamic scaling based on traffic, ensuring reliability and scalability while minimizing costs.

Feedback (if wrong):

Option A is incorrect because utilizing the Free tier of Azure App Service may not provide sufficient scalability and reliability for a high-traffic mobile game backend. Option C is incorrect because opting for a dedicated App Service environment may be more costly than necessary and may not offer the same level of scalability. Option D is incorrect because Azure Container Instances may not provide the necessary scalability and reliability for hosting a mobile game backend.

Skill: Azure Developer Certification (AZ-204)

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

Competency: Configure authentication for APIs, Develop an App Service Logic App, Create a Logic App, Create a custom connector for Logic Apps, Define policies for APIs

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

45. Your corporation depends on Azure Content Delivery Network (CDN) for disseminating a static organizational logo image across assorted websites. Specify the appropriate sequence of movements associated with fulfilling a user's demand for the image by referring to the CDN URL.

A. Initiated by a user, the image request is submitted to the CDN URL. The DNS channels the request to the top-performing Point of Presence (PoP) terminal.

B. Lacking the presence of the image in the PoP's edge server cache, the POP turns to the origin server for the file acquisition.

C. The origin server releases the logo image to a PoP-residing edge server. That server then captures the logo image while relaying it back to the user.

D. Potential forthcoming appeals for the same file might be guided to the analogous PoP utilizing the CDN logo image URL. Supposing the Time-to-Live (TTL) hasn't transpired, the PoP's edge server pulls the file from the inventory and serves it.

Answer: A

Feedback (if correct):

True enough, this ordered listing matches the interactions happening when a user pursues a file from Azure CDN. Efficiently guiding the request to the best PoP, consecutive appeals capitalize on the same PoP to heighten performance.

Feedback (if wrong):

if no edge servers in the Point of Presence (POP) have the image in cache, the POP requests the file from the origin server, Subsequently, the origin server returns the logo image to an edge server in the POP. The accurate sequence is, where the POP edge server returns the image to the client. This correction ensures a precise representation of the Azure Content Delivery Network (CDN) workflow.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

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

Competency: Configure and use Azure Content Delivery Network (CDN)

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

46. You are developing a traffic monitoring system tracking vehicles on six highways. The data from traffic sensors gets saved to Azure Event Hub. Four departments, each with an Azure Web App and a WebJob for processing Event Hub data, rely on the traffic data. All Web Apps run on App Service Plans with three instances. Set up Azure Event Hub to maximize data throughput and minimize latency. Select the right combination of settings: Setting | # of Partitions | Partition Key

A. 6 | Department

B. 6 | Highway

C. 3 | Department

D. 3 | VM name

Answer: B

Feedback(if correct):

Configuring the number of partitions equal to the number of tracked highways (six) and setting the partition key to 'Highway' ensures that data relating to each highway is consolidated in a single partition, promoting organized processing and lowered latency.

Feedback(if wrong):

using "Department" as the Partition Key may not distribute data evenly across partitions, impacting throughput and latency. "VM name" is not suitable for efficient partitioning based on the scenario requirements. using "Timestamp" as the Partition Key may lead to uneven data distribution, affecting system performance.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

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

Competency: Configure and use Azure Event Hubs

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

47. You are developing Azure WebJobs and need to recommend a WebJob type for each scenario. Which WebJob type should you recommend?

Scenarios:

1. Run on all instances that the web app runs on. Optionally restrict the WebJob to a single instance.

2. Run on a single instance that Azure selects for load balancing. Supports remote debugging.

WebJob Type Options:

A. Triggered

B. Continuous

C. Singleton

D. Long Polling

Answer Choices:

A. Scenario 1: A, Scenario 2: D

B. Scenario 1: B, Scenario 2: A

C. Scenario 1: C, Scenario 2: B

D. Scenario 1: D, Scenario 2: C

Answer: C

Feedback (if correct):

Continuous WebJob is recommended for Scenario 1, where it can run on all instances or optionally on a single instance. Additionally, a Singleton WebJob is suitable for Scenario 2, running on a single instance selected by Azure for load balancing and supporting remote debugging.

Feedback (if wrong):

(Triggered WebJob): Incorrect, as it doesn't align with the requirements of running on all instances and optionally on a single instance.

(Continuous WebJob): Incorrect, as it does not match the specified requirements for Scenario 2.

(Long Polling WebJob): Incorrect, as it is not a recognized type of WebJob in Azure.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

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

Competency: Develop Azure WebJobs

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

48. You are developing a solution that involves an Azure SQL Database to store sensitive user information for a mobile app. To protect sensitive data from developers querying the database, which three components should you consider when implementing dynamic data masking? Choose the correct options.

A. Column

B. Table

C. Trigger

D. Schema

Answer: A, B , D.

Feedback(if correct): Dynamic data masking is applied at the column level to hide sensitive information from unauthorized users querying the database. Tables and schemas are not directly involved in dynamic data masking but can influence data visibility indirectly.

Feedback(if wrong):

Table (incorrect): While tables are essential components of a database, dynamic data masking is applied at the column level to hide sensitive information. Trigger (incorrect): Triggers are used to execute a set of SQL statements in response to specific events, but they are not directly related to dynamic data masking. Schema (incorrect): Schemas provide a way to organize database objects, but they are not directly involved in dynamic data masking.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

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

Competency: Implement dynamic data masking in Azure SQL Database

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

49. You are tasked with optimizing the Azure Functions application for a healthcare system. The system handles sensitive patient data and requires secure communication with external systems. Your goal is to ensure that the Azure Functions securely communicate with external APIs using certificates.

What should you do to achieve this goal?

A. Generate a self-signed certificate in Azure Key Vault.

B. Configure Azure Functions to use client certificates for outbound communication.

C. Implement Azure API Management to handle communication with external systems.

D. Use Azure Managed Identities to authenticate Azure Functions with external APIs.

Answer: B

Feedback(if correct):-

To ensure secure communication with external APIs using certificates, you should configure Azure Functions to use client certificates for outbound communication. This ensures that the communication is authenticated and encrypted, providing a secure channel for transmitting sensitive patient data.

Feedback(if wrong):-

Option A (Generate a self-signed certificate in Azure Key Vault) is incorrect because self-signed certificates may not provide the necessary level of trust and security required for communication with external systems.

Option C (Implement Azure API Management) is incorrect because while API Management can help with managing APIs, it does not directly address the requirement for secure communication using certificates.

Option D (Use Azure Managed Identities to authenticate Azure Functions with external APIs) is incorrect because managed identities are primarily used for authenticating with Azure services, not external APIs.

Skill Mapping:

Skill: Azure Developer Certification (AZ-204)

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

Competency: Configure authentication for APIs

Bloom's Taxonomy Level: Application

Difficulty Level: Intermediate

50. You are tasked with optimizing the Azure Functions application for a healthcare system. The system handles sensitive patient data and requires secure communication with external systems. Your goal is to ensure that the Azure Functions securely communicate with external APIs using certificates. What should you do to achieve this goal?

A. Develop an App Service Logic App to act as an intermediary for secure communication.

B. Create a Logic App to manage the interaction between Azure Functions and external APIs.

C. Create a custom connector for Logic Apps to establish secure communication with external APIs.

D. Use Azure Managed Identities to authenticate Azure Functions with external APIs.

Feedback(if correct):

Option B is correct because configuring Azure Functions to use client certificates for outbound communication ensures secure communication with external APIs, which is crucial for handling sensitive patient data in a healthcare system.

Feedback(if wrong):

Option A is incorrect because using a self-signed certificate may not provide sufficient security for communication with external systems, especially in a healthcare environment handling sensitive patient data.

Option C is incorrect because while Azure API Management is useful for managing APIs, it does not directly address the requirement of securely communicating with external systems using certificates.

Option D is incorrect because while Azure Managed Identities offer authentication for Azure resources, they do not directly handle client certificates for outbound communication with external APIs.

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